UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

TESIS

DISEÑO E IMPLEMENTACIÓN DE UN SOFTWARE PARA SISTEMAS SCADA, BASADO EN EL LENGUAJE PHP

PARA OBTENER EL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

ELABORADO POR:

CAMILO ADOLFO PORTELLA RETUERTO

ASESOR

M.Sc. ING. RUBEN DARÍO AQUIZE PALACIOS

LIMA – PERÚ

2020

“DISEÑO E IMPLEMENTACIÓN DE UN SOFTWARE PARA SISTEMAS SCADA, BASADO EN EL LENGUAJE PHP”

Con todo cariño, a mi madre.

Deseo agradecer a mi asesor y profesor, el Ing. Rubén Aquize Palacios, ya que gracias a su curso “Controles Eléctricos y Automatización” aprendí mucho de lo que conozco de la “Automatización de Procesos Industriales”.

SUMARIO

En la actualidad las empresas que desarrollan procesos industriales trabajan con sistemas SCADAs para monitorear y controlar sus procesos, en el mercado se pueden encontrar SCADAs de reconocidas marcas, si bien no se puede negar que estos sistemas SCADAs son bastante confiables, pero tienen el detalle que su implementación suele ser bastante costosa.

Otro de los inconvenientes que se encuentran en estos sistemas SCADAs es que ofrecen servicios que actúan únicamente sobre los datos de las RTUs, ejemplo de estos servicios son : interfaces gráficas, alarmas, históricos etc.

La solución que este proyecto de Tesis presenta en lo referente a lo económico, está basado en el uso hardware de bajo coste, y programas que no requieran licencia.

La solución que se presenta para que el sistema SCADA no limite su trabajo solamente a los datos de las RTUs, se basó en la construcción de un sistema cuya arquitectura está basada en el paradigma de la programación orientada a objetos (POO), lo cual permitirá incluso trabajar con elementos que no sean necesariamente electrónicos.

La importancia de este proyecto radica, en que va a ofrecer a las empresas que no cuentan con un presupuesto elevado como para implementar un SCADA convencional, la posibilidad de implementar un sistema que les permita realizar las tareas de monitoreo y control de sus procesos industriales a un bajo costo, usando para ello tecnología Web.

Otro punto importante es que el sistema de este proyecto al usar tecnología Web, hace que sea posible la protección de la información, ya que la tecnología Web ofrece herramientas que protegen al sistema de ataques informáticos, una de esas herramientas es HTTPS.

ABSTRACT

Currently, companies that develop industrial processes work with SCADAs systems to monitor and control their processes, on the marketplace you can find SCADAs of recognized brands, although it cannot be denied that these SCADAs systems are quite reliable, but they have the detail that Its implementation is usually too expensive.

Another drawback found in these SCADAs systems is that they offer services that only act on the data of the RTUs, such as graphic interfaces, alarms, history, etc.

The solution that this thesis project presents in terms of cost, is based on the use of low-cost hardware, and programs that do not require a license.

The solution presented so that the SCADA system does not limit its work only to the data of the RTUs, was based on the construction of a system whose architecture is based on the paradigm of object-oriented programming (OOP), which will allow even working with elements that are not necessarily electronic.

The importance of this project lies in that it will offer companies that do not have a high budget to implement a conventional SCADA, the possibility of implementing a system that allows them to carry out the tasks of monitoring and controlling their industrial processes through a low cost, using for it Web technology.

Another important point is that the system of this project when using Web technology, makes the protection of information possible, since Web technology offers tools that protect the system from computer attacks, one of those tools is HTTPS.

PRÓLOGO

En el presente trabajo de Tesis se ha realizado el diseño e implementación de un software para la construcción de un Sistema SCADA (Supervisory Control and Data Acquisition) basado en el lenguaje PHP (Hypertext Preprocessor) el cual sea compatible con RTUs que tienen implementados el protocolo de comunicación Modbus-RTU, para lo cual se usó el lenguaje de programación C. Este SCADA debe proporcionar el uso de pantallas y usuarios ilimitados con un costo muy bajo.

Para la implementación del SCADA con costos bajos se ha empleado software que no requieren licencia, tales como PHP, Apache y Pentaho. Por otra parte, para abaratar los costos del hardware se ha empleado un microcontrolador PIC y un microservidor-web SitePlayer.

El desarrollo del trabajo de Tesis se ha dividido en 6 capítulos, conclusiones y recomendaciones. En el Capítulo 3.1 se modela un sistema SCADA usando POO (Programación Orientada a Objetos).

En el Capítulo 3.2 se aborda las Clases sobre las cuales trabaja el sistema SCADA y sus respectivas instancias, esta idea de estructurar el sistema SCADA en función a Clases es la parte medular de este proyecto de Tesis.

En el Capítulo 3.3 se define la infraestructura encargada de proporcionar a las variables de campo sus valores en tiempo real, accediendo para ello a la memoria RAM de las RTUs.

En el Capítulo 3.4 se expone acerca de toda la infraestructura usada para que el operador de campo pueda interpretar y maniobrar la dinámica del proceso industrial.

En el Capítulo 3.5 se aborda la infraestructura necesaria para el almacenamiento de los valores de campo en Bases de Datos.

En el Capítulo 3.6 se exponen los reportes que se pueden generar a partir de los datos almacenados en la Base de Datos.

Para la construcción del SCADA mencionado se ha usado una PIC como Modbus Master y para la simulación de unidades remotas se ha utilizado una PC con el software Modbus Slave.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

LISTA DE ACRÓNIMOS

SCADA :	Supervisory Control and Data Acquisition
PHP :	Hypertext Preprocessor
RTU :	Remote Terminal Unit
PIC :	Programmable Integrated Circuited
WEB :	Red (vocablo ingles)
INFORMATICA:	Información Automática
PC :	Computadora Personal
SQL :	Structured Query Language
Modbus :	Modicon BUS.
POO :	Programación Orientada a Objetos.
RAM :	Random Access Memory
HTTP :	Hypertext Transfer Protocol
P&ID :	Piping and Instrumentation Diagram.
EEPROM :	Electrically Erasable Programmable Read-Only Memory

CAPÍTULO I
INTRODUCCIÓN

1.1 Generalidades.

Dado que en este proyecto de Tesis se va a realizar la implementación de un sistema SCADA, este mismo iniciará con la descripción general de diversas tareas que suelen estar presentes en estos sistemas.

Supervisión y Control*:* Se refiere a la capacidad que ofrece el sistema SCADA al operador de campo para **interpretar** y **maniobrar** un proceso industrial, proporcionando para ello animaciones gráficas, tendencias, alarmas etc.

La Fig.1.1 muestra 2 operadores haciendo uso de un sistema SCADA.

Fig. 1.1. Supervisión y Control [***Fuente****: www.reliance-scada.com*]

Adquisición de Datos*:* Se refiere a la extracción de datos que el sistema SCADA realiza mediante las RTUs, las cuales a su vez están conectadas a dispositivos de campo, como sensores y actuadores, los cuales se muestran en la Fig. 1.2.

Fig. 1.2. Adquisición de Datos [***Fuente****: Propia*]

Almacenamiento de Datos*:* Tarea mediante la cual se guardan los datos extraídos de las RTUs en bases de datos para su posterior análisis. En la Fig. 1.3. se muestran los software que se usan para el almacenamiento de datos.

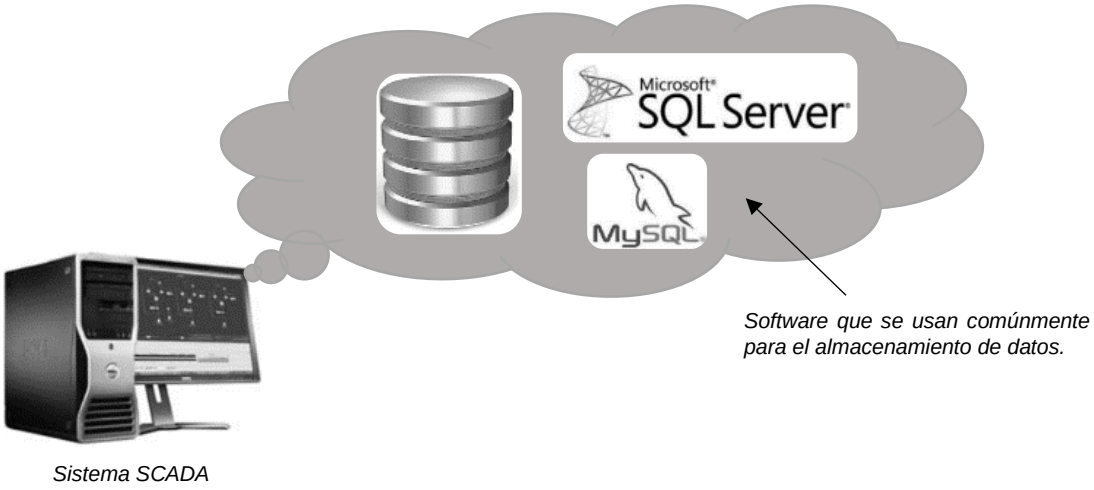


Fig. 1.3. Almacenamiento de Datos. [***Fuente****: Propia*]

Procesamiento de Datos Almacenados*:* Básicamente se refiere a los reportes que se generan a partir de los datos almacenados, estos reportes suelen ser interpretados por la gerencia y no por los operadores de campo. En la Fig. 1.4 se muestran algunos software que se usan para la generación de reportes.

Fig. 1.4. Procesamiento de datos almacenados en la Base de Datos. [***Fuente****: Propia*]

1.2 Descripción del problema de investigación

En este proyecto de Tesis describir la problemática es sinónimo de realizar un análisis sobre las falencias que existen en los sistemas SCADA que actualmente hay en el mercado industrial.

Si se hace una breve revisión a través de los sistemas SCADA más reconocidos del mercado, como lo son FactoryTalk, iFIX, WinCC etc., se puede observar que las herramientas que ofrecen, giran en torno a algo en común: *LAS DIRECCIONES DE MEMORIA RAM DE LAS RTUs.*

En estos SCADAs comerciales, la secuencia de trabajo casi siempre es la siguiente:

- Creación del tag y vinculación de este mismo con la *dirección de memoria* de la RTU correspondiente.
- Luego a este tag se le puede configurar alarmas, así como también se le puede configurar históricos etc.

En la Fig. 1.5 se muestra cómo las herramientas de los sistemas SCADA actúan sobre los tags, los cuales no son otra cosa más que las direcciones de memoria RAM de las RTUs.

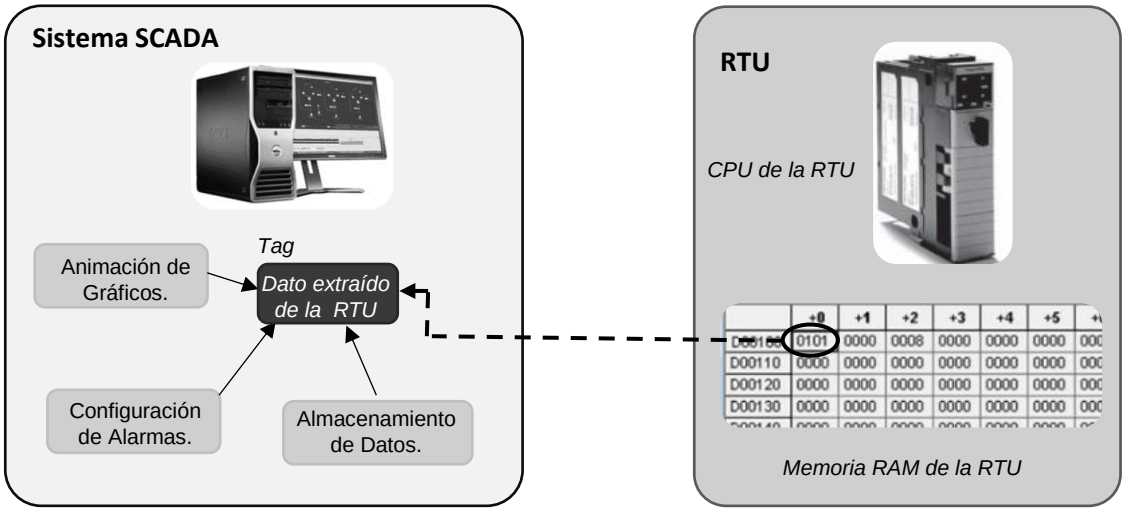


Fig. 1.5. Sistemas SCADA sobre direcciones de la memoria RAM de la RTU. [***Fuente**: Propia*]

Con lo explicado líneas arriba se concluye que las siguientes acciones no se pueden realizar sobre un sistema SCADA comercial:

- Guardar un permiso de trabajo.
- Almacenar el datasheets de algún dispositivo.
- Cargar un video que muestre la calibración de algún dispositivo.

El problema radica justamente en el modelado del sistema SCADA, el cual está en función de la memoria RAM de la RTU, tal como lo muestra la Fig. 1.6.

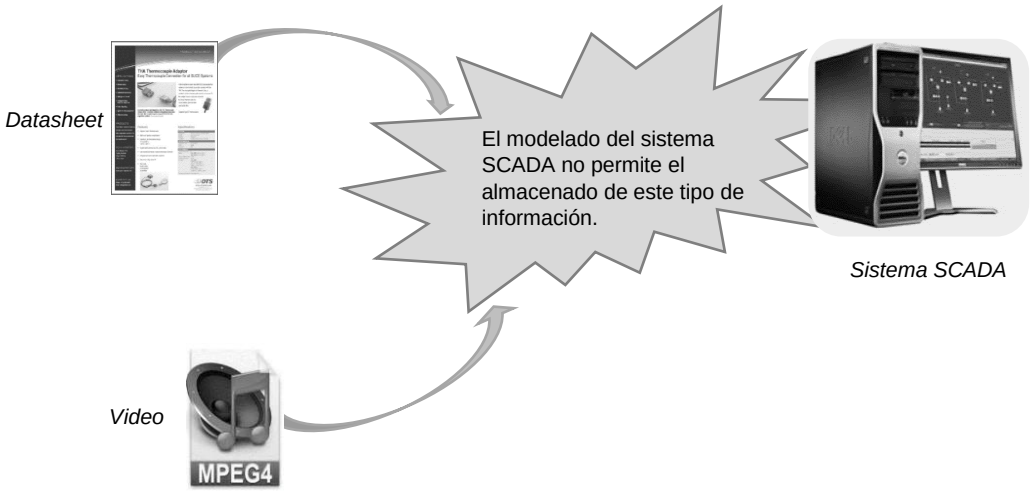


Fig. 1.6. Información que no es asimilada por la arquitectura SCADA.
[***Fuente**: Propia*]

1.2.1 Formulación del problema:

¿De qué manera es posible diseñar e implementar sistemas SCADA aplicando el lenguaje de programación PHP?

1.2.2 Justificación

Años atrás, era aceptable tener sistemas SCADA que estuvieran aislados de los sistemas informáticos, procesando solamente la información proveniente de las RTUs, pero con el desarrollo de la Internet, se hace cada vez más importante la integración de toda la información que pueda manejar una empresa en una plataforma Web, en ese sentido, este proyecto de Tesis plantea una mejora bastante interesante, ya que realiza casi todas sus tareas sobre la plataforma Web y permite el ingreso de información como imágenes y videos.

Es importante mencionar que el hardware usado en este proyecto de Tesis podría ser considerado no apto para la aplicación en el sector industrial debido a que es más de uso académico y no cumple con muchos de los estándares requeridos por la industria, pero este inconveniente no será complicado de superar, ya que la programación que hay en el microcontrolador PIC está en lenguaje C y por consiguiente, trasladar la lógica contenida en el PIC a otro procesador no resultará ser una tarea engorrosa, una vez superada esta limitación (lo cual tendría que realizarse en proyectos futuros), el impacto que podría tener este sistema en el mercado podría ser bastante positivo dado el costo relativamente bajo que tendría en comparación con los SCADAs convencionales.

1.3 Objetivos del estudio

Están conformados por objetivos Generales y objetivos Específicos.

1.3.1 Objetivo general

Diseñar e implementar un sistema SCADA usando el lenguaje PHP.

1.3.2 Objetivos específicos.

- Diseñar un sistema SCADA que se interconecte con RTUs y PLCs.
- Implementar un sistema SCADA usando el lenguaje PHP.
- Implementar una red Modbus RTU que trabaje con la norma RS-485.
- Implementar un algoritmo que permita que el microservidor-web SitePlayer establezca comunicación con el microcontrolador PIC.

1.3.3 Alcances.

- Este proyecto tiene como finalidad ser utilizado en sistemas industriales, sobre todo a procesos relacionados con el oil and gas y minería, haciendo la salvedad de que aún habría que cambiar el hardware usado a uno más apto para condiciones difíciles en cuanto a temperatura, polvo, etc.
- De momento, este SCADA solamente estará en capacidad de establecer comunicaciones con RTUs que tengan implementado el protocolo de comunicación Modbus-RTU, los cuales a su vez tienen que estar soportados sobre la norma RS-485, en futuros proyectos se podría agregar más protocolos.
- El sistema SCADA implementado tiene una arquitectura que le permite trabajar con diversos tipos de información que estén relacionadas de alguna manera al proceso industrial, pero de momento solo realizará tareas que le permitan igualar en gran medida las prestaciones de un SCADA comercial, estas prestaciones incluyen interfaces gráficas desarrolladas en HTML, sistema de alarmas, histórico de datos, escritura para el seteo de variables, etc.
- En cuanto a la distancia que puede haber entre los RTUs a ser monitoreados y el sistema SCADA implementado, esta puede ser ilimitada, ya que el SCADA trabaja con tecnología Web.

1.4 Antecedentes investigativos referidos al tema de investigación.

1.4.1 Programación Orientada a Objetos en IoT.

El proyecto de Tesis se basa principalmente en el paradigma de la Programación Orientada a Objetos (POO), si bien en lo que refiere a sistemas SCADA no es muy común el uso de este paradigma para el desarrollo de su arquitectura, ya que por lo general (por no decir siempre) los SCADAs trabajan sobre las direcciones de los registros de las RTUs (algo que tiene más similitud con la Programación Procedimental), pero sí es posible encontrar casos donde este paradigma(POO) es aplicado, como por ejemplo, la tecnología del Internet de las Cosas(IoT), una investigación bastante interesante de como este paradigma es aplicado en IoT es uno publicado por Boris Ulitin y Eduard Babkin, titulado "An Object-Oriented Model for Smart Devices in Internet of Things" del año 2018 [1], donde se crean clases para representar a los dispositivos inteligentes, tal como se muestra en la Fig. 1.7.

La abstracción se basa en modelar al dispositivo inteligente o bien como un sensor o como un actuador.

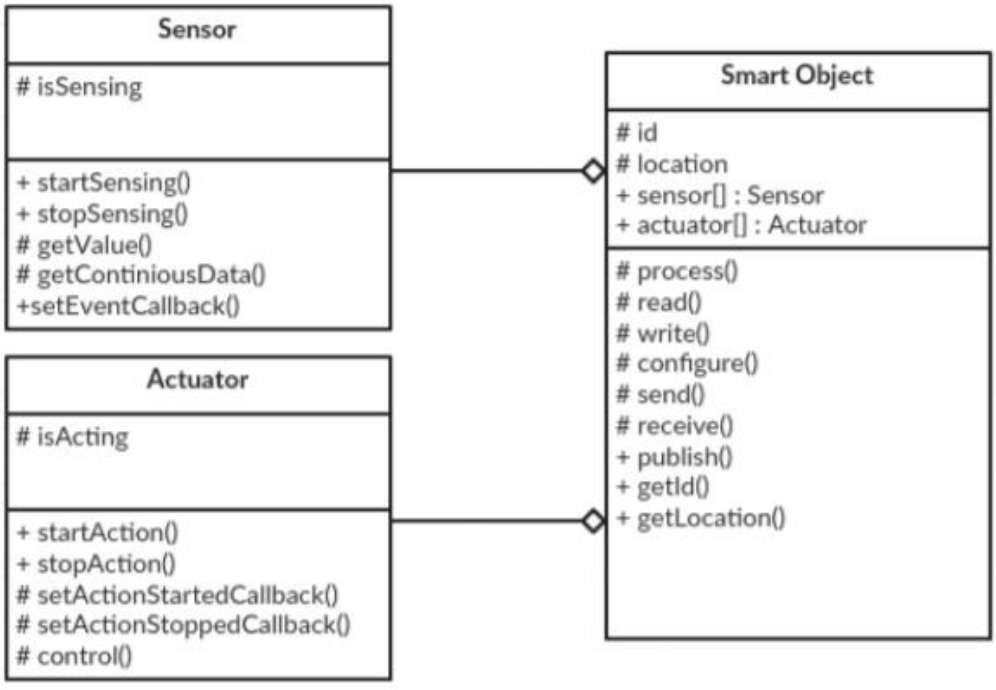
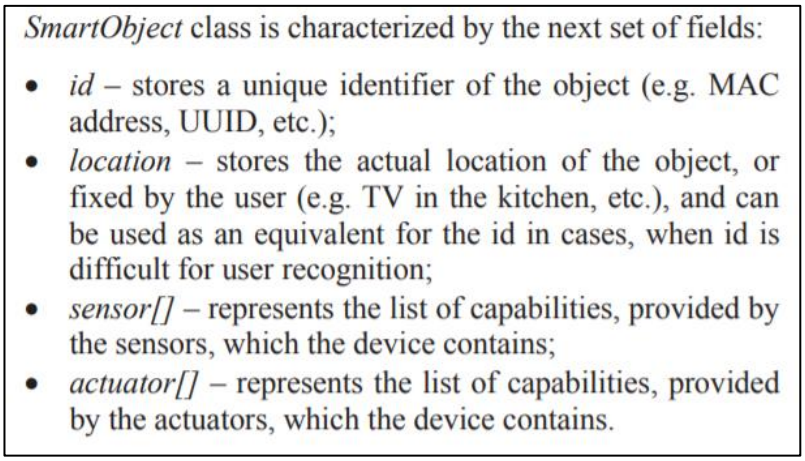


Fig. 3. UML-representation of the object-oriented device model

Fig. 1.7. Clase que representa a un dispositivo inteligente.
[***Fuente**: Ulitin, Boris & Babkin, Eduard. (2018). An Object-Oriented Model for Smart Devices in Internet of Things.*]

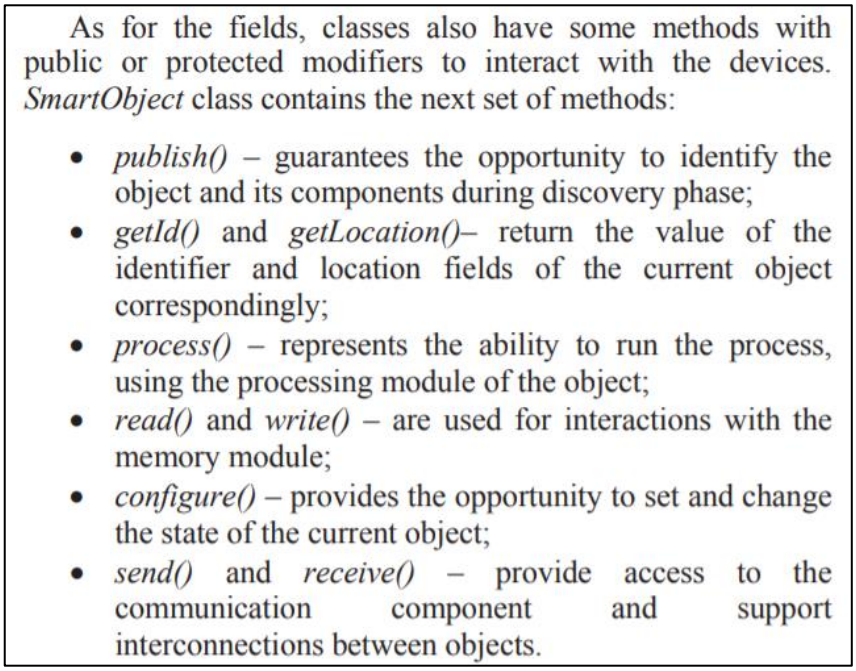
Las definiciones de estos atributos (en el artículo los llaman fields) y métodos se describen en las siguientes imágenes Fig. 1.8 y Fig. 1.9:



SmartObject class is characterized by the next set of fields:

- *id* – stores a unique identifier of the object (e.g. MAC address, UUID, etc.);
- *location* – stores the actual location of the object, or fixed by the user (e.g. TV in the kitchen, etc.), and can be used as an equivalent for the id in cases, when id is difficult for user recognition;
- *sensor[]* – represents the list of capabilities, provided by the sensors, which the device contains;
- *actuator[]* – represents the list of capabilities, provided by the actuators, which the device contains.

Fig. 1.8. Lista de atributos de la clase Smart Object.
[***Fuente**: Ulitin, Boris & Babkin, Eduard. (2018). An Object-Oriented Model for Smart Devices in Internet of Things.*]



As for the fields, classes also have some methods with public or protected modifiers to interact with the devices. *SmartObject* class contains the next set of methods:

- *publish()* – guarantees the opportunity to identify the object and its components during discovery phase;
- *getId()* and *getLocation()*– return the value of the identifier and location fields of the current object correspondingly;
- *process()* – represents the ability to run the process, using the processing module of the object;
- *read()* and *write()* – are used for interactions with the memory module;
- *configure()* – provides the opportunity to set and change the state of the current object;
- *send()* and *receive()* – provide access to the communication component and support interconnections between objects.

Fig. 1.9. Lista de métodos de la clase Smart Object.
[***Fuente**: Ulitin, Boris & Babkin, Eduard. (2018). An Object-Oriented Model for Smart Devices in Internet of Things.*]

Este enfoque resulta ser importante para los autores del artículo ya que les permite representar de forma abstracta a un dispositivo en general, algo que permite superar el problema de la variedad de protocolos, así como el de la variedad de plataformas, lo que suele limitar la integración de los dispositivos, aprovechando por sobre todo características

como la reutilización de una clase, así como también características propias de POO como la herencia, tal como se muestra en la Fig. 1.10.

Para este trabajo también se recurrió al concepto de DOM (Document Object Model) el cual permite la representación de los dispositivos y los recursos de estos de forma estructurada, es importante mencionar que el concepto de DOM fue un concepto que se usó para lograr una estructura jerárquica en el diseño de las paginas Web.

Por consiguiente, el artículo en mención fue de mucha relevancia para el diseño de la arquitectura del sistema SCADA desarrollado en este proyecto de Tesis ya que será la POO lo que permita superar la problemática descrita en la subsección 1.2.

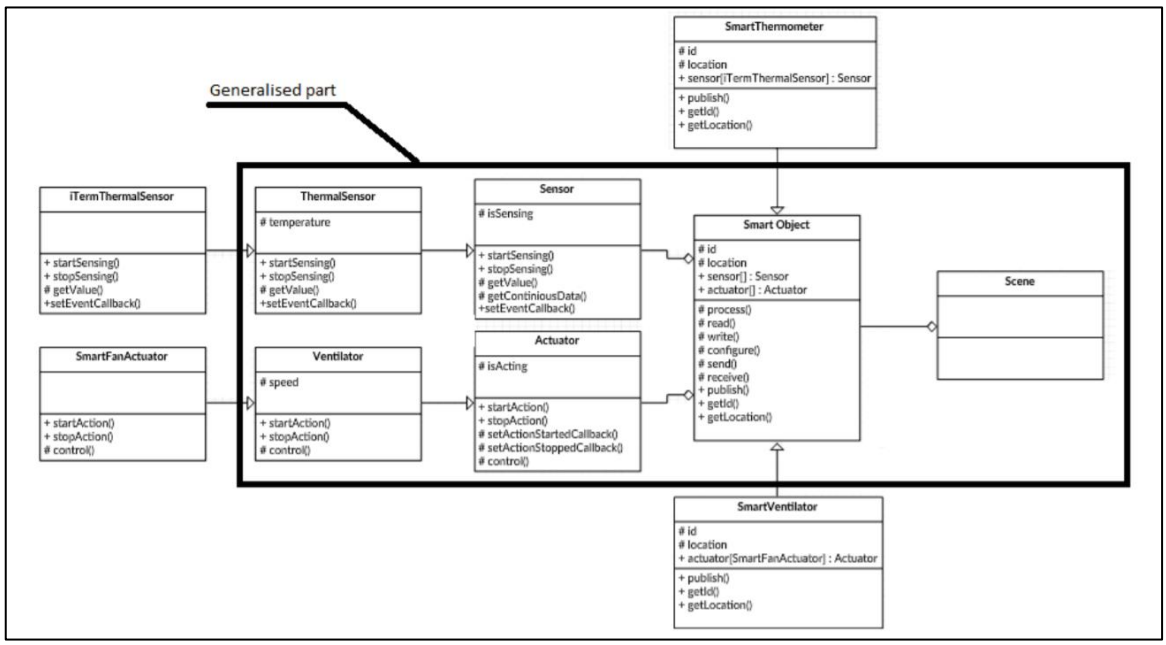


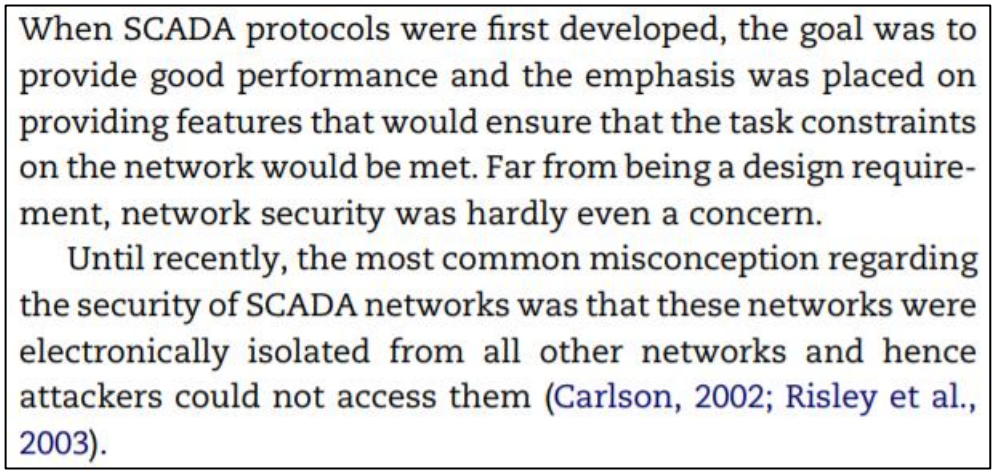
Fig. 1.10. Modelamiento de dispositivos basados en POO y herencia.
[***Fuente**: Ulitin, Boris & Babkin, Eduard. (2018). An Object-Oriented Model for Smart Devices in Internet of Things.*]

1.4.2 Seguridad de sistemas SCADAs.

Los sistemas de control y automatización y más precisamente los sistemas SCADAs, se caracterizan por ser aplicaciones muy vulnerables, especialmente frente a ataques informáticos como la captura de información, virus, etc.

El motivo por el cual los sistemas SCADAs presentan estas falencias es abordado en un artículo de investigación elaborado por V. M. Igure, S. A. Laughter y R. D. Williams titulado “Security issues in SCADA networks” [2], en donde menciona que los sistemas SCADAs tienen tales problemas de seguridad debido a que cuando fueron concebidos, el principal objetivo de los protocolos de los sistemas SCADA era garantizar un buen

rendimiento y que las tareas se cumplan sin inconvenientes, la seguridad era algo que se ubicaba en un segundo plano, en la Fig. 1.11 se da muestra de ello.



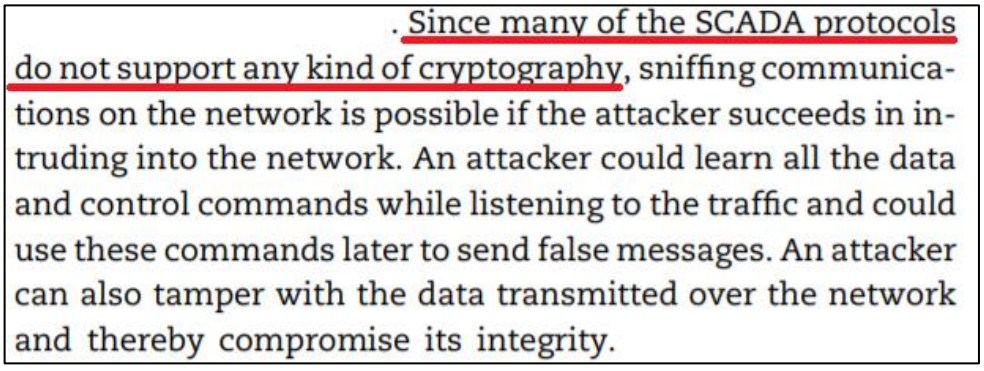
When SCADA protocols were first developed, the goal was to provide good performance and the emphasis was placed on providing features that would ensure that the task constraints on the network would be met. Far from being a design requirement, network security was hardly even a concern.

Until recently, the most common misconception regarding the security of SCADA networks was that these networks were electronically isolated from all other networks and hence attackers could not access them (Carlson, 2002; Risley et al., 2003).

Fig. 1.11. SCADA en sus inicios.
[***Fuente**: V. M. Igure, S. A. Laughter, and R. D. Williams. Security issues in SCADA networks*.]

Del articulo también se rescata algo muy importante que siempre se observa cuando se trabaja en plantas industriales, y es el hecho de creer que aislando físicamente el SCADA de las redes exteriores (del Internet, por ejemplo), el SCADA va estar protegido, nada más lejos de la realidad.

Otro punto que también es importante rescatar del artículo es el hecho de que los protocolos de los sistemas SCADA no tienen ningún tipo de encriptación, lo cual permite a un atacante poder “aprender” los comandos y de esta manera comprometer al sistema, prueba de ello se encuentra en la Fig. 1.12



. Since many of the SCADA protocols do not support any kind of cryptography, sniffing communications on the network is possible if the attacker succeeds in intruding into the network. An attacker could learn all the data and control commands while listening to the traffic and could use these commands later to send false messages. An attacker can also tamper with the data transmitted over the network and thereby compromise its integrity.

Fig. 1.12. Falencias de protocolos SCADAs.
[***Fuente**: V. M. Igure, S. A. Laughter, and R. D. Williams. Security issues in SCADA networks*.]

El citado artículo también hace referencia a los protocolos que por esos años (2006) eran los más utilizados y presentaban falencias en cuanto a seguridad, pero que sin embargo al día de hoy, son bastante utilizados y comercializados, en la Fig. 1.13 se muestran tales protocolos de comunicación.

En conclusión, este artículo también resultó de mucha utilidad por que le da respaldo a la decisión de implementar el sistema SCADA utilizando tecnología web, para ser más específicos, utilizando el protocolo HTTP, el cual es un protocolo que dispone de herramientas de protección de la información, que lo hace más seguro que los protocolos que hay en los sistemas SCADA convencionales.

Table 1 – SCADA protocols

	Protocol	Organization/standard	Main features
1	Ethernet/IP (Industrial Protocol)	Open DeviceNet Vendors Association (ODVA) (www.odva.org)	Object-oriented, protocol; provides interoperability over Ethernet and fieldbus networks
2	DeviceNet	Open DeviceNet Vendors Association (ODVA) (www.odva.org)	Belongs to the CIP (Control and Information Protocol) family; CAN protocol defines layers 1 & 2; the rest are defined by DeviceNet and CIP
3	ControlNet	ControlNet International (www.controlnet.org)	Belongs to the same CIP (Control and Information Protocol) family; new physical layer with higher speed, strict determinism and repeatability with greater range
4	PROFIBUS	Type 3 protocol of IEC Standard 11674 and 61158 (www.profibus.org)	3-layer OSI model; has extensions for safety features; ProfiNet version provides Ethernet compatibility
5	MODBUS TCP/IP	MODBUS-IDA (www.modbus.org)	Encapsulates fieldbus packets over TCP; attempting to become an IETF standard
6	DNP3	(IEC) Technical Committee 57, Working Group 03 standard	It is also a based on the 3-layer OSI model
7	Foundation Fieldbus	The Fieldbus Foundation/open standard protocol (www.fieldbus.org)	Incorporates many safety features that make it a good candidate for mission-critical applications

Fig. 1.13. Lista de protocolos SCADAs.
[***Fuente**: V. M. Igure, S. A. Laughter, and R. D. Williams. Security issues in SCADA networks.*]

CAPÍTULO II
MARCO TEÓRICO

2.1 Marco Teórico

2.1.1 Evolución del sistema SCADA

En la página web de la empresa norteamericana Inductive Automation [3] se escribió un artículo acerca de la evolución de los sistema SCADA, y se menciona que, antes que se introdujeran los sistemas SCADAs a mediados del siglo XX las plantas industriales eran controladas manualmente por el personal a través de botones y diales analógicos, conforme fueron creciendo las plantas se comenzó a utilizar relays y temporizadores ,los cuales proporcionaban un manejo más automatizado, pero tenían la desventaja de ocupar demasiado espacio, aparte de ser un sistema sobre el cual era muy complicado realizar algún tipo de diagnóstico para la corrección de errores.

Se menciona también que, a principios de los años 50, se comenzaron a utilizar las computadoras con fines de control industrial, en la década de los años 60 se introdujo la telemetría para la transmisión de datos desde lugares lejanos, fue a principios de los años 70 que se acuño el termino SCADA, y fue en esa misma década que estuvo en auge el uso de los microprocesadores y PLCs en los sistemas de supervisión industrial.

En un primer momento los SCADA trabajaban sobre mainframes, estos no se conectaban en redes, recién por los años 80s y 90s los PLCs se empezaron a conectar en redes y los HMI empezaron a basarse en PC, un problema que hubo con los protocolos LAN usados por estos PLCs y SCADAs fue que eran propietarios y no podían sostener comunicación con ningún sistema de otra marca.

Entre los años 90 y comienzos del 2000 los sistemas SCADAs y PLCs comenzaron a adoptar la norma Ethernet lo cual, si bien no permitió que los sistemas de distintos vendedores pudiesen comunicarse, al menos podían usar la misma red física.

Hubo un desfasaje en cuanto al formato que usaban los sistemas SCADA para almacenar sus datos históricos, ya que se reusaron a adoptar el lenguaje SQL cuando para los sistemas de IT este era el standard, las distintas marcas de SCADAs mantuvieron sus formatos propietarios.

Actualmente muchos de los sistemas SCADAs siguen realizando mejoras basándose principalmente en las bondades de la tecnología informática.

2.1.2 La norma Ethernet en la Automatización

En la subsección 2.1.1 se pudo apreciar a grandes rasgos la evolución de los sistemas SCADA, evolución que significó entre otras cosas, la asimilación de las tecnologías de la informática, como por ejemplo la incorporación de la norma Ethernet (IEEE 802.3) como standard para la capa física en las redes de control.

Sin embargo, Ethernet es una norma que no cumplía con ciertos requerimientos solicitados en determinados procesos industriales, eso es algo que es muy difícil de creer teniendo en cuenta que Ethernet es un protocolo que trabaja con velocidades muy altas, en un artículo publicado por la empresa norteamericana Polytron titulada “Is Ethernet Deterministic? Does it matter?” [4] se describe cuáles eran esas falencias:

a) Las colisiones: Si bien en la actualidad este problema está solucionado con el uso de las tecnologías conmutadas (en otras palabras, el uso de switches), ya que las colisiones no existen debido a que el medio por donde transitan los datos solo es usado exclusivamente para un dispositivo. Pero hay que tener en cuenta que Ethernet en sus inicios compartía el medio por donde transitaba la data, lo cual hacia posible la existencia de colisiones lo cual traía como consecuencia no poder fijar el tiempo de llegada de la información al dispositivo receptor, ya que en caso las colisiones se produjeran, la comunicación sufriría una ralentización.

b) Falta de un sistema de priorización: Si bien el problema de las colisiones ya fue superado con el uso de las tecnologías de conmutación, un problema que aún persiste es la falta de priorización, veamos, si un switch recibe una gran cantidad de información proveniente de distintos puertos para ser entregada por un mismo puerto, la información no se va a perder debido a la existencia de un buffer que le permite al switch guardar la información, pero Ethernet no tiene implementada alguna tecnología que le permita priorizar que información tendría que ser conmutada primero.

   Para dar solución a tales problemas los protocolos usados en aplicaciones industriales, como Ethernet/IP, han implementado la norma IEEE1588, la cual les permite sincronizar los relojes a través de la red.

Es importante tener en consideración que los procesos industriales que requieren el levantamiento de las falencias, mencionadas líneas arriba, son muy pocas. Las aplicaciones con servomotores es uno de esos casos, ya que dada la naturaleza de sus lazos es necesario saber cuándo una respuesta va a ser recibida.

2.2 Marco conceptual

El proyecto de Tesis abarca 3 conceptos principalmente, la programación orientada a objetos (POO), la tecnología Web y el protocolo Modbus-RTU, las 2 primeras son tecnologías usadas principalmente en la informática, y no son muy difundidas aún en el rubro industrial, y el último se trata de una tecnología usada exclusivamente en el rubro industrial.

Por consiguiente, en esta sección se describirán tecnologías que de una u otra forma están relacionados con los 3 conceptos mencionados líneas arriba.

2.2.1 Programación Orientada a Objetos

En un artículo publicado el 2012 por la empresa Northware, Jesús Alberto Zamarripa [5], explica de forma bastante didáctica el significado de algunos conceptos fundamentales de la programación orientada a objetos, es de este artículo que se extraen las definiciones.

a) Clase: Al igual que un diseñador crea prototipos de dispositivos que podrían utilizarse en repetidas ocasiones para construir los dispositivos reales, una clase es un prototipo de software que puede utilizarse para instanciar (es decir crear) muchos objetos iguales.

b) Atributos y Métodos**:** Los atributos pueden definirse como el conjunto de elementos que definen a una Clase. Y los métodos son como comportamientos o funciones que maneja una clase.

En la Fig.2.1 se muestra cómo se define en Java una Clase, sus atributos y métodos.

```
public class Perro
{
      private int Peso;
      public int GetPeso()
      {
            return Peso;
      }
      public void SetPeso(int newPeso)
      {
            if (newPeso > 0)
            {
                  Peso = newPeso;
            }
      }
}
```

Fig. 2.1. Clases, Atributos y Métodos.
[***Fuente**: https://www.northware.mx/wp-content/uploads/2013/04/Programacion-orientada-a-objetos.pdf.*]

c) Constructor de una Clase**:** Es un miembro que implementa las acciones requeridas para inicializar la instancia de una clase. El constructor es parecido a los métodos, pero no tiene un tipo de retorno y su nombre es el mismo que el de la clase y también puede o no recibir parámetro como los métodos.

En la Fig. 2.2 se muestra la función constructor para una clase de nombre Persona.

```
public class Persona
{
      public Persona() {}
      {
      this.Nombres = string.Empty;
      this.Apellidoss = string.Empty;
      this.Edad = 0;
      }

      #region Properties
      private string Nombres { get; set; }
      private string Apellidos { get; set; }
      private int Edad { get; set; }
      private string NombreCompleto [...]
      #endregion

}
```

Fig. 2.2. Constructores de una Clase.
[***Fuente**: https://www.northware.mx/wp-content/uploads/2013/04/Programacion-orientada-a-objetos.pdf.*]

d) Herencia: Se da cuando una clase hereda sus propiedades y métodos a otra clase, esta segunda clase se convierte en la subclase y la primera en la clase superior o clase padre. Es decir, una clase base o clase padre le hereda sus propiedades a la clase hija o subclase.

En la Fig. 2.3, se muestra un ejemplo de una clase llamada Persona y otra llamada Empleado, esta última hereda los atributos y métodos de la clase Persona.

e) Encapsulación: Es un mecanismo que permite a los diseñadores de datos determinar que miembros de la clase pueden ser utilizados por otros programadores y cuáles no. Las principales ventajas son: el ocultamiento de detalles en lo referente a la implementación interna de un tipo de dato al cliente y la modificación del tipo de datos, por parte del programador, sin que ello implique afectar la filosofía de programación del cliente.

```
public class Persona
{
     public Persona()[...]
     public Persona (string N, string A, int E)[...]

     #region Properties
     private string Nombres { get; set; }
     private string Apellidos { get; set; }
     private int Edad { get; set; }
     private string NombreCompleto[...]
     #endregion

     #region Methods
     private string GetFullName()[...]
     private int GetAge()[...]
     private void SetName(string N)[...]
     private void SetLastName(string A)[...]
     private void SetAge(int E)[...]
     #endregion

}

public class Empleado:Persona
{
     [constructor]

     private long Nomina { get; set; }
     private string Imss{ get; set; }

     public long GetNomina[...]
     public string GetImss[...]
}
```

Fig. 2.3. Herencia de una Clase.
[***Fuente**: https://www.northware.mx/wp-content/uploads/2013/04/Programacion-orientada-a-objetos.pdf.*]

2.2.2 La tecnología Web

Es sin lugar a dudas la más usada en la informática, y existe una extensa cantidad de términos y definiciones asociados con esta tecnología, de la página CISCO NETWORK ACADEMY (CCNA R&S: Introducción to Networks) [6] se extraerán las definiciones que se dictan a continuación:

a) Capa de Aplicación: La capa de aplicación es la más cercana al usuario final, los protocolos de la capa de aplicación se utilizan para intercambiar datos entre los programas que se ejecutan en los hosts de origen y destino. Existen muchos protocolos de la capa de aplicación, y están en constante desarrollo, algunos de los más conocidos son el protocolo de transferencia de hipertexto (HTTP), el protocolo de transferencia de archivos (FTP) y el sistema de nombres de dominios (DNS) tal como se muestra en la Fig. 2.4.

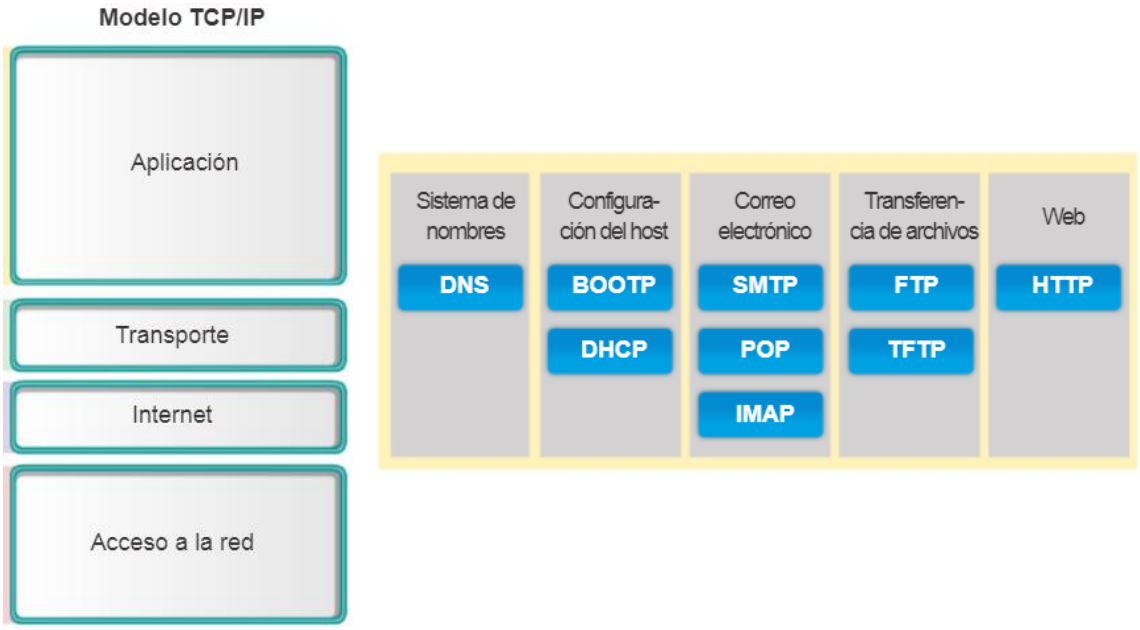


Fig. 2.4. A la izquierda, las capas del modelo TCP/IP y a la derecha los protocolos más conocidos de la capa de Aplicación del modelo TCP/IP.
[***Fuente**: CCNA R&S: Introducción to Networks, Capitulo 10: Capa de Aplicación, 10.1.1.4 Protocolos de la Capa de Aplicación]*

b) Protocolo HTTP: Originalmente, este protocolo se desarrolló solo para publicar y recuperar paginas HTML. Sin embargo, la flexibilidad de HTTP lo convirtió en una aplicación fundamental de los sistemas de información distribuidos y cooperativos.

HTTP es un protocolo de solicitud/respuesta. Cuando un cliente, por lo general un explorador Web envía una solicitud a un servidor Web, HTTP especifica los tipos de mensaje que se utilizan para esa comunicación. Los 3 tipos de mensajes comunes son GET, POST y PUT.

GET es una solicitud de datos por parte del cliente. Un cliente (explorador Web) envía el mensaje GET al servidor Web para solicitar las paginas HTML, tal como se muestra en la Fig. 2.5, cuando el servidor recibe la solicitud GET, este responde con una línea de estado y un mensaje propio, el mensaje puede incluir el archivo HTML solicitado o puede contener un mensaje de error.

POST se utiliza generalmente para enviar datos al servidor Web provenientes de formularios que estuvieron integrados en el navegador Web.

El conocimiento de cómo trabaja el comando GET es de suma importancia para la extracción de datos del Gateway que se ha construido en este proyecto de Tesis, ya que para acceder al Gateway no será posible hacerlo a través del navegador Web.

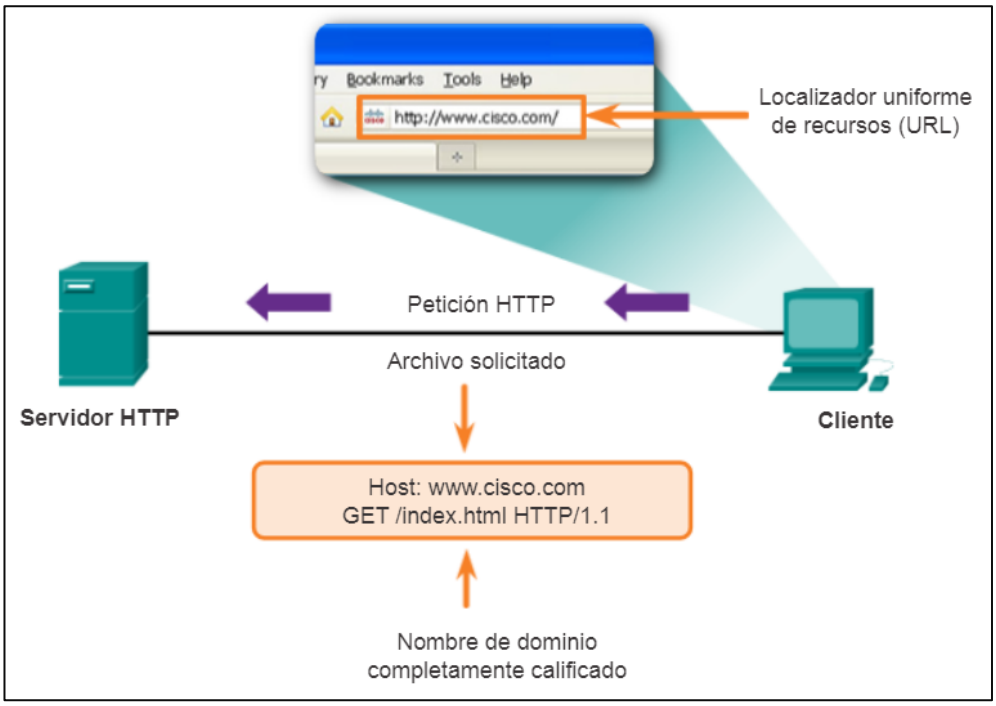


Fig. 2.5. Solicitud de un archivo mediante GET.
[***Fuente**:* CCNA R&S: Introducción to Networks*,* Capitulo 10: Capa de Aplicación, 10.2.1.3 HTTP y HTTPS]

c) HTTPS: Aunque HTTP es sumamente flexible, no es un protocolo seguro. Los mensajes de solicitud envían información al servidor en un texto sin formato que puede ser interceptado y leído, lo mismo sucede con las respuestas del servidor. Para una comunicación segura a través de Internet, se utiliza HTTPS, este puede utilizar autenticación y encriptación para asegurar los datos mientras viajan entre el cliente y servidor, HTTPS especifica reglas adicionales para pasar datos entre la capa de aplicación y la capa de transporte, los cuales son encriptados con capa de sockets seguros (SSL) antes de transportarse a través de la red.

SSL, una vez que fue un standard, su nombre fue cambiado a TSL, aunque hoy en día muchos suelen usarlo de forma indistinta.

En un libro de ILYA GRIGORIK cuyo nombre es “High Performance Browser Networking” [7] en el capítulo 4 “Transport Layer Security (TLS)” se muestra una gráfica muy precisa en donde se aprecia, entre que capas del modelo TCP/IP (del cual se estuvo hablando líneas arriba) se realiza la actividad de TLS (o SSL), esa grafica se plasma en la Fig. 2.6.

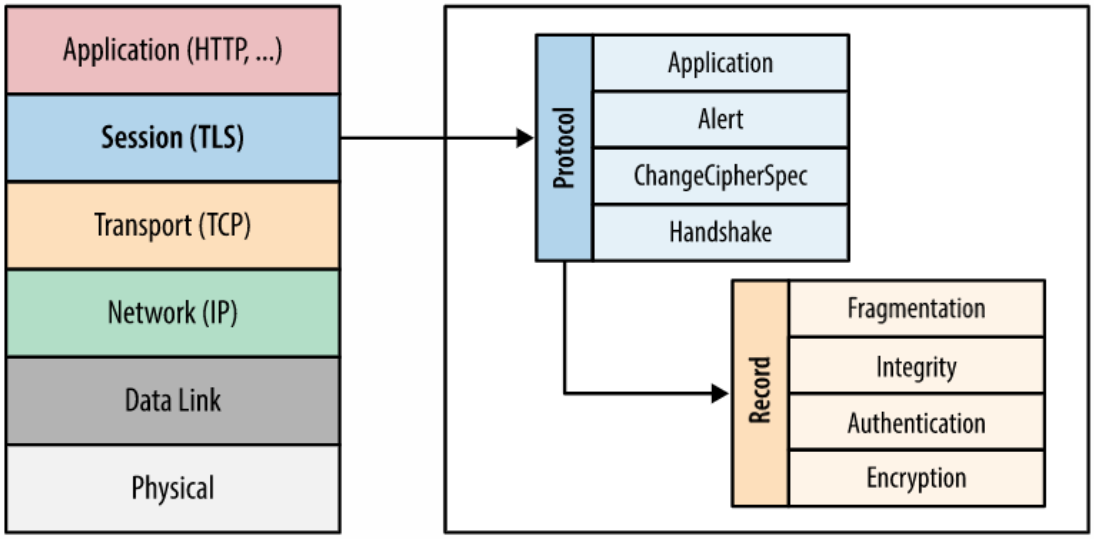


Fig. 2.6. Ubicación de SSL en la arquitectura de capas TCP/IP.
[***Fuente**: https://hpbn.co/transport-layer-security-tls/#encryption-authentication-and-integrity*]

2.2.3 El protocolo Modbus-RTU

Si bien el SCADA a implementar solo estará en capacidad de trabajar con RTUs que incluyan el protocolo Modbus-RTU, ello no debería llevar a pensar que el SCADA que se va desarrollar trabajará solamente con un número limitado de dispositivos, todo lo contrario, Gordon Clarke y Deon Reynders en su libro “Modern SCADA Protocols” (2004) en resumidas cuentas mencionan lo siguiente sobre el protocolo Modbus-RTU [8].

Aunque Modbus es relativamente lento en comparación con otros buses, tiene la ventaja de una amplia aceptación entre los fabricantes y usuarios de instrumentos. Alrededor de 20 a 30 fabricantes producen equipos con el protocolo Modbus, y muchos de estos sistemas están en la industria. Por lo tanto, puede considerarse como un estándar industrial de facto con probadas capacidades. Una encuesta reciente en la conocida revista norteamericana Control Engineering indicó que más del 40% de las aplicaciones de comunicación industrial utilizan el protocolo Modbus para la interfaz.

Si bien el libro mencionado fue publicado en el 2004, al día de hoy se puede decir que Modbus sigue siendo de lejos el protocolo más usado en las RTUs, mas no aún en los PLCs, los cuales siguen trabajando con protocolos cerrados, los cuales son propiedad de los fabricantes.

En la Fig. 2.7 se muestra una lista de todas las empresas que implementan el protocolo Modbus-RTU en sus dispositivos, la imagen fue extraída de la página de Modbus Organization, la cual es una organización que busca impulsar la adopción del protocolo Modbus.

Fig. 2.7. Empresas que incorporan el protocolo Modbus en sus dispositivos.
[***Fuente**: http://modbus.org/about.php*]

a) Capas en el protocolo Modbus-RTU: Del documento publicado por Modbus Organization, cuyo nombre es "MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3" (2012) [9], se extrajo la siguiente información referente a las capas del modelo OSI con las que trabaja el protocolo Modbus-RTU, aunque es importante mencionar que Modbus-RTU no está basado en el modelo OSI, pero ayudará a entender el protocolo.

"Modbus es un protocolo de mensajes de la capa de Aplicación, posicionado en la capa 7 del modelo OSI, el cual proporciona una comunicación Cliente/Servidor entre los dispositivos que integran la red".

En la Fig. 2.8 se muestra las capas que usa el protocolo Modbus-RTU y la Fig. 2.9 muestra cómo es que el protocolo Modbus-RTU lleva a cabo la comunicación Cliente/Servidor.

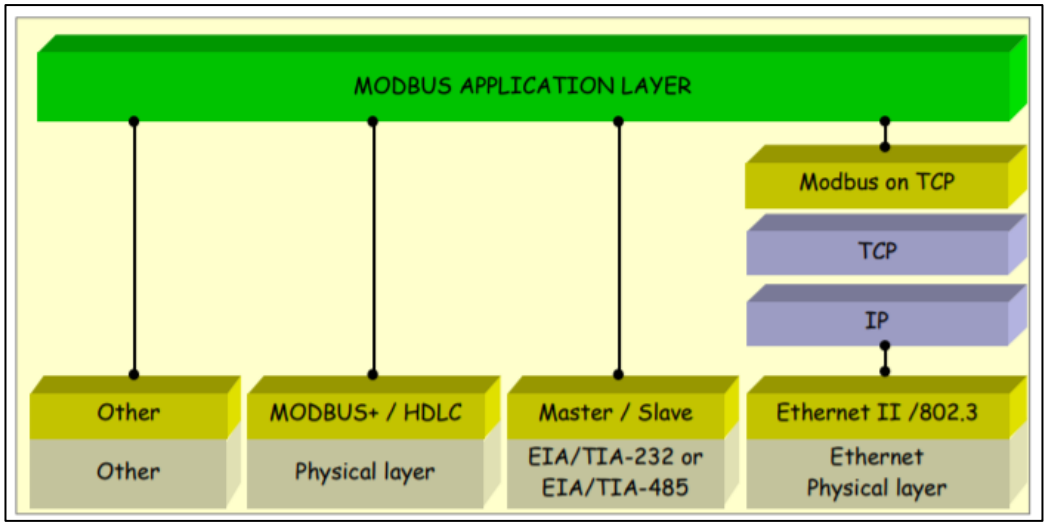


Fig. 2.8. Capas del modelo OSI con las que trabaja el protocolo Modbus-RTU (tercera columna empezando desde la izquierda).
[***Fuente**: http://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf*]

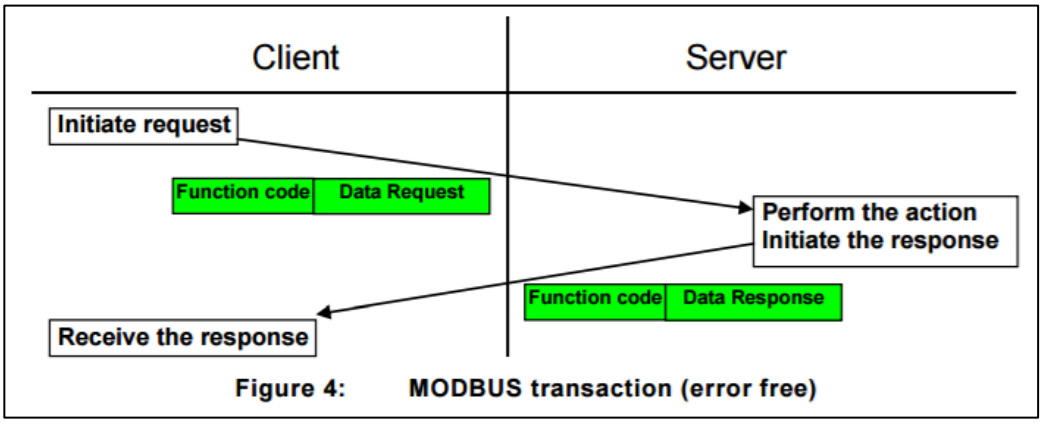


Fig. 2.9. Transacción de mensajes entre un cliente y servidor Modbus-RTU
[***Fuente**: http://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf*]

b) Norma RS-485(EIA/TIA-485): Si se observa la Fig. 2.8, en lo que corresponde a la capa física del protocolo Modbus-RTU se encuentra la norma RS-485, la cual es oficialmente conocida como TIA/EIA-485-A.

RS-485 especifica determinadas características asociadas a la capa física, como por ejemplo la longitud máxima del cable, la disposición de los nodos, resistencias en las terminales de la red, velocidad de transmisión etc.

Muchos protocolos usan esta norma como su capa física, como por ejemplo Modbus-RTU, DH-485, Profibus etc.

Es importante mencionar que si 2 protocolos usan RS-485 como capa física, esto no significa de ninguna manera que estos protocolos puedan sostener una comunicación.

RS-485 se caracteriza por usar un sistema del tipo balanceado, lo cual significa que ninguno de los cables que transporta la información está conectado a tierra.

En la Fig. 2.10 se muestra un típico cableado RS-485, el cual usa cable trenzado para evitar las interferencias, como se puede observar, ningunos de los cables que transportan información está conectado a tierra, esta información fue extraída de la página de Maxin Integrated (Design > Technical Documents > RS-485 Cable Specification Guide) [10].

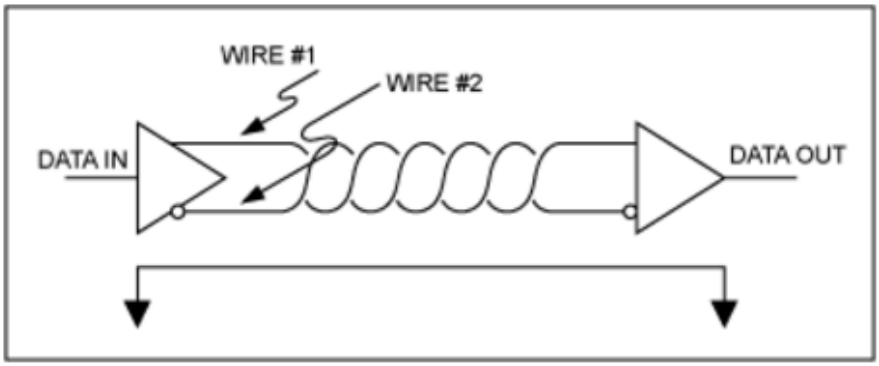


Fig. 2.10. La norma RS-485 recomienda usar cable trenzado de calibre 24 AWG. [***Fuente***:*https://www.maximintegrated.com/en/design/technicaldocuments/tutorials/7/763.html*]

En Fig. 2.11 muestra el voltaje de cada uno de los cables de la imagen anterior (Wire #1 y Wire #2) respecto de tierra, y podemos ver que estos son opuestos, lo cual ayuda a reducción de interferencias.

La reducción de interferencias es muy importante cuando se transmiten a altas tasas de velocidad y largas distancias.

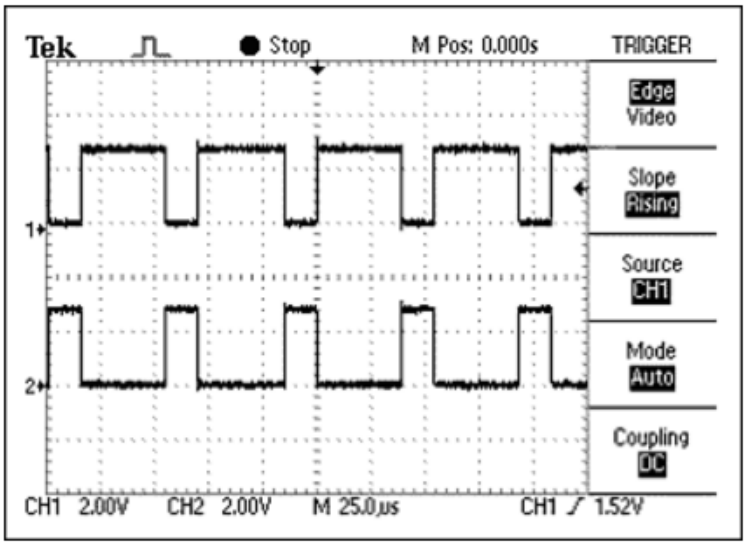


Fig. 2.11. La norma RS-485 trabaja con un sistema balanceado en el cual se transmiten voltaje opuestos.
[***Fuente**:https://www.maximintegrated.com/en/design/technicaldocuments/tutorials/7/763.html#*]

CAPÍTULO III
DESARROLLO DEL TRABAJO DE INVESTIGACIÓN

3.1 Modelamiento de un sistema SCADA usando POO

Al igual que en la POO se tiene que crear lo que son las clases, y para este proyecto las clases son las siguientes:

- Planta.
- Proceso.
- Variable.
- Usuario.

3.1.1 Clase Planta y sus instancias

Esta clase sería la representación de lo que vendría a ser la locación de una Planta Industrial. En la Fig. 3.1 se muestra cómo sería la INSTANCIACIÓN de las Plantas para una empresa determinada. Los nombres de las plantas son solo de ejemplo.

3.1.2 Atributos de la clase Planta

Algunos de los atributos de la clase PLANTA, y que se podrían ser más, serían los que se muestran en la Fig. 3.2.

Algo que es muy importante notar es que uno de los atributos es de tipo binario como lo sería la Foto de una Planta.

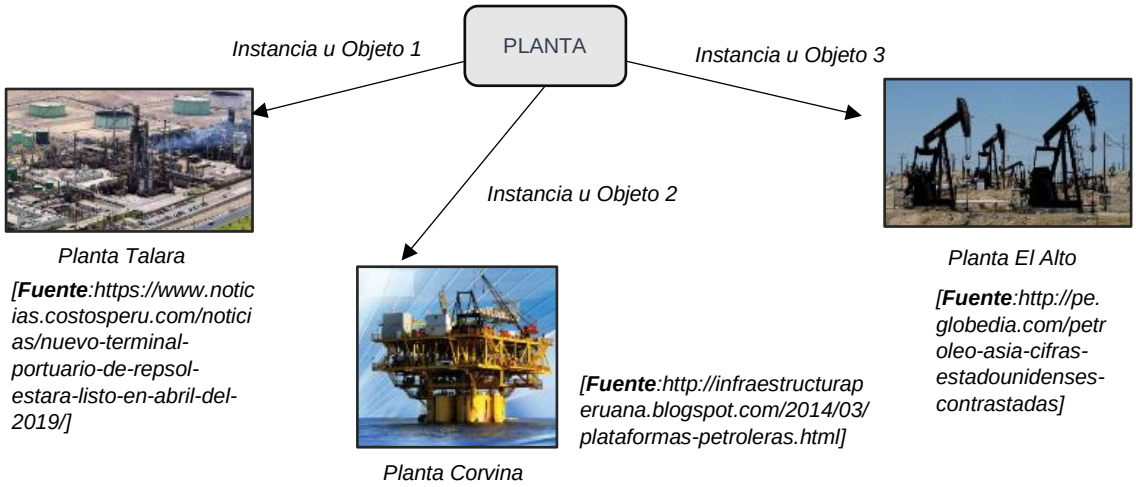


Fig. 3.1. Instanciación de la clase Planta.
[***Fuente**: Propia*]

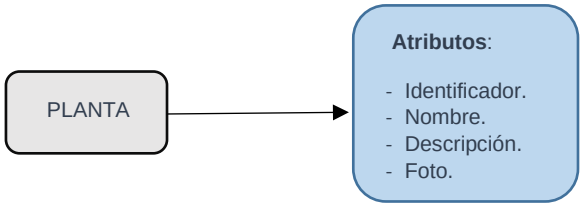


Fig. 3.2. Atributos de la clase Planta.
[***Fuente**: Propia*]

3.1.3 Clase Proceso y sus instancias

Esta clase es la representación de un conjunto de dispositivos (electrónicos, mecánicos, neumáticos etc.) que unidos realizan una tarea específica. En la Fig. 3.3 se muestra cómo es la INSTANCIACION de PROCESOS (Los nombres de los procesos son solo de ejemplo).

Cuando la empresa que solicita el servicio SCADA es una empresa grande, la instanciación de la clase PROCESO puede ser sencilla ya que los P&ID que disponen fueron elaborados en función a los procesos que existen en una planta, pero si la empresa es pequeña se puede tener problemas, ya que se tendría que desarrollar criterios para determinar dónde termina un proceso y dónde comienza otro, porque puede darse el caso que no se disponga de los P&ID.

3.1.4 Atributos de la clase Proceso

Algunos de los atributos que tiene la clase PROCESO son los que se muestran en la Fig. 3.4.

3.1.5 Clase Variable y sus instancias

Esta clase es la representación de un **parámetro físico** o la representación del **estado físico** de un dispositivo. En la Fig. 3.5 se muestra cómo es la INSTANCIACIÓN de VARIABLES. Los nombres de las variables son solo de ejemplo.

La instanciación de la clase VARIABLE es relativamente sencilla, porque se pueden obtener de los P&ID, y si se presentara el caso que la empresa no disponga de los P&ID, lo que se tendría que hacer es elaborar una lista con todos los parámetros físicos y estados físicos que están siendo monitoreados y/o controlados por las RTU.

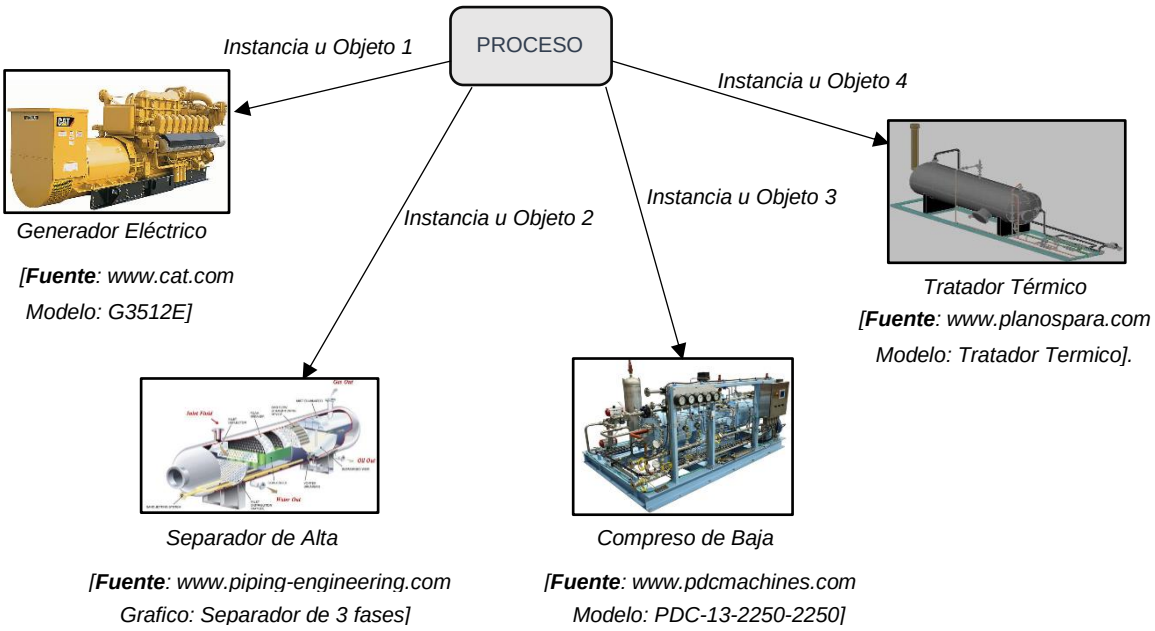


Fig. 3.3. Instanciación de la clase Proceso. [***Fuente**: Propia*]

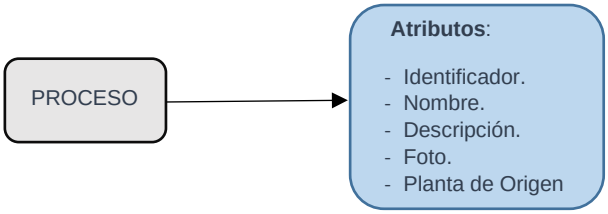


Fig. 3.4. Atributos de la clase Proceso. [***Fuente**: Propia*]

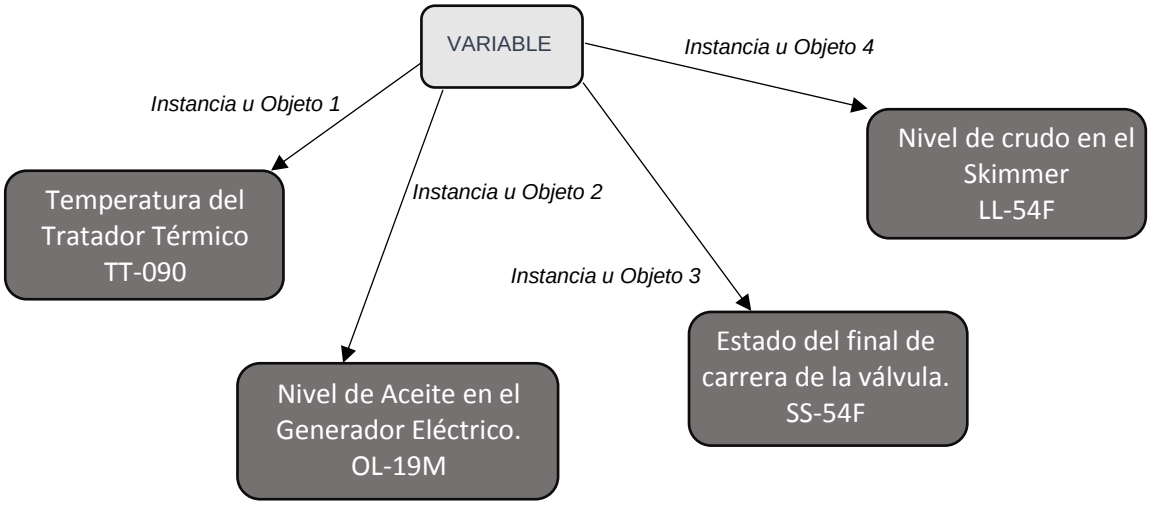


Fig. 3.5. Instanciación de la clase Variable. [***Fuente**: Propia*]

3.1.6 Atributos de la clase Variable

Algunos de los atributos que tiene una clase VARIABLE serían los que se muestran en la Fig. 3.6.

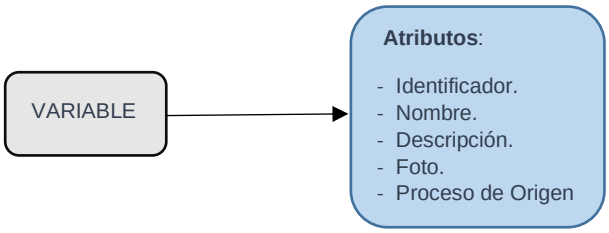


Fig. 3.6. Atributos de la clase Variable. [***Fuente**: Propia*]

3.1.7 Clase Usuario y sus instancias

Esta clase es la representación de una persona con autorización para ingresar al sistema SCADA. En la Fig. 3.7 se muestra cómo sería la INSTANCIACION de los USUARIOS. Los nombres de los usuarios son solo de ejemplo.

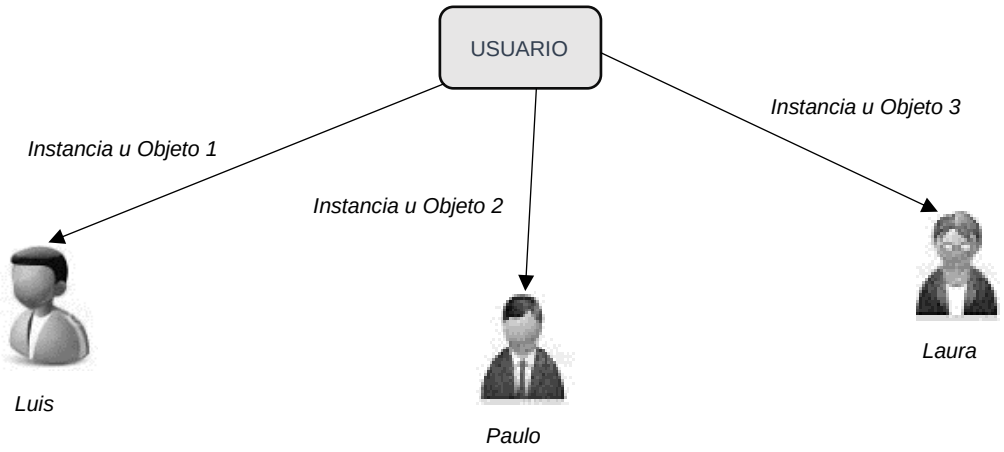


Fig. 3.7. Instanciación de la clase Usuario. [***Fuente**: Propia*]

La instanciación de la clase USUARIO es bastante sencilla, porque éste simplemente constará de la lista de personas que pueden acceder al sistema SCADA.

3.1.8 Atributos de la clase Usuario

Algunos de los atributos que tiene una clase USUARIO serían los que se muestran en la Fig. 3.8.

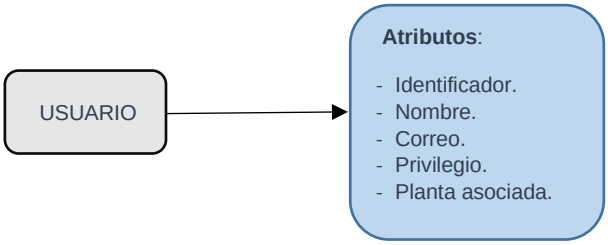


Fig. 3.8. Atributos de la clase Usuario. [***Fuente**: Propia*]

Por el momento estas serían las 4 clases con las que se trabajaran en este proyecto de Tesis, ello no significa que se puedan añadir más conforme aumente las necesidades del sistema SCADA.

3.1.9 Definición de Servicios de una Clase

Se ha visto en secciones anteriores la instanciación (o creación de objetos) de las 4 clases definidas en este proyecto (Planta, Proceso, Variable, Usuario), además de haber definido atributos a estas clases. Pero quedó pendiente la creación de métodos para las clases, si recordamos los métodos eran las acciones que podían ser realizadas por determinada clase. En este proyecto no trabajaremos con METODOS, en su lugar trabajaremos con un nuevo concepto, los SERVICIOS.

SERVICIO: hace referencia a una determinada tarea que se desarrolla sobre una o varias clases y que tiene un claro objetivo.

En las siguientes secciones mencionaremos qué tipos de SERVICIOS se pueden aplicar sobre las distintas clases (Planta, Proceso, Variable, Usuario).

a) Clase Planta y sus Servicios

Algunos de los SERVICIOS que se pueden aplicar sobre la clase Planta se muestran en la Fig. 3.9.

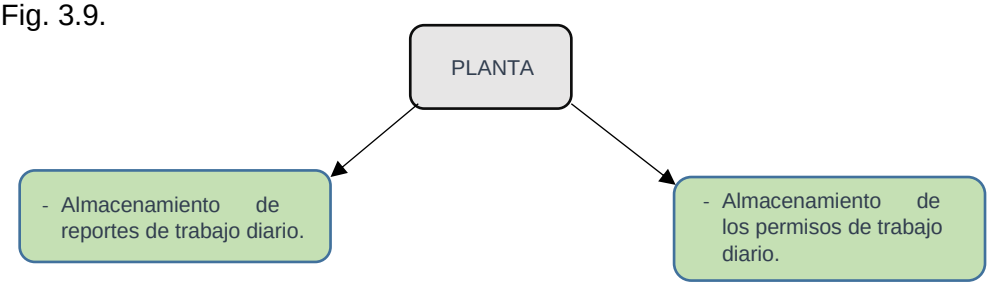


Fig. 3.9. Servicios que se pueden asociar a la clase Planta. [***Fuente**: Propia*]

b) Clase Proceso y sus Servicios

Algunos de los SERVICIOS que se pueden aplicar sobre la clase Proceso se muestran en la Fig. 3.10.

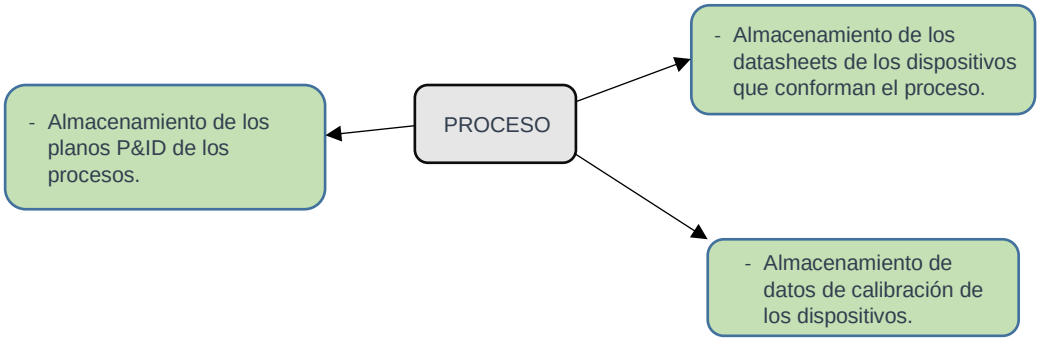


Fig. 3.10. Servicios que se pueden asociar a la clase Proceso. [***Fuente**: Propia*]

c) Clase Variable y sus Servicios

Algunos de los SERVICIOS que se pueden aplicar sobre la clase Variable se muestran en la Fig. 3.11.

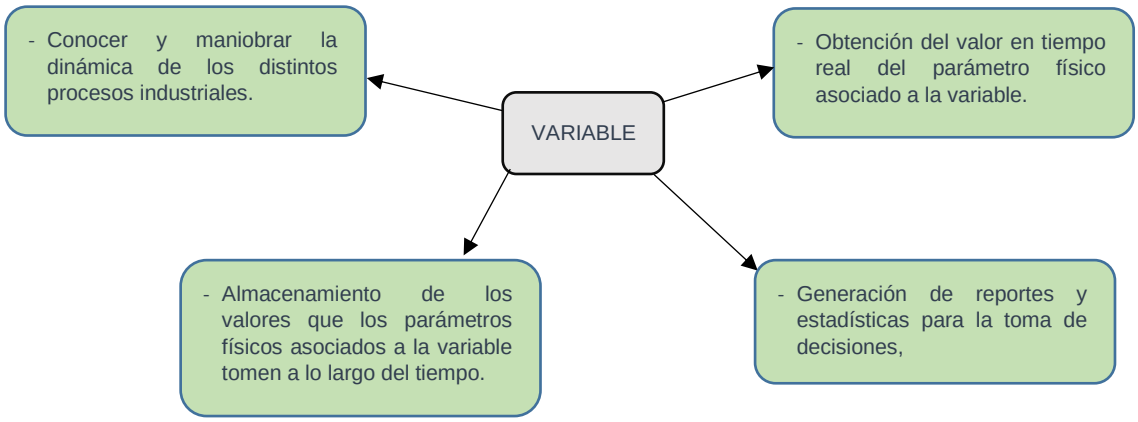


Fig. 3.11. Servicios que se pueden asociar a la clase Variable. [***Fuente**: Propia*]

d) Clase Usuario y sus Servicios

Algunos de los SERVICIOS que se pueden aplicar sobre la clase Usuario se muestran en la Fig. 3.12.

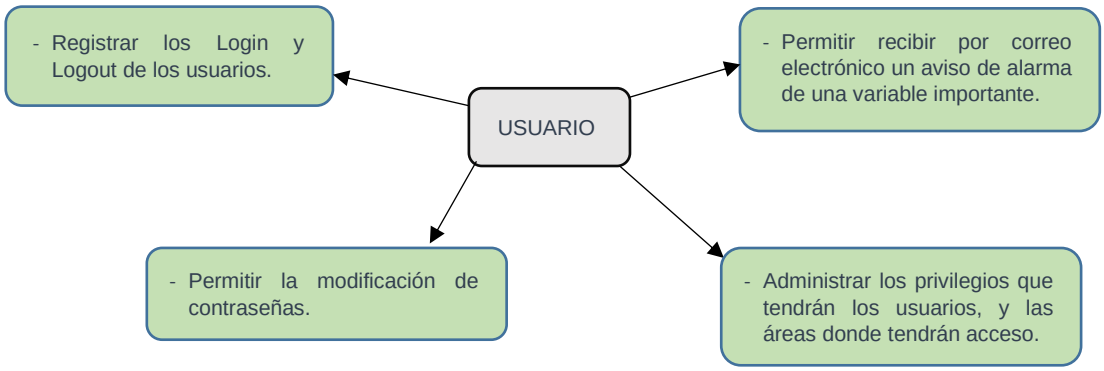


Fig. 3.12. Servicios que se pueden asociar a la clase Usuario. [***Fuente**: Propia*]

Algo que se puede observar en esta arquitectura basada en POO, es que tareas como almacenamiento de información de planos, registros de calibración, permisos de trabajos, los cuales eran muy difícil incluirlos de forma **estructurada** en un sistema SCADA, ahora sí son más sencillas de incluir.

Por ejemplo, antes se podía ubicar una pantalla del sistema SCADA correspondiente a un proceso en particular, y lo que tal vez se podía hacer era colocar en la pantalla una especie de vínculo con el plano P&ID, pero el problema es que el proceso y el plano P&ID no tendrían una relación a nivel de la programación, la relación solo sería visual.

3.1.10 Definición de Espacios

Anteriormente, se han mencionado los SERVICIOS que se pueden desarrollar sobre las CLASES (Planta, Proceso, Variable y Usuario), pero para que el SERVICIO se pueda llevar a cabo, requiere de toda una infraestructura, que incluye software y hardware. A esta infraestructura se le conoce como ESPACIO. Las herramientas que pueden estar involucradas en la construcción de un ESPACIO se muestran en la Fig. 3.13, donde las herramientas mostradas son solo de referencia y no siempre se requerirán todas.

3.1.11 Alcance

En las subsecciones anteriores se había mencionado una lista de posibles SERVICIOS que se podrían desarrollar sobre las CLASES, en este proyecto de Tesis solo se desarrollaran los SERVICIOS que se muestran en la Tabla. 3.1.

Tabla. 3.1. Lista de Servicios y Espacios que se desarrollaran en la Tesis.
[***Fuente**: Propia*]

Objetivo del SERVICIO.	Nombre que se le dará al ESPACIO asociado a este SERVICIO.	CLASE sobre la cual actúa el SERVICIO.
Instanciación de las CLASES.	SISTEMA	NO APLICA
Obtención del valor en tiempo real del parámetro físico (o estado) asociado a la variable.	CONTROL	VARIABLE
Permita conocer y maniobrar la dinámica de los distintos procesos industriales.	PROCESO	VARIABLE
Registrar los valores históricos de las variables.	HISTORICOS	VARIABLE
Generación de reportes y estadísticas para la toma de decisiones.	REPORTES	VARIABLE

A lo largo de la Tesis se irá detallando la construcción de estos ESPACIOS.

3.1.12 Conclusiones

El modelado de un sistema SCADA usando conceptos de POO permite realizar tareas que no se podían realizar en un sistema SCADA que es modelado en función a una dirección de la memoria RAM de la RTU.

Cuando se desee añadir más capacidades al sistema SCADA lo único que se tiene que hacer es definir el objetivo del SERVICIO y sobre qué CLASE se va a desarrollar.

Dada la cantidad de SERVICIOS que puede realizar este sistema de forma estructurada, los cuales exceden las capacidades de un sistema SCADA tradicional, este sistema se podría considerar como un SISTEMA DE CENTRALIZACION DE INFORMACION PARA PLANTAS INDUSTRIALES.

También, se podría decir que este sistema es un SISTEMA ORIENTADO A SERVICIOS.

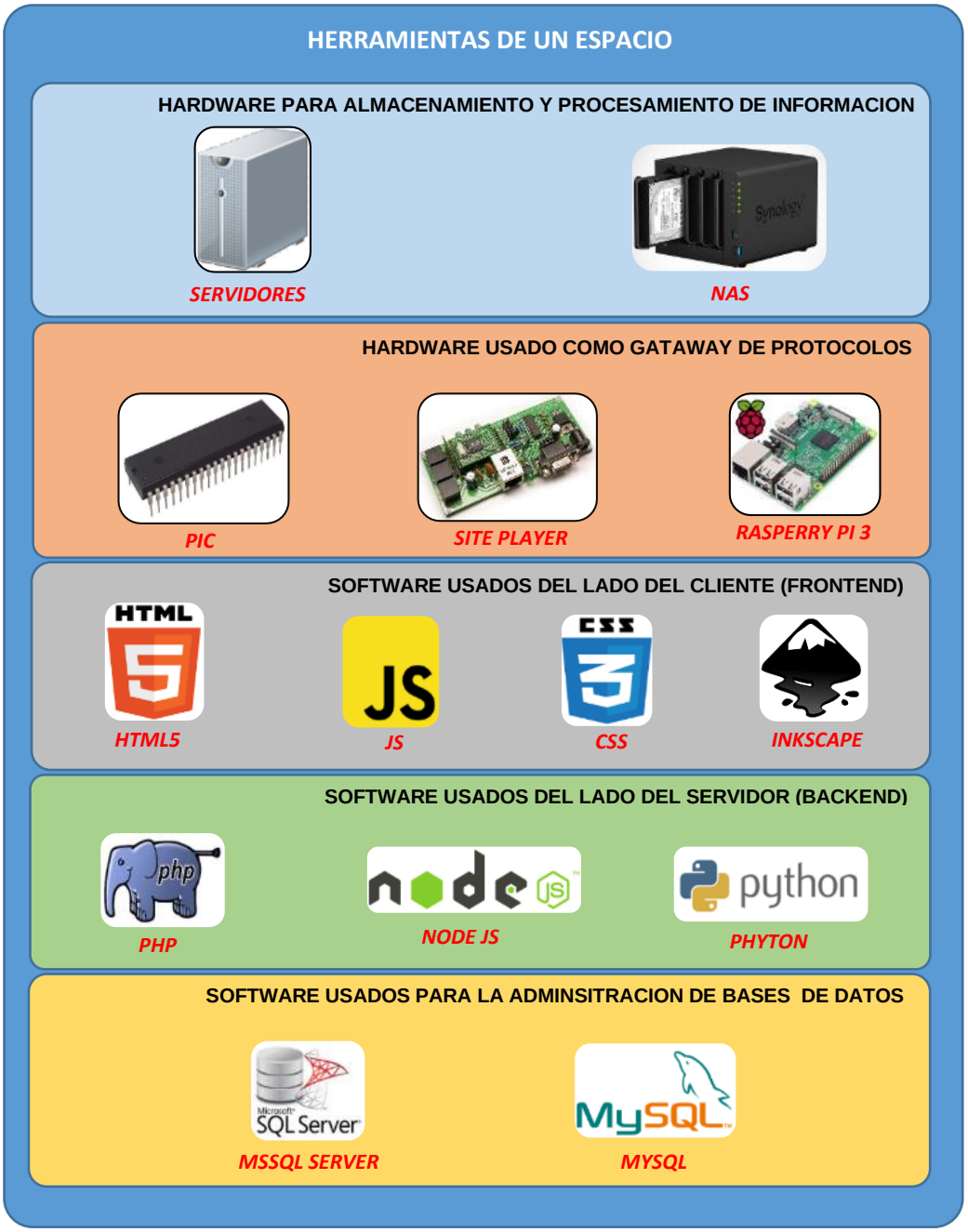


Fig. 3.13. Herramientas que usa un Espacio para poder cumplir con el Servicio establecido. [***Fuente**: Propia*]

3.2 Instanciación de las Clases (Espacio Sistema)

3.2.1 Objetivo del Espacio Sistema

Según lo visto en el capítulo anterior, el Espacio SISTEMA es la infraestructura requerida para proporcionar el siguiente servicio: “La instanciación de las CLASES”.

Teniendo en cuenta que se han definido 4 CLASES: Planta, Proceso, Variable y Usuario, entonces “La instanciación de CLASES” no es otra cosa más que enlistar todas las Plantas, Procesos, Variables y Usuarios que hay en una determinada empresa.

3.2.2 Pasos para el uso del Espacio Sistema

a) Generación de documentos

Consiste en obtener la lista de las Plantas, Procesos, Variables y Usuarios.

Esta lista puede ser proporcionada por la empresa que solicita el sistema SCADA, mediante la entrega de planos eléctricos, P&ID y otros documentos de ingeniería. Si la empresa no dispone de esta documentación, se tendrá que crear la lista tomando el criterio que se crea más conveniente. En esta situación la instanciación de la clase PROCESO será donde se debe tener más cuidado, ya que los límites entre los PROCESOS, a veces no están muy bien definidos.

b) Ingreso de los Objetos

La lista de objetos (o instancias de las CLASES) serán ingresadas al sistema SCADA a través de formularios hechos con tecnologías de Frontend (HTML5, Java Script, CSS y otros Frameworks), los cuales en conjunto con programas de Backend (PHP nativo) terminarán almacenando la información en la Base de Datos.

En la Fig. 3.14 se puede observar la secuencia que se acaba de explicar.

3.2.3 Aplicación a un ejemplo real

En la Tesis se va suponer que hay una empresa que solicita la implementación de un sistema SCADA. Se supone que esta empresa dispone de las siguientes Plantas, Procesos y Variables, mostradas en la Fig. 3.15.

3.2.4 Ingreso de documentación al sistema SCADA

En Fig. 3.15, Fig. 3.16 y Fig. 3.17 se muestra la instanciación de las distintas CLASES del sistema SCADA. Se tomará como referencia la lista que se muestra en la Fig. 3.15.

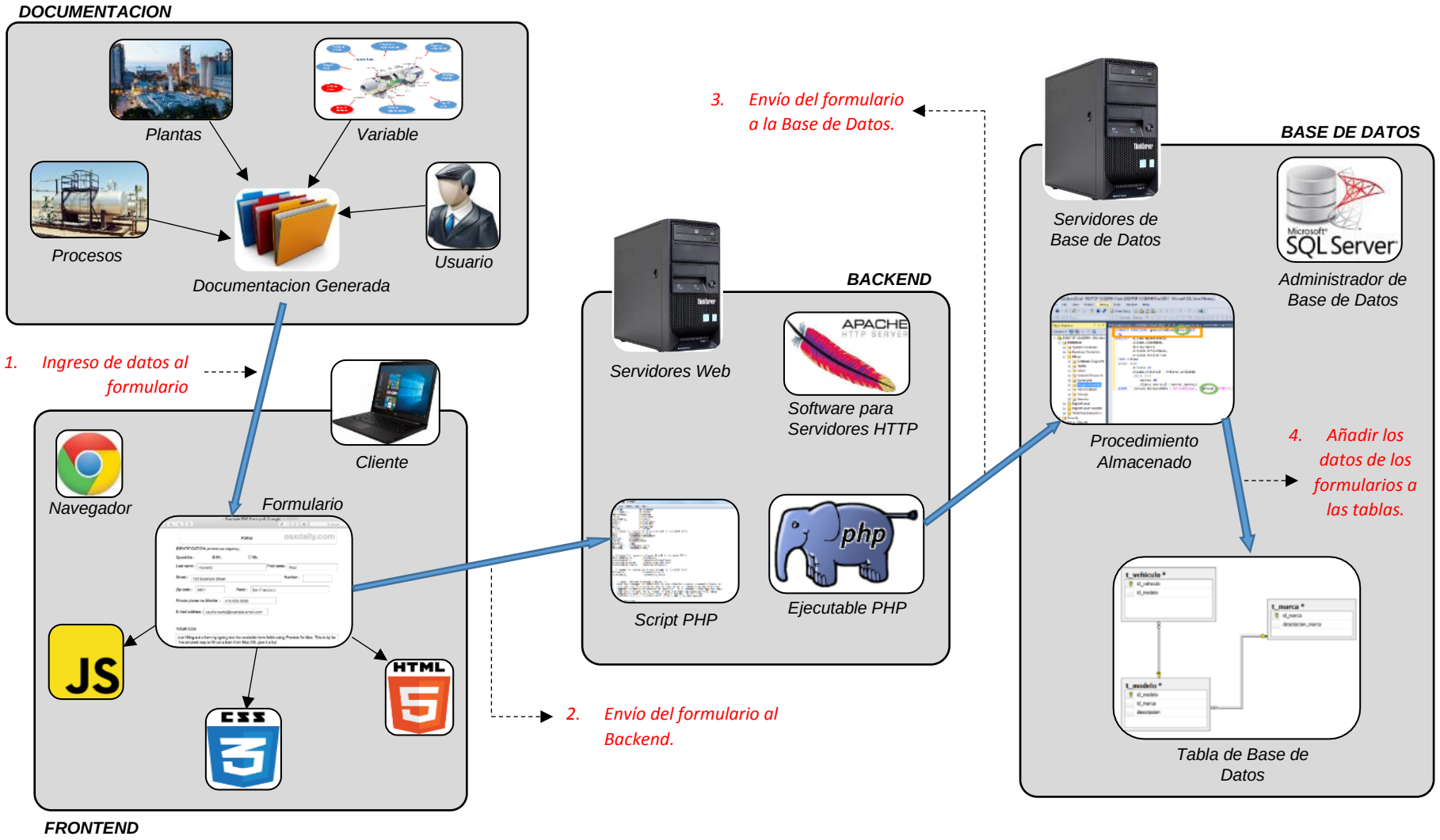


Fig. 3.14. Herramientas del Espacio Sistema [***Fuente**: Propia*]

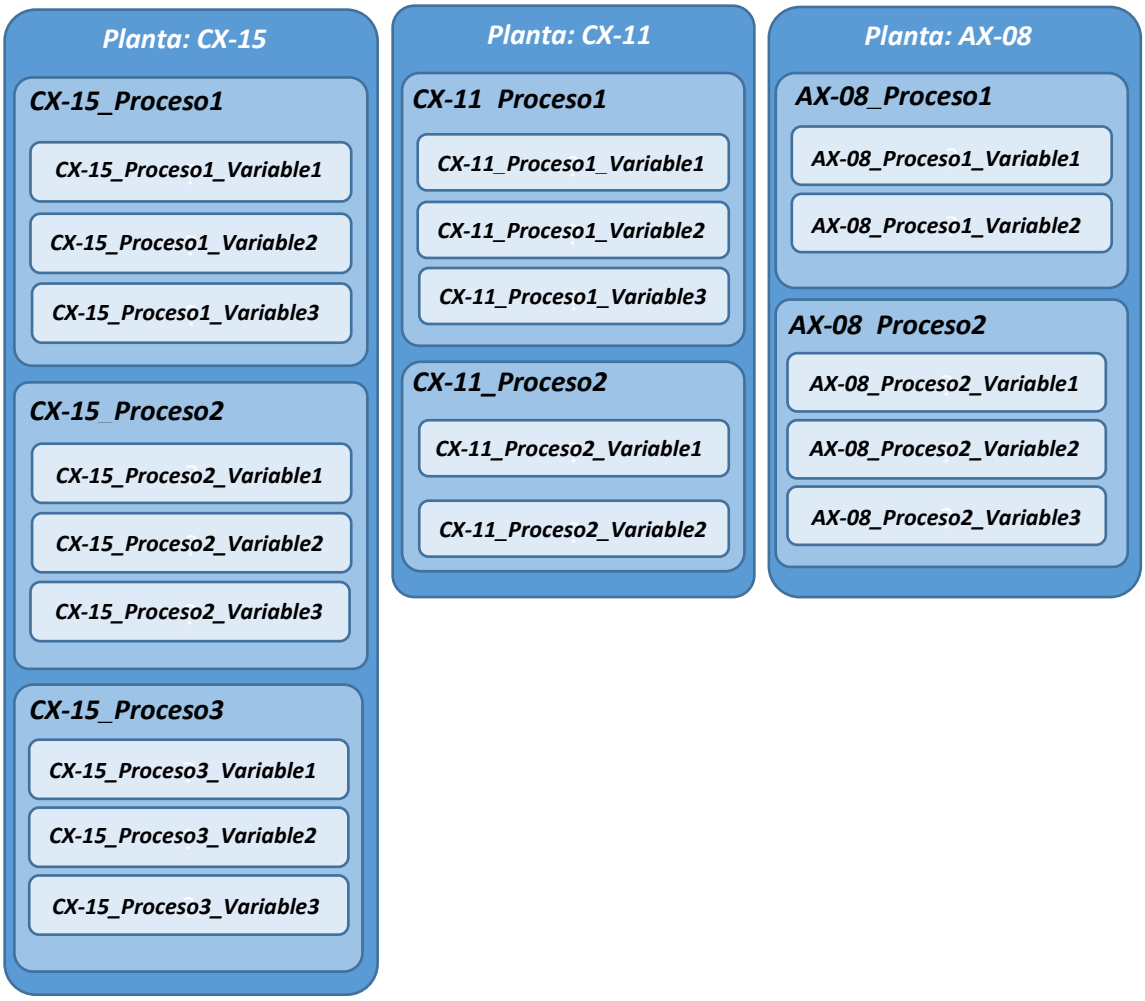


Fig. 3.15. Lista que se utilizará como ejemplo en la Tesis [***Fuente**: Propia*]

3.2.5 Las instancias de las Clases en la Base de Datos

Todos los datos de las instancias que se han generado se almacenan en la Base de Datos tal como lo muestra el esquema general de la Fig. 3.14.

La Fig. 3.19 muestra el vínculo que existe entre las 3 CLASES, aparte de mostrar sus respectivos atributos.

La Fig. 3.20 muestra los datos que se han almacenado en la Base de Datos, los mismos que se basan en las instancias de la Fig. 3.15.

Fig. 3.16. Instanciación de una Clase tipo Planta [***Fuente**: Propia*]

Fig. 3.17. Instanciación de una Clase tipo Proceso [***Fuente**: Propia*]

Fig. 3.18. Asignación de instancias de una Clase tipo Variable [***Fuente**: Propia*]

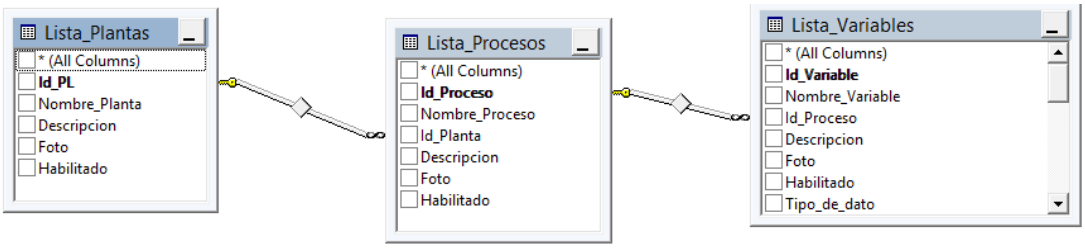


Fig. 3.19. Tablas correspondientes a las 3 Clases [***Fuente**: Propia*]

dbo.Lista_Plantas

	Id_PL	Nombre_Planta	Descripcion	Foto	Habilitado
1	PL001	CX_15	Esta plataforma se llama Delfines	cx 15.png	No
2	PL002	CX_11	Este es llamado Corvina	CX_11.png	Si
3	PL003	AX_08	Esta plataforma tambien se le conoce como Albacora	AX-08.png	Si

dbo.Lista_Procesos

	Id_Proceso	Nombre_Proceso	Id_Planta	Descripcion	Foto	Habilitado
1	PRC0001	cx15_proceso1	PL001		cx15proceso1.png	Si
2	PRC0002	cx15_proceso2	PL001	Este proceso corresponde al Tratador Termico.	cx15proceso2.png	Si
3	PRC0003	cx15_proceso3	PL001		cx15proceso3.png	Si
4	PRC0004	cx11_proceso1	PL002		cx11proceso1.png	Si
5	PRC0005	cx11_proceso2	PL002		cx11proceso2.png	Si
6	PRC0006	ax08_proceso1	PL003		ax08proceso1.png	Si
7	PRC0007	ax08_proceso2	PL003		ax08proceso2.png	Si

dbo.Lista_Variables

	Id_Variable	Nombre_Variable	Id_Proceso	Descripcion	Foto	Habilitado	Tipo_de_dato
1	VR000015	ax08_proceso1_variable1	PRC0006		ax08_proceso1_variable1.png	Si	Digital
2	VR000016	ax08_proceso1_variable2	PRC0006		ax08_proceso1_variable2.png	No	Analogico
3	VR000017	ax08_proceso2_variable1	PRC0007		ax08_proceso2_variable1.png	No	Analogico
4	VR000018	ax08_proceso2_variable2	PRC0007		ax08_proceso2_variable2.png	No	Analogico
5	VR000019	ax08_proceso2_variable3	PRC0007		ax08_proceso2_variable3.png	Si	Digital
6	VR000010	cx11_proceso1_variable1	PRC0004		cx11_proceso1_variable1.png	Si	Analogico
7	VR000011	cx11_proceso1_variable2	PRC0004		cx11_proceso1_variable2.png	Si	Digital
8	VR000012	cx11_proceso1_variable3	PRC0004		cx11_proceso1_variable3.png	No	Analogico
9	VR000013	cx11_proceso2_variable1	PRC0005		cx11_proceso2_variable1.png	Si	Digital
10	VR000014	cx11_proceso2_variable2	PRC0005		cx11_proceso2_variable2.png	Si	Digital
11	VR000001	cx15_proceso1_variable1	PRC0001		cx15_proceso1_variable1.png	No	Digital
12	VR000002	cx15_proceso1_variable2	PRC0001		cx15_proceso1_variable2.png	No	Analogico
13	VR000003	cx15_proceso1_variable3	PRC0001		cx15_proceso1_variable3.png	No	Analogico
14	VR000004	cx15_proceso2_variable1	PRC0002		cx15_proceso2_variable1.png	No	Analogico
15	VR000005	cx15_proceso2_variable2	PRC0002		cx15_proceso2_variable2.png	No	Digital
16	VR000006	cx15_proceso2_variable3	PRC0002		cx15_proceso2_variable3.png	No	Analogico
17	VR000007	cx15_proceso3_variable1	PRC0003		cx15_proceso3_variable1.png	Si	Digital
18	VR000008	cx15_proceso3_variable2	PRC0003		cx15_proceso3_variable2.png	No	Analogico
19	VR000009	cx15_proceso3_variable3	PRC0003		cx15_proceso3_variable3.png	Si	Digital

Fig. 3.20. Instancias de las Clases: Planta, Proceso y Variable, almacenadas en la Base de Datos.

3.3 Acceso a la memoria RAM del RTU (Espacio Control)

3.3.1 Objetivo del Espacio Control

Tal como se había definido en secciones anteriores, este espacio tiene por finalidad la obtención de los valores en tiempo real que toman las instancias de la CLASE VARIABLE.

Por ejemplo, si se toma una de las variables definidas en la Fig. 3.15., cx11_proceso2_variable2, y se asume que esta variable está asociada a un parámetro físico, por ejemplo, la temperatura de un tanque, entonces el ESPACIO CONTROL tiene que proporcionar en tiempo real los valores de la temperatura y hacer que se encuentre disponible en el sistema SCADA. En la Fig. 3.21., se muestra gráficamente el objetivo del ESPACIO CONTROL.

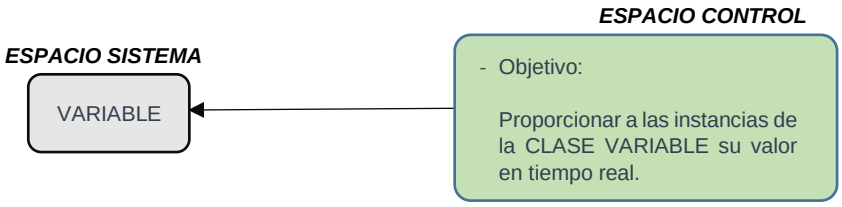


Fig. 3.21. Objetivo del Espacio Control. [***Fuente**: Propia*]

3.3.2 Acceso a la RTU

La única forma que el sistema SCADA tiene para poder obtener el valor de una variable, es accediendo a la memoria RAM de la RTU, el cual está conectado al sensor o actuador asociado a la variable, tal como se muestra en la Fig. 3.22.

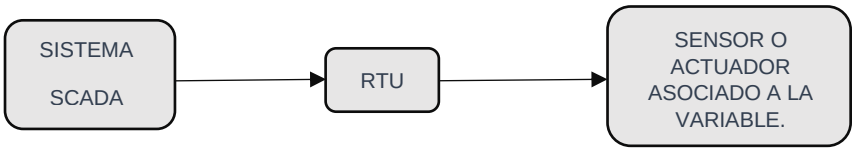


Fig. 3.22. Sistema SCADA y dispositivo final de control conectados a través de la RTU. [***Fuente**: Propia*]

3.3.3 Comunicación del sistema SCADA con las RTUs

Las RTU son dispositivos electrónicos que procesan los valores que le son entregados por los dispositivos finales de control (sensores y actuadores) mediante el uso de algoritmos, y una vez finalizado el procesamiento de estos datos, son entregados al sistema SCADA.

La comunicación que establece el sistema SCADA con las RTUs se realiza mediante protocolos, que son un conjunto de reglas que usan los dispositivos electrónicos para transmitir y recibir información. El sistema SCADA y la RTU deben tener implementado necesariamente el mismo protocolo de comunicación, para poder intercambiar información.

El protocolo más ampliamente usado para comunicar RTUs con sistemas SCADA es el protocolo Modbus-RTU, y salvo los PLCs, casi todos las demás RTUs (como medidores de flujo, variadores de velocidad, sistemas contraincendios) tienen implementados el protocolo Modbus-RTU**.** Un problema para Modbus-RTU apareció cuando los PLCs comenzaron a implementar sus protocolos propietarios basándose en la norma IEEE802.3 y en la suite TCP/IP. Desde entonces, el sistema SCADA podía estar a una distancia "infinita" de los PLC, haciendo imposible que las RTUs que tenían implementado el protocolo Modbus-RTU puedan comunicarse con el sistema SCADA, ya que el protocolo Modbus RTU solo tiene un alcance de 1200 metros. Ver la Fig. 3.23. para una mejor compresión.

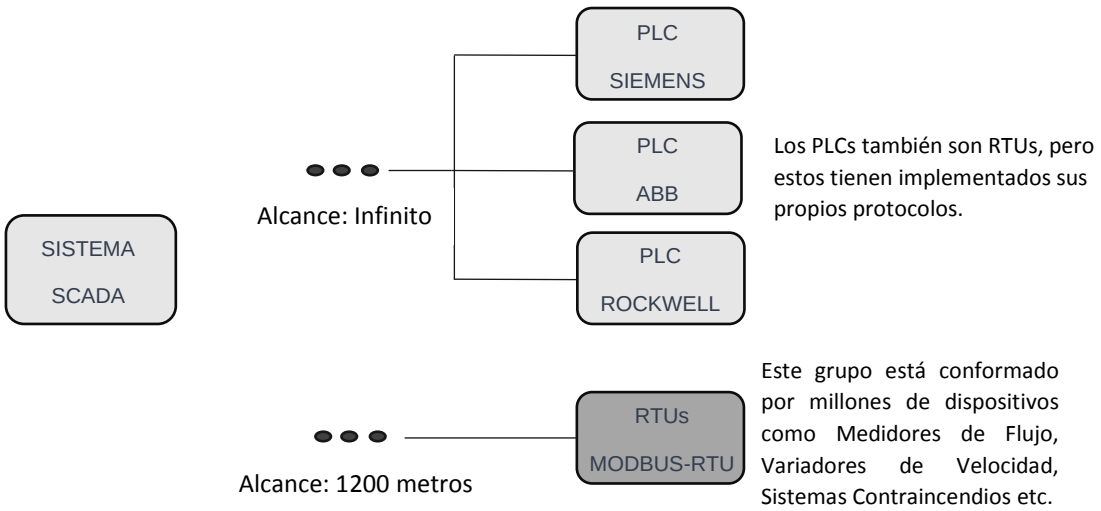


Fig. 3.23. Falencias de las RTUs implementadas con el protocolo Modbus-RTU en lo que respecta a su alcance. [***Fuente**: Propia*]

Para superar el problema del alcance, se implementó módulos en los PLCs, que permiten conectase con las RTUs que tenían implementados Modbus-RTU, de esta forma el PLC serviría como un Gateway, tal como se ve en la Fig. 3.24.

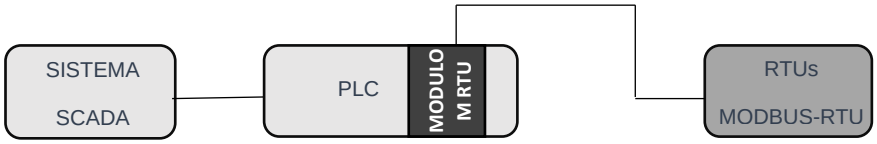


Fig. 3.24. Método más usado para que las RTU lleguen hasta el sistema SCADA. [***Fuente**: Propia*]

El problema que hay con el método mostrado en la figura anterior, es que colocar un PLC para funcionar solamente de "puente" es algo muy costoso, aunque es al día de hoy el método más usado.

Otro método (no muy usado) para que los datos de una RTU que tiene implementado el protocolo Modbus-RTU lleguen hasta el sistema SCADA, es mediante el uso de un conversor de Modbus-RTU a Modbus-TCP/IP, tal como se muestra en la Fig. 3.25.

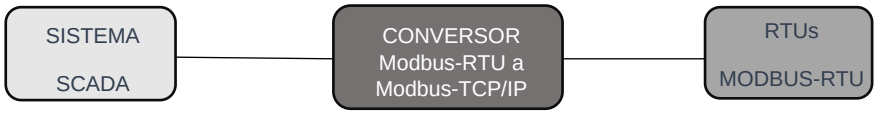


Fig. 3.25. Conversor de Modbus-RTU a Modbus-TCP/IP, para que la RTU supere el problema del alcance. [***Fuente**: Propia*]

El inconveniente con el método anterior es que el conversor es un dispositivo muy costoso, aparte que este protocolo (Modbus TCP/IP) es vulnerable a los ataques, ya que carece de encriptación, entre otras cosas. La Tesis enfocará toda la atención en la comunicación del sistema SCADA con RTUs que tengan implementado el protocolo de comunicación Modbus-RTU, y se proporcionará una solución más económica que las vistas líneas arriba.

3.3.4 Estrategias para el correcto uso del Espacio Control

Si bien se ha mencionado que en la Tesis solo se trabajará con el protocolo Modbus-RTU, la estrategia que se mencionará aplica para cualquier protocolo que se piense adicionar en el futuro, por tal razón se trabajará con un protocolo genérico de nombre "Protocolo-X". Este protocolo NO existe en el mercado, solo es de ejemplo.

a) Reconocimiento de las RTU

Este paso consiste en ubicar a todos las RTUs que se comunicarán con el sistema SCADA. En la Fig. 3.26., se muestra una lista de RTUs que tienen implementados el "Protocolo-X", que como ya se había mencionado, se usa de referencia solamente. En caso de tener más RTUs con otros protocolos diferentes, se debe hacer una lista de las RTUs que hay por cada protocolo.

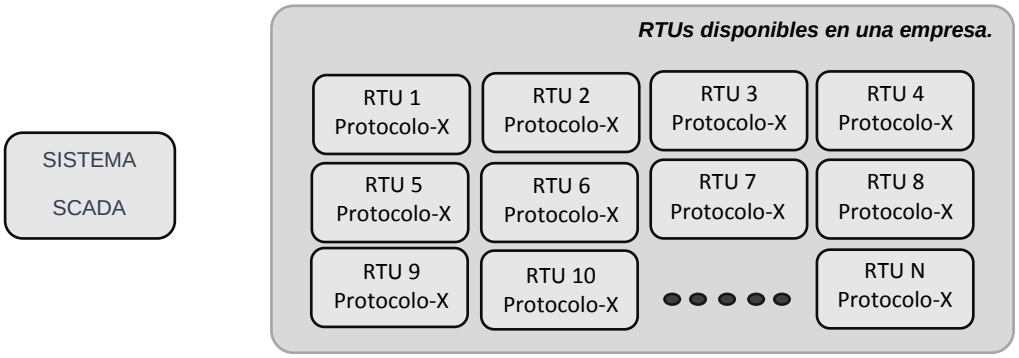


Fig. 3.26. Tener una lista de las RTUs es el primer paso para el uso del Espacio Control. [***Fuente**: Propia*]

b) Establecimiento del número de Gateways

Es importante mencionar que el sistema SCADA no se conectará directamente a las RTUs, sino que lo hará a través de dispositivos electrónicos llamados Gateways. Estos Gateways se comunicarán con la RTU (o RTUs) a través del "Protocolo-X", el cual es un protocolo que en realidad no existe en el mercado y solo ha sido creado como referencia en este proyecto de Tesis, y se comunicarán con el sistema SCADA usando el protocolo HTTP. La Fig. 3.27., muestra de forma detallada lo que se acaba de explicar.

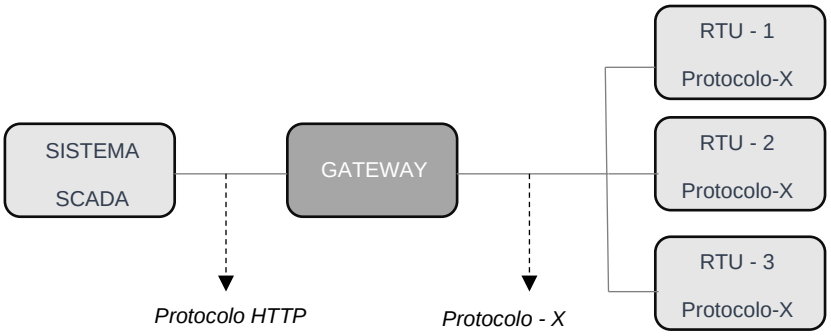


Fig. 3.27. Los Gateway funcionan como intermediarios entre el sistema SCADA y las RTUs [***Fuente**: Propia*].

Otro tema que es importante recalcar es que la construcción de los Gateways es parte de la elaboración de este sistema SCADA, lo cual significa que se va a tener que implementar el Protocolo-X, así como el protocolo HTTP, tanto a nivel de hardware como software**,** la construcción de los Gateways es la parte más compleja de todo este proyecto.

Los Gateways siempre serán ubicados en la misma locación donde se encuentra la RTU (o RTUs), es decir, en el mismo tablero o caseta de control. Cuando el protocolo de la RTU (Protocolo-X) no tiene gran alcance, es decir, la distancia de separación entre la RTU y el sistema SCADA no puede ser muy larga, como en el caso de Modbus-RTU,

entonces un beneficio que se puede obtener a partir del uso del Gateway sería el nuevo alcance de las RTU. Los Gateways serán usados aún así el protocolo de la RTU (Protocolo-X) pueda establecer una conexión directa entre la RTU y un sistema SCADA que está muy alejado, como es el caso de los protocolos que usan los PLCs en la actualidad, los cuales tienen alcance infinito.

Ante lo afirmado líneas arriba, una pregunta que probablemente se formule el lector es ¿Por qué sería necesario usar Gateway incluso si el protocolo de la RTU puede establecer conexión directa con un sistema SCADA que está ubicado geográficamente lejos? ¿Cuál sería el beneficio? A continuación, se mencionarán las razones:

1. El principal beneficio está relacionado con la seguridad, pues muchas veces el protocolo de la RTU (Protocolo-X), ver Fig. 3.28., no tiene implementada herramientas para la encriptación, lo cual puede permitir a un intruso obtener la información mediante el uso de un analizador de protocolos. HTTP es el protocolo donde se puede encontrar más herramientas relacionas con la encriptación (Keith Stouffer, Joe Falco, Karen Scarfone. - Mayo 2013 - Guide to Industrial Control Systems (ICS) Security) [11].

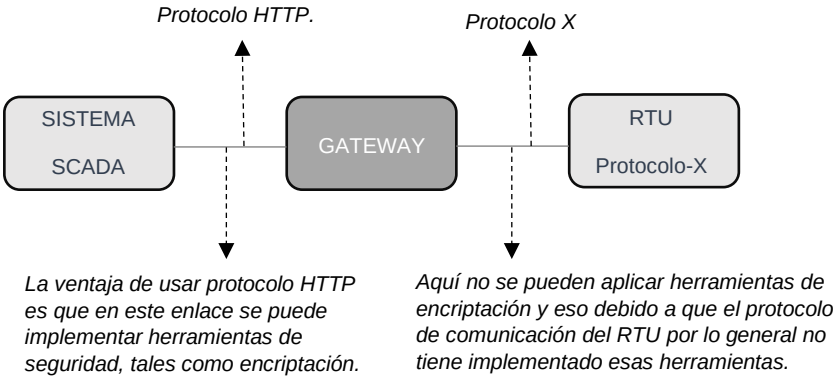


Fig. 3.28. Ventajas de usar el protocolo HTTP, las cuales se pueden aprovechar cuando se usa un Gateway. [***Fuente**: Propia*].

2. Se reduce la cantidad de comandos que se generan desde el sistema SCADA, ya que el sistema SCADA ejecutará solo un comando hacia el Gateway, evitando de esta forma que el sistema SCADA se vea en la necesidad de usar tantos puertos TCP, los cuales consumen muchos recursos y procesador. En la Fig.3.29. los “m” comandos que genera el Gateway se aplican sobre una sola RTU, pero pueden estar distribuidos en varias RTUs, ello no altera el análisis. El sistema SCADA solo necesita ejecutar un comando para obtener todos los datos de un Gateway, sin importar a cuantas RTUs esté conectado el Gateway, lo que le permite ahorrar recursos como velocidad de procesador, algo que no ocurriría si el sistema SCADA tendría que ejecutar los “m” comandos que ejecuta el Gateway.

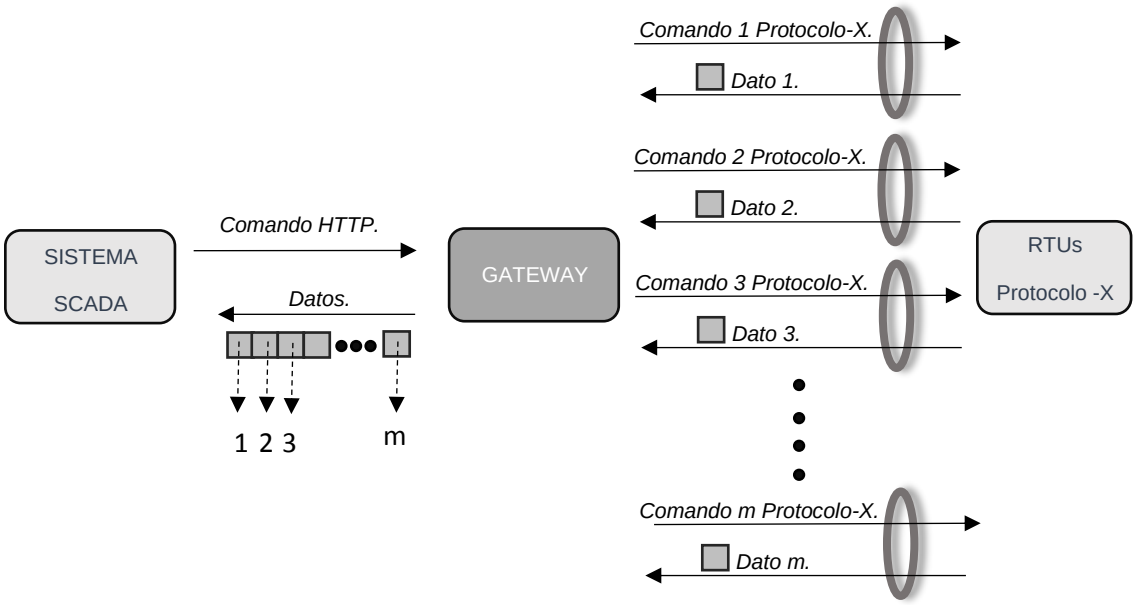


Fig. 3.29. Comando generado por el sistema SCADA hacia un Gateway.
[***Fuente**: Propia*]

3. Otra ventaja es que la RTU en caso de tener un protocolo que esté soportado sobre la suite TCP/IP, este puede ser configurado con una dirección IP privada, lo cual permite "ocultar" la RTU de ordenadores externos (como hackers) que tengan la intención de dañar la RTU.
4. La ventaja más importante radicaría en los costos bajos de los servicios de internet, porque el sistema SCADA ejecutará menos comandos debido a la existencia del Gateway y no será indispensable tener velocidades tan elevadas.

También se debe tener en cuenta, la cantidad de Gateways que se requerirán, porque dependiendo de varios factores se puede necesitar muchos Gateways. Entre otros, estos factores son:

1. **La distancia.** Los Gateway siempre deben estar lo más cerca posible de su correspondiente RTU porque esto posibilita que la comunicación con las RTU se dé a una velocidad bastante elevada, a parte que la tasa de errores baja considerablemente conforme la distancia es menor.

   Por esta razón, se debe evitar usar el mismo Gateway para comunicarse con RTUs que estén muy alejadas entre sí.

   En la Fig. 3.30., se observa que, la RTU 4, está muy alejada de las otras RTU, por consiguiente, no hubiera sido buena idea comunicar el GATEWAY 1 con la RTU 4, ya que estos estarían muy alejados, por tal razón se decidió añadir un Gateway más, con el nombre GATEWAY 2, que si está cerca de la RTU 4.

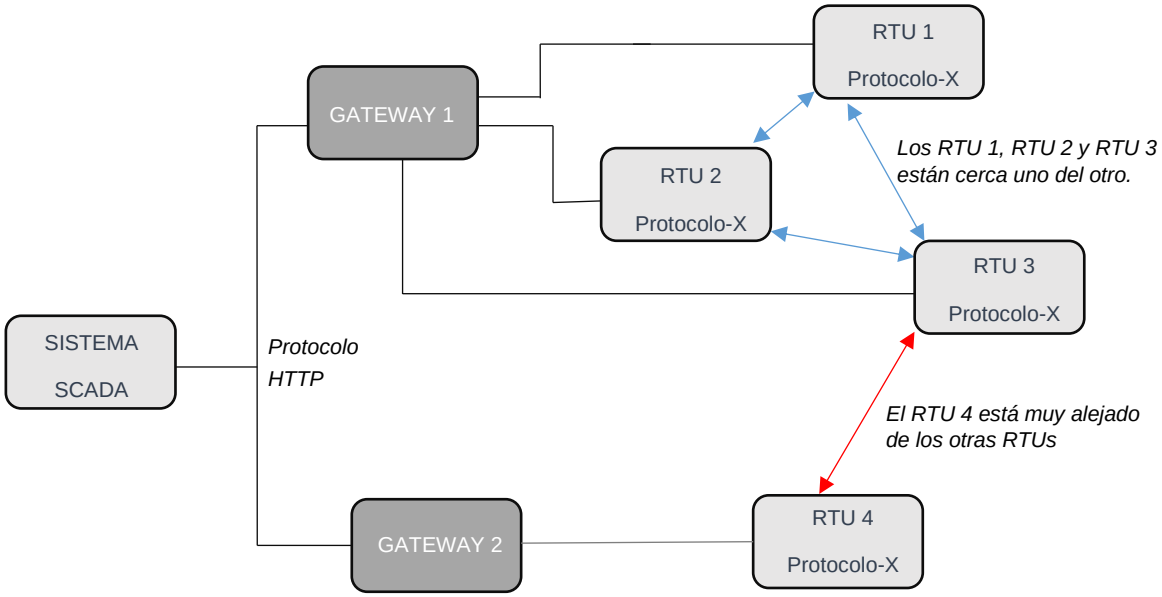


Fig. 3.30. Gateways ubicadas muy cerca de sus correspondientes RTUs.
[***Fuente**: Propia*]

2. **La cantidad de variables por RTU.** Cuando una RTU está asociado a un gran número de variables de campo, será recomendable el uso de un Gateway exclusivamente para esa RTU y ello se debe a lo siguiente:
   - Ya que por cada variable de campo que tenga asociado la RTU, el Gateway ejecutará un comando de lectura (política establecida en la Tesis).
   - Los Gateway a implementar ejecutarán los comandos hacia la RTU (o RTUs) de forma secuencial, es decir, para que un comando se ejecute, el proceso del anterior comando tiene que haber finalizado completamente, o dicho en otras palabras, no se usará multiplexaciones, con la finalidad de no consumir muchos recursos a la RTU, tal como procesador, memoria puertos TCP etc.
   - Lo anterior lleva a una conclusión, que conforme una RTU tenga más variables de campo asociadas, el ciclo de lectura de estas variables tomara más tiempo. El ciclo de lectura significa el tiempo que le toma a un Gateway en obtener el valor de todas las variables que están almacenadas en sus correspondientes RTUs.

Por tal razón, se recomendará que, si una RTU tiene demasiadas variables de campo asociadas, se destine exclusivamente un Gateway a esa RTU. Por ejemplo, si se observa la Fig. 3.31., en la que hay 2 RTUs, RTU 1 y RTU 2, que tienen una gran cantidad de variables asociadas, “n” y “m” respectivamente, No hubiese sido buena idea que Gateway 1 se conectara con RTU 1 y con RTU 2, porque el ciclo de lectura sería de “t1+t2”, lo que haría más lento el proceso de actualización de valores, por tal razón RTU 1 y RTU 2 tienen asignados un Gateway cada una.

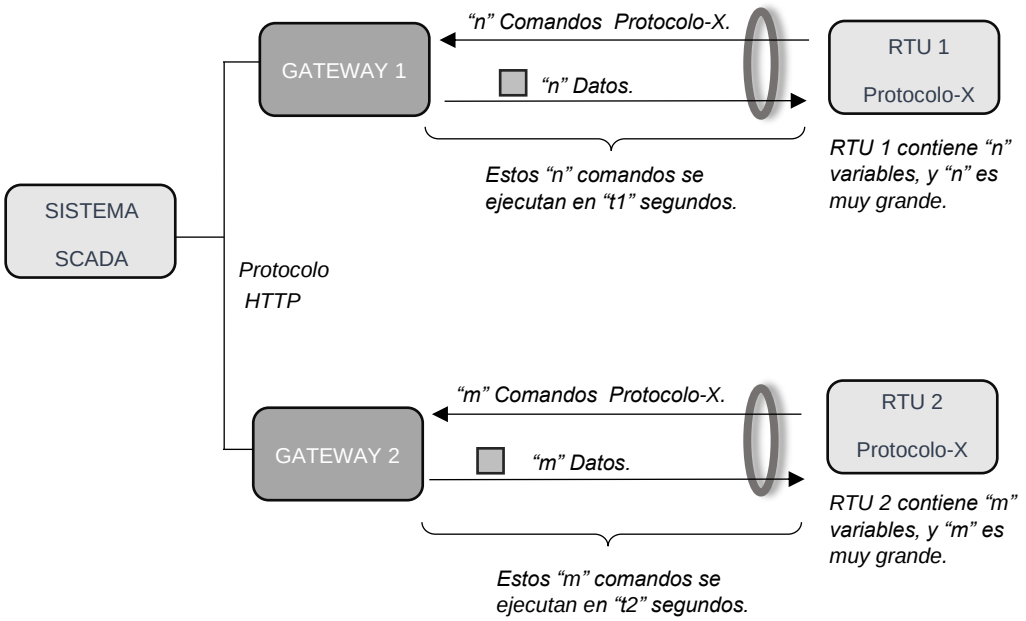


Fig. 3.31. Un Gateway se debe asociar solamente a un RTU en caso este último tenga una gran cantidad de variables. [***Fuente**: Propia*]

c) Definición de registros que se leerán de la RTU

Las RTU por lo general siempre ofrecen a los sistemas SCADA una cantidad enorme de registros (o datos), que incluyen:

- Datos de parámetros físicos y/o estados físicos obtenidos por los dispositivos finales con los que está conectada la RTU.
- Datos que el usuario pudo haber ingresado manualmente a la RTU durante la configuración, por ejemplo, en el caso de un variador de velocidad, los datos de la placa de un motor son ingresados manualmente.
- Datos concernientes a fallas que pueden haber ocurrido durante el funcionamiento de la RTU.

Luego, es muy importante definir cuáles serán los registros o datos que el sistema SCADA extraerá de la RTU porque no se debe saturar innecesariamente al sistema SCADA con datos que nunca serán analizados por el operador.

En la Fig. 3.32., se puede observar un ejemplo de una RTU con nombre RTU-1, donde se hizo una selección de los registros que van a ser extraídos para el sistema SCADA. En este ejemplo, si bien la RTU-1 tenía "k" registros, solo se van a extraer "n" registros o variables, por tal razón, Gateway 1 tendrá que generar "n" comandos para extraer las "n" variables de RTU-1. En la Fig. 3.32., solo se dibujó una RTU, pudo haberse dibujado más RTUs conectadas al Gateway 1.

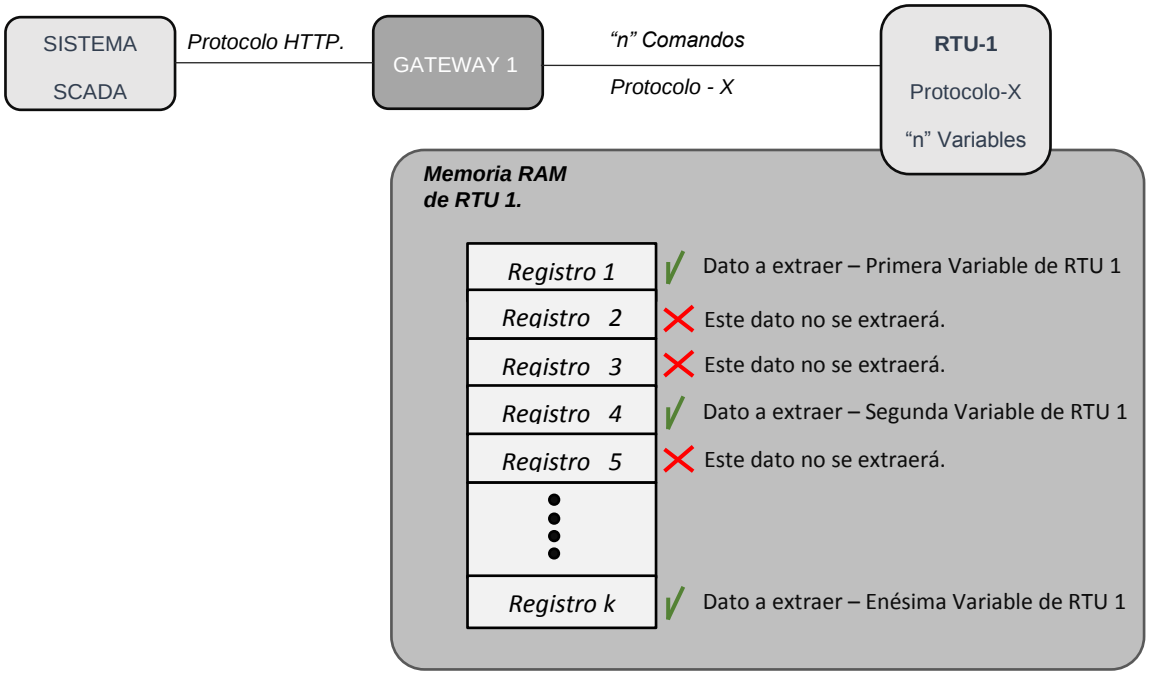


Fig. 3.32. Registros o variables de la RAM de la RTU que serán extraídos por el sistema SCADA. [***Fuente**: Propia*]

d) Vínculo del Espacio Control con el Espacio Sistema

Se debe recordar que el ESPACIO CONTROL es la infraestructura construida para brindar un servicio muy concreto al ESPACIO SISTEMA. Este servicio es "Obtener los valores en tiempo real que toman las Instancias (u OBJETOS) de la CLASE VARIABLE".

Entonces, lo que se realizará ahora es vincular las variables o registros que se extrajeron de las RTUs, tal como se mostró en la Fig. 3.32. y las Instancias (u OBJETOS) de la CLASE VARIABLES, tal y como lo puede observar en la Fig. 3.33.

Como se verá más adelante una instancia de la CLASE VARIABLE del ESPACIO SISTEMA no puede estar vinculado a más de un registro (o variable) del ESPACIO CONTROL.

Para que la Fig. 3.33., no genere confusión en el lector se debe tener en cuenta que:

- Las RTUs pueden tener implementados diferentes protocolos de comunicación, obviamente tendrían que estar conectados a distintos Gateways.
- La Fig. 3.33. es de tipo Lógico y no físico porque la RTU se conecta a un Gateway, y este se conecta al SCADA, que es donde está alojada todo el ESPACIO SISTEMA.
- Algunas de las variables, que el operador podría considerar necesario extraer de las RTUs pueden no tener su correspondiente variable en el ESPACIO SISTEMA, ante ello se debe crear una nueva instancia de la CLASE VARIABLE en el ESPACIO SISTEMA. El problema que se acaba de mencionar puede ocurrir, porque el

ESPACIO SISTEMA en un comienzo obtiene la lista de sus variables (o instancias de la CLASE VARIABLES) de documentos tal como el P&ID. Las RTUs pueden contener datos que no estén relacionados solamente al P&ID, sino también al diagnóstico de algunos de los dispositivos de campo con los que está conectada la RTU, algo muy común en dispositivos de campo que se comunican con su respectiva RTU a través de protocolos como Fieldbus o Devicenet.

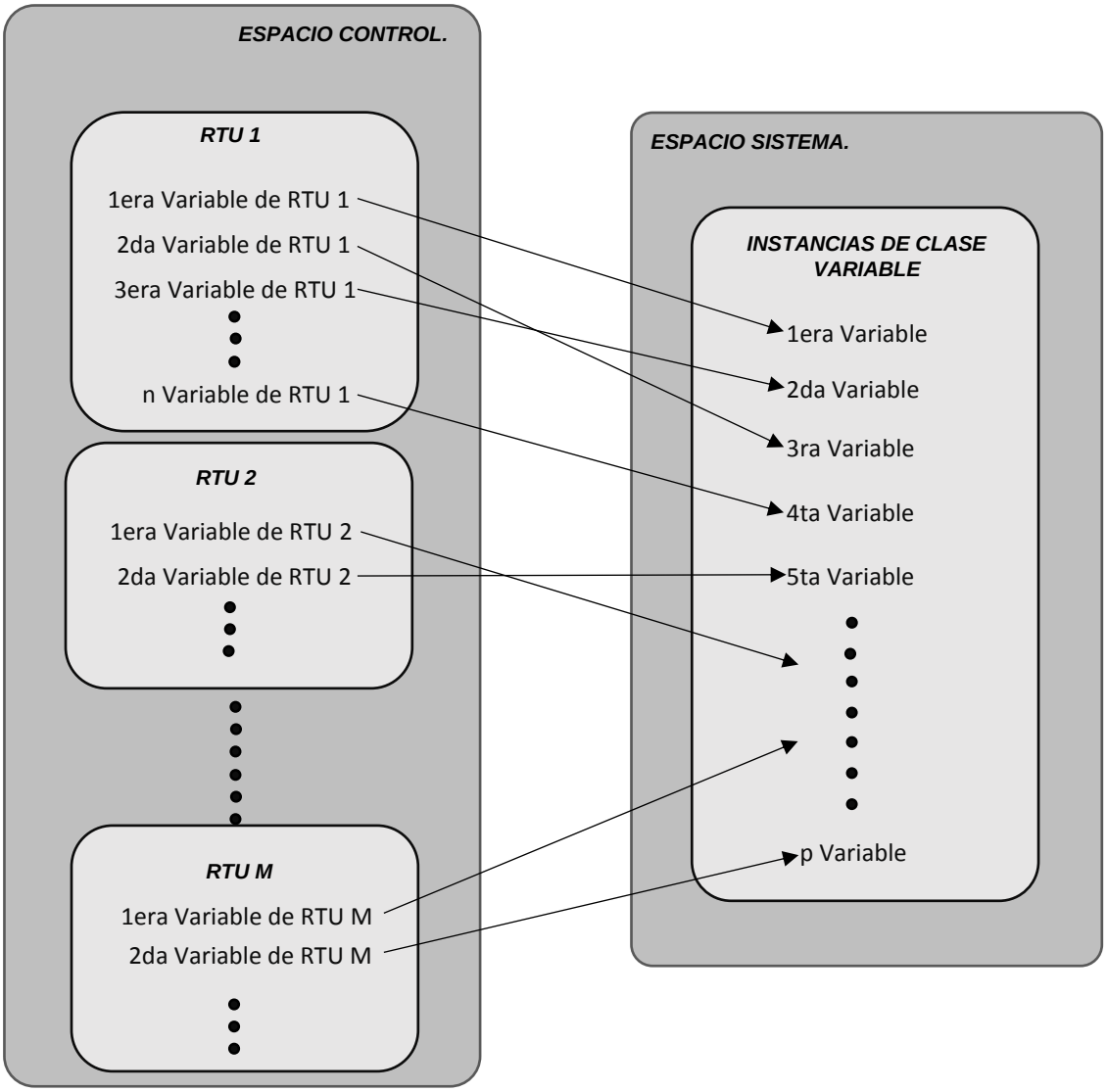


Fig. 3.33. Vínculos que se establecen entre los objetos del Espacio Control con los objetos (o instancias) del Espacio Sistema [***Fuente**: Propia*].

e) Ingreso del esquema de la red al Espacio Control

Se refiere al ingreso de los elementos analizados en los subcapítulos anteriores, al sistema SCADA, tales como:

- **RTUs.** La cantidad de estos se conoce antes de la implementación del SCADA.

- **Gateways.** Por cada tipo de protocolo tiene que haber por lo menos 1 Gateway, no puede haber un Gateway usando más de un protocolo.

  También se había visto que una RTU puede tener asignado su propio Gateway en caso de estar muy alejado de otras RTUs, o en caso de tener muchas variables por extraer.
- **Variables o registros a extraer de las RTUs.** Establecido por los operadores de la empresa porque son ellos son los que saben que variables son necesarias monitorear para un buen manejo de la planta.

Se debe tener presente que un Gateway solo puede estar conectado a RTUs de una misma planta.

Entonces, se procede con el ingreso de los elementos mencionados líneas arriba dentro del sistema SCADA que se ha implementado en la Tesis. Previamente, se procederá a mencionar los lineamientos sobre los cuales se desarrollará el siguiente ejemplo:

- Las instancias del ESPACIO SISTEMA son las mismas que las mencionadas en Fig. 3.15., es decir, se trabajará con las 3 plantas mencionadas (CX_15, CX_11 y AX_08), y sus respectivos PROCESOS y VARIABLES.
- Todas las RTUs de las 3 plantas tienen implementados el protocolo Modbus-RTU, por lo que no se trabajará con otros protocolos en la Tesis.

En Fig. 3.34, Fig. 3.35 y Fig. 3.36, se muestran las tres plantas, CX_15, CX_11 y AX_08 con sus respectivos Gateways, y estos a su vez, conectados con sus respectivas RTUs. Además, se muestran las variables o registros que serán extraídas de cada RTU. Por último, se muestra la asociación entre las variables de las RTUs con las respectivas variables o instancias de la CLASE VARIABLE del ESPACIO SISTEMAS.

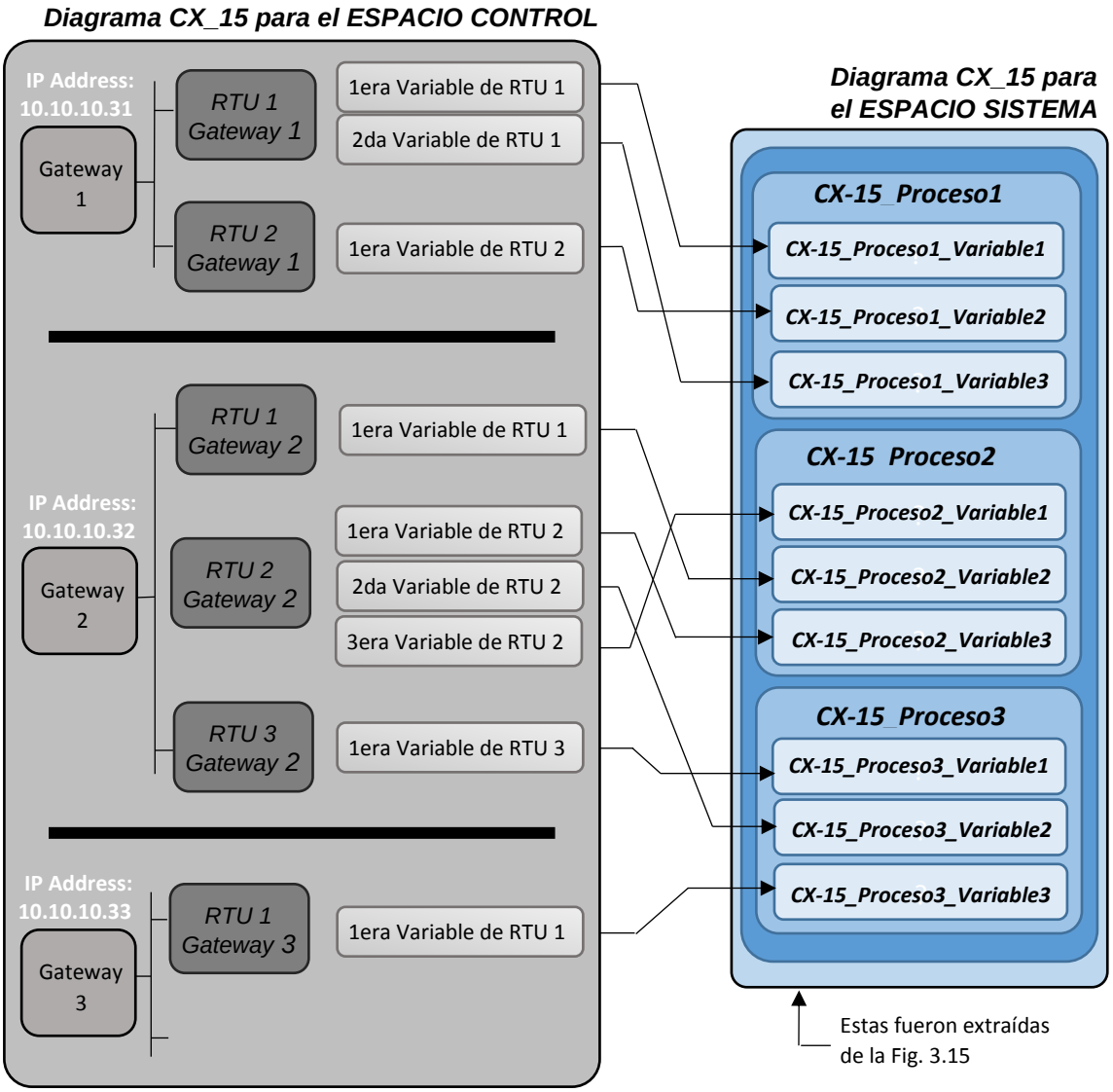


Fig. 3.34. Vínculos entre los objetos del Espacio Control con los objetos o instancias del Espacio Sistema, aplicado a la planta CX_15. [***Fuente**: Propia*]

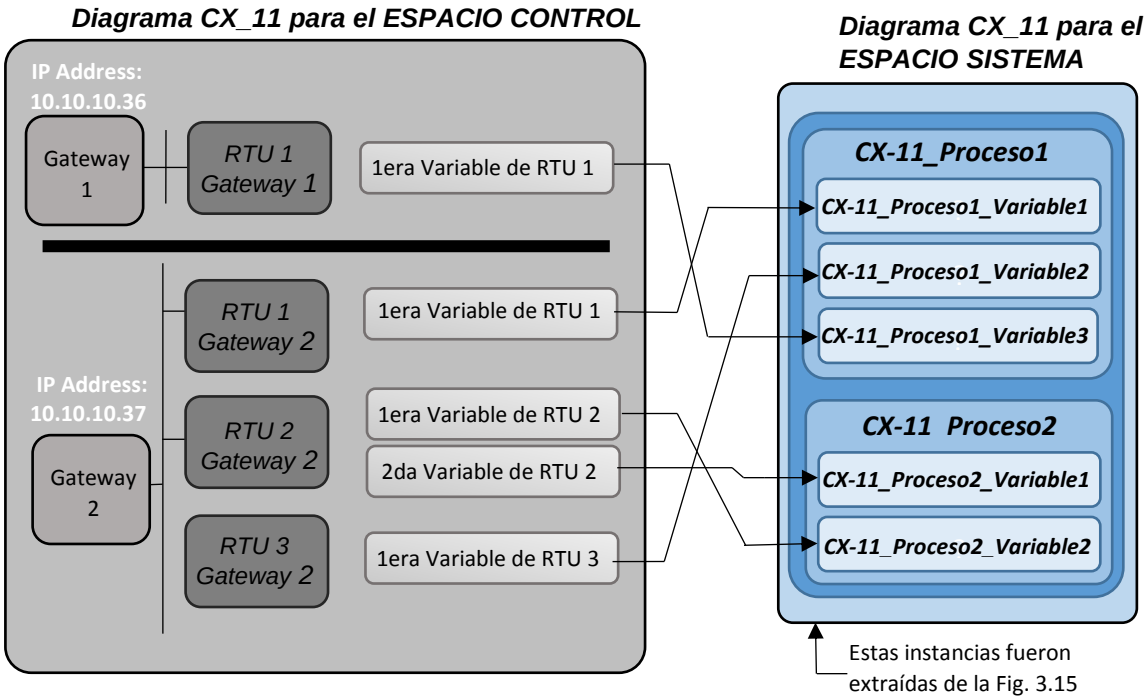


Fig. 3.35. Vínculos entre los objetos del Espacio Control con los objetos o instancias del Espacio Sistema, aplicado a la planta CX_11. [***Fuente**: Propia*]

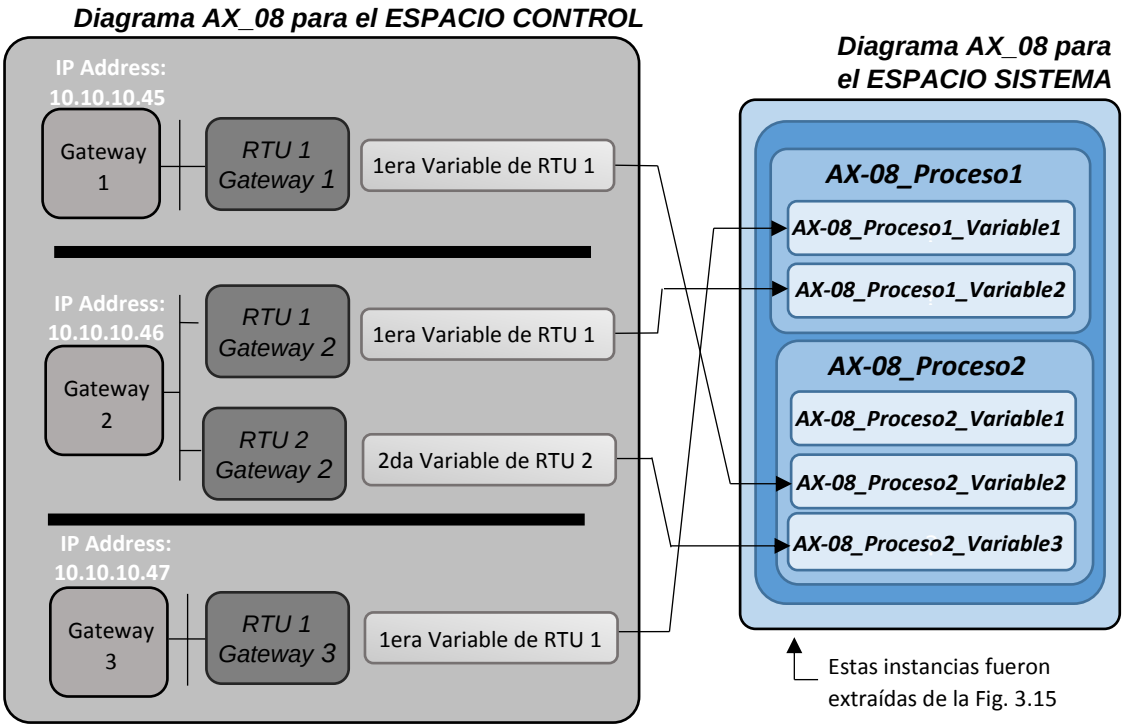


Fig. 3.36. Vínculos entre los objetos del Espacio Control con los objetos o instancias del Espacio Sistema, aplicado a la planta AX_08. [***Fuente**: Propia*]

Anteriormente se mencionó que se trabajaría con RTUs que tengan implementado el protocolo Modbus-RTU, pero aún no se ha mencionado cuáles serán los dispositivos que conforman el Gateway.

Antes que nada, se tiene que mencionar que el Gateway no está condicionado a una marca o modelo específico de dispositivo electrónico, lo único que tiene que tener implementado (ya sea porque usted mismo lo implemento o porque ya vino implementado de fábrica) es el protocolo de la RTU (en este proyecto de Tesis será Modbus-RTU) y el protocolo HTTP.

Como se puede observar en la Fig. 3.37., el Gateway está conformado por 2 dispositivos, el microcontrolador PIC con el protocolo Modbus-RTU implementado (Microchip – 2003 – "PIC16F87XA Data Sheet 28/40/44-Pin Enhanced Flash Microcontrollers") [12] y el microprocesador SITE PLAYER implementado con el protocolo HTTP (Site Player - 2003 – "Development System Rev 0310A") [13].

El código fuente de los dispositivos que integran el Gateway se muestra en detalle en el Anexo A.

Las especificaciones técnicas de los dispositivos que integran el Gateway se mostrarán en detalle en el Anexo B.

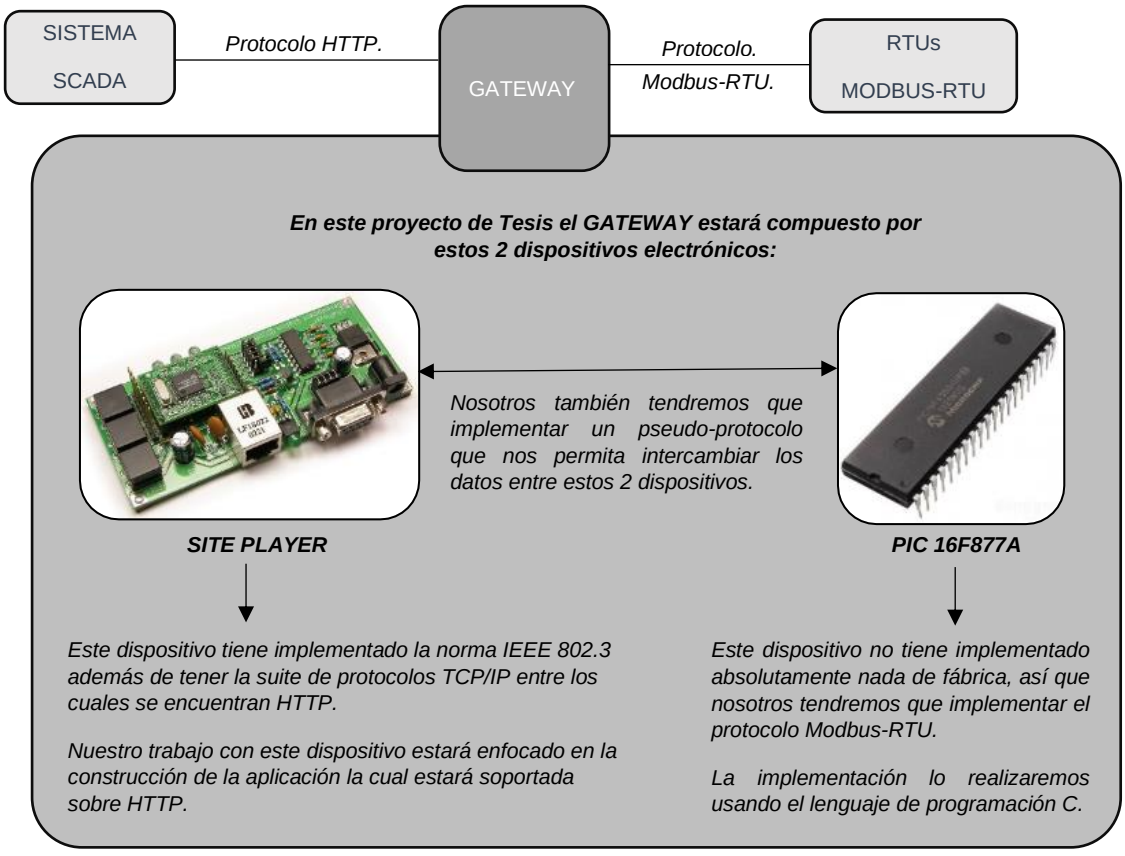


Fig. 3.37. Componentes internos del Gateway. [***Fuente**: Propia*]

A continuación, se muestra cómo es que se ingresaran los componentes que conforman la red, el cual se basara en los datos de las Fig. 3.34., Fig. 3.35. y Fig. 3.36.

Fig. 3.38. y Fig. 3.39. muestran las pantallas que permitirán acceder a los formularios con los cuales se enlistara los componentes de la red.

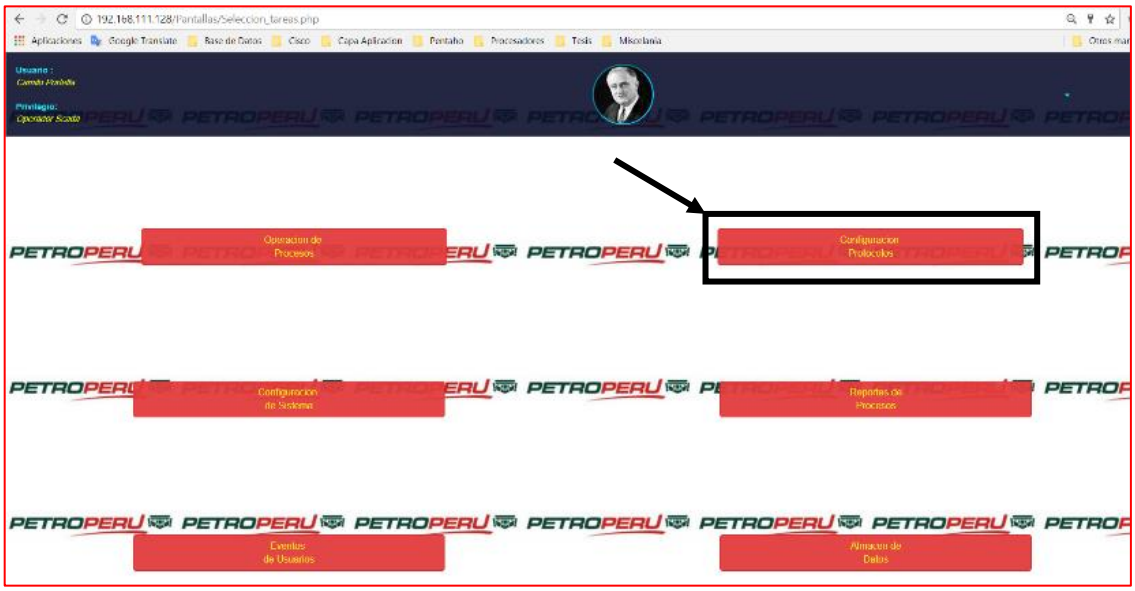


Fig. 3.38. Acceso a los servicios que proporciona el Espacio Control [***Fuente**: Propia*]

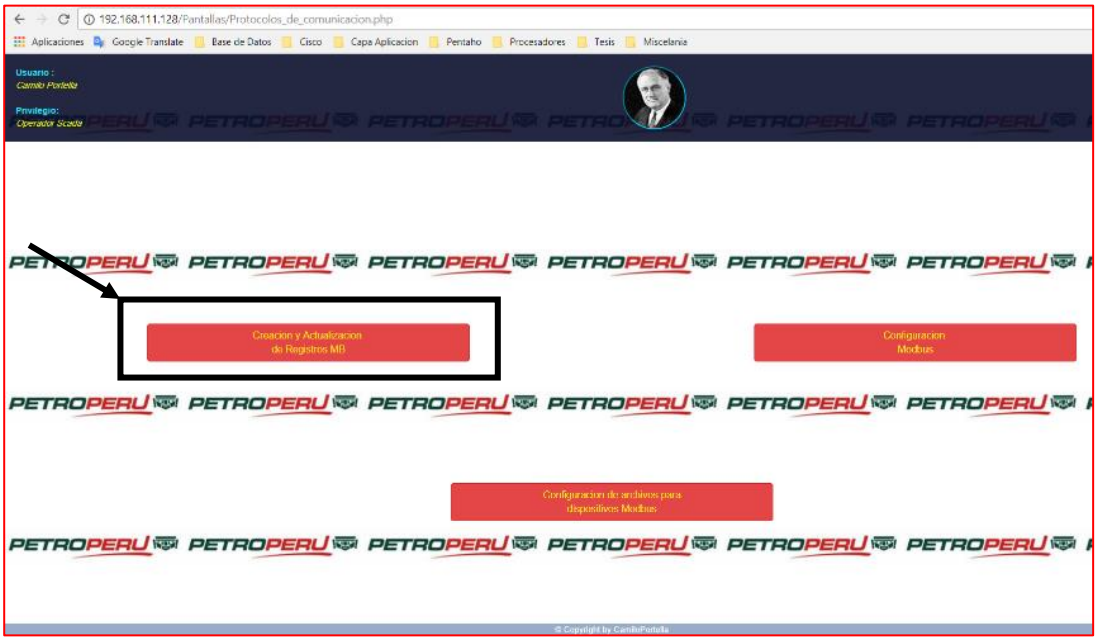


Fig. 3.39. Creación de lista de los Gateways, los RTU y los registros de cada RTU. [***Fuente**: Propia*]

En la Tesis, los Gateways tendrán los siguientes atributos como se puede ver en el formulario de la Fig. 3.40.:

- **Planta**. Se refiere a la planta (o, dicho en otras palabras, instancia de la CLASE PLANTA del ESPACIO SISTEMA) donde estará ubicado el Gateway.
- **Id de MB_Master.** Se carga automáticamente, y sirve para identificar a un Gateway de forma exclusiva, es decir, que no puede haber otro Gateway con ese mismo identificador en toda la empresa.
- **Nombre de MB_Master.** Se carga automáticamente y sirve para identificar a un Gateway dentro de una misma planta.
- **IP de MB_Master.** Es la dirección IPv4 que usará el sistema SCADA para comunicarse con los Gateways.
- **Descripción de MB_Master**. El usuario puede colocar cualquier descripción que considere conveniente.
- **Habilitado.** Por el momento no tiene ninguna función.

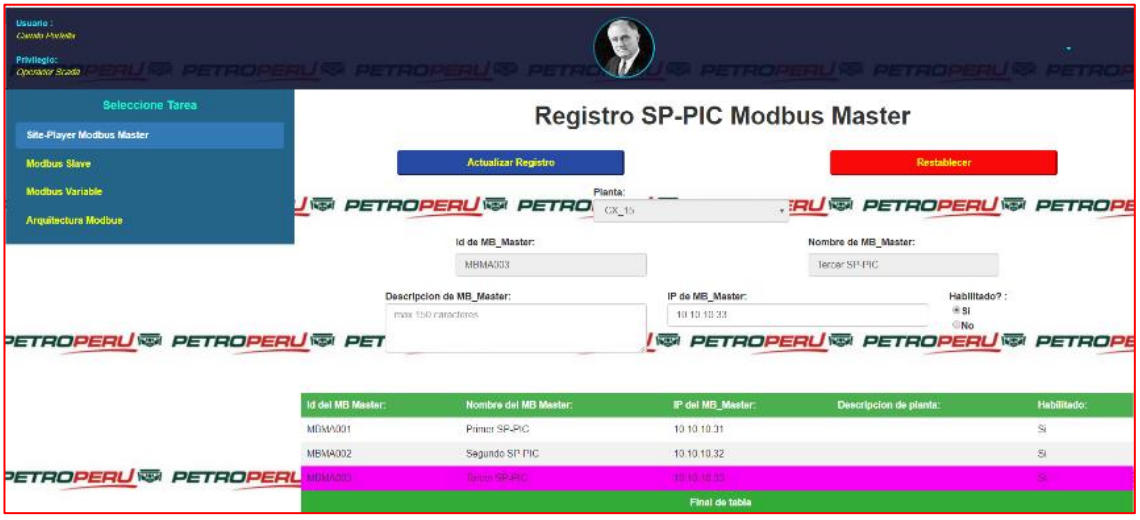


Fig. 3.40. Formulario para el ingreso y actualización de los Gateways. [***Fuente**: Propia*]

En la Tesis, las RTUs tendrán los siguientes atributos como se puede observar en el formulario de la Fig. 3.41.:

- **Planta.** Se refiere a la planta o instancia de la CLASE PLANTA del ESPACIO SISTEMA, donde estará la RTU.
- **Nombre de MB_Master.** Sirve para indicar a qué Gateway pertenece la RTU que se quiere agregar o actualizar.
- **Id de MB Slave.** Se carga automáticamente y sirve para identificar a una RTU de forma exclusiva, es decir, en toda la empresa no podrá haber otra RTU con ese mismo identificador.

- **Nombre de la MB Slave.** Se carga automáticamente y sirve para identificar a una RTU dentro de su correspondiente Gateway.
- **Descripción.** El usuario puede colocar cualquier descripción que considere conveniente para la RTU que se va agregar o actualizar.
- **Habilitado**. Por el momento no tiene ninguna función.

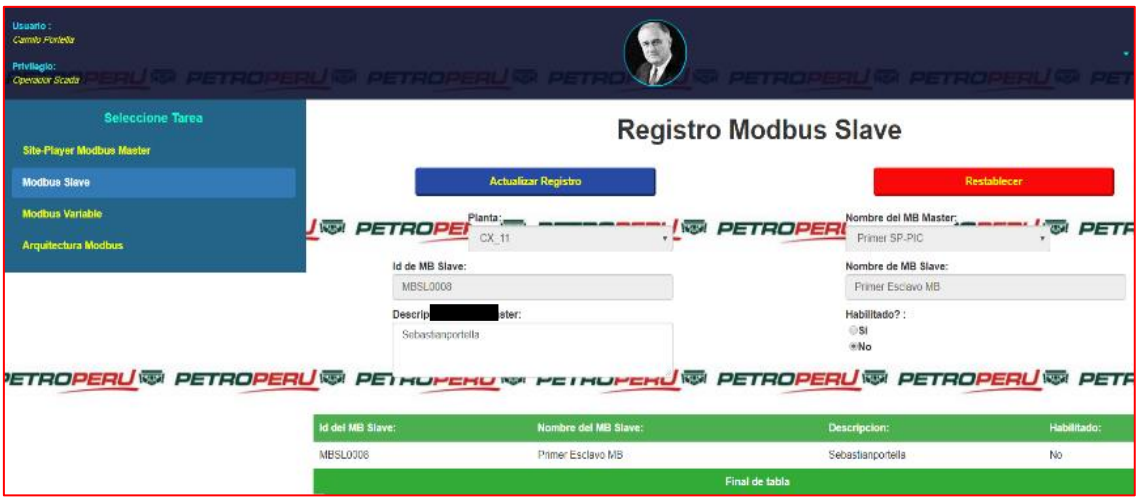


Fig. 3.41. Formulario que permite el ingreso y actualización de las RTUs.
[***Fuente**: Propia*]

En la Tesis, los registros de cada RTU tendrán los siguientes atributos como se puede ver en el formulario de la Fig. 3.42.:

- **Planta.** Se refiere a la planta o la instancia de la CLASE PLANTA del ESPACIO SISTEMA donde estará el registro.
- **Nombre de MB_Master.** En función de la planta seleccionada anteriormente, se mostrará una lista de Gateways, se tiene que seleccionar uno de ellos.
- **Nombre del MB_Slave.** En función del Gateway seleccionado anteriormente, se mostrará una lista de RTUs, se tiene que seleccionar uno de ellos.
- **Id de MB Variable.** Se carga automáticamente, sirve para identificar de forma exclusiva a un registro o variable de una RTU. En toda la empresa no podrá haber otro registro con ese mismo identificador.
- **Nombre de MB Variable.** Se carga automáticamente y sirve para identificar un registro o variable dentro de un RTU.
- **Proceso.** En función a la planta seleccionada anteriormente, se mostrará una lista de Procesos o instancias de la CLASE PROCESOS del ESPACIO SISTEMA. Debe ser seleccionado uno de ellos.
- **Variable.** En función al Proceso seleccionado anteriormente, se mostrará una lista de instancias de la CLASE VARIABLE del ESPACIO SISTEMA. Se tiene que

seleccionar cuál de esas instancias será vinculada con el registro que se acaba de crear.

- **Descripción.** El usuario puede colocar cualquier descripción que considere conveniente para el registro que se va agregar o actualizar.
- **Habilitado.** Por el momento, no tiene ninguna función.

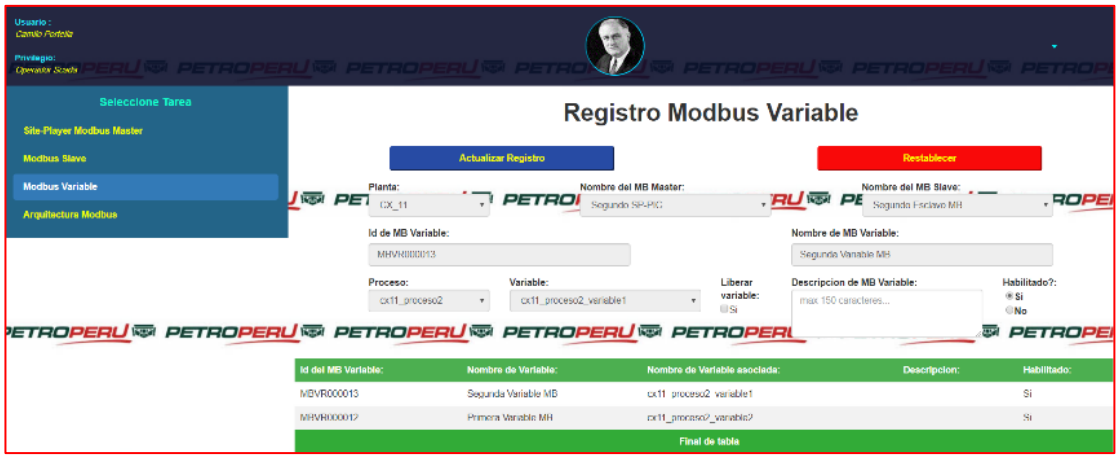
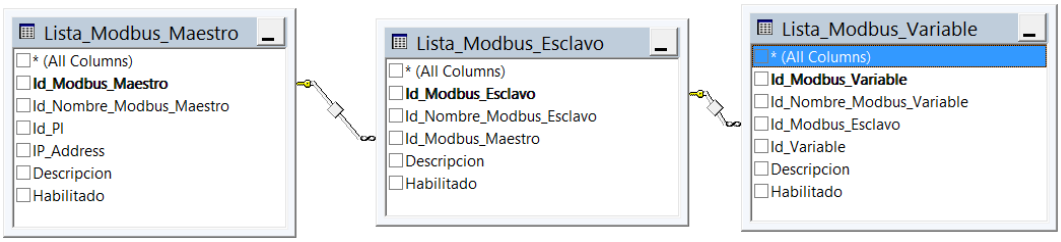


Fig. 3.42. Ingreso y actualización de un registro o variable de una RTU.
[***Fuente**: Propia*]

La Fig. 3.43., muestra 3 tablas de la Base de Datos, una tabla corresponde a los Gateways, otra a las RTUs y la última corresponde a los registros de las RTUs. Es importante mencionar que estas 3 tablas solo están asociadas al protocolo Modbus-RTU, que es el único protocolo que se está usando en la Tesis.

Si más adelante se agregan otros protocolos de comunicación como Modbus-TCP o CIP, entonces correspondería agregar otras 3 tablas por cada protocolo, una de Gateways, otra tabla de RTUs u otra tabla para los registros, y se tendrá que crear otros formularios para el ingreso de los mismos.



Tablas de Gateway. *Tablas de RTUs.* *Tablas de registros de RTU.*

Fig. 3.43. Tablas correspondientes a los Gateway, RTUs y registros de las RTUs.
[***Fuente**: Propia*]

Las Fig. 3.44., Fig. 3.45. y Fig. 3.46., muestran en la Base de Datos todos los Gateways, RTUs y registros de las RTUs, basados en las listas de Fig. 3.34., Fig. 3.35. y Fig. 3.36.

	Id_Modbus_Maestro	Id_Nombre_Modbus_Maestro	Id_PI	IP_Address	Descripcion	Habilita...
1	MBMA001	1	PL001	10.10.10.31		Si
2	MBMA002	2	PL001	10.10.10.32		Si
3	MBMA003	3	PL001	10.10.10.33		Si
4	MBMA004	1	PL002	10.10.10.36	MBMA004	Si
5	MBMA005	2	PL002	10.10.10.37	Segundo SP-PIC	Si
6	MBMA006	1	PL003	10.10.10.45	MBMA006	Si
7	MBMA007	2	PL003	10.10.10.46	MBMA007	Si
8	MBMA008	3	PL003	10.10.10.47		Si

Fig. 3.44. Gateway ingresados a través del formulario de la Fig. 3.40. [***Fuente**: Propia*]

	Id_Modbus_Esclavo	Id_Nombre_Modbus_Esclavo	Id_Modbus_Maestro	Descripcion	Habilitado
1	MBSL0001	1	MBMA001		Si
2	MBSL0002	2	MBMA001		Si
3	MBSL0003	1	MBMA002		Si
4	MBSL0004	2	MBMA002		Si
5	MBSL0005	3	MBMA002		Si
6	MBSL0006	1	MBMA003		Si
7	MBSL0007	2	MBMA003		Si
8	MBSL0008	1	MBMA004	Sebastianportella	No
9	MBSL0009	1	MBMA005		Si
10	MBSL0010	2	MBMA005		Si
11	MBSL0011	3	MBMA005		Si
12	MBSL0012	1	MBMA006		Si
13	MBSL0013	1	MBMA007		Si
14	MBSL0014	2	MBMA007		Si
15	MBSL0015	1	MBMA008		Si

Fig. 3.45. RTUs ingresadas a través del formulario de la Fig. 3.41. [***Fuente**: Propia*]

	Id_Modbus_Variable	Id_Nombre_Modbus_Variable	Id_Modbus_Esclavo	Id_Variable	Descripcion	Habilitado
1	MBVR000001	1	MBSL0001	VR000001		Si
2	MBVR000003	1	MBSL0002	VR000002		Si
3	MBVR000002	2	MBSL0001	VR000003	Sebastianportella	No
4	MBVR000007	3	MBSL0004	VR000004		Si
5	MBVR000004	1	MBSL0003	VR000005		Si
6	MBVR000005	1	MBSL0004	VR000006		Si
7	MBVR000008	1	MBSL0005	VR000007		Si
8	MBVR000006	2	MBSL0004	VR000008		Si
9	MBVR000009	1	MBSL0006	VR000009		Si
10	MBVR000011	1	MBSL0009	VR000010		Si
11	MBVR000014	1	MBSL0011	VR000011		Si
12	MBVR000010	1	MBSL0008	VR000012		Si
13	MBVR000013	2	MBSL0010	VR000013		Si
14	MBVR000012	1	MBSL0010	VR000014		Si
15	MBVR000018	1	MBSL0015	VR000015		Si
16	MBVR000016	1	MBSL0013	VR000016		Si
17	MBVR000015	1	MBSL0012	VR000018		Si
18	MBVR000017	1	MBSL0014	VR000019		Si

Fig. 3.46. Registros de las RTUs ingresadas a través del formulario de la Fig. 3.42. [***Fuente**: Propia*]

f) Ingreso del esquema de las redes a los Gateways

En la subsección anterior, se ingresó al sistema SCADA información sobre cuántos elementos conformaban la red (Gateways, RTUs y registros de cada RTUs), pero ahora se realizará el análisis de la información que debe ser ingresada al Gateway.

Cuando un sistema SCADA envía un mensaje cuyo destino final es un Gateway, lo único que tiene que hacer el sistema SCADA es indicar la dirección IP del Gateway, para que de esta forma se pueda acceder al Gateway correcto. Recuérdese que el protocolo de comunicación que se establece entre el sistema SCADA y los Gateways es HTTP, ver Fig. 3.47.

Los mensajes que el sistema SCADA envía al Gateway, usualmente tienen como destino final a un registro de una RTU, pero como ya se mencionó anteriormente, este mensaje tiene que transitar por el Gateway. Recuérdese que el sistema SCADA no tiene conexión directa con las RTUs.

La tarea del Gateway, ver Fig. 3.47., es la de "acceder a un registro de la RTU, el cual le fue encomendado por el sistema SCADA".

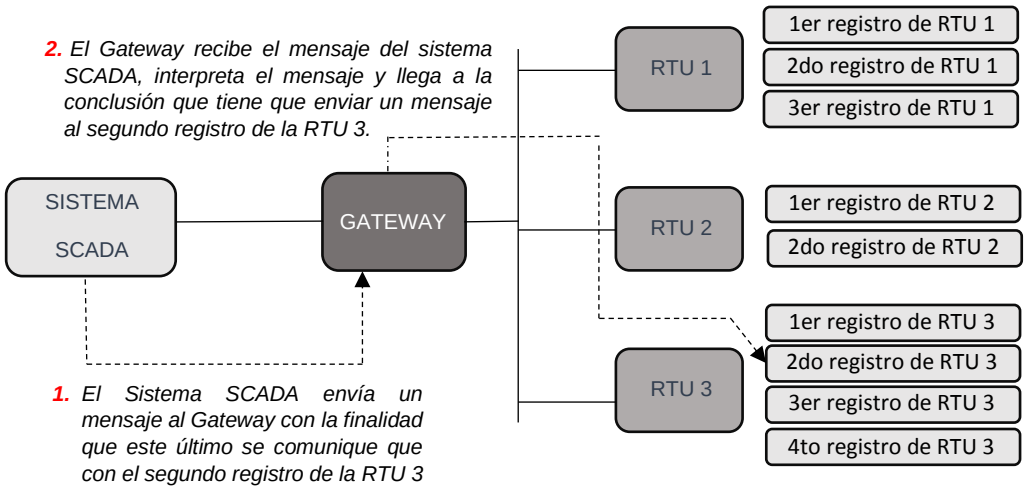


Fig. 3.47. Tarea realizada por el Gateway: Acceder al registro de la RTU encomendado por el sistema SCADA. [***Fuente**: Propia*]

Para lograr que el Gateway pueda interpretar los mensajes que le envía el sistema SCADA, y poder deducir a qué RTU y a qué registro se le está encomendando acceder, lo que se hará es ingresar el esquema de red dentro de la programación del Gateway mediante la creación de un archivo.

Para lograr obtener tal archivo, que le permita al Gateway interpretar los mensajes del sistema SCADA sin errores, se realizara lo siguiente:

- A partir de la Base de Datos, se obtendrá los elementos que componen la red (RTUs y sus registros).

- Conjuntamente con el programa PHP se obtendrá de forma automatizada el archivo para que sea posteriormente descargado en el Gateway. En la Fig. 3.48., se muestra de forma gráfica lo explicado líneas arriba.

En la Fig. 3.48. se muestra el proceso para la obtención del archivo que será descargado en el Gateway, este procedimiento aplica para cualquier protocolo que tenga implementado la RTU (Protocolo-X). Se mostrará el proceso para un solo Gateway y en caso de haber más Gateways el proceso se repite.

No olvidar, que el objetivo es poder llegar desde el sistema SCADA hacia los registros de la RTU. Se analizará los avances conseguidos y qué cosas aún faltan para lograr tal objetivo, para ello se trabajará con un ejemplo sencillo como el que se muestra en la Fig. 3.49. Se asume que el Gateway ya tiene su archivo descargado. En el ejemplo muestra el análisis para un solo Gateway, pero si hubiera más Gateways el análisis sería el mismo.

Es importante mencionar que el documento que se descarga sobre un Gateway se debe actualizar, es decir, se debe realizar una nueva descarga cada vez que ocurre lo siguiente:

- Cuando se agrega un nuevo RTU a la red del Gateway.
- Cuando se desea incrementar la cantidad de registros que se extraen de una RTU correspondiente a un Gateway.

Hasta el momento todo lo explicado aplica para cualquier protocolo de comunicación que esté implementado en la RTU y por consiguiente, en su respectivo Gateway. En la Tesis se hace referencia a un “Protocolo-X” para citar un protocolo cualquiera, ya que en el mercado no existe ningún protocolo de comunicación con ese nombre y de esta forma se evita que el lector piense que lo explicado está restringido a un solo protocolo.

Ahora se aplicará todo lo explicado anteriormente para la red que se mostró en la Fig. 3.34., Fig. 3.35. y Fig. 3.36. Además, se considerará en la tesis, que el Gateway está compuesto por un micro-servidor SitePlayer y un microcontrolador PIC, como se muestra en la Fig. 3.37.

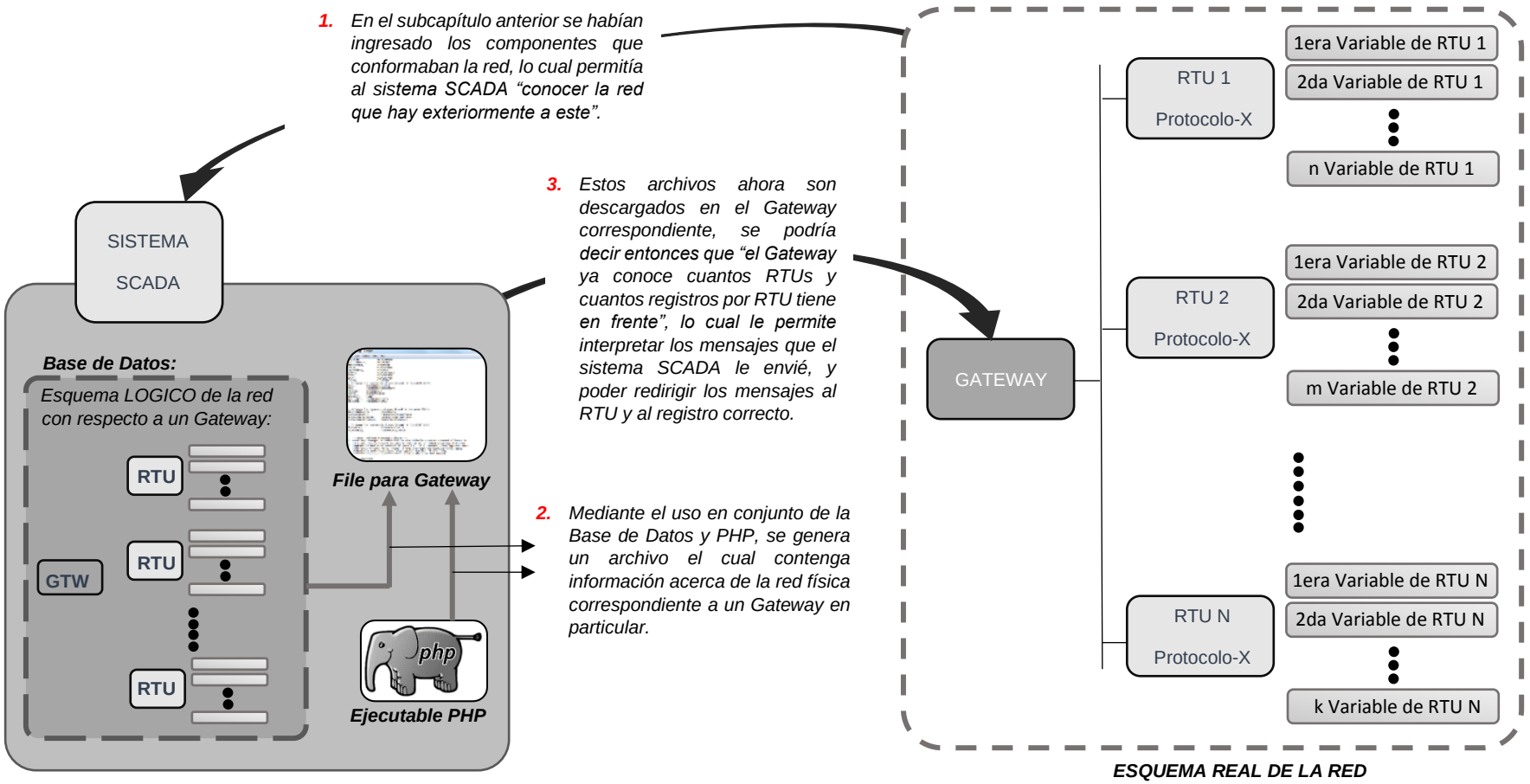


Fig. 3.48. Proceso para el ingreso de información relacionada con la red dentro de un determinado Gateway. [***Fuente**: Propia*]

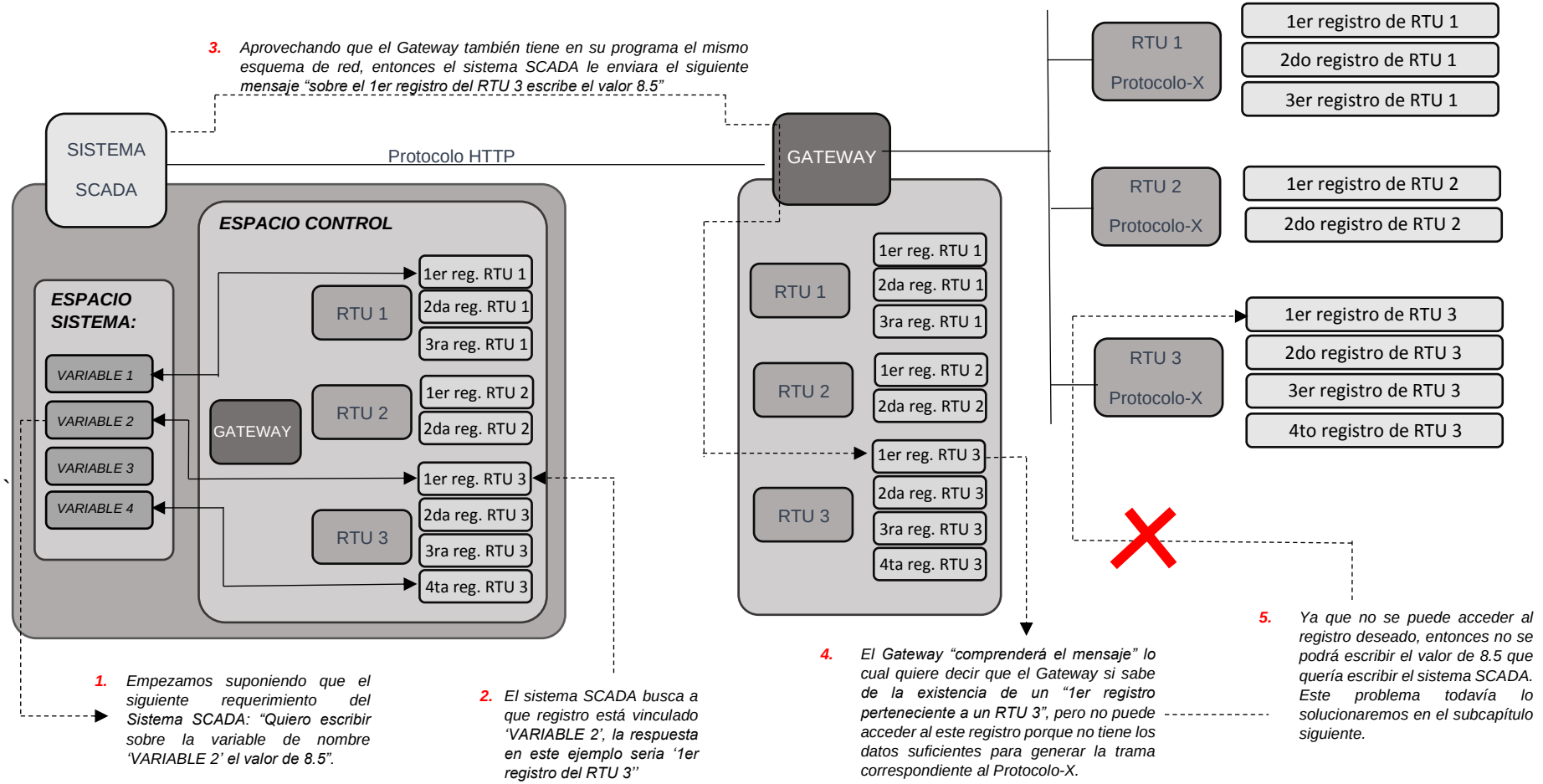


Fig. 3.49. Análisis del flujo de un mensaje generado por el sistema SCADA. [***Fuente**: Propia*]

Desde la Fig. 3.50. hasta la Fig. 3.53., se muestra cómo se aplicó en la Tesis lo explicado en este subcapítulo, en relación a Modbus-RTU.

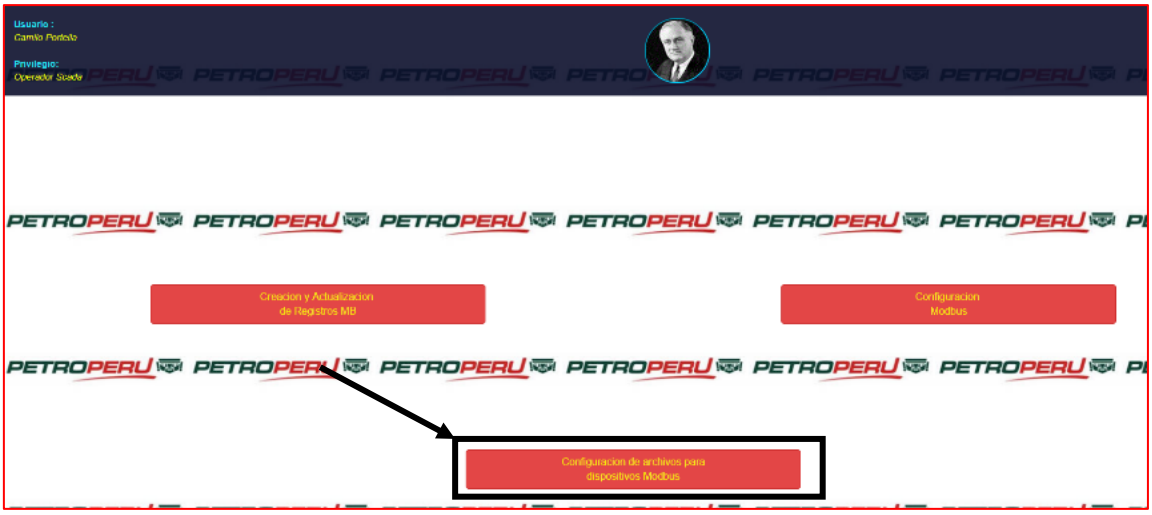


Fig. 3.50. Acceso a la página que genera los files necesarios para el Gateway [***Fuente**: Propia*]

La Fig. 3.51., permite observar cómo se generan los archivos que se descargaran sobre el Site Player y el PIC.

En la Fig. 3.52., se muestra de forma más detallada el procedimiento para descargar los archivos correspondientes al SitePlayer desde el sistema SCADA.

En la Fig. 3.53., se muestra de forma más detallada el procedimiento para descargar el archivo correspondiente a un PIC desde el sistema SCADA.

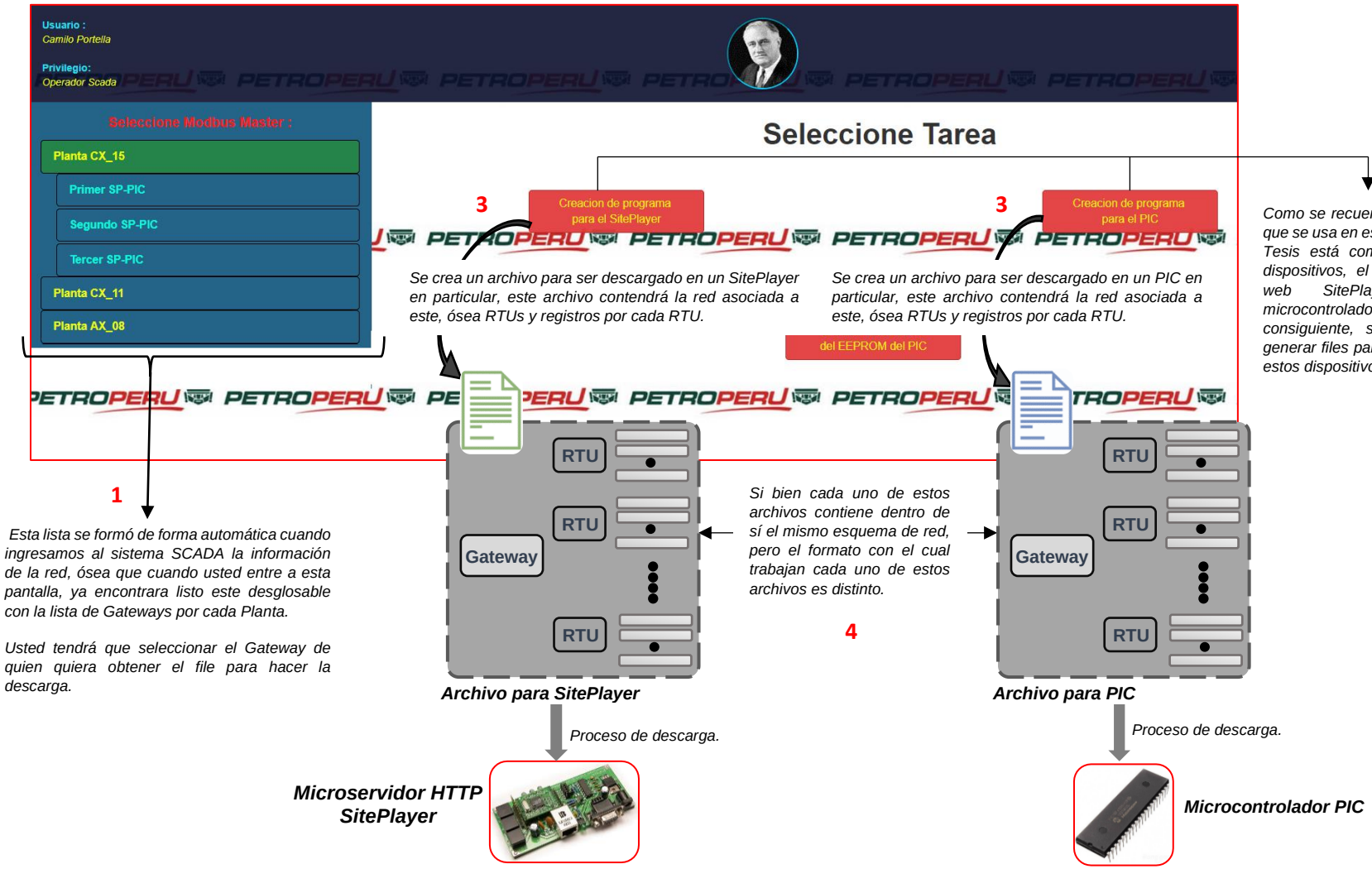


Fig. 3.51. Pantalla del SCADA que permite la generación de archivos para el SitePlayer y PIC. [***Fuente**: Propia*]

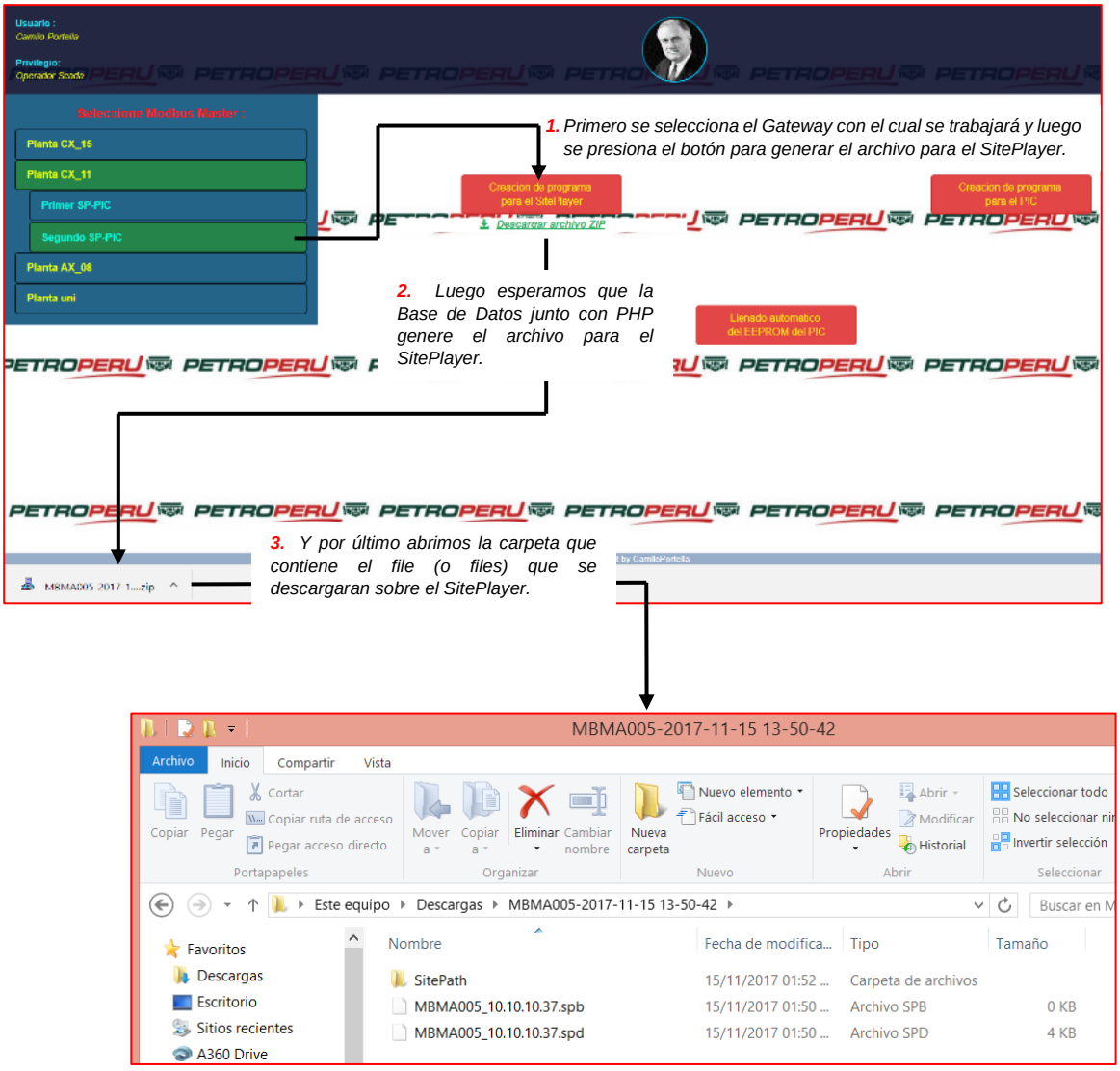


Fig. 3.52. Pasos para obtener los archivos para un SitePlayer desde la interface del sistema SCADA. [***Fuente**: Propia*]

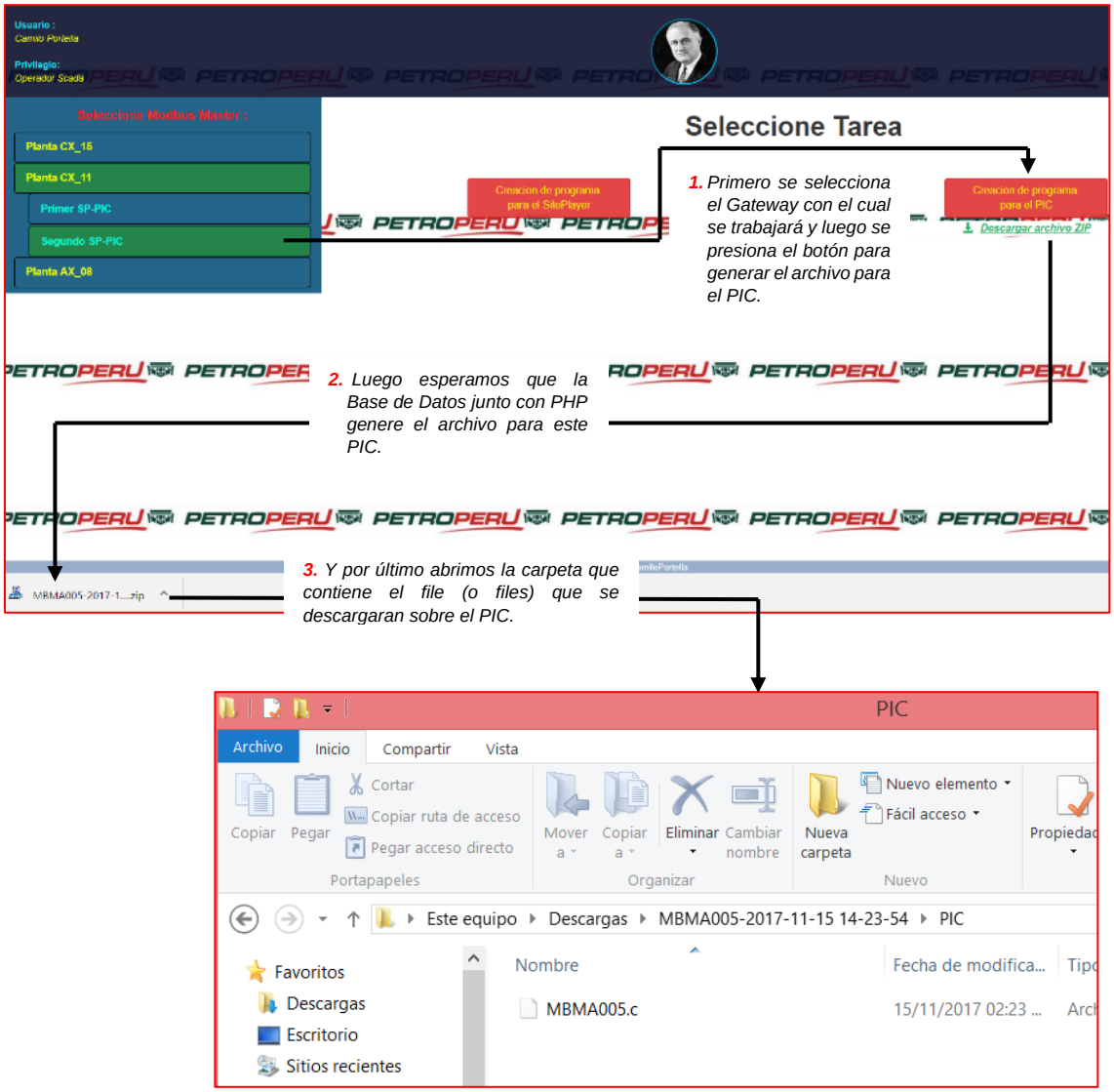


Fig. 3.53. Pasos para obtener los archivos para el PIC desde la interface del sistema SCADA. [***Fuente**: Propia*]

g) Generación de comandos desde el Gateway hacia las RTUs y desde el sistema SCADA al Gateway

En las subsecciones anteriores se consiguieron 2 objetivos muy importantes:

- El sistema SCADA tiene conocimiento de cuantos elementos integran la red (Gateways, RTUs y registros por cada RTU).
- Cada uno de los Gateways tiene conocimiento de cuantos elementos integran su respectiva red (RTUs y registros por cada RTU).

Lo mencionado líneas arriba es muy importante, porque existe una especie de "sincronización" en cuanto al esquema de red que tiene almacenado el sistema SCADA y los Gateways. Pero si se revisa el análisis que se hizo en la Fig. 3.49., se podrá observar que el Gateway aún no puede comunicarse con la RTU, porque para ello el Gateway tiene que hacer envió de comandos acorde al "Protocolo-X" (comandos que son los mismos que se mencionaron en la Fig.3.29) y para ello necesita datos correspondientes al Protocolo-X.

Téngase presente que cuando se hizo referencia al "Protocolo-X" se hacía referencia a cualquier protocolo en general, con la finalidad de que el lector no piense que el análisis está restringido a un protocolo en particular. El "Protocolo-X", que se está usando, es para representar el protocolo que usan el Gateway y sus RTUs para comunicarse.

También, se debe tener presente que para la comunicación entre el sistema SCADA y los Gateways es el protocolo HTTP (ver Fig. 3.27.).

Comunicación entre un Gateway y una RTU.

En primer lugar, el tipo de comunicación que se establece entre estos 2 dispositivos siempre depende del tipo de protocolo que tenga implementado la RTU (en este caso el "Protocolo-X"), y lo que se tiene que hacer es implementar el "Protocolo-X" en el Gateway, para que de esta forma el Gateway sea compatible con la RTU (o RTUs).

Existen protocolos que son ABIERTOS, lo cual significa que el documento que contiene sus especificaciones está disponible para el público en general, mientras otros protocolos son CERRADOS, que significa que la empresa que desarrolla el protocolo, mantiene las especificaciones de este en reserva.

Es importante mencionar que el Gateway siempre será el que inicie la comunicación con la RTU y nunca será la RTU la que tenga la iniciativa. La RTU simplemente se limitará a responder las solicitudes del Gateway. Por lo que al Gateway se le conoce como MASTER o CLIENTE y a la RTU se le conoce como SLAVE o CLIENTE.

En la Fig. 3.54., se puede observar la **composición básica** de una trama para que el Gateway pueda acceder para leer o escribir un dato en un registro específico de una RTU. Se supondrá que se está trabajando con el protocolo genérico de nombre "Protocolo-X".

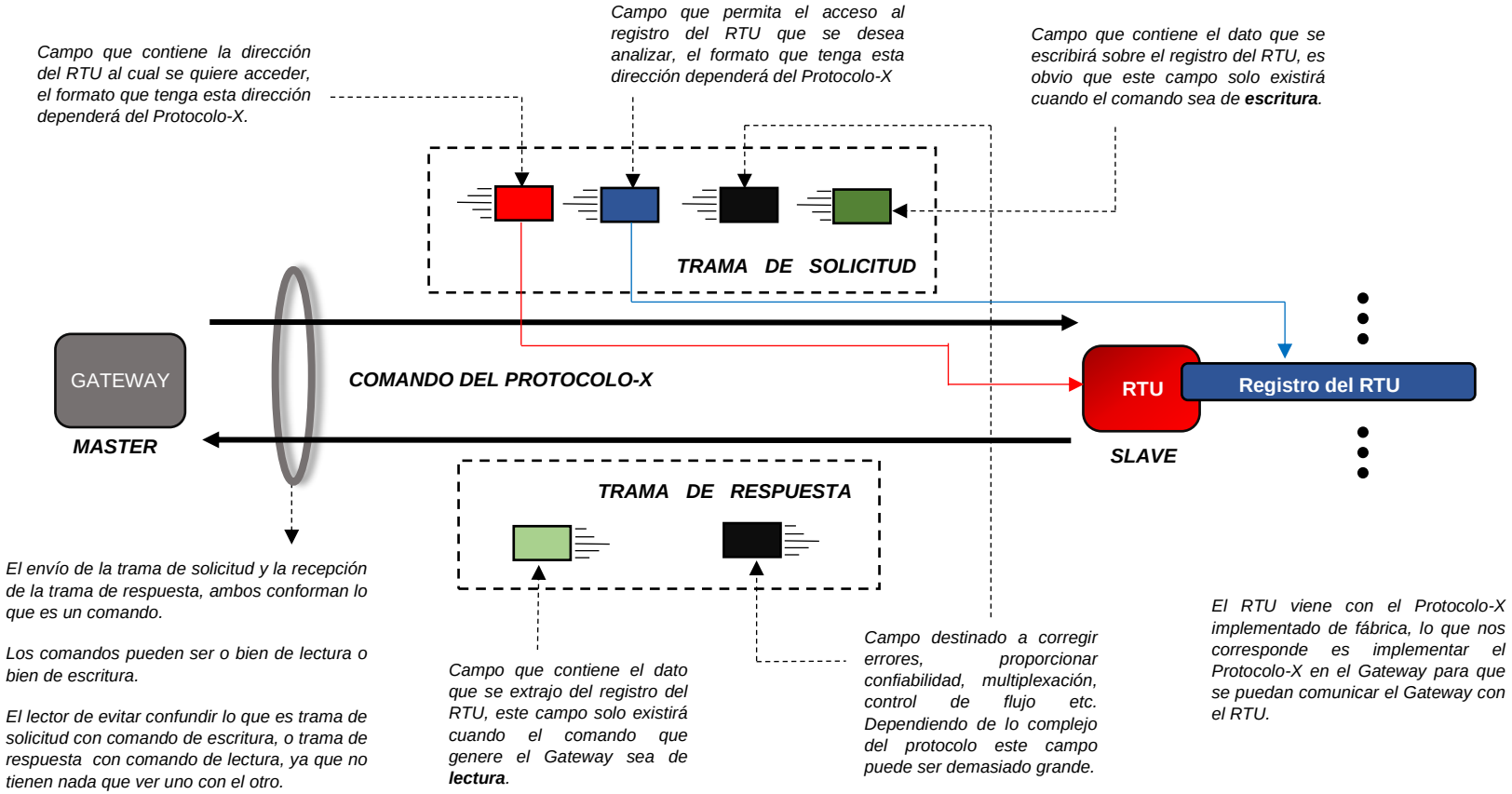


Fig. 3.54. La comunicación entre el Gateway y una RTU. [***Fuente**: Propia*]

Si se revisa la Fig. 3.54., se puede observar que para que los comandos se puedan construir, al margen de si el comando es de lectura o de escritura, es necesario que el Gateway disponga de lo siguiente:

- Información sobre el direccionamiento de la RTU, representado con el campo de color rojo.
- Información sobre el direccionamiento de cada uno de los registros de la RTU, representado con el campo de color azul.

Hasta el momento el Gateway sabe cuántas RTUs y cuántos registros por RTU tiene, pero no tiene información sobre el direccionamiento de las RTUs ni de los registros de estas RTUs.

El primer paso a realizar será, separar un espacio de memoria en el Gateway y destinarlo para el almacenamiento de las direcciones de las RTUs y de sus respectivos registros. Para almacenar los direccionamientos de las RTUs y sus registros se usa la misma disposición que se usó en el esquema de red, para ello se mostrará un ejemplo sencillo en la Fig. 3.55.

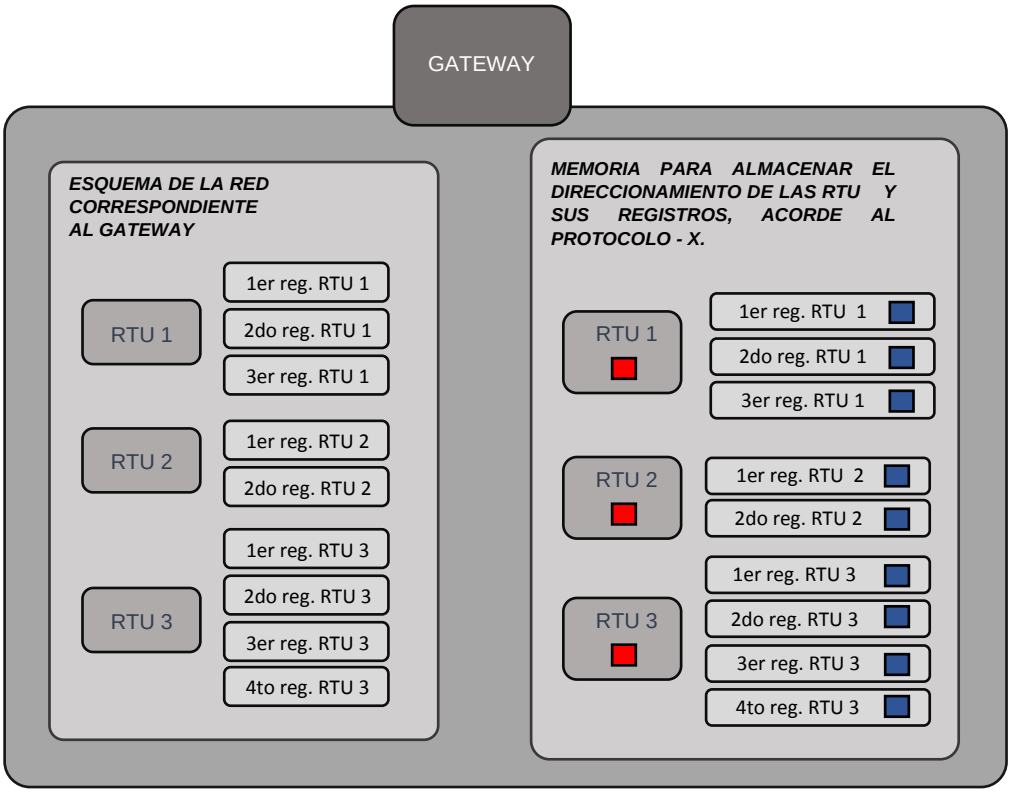


Fig. 3.55. Creación de un espacio de memoria en el Gateway (cuadro de la derecha), siguiendo el mismo esquema de la red correspondiente al Gateway (cuadro izquierdo). [***Fuente**: Propia*]

Proceso de llenado de la memoria del Gateway en donde se almacena el direccionamiento de las RTUs y sus respectivos registros

En la Fig. 3.56., se puede observar un ejemplo que detalla el flujo de la información relacionada con la dirección de una RTU desde su origen en el sistema SCADA hasta su almacenamiento en el Gateway.

En la Fig. 3.57., se puede observar un ejemplo que detalla el flujo de la información relacionada con la dirección de los registros de una RTU, desde su origen en el sistema SCADA hasta su almacenamiento en el Gateway.

Proceso de escritura sobre los registros de las RTUs:

Si se revisa el ejemplo de la Fig. 3.49., se podrá observar que el proceso de escritura sobre uno de los registros de la RTU no se pudo completar, porque el Gateway en ese momento no disponía de información que le permitiese acceder ni a la RTU ni a sus registros, pero ahora que el Gateway ya destinó un espacio de memoria para almacenar información de direccionamiento de las RTUs y de sus respectivos registros, se podrá corroborar mediante la Fig. 3.58. que el proceso de escritura sí se puede completar.

Proceso de lectura sobre los registros de las RTUs:

El proceso de lectura básicamente se refiere a la extracción del valor que está almacenado en un registro de una RTU, a diferencia del proceso de escritura el proceso de lectura tiene algunas particularidades:

- Los comandos de lectura no parten de una orden que se originó en el sistema SCADA, a diferencia de los comandos de escritura, lo ejecuta periódicamente el Gateway sobre cada uno de los registros de las RTUs.
- Los datos que el Gateway extrae de los registros de las RTUs son almacenados en un nuevo espacio de memoria que el Gateway destinará para estos datos, los cuales serán ordenados siguiendo la misma disposición de la red, como se hizo con el almacenamiento de las direcciones de las RTUs y sus registros en la Fig.3.55.

En la Fig. 3.59., se muestra el Gateway con el nuevo espacio de memoria (cuadros de color verde) destinado a almacenar los datos que se extrajo de los registros de las RTUs.

La cantidad de RTUs y de registros por cada RTU que hay en este ejemplo de eligió de forma arbitraria.

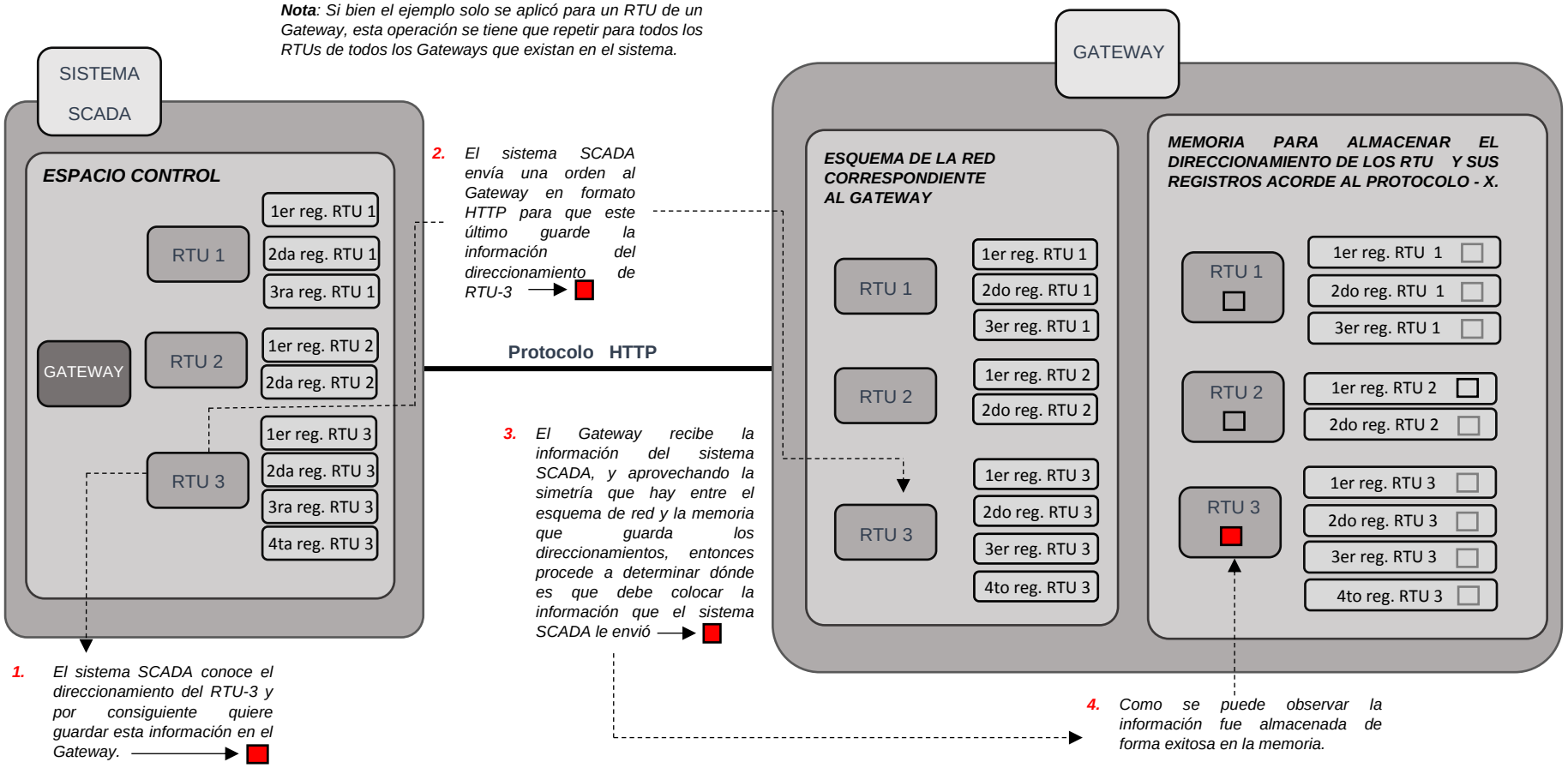


Fig. 3.56. Procedimiento para almacenar el direccionamiento de un RTU en el Gateway. [***Fuente**: Propia*]

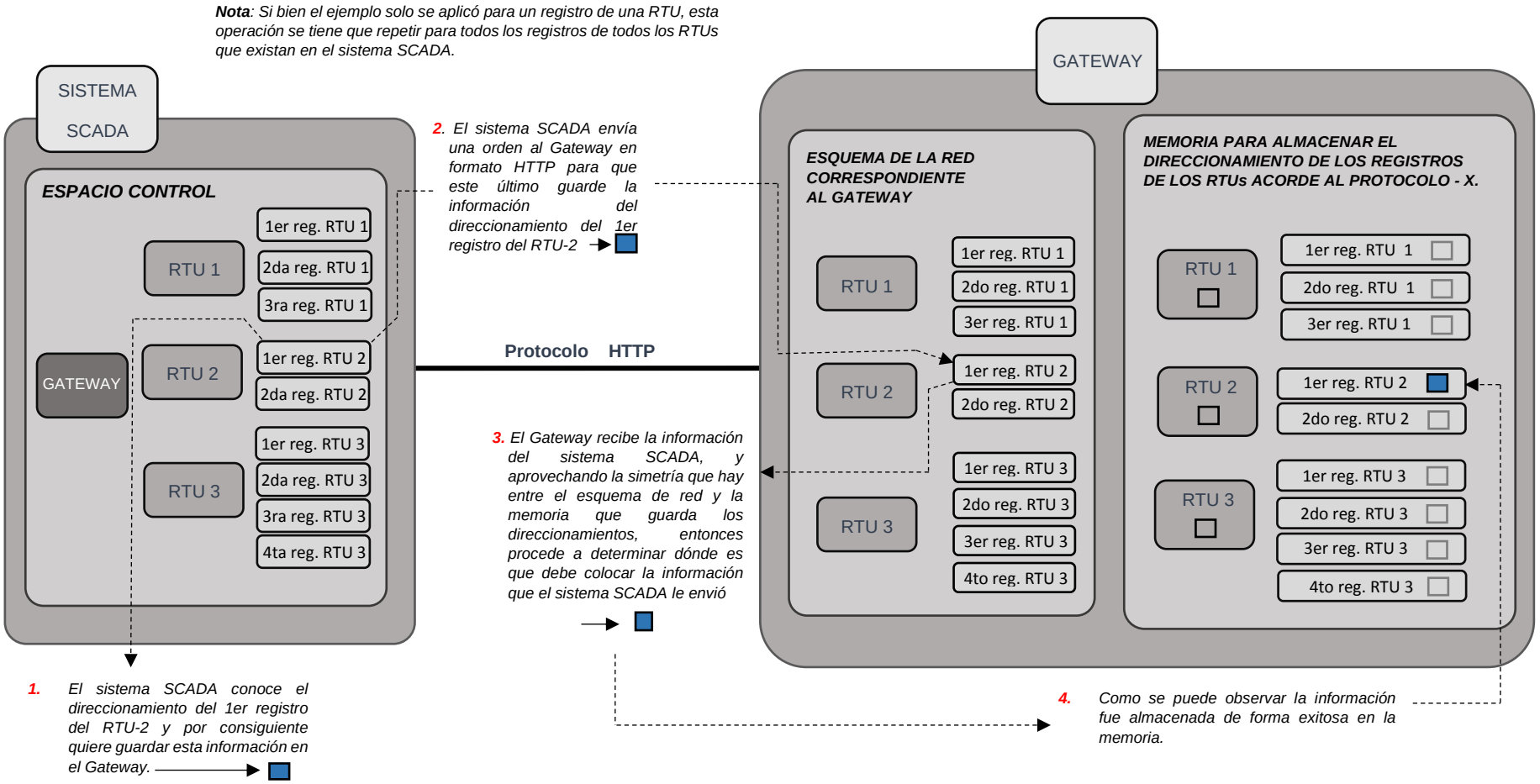


Fig. 3.57. Procedimiento para almacenar el direccionamiento del registro de una RTU en el Gateway. [***Fuente**: Propia*]

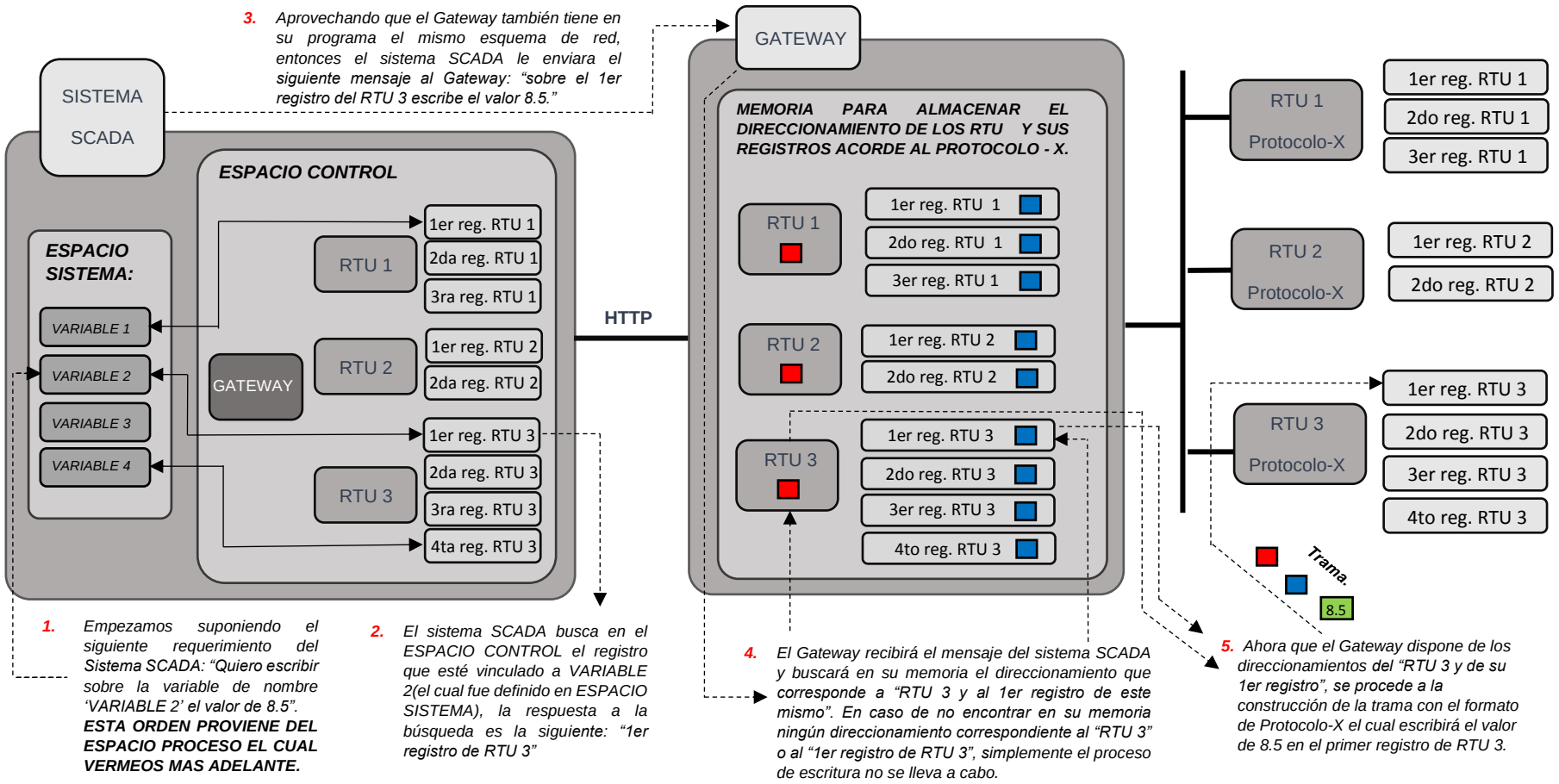


Fig. 3.58. Procedimiento para escribir un dato sobre un registro de una RTU. [***Fuente**: Propia*]

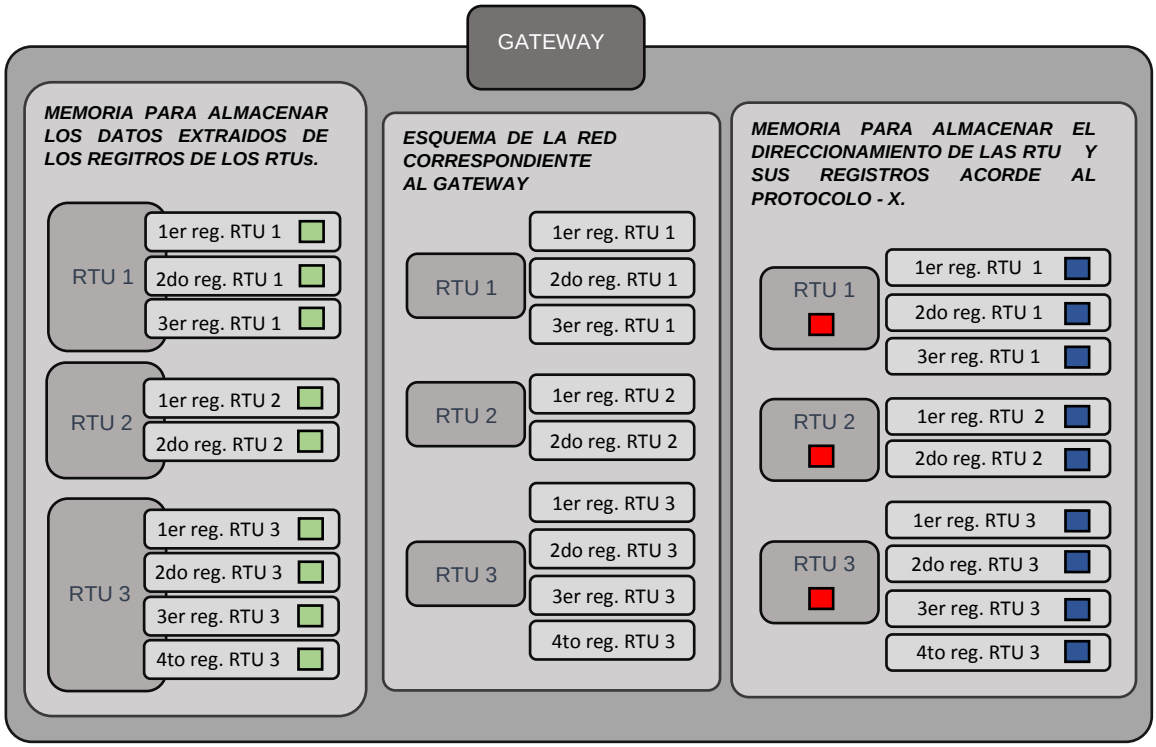


Fig. 3.59. Espacio de memoria en el Gateway (cuadro de la izquierda), que almacena los datos extraídos a de los registros de las RTUs del correspondiente Gateway. [***Fuente**: Propia*]

Proceso de llenado de la memoria del Gateway que almacena los datos extraídos de los registros de las RTUs.

En la Fig. 3.60., se muestra un ejemplo que muestra el proceso de lectura que está compuesto por la ejecución de los comandos de lectura y la posterior colocación de los datos extraídos en la memoria del Gateway, que está destinada a este propósito. Se usa el mismo esquema que se usó en la imagen anterior.

Es importante tener en cuenta que salvo en contadas ocasiones, los registros de las RTUs que son extraídos por el Gateway no son directamente colocados en la memoria de este último, sino que son convertidos previamente a un valor numérico que resulte más fácil de entender para el usuario.

En la Fig. 3.61., se puede observar cómo se realiza la transformación a un valor numérico. Este ejemplo aplica para un solo registro extraído, para los otros registros el razonamiento es el mismo. El mismo razonamiento aplica cuando se realiza el proceso de escritura sobre los registros de las RTUs, pero a la inversa.

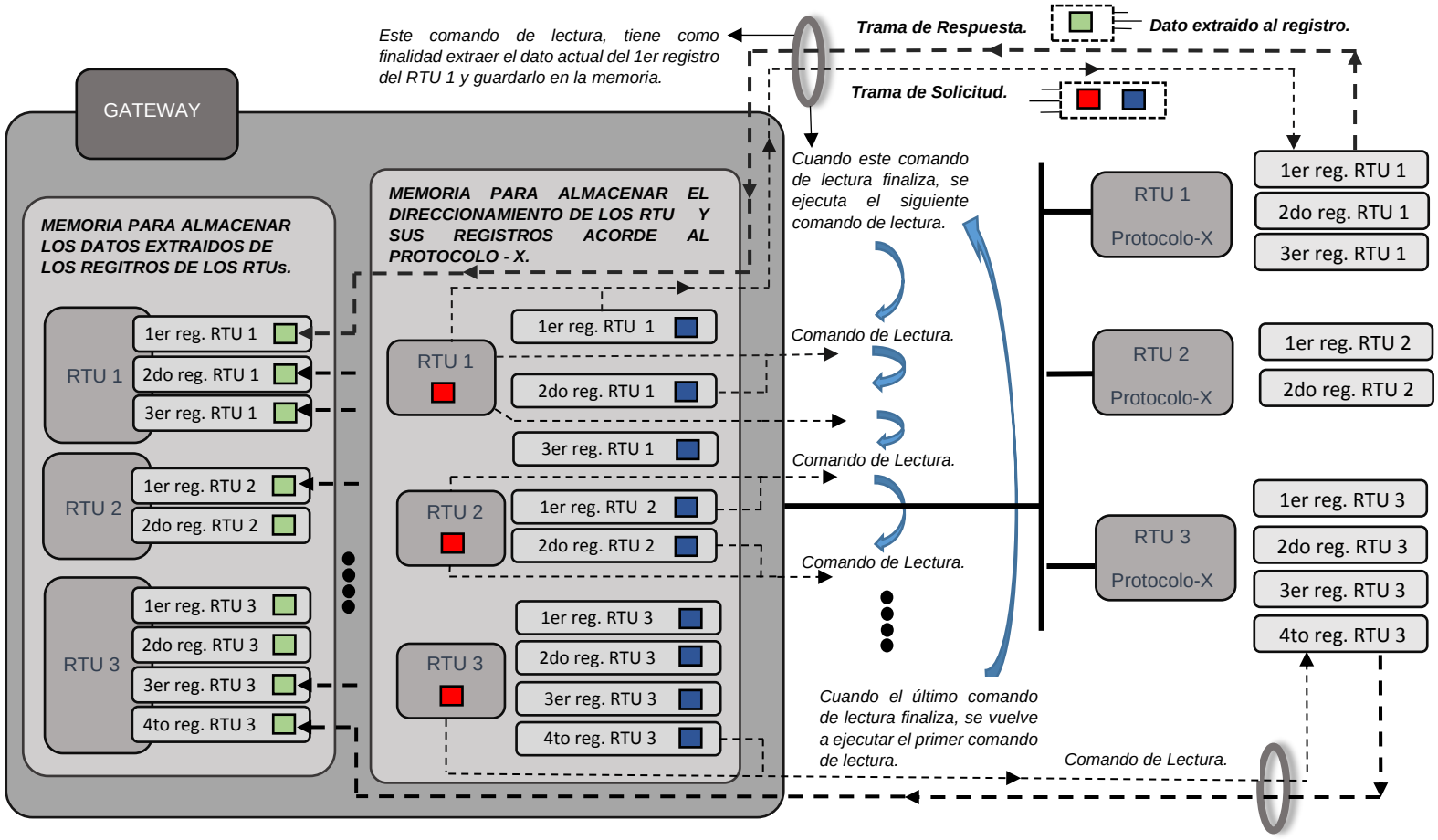


Fig. 3.60. Procedimiento para leer los datos de los registros de una RTU. [***Fuente**: Propia*]

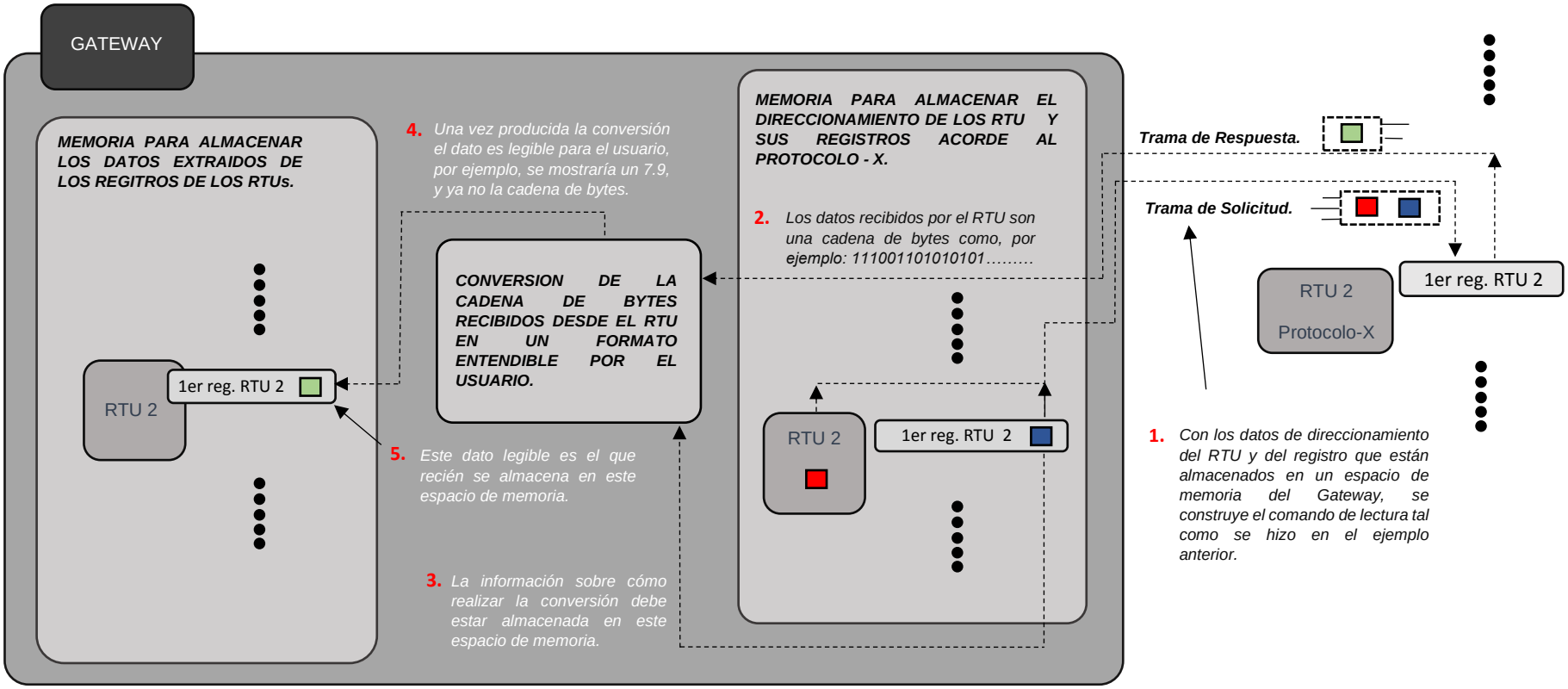


Fig. 3.61. Conversión de un dato correspondiente al registro de una RTU y su posterior almacenamiento en el Gateway.
[***Fuente**: Propia*]

Extracción de los datos almacenados en el Gateway desde el sistema SCADA

La Fig. 3.62., muestra un ejemplo que detalla el proceso que lleva a cabo el sistema SCADA para extraer los datos que están almacenados en el Gateway, que fueron extraídos de los registros de las RTUs mediante comandos de lectura que se vieron en el ejemplo anterior, ver Fig. 3.60. y Fig. 3.61. Se usa el mismo esquema que se usó en la imagen anterior.

Al final se puede observar cómo el ESPACIO CONTROL hace entrega a cada una de las INSTACIAS del ESPACIO SISTEMA sus respectivos valores en tiempo real, con lo cual se puede dar por concluido el servicio que ofrece ESPACIO CONTROL.

Es importante mencionar que el sistema SCADA mediante un único comando HTTP extrae TODOS los datos que tiene almacenado el Gateway, es decir, no importa si el Gateway tiene 200; 500 o 1000 registros asociados. El sistema SCADA con solo 1 comando HTTP obtendrá todos los datos del Gateway.

También es importante mencionar que el protocolo de comunicación entre el sistema SCADA y sus Gateways es el protocolo HTTP, pero también se puede usar el protocolo HTTPS, que ofrece encriptación, pero para ello el Gateway también tiene que tener implementado HTTPS.

Se debe notar que en la operación de extracción de datos del sistema SCADA sobre el Gateway, el sistema SCADA se comporta como un cliente HTTP (también se le conoce como cliente WEB) y el Gateway se comporta como un servidor HTTP (también se le conoce como servidor WEB).

Lo mencionado líneas arriba es muy importante tenerlo en cuenta ya que cuando se ingresa a los formularios la información de los elementos que componen la red como Gateways, RTUs y registros de las RTUs (ver Fig. 3.40., Fig. 3.41. y Fig. 3.42.) en estos casos el navegador (Chrome, Explorer, Mozilla etc.) es cliente HTTP y el sistema SCADA es el servidor HTTP. Es decir, el sistema SCADA dependiendo de las tareas que realiza, puede comportarse como cliente HTTP o servidor HTTP.

Restricción en cuanto a los vínculos del Espacio Sistema con el Espacio Control

De la Fig. 3.62., se puede observar que, si 2 registros del ESPACIO CONTROL se vincularían a una misma VARIABLE del ESPACIO SISTEMA, los datos de los 2 registros se traslaparían de forma indeterminada. Para evitar esto, al momento de vincular un registro del ESPACIO CONTROL con una VARIABLE del ESPACIO SISTEMA, se debe realizar una verificación previa en la que se garantice que la VARIABLE del ESPACIO SISTEMA aún no está vinculada aun con algún registro del ESPACIO CONTROL.

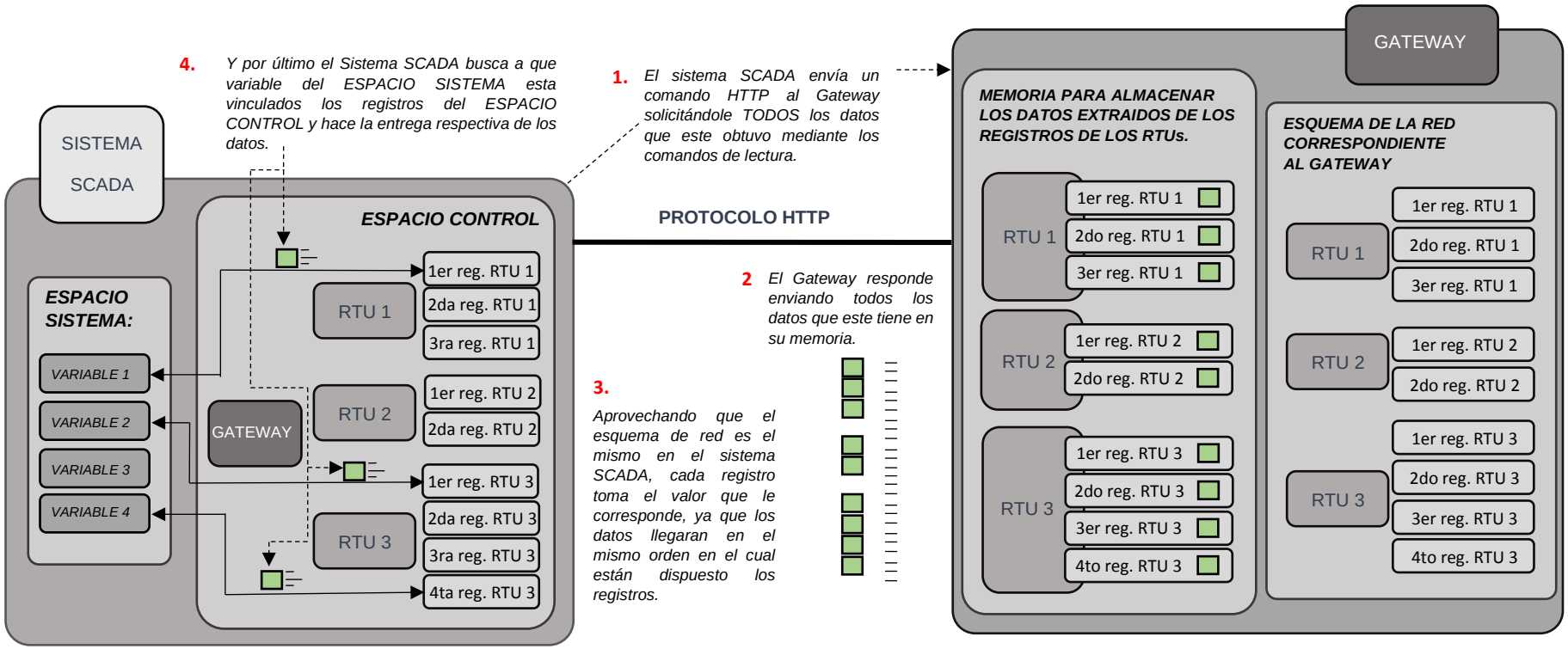


Fig. 3.62. Extracción periódica de datos almacenados en el Gateway realizadas por el sistema SCADA, típicamente cada segundo. [***Fuente**: Propia*]

h) Proceso de configuración del Gateway

En la subsección anterior se analizó el proceso para hacer posible la generación de los comandos de escritura y de lectura sobre los registros de las RTUs. En esta subsección se aplicará dichos procesos a un ejemplo real para la Tesis.

Para la Tesis se supuso la existencia de 3 plantas industriales: CX_15, CX_11 y AX_08; con sus respectivos Gateways y RTUs como se puede observar en las Fig. 3.34., 3.35. y 3.36., así como también se había supuesto una determinada cantidad de registros que iban a ser extraídos de cada RTU.

Si bien entre las 3 plantas se enlistan un total de 8 Gateways (3 en CX_15, 2 en CX_11 y AX_08) que tienen implementados los protocolos HTTP y Modbus-RTU. En la Tesis solo se trabajará con un solo Gateway en un circuito real. El Gateway elegido es el GATEWAY 2 de la planta CX_11, tal como se muestra en la Fig. 3.63., que fue extraída de la Fig. 3.35. Se pudo haber elegido cualquiera de los 8 Gateways que había en las 3 plantas.

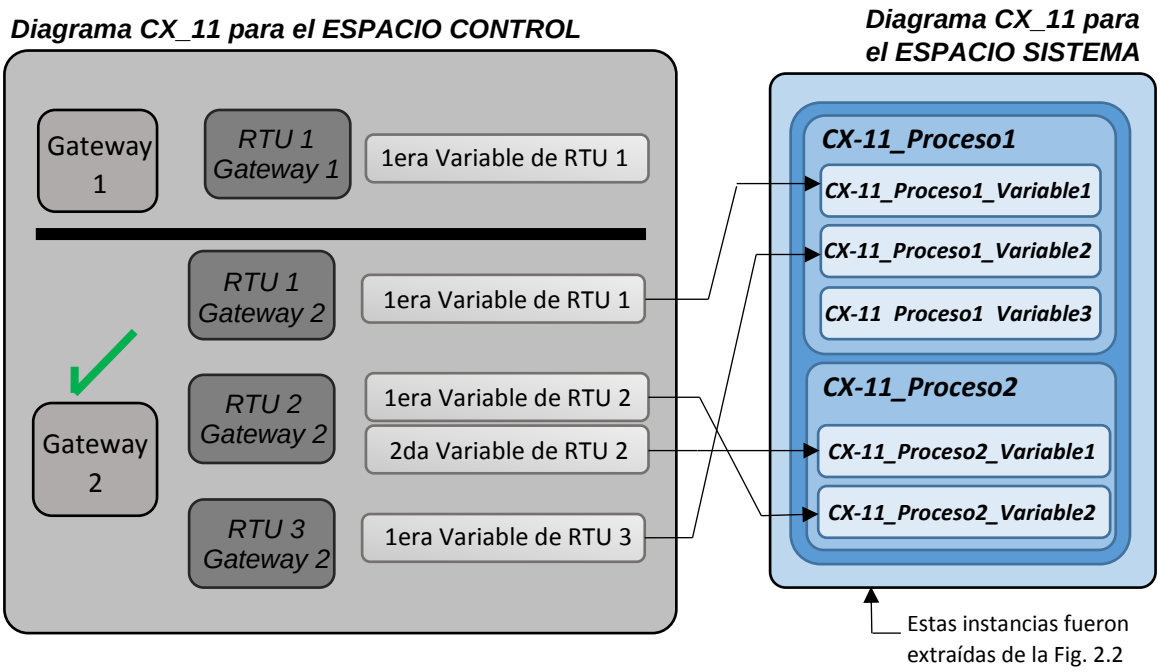


Fig. 3.63. Gateway 2 de la planta CX_11 y su correspondiente red (RTUs y registros) será implementado en un circuito real. [***Fuente**: Propia*]

En la Fig. 3.64., se muestra una estructura real integrada por el sistema SCADA, el Gateway 2 de la Planta CX_11 y sus 3 RTUs con las cuales está conectado el Gateway, ver la Fig. 3.63.

.

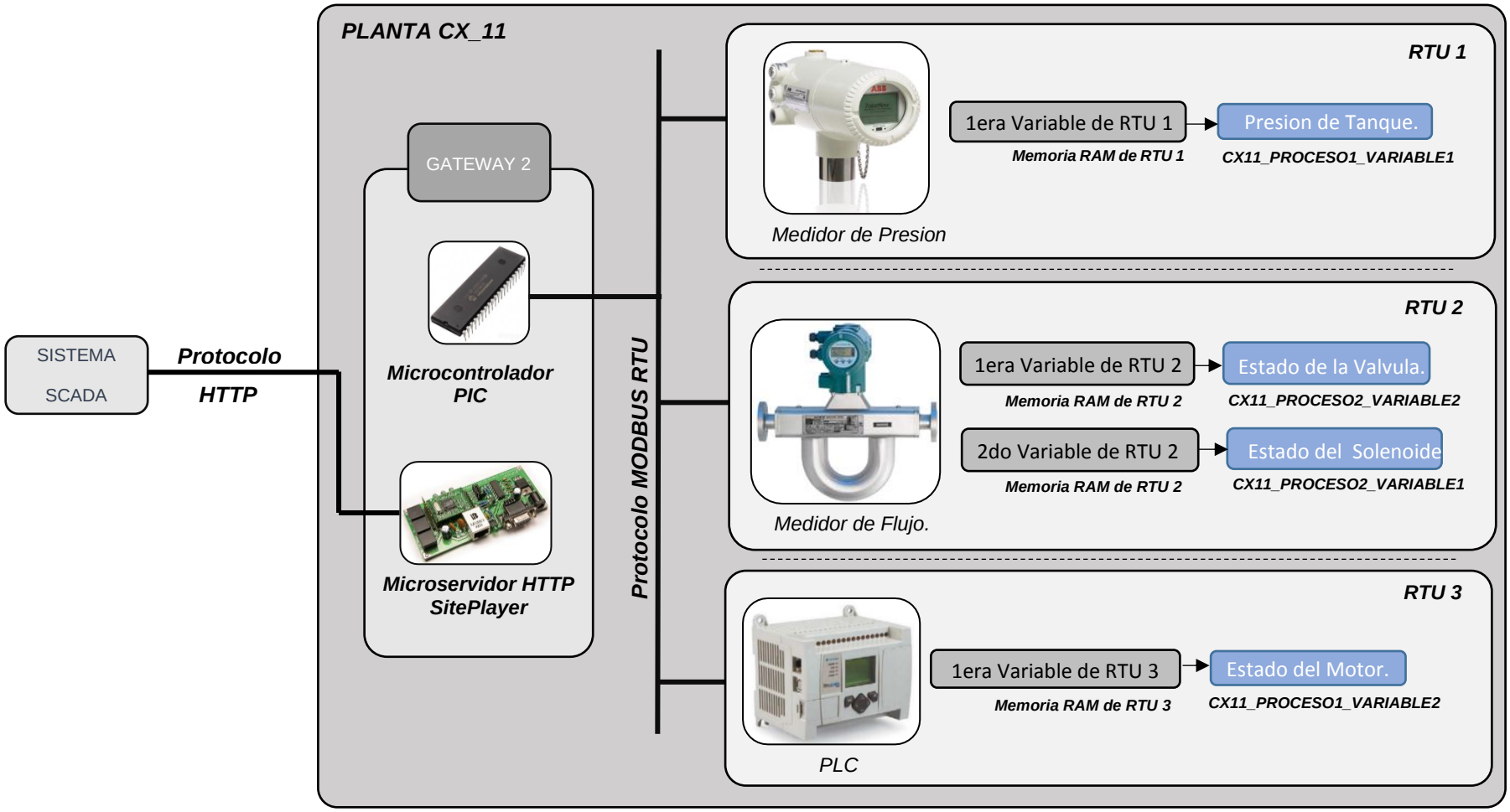


Fig. 3.64. RTUs que integraran una red Modbus-RTU y se comunican con el Gateway 2, el cual se implementó en un circuito real. [***Fuente**: Propia*]

Conocidas las RTUs con las que se comunicará el Gateway 2, a partir de las hojas de datos o datasheets de las RTUs se buscará la información relacionada con los registros que se desean extraer de estas RTUs, como se mostró en la Fig. 3.32. Como las RTUs tienen implementado el protocolo Modbus-RTU, se debe conocer cómo el protocolo Modbus-RTU establece el direccionamiento de sus RTUs y de los registros de éstos.

En la siguiente subsección se describe los puntos más importantes del protocolo Modbus-RTU.

i) Breve descripción del Protocolo Modbus-RTU

Este protocolo fue creado en 1979 por la empresa Modicon (comprada años después por Schneider Electric), es un protocolo simple comparado con los protocolos modernos que hay en la industria. Modbus-RTU es el protocolo más usado por las RTUs actualmente.

Antes de analizar la trama Modbus-RTU observe las Fig. 3.56., Fig. 3.57. y Fig. 3.61. Estas figuras muestran que los principales objetivos serán:

- Obtener el direccionamiento de la RTU. Se refiere al formato con el cual se puede acceder a una RTU, ver Fig. 3.56.
- Obtener el direccionamiento del registro o variable de la RTU. Se refiere al formato con el que se puede acceder al registro de una RTU, ver Fig. 3.57.
- Obtener el método de conversión de los registros extraídos. Se refiere a cómo son procesados las cadenas binarias recibidas por la RTU para convertirlas a un formato numérico legible para el usuario, el cual será posteriormente almacenado en el Gateway, ver Fig. 3.61.

Otro punto a tener en consideración es que la información relacionada con los direccionamientos y la conversión, muchas veces no es mostrada de forma directa en la trama (esto es en general para cualquier protocolo), sino que se debe realizar algún tipo de análisis para obtener esta información. Como se observará más adelante, incluso Modbus- RTU, no contempla el tema de la conversión.

Composición de una trama de lectura Modbus.

La Fig. 3.65., muestra la composición de una trama de lectura Modbus-RTU y cómo es que la memoria del Gateway almacena parte de los campos de la trama.

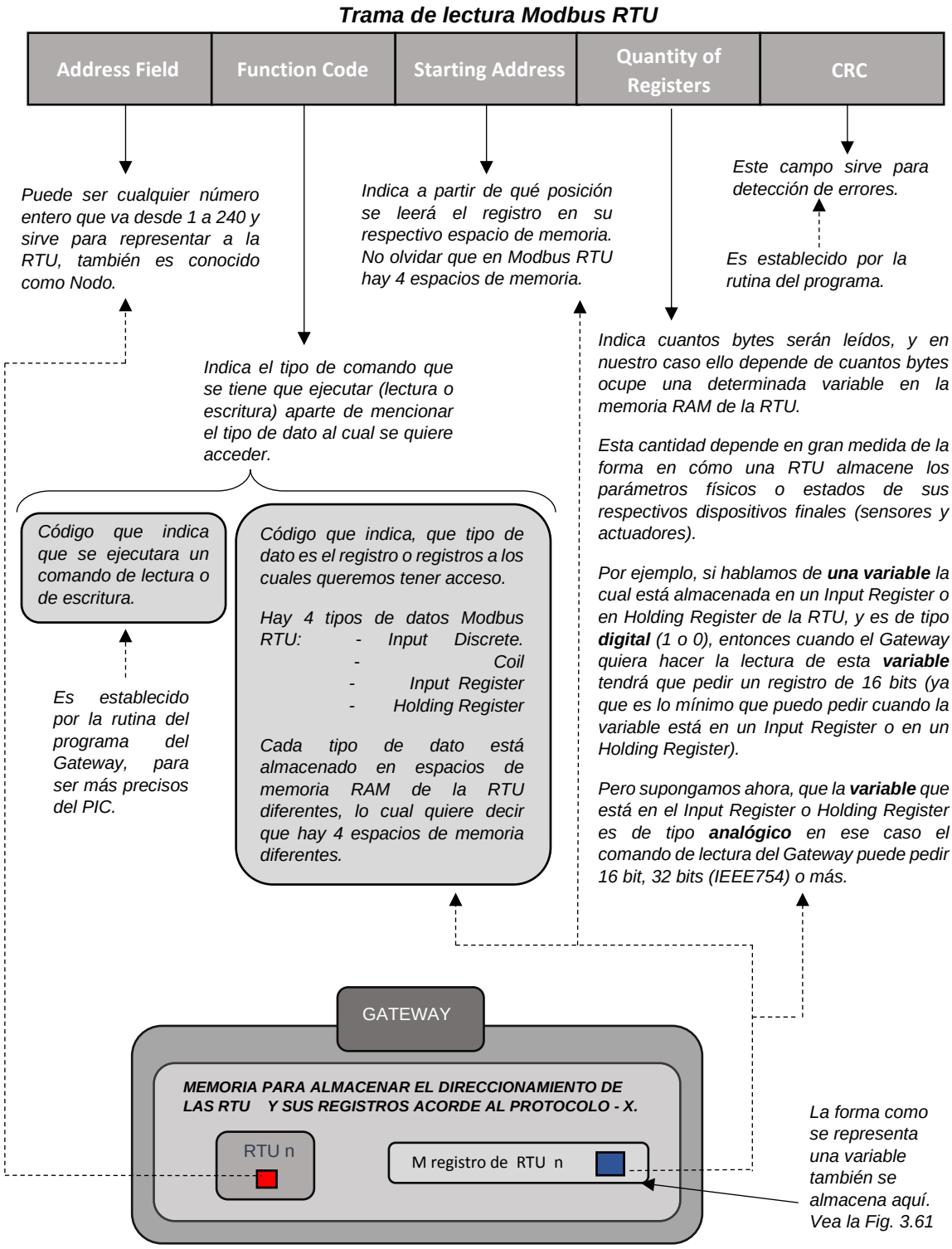


Fig. 3.65. Trama de lectura Modbus-RTU. [***Fuente**: Propia*]

Composición de una trama de escritura Modbus

La Fig. 3.66., muestra la composición de una trama de escritura Modbus-RTU y cómo la memoria del Gateway almacena parte de los campos de la trama.

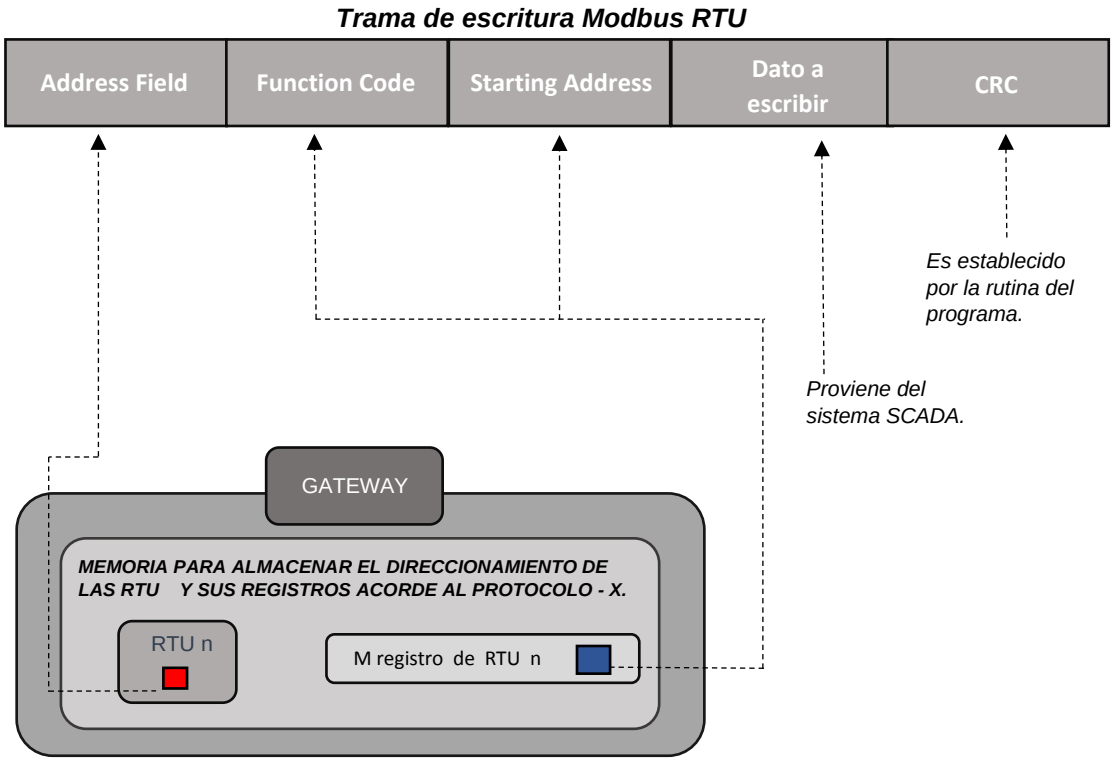


Fig. 3.66. Campos de una trama de escritura. [***Fuente**: Propia*]

j) Configuración del direccionamiento de las RTUs y sus respectivos registros

Se usará el software MODBUS SLAVE (Witte Software – "Modbus-Slave") [14] para simular el funcionamiento de la RTU 1, RTU 2 y RTU 3, mostradas en la Fig. 3.64. En la Fig. 3.67., se muestra una escueta descripción de este software.

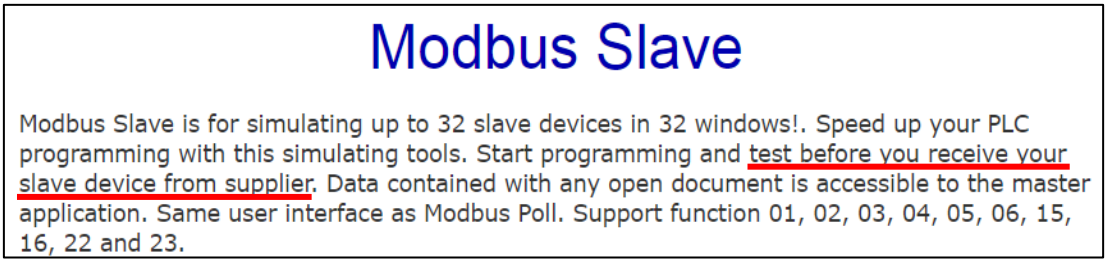


Fig. 3.67. Descripción del software MODBUS SLAVE.

[***Fuente**: https://www.modbustools.com/modbus_slave.html*]

La configuración de la RTU 1 se muestra en la Fig. 3.68.

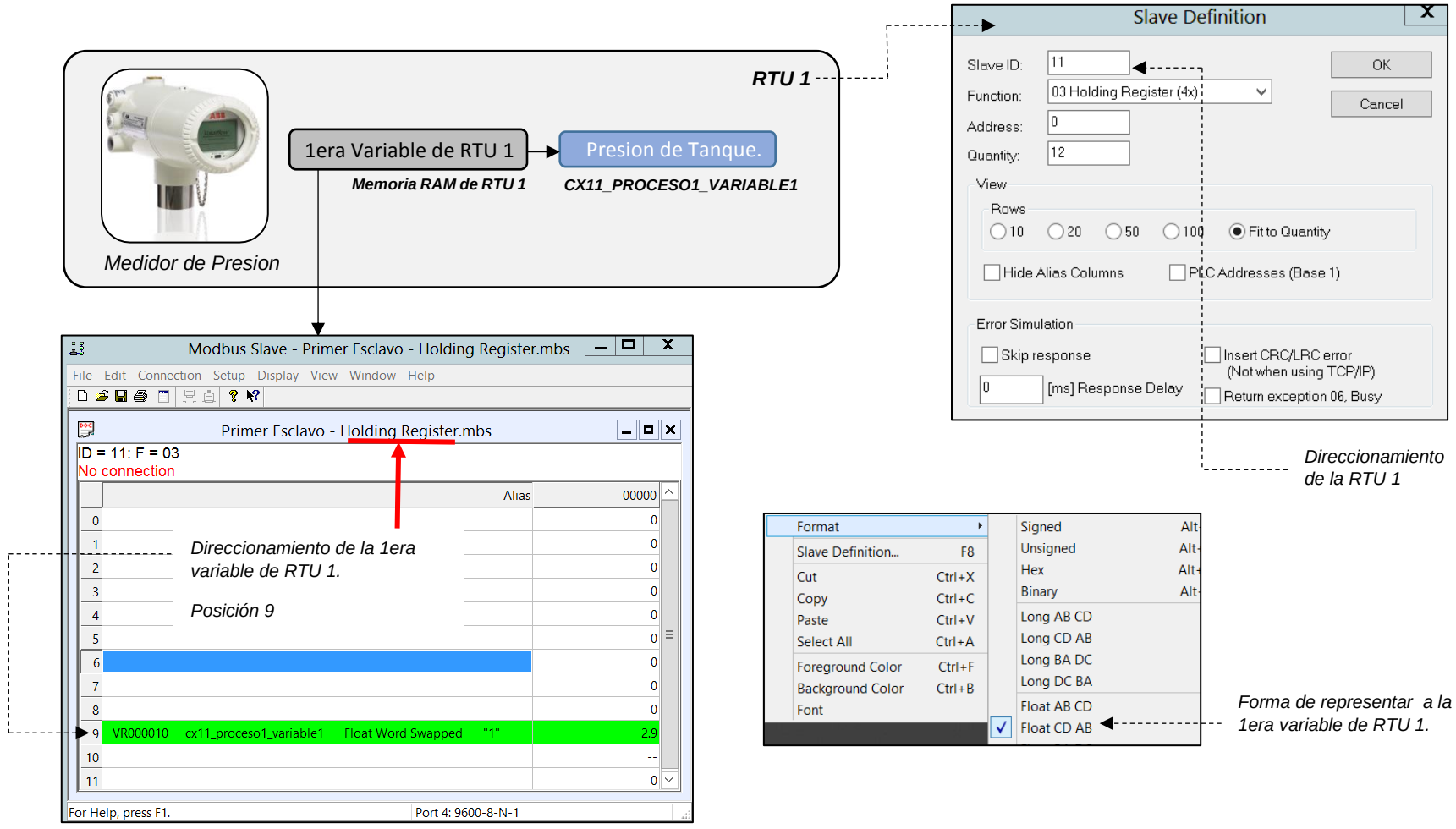


Fig. 3.68. Configuración de la RTU 1 y su variable. [***Fuente**: Propia*]

La configuración de la RTU 2 se muestra en la Fig. 3.69.

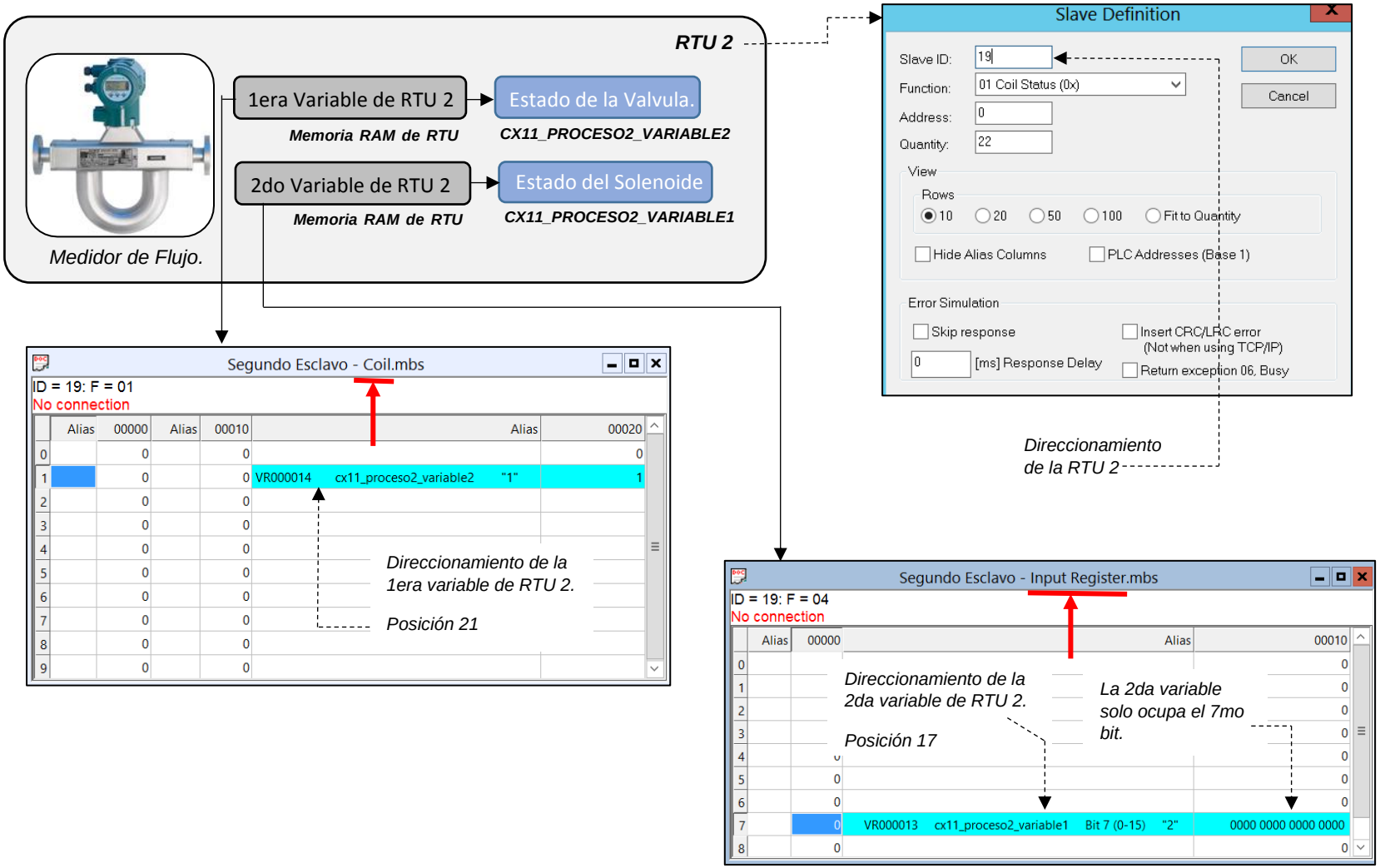


Fig. 3.69. Configuración de la RTU 2 y sus 2 variables. [***Fuente**: Propia*]

La configuración de la RTU 3 se muestra en la Fig. 3.70.

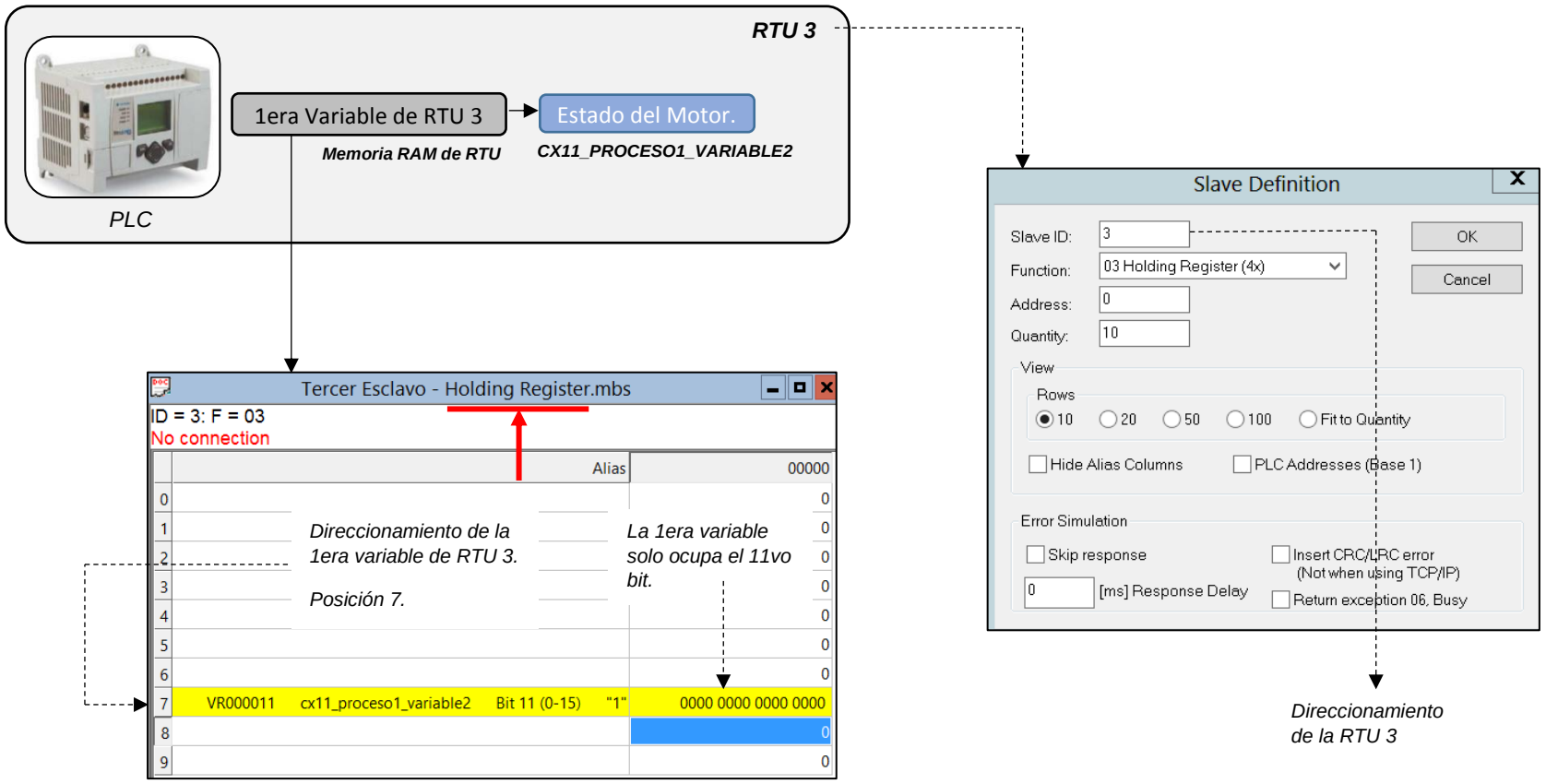


Fig. 3.70. Configuración de la RTU 3 y su variable. [***Fuente**: Propia*]

El llenado de los direccionamientos como se mostró en la Fig. 3.55. para el Gateway de la Tesis, tiene que ser acorde a Fig. 3.68., Fig. 3.69. y Fig. 3.70.

Como se había mencionado anteriormente, el llenado de los direccionamientos se realiza desde el sistema SCADA, tal como se muestra en Fig. 3.56. y Fig. 3.57.

A continuación, se muestra como se llenaron los formularios desde el sistema SCADA que permitieron el llenado de direccionamientos, para ello se presionara el botón que se muestra en la Fig. 3.71.

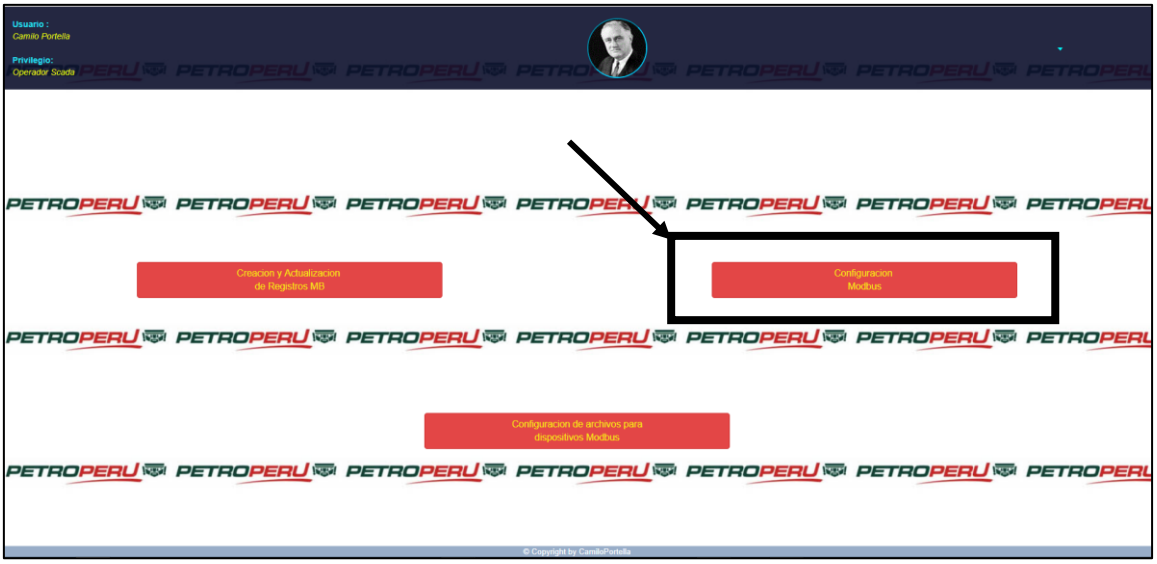


Fig. 3.71. El botón para acceder al llenado de direccionamiento de las RTUs y sus registros. [***Fuente**: Propia*]

En la Fig. 3.72. se muestra el formulario correspondiente a la RTU 1 y su variable, como se puede apreciar los datos ingresados provienen de la Fig. 3.68

Fig. 3.72. Configuración de la RTU 1 su variable. [***Fuente**: Propia*]

La Fig. 3.73., muestra el formulario correspondiente a la RTU 2 y sus 2 variables, como se puede observar los datos ingresados provienen de la Fig. 3.69.

Fig. 3.73. Configuración de la RTU 2 y sus variables. [***Fuente**: Propia*]

La Fig. 3.74., muestra el formulario correspondiente a la RTU 3 y su variable, como se puede observar los datos ingresado provienen de la Fig. 3.70.

Fig. 3.74. Configuración de la RTU 3 y su variable. [***Fuente**: Propia*]

A continuación, se comprueba que la información de direccionamiento de las RTU y sus respectivos registros (establecidas en Fig. 3.68, Fig. 3.69 y Fig. 3.70) se guardó en el Gateway. El Gateway está compuesto por un microservidor SitePlayer y por un microcontrolador PIC tal como se muestra en la Fig. 3.64. Estos datos de direccionamiento podrían haberse guardado en el SitePlayer o en el PIC, pero como el SitePlayer no tiene una memoria de datos permanente, entonces se guardarán los datos de direccionamiento en la memoria EEPROM del PIC.

En la Fig.3.75., se muestra la memoria EEPROM del PIC, y se podrá observar cómo es que este último almacenó los direccionamientos que se envió a través de los formularios de las Fig. 3.72., Fig. 3.73. y Fig. 3.74.

Se debe tener presente que el Gateway que se está analizando corresponde al 2do Gateway de la Planta CX_11, ver la Fig. 3.73.

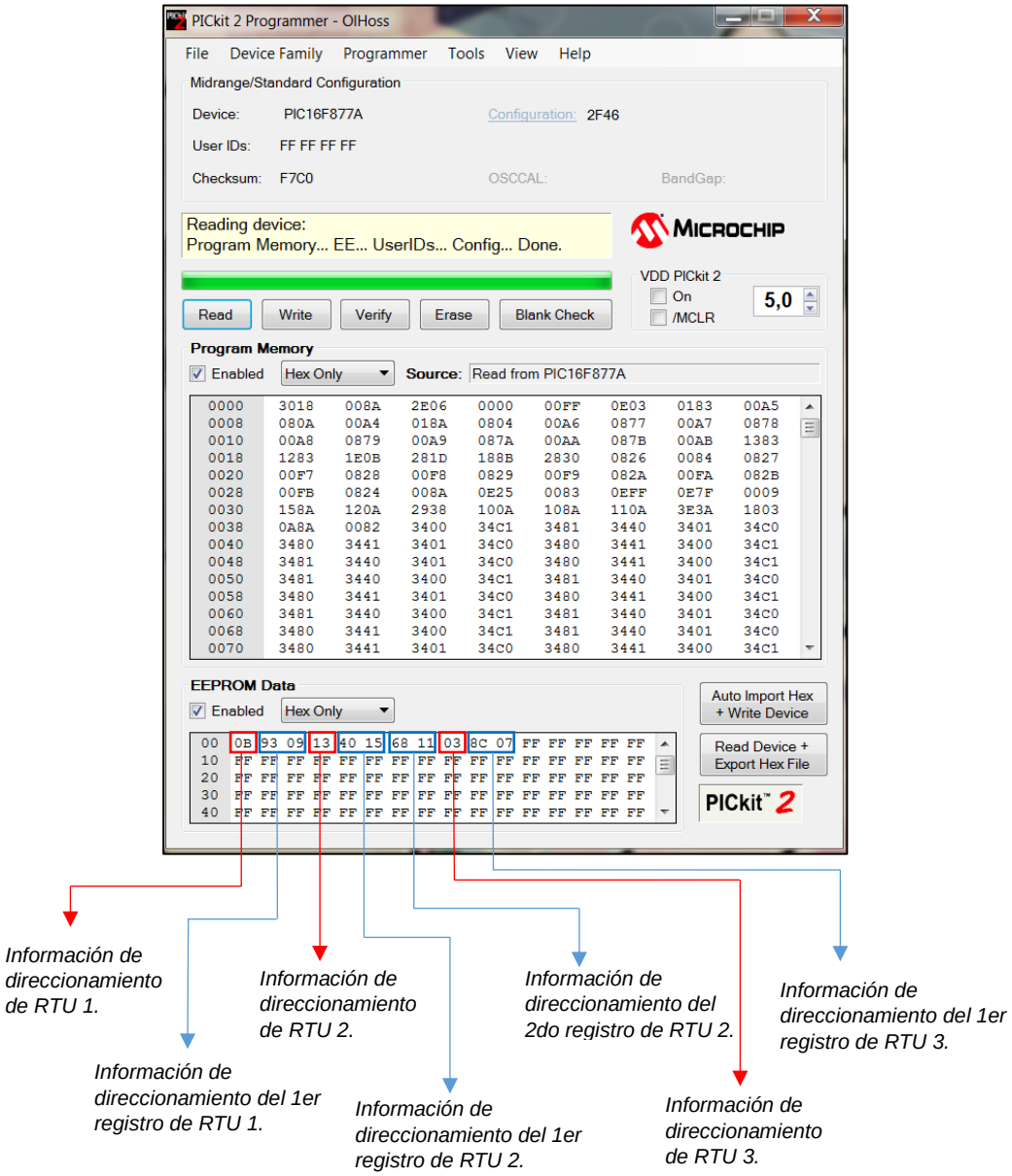


Fig. 3.75. Memoria EEPROM del PIC. [***Fuente**: Propia*]

De acuerdo a la Fig. 3.60., el proceso de generación del comando de lectura desde el Gateway hacia las RTUs era posible realizarlo con la condición de tener almacenados los direccionamientos de las RTUs y los direccionamientos de sus respectivos registros.

Los comandos de lectura que se ejecutan acorde al ejemplo que se ha planteado en este proyecto de Tesis (tome como referencia la Fig. 3.68, 3.69 y 3.70 y Fig. 3.60), se muestran en la Fig. 3.76.

El software MODBUS SLAVE ofrece la posibilidad de poder observar el tráfico que se produce en la comunicación Modbus-RTU.

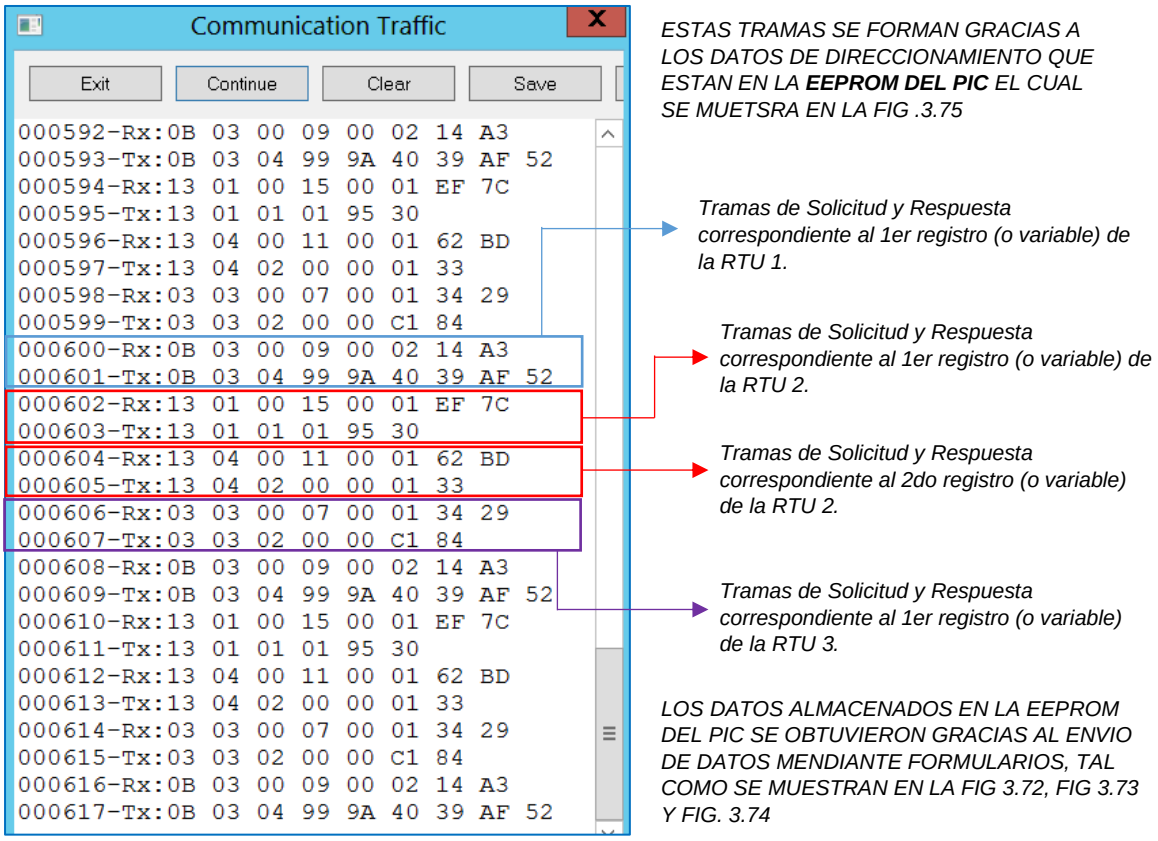


Fig. 3.76. Comandos de lectura que se ejecutan sobre las RTUs del Gateway 2. [***Fuente**: Propia*]

k) Depuración del contenido de los Gateway

Lo que se explica a continuación aplica a cualquier Gateway de cualquier protocolo. Ver la Fig. 3.62., para que pueda observar como el Gateway almacena los valores extraídos de los registros de las RTUs en un espacio de memoria.

Como se puede observar también en la Fig. 3.62., estos datos son extraídos a su vez por el sistema SCADA para luego ser enlazados a su respectiva Instancia de Variable. Entonces, una forma de verificar que el contenido almacenado en el Gateway, es correcto, sería a través de una interface colocada en el sistema SCADA.

El objetivo ahora es proporcionar herramientas que permitan verificar el correcto funcionamiento del Gateway de forma más directa, lo que permitirá al usuario identificar el lugar de la falla, si esta se presentará. Para ello se debe crear dentro del Gateway archivos depuradores de lectura y de escritura, para evitar tener que usar el sistema SCADA en la verificación de los errores que presenten en el Gateway.

En la Fig. 3.77, se muestra el Gateway y sus nuevos archivos de depuración, como se puede apreciar hay una carga demasiado grande sobre un Gateway, dicho en otras palabras, hay demasiada programación que desarrollar sobre estos dispositivos.

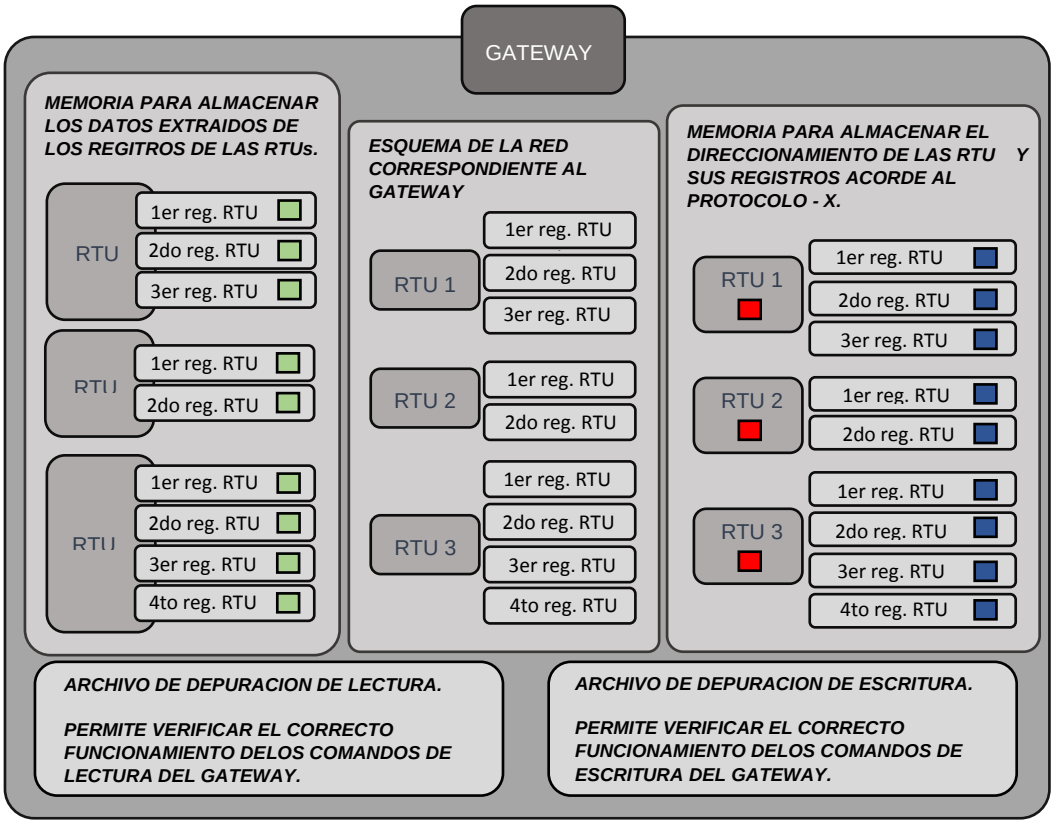


Fig. 3.77. Archivos que permiten verificar funcionamiento de los comandos de lectura y escritura. [***Fuente**: Propia*]

Estos archivos de depuración tienen que estar incluidos en los files que se generan desde el sistema SCADA y que luego son descargados en el Gateway, tal como se muestra en la Fig. 3.48.

En la Fig. 3.78., se muestra cómo es el funcionamiento el proceso de depuración.

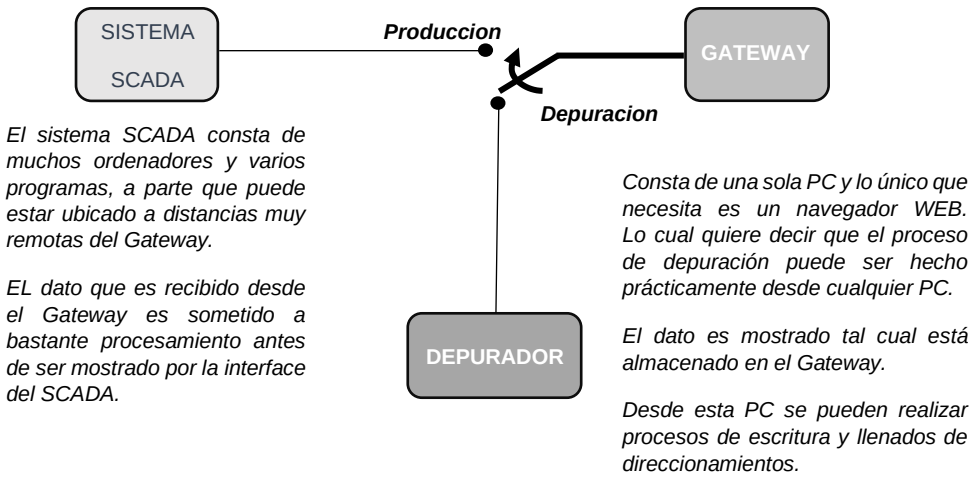


Fig. 3.78. A través de un navegador WEB, se puede depurar o verificar errores. [***Fuente**: Propia*]

l) Generación de archivos para la depuración en los Gateway

En la Fig. 3.79. y la Fig. 3.80. se muestran cómo se visualizan los archivos de depuración.

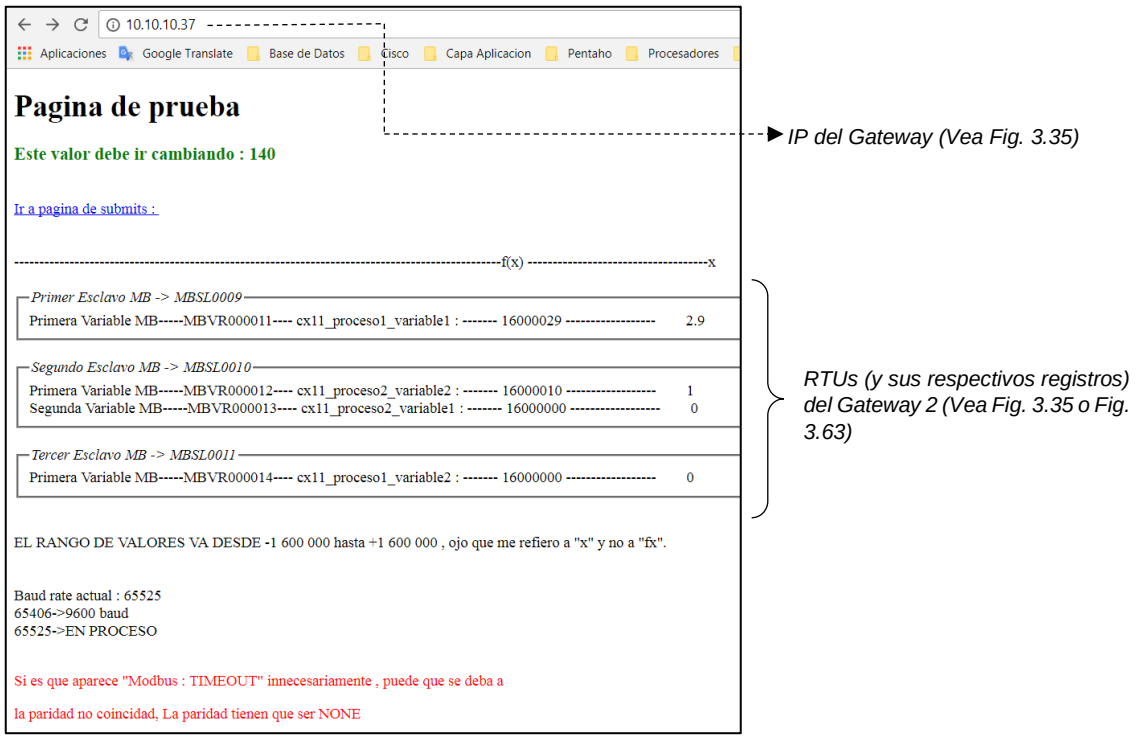


Fig. 3.79. Archivo de depuración de lectura del Gateway 2, que pertenece a la planta CX_11. [***Fuente**: Propia*]

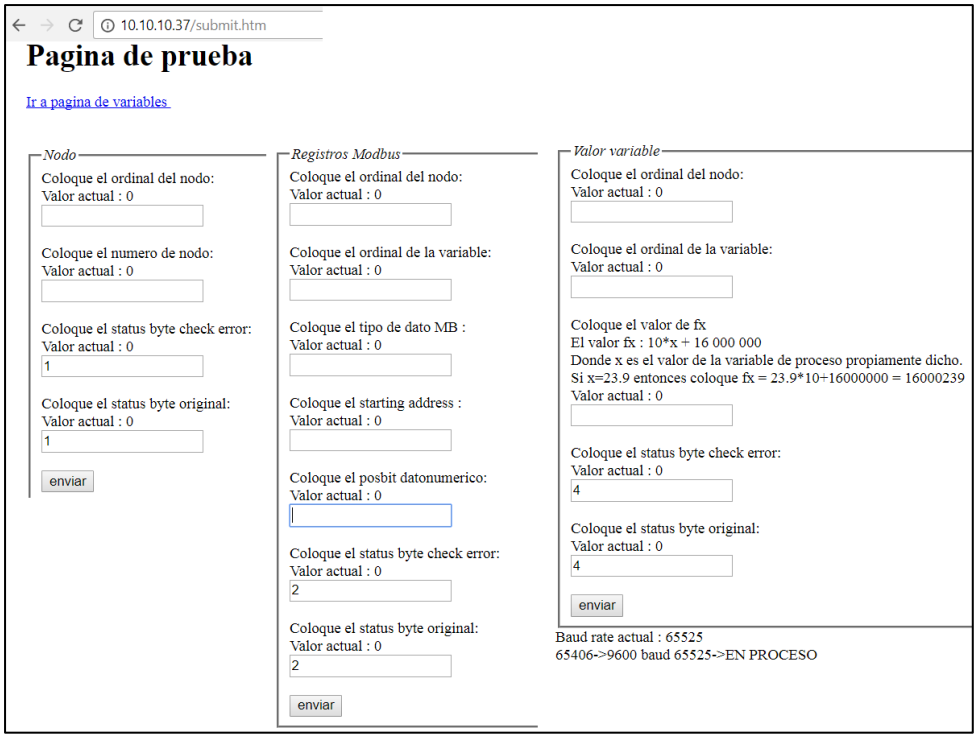


Fig. 3.80. Archivo de depuración de escritura del Gateway 2, que pertenece a la planta CX_11. [***Fuente**: Propia*]

m) Extracción de datos almacenados en los Gateways por el sistema SCADA y su vínculo con la Clase Variable

Se observó en la Fig. 3.62., los procedimientos de cómo el sistema SCADA extraía los datos de los Gateways, datos que este último había obtenido de los registros de sus RTUs a través de los comandos de lectura tal como se mostró en la Fig. 3.60.

A continuación, se realiza un diseño de tal forma que todos los scripts del sistema SCADA que se encargarán de la extracción de datos de los Gateways estén almacenados en una carpeta con el nombre del Gateway correspondiente.

En la Tesis se aplicará el criterio mencionado líneas arriba, entonces los scripts que se encargan de la extracción de datos de los Gateways se distribuirán tal como se muestra en Fig. 3.81.

En la tesis solo se ha implementado uno de los Gateway en un circuito real. Tal Gateway es el Gateway 2 de la Planta CX_11, tal como lo muestra la Fig. 3.63. Ese mismo Gateway se puede encontrar en la tabla de la Fig. 3.44., con el código MBMA005. Notar que en el campo ID Nombre Modbus Maestro dice 2, lo que significa que es el segundo Gateway o Gateway 2, y en el campo ID Planta dice PL002 o Planta CX_11.

Se activará los scripts correspondientes al Gateway MBMA005 tal como se muestra en la Fig. 3.82.

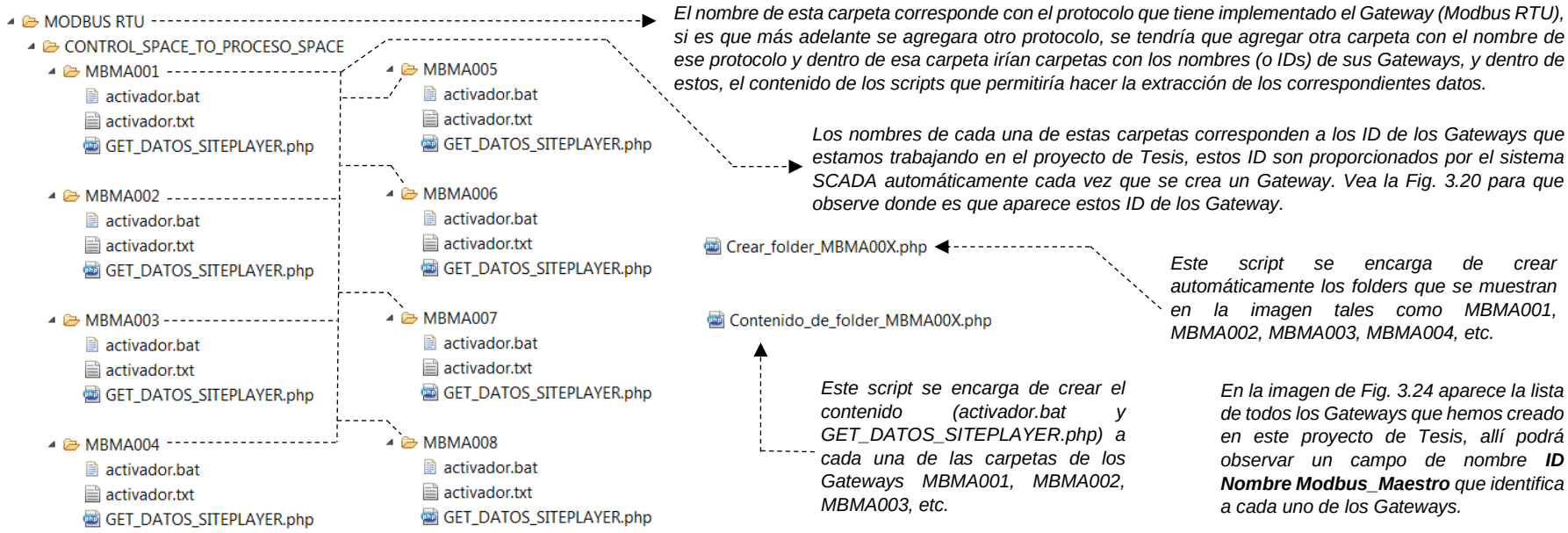
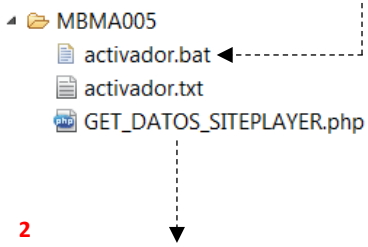


Fig. 3.81. Archivos que contienen los scripts encargados de la extracción de datos de sus respectivos Gateway. [***Fuente**: Propia*]

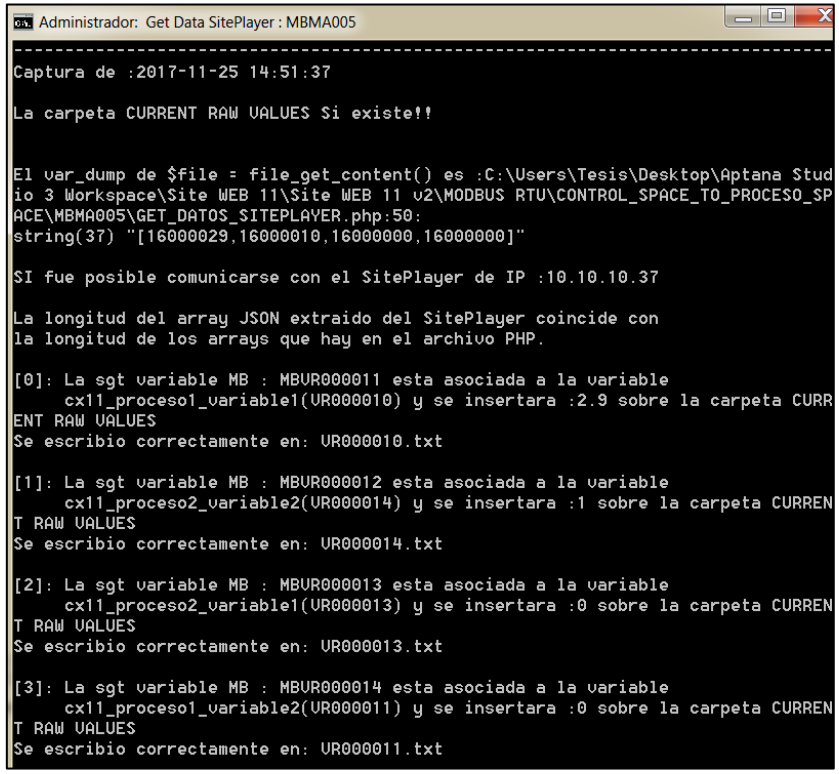
1

El archivo de nombre activador.bat se encargará de activar periódicamente el archivo de nombre GET_DATOS_SITEPLAYER.php



2

El archivo de nombre GET_DATOS_SITEPLAYER.php se encarga de extraer los datos que su correspondiente Gateway tiene almacenando



No olvidar que este Gateway con el que nos estamos comunicando es el mismo Gateway de la Fig. 3.63

3

Como se puede observar los datos extraídos son correctos, para ello compare los resultados que se muestra en esta figura y los resultados que se visualizaron en el depurador de lectura en la Fig. 3.79

Luego se procede a realizar el enlace entre los registros de las RTUs con sus respectivas INSTANCIAS de la CLASE VARIABLE.

Por ejemplo, MBVR000011, que es el primer registro (o variable) de RTU 1 se enlaza a cx11_proceso1_variable 1, que era lo que se había establecido en la Fig. 3.63

Fig. 3.82. Extracción de datos que el sistema SCADA realiza sobre el Gateway 2 de la planta CX_11, que tiene el ID: MBMA005 como se muestra en la tabla de la Fig. 3.63. [***Fuente**: Propia*]

n) Almacenamiento de los datos extraídos en el sistema SCADA

La última etapa del proceso de lectura está relacionada con el almacenamiento de los valores obtenidos por el sistema SCADA en una carpeta llamada CURRENT RAW VALUES, ver Fig. 3.83. Cada valor que se haya extraído de los registros de las RTUs (ESPACIO CONTROL) será guardado en esta carpeta, pero en un archivo de texto plano cuyo nombre será el ID de la instancia de la CLASE VARIABLE correspondiente (ESPACIO SISTEMA).

Este almacenamiento se llevó a cabo gracias a la ejecución de los archivos de nombre GET_DATOS_SITEPLAYER.php por parte del archivo activador.bat tal como se muestra en la Fig. 3.82. La ID de una instancia de la CLASE VARIABLE se crea de forma automática al momento de crear la instancia, las instancias se almacenan en la tabla de la Fig. 3.20.

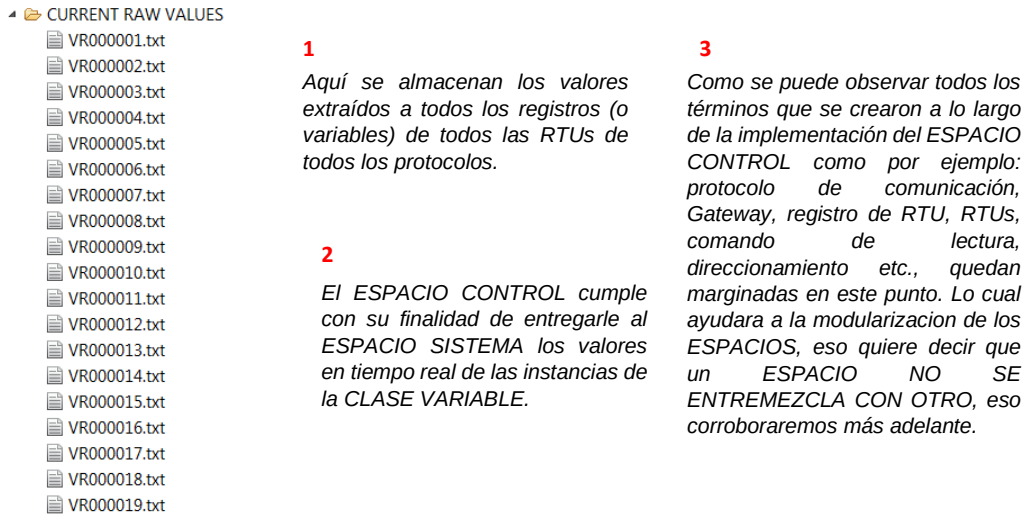


Fig. 3.83. Los datos extraídos por el sistema SCADA son almacenados y se finaliza el trabajo de lectura del Espacio Control. [***Fuente**: Propia*]

o) Generador de comandos de escritura desde el sistema SCADA

A diferencia de los comandos de lectura que es generado exclusivamente por decisión del Gateway, los comandos de escritura tienen su origen en el sistema SCADA, tal como se muestra en la Fig. 3.58.

El sistema SCADA generará tales comandos de escritura, mencionado líneas arriaba, mediante un archivo de nombre SET_VALUES_PROTOCOLS.php (ver Fig.3.84.), realizando la secuencia que se muestra en la Fig. 3.58.

No se debe confundir el comando de escritura que genera el sistema SCADA con el comando de escritura que se genera en el Gateway, porque como se puede observar en la Fig. 3.58., el segundo es consecuencia del primero.

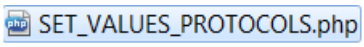


Fig. 3.84. Archivo para la ejecución de los comandos de escritura.
[***Fuente**: Propia*]

3.3.5 Componentes que integran el Espacio Control

La Fig. 3.85., muestra la infraestructura general de ESPACIO CONTROL

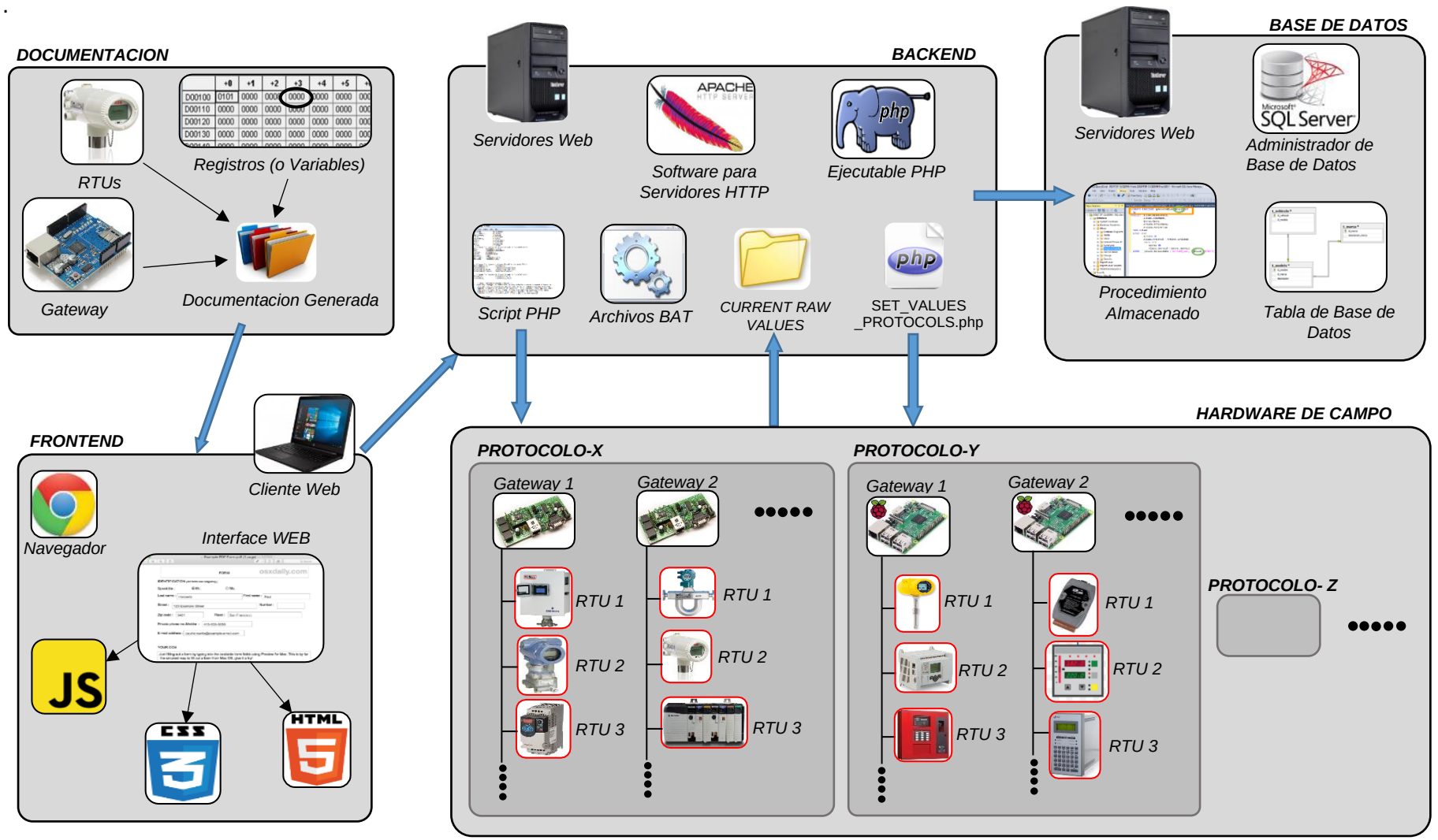


Fig. 3.85. Herramientas usadas para la construcción del Espacio Control. [***Fuente**: Propia*]

3.4 Monitoreo y Control de Procesos (Espacio Proceso)

3.4.1 Objetivo del Espacio Proceso

El ESPACIO PROCESO tiene por finalidad hacer que el usuario del sistema SCADA "interprete y maniobre la dinámica del proceso industrial de forma eficiente".

Las tareas que el ESPACIO PROCESO lleva a cabo para lograr tal objetivo, se ejecutarán sobre las INSTANCIAS de la CLASE VARIABLE definidos en el ESPACIO SISTEMA. El ESPACIO PROCESO ha sido creado esencialmente para ser usado por el operador de campo, a través de interfaces Web.

3.4.2 Interpretación y maniobra de la dinámica de un proceso industrial

A continuación, se explica que significa la frase: *interpretar y maniobrar la dinámica de un proceso industrial.*

Por ejemplo, en la Fig. 3.86., se muestra una pantalla web en la que se puede visualizar los valores en tiempo real que van adquiriendo algunas variables de la planta, que fueron extraídos del ESPACIO SISTEMA, que a su vez fueron extraídos del ESPACIO CONTROL.

La pantalla mostrada en la Fig. 3.86., permite al operador estar al tanto con los valores que van tomando los distintos parámetros físicos (variables analógicas) y estados físicos (variables digitales), pero también presentará algunas deficiencias como las siguientes:

- El operador no puede hacer una interpretación general de lo que ocurre en el proceso industrial, tal vez podría realizar tal interpretación si en una pantalla encuentra solo 10 variables, pero si encuentra, por ejemplo 100 variables en una sola pantalla, definitivamente se va a confundir, porque recordar el nombre de cada variable sería imposible.
- El operador tendría que estar al tanto de los valores que toma cada variable durante las 24 horas del día, para poder tomar acciones inmediatas cada vez que el valor de una variable supere su valor permitido. Como se puede apreciar, esta es una situación muy complicada para el operador, más aún si muchas variables superan sus respectivos umbrales al mismo tiempo.
- Otro asunto que se puede presentar es con respecto a las unidades de los parámetros físicos (variables analógicas), ya que el ESPACIO CONTROL pudo haber entregado, por ejemplo, un parámetro en psi, cuando el operador tal vez, lo requiere en bar.
- Igualmente sucede con los estados (variables de tipo digital que toman valores como ON-OFF, Activado-Desactivado, Open-Close etc.). En este caso, el ESPACIO CONTROL lo suele entregar como 1 o 0. El problema está en que si se entrega un 1

no se podrá determinar si este corresponde a un ON o a un OFF, esto también generará confusión al operador.

Como se puede observar el operador tiene muchos problemas con esta clase de interfaces. Por consiguiente, el ESPACIO PROCESO tiene que proporcionar interfaces que no tengan los problemas mencionados, recién en ese momento se podrá decir que el ESPACIO PROCESO permite interpretar un proceso industrial.

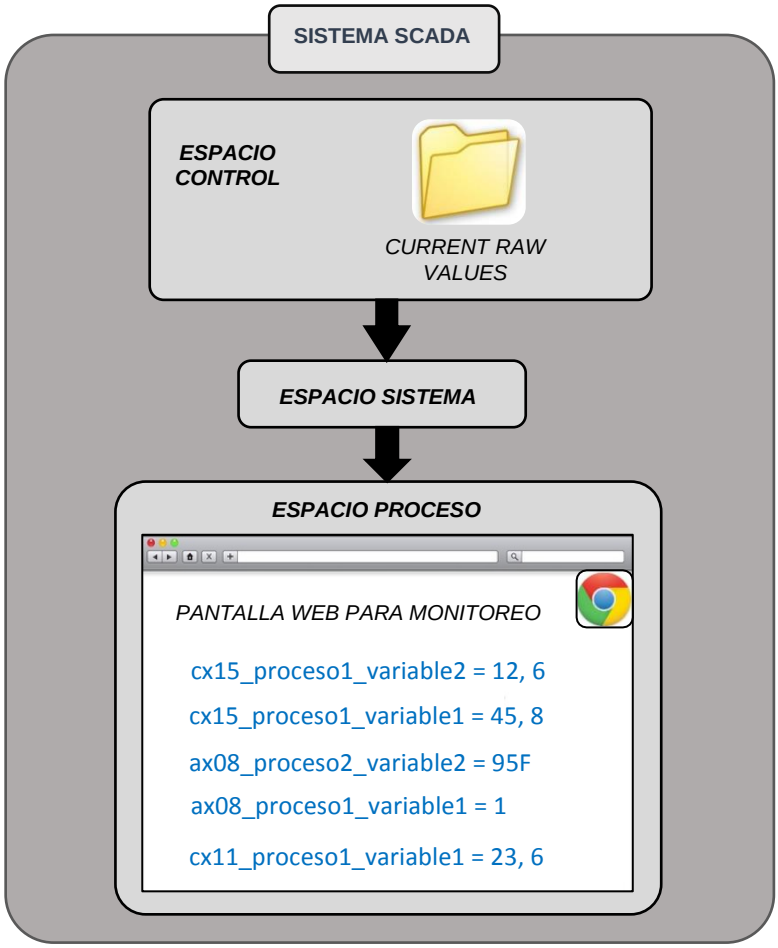


Fig. 3.86. Pantalla web que muestra los valores de algunas variables.
[***Fuente**: Propia*]

En la Fig. 3.87., se muestra una pantalla que permite establecer las variables a un valor deseado. Se debe observar que se trabaja de la mano con el ESPACIO CONTROL (ya que este proporciona la ruta para que el valor llegue a la memoria RAM de la RTU correspondiente, tal como se mostró en la Fig. 3.58.).

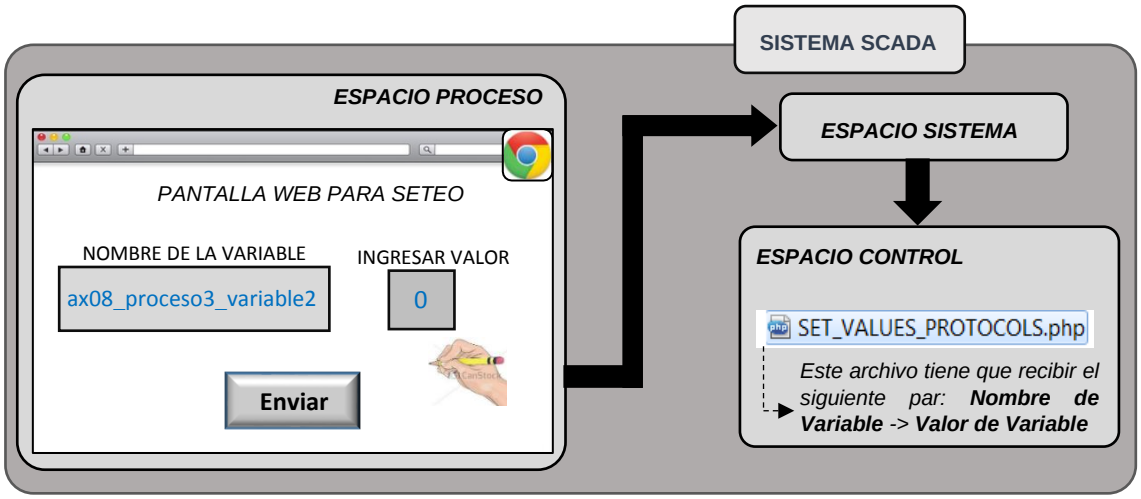


Fig. 3.87. Pantalla web que permite establecer el valor de una variable.
[***Fuente**: Propia*]

La pantalla de la Fig. 3.87., permite establecer valores a las variables, pero tiene algunas falencias como las que se describen a continuación:

- La pantalla anterior implicaría que el usuario del sistema SCADA debe de saber cuál es el nombre de la variable sobre la que quiere hacer el establecimiento de un valor.
- En el caso de los parámetros físicos (variable analógica) forzaría al usuario a trabajar con las unidades que manejan los dispositivos asociados a esas variables, por ejemplo, si se quiere establecer la presión de un tanque a 38 psi, pero resulta que el dispositivo que controla la presión del tanque trabaja con bar, el usuario tendría que calculando el equivalente en bar permanentemente. Esto implicaría pérdida de tiempo.
- En el caso de los estados físicos (variables digitales), el usuario tendría que conocer si un 1 corresponde a un ON o a un OFF, o si un 0 corresponde a un ABIERTO o CERRADO, esta también es otra situación en la cual el usuario pierde tiempo.
- Otro problema que se podría presentar es con respecto al rango de los valores permitidos, ya que según la pantalla que se muestra en la Fig. 3.87., el usuario estaría en libertad de colocar cualquier valor sin restricción alguna, lo cual podría resultar peligroso.

Como se puede apreciar, el hecho que un operador pueda escribir un valor sobre una variable no quiere decir que pueda maniobrar un proceso industrial de forma eficiente, recién cuando el ESPACIO PROCESO (que es el que vamos a desarrollar en esta sección) proporcione interfaces que no tengan los problemas que acabamos de mencionar en lo referente a la escritura, entonces se podrá decir que ESPACIO PROCESO permite maniobrar un proceso.

3.4.3 Herramientas del Espacio Proceso

Para que se puedan superar los problemas mostrados en la subsección anterior, y de esta forma hacer que el ESPACIO PROCESO pueda lograr su objetivo. Se ha dispuesto que se deben proporcionar las siguientes herramientas:

- Linealización y Titulación de valores.
- Estudio de las Alarmas y rango de valores permitidos para el establecimiento de valores.
- Interfaces que representan el proceso industrial.
- Tendencias.

Los software que encontramos en el mercado y que ofrecen estas herramientas son los que comúnmente son conocidos como sistemas SCADA, como por ejemplo el RSView32 (Rockwell Software – 2001 – "RSView32 User's Guide") [15].

En las siguientes subsecciones se irán mencionado en detalle cada uno de estas herramientas, y cómo fueron aplicadas a la Tesis.

3.4.4 Linealización y Titulación de variables

a) Linealización de parámetros físicos

Aplica solo para parámetros físicos (variables analógicas).

Esta herramienta básicamente lo que hace es convertir los valores que el ESPACIO CONTROL obtuvo (cuyas unidades de medición depende del dispositivo de campo), a valores con unidades de medición con las que esté familiarizado el operador.

Para ello, simplemente se realiza una linealización tal como se muestra en la ecuación 3.1.

$$g(x) = ax + b \tag{3.1}$$

Donde:

$g(x)$: Representa valores de los parámetros físicos en unidades de medición legibles para el operador de campo. Este valor será con el que trabajará el ESPACIO PROCESOS.

x : Representa los valores de los parámetros físicos en unidades de medición establecidos por sus respectivos dispositivos de campo. Este valor fue proporcionado por el ESPACIO CONTROL.

a : Este valor se denomina Escalador.

b : Este valor se denomina Offset.

Los escaladores y offsets de los parámetros físicos deben ser establecidos por el operador del sistema SCADA. Por ejemplo, supongamos que en un determinado punto del proceso industrial se mide la temperatura, que, a su vez, es entregada por su respectivo dispositivo de campo en Celsius, pero el operador del sistema SCADA la requiere en Fahrenheit, en ese caso se procedería de la siguiente manera:

Sabemos que: $F = 1.8\ C + 32$ (3.2)

Donde las variables de la ecuación (3.2) representan lo siguiente:

F: Temperatura en Fahrenheit (requerido por el operador del sistema SCADA).

C: Temperatura en Celsius (entregado por el dispositivo de campo).

Entonces de la ecuación (3.1) y (3.2) se deduce que:

Escalador a = 1.8 y

Offset b = 32

En caso de que el dispositivo de campo entregue los valores de una determinada variable en las mismas unidades que las requeridas por el operador del sistema SCADA:

Escalador a = 1 y

Offset b = 0

Estos valores mencionados líneas arriba (Escalador a =1 y Offset b =0) son los que el sistema SCADA ofrece por defecto. Es importante mencionar que el proceso de Linealización es periódico, ya que el valor de los parámetros físicos cambia con el tiempo.

b) Titulación de estados físicos

Aplica solo a estados físicos (variables digitales).

Como ya se había mencionado anteriormente, el ESPACIO CONTROL en el caso de las variables digitales solo maneja el 1 y el 0, entonces lo que el ESPACIO PROCESO hará es convertir los 1s y 0s en valores legibles para el usuario del sistema SCADA, como por ejemplo: ON-OFF, Close-Open, Abierto-Cerrado, Energizado-Desenergizado, etc.

En la ecuación (3.3) se muestra cómo se realiza un proceso de Titulación.

$$g(x) = Titulacion\_para\_1 \ \ si \ \ x = 1 \ \ \text{ó}$$
$$g(x) = \ Titulacion\_para\_0 \ \ si \ \ x = 0 \quad (3.3)$$

Donde:

$g(x)$: Representa los valores de los estados físicos en formato legible para el operador del sistema SCADA, este valor es con el que trabajará el ESPACIO PROCESO.

x : Representa los valores de los estados físicos en formatos de 1s y 0s. Este valor fue proporcionado por el ESPACIO CONTROL.

$Titulacion\_para\_1$ = Es la forma como se representará el “1” que entregó el ESPACIO CONTROL, puede tomar valores como OFF, ON, CLOSE, OPEN etc.

$Titulacion\_para\_0$ = Es la forma como se representará el "0" que entregó el ESPACIO CONTROL, puede tomar valores como: OFF, ON, CLOSE, OPEN etc.

El valor de $Titulacion\_para\_1$ y $Titulacion\_para\_0$ debe ser establecido por el operador del sistema SCADA.

Por ejemplo, supongamos que hay una variable asociada al estado de una válvula, y su correspondiente dispositivo de campo nos entrega un 0 cuando está abierto y nos entrega un 1 cuando está cerrado, entonces:

$Titulacion\_para\_0$ = ABIERTO (u OPEN).

$Titulacion\_para\_1$ = CERRADO (o CLOSE).

El sistema SCADA establecerá por defecto lo siguiente:

$Titulacion\_para\_0$ = 0.

$Titulacion\_para\_1$ = 1.

Es importante mencionar que el proceso de Titulación es periódico, ya que el valor de los estados físicos cambia con el tiempo.

c) Casos particulares de Linealización y Titulación

A continuación, se menciona los casos en los cuales no se realizará ni la Linealización ni la Titulación:

Linealización:

En el caso de la Linealización, esta no se producirá cuando el valor de un parámetro físico recibido desde el ESPACIO CONTROL no es un valor numérico, por consiguiente, el ESPACIO PROCESO trabajará con un valor obtenido tal como se muestra en la ecuación (3.4):

$$g(x) = x \tag{3.4}$$

Donde:

$g(x)$: Valor con el que trabajará el ESPACIO PROCESO.

x: Información entregada por el ESPACIO CONTROL, no es un número.

Las razones por la cual el ESPACIO CONTROL no entregaría un valor numérico pueden ser muchas, por lo general sería, cuando ocurra un error en la comunicación, ya que en esos casos el ESPACIO CONTROL suele entregar una descripción del error correspondiente, como por ejemplo "Modbus: TIMEOUT", lo cual es una cadena de texto y no un valor numérico.

Titulación:

En el caso de la Titulación, esta no se lleva a cabo cuando el valor de un estado físico proporcionado por el ESPACIO CONTROL es un valor distinto de 1 o 0, por consiguiente, el ESPACIO PROCESO trabajará, con un valor obtenido tal como se muestra en la ecuación (3.5):

$$g(x) = x \tag{3.5}$$

Donde:

$g(x)$: Valor con el cual trabajará el ESPACIO PROCESO.

x: Información entregada por el ESPACIO CONTROL, no es ni 1 ni 0.

Las razones por la cual el ESPACIO CONTROL no entregaría ni 1 ni 0 pueden ser muchas, por lo general seria, cuando ocurra un error en la comunicación, ya que en esos casos el ESPACIO CONTROL suele entregar una descripción del error correspondiente, como por ejemplo “Modbus: TIMEOUT”, lo cual es una cadena de texto y no un 1 ni 0.

3.4.5 Análisis de alarmas

a) Condición para el análisis de las alarmas en un proceso industrial

En la Tesis para que el análisis de las alarmas se pueda llevar a cabo, es INDISPENSABLE que no se produzcan los errores que se mencionaron en la sección 3.3.4 parte c), en caso de producirse esos errores, no se desarrollará ninguno de los procedimientos mostrados a continuación.

b) Análisis de alarmas para parámetros físicos (variables analógicas)

Regiones de las alarmas y sus umbrales:

Los valores de los parámetros físicos trabajados por el ESPACIO PROCESO ya están linealizados y se muestran en la Ecuación 3.1 como $g(x)$, se pueden representar mediante una curva que va cambiando con el tiempo tal como se puede observar en la Fig. 3.88. De acuerdo a los estudios realizados por especialistas en control de procesos y la experiencia de los operadores se establecerán ciertos límites o umbrales para los parámetros físicos, que una vez sobrepasados, activaran algún tipo de señal o alarma que indique al operador que tiene que tomar algún tipo de acción.

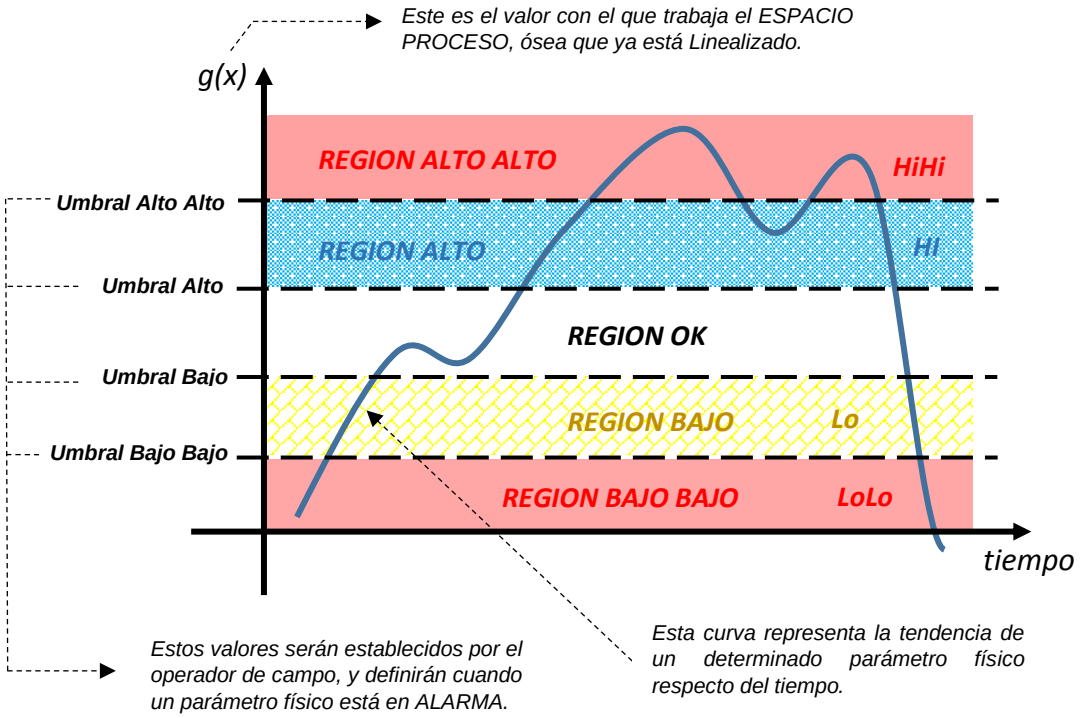


Fig. 3.88. Tendencia de un parámetro físico y las distintas regiones que puede ir ocupando, de acuerdo a los umbrales establecidos. [***Fuente**: Propia*]

Registros "In-Alarm" y "Out of Alarm":

Los parámetros físicos pueden estar en estado de alarma durante el tiempo en que la curva esté ubicada en una región de alarma que puede ser Alta-Alta, Alta, Baja y Baja-Baja (ver Fig. 3.88.), pero una situación distinta es cuando se registra la alarma en la Base de Datos. Esto solo ocurre en determinadas transiciones, que son conocidas como **In-Alarm**, y se muestran en la Tabla. 3.2.

Igualmente ocurre con el estado de No Alarma, que ocurre durante el tiempo en el cual la curva está en la región OK, ver Fig. 3.88., pero No Alarma solo es registrado al momento de determinadas transiciones, que es conocida como Out of Alarm, y se muestra en la Tabla. 3.2.

De aquí, se puede concluir que las alarmas no están permanentemente registrándose, sino que lo hacen en situaciones muy puntuales.

Tanto los registros **In-Alarm** como los **Out-of-Alarm** son los que más adelante se mostraran en los Banner de Alarmas.

Tabla. 3.2. Transiciones en las que se produce el registro de una alarma en una Base de Datos. [***Fuente**: Propia*]

Estado de Alarma del muestreo actual. ↓ / → *Estado de Alarma del muestreo anterior.*	*HiHi*	*Hi*	*OK*	*Lo*	*LoLo*
HiHi	**N/A**	**N/A**	Out of Alarm	In-Alarm	In-Alarm
Hi	In-Alarm	**N/A**	Out of Alarm	In-Alarm	In-Alarm
OK	In-Alarm	In-Alarm	**N/A**	In-Alarm	In-Alarm
Lo	In-Alarm	In-Alarm	Out of Alarm	**N/A**	In-Alarm
LoLo	In-Alarm	In-Alarm	Out of Alarm	**N/A**	**N/A**

Cuando se habla de parámetro físico, se debe tener presente que se está haciendo referencia a una INSTANCIA de la CLASE VARIABLE del ESPACIO SISTEMA, cuando se realiza la instanciación se determina si el objeto es un parámetro físico (variable analógica) o estado físico (variable digital), ver la Fig. 3.18.

c) Análisis de alarmas para estados físicos (variables digitales)

Curva de un estado físico:

Los valores de los estados físicos trabajados por el ESPACIO PROCESO se pueden representar mediante una curva digital que va cambiando con el tiempo tal como se puede ver en la Fig. 3.89.

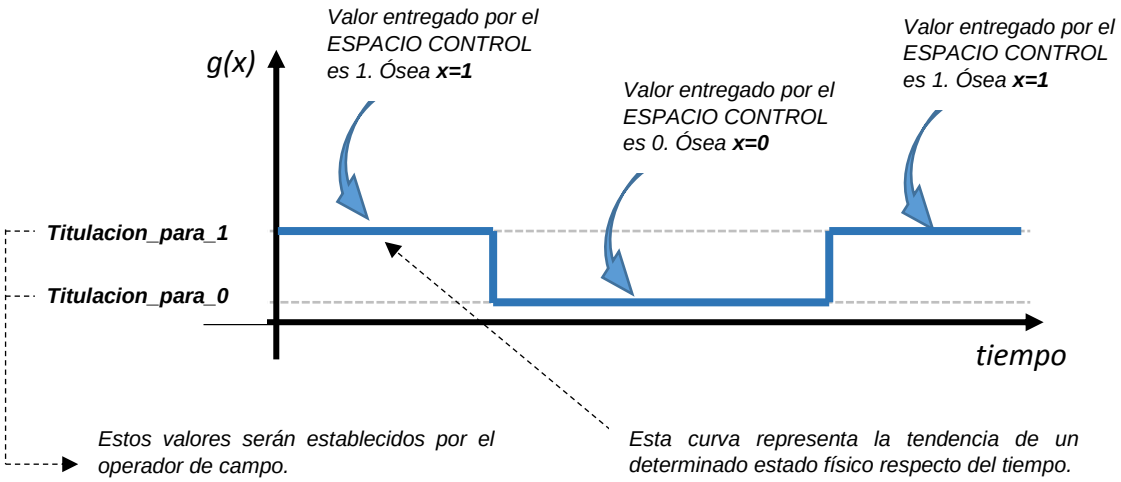


Fig. 3.89. Tendencia de un estado físico. [***Fuente**: Propia*]

Estableciendo las alarmas:

Un estado físico estará en alarma dependiendo del valor que le haya sido entregado por el ESPACIO CONTROL, es decir, x = 1 o x = 0, ya que como se recuerda el ESPACIO CONTROL solo proporciona estos 2 valores.

La persona que decide con qué valor (x = 1 o x = 0) el estado físico entra en alarma es el operador de campo. En la Fig. 3.90. y la Fig. 3.91., se mostrará las regiones de alarmas dependiendo de la elección del usuario.

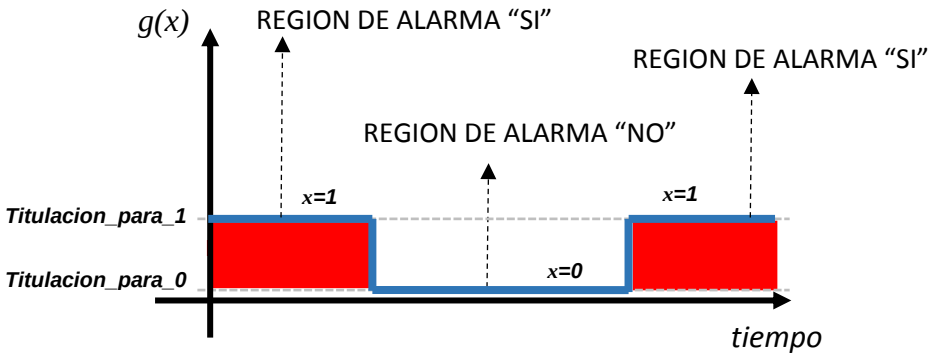


Fig. 3.90. Región de alarma cuando (x = 1). [***Fuente**: Propia*]

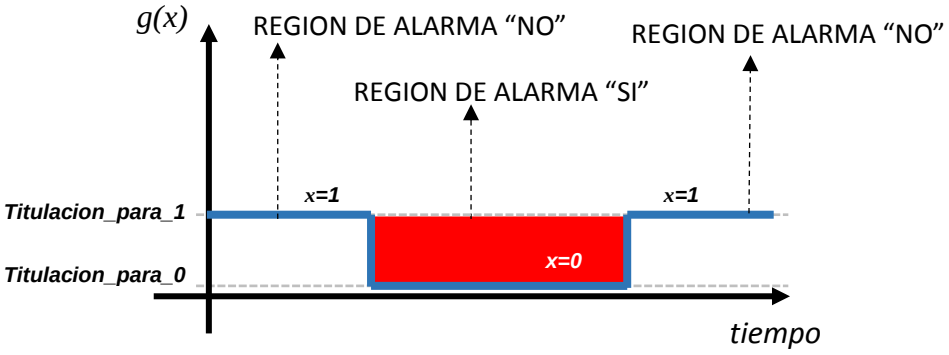


Fig. 3.91. Región de alarma cuando (x = 0). [***Fuente**: Propia*]

Registros “In-Alarm” y “Out of Alarm”:

Si bien un estado físico está en alarma durante el tiempo en que la curva esté ubicada en una “región de alarma” tal como se muestra en la Fig. 3.90. y la Fig. 3.91., pero solo serán registradas en la Base de Datos en determinadas transiciones, conocidas como “In-Alarm” y se muestran en la Tabla. 3.3.

También existen las transiciones “Out of Alarm” que serán registradas en la Base de Datos y que hacen referencia al momento en el que se dejó la región de alarma y se muestran en la Tabla. 3.3.

Entonces, se puede concluir que las alarmas no se registran permanentemente, sino que lo hacen en situaciones muy puntuales.

Tanto los registros In-Alarm como los Out-of-Alarm se mostrarán más adelante en los Banner de Alarmas.

Tabla. 3.3. Registro de una alarma en una Base de Datos, válido para la Fig. 3.90. y la Fig. 3.91. [***Fuente**: Propia*]

Estado de Alarma del muestreo actual. → / *Estado de Alarma del muestreo anterior.* ↓	*No*	*Si*
No	**N/A**	In-Alarm
Si	Out of Alarm	**N/A**

d) Condición de Alarma deshabilitada

Cuando el operador decide deshabilitar una alarma para un parámetro físico o para un estado físico, lo que ocurrirá es que no se producirá registro de las alarmas en la Base de Datos.

3.4.6 Registros temporales de las variables

Los registros temporales de las variables, son archivos de texto plano con el nombre de la variable, es decir, con el nombre de la INSTANCIA de la CLASE VARIABLE, que fue creada en el ESPACIO SISTEMA, en cuyo interior estará la siguiente información correspondiente a la variable, sea un parámetro físico o un estado físico, ver la Fig. 3.92.

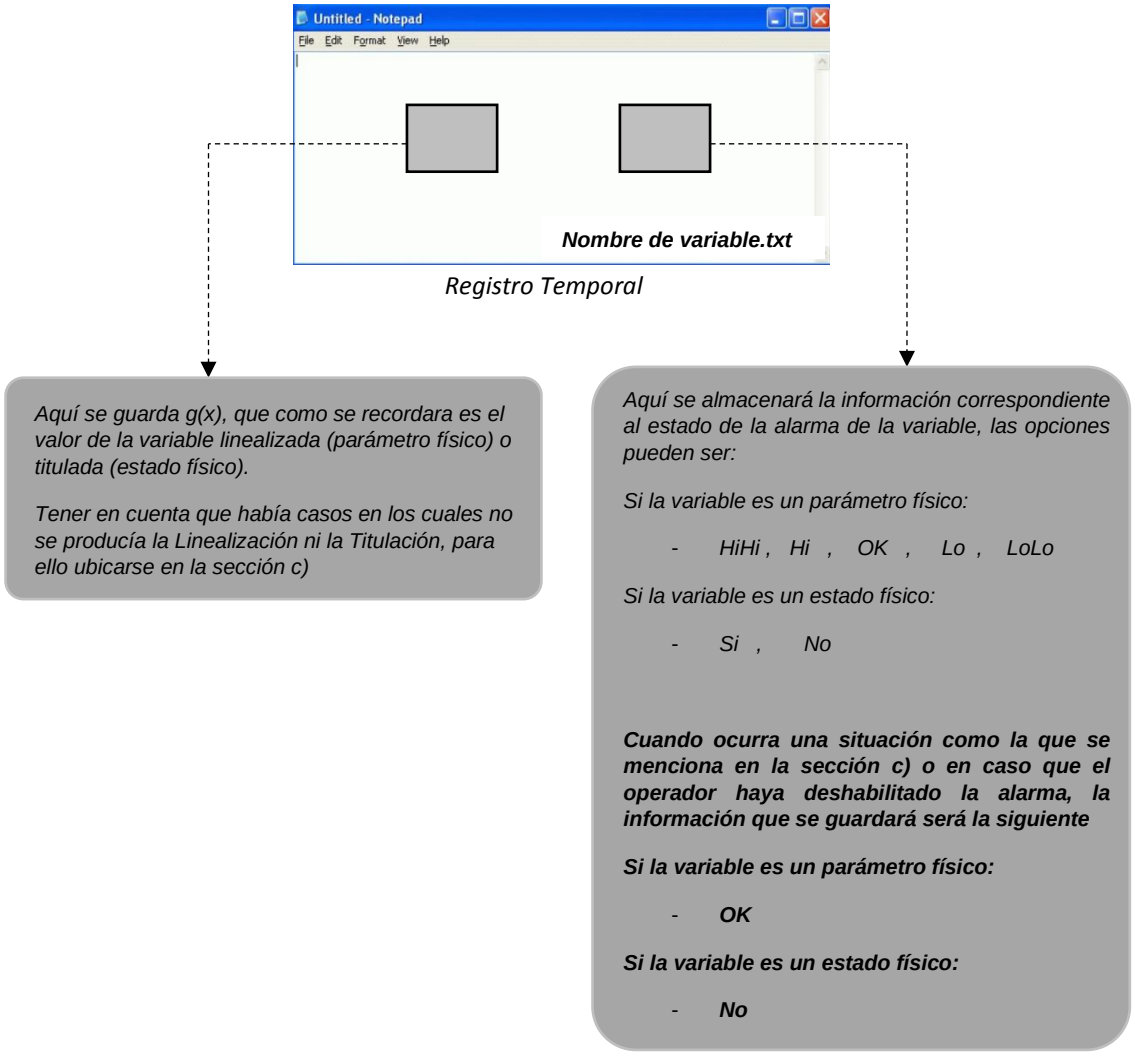


Fig. 3.92. Contenido interno del archivo de texto correspondiente a una variable. [***Fuente**: Propia*]

Es importante mencionar que habrá un archivo de texto por cada variable y que estos archivos actualizarán sus contenidos internos de forma periódica, por eso toman el nombre Registros Temporales, porque solo guarda la información del momento. La razón de la creación de estos archivos temporales es, que las interfaces gráficas, que más adelante se verán, puedan acceder a la información de las variables en tiempo real.

3.4.7 Contenido de los registros de alarmas en la Base de Datos y contenido en los Registros Temporales

La Fig. 3.93., muestra de forma detallada como va cambiando el contenido de los Registros Temporales y el contenido de los registros de las alarmas que se van agregando a la Base de Datos (ver la tabla 3.2 como referencia), incluido cuando el ESPACIO CONTROL entrega un mensaje de error correspondiente a un parámetro físico (variables analógicas).

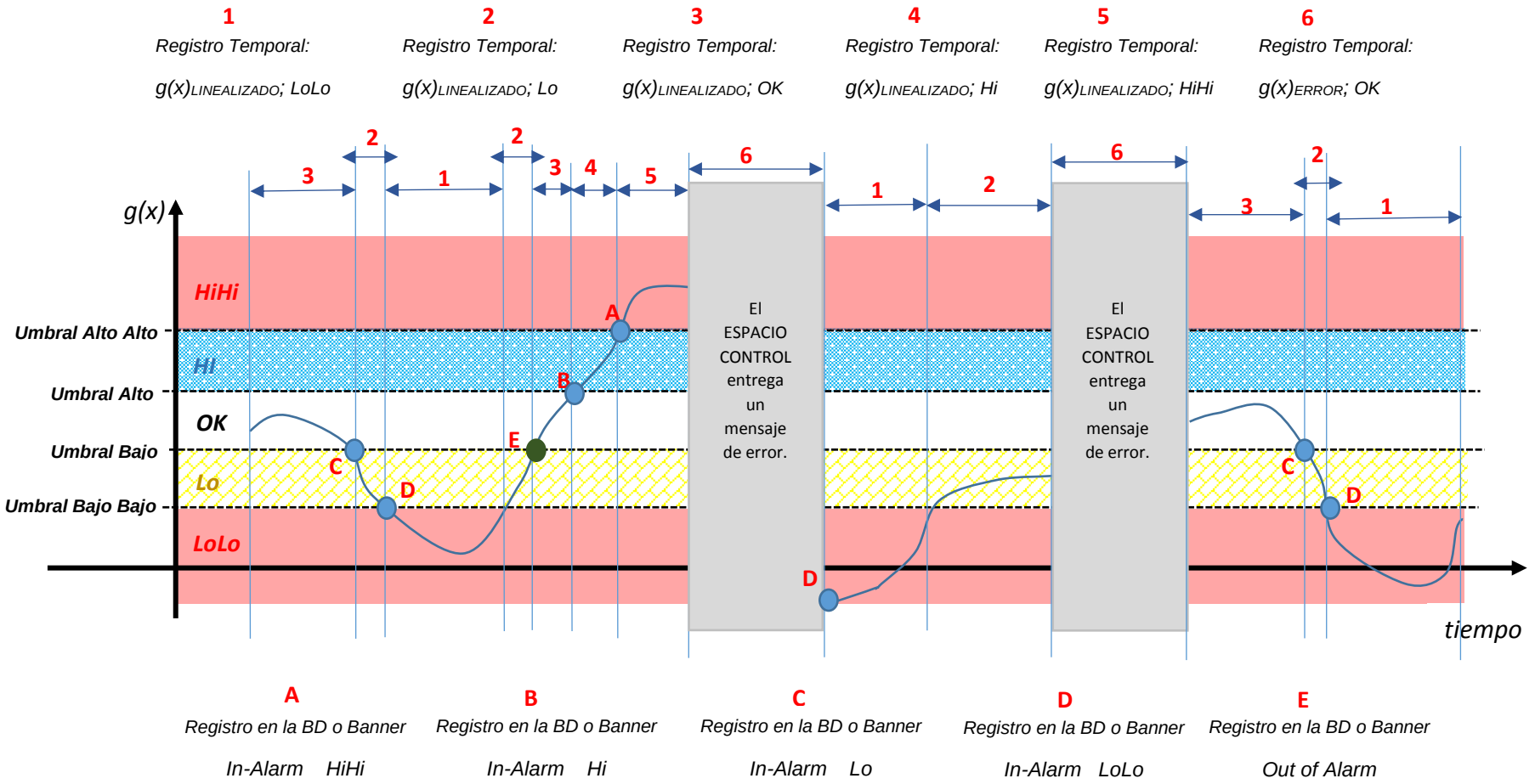


Fig. 3.93. Variación de un parámetro físico con el tiempo y los valores que va tomando su respectivo registro temporal y en qué momentos se registran sus alarmas en la Base de Datos. [***Fuente**: Propia*]

La Fig. 3.94., muestra de forma detallada como va cambiando el contenido de los Registros Temporales y el contenido de los registros de las alarmas que se van agregando a la Base de Datos (ver la tabla 3.2 como referencia), incluido cuando el operador del SCADA inhabilita la alarma de un parámetro físico.

La Fig. 3.95., muestra de forma detallada como va cambiando el contenido de los Registros Temporales y el contenido de los registros de las alarmas que se van agregando a la Base de Datos (ver la tabla 3.3 como referencia), incluido cuando el ESPACIO CONTROL entrega un mensaje de error correspondiente a un estado físico (variable digital). En este ejemplo se supondrá, que el operador decidió que el estado de alarma ocurre cuando el ESPACIO CONTROL entrega un 1, es decir, x = 1.

La Fig. 3.96., muestra de forma detallada como va cambiando el contenido de los Registros Temporales y el contenido de los registros de las alarmas que se van agregando a la Base de Datos (ver la tabla 3.3 como referencia), incluido cuando el operador del SCADA inhabilita la alarma de un estado físico (variable digital). En este ejemplo se supondrá, que el operador decidió que el estado de alarma ocurre cuando el ESPACIO CONTROL entrega un 0, es decir, x = 0.

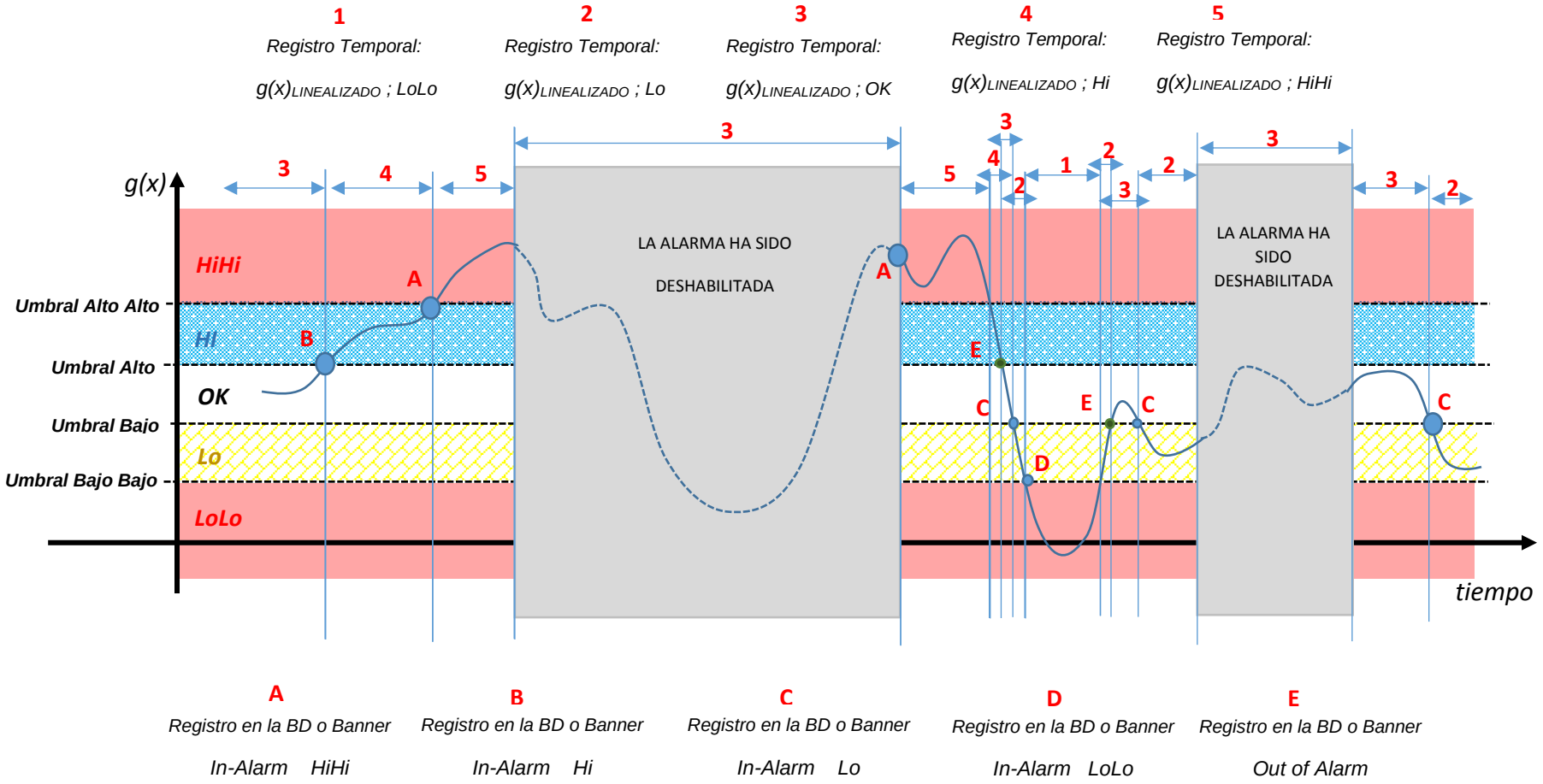


Fig. 3.94. Variación de un parámetro físico con el tiempo y los valores que va tomando su respectivo registro temporal y en qué momentos se registran sus alarmas en la Base de Datos. [***Fuente**: Propia*]

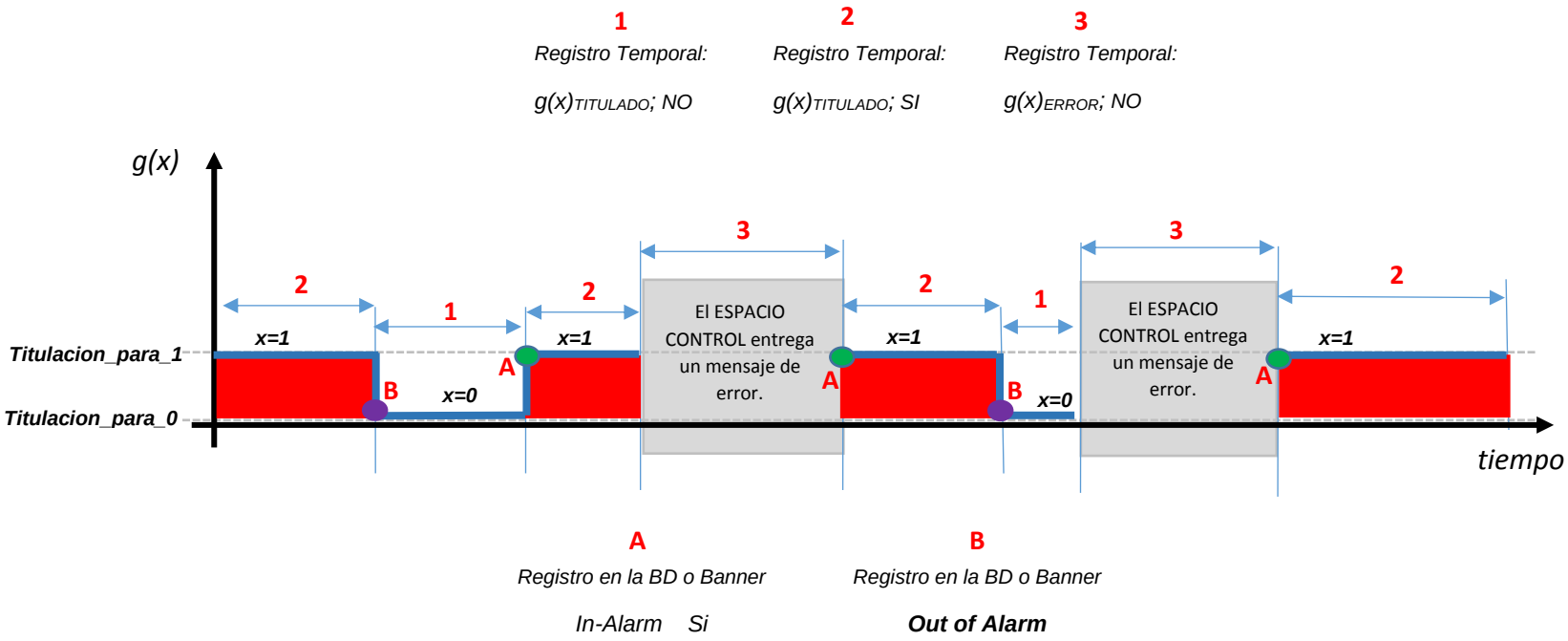


Fig. 3.95. Variación de un estado físico con el tiempo y los valores que va tomando su respectivo registro temporal y en qué momentos se registran sus alarmas en la Base de Datos. [***Fuente**: Propia*]

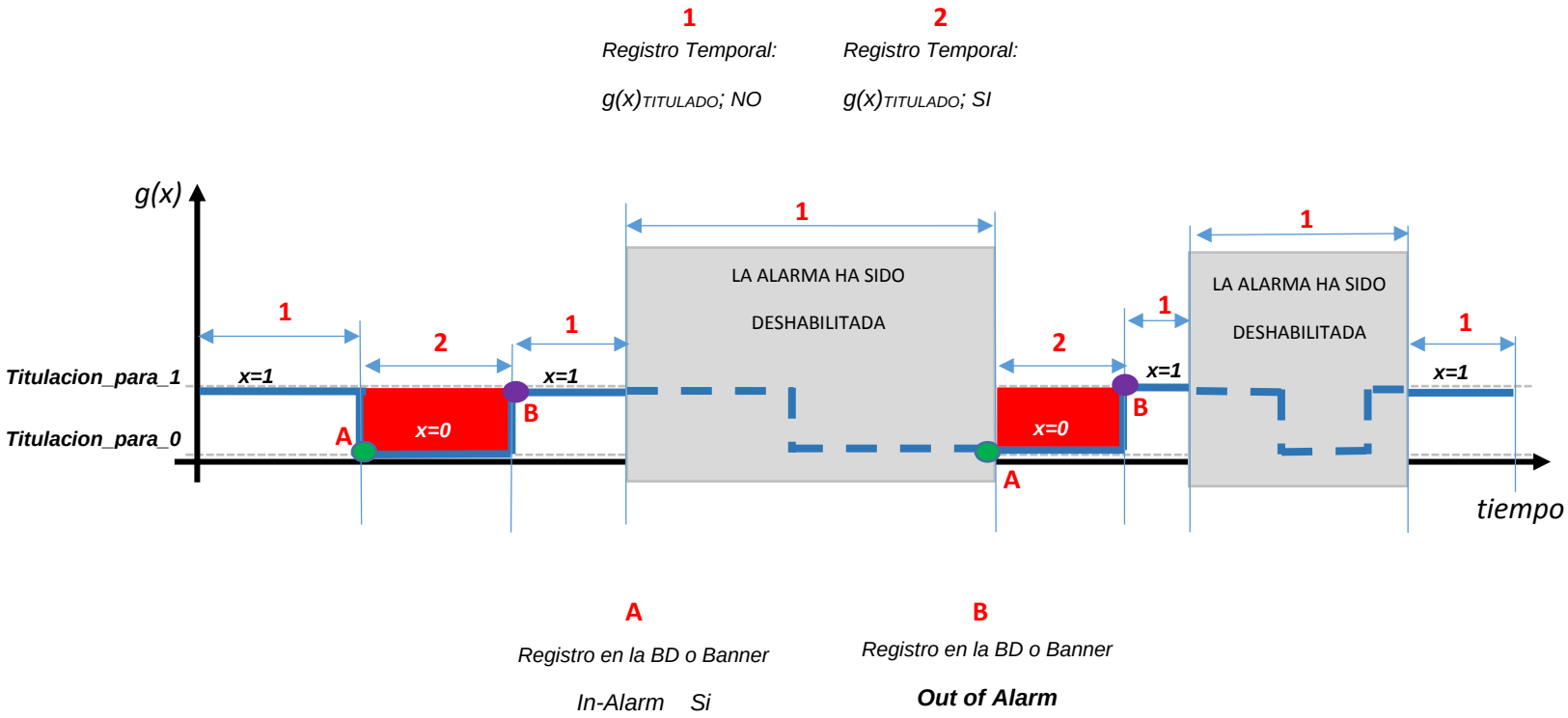


Fig. 3.96. Variación de un estado físico con el tiempo y los valores que va tomando su respectivo registro temporal y en qué momentos se registran sus alarmas en la Base de Datos. [***Fuente**: Propia*]

3.4.8 Banner de Alarmas

Es una herramienta incorporada en la interfaz de procesos que le permitirá al operador de campo visualizar en tipo real, las alarmas que se están produciendo en determinada planta, ver Fig. 3.97.

En el banner de alarmas un operador de campo podrá encontrarse con 3 tipos de registros alarmas:

- **In-Alarm.** Es un registro que se crea cuando la variable entra en alarma y solo se produce en la transición hacia una alarma. Esto aplica a un parámetro físico o a un estado físico.
- **Out of Alarm**. Es un registro que se crea cuando una variable sale de alarma y solo se produce durante la transición. Esto aplica a un parámetro físico o a un estado físico.

  Estos 2 tipos de registros son creados luego del proceso de linealización de parámetros físicos o, del proceso de titulación de estados físicos, tal como se puede observar en las Fig. 3.93., Fig. 3.94., Fig. 3.95. y Fig. 3.96.
- **Acknowlegement.** Este registro es generado desde el mismo banner a pedido del usuario, y tiene la finalidad de informar que determinada alarma ya fue reconocida y que se tomarán las acciones correspondientes, lo cual no significa que las condiciones que activaron alarma hayan desaparecido.

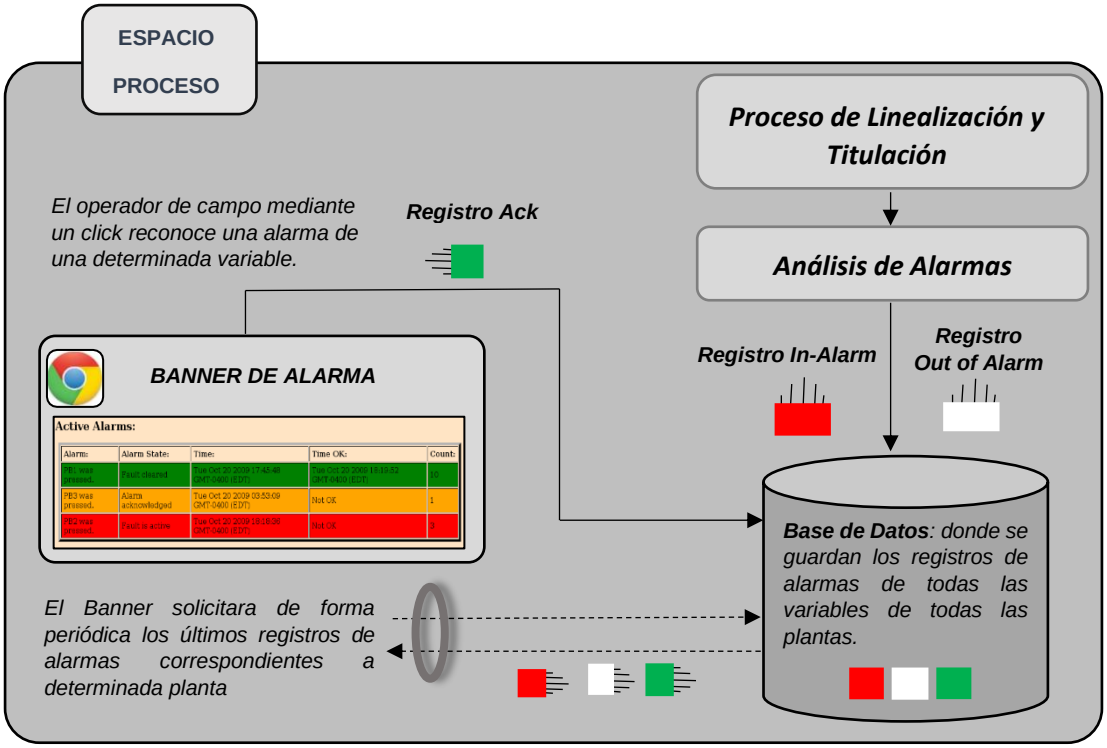


Fig. 3.97. Banner de alarmas y la base de datos que almacena los registros de las alarmas. [***Fuente**: Propia*]

a) Formato de colores de acuerdo al tipo de alarma

Ahora lo que se hará es establecer indicadores de colores que ayudaran al operador a saber que variables se encuentran en alarma, que variables ya salieron de alarma, que variables están en alarma, pero ya fueron reconocidas, y en el mejor de los casos, que variables ya salieron del estado de alarma y ya fueron reconocidas.

A continuación, se mostrarán imágenes (Fig. 3.98., Fig. 3.99., Fig. 3.100. y Fig. 3.101.) del banner del sistema SCADA del proyecto de Tesis con cada uno de los formatos mencionados.

Nota**:** Si el lector observa detenidamente en la columna izquierda del banner, podrá percatarse que hay una campana (la cual parpadea y emite sonido mientras una variable esté en alarma), la imagen y el sonido desaparecen cuando la variable está fuera de alarma o cuando la alarma es reconocida por el operador de campo.

Id Alarma:	Proceso:	Variable:	Tipo de Alarma:	Severidad:	Descripcion:	Fecha y Hora:	Id_Variable:
1566	cx11_proceso1	cx11_proceso1_variable1	In Alarm	HiHi	Cuidado, el tanke puede explotar !!!!	07 Dec 2017 11:49:15	VR000010
1565	cx11_proceso1	cx11_proceso1_variable1	In Alarm	Hi	La presión esta incrementando.	07 Dec 2017 11:48:57	VR000010
1564	cx11_proceso1	cx11_proceso1_variable1	In Alarm	LoLo	La presión es extremadamente baja !!!	07 Dec 2017 11:48:43	VR000010
1563	cx11_proceso1	cx11_proceso1_variable1	In Alarm	Lo	La presión es baja.	07 Dec 2017 11:48:22	VR000010

Fig. 3.98. Banner de alarmas con registros correspondientes a una variable que se encuentra actualmente en alarma. [***Fuente**: Propia*]

Fig. 3.99. Banner de alarmas con registros correspondientes a una variable que se encuentra actualmente fuera alarma y que no ha sido reconocida por el operador de campo. [***Fuente**: Propia*]

Fig. 3.100. Banner de alarmas con registros correspondientes a una variable que se encuentra actualmente en alarma y que acaba de ser reconocida por el operador de campo. [***Fuente**: Propia*]

Fig. 3.101. Banner de alarmas con registros correspondientes a una variable que se encuentra actualmente fuera de alarma y que acaba de ser reconocida por el operador de campo. [***Fuente**: Propia*]

3.4.9 Interfaces de los procesos industriales

Hasta el momento se dispone de los registros temporales que se mostraron en la Fig. 3.92., y que contienen en su interior los valores de la variables y si estan en alarma o no. Pero estos registros temporales, que no son más que block de notas, por si solos no son de mucha utilidad, pero se vuelven realmente potentes cuando trabajan de la mano con las intefaces web, que no son más que representaciones gráficas que se enlazan a sus respectivos registros temporales con la finalidad de lograr que el operador de campo pueda tener conocimiento del funcionamiento de todo el proceso industrial de forma más rápida, ya que no es novedad que el ser humano entiende las cosas de forma más rápida cuando las ve de forma gráfica.

En la Fig. 3.102., se puede ver de manera sencilla el mecanismo de funcionamiento de las interfaces de monitoreo.

En la Fig. 3.103., se ve cómo actuan las pantallas cuando el operador de campo establece el valor de una variable en un valor definido. Las interfaces que representarán de forma gráfica los procesos industriales han sido elaboradas con una herramienta llamada Inkscape, que permite hacer gráficos vectoriales [16].

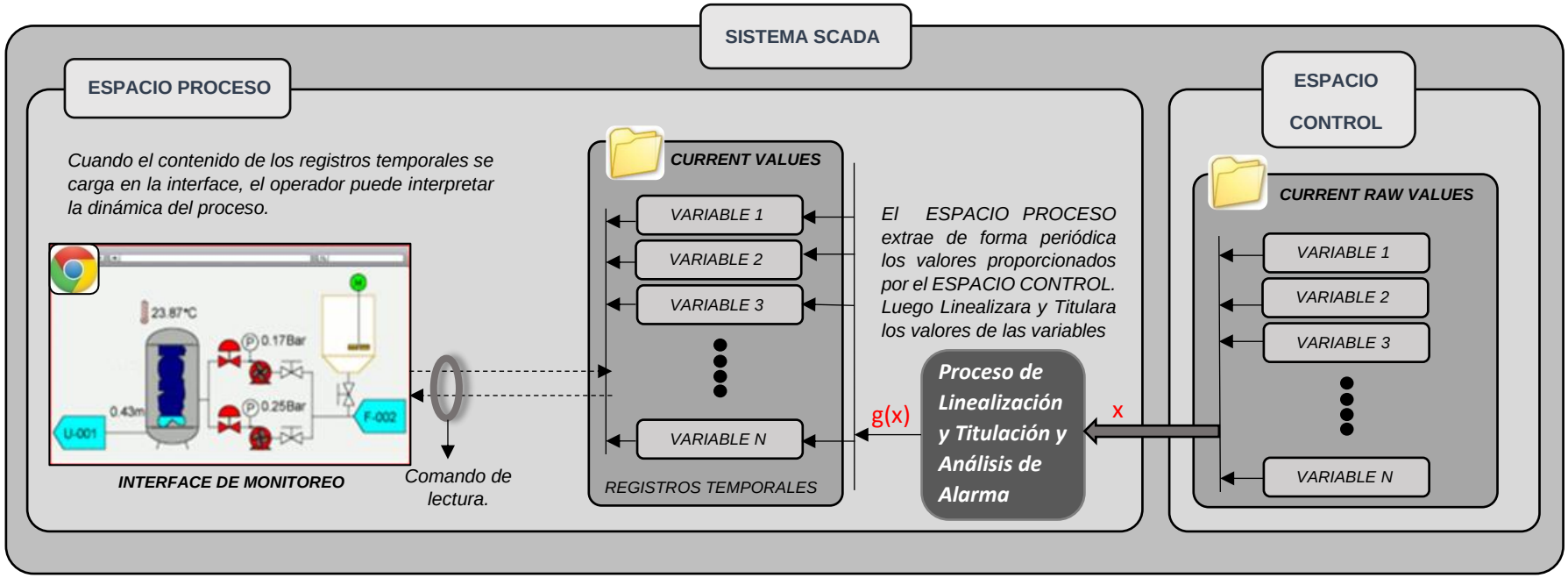


Fig. 3.102. Ruta que siguen los datos extraídos desde el ESPACIO CONTROL hasta las pantallas de monitoreo. [***Fuente**: Propia*]

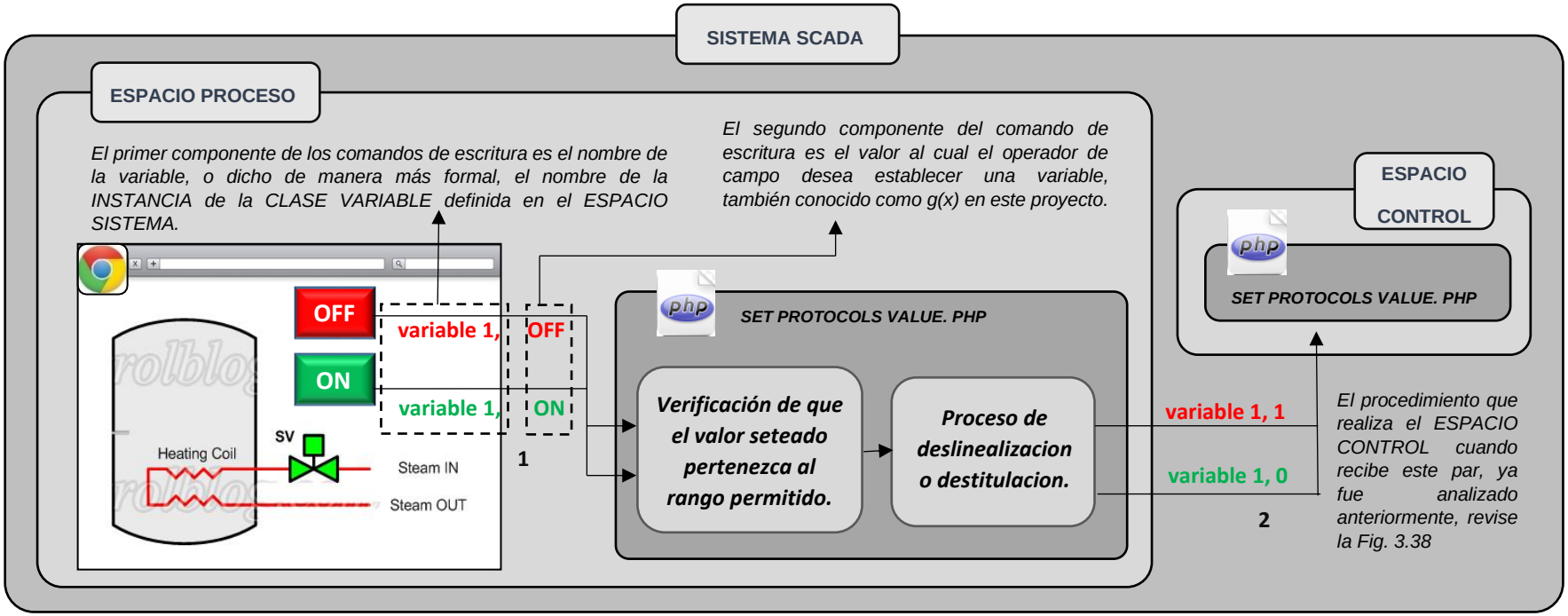


Fig. 3.103. Ruta que siguen los datos que se envían en un comando de escritura. [***Fuente**: Propia*]

3.4.10 Interfaces usadas en la Tesis

A continuación, se muestran las interfaces y pantallas gráficas que se usaron para la construcción del ESPACIO PROCESO.

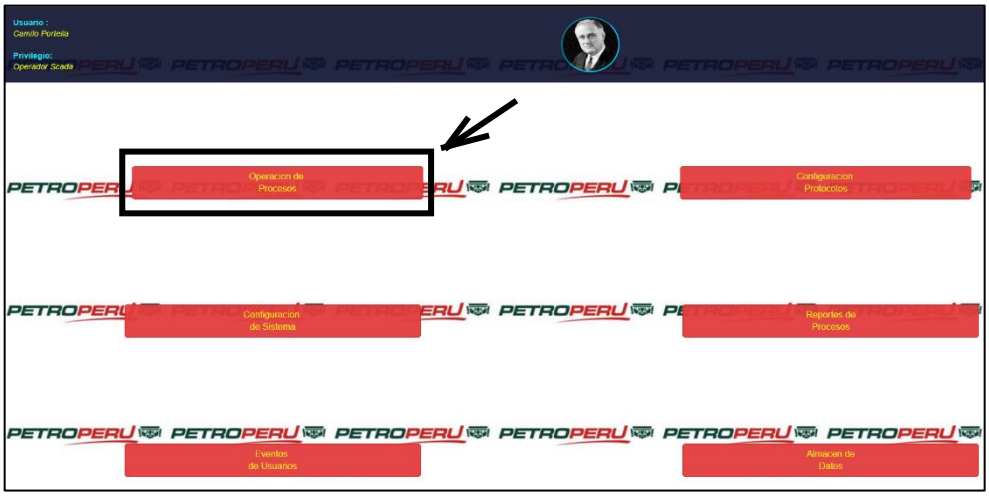


Fig. 3.104. El botón para configurar lo relacionado al Espacio Proceso. [***Fuente**: Propia*]

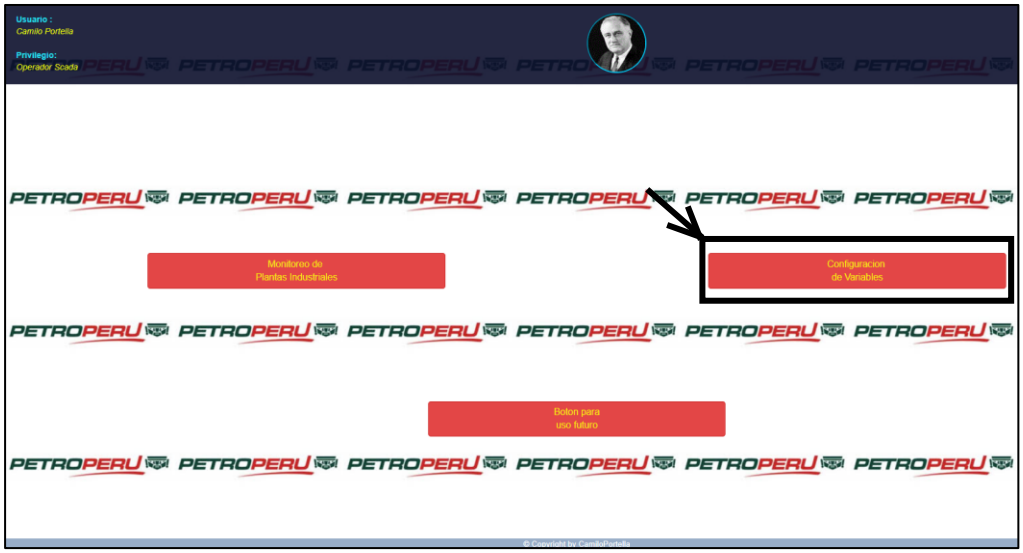


Fig. 3.105. El botón para configurar algunos atributos de las Instancias de la Clase Variable [***Fuente**: Propia*]

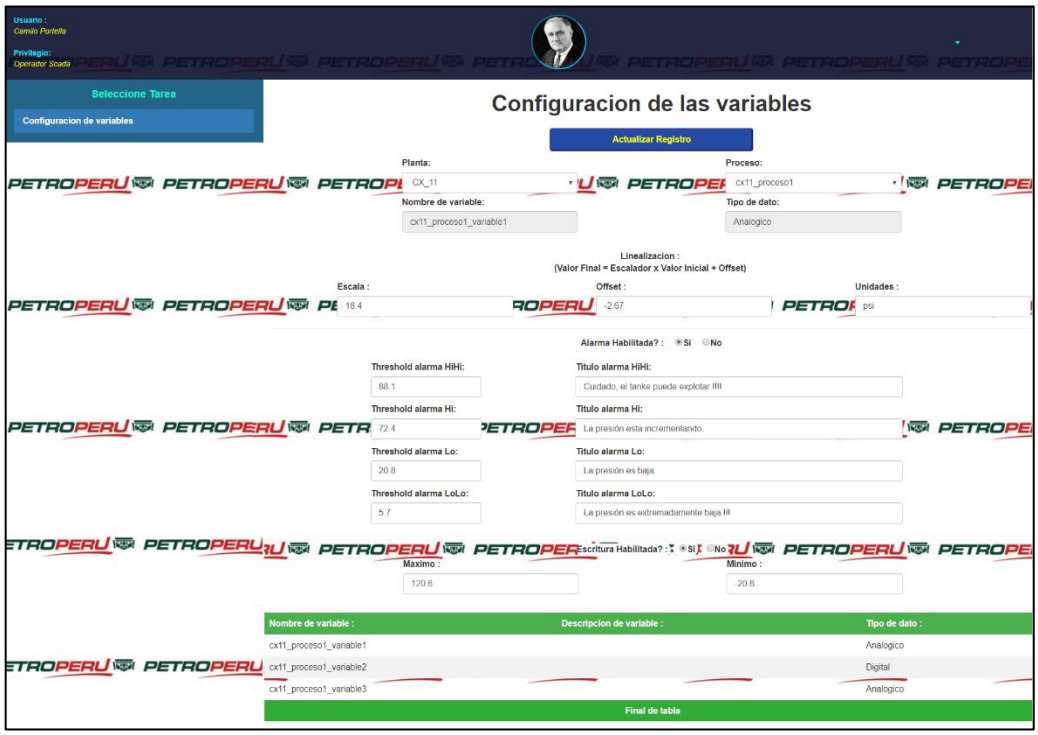


Fig. 3.106. Atributos correspondientes un parámetro físico (variable analógica). [***Fuente**: Propia*]

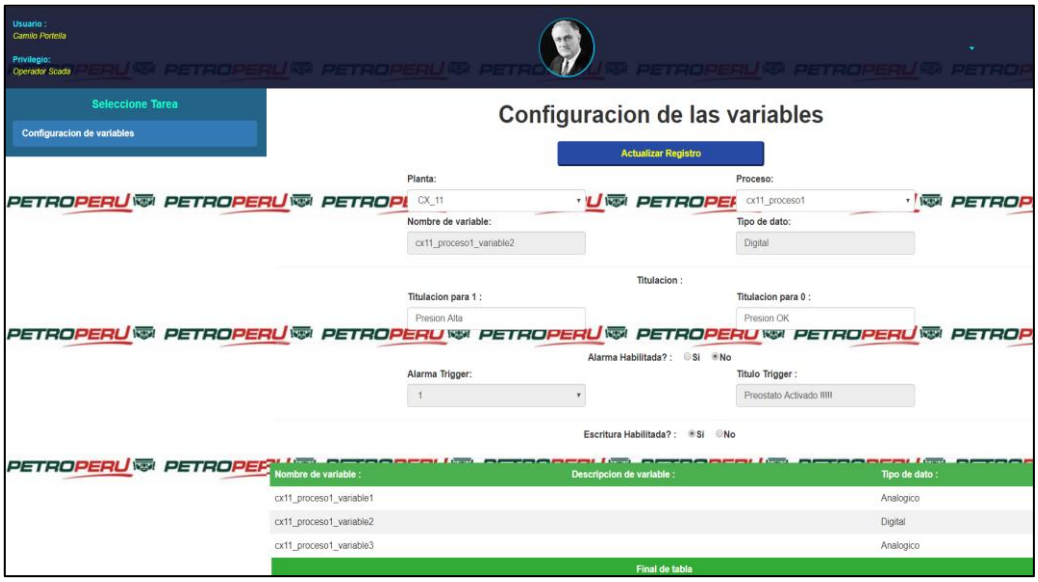


Fig. 3.107. Atributos correspondientes a un estado físico (variable digital). [***Fuente**: Propia*]

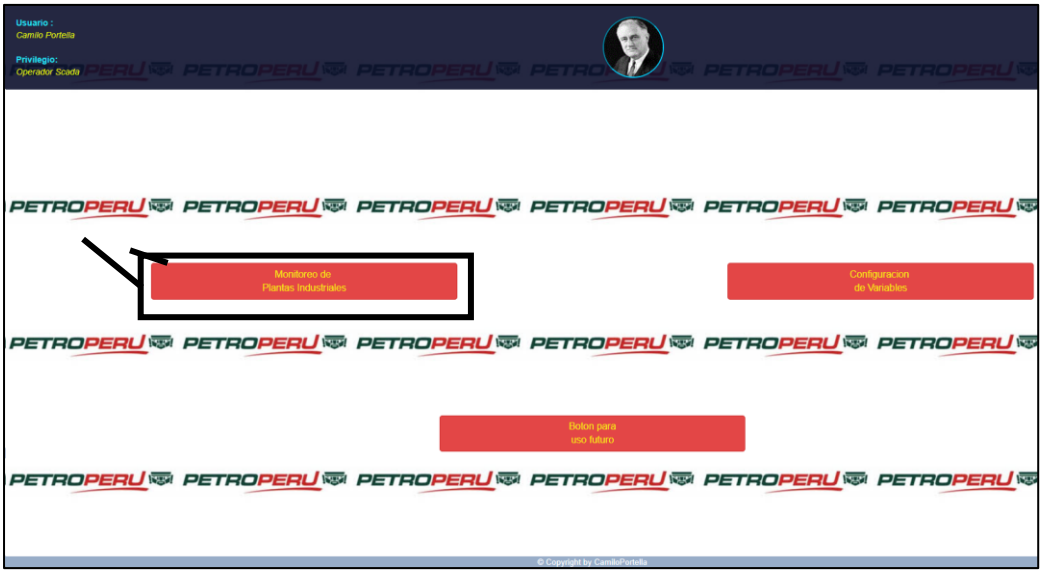


Fig. 3.108. El botón dentro del recuadro negro permite visualizar todas las plantas disponibles en la empresa. [***Fuente**: Propia*]

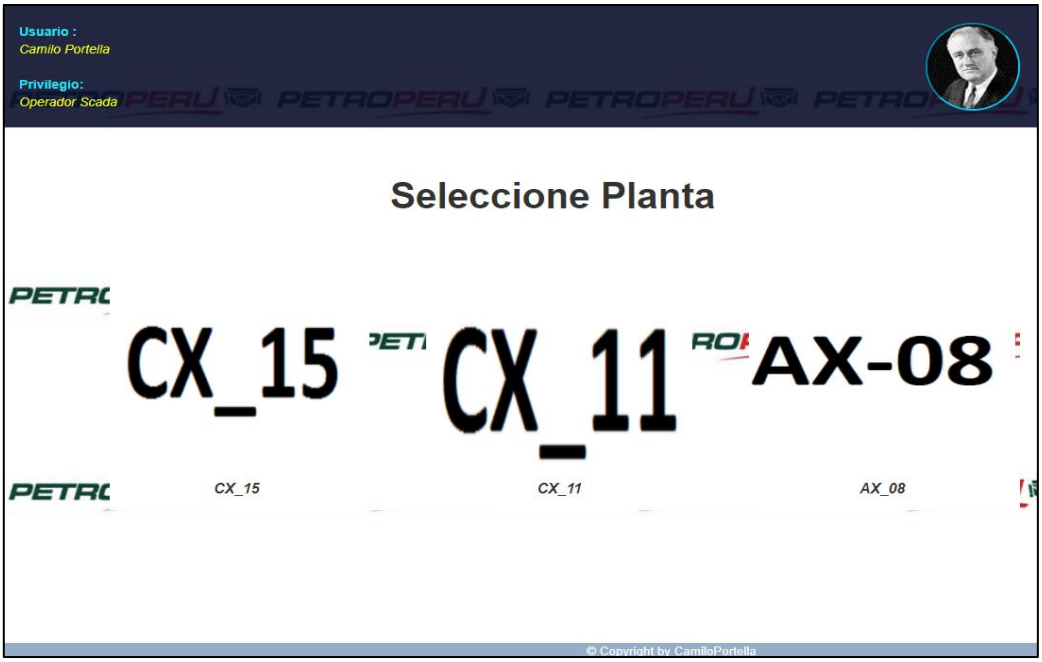


Fig. 3.109. En esta pantalla, el operador puede seleccionar la planta que requiere monitorear. [***Fuente**: Propia*]

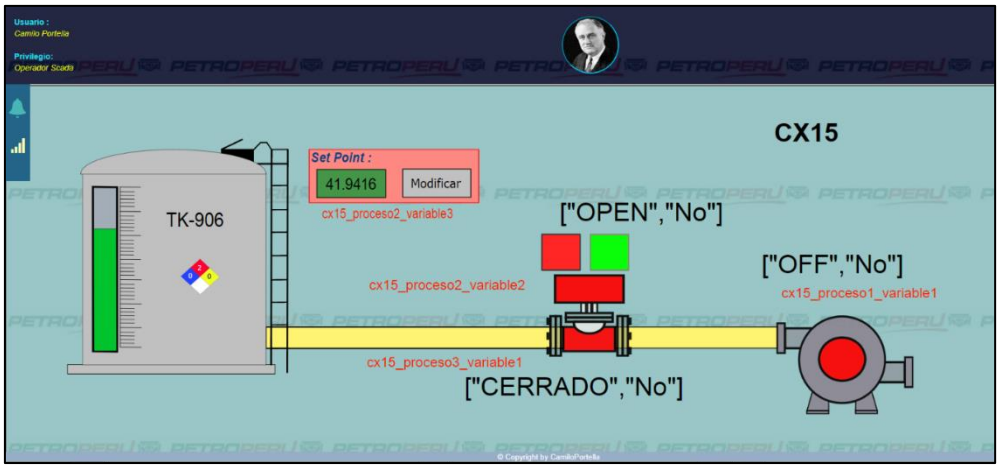


Fig. 3.110. Pantalla de monitoreo de la planta CX_15. [***Fuente**: Propia*]

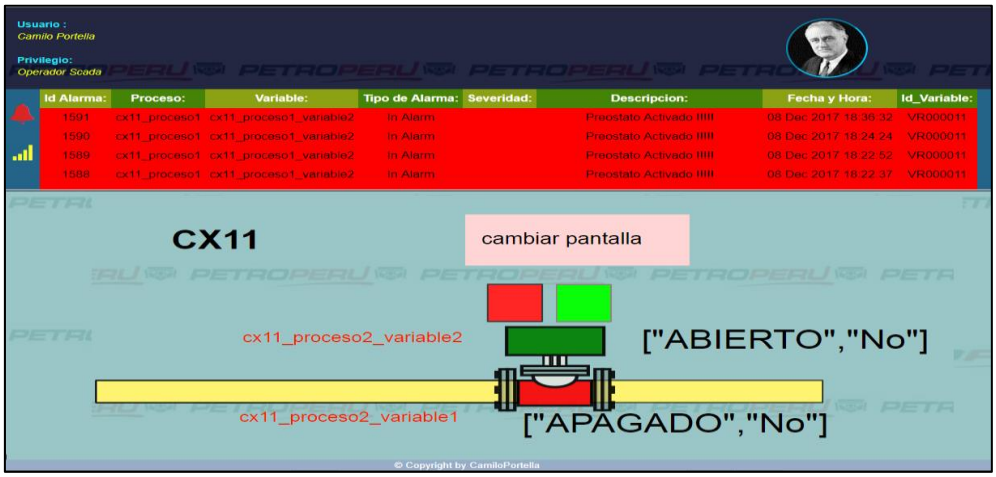


Fig. 3.111. Primera pantalla de monitoreo de la planta CX_11. [***Fuente**: Propia*]

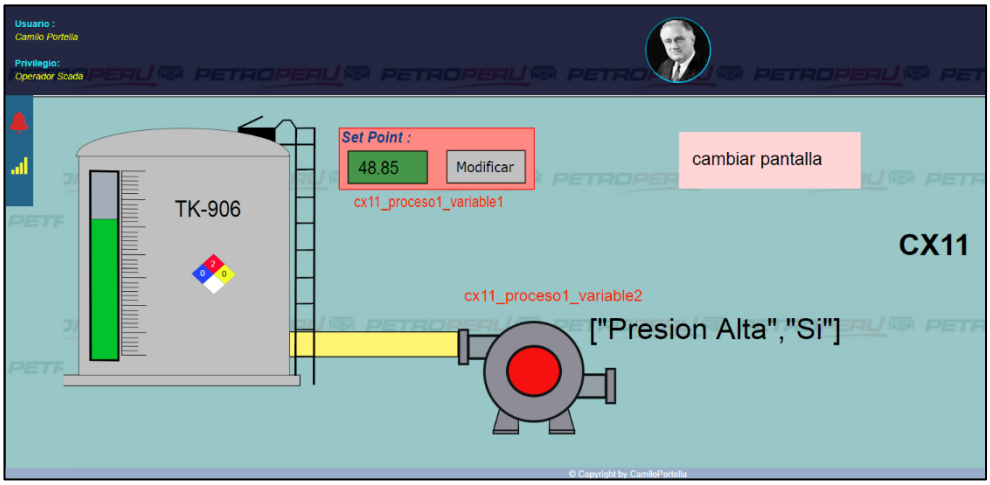


Fig. 3.112. Segunda pantalla de monitoreo de la planta CX_11. [***Fuente**: Propia*]

3.4.11 Componentes que integran el Espacio Proceso

En la Fig. 3.113., se puede ver todas las herramientas tanto a nivel de software como de hardware que se utilizaron para la construccion del ESPACIO PROCESO.

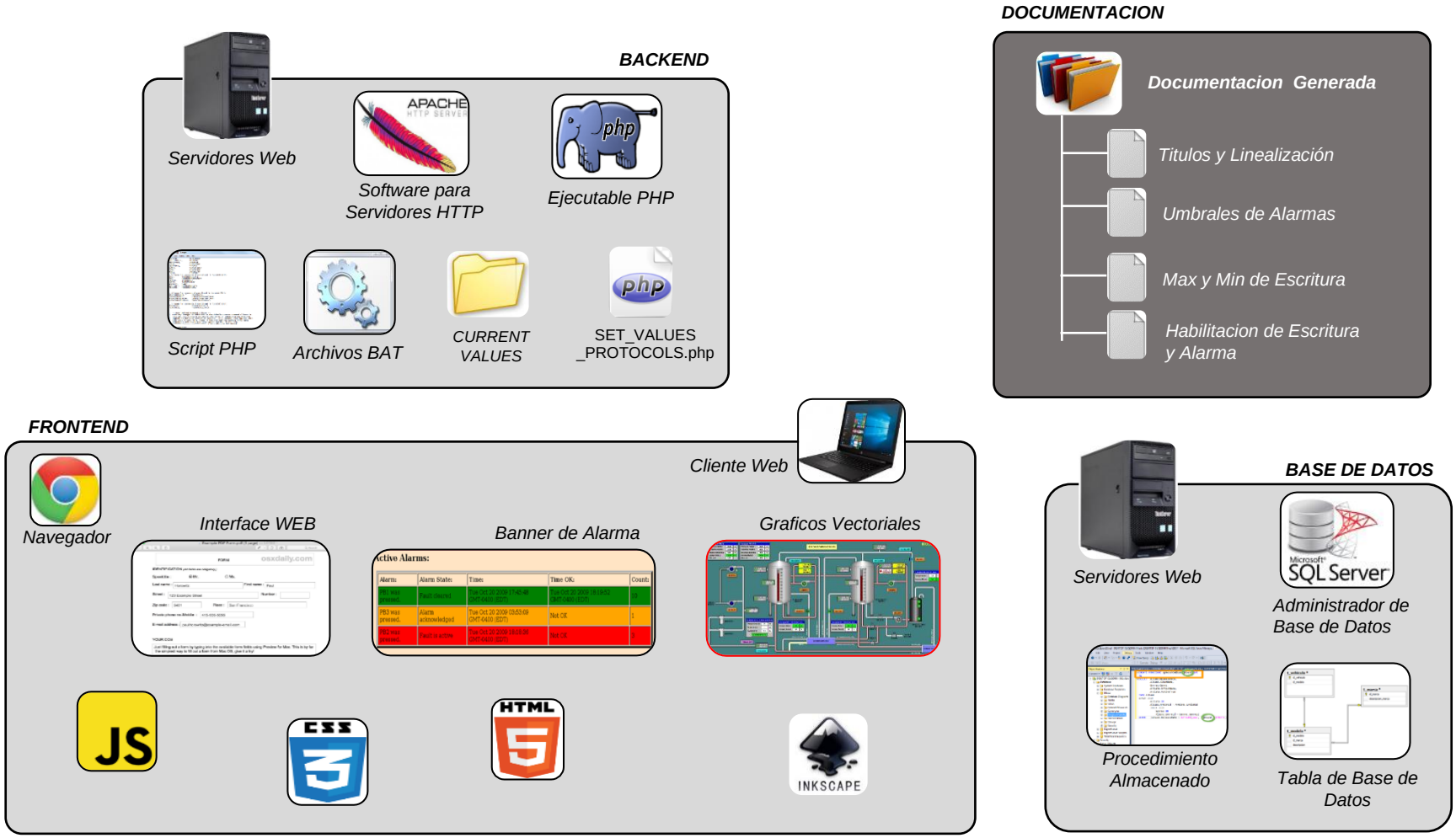


Fig. 3.113. Herramientas usadas para la construcción del Espacio Proceso [***Fuente**: Propia*]

3.5 Histórico de datos (Espacio Históricos)

3.5.1 Objetivo del Espacio Históricos.

El Espacio HISTÓRICOS se definió en la Tabla. 3.1 y fue creado con la finalidad de almacenar los valores que una determinada variable, o una INSTANCIA de la CLASE VARIABLE definida en el ESPACIO SISTEMA, ha tomado a lo largo del tiempo.

Los valores almacenados de una determinada variable o sus históricos, adquieren importancia relevante cuando se deben procesar y analizar a través de reportes, cuadros de mando, u otras herramientas estadísticas para la toma de decisiones en una empresa. La Fig. 3.114., ilustra gráficamente lo explicado líneas arriba.

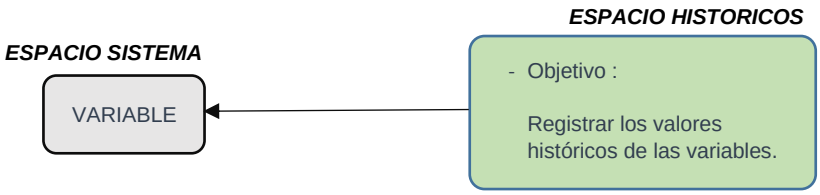


Fig. 3.114. Objetivo del Espacio Históricos [***Fuente**: Propia*]

3.5.2 Almacenamiento de los Registros

Los históricos se almacenan en bases de datos estructuradas SQL para poder ser integrados en diversas aplicaciones.

3.5.3 Composición de un Registro del Espacio Históricos

El histórico almacena el valor linealizado de parámetros físicos o el titulado cuando se trata de estados físicos, incluída la fecha y hora a la cual fue capturado el valor de la variable. En el caso de tratarse de parámetros físicos, también se incluirá la unidad de medida del parámetro físico, por ejemplo kg, psi, etc.

3.5.4 Funcionamiento del Espacio Históricos

La Fig. 3.115., muestra el mecanismo de funcionamiento del ESPACIO HISTORICOS.

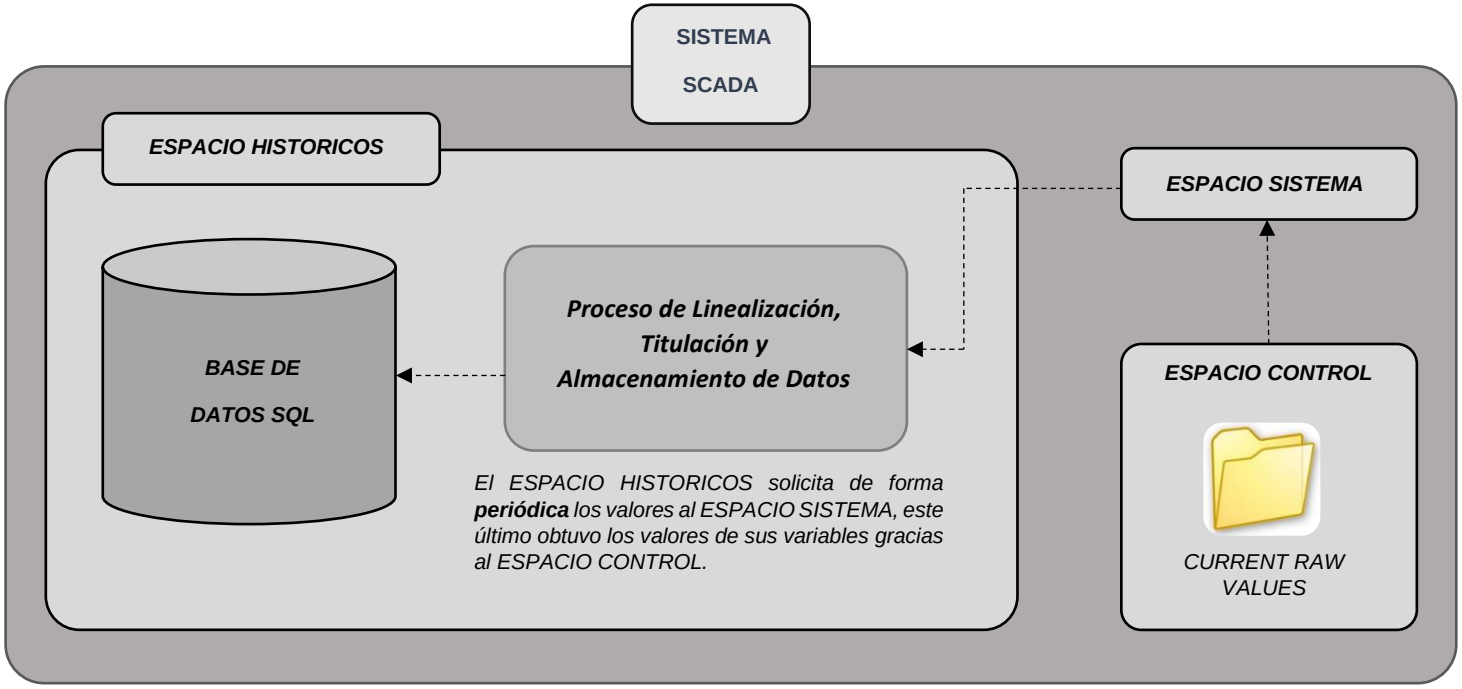


Fig. 3.115. Ruta de los datos extraídos desde el Espacio Control hasta llegar del Espacio Históricos [***Fuente**: Propia*]

3.5.5 Espacio Históricos en el proyecto de Tesis.

En Fig. 3.116., Fig. 3.117. y Fig. 3.118., se muestran las pantallas que integran el ESPACIO HISTORICOS y la función que cumple cada una de estas pantallas.

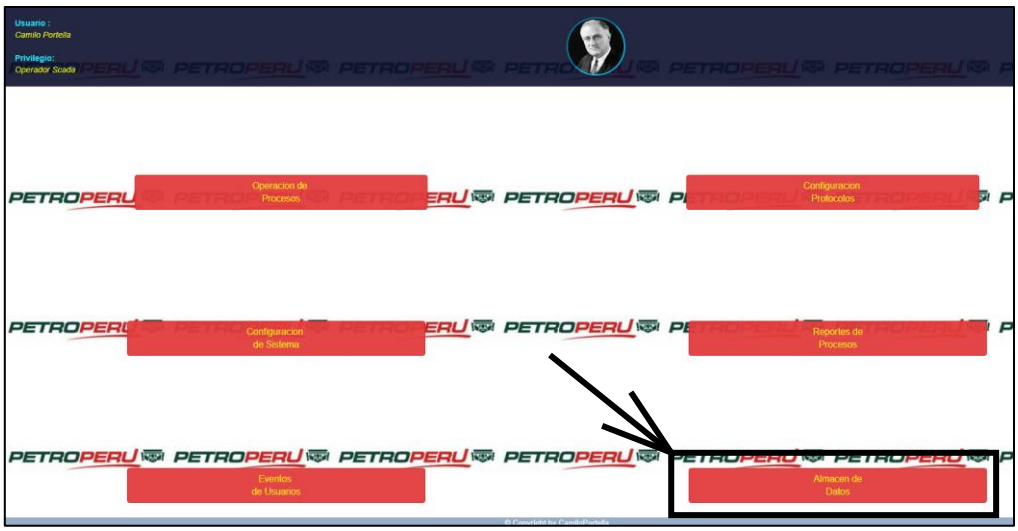


Fig. 3.116. Ilustración del botón que permite acceder al Espacio Históricos [***Fuente**: Propia*]

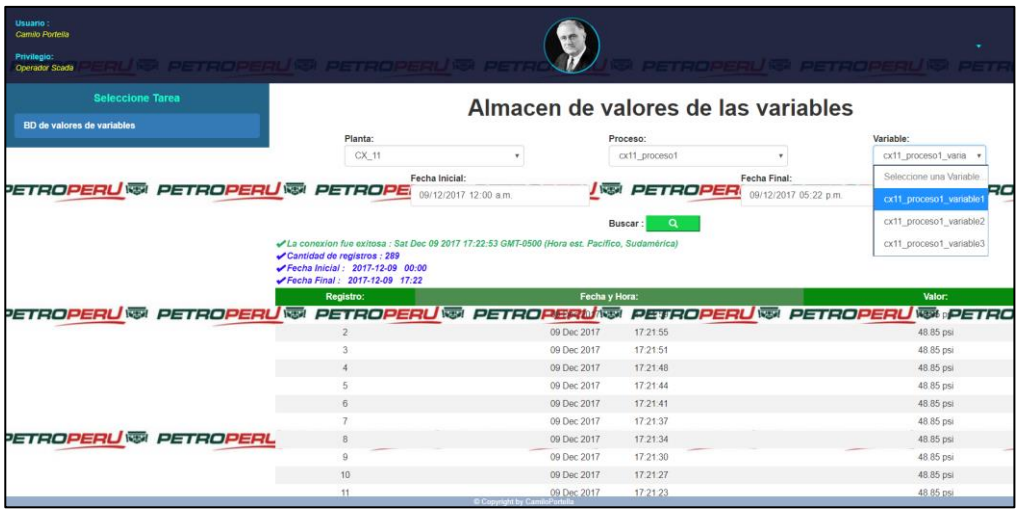


Fig. 3.117. Pantalla que permite visualizar los valores de determinada variable [***Fuente**: Propia*]

	Id_Variable	Fecha_Hora	Habilitado	Valor
1	VR000011	2017-12-09 17:31:10.000	Si	Presion OK
2	VR000014	2017-12-09 17:31:07.000	Si	ABIERTO
3	VR000013	2017-12-09 17:31:07.000	Si	APAGADO
4	VR000012	2017-12-09 17:31:07.000	Si	4.16 gal
5	VR000010	2017-12-09 17:31:07.000	Si	48.85 psi
6	VR000011	2017-12-09 17:31:06.000	Si	Presion OK
7	VR000014	2017-12-09 17:31:04.000	Si	ABIERTO
8	VR000013	2017-12-09 17:31:04.000	Si	APAGADO
9	VR000012	2017-12-09 17:31:04.000	Si	4.16 gal
10	VR000011	2017-12-09 17:31:03.000	Si	Presion OK
11	VR000010	2017-12-09 17:31:03.000	Si	48.85 psi
12	VR000014	2017-12-09 17:31:00.000	Si	ABIERTO
13	VR000013	2017-12-09 17:31:00.000	Si	APAGADO
14	VR000012	2017-12-09 17:31:00.000	Si	4.16 gal
15	VR000010	2017-12-09 17:31:00.000	Si	48.85 psi
16	VR000011	2017-12-09 17:30:59.000	Si	Presion OK
17	VR000014	2017-12-09 17:30:56.000	Si	ABIERTO
18	VR000013	2017-12-09 17:30:56.000	Si	APAGADO
19	VR000012	2017-12-09 17:30:56.000	Si	4.16 gal
20	VR000010	2017-12-09 17:30:56.000	Si	48.85 psi
21	VR000011	2017-12-09 17:30:55.000	Si	Presion OK
22	VR000014	2017-12-09 17:30:53.000	Si	ABIERTO

Fig. 3.118. Valores de las variables que son almacenados en la Base de Datos SQL [***Fuente**: Propia*]

3.6 Generación de reportes (Espacio Reportes)

3.6.1 Objetivo del Espacio Reportes

El ESPACIO REPORTES se crea con la finalidad de elaborar reportes y análisis estadísticos que permitan tomar decisiones que mejoren la productividad de una empresa de forma rápida y eficiente.

Para la construcción de este ESPACIO se usarán herramientas de Business Intelligence. En la Tesis se usará PENTAHO, que es un software gratuito.

El ESPACIO REPORTES no aplica sobre una CLASE determinada, porque estas herramientas de Business Intelligence son tan poderosas que pueden aplicar sobre cualquier recurso que se encuentre en un ordenador como, por ejemplo:

- Archivos de texto plano, archivos PDF, archivos XML.
- Datos almacenados en una Base de Datos, tanto SQL como NOSQL.
- Archivos de Sonido y Video.
- Correos Electrónicos.
- Y muchos recursos más.

En la Tesis el objetivo de este ESPACIO REPORTES será solamente el de generar un reporte sencillo, para que el lector pueda constatar lo que se podría llegar a obtener a partir de los datos almacenados en el ESPACIO HISTORICOS.

3.6.2 PENTAHO

Esta herramienta gratuita, puede correr sobre Linux y será la que se usara para generar el reporte, aunque es importante mencionar que esta herramienta de Business Intelligence tiene incorporado muchas sub herramientas **[17]**, tales como las siguientes:

- Pentaho Server BI.
- Pentaho Data Integration.
- Pentaho Reporting Services.
- Pentaho Metadata Editor.
- Pentaho Schema Workbench.

3.6.3 Creación de un Reporte

A continuación, se detalla la creación de un reporte que muestre la configuración actual del direccionamiento de cada una de las RTUs y sus respectivas variables, tal como se muestra en Fig. 3.72., Fig. 3.73. y Fig. 3.74.

El reporte mostrado en Fig.3.123., se crea con el software Pentaho Reporting Services, que es una de las herramientas que PENTAHO ofrece en el mercado. Una vez generado el reporte se publicará este mismo en la web, con la finalidad de hacerlo accesible para cualquier usuario desde la Internet. En Fig. 3.119., Fig. 3.120., Fig. 3.121., Fig. 3.122., y Fig. 3.123., se mostrarán algunas configuraciones que se realizaron para la creación del reporte.

Fig. 3.119. Software Pentaho reportes [***Fuente**: Propia*]

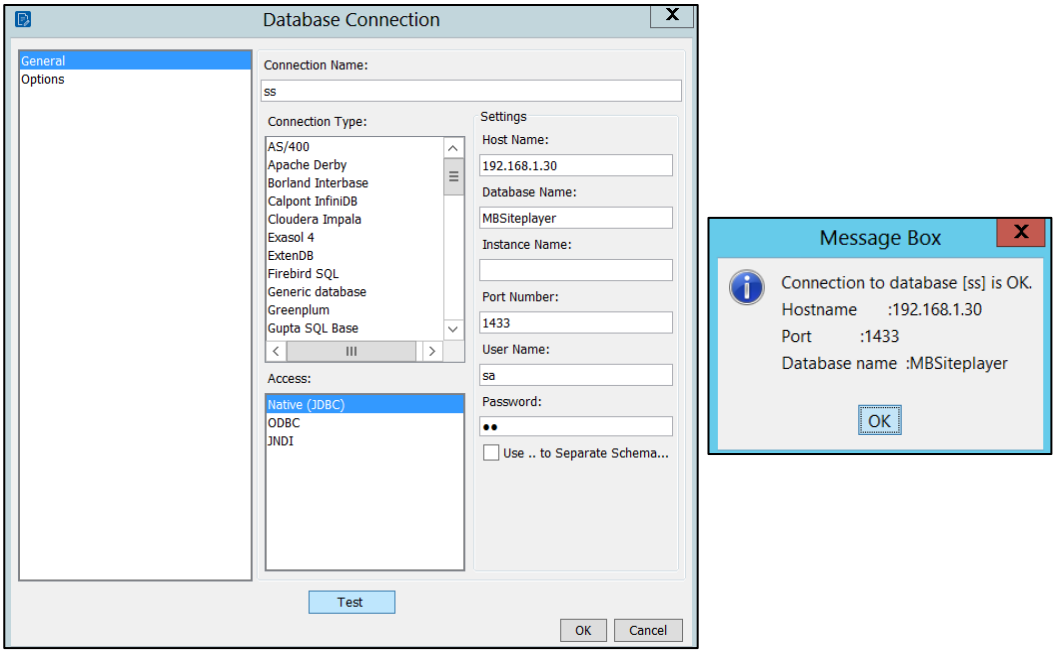


Fig. 3.120. Conexión de Pentaho Report Designer con la Base de Datos [***Fuente**: Propia*]

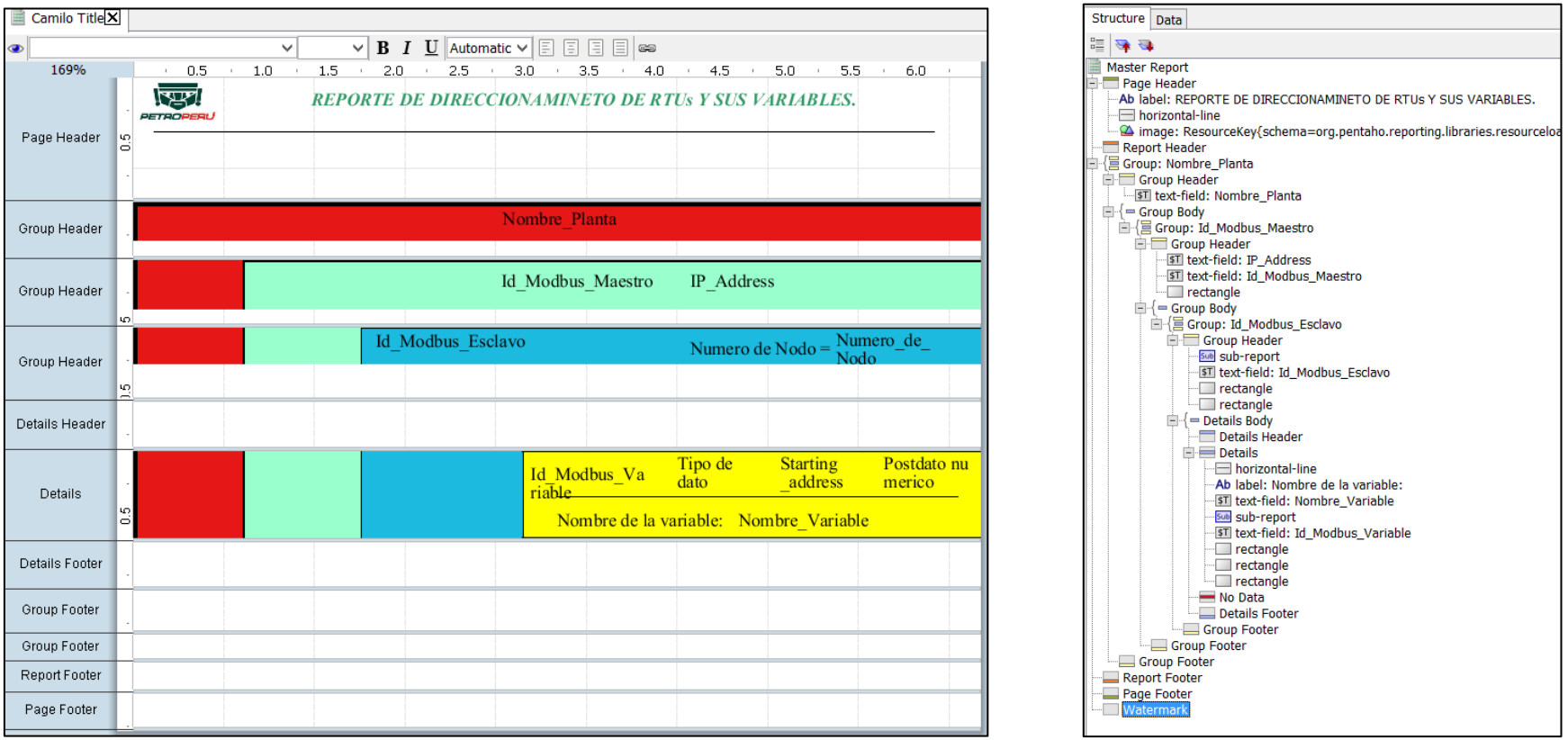


Fig. 3.121. Estructura del reporte relacionado a los direccionamientos de las RTUs y sus variables. [***Fuente**: Propia*]

Query Name

Query 1

Static Query | Query Scripting

Query

```
SELECT
	TOP (100) PERCENT dbo.Lista_Plantas.Nombre_Planta,
	"dbo"."Lista_Modbus_Maestro"."Id_Modbus_Maestro",
	"dbo"."Lista_Modbus_Maestro"."IP_Address",
	"dbo"."Lista_Modbus_Esclavo"."Id_Modbus_Esclavo",
	"dbo"."Lista_Modbus_Variable"."Id_Variable",
	"dbo"."Lista_Modbus_Variable"."Id_Modbus_Variable",
	"dbo"."Lista_Variables"."Nombre_Variable"
FROM
	"dbo"."Lista_Plantas" INNER JOIN "dbo"."Lista_Modbus_Maestro" ON "dbo"."Lista_Plantas"."Id_PL" = "dbo"."Lista_Modbus_Maestro"."Id_Pl"
	INNER JOIN "dbo"."Lista_Modbus_Esclavo" ON "dbo"."Lista_Modbus_Maestro"."Id_Modbus_Maestro" = "dbo"."Lista_Modbus_Esclavo"."Id_Modbus_Maestro"
	INNER JOIN "dbo"."Lista_Modbus_Variable" ON "dbo"."Lista_Modbus_Esclavo"."Id_Modbus_Esclavo" = "dbo"."Lista_Modbus_Variable"."Id_Modbus_Esclavo"
	INNER JOIN "dbo"."Lista_Variables" ON "dbo"."Lista_Modbus_Variable"."Id_Variable" = "dbo"."Lista_Variables"."Id_Variable"
ORDER BY
	dbo.Lista_Plantas.Nombre_Planta ASC,
	dbo.Lista_Modbus_Maestro.Id_Modbus_Maestro ASC,
	dbo.Lista_Modbus_Esclavo.Id_Modbus_Esclavo ASC
```

Query Name

Query 2

Static Query | Query Scripting

Query

```
SELECT Numero_de_Nodo
FROM   dbo.Lista_Numeros_de_Nodo_de_Esclavos_MB
WHERE  Id_Modbus_Esclavo = ${Id_Modbus_Esclavo} AND Fecha_Hora = (SELECT MAX(Fecha_Hora)
FROM     dbo.Lista_Numeros_de_Nodo_de_Esclavos_MB
WHERE  Id_Modbus_Esclavo = ${Id_Modbus_Esclavo}
GROUP BY Id_Modbus_Esclavo)
```

Query Name

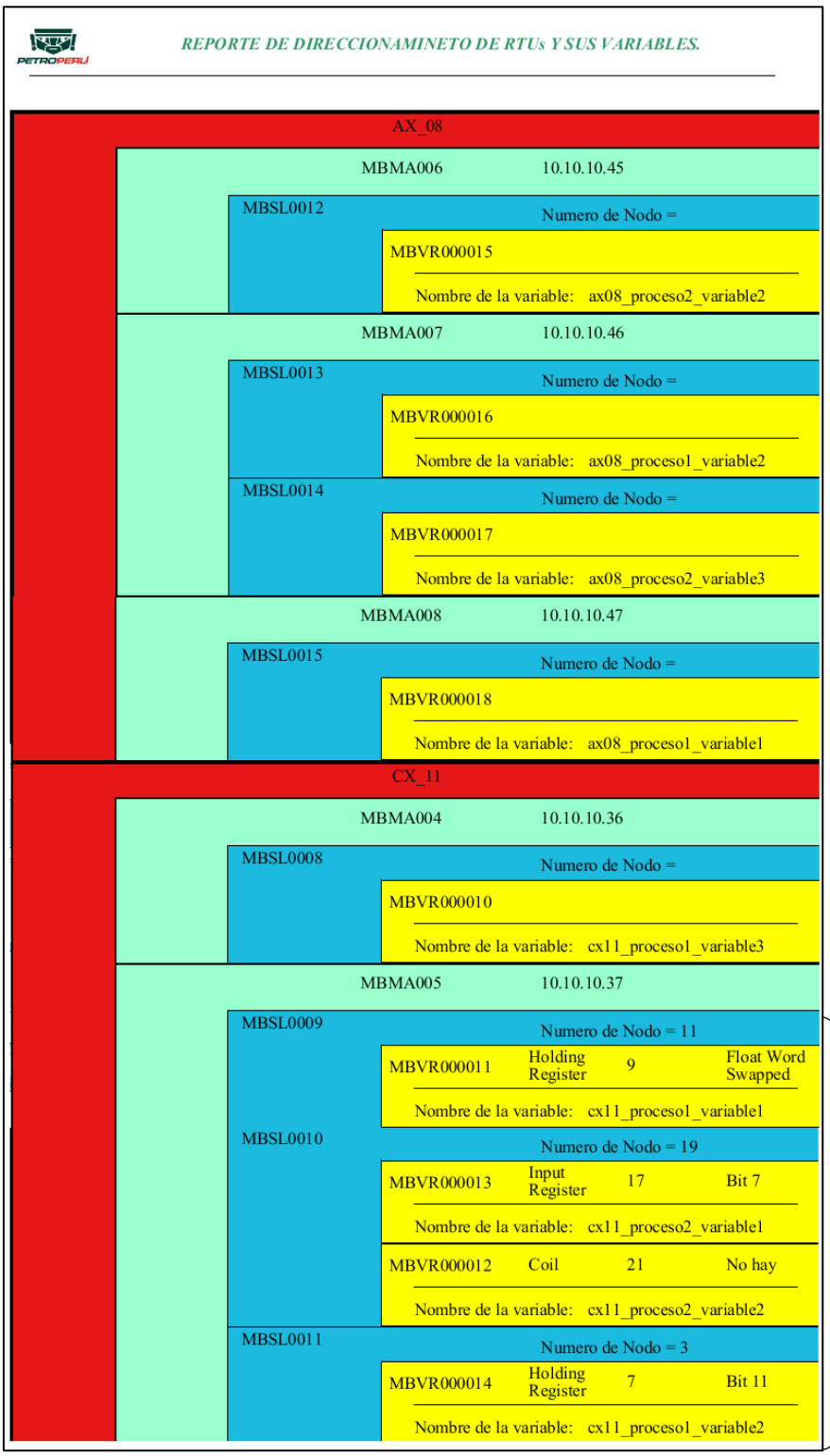
Query 3

Static Query | Query Scripting

Query

```
SELECT Tipo_de_dato, Starting_address, Posbit_datonumerico, Fecha_Hora, Id_Modbus_Variable
FROM   dbo.Lista_Parametros_de_Variables_MB
WHERE  Id_Modbus_Variable = ${Id_Modbus_Variable} AND Fecha_Hora = (SELECT MAX(Fecha_Hora)
FROM   dbo.Lista_Parametros_de_Variables_MB
WHERE  Id_Modbus_Variable = ${Id_Modbus_Variable}
GROUP BY Id_Modbus_Variable)
```

Fig. 3.122. Consultas usadas para la creación del reporte [***Fuente**: Propia*]



Como se puede observar los direccionamientos de las RTUs del Gateway de código MBMA005 y los direccionamientos de las variables de estas RTUs están acorde a lo mostrado en la Fig. 3.72, Fig. 3.73 y Fig. 3.74.

Fig. 3.123. Vista previa del reporte, que contiene los direccionamientos de las RTUs y de sus respectivas variables [***Fuente**: Propia*]

A continuación, en la Fig. 3.124., se muestra el mismo reporte que se generó en la Fig. 3.123., pero hosteado en el servidor web de Pentaho de tal forma que cualquier usuario pueda visualizarlo desde la Internet.

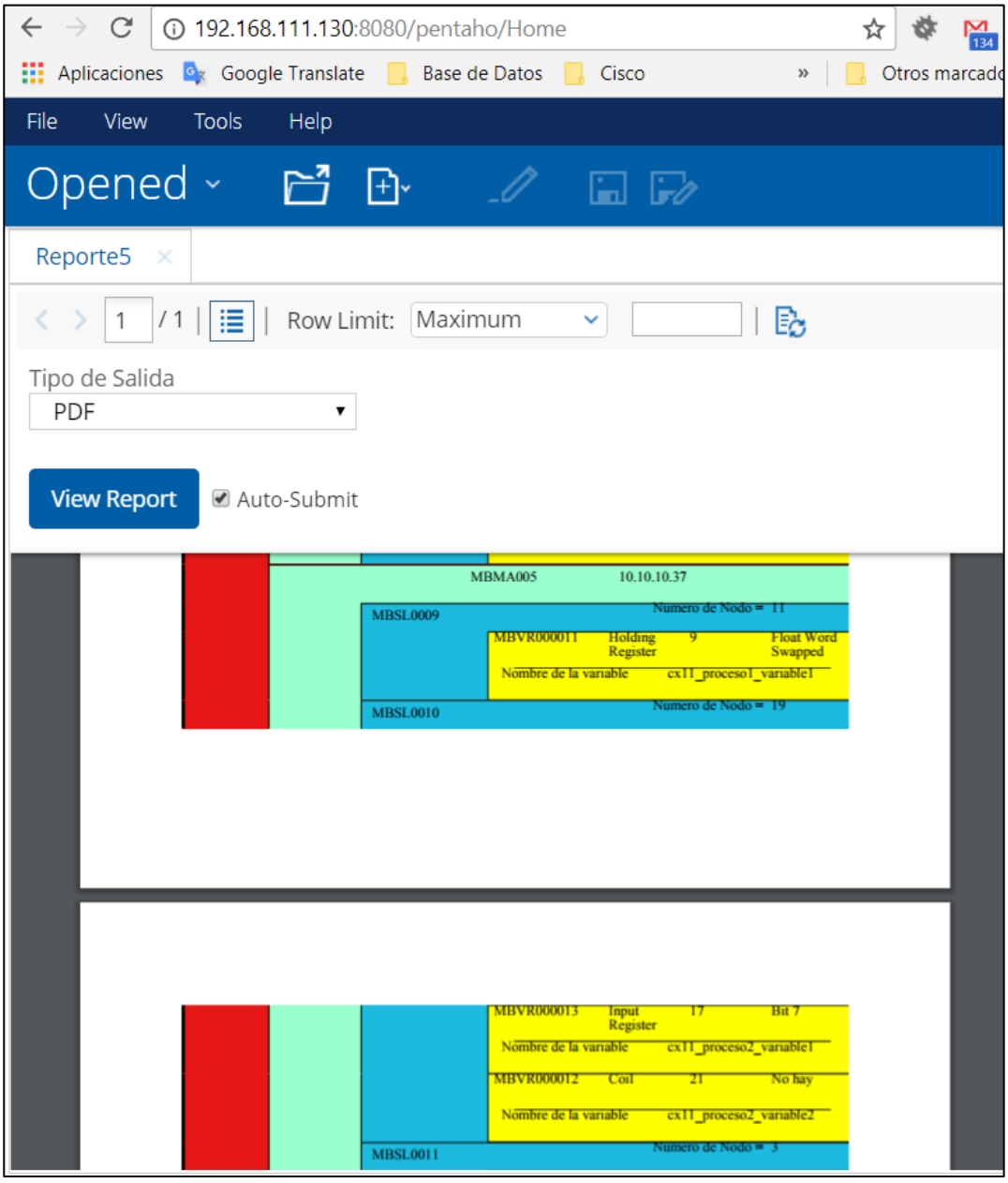


Fig. 3.124. Reporte accedido desde un navegador web [***Fuente**: Propia*]

3.6.4 Componentes que integran el Espacio Reporte

La Fig. 3.125. muestra las herramientas que componen el ESPACIO REPORTES.

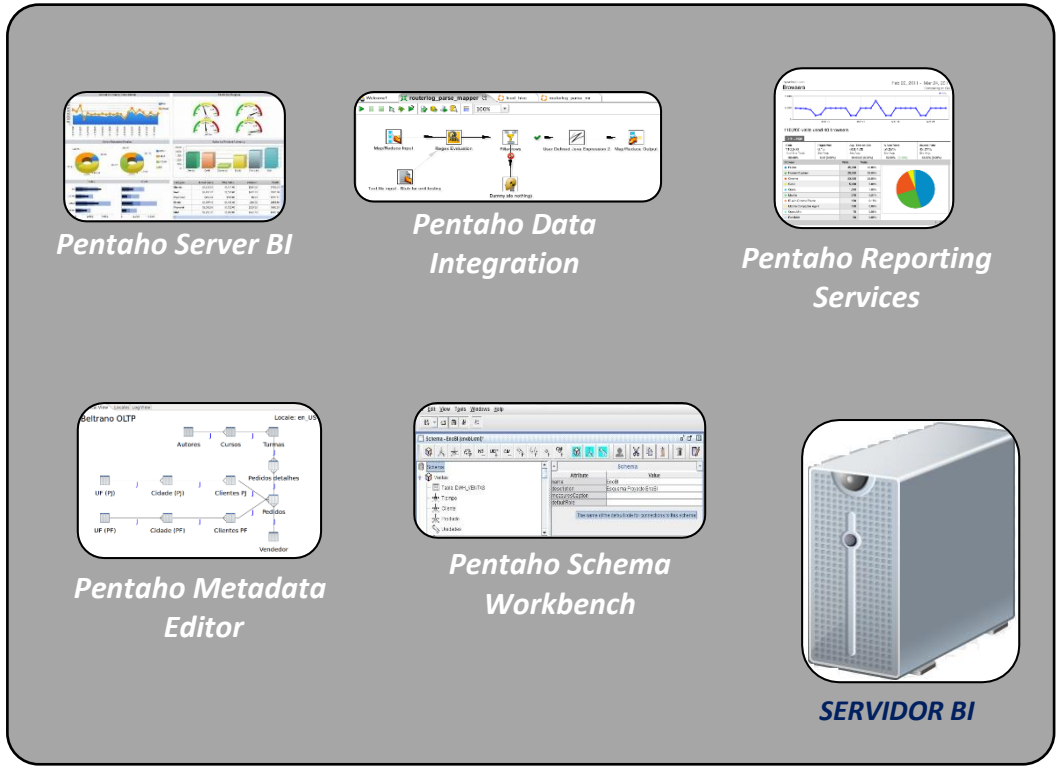


Fig. 3.125. Herramientas usadas para la construcción del Espacio Reportes [***Fuente**: Propia*]

3.7 Estimación de Costos

3.7.1 Introducción

En este capítulo se abordará el tema del gasto estimado que una empresa tendría que realizar para implementar el sistema SCADA expuesto en la Tesis. Solo se contemplan los costos de hardware y software, mas no los costos de mano de obra calificada ni de dirección técnica.

Aunque es importante mencionar que el hardware utilizado para lo que es el Gateway no es lo más robusto que una aplicación industrial solicitaría.

3.7.2 Costos de Hardware

Los costos de hardware contemplados en la tesis son los referidos al Gateway y el Servidor SCADA. No se está considerando los costos de las RTUs ni de los dispositivos de campo, tales como dispositivos captadores y actuadores. En la Tabla 3.4., se presenta el resumen de costos del hardware contemplado en la tesis.

Tabla 3.4. Costos de los materiales necesarios para la instalación del sistema SCADA en una empresa. [***Fuente**: Propia*]

Dispositivo	Costo del dispositivo (en soles)
Gateway	
Microcontrolador PIC 16F877A	20.00
Microservidor Web SitePlayer	200.00
Cableado RS485 hacia las RTUs.	100.00
Fuente de 15Voltios para Gateway.	300.00
Switch Ethernet	200.00
Tablero para alojar el Gateway	600.00
Servidor SCADA	
1PC + Disco duro 1TB + 16 GB de RAM + 2 Tarjetas PCI 10/100/1000 Ethernet.	6000.00
UPS	2000.00
Switch Ethernet	500.00
Costo Total del Hardware (En soles)	**9920.00**

3.7.3 Costos de Software

Los costos de software contemplados en la tesis están relacionados al software Modbus Slave para simular el funcionamiento de las RTU. Adicionalmente, se consideran los costos referidos al Gateway y el Servidor SCADA. En la Tabla 3.5, se presenta el resumen de costos del software contemplado en la tesis.

Tabla 3.5. Costo del software que se tendría que adquirir para instalar el sistema SCADA en una empresa. [***Fuente**: Propia*]

Software	Costo del software(en soles)
RTUs y Dispositivos Finales	
Software para simular RTUs, Modbus Slave.	400.00
Gateway	
Software de complicación PICC. Sirve para compilar el programa de PIC.	500.00
Software de compilación SiteLinker. Sirve para compilar el programa del SitePlayer.	Gratuito.
Servidor SCADA	
S.O Windows	3100.00
MS SQL Server Express	Gratuito.
PHP	Gratuito.
Inkscape	Gratuito.
Pentaho	Gratuito.
Costo Total del Software (En soles)	**4000.00**

3.7.4 Costo Total

El costo total que demandaría la implementación del sistema SCADA propuesto en la tesis, sería el que se muestra en la Tabla 3.6.

Tabla 3.6. Costo Total del Sistema SCADA propuesto en la tesis. [***Fuente**: Propia*]

Componente	Costo (en soles)
Hardware	9920.00
Software	4000.00
COSTO TOTAL (En soles)	**13920.00**

Nota: Es importante mencionar que este costo no incluye el costo del equipo diseñador, ya que este estará en función de algunas variables ajenas a lo estrictamente relacionado a este proyecto de Tesis, como lo puede ser la arquitectura de red la cual ya dispone el cliente, las distancias geográficas de las plantas, etc.

CAPÍTULO IV
ANÁLISIS Y DISCUSIÓN DE RESULTADOS

4.1 Objetivos Alcanzados

A lo largo de todo el trabajo de Tesis algo que se puede considerar como un objetivo logrado bastante importante (y el principal), es que casi todas las tareas que realiza el sistema SCADA, se pueden llevar a cabo desde una interface WEB, como se enlista a continuación:

a) La instanciación de las clases Planta, Proceso y Variables, tal como lo muestra la Fig. 3.16, 3.17 y 3.18.

b) La creación de la red que compone la red Modbus-RTU, la cual está integrada por Gateway, RTUs y Registros de cada RTUs tal como se muestra en la Fig. 3.40, Fig. 3.41 y Fig. 3.42.

c) Creación de forma automatizada de los archivos que se descargan sobre los dispositivos que integran el Gateway, en este caso nos referimos al Microcontrolador PIC y al micro-servidor SitePlayer tal como se puede apreciar en la Fig. 3.51, Fig. 3.52 y Fig. 3.53.

d) Configuración del direccionamiento de los RTUs, así como también el direccionamiento de los registros de estas RTUs tal como se indican la Fig. 3.71, Fig. 3.72, Fig. 3.73 y Fig. 3.74.

e) Posibilidad de verificar el correcto funcionamiento del Gateway, accediendo directamente a este desde una PC, pudiendo realizar tareas de lectura y escritura sobre el Gateway tal como lo muestra la Fig. 3.79 y Fig. 3.80.

f) El análisis de los procesos mediante gráficos que reflejen de forma genérica la infraestructura de estos mismos tal como se puede observar en la Fig. 3.110, Fig. 111 y Fig. 3.112.

g) El análisis del estado de alarma de las variables a través de un banner, el cual emite una alarma sonora, así como también indica el lugar de donde procede la alarma y además permite el reconocimiento de esta por parte del usuario de campo tal como se muestra en la Fig. 3.99, Fig. 3.100 y Fig. 3.101.

h) Configuración de los distintos parámetros de una variable como lo son: sus umbrales de alarmas (HiHi, Hi, Lo y LoLo), los mensajes que se muestran cuando estos umbrales son sobrepasados, habilitación/deshabilitacion de las alarmas, límite

máximo y mínimo de escritura y habilitación/deshabilitación de escritura tal como se muestra en la Fig. 3.106 y Fig. 107.

i) Visualización de los históricos de las variables de acuerdo al periodo de tiempo seleccionado por el usuario de campo tal como se puede constatar en la Fig.117.

j) Generación de reportes acorde a lo seleccionado por el usuario en los formularios tal como se muestra en la Fig. 3.124.

Es importante mencionar que no solo lo referente a las interfaces fue implementado con tecnología web, sino también los archivos que periódicamente se comunican con los Gateways tal como se puede visualizar en la Fig.3.81.

Los resultados mostrados líneas arriba nos muestra que el objetivo de construir un sistema SCADA usando tecnología web, fue conseguido de forma satisfactoria, obviamente hay varios puntos que se deben afinar, estos serán enlistados en la sección RECOMENDACIONES.

Otro objetivo que también se alcanzó, es el hecho de que el sistema SCADA implementado estuvo en capacidad de asimilar información como imágenes, tal como se muestra en las Fig. 3.16, Fig. 3.17, Fig. 3.18, y no solo ello, sino que esta información se encuentra estructurada en la base de datos, lo cual permitió la reutilización de esta información tal como ocurrió en la Fig. 3.109, todo ello debido a la arquitectura basada en Programación Orientada a Objetos.

El hecho que el microcontrolador PIC pueda tener implementado el protocolo Modbus-RTU sobre la norma RS-485, es algo que por sí solo no podría ser considerado como un mérito de esta Tesis, ya que al ser Modbus-RTU un protocolo abierto, se puede encontrar mucha información de como implementar este protocolo en un procesador, pero algo que sí fue un resultado propio de esta tesis, es la generación automatizada de este código en función a la red existente (RTUs y sus registros), evitando que el usuario final se vea en la necesidad de crear código fuente.

Otro resultado bastante importante obtenido fue la creación de un protocolo de comunicación dentro del PIC, el cual se comunica de forma serial con el micro-servidor WEB SitePlayer. Este protocolo básicamente permite que el SitePlayer pueda identificar qué tipo de información está recibiendo, tal como se muestra en la Fig. 3.56, Fig. 3.57 y Fig. 3.58.

Un resultado que también fue bastante positivo fue la velocidad con la cual se llevaba a cabo el seteo de las variables a través del Gateway, estas tenían en promedio un tiempo de menos de un segundo cuando se realizaba desde una PC local, y de poco más de un segundo cuando el seteo se produce desde la Internet, lo cual demuestra que las herramientas de software como PHP y SQL Server son bastante potentes.

4.2 Objetivos que no fueron alcanzados como se esperaba

El objetivo más importante que no se alcanzó, y que hace poco viable la implementación del sistema en una planta industrial es con respecto a la robustez que debería tener el Gateway, el cual está conformado por un microcontrolador PIC y un micro-servidor web SitePlayer, ya que fue muy común al menos en el caso del PIC que el ruido afectara su funcionamiento, provocando que el programa se congele, lo cual provocaba la necesidad de reiniciar el PIC.

Aunque es importante recalcar que el problema que se mencionó líneas arriba, no esta referido al software que se implementó, sino al hardware, ya que algo que sí se puede asegurar es que durante todo el tiempo que se estuvo testeando el Gateway, este nunca distorsionó la data que extraía del RTU, eso quedó registrado en la base de datos tal como se muestra en la Fig. 3.118.

Pero algo muy positivo que también se debe mencionar, es que debido a la arquitectura con la cual se diseñó el sistema SCADA, el remplazo del Gateway actual por uno de mejores prestaciones, no supondría alteraciones drásticas en la programación del sistema SCADA, todo lo contrario, el remplazo del código estaría focalizado en algunos puntos, lo cual demuestra lo modularizado del sistema implementado.

Otro aspecto que tal vez no se esperaba, era lo referido a la compatibilidad de algunos navegadores con las funciones de JavaScript, sobre todo con el navegador Safary se presentaron esos problemas, lo más recomendable siempre fue usar Google Chrome o Mozilla Firefox.

CONCLUSIONES

1 Con el proyecto de Tesis se demuestra que la implementación de un sistema SCADA es posible usando la tecnología Web, el cual proporciona muchas de las bondades que ofrece un SCADA comercial.

2 Utilizando el microcontrolador PIC y el microservidor-web SitePlayer se pueden extraer datos de los dispositivos Modbus-RTU y publicarlos en la Internet de manera confiable.

3 La integración del Gateway al sistema SCADA como "concentrador de datos" permite la rápida actualización de las interfaces gráficas del SCADA, lo cual a su vez permite que el servidor SCADA ahorre recursos como puertos TCP, velocidad del procesador, timers, etc.

4 Aplicando POO, es posible implementar un SCADA que permita a este almacenar información de tipo binaria, como imágenes, algo que, si bien los sistemas SCADAs comerciales también pueden realizar, pero el SCADA implementado en la Tesis tiene una ventaja: lo almacena de forma estructurada, lo cual permite la reutilización de esta información para la implementación de futuras tareas.

5 Se demostró que es posible la implementación de un conversor Modbus-RTU a HTTP usando dispositivos de bajo coste, sin necesidad de adquirir un PLC, que es la solución a la cual recurren las empresas cada vez que necesitan integrar RTUs a la red Ethernet, lo cual resulta ser bastante costoso.

6 Las interfaces web demostraron su correcto funcionamiento tanto para el monitoreo como para el control de los procesos, aparte de ofrecer una calidad de gráficos de mejor resolución que el de los SCADAs convencionales ya que los gráficos de la Tesis usan la tecnología de Gráficos Vectoriales conocida como SVG.

7 Se demostró que el sistema SCADA al estar dividido en Servicios, hace posible la modularidad de este mismo, lo cual permite que cualquier modificación que se realiza en un Servicio no altere la programación de otros Servicios.

8 Se concluye que la transmisión "concentrada" de datos que hace el Gateway tiene también, un impacto económico en caso de usar la red celular, donde por ejemplo la transmisión "concentrada" de solo10 variables cada 5 segundos puede suponer que en un día haya un ahorro de más 20 Gb de datos (y su correspondiente costo

monetario) si lo comparamos con una trasmisión que no usa Gateway.

9 Queda demostrado que el gasto de implementación del SCADA de esta tesis es mucho menor que el de un SCADA tradicional, ya el SCADA desarrollado en este proyecto tendría un costo de poco menos de 10 000 soles, en el cual se incluiría todo el hardware y software, mientras que solamente las licencias de un SCADA como FactoryTalk cuestan cerca de 10 000 dólares.

RECOMENDACIONES

1 Se recomienda remplazar el microcontrolador PIC y el microservidor Web Site Player por microcomputadoras con mayor velocidad de CPU y capacidad almacenamiento, ello con la finalidad de poder trabajar con la mayor cantidad de variables de campo y transmitirlas con la mayor rapidez posible, una buena alternativa es la Raspberry Pi.

2 Se recomienda utilizar la tecnología Server Sent Event (SSE) con la finalidad de incrementar la velocidad de actualización de los datos que intercambian el servidor Web y cliente Web, ya que SSE elimina la necesidad que el cliente Web tenga que solicitar periódicamente la data al servidor Web.

3 Se recomienda que la nueva microcomputadora que remplace al microcontrolador PIC y al micro-servidor SitePlayer tenga embebido el protocolo HTTPS, ello con la finalidad de hacer el sistema más seguro frente a los ataques cibernéticos, ya que HTTPS encripta los datos antes de enviarlos por la red.

4 Se recomienda la encriptación del código fuente generado en PHP, ello con la finalidad de no permitir la alteración del software y garantizar su integridad frente a un eventual ataque cibernético.

5 Se recomienda que la nueva microcomputadora que remplace al microcontrolador PIC y al micro-servidor SitePlayer opere con lenguajes interpretados, como PHP o Python, ello con la finalidad de evitar compilaciones previas, lo cual a su vez permite que el archivo creado por el sistema SCADA sea descargado inmediatamente sobre el Gateway usando el protocolo FTP y ejecutándose al instante.

6 Se recomienda usar la herramienta PDO de PHP con la finalidad de mantener invariable la programación, ya que PDO permite usar el mismo código fuente para acceder a la base de datos SQL, sin importar cual sea la marca de este último.

REFERENCIAS BIBLIOGRÁFICAS

[1] Ulitin, Boris & Babkin, Eduard. (2018). An Object-Oriented Model for Smart Devices in Internet of Things. Proceedings of the XXth Conference of Open Innovations Association FRUCT. 426. 263-271. 10.23919/FRUCT.2018.8468302.

[2] V. M. Igure, S. A. Laughter, and R. D. Williams. (6 de Marzo del 2006) Security issues in SCADA networks. Computers & Security.

[3] Inductive Automation. (12 de Septiembre del 2018) What is SCADA? Descargado el 2018 de: https://inductiveautomation.com/resources/article/what-is-scada

[4] Polytron (s.f) Descargado el 2018 de: https://polytron.com/blog/is-ethernet-deterministic-does-it-matter-part-1/

[5] Northware Software Development (Febrero del 2012) Programación Orientada a Objetos. Descargado el 2018 de: https://www.northware.mx/wp-content/uploads/2013/04/Programacion-orientada-a-objetos.pdf

[6] Cisco Network Academy (s.f) Descargado el 2017 de: https://www.netacad.com/es/courses/networking/ccna-introduction-networks

[7] Ilya Grigorik (2013) High Performance Browser Networking Descargado el 2017 de: https://hpbn.co/

[8] Gordon Clarke, Deon Reynders (2004) Modern SCADA Protocols Descargado el 2017 de: https://www.julesbartow.com/Pictures/RF/Practical_modern_SCADA_protocols_-_dnp3,_60870-5_and_Related_Systems.pdf

[9] Modbus Organization (2012) Modbus Application Protocol Specification V1.1b3 Descargado el 2017 de: http://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[10] Maxim Integrated (2001) Guidelines for Proper Wiring of an RS-485 (TIA/EIA-485-A) Network Descargado el 2017 de: https://www.maximintegrated.com/en/design/technicaldocuments/tutorials/7/763.html

[11] Keith Stouffer, Joe Falco, Karen Scarfone. (Mayo 2013) Guide to Industrial Control Systems (ICS) Security.
Descargado el 2017 de :
http://electrical-engineering-portal.com/res/Guide-to-Industrial-Control-Systems-ICS-Security-800-82r1.pdf

[12] Microchip (2003) PIC16F87XA Data Sheet 28/40/44-Pin Enhanced Flash Microcontrollers
Descargado el 2017 de:
http://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf

[13] Site Player (2003) Development System Rev 0310A
Descargado el 2017 de:
http://www.telecom.uff.br/~pablo/SitePlayer/SitePlayer_Development_Manual.pdf

[14] Witte Software (2017) Modbus-Slave
Descargado el 2017 de:
http://www.modbustools.com/modbus_slave.html

[15] Rockwell Software (2001) RSView32 User's Guide
Descargado el 2017 de:
http://literature.rockwellautomation.com/idc/groups/literature/documents/um/vw32-um001_-en-e.pdf

[16] Inkscape (s.f)
Descargado el 2017 de https://inkscape.org/en/

[17] Pentaho (s.f)
Descargado el 2017 de http://www.pentaho.com/

[18] Maxim Integrated (2014) Low-Power Slew-Rate-Limited RS-485/RS-422 Transceivers.
Descargado el 2017 de:
https://datasheets.maximintegrated.com/en/ds/MAX1487-MAX491.pdf

ANEXO A
CÓDIGO FUENTE USADO EN EL PROYECTO DE TESIS

A.1 Código fuente usado en microcontrolador PIC (Lenguaje C++)

Para la construcción del código en el microcontrolador PIC, se usó como referencia algunos de los ejemplos que vienen en el software CCS, ya que este software permite programar al PIC usando el lenguaje C.

A.1.1 Parámetros Generales

En Fig. A.1 se puede observar la configuración más básica del PIC, correspondiente a la velocidad del reloj, baudiaje en la comunicación Modbus etc.

```
91  ///////////////    Configuracion general del PIC MODBUS MASTER  ////////////////
92  //                                                                            //
93  #include <16f877a.h>                                                         //
94  #fuses HS    //Uso HS ya que el cristal que voy a utilizar es mayor que 4Mhz  //
95  #use delay(clock=20Mhz)                                                      //
96  #include <math.h>                                                           //
97  #include <stdlib.h>                                                         //
98  //                                                                          //
99  ////////////////////////////////////////////////////////////////////////////////
100
101
102 ////////////////////// Bits indicadores de encendido    ////////////////////////
103 #use standard_io(D)
104
105 ///////      Configuracion de la comunicacion RS232 entre el     ///////////////
106 ///////          PIC MODBUS MASTER y una PC para pruebas.        ///////////////
107 //                                                                          //
108 //NOTA: Esta es la configuracion por defecto que se usa en el HIPERTERMINAL   //
109 #use rs232(baud=9600,parity=N,xmit=PIN_C1,rcv=PIN_C2,bits=8,stream=PC,errors) //
110 ////////////////////////////////////////////////////////////////////////////////
```

Fig. A.1. Configuración de parámetros generales del PIC [***Fuente**: Propia*]

A.1.2 Limpieza de Buffer de Transmisión

Esto sirve para no transmitir datos remanentes del buffer de transmisión hacia el Site Player, esto es muy importante, ya que no incluir este código puede corromper el proceso industrial, en Fig. A.2 se muestra la función encargada de eliminar los remanentes mencionados.

```
162 //  Esta funcion nos sirve para que no se ejecute ninguna instruccion        //
163 //  subsiguiente hasta que se haiga vaceado todo el buffer de transmision    //
164 //  de la UART.                                                              //
165 //  NOTA: Este tipo de funcion no es necesaria cuando usamos el              //
166 //  modo SOFTWARE.                                                           //
167 #define funcion_buffer_tx_limpio()\
168   {\
169     while(!TRMT);\
170   }                                                                          //
```

Fig. A.2. Limpieza de Buffer de Transmisión [***Fuente**: Propia*]

A.1.3 Comunicación con el Site Player

En la imagen Fig. A.3 se muestra el proceso de escritura que el PIC ejecuta sobre el SitePlayer, es importante tener en claro que es el microcontrolador PIC el que siempre tiene la iniciativa de comunicación, o dicho en otras palabras el SitePlayer nunca inicia la comunicación con el PIC, el SitePlayer solo se limita a responder.

```
434  void enviar_data_to_SP(int32 fx,int16 address_SitePlayer){
435
436      int8 comando_WriteX;
437
438      // Antes de enviar la trama al Siteplayer deshabilito la
439      // interrupcion por flanco de bajada, para que no se interrumpa
440      // el envio de la trama.
441         //disable_interrupts(INT_EXT);//* ESTO YA NO ES NECESARIO YA QUE LA
442      //                                 DESHABILITACION LO HE REALIZADO EN
443      //                                 "transaccion_read_MB_and_write_SP" EL
444      //                                 CUAL CONTIENE A "enviar_data_to_SP".
445
446  //    El comando de escritura lo generamos con el sgt comando:
447  //    WriteX = 90h => Write Using Extended Two Byte Addressing
448  //    Para escribir la cantidad de bytes deseados se debe usar :
449  //    WriteX +  cant_de_bytes -1
450       comando_WriteX = 0x90 + 4 - 1;
451
452       fputc(comando_WriteX,SP_RS232);       //Comando escritura WriteX
453
454       fputc(make8(address_SitePlayer,0),SP_RS232);
455
456       fputc(make8(address_SitePlayer,1),SP_RS232);
457
458       fputc(make8(fx,0),SP_RS232);
459
460       fputc(make8(fx,1),SP_RS232);
461
462       fputc(make8(fx,2),SP_RS232);
463
464       fputc(make8(fx,3),SP_RS232);
465
466       // Habiltacion de la interrupcion de flanco de bajada,
467       // esto se vuelve a habilitar despues de haber enviado
468       // toda la trama al Siteplayer.
469          //enable_interrupts(INT_EXT);//**
```

Fig. A.3. Escritura de datos sobre él SitePlayer [***Fuente**: Propia*]

A.1.4 Estructura de la trama Modbus

Basándose en las especificaciones del protocolo Modbus, se construye la estructura que se muestra en la Fig. A.4. el cual almacenará los datos de la trama Modbus, estructura que será utilizada tanto para las tramas de lectura y escritura.

```
310  struct {
311     int8 address;
312     int8 len;
313     function func;
314     exception error;
315     int8 data[MODBUS_SERIAL_RX_BUFFER_SIZE];
316  } modbus_rx;
317
```

```
304  union {
305     int8 b[2];
306     int16 d;
307  } modbus_serial_crc;
308
```

Fig. A.4. Estructura de la trama Modbus [***Fuente**: Propia*]

A.1.5 Funciones para la recepción de los componentes de la trama Modbus

Las funciones que se muestran en la Fig. A.5. permiten recibir los componentes de la trama Modbus que son enviados por el Modbus Esclavo, o RTU, ya sean de lectura o de escritura. Posteriormente estos componentes de la trama son colocados en la estructura que se muestra en la Fig. A.4.

Estas funciones de verificación están basadas en las especificaciones del Protocolo Modbus RTU [7].

```
520 int1 kbhit_address_MB(int8 direccion_MB){ // Verificado
521
522    if(kbhit(MODBUS_SERIAL)){
523       if(direccion_MB == fgetc(MODBUS_SERIAL)){
524          modbus_rx.address = direccion_MB;
525          modbus_calc_crc(direccion_MB);
526          return TRUE;
527       }else
528          return FALSE;
529    }else
530          return FALSE;
531 }
532
533 int1 kbhit_funcion_MB(){ // Verificado.
534
535       if(kbhit(MODBUS_SERIAL)){
536          modbus_rx.func = fgetc(MODBUS_SERIAL);
537          modbus_calc_crc(modbus_rx.func);
538          return TRUE;
539       }else
540          return FALSE;
541  }
542
543 int1 kbhit_data_crc_MB(){
544
545       if(kbhit(MODBUS_SERIAL)){
546          modbus_rx.data[modbus_rx.len] = fgetc(MODBUS_SERIAL);
547          modbus_calc_crc(modbus_rx.data[modbus_rx.len]);
548          modbus_rx.len++;
549          //fprintf(PC,"Data %x",modbus_rx.data[modbus_rx.len]);**
550          return TRUE;
551       }else{
552          return FALSE;
553       }
554 }
```

Fig. A.5. Verificación de la Trama [***Fuente**: Propia*]

A.1.6 Funciones para la generación de la trama de lectura Modbus

La imagen Fig. A.6 muestra las funciones que son invocadas cada vez que se ejecuta un comando de lectura Modbus. Como podemos observar en esta misma imagen, el comando de lectura que se ejecute depende del dato al cual se desee acceder, como sabemos hay 4 tipos de datos: Input Discrete, Coil, Input Register, Holding Register.

```
void transaccion read MB and write SP(int8 numero nodo modbus,type data tipo de dato,
     posbit_datanumeric posbit_datanumerico,int8 starting_address,int16 address_SP){

  int8 cantidad_de_regitros;
  retry_count_read_MB = 3;
  // Proceso de lectura de los esclavos Modbus.
  switch(tipo_de_dato){

    case INPUT_DISCRETE :

     do{
        modbus_read_discrete_input(numero_nodo_modbus,starting_address,0x0001);
     }while((modbus_rx.error == TIMEOUT) && --retry_count_read_MB);

    break;

    case COIL :

     do{
        modbus_read_coils(numero_nodo_modbus,starting_address,0x0001);
     }while((modbus_rx.error == TIMEOUT) && --retry_count_read_MB);

    break;

    case INPUT_REGISTER :

     if((posbit_datanumerico==Float_Little_Endian) || (posbit_datanumerico==Float_Word_Swapped))
        cantidad_de_regitros = 2;
     else
        cantidad_de_regitros = 1;

     do{
        modbus_read_input_registers(numero_nodo_modbus,starting_address,cantidad_de_regitros);
     }while((modbus_rx.error == TIMEOUT) && --retry_count_read_MB);

    break;

    case HOLDING_REGISTER :

      if((posbit_datanumerico==Float_Little_Endian) || (posbit_datanumerico==Float_Word_Swapped))
        cantidad_de_regitros = 2;
     else
        cantidad_de_regitros = 1;

     do{
        modbus_read_holding_registers(numero_nodo_modbus,starting_address,cantidad_de_regitros);
     }while((modbus_rx.error == TIMEOUT) && --retry_count_read_MB);

    break;
```

Fig. A.6. Funciones para la generación de comandos de lectura Modbus [***Fuente**: Propia*]

A.1.7 Funciones para la generación de la trama de escritura Modbus

Como sabemos solo se puede generar la escritura Modbus sobre 2 tipos de datos: Coil y Holding Register.

En la imagen Fig. A.7. se muestra la función que se encarga de generar el comando de escritura Modbus en un tipo de dato Coil.

En la imagen Fig. A.8. se muestra la función que se encarga de generar el comando de escritura Modbus en un tipo de dato Holding Register.

Es importante tener en cuenta que el microcontrolador PIC, quien hace el papel de Modbus Maestro, solo genera la trama de escritura cuando recibe la orden del SitePlayer a través de un flanco de bajada, de 5VDC a 0VDC. Luego de ello el SitePlayer le entregará al PIC, el dato que este último tendrá que escribir en el Modbus Esclavo o RTU.

```
void transaccion_write_MB(int8 numero_nodo_modbus,type_data tipo_de_dato,posbit_datanumeric posbit_datanumerico,int8 starting_address, float value){

    int8 cantidad_de_regitros;
    int16 reg_array[2];
    int16 reg;
    retry_count_write_MB = 2;
    retry_count_read_MB = 2;

    // Proceso de escritura de los esclavos Modbus.

    switch(tipo_de_dato){

      case COIL :

          if( (value ==1) || (value ==0)){//Me aseguro que el valor es de tipo binario.

             //fprintf(PC,"El valor de float value = %f y el bit = %u y el starting addres es %u\r\n",value,(int1)value,starting_address);
             do{
                modbus_write_single_coil(numero_nodo_modbus,starting_address, (int1)value);
             }while((modbus_rx.error == TIMEOUT) && --retry_count_write_MB);

          }
      break;
```

Fig. A.7. Función para la generación de comandos de escritura Modbus en un dato de tipo Coil [***Fuente**: Propia*]

```
case HOLDING_REGISTER :
    if(posbit_datanumerico == Float_Little_Endian){    // Verificado !!
       cantidad_de_regitros = 2;
       reg_array[1] = make16(make8(f_PICtoIEEE(value),1),make8(f_PICtoIEEE(value),0));
       reg_array[0] = make16(make8(f_PICtoIEEE(value),3),make8(f_PICtoIEEE(value),2));

       do{
          modbus_write_multiple_registers(numero_nodo_modbus,starting_address,cantidad_de_regitros,reg_array);
       }while((modbus_rx.error == TIMEOUT) && --retry_count_write_MB);

    }else if(posbit_datanumerico == Float_Word_Swapped){  // Verificado !!
       cantidad_de_regitros = 2;
       reg_array[0] = make16(make8(f_PICtoIEEE(value),1),make8(f_PICtoIEEE(value),0));
       reg_array[1] = make16(make8(f_PICtoIEEE(value),3),make8(f_PICtoIEEE(value),2));

       do{
          modbus_write_multiple_registers(numero_nodo_modbus,starting_address,cantidad_de_regitros,reg_array);
       }while((modbus_rx.error == TIMEOUT) && --retry_count_write_MB);

    }else if(posbit_datanumerico == Entero_de_16_bits) {  // Verificado !! incluso con decimales !!

       do{
          modbus_write_single_register(numero_nodo_modbus,starting_address,(int16)value);
       }while((modbus_rx.error == TIMEOUT) && --retry_count_write_MB);

    }else if( (posbit_datanumerico >= BIT_0) && (posbit_datanumerico <= BIT_15) ){
       cantidad_de_regitros = 1;
       do{
         modbus_read_holding_registers(numero_nodo_modbus,starting_address,cantidad_de_regitros);
       }while((modbus_rx.error == TIMEOUT) && --retry_count_read_MB);

       if( (retry_count_read_MB >0) && (modbus_rx.error == OK)    ){

           if(value ==0){
             reg = make16(modbus_rx.data[1],modbus_rx.data[2]);

             bit_clear(reg,posbit_datanumerico-1);

           }else if(value ==1){
             reg = make16(modbus_rx.data[1],modbus_rx.data[2]);

             bit_set(reg,posbit_datanumerico-1);
           }
           do{
              modbus_write_single_register(numero_nodo_modbus,starting_address,reg);
           }while((modbus_rx.error == TIMEOUT) && --retry_count_write_MB);

       }
    }

 break;
```

Fig. A.8. Funciones para la generación de comandos de escritura Modbus en un dato de tipo Holding Register. [***Fuente**: Propia*]

A.1.8 Código raíz del microcontrolador PIC

En la Fig. A.9. se muestra el código raíz que se ejecuta en el microcontrolador PIC, tal código incluye funciones que permiten configurar la interrupción de flanco de bajada de 5VDC a 0VDC, sincronizar la baudiaje con el cual se realizara la comunicación entre el microcontrolador PIC y el SitePlayer, configurar el Watchdog Timer que sirve para reanudar la ejecución de tareas del PIC en caso este último se haya "colgado". Por último, se ejecuta el bucle infinito, while (1), el cual se usa para la generación de los comandos de lectura Modbus y la posterior escritura de los datos obtenidos en el SitePlayer.

```
2145  void main(void){
2146
2147   //output_high(PIN_D0);
2148
2149   // Este delay es para esperar que el Siteplayer Inicializa
2150      restart_wdt();delay_ms(1000);
2151      restart_wdt();delay_ms(1000);
2152      restart_wdt();delay_ms(1000);
2153
2154   //Configuracion de las interrupciones
2155   funcion_configurar_interrupciones();
2156
2157   //Envio de 20 nulls.
2158   funcion_NOP();
2159
2160   //Modificar el Baudiaje(si es que es posible) del PIC MODBUS MASTER y SP.
2161   #if (TX_RX_SITEPALYER_RS232_MODO == HARDWARE)
2162      funcion_ADAPTAR_BAUD_SP_Y_PIC();
2163   #endif
2164
2165   //Visualizar los datos obtenidos de la BD.
2166   funcion_show_data_from_BD_to_LCD_PC();
2167
2168   #if (MODO_DE_TRABAJO == COMANDOS_MODBUS)
2169        fprintf(PC,"Actualmente esta en el modo COMANDOS_MODBUS\r\n");
2170        fprintf(PC,"Si quiere ver valores atraves del Hiperterminal  \r\n");
2171        fprintf(PC,"cambie al modo PRUEBA_PC  \r\n\r\n\r\n");
2172   #endif
2173
2174
2175   #fuses WDT
2176   setup_wdt(WDT_2304MS);
2177
2178   switch ( restart_cause() ) {
2179   case NORMAL_POWER_UP:
2180     output_bit( PIN_D0, 1);
2181   break;
2182   case WDT_TIMEOUT:
2183     output_bit( PIN_D7, 1);
2184   break;
2185   }
2186
2187     while(1){
2188         restart_wdt();
2189         ramdon_value_to_MBMA00X();
2190         funcion_cmd_read_MB_write_SP();
2191         //funcion_comando_LCD();
2192      }
2193   }
```

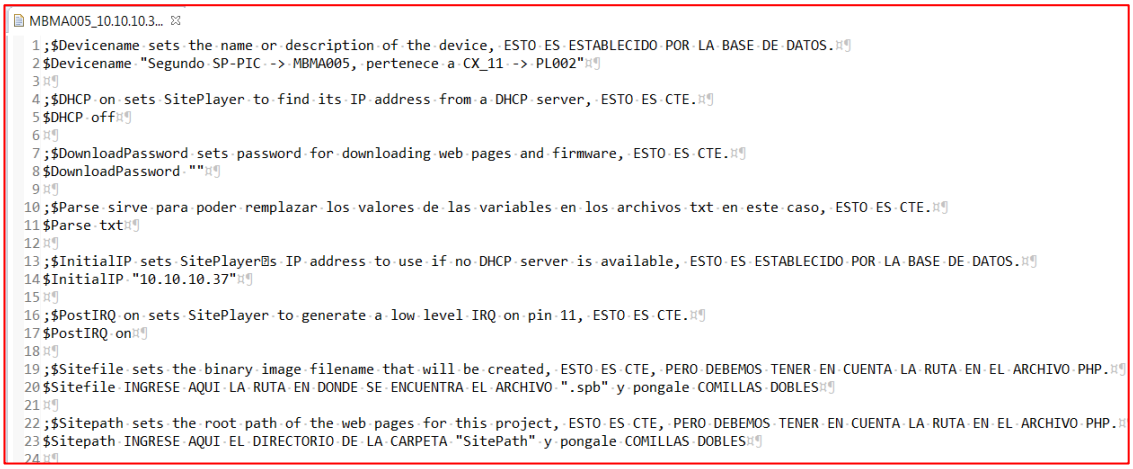
Fig. A.9. Programa principal del microcontrolador PIC, a partir de este código se invocan al resto de funciones que hay en el PIC. [***Fuente**: Propia*]

A.2 Código fuente implementado en el SitePlayer

A.2.1 Parámetros Generales

Si bien en el micro-servidor web SitePlayer no hay una programación propiamente dicha, lo que sí hay es, una serie de archivos los cuales tienen como finalidad entre otras cosas, configurar la dirección IP del SitePlayer, contraseña para la autenticación, flanco de caída de voltaje de 5VDC a 0VDC para la interrupción del PIC, entre otros.

En la Fig. A.10. se muestra parte de la configuración mencionada. Se recomienda revisar las Fig. 3.28, Fig. 3.31 y Fig. 3.32, ya que en tales imágenes se puede observar cómo es el proceso de creación de los archivos del SitePlayer desde el sistema SCADA.



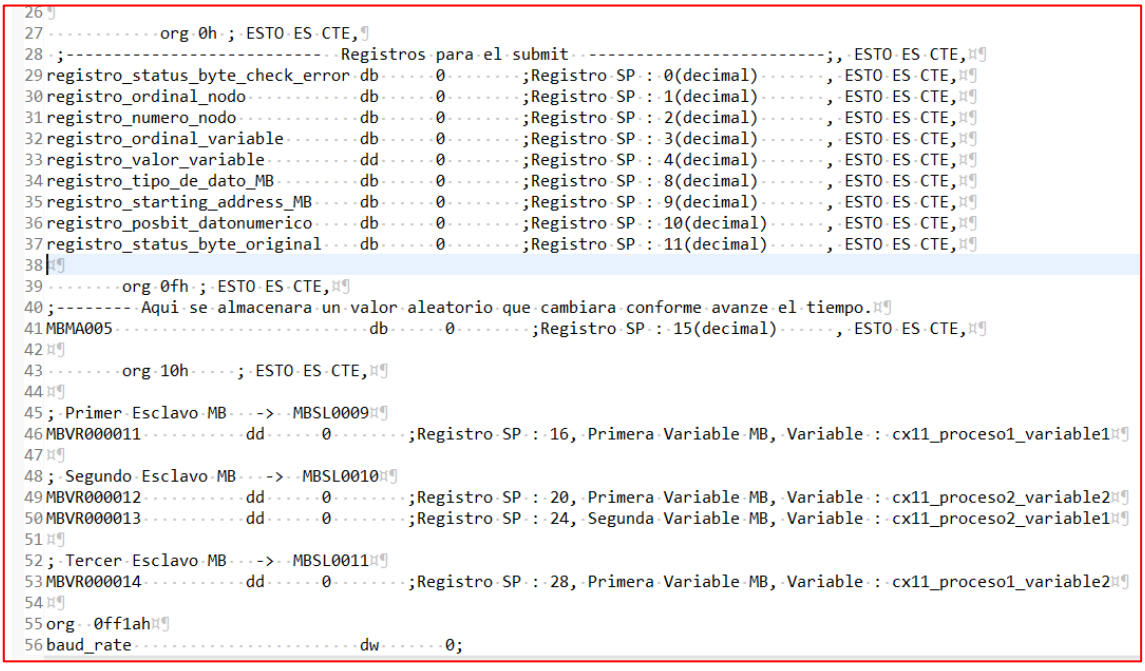
```
MBMA005_10.10.10.3...
;$Devicename sets the name or description of the device, ESTO ES ESTABLECIDO POR LA BASE DE DATOS.
$Devicename "Segundo SP-PIC -> MBMA005, pertenece a CX_11 -> PL002"

;$DHCP on sets SitePlayer to find its IP address from a DHCP server, ESTO ES CTE.
$DHCP off

;$DownloadPassword sets password for downloading web pages and firmware, ESTO ES CTE.
$DownloadPassword ""

;$Parse sirve para poder remplazar los valores de las variables en los archivos txt en este caso, ESTO ES CTE.
$Parse txt

;$InitialIP sets SitePlayer's IP address to use if no DHCP server is available, ESTO ES ESTABLECIDO POR LA BASE DE DATOS.
$InitialIP "10.10.10.37"

;$PostIRQ on sets SitePlayer to generate a low level IRQ on pin 11, ESTO ES CTE.
$PostIRQ on

;$Sitefile sets the binary image filename that will be created, ESTO ES CTE, PERO DEBEMOS TENER EN CUENTA LA RUTA EN EL ARCHIVO PHP.
$Sitefile INGRESE AQUI LA RUTA EN DONDE SE ENCUENTRA EL ARCHIVO ".spb" y pongale COMILLAS DOBLES

;$Sitepath sets the root path of the web pages for this project, ESTO ES CTE, PERO DEBEMOS TENER EN CUENTA LA RUTA EN EL ARCHIVO PHP.
$Sitepath INGRESE AQUI EL DIRECTORIO DE LA CARPETA "SitePath" y pongale COMILLAS DOBLES
```

Fig. A.10. Configuración de los parámetros que determinan el comportamiento del micro-servidor web SitePlayer [***Fuente**: Propia*]

A.2.2 Configuración de la memoria RAM del SitePlayer

En la Fig. A.11. se muestra como es el mecanismo de asignación entre los registros de la memoria RAM del SitePlayer y las variables que se piensan extraer de los Modbus Esclavos (o RTUs), no olvidar que el SitePlayer es parte del Modbus Master (o Gateway).

Se le recomienda al lector que revise la Fig. 3.43, ya que la estructura que se muestra en la Fig. A.11 ha sido extraída de la figura mencionada.



```
            org 0h ; ESTO ES CTE,
;--------------------------- Registros para el submit -------------------------;, ESTO ES CTE,
registro_status_byte_check_error db      0       ;Registro SP : 0(decimal)      , ESTO ES CTE,
registro_ordinal_nodo            db      0       ;Registro SP : 1(decimal)      , ESTO ES CTE,
registro_numero_nodo             db      0       ;Registro SP : 2(decimal)      , ESTO ES CTE,
registro_ordinal_variable        db      0       ;Registro SP : 3(decimal)      , ESTO ES CTE,
registro_valor_variable          dd      0       ;Registro SP : 4(decimal)      , ESTO ES CTE,
registro_tipo_de_dato_MB         db      0       ;Registro SP : 8(decimal)      , ESTO ES CTE,
registro_starting_address_MB     db      0       ;Registro SP : 9(decimal)      , ESTO ES CTE,
registro_posbit_datonumerico     db      0       ;Registro SP : 10(decimal)     , ESTO ES CTE,
registro_status_byte_original    db      0       ;Registro SP : 11(decimal)     , ESTO ES CTE,

        org 0fh ; ESTO ES CTE,
;-------- Aqui se almacenara un valor aleatorio que cambiara conforme avanze el tiempo.
MBMA005                          db      0       ;Registro SP : 15(decimal)     , ESTO ES CTE,

        org 10h     ; ESTO ES CTE,

; Primer Esclavo MB    ->   MBSL0009
MBVR000011           dd      0       ;Registro SP : 16, Primera Variable MB, Variable : cx11_proceso1_variable1

; Segundo Esclavo MB    ->   MBSL0010
MBVR000012           dd      0       ;Registro SP : 20, Primera Variable MB, Variable : cx11_proceso2_variable2
MBVR000013           dd      0       ;Registro SP : 24, Segunda Variable MB, Variable : cx11_proceso2_variable1

; Tercer Esclavo MB    ->   MBSL0011
MBVR000014           dd      0       ;Registro SP : 28, Primera Variable MB, Variable : cx11_proceso1_variable2

org  0ff1ah
baud_rate                        dw      0;
```

Fig. A.11. Registros de la Memoria RAM del SitePlayer con sus respectivas variables de campo. [***Fuente**: Propia*]

A.3 Código fuente del Frontend (Lenguaje HTML, JavaScript y CSS)

A.3.1 Animación de gráficos del Proceso

En la Fig. A.12. se muestra el código fuente en JavaScript con el cual se realiza la extracción de datos del Servidor WEB y como estos datos son utilizados para la animación de los gráficos que representan a los procesos industriales. La imagen mencionada toma como referencia el proceso representado en la Fig. 4.26, la cual muestra la actualización del estado del solenoide de una válvula (cx11_proceso2_variable 2), la actualización del estado del final de carrera de esa misma válvula (cx11_proceso2_variable1) y como estas actualizaciones repercuten en el color (animación) de estos mismos.

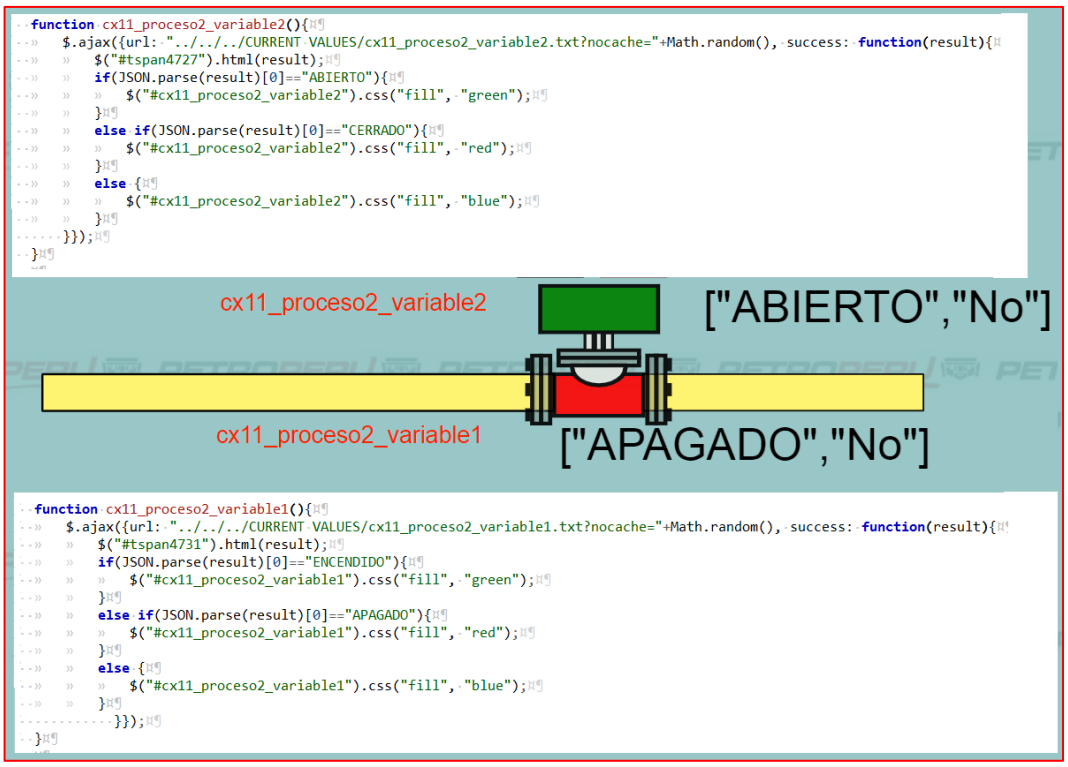
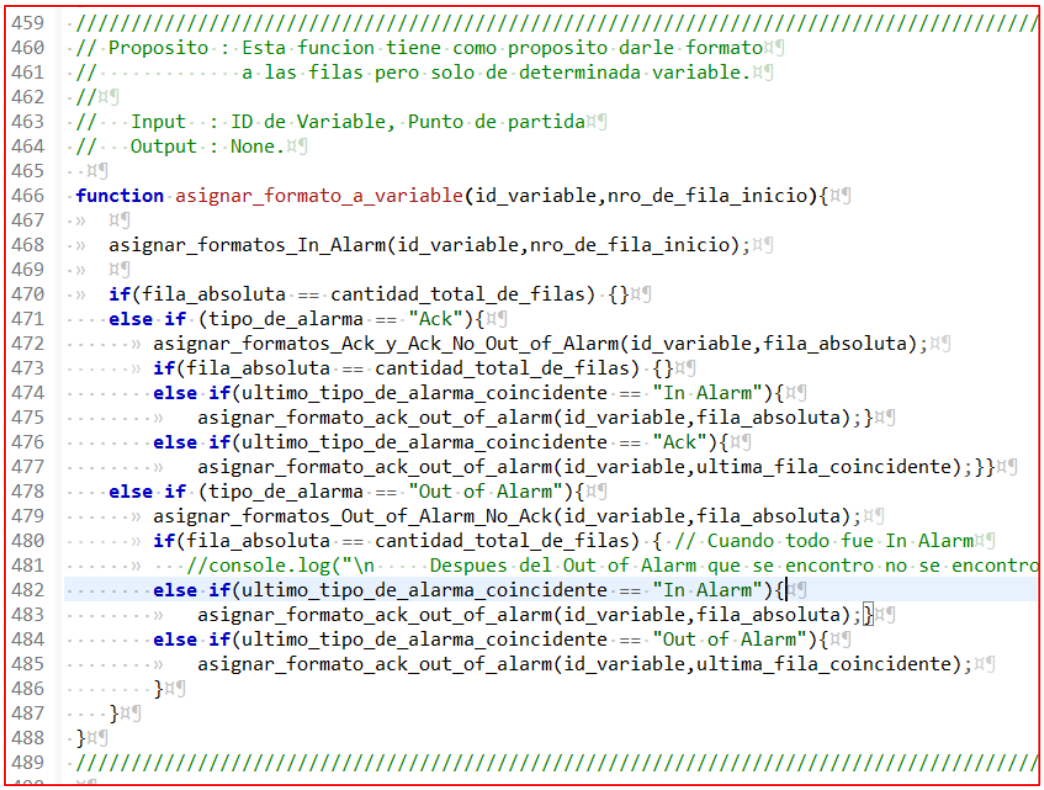


Fig. A.12. Animaciones asociadas al estado de un solenoide y al final de carrera de una válvula. **[*Fuente*: *Propia*]**

A.3.2 Animación del Banner de Alarmas

En la Fig. A.13. se muestra parte del código fuente utilizado para darle formato al banner de alarmas, como se recordará dependiendo del tipo de alarma (In-Alarm, Out of Alarm y Ack) en el que se encuentre una variable, está tomará diferentes aspectos en cuanto al color, así como al parpadeo, más detalles sobre estas animaciones lo puede ver en las Fig. 4.13, Fig. 4.14, Fig. 4.15 y Fig. 4.16.

En la Fig. A.14 se muestra parte del código fuente correspondiente a la emisión de sonido cuando se activa alguna de las alarmas, así como también el código correspondiente a la desactivación de una alarma ya sea porque esta fue reconocida por el operador de campo o porque la variable dejo de estar en condición de alarma.



```
////////////////////////////////////////////////////////////////////////////////
// Proposito : Esta funcion tiene como proposito darle formato
//            a las filas pero solo de determinada variable.
//
//   Input  : ID de Variable, Punto de partida
//   Output : None.

function asignar_formato_a_variable(id_variable,nro_de_fila_inicio){

    asignar_formatos_In_Alarm(id_variable,nro_de_fila_inicio);

    if(fila_absoluta == cantidad_total_de_filas) {}
  else if (tipo_de_alarma == "Ack"){
      asignar_formatos_Ack_y_Ack_No_Out_of_Alarm(id_variable,fila_absoluta);
      if(fila_absoluta == cantidad_total_de_filas) {}
        else if(ultimo_tipo_de_alarma_coincidente == "In Alarm"){
            asignar_formato_ack_out_of_alarm(id_variable,fila_absoluta);}
        else if(ultimo_tipo_de_alarma_coincidente == "Ack"){
            asignar_formato_ack_out_of_alarm(id_variable,ultima_fila_coincidente);}}
  else if (tipo_de_alarma == "Out of Alarm"){
      asignar_formatos_Out_of_Alarm_No_Ack(id_variable,fila_absoluta);
      if(fila_absoluta == cantidad_total_de_filas) { // Cuando todo fue In Alarm
      //console.log("\n    Despues del Out of Alarm que se encontro no se encontro
        else if(ultimo_tipo_de_alarma_coincidente == "In Alarm"){
            asignar_formato_ack_out_of_alarm(id_variable,fila_absoluta);}
        else if(ultimo_tipo_de_alarma_coincidente == "Out of Alarm"){
            asignar_formato_ack_out_of_alarm(id_variable,ultima_fila_coincidente);
        }
  }
}
////////////////////////////////////////////////////////////////////////////////
```

Fig. A.13. Código con el que se le da formato a banner de alarmas. **[*Fuente*: *Propia*]**

```
////////////////////////////////////////////////////////////////////////////////
// Proposito : Estas funciones tienen por finalidad prender y apagar
//            el sonido de las alarmas.
//
//    Input : None.
//   Output : None.
//
      function activar_sonido(){
      //console.log("Se activara el sonido");
        window.document.getElementById("sonido_alarma").play();
        window.document.getElementById("sonido_alarma").loop = true;
      }
      function pausar_sonido(){
      //console.log("Se pausara el sonido");
        window.document.getElementById("sonido_alarma").pause();
      }
////////////////////////////////////////////////////////////////////////////////
// Proposito : Esta funcion tiene como proposito mostrar y ocultar la
//            campana de la barra lateral.
//
//    Input : None.
//   Output : None.

      function apagar_campana(){
      window.document.getElementById("campana").style.visibility = "hidden";
      }
      function encender_campana(){
      window.document.getElementById("campana").style.visibility = "visible";
      }

////////////////////////////////////////////////////////////////////////////////
```

Fig. A.14. Código asociado a la emisión de sonido cuando una variable entra en alarma. **[*Fuente*: *Propia*]**

A.4 Código fuente del Backend (Lenguaje PHP)

A.4.1 Extracción de datos del SitePlayer desde el sistema SCADA

En la Fig. A.15 se muestra parte del código con el cual el sistema SCADA extrae los datos de un SitePlayer en particular, para ello se generó una trama HTTP con el lenguaje de programación PHP.

```
80  echo "SI fue posible comunicarse con el SitePlayer de IP :".$ip_modbus_master."\r\n\r\n";
81
82  // Takes a JSON encoded string and converts it into a PHP variable.
83  // When TRUE, returned objects will be converted into associative arrays.
84  $obj_value = json_decode($file,TRUE);
85
86  // Ahora verificamos que la logitud del array extraido al SitePlayer coincide con la logitud
87  // de los array que hemos mencionado lineas arriba(que tienen la misma longitud entre ellos).
88  if(count($obj_value) == count($obj_nombre_variable)){
89
90      echo "La longitud del array JSON extraido del SitePlayer coincide con \r\nla longitud de los arrays que hay en el archivo PHP. \r\n\r\n";
91
92      // Ahora lo que haremos es un barrido por las variables asocidas(VR00000X)
93      for($i = 0; $i < count($obj_nombre_variable); ++$i) {
94
95          if($obj_nombre_variable[$i] == ''){ // La variable Modbus(MBVR00000X) NO esta asociada a la variable, por que esta ultima ha sido LIBERADA.
96
97              echo "[".$i."]: "."La sgt variable MB : ".$obj_id_mb_variable[$i]." NO esta asociada a ninguna variable y por consiguiente no se insertara r
98
99          }else{
100
101             // Visualizacion de todas las variables con sus respectivos valores.
102             echo "[".$i."]: "."La sgt variable MB : ".$obj_id_mb_variable[$i]." esta asociada a la variable "."\r\n"."     ".$obj_nombre_variable[$i]."(
103
104             $handle = fopen('C:\Users\Tesis\Desktop\Aptana Studio 3 Workspace\Site WEB 11\Site WEB 11 v2\CURRENT RAW VALUES/'.$obj_id_variable[$i].".txt"
105
106             if(fwrite($handle,funcion_inversa($obj_value[$i])) == FALSE) {
107                 echo "No se pudo escribir en".$obj_id_variable[$i].".txt"."\r\n"."\r\n";
108             }else {
109                 echo "Se escribio correctamente en: ".$obj_id_variable[$i].".txt"."\r\n"."\r\n";
110             }
111         }
112     }
```

Fig. A.15. Código asociado a la extracción de los datos que el SitePlayer aloja en su memoria RAM. **[*Fuente*: *Propia*]**

A.4.2 Linealización de parámetros físicos

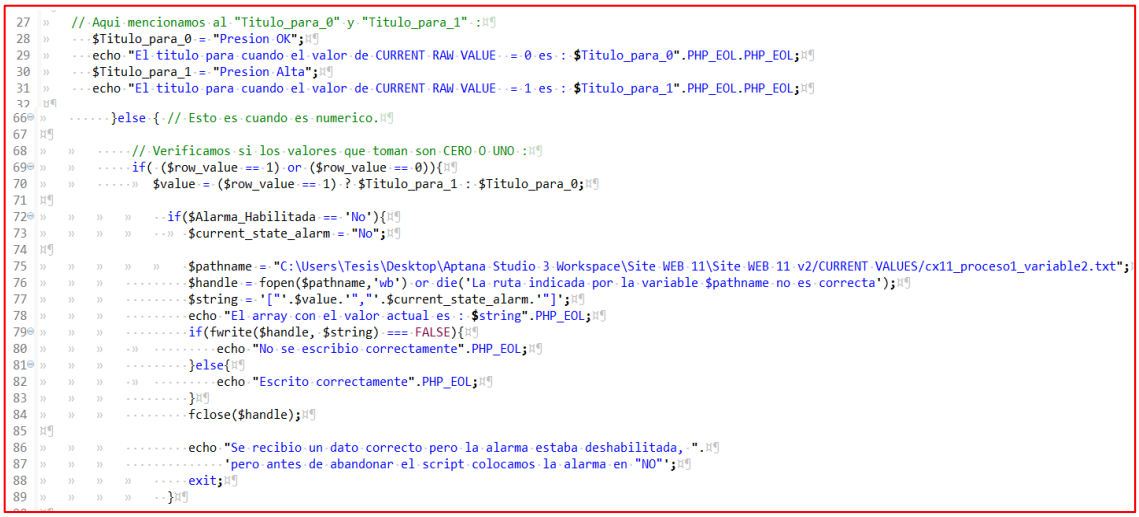
En la Fig. A.16. se muestra el código relacionado con la linealización de un parámetro físico en particular, el proceso de la linealización de parámetros físicos se explicó en el capítulo 4.4.1.

```
27      // Aqui mencionamos al Escalador, Offset y Unidades.
28      $Escalador = "18.4";
29      $Offset    = "-2.67";
30      $Unidades  = "psi";
31
71          echo "El valor obtenido de la carpeta CURRENT RAW VALUES es numerico";
72              echo "\r\n";
73
74              // Realizamos el proceso de linealizacion ax+b :
75              $value = $Escalador*$row_value + $Offset;
76              echo 'El var_dump de $value es :';
77              var_dump($value);
78              echo "\r\n";
79
80              if($Alarma_Habilitada == 'No'){
81
82                  $current_state_alarm = "OK";
83
84                      $pathname = "C:\Users\Tesis\Desktop\Aptana Studio 3 Workspace\Site WEB 11\Site WEB 11 v2/CURRENT VALUES/cx11_proceso1_vari
85                  $handle = fopen($pathname,'wb') or die('La ruta indicada por la variable $pathname no es correcta.');
86                  $string = '["'.$value.'","'.$current_state_alarm.'"]';
87
88                  if(fwrite($handle, $string) === FALSE){
89                      echo "No se escribio correctamente sobre la carpeta CURRENT VALUES.".PHP_EOL.PHP_EOL;
90                  }else{
91                      echo "Si se escribio correctamente sobre la carpeta CURRENT VALUES.".PHP_EOL.PHP_EOL;
92                  }
93
94                  fclose($handle);
95
96                  echo "Ya que la alarma esta deshabilitada,".
97                      ' entonces estableceremos la alarma en "OK".'.PHP_EOL;
98                  exit;
99
100             }
```

Fig. A.16. Código asociado a la linealización de un parámetro físico. **[*Fuente*: *Propia*]**

A.4.3 Titulación de estados físicos

En la Fig. A.17. se muestra el código relacionado con la titulación de un estado físico en particular. El proceso de la titulación de estados físicos se explicó en el capítulo 4.4.2.



```
// Aqui mencionamos al "Titulo_para_0" y "Titulo_para_1" :
    $Titulo_para_0 = "Presion OK";
    echo "El titulo para cuando el valor de CURRENT RAW VALUE  = 0 es : $Titulo_para_0".PHP_EOL.PHP_EOL;
    $Titulo_para_1 = "Presion Alta";
    echo "El titulo para cuando el valor de CURRENT RAW VALUE  = 1 es : $Titulo_para_1".PHP_EOL.PHP_EOL;

  }else { // Esto es cuando es numerico.

      // Verificamos si los valores que toman son CERO O UNO :
      if( ($row_value == 1) or ($row_value == 0)){
        $value = ($row_value == 1) ? $Titulo_para_1 : $Titulo_para_0;

          if($Alarma_Habilitada == 'No'){
            $current_state_alarm = "No";

              $pathname = "C:\Users\Tesis\Desktop\Aptana Studio 3 Workspace\Site WEB 11\Site WEB 11 v2/CURRENT VALUES/cx11_proceso1_variable2.txt";
              $handle = fopen($pathname,'wb') or die('La ruta indicada por la variable $pathname no es correcta');
              $string = '["'.$value.'","'.$current_state_alarm.'"]';
              echo "El array con el valor actual es : $string".PHP_EOL;
              if(fwrite($handle, $string) === FALSE){
                  echo "No se escribio correctamente".PHP_EOL;
              }else{
                  echo "Escrito correctamente".PHP_EOL;
              }
              fclose($handle);

              echo "Se recibio un dato correcto pero la alarma estaba deshabilitada, ".
              'pero antes de abandonar el script colocamos la alarma en "NO"';
              exit;
          }
```

Fig. A.17. Código asociado a la titulación de un estado físico. **[*Fuente*: *Propia*]**

A.4.4 Código asociado a la autenticación de un usuario en el sistema SCADA

En la Fig. A.18. se muestra lo que podría ocurrir cuando un usuario se registra en el sistema SCADA, si bien lo ideal es que el usuario pueda ingresar al sistema y hacer uso de este, pero por distintas razones muchas veces no ocurre ello, la imagen muestra cuáles podrían ser esas razones.

```
    switch ($datos_usuarios_out['resultado']){ //No nos olvidemos que el valor que tome este campo es como sigue
        //Usuario incorrecto                                    => resultado = 1;
        //Usuario correcto y contraseña incorrecta              => resultado = 2;
        //Usuario correcto y contraseña correcta, pero inhabilitado  => resultado = 3;
        //Usuario correcto y contraseña correcta y habilitado   => resultado = 4;
        case '1':
            echo "<body background=./Imagenes/Imagenes_de_fondo/Imagen_fondo_inicio.gif>";
            echo "<p>El usuario ingresado es incorrecto</p>";
            echo "</body>";
            break;
        case '2':
            echo "<body background=./Imagenes/Imagenes_de_fondo/Imagen_fondo_inicio.gif>";
            echo "<p>La clave ingresada es incorrecta</p>";
            echo "</body>";
            break;
        case '3':
            echo "<body background=./Imagenes/Imagenes_de_fondo/Imagen_fondo_inicio.gif>";
            echo "<p>El usuario ha sido deshabilitado, comuniquese con el DBA.</p>";
            echo "</body>";
            break;
        case '4': //Este seria el caso en el que todo esta OK
            session_start();
            $_SESSION = array_merge($datos_usuarios_out,$datos_BD_MBSiteplayer);
            header("Location:Pantallas/Seleccion_tareas.php");
            break;
        default:// Este seria el caso en el cual se intente acceder directamente a esta pagina
            // ya que en este caso $datos_usuarios_out['resultado'] = "".
            //header("Location:Index.html");
            header("Location:Index.php");
        break;
    }
```

Fig. A.18. Código asociado a los resultados de una autenticación de usuario. **[*Fuente*: *Propia*]**

A.5 Código fuente de la Base de Datos (Lenguaje SQL)

A.5.1. Tablas de las Clases del Espacio Sistema

En las Fig. A.19, Fig. A.20, Fig. A.21 y Fig. A.22 se muestran los códigos en lenguaje SQL, para la creación de tablas asociadas a las Clases del Espacio Sistemas, las cuales habían sido definidas en el capítulo 1.5, y son las siguientes: Planta, Proceso, Variable y Usuario.

```
14  ------------------------------------------------------   CREACION DE LA TABLA Lista_Plantas   ---------------
15  ------------------------------------------------------------------------------------------------------------
16  USE MBSiteplayer
17  CREATE TABLE Lista_Plantas (
18       Id_PL         varchar(50),  --PL00X
19       Nombre_Planta varchar(50)   NOT NULL UNIQUE,
20       Descripcion   varchar(200)   ,
21       Foto          varchar(100)  ,
22       Habilitado    varchar(50)   NOT NULL CONSTRAINT "DF_Lista_Plantas_Habilitado" DEFAULT ('Si'),
23       CONSTRAINT  PK_Lista_Plantas PRIMARY KEY (
24             Id_PL
25           ),
26       CONSTRAINT CK_Lista_Plantas_Habilitado CHECK (
27             Habilitado in ('Si','No')
28            )
29      )
30    GO
31
32  --  NOTA  :  En el caso del campo "Foto",este no almacenara la ruta de la foto, sino solamente el nombre,osea
33  --           no se guardaria algo asi : "/imagen/imagen de planta/CX15.jpg", sino simplemente "CX15.jpg"
34  --           La ruta completa se completara en el servidor WEB mendiante un programa SERVER SIDE , como php
35  --           por ejemplo.
```

Fig. A.19. Código SQL para la creación de una tabla, la cual contendrá información de todas las instancias de la clase Plantas. **[*Fuente*: *Propia*]**

```
236
237 ----------------------------CREACION DE LA TABLA Lista_Usuarios-----------------------------------------
238 ------------------------------------------------------------------------------------------------------------
239
240 CREATE TABLE Lista_Usuarios (
241       Id_Lista_US        varchar(50) , --US00X
242       Usuario            varchar(50)   NOT NULL UNIQUE,
243       Contraseña         varchar(50)   NOT NULL,
244       Nombre_y_Apellido  varchar(50)   NOT NULL UNIQUE,
245       Id_Privilegio      varchar(50)   NOT NULL,
246       Correo             varchar(50) ,
247       Foto               varchar(100),
248       Id_Planta          varchar(50)   NOT NULL,
249       Habilitado         varchar(50)   NOT NULL CONSTRAINT DF_Lista_Usuarios_Habilitado DEFAULT ('Si'),
250       CONSTRAINT PK_Lista_Usuarios PRIMARY KEY (
251             Id_Lista_US
252           ),
253       CONSTRAINT CK_Lista_Usuarios_Habilitado CHECK (
254             Habilitado in ('Si','No')
255            ),
256       CONSTRAINT FK_Lista_Usuarios_Id_Privilegio FOREIGN KEY (
257             Id_Privilegio
258          ) REFERENCES Lista_Privilegios (
259             Id_PR
260        )
261       --////////////////////////////////  ESTO SE ELIMINA   ////////////////////////////
262       --//CONSTRAINT FK_Lista_Usuarios_Id_Planta FOREIGN KEY (
263       --//       Id_Planta
264       --//   ) REFERENCES Lista_Plantas (
265       --//   Id_PL
266       --//  )
267       --////////////////////////////////////////////////////////////////////////////////
268     )
269   GO
```

Fig. A.20. Código SQL para la creación de una tabla, la cual contendrá información de todas las instancias de la clase Procesos. **[*Fuente*: *Propia*]**

```
----------------------------------------        CREACION DE LA TABLA Lista_Procesos          --------------------
-------------------------------------------------------------------------------------------------------------------


CREATE TABLE Lista_Procesos(
     Id_Proceso      varchar(50) ,  --PRC000X
     Nombre_Proceso  varchar(50)   NOT NULL UNIQUE,
     Id_Planta       varchar(50)   NOT NULL,
     Descripcion     varchar(200) ,
     Foto            varchar(100),
     Habilitado      varchar(50)   NOT NULL CONSTRAINT "DF_Lista_Procesos_Habilitado" DEFAULT ('Si'),
     CONSTRAINT  PK_Lista_Procesos PRIMARY KEY (
           Id_Proceso
         ),
     CONSTRAINT CK_Lista_Procesos_Habilitado CHECK (
           Habilitado in ('Si','No')
         ),
     CONSTRAINT FK_Lista_Procesos_Id_Planta FOREIGN KEY (
           Id_Planta
        ) REFERENCES Lista_Plantas (
           Id_PL
      )
)
 GO

--  NOTA  :  En el caso del campo "Foto",este no almacenara la ruta de la foto, sino solamente el nombre,osea
--           no se guardaria algo asi : "/imagen/imagen de proceso/SKIMER.jpg", sino simplemente "SKIMER.jpg"
--           La ruta completa se completara en el servidor WEB mediante un programa SERVER SIDE , como php
--           por ejemplo.
```

Fig. A.21. Código SQL para la creación de una tabla, la cual contendrá información de todas las instancias de la clase Procesos. **[*Fuente*: *Propia*]**

```
----------------------------CREACION DE LA TABLA Lista_Variables-----------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------

  CREATE TABLE Lista_Variables(
     Id_Variable                 varchar(50),  --VR00000X
     Nombre_Variable             varchar(50)   NOT NULL UNIQUE CLUSTERED,
     Id_Proceso                  varchar(50)   NOT NULL,
     Descripcion                 varchar(200),
     Foto                        varchar(50),
     Habilitado                  varchar(50)   NOT NULL  CONSTRAINT "DF_Lista_Variables_Habilitado" DEFAULT   ('Si'),  --Esto es habiltado para ESCRIBIR O NO.
     Tipo_de_dato                varchar(50)   NOT NULL,
     Alarma_Habilitada           varchar(50)   NOT NULL  CONSTRAINT "DF_Lista_Variables_Alarma_Habilitada" DEFAULT ('No'),
     Escalador                   varchar(10)   NOT NULL  CONSTRAINT "DF_Lista_Variables_Escalador" DEFAULT ('1'),
     Offset                      varchar(10)   NOT NULL  CONSTRAINT "DF_Lista_Variables_Offset" DEFAULT ('0'),
     Titulo_para_1               varchar(100)  NOT NULL  CONSTRAINT "DF_Lista_Variables_Titulo_para_1" DEFAULT ('1'),
     Titulo_para_0               varchar(100)  NOT NULL  CONSTRAINT "DF_Lista_Variables_Titulo_para_0" DEFAULT ('0'),
     Unidades                    varchar(50),
     Alarm_Analog_HiHi           varchar(50),
     Alarm_Analog_HiHi_Titulo    varchar(200),   -- Esto es solamente para la descripcion.
     Alarm_Analog_Hi             varchar(50),
     Alarm_Analog_Hi_Titulo      varchar(200),   -- Esto es solamente para la descripcion.
     Alarm_Analog_Lo             varchar(50),
     Alarm_Analog_Lo_Titulo      varchar(200),   -- Esto es solamente para la descripcion.
     Alarm_Analog_LoLo           varchar(50),
     Alarm_Analog_LoLo_Titulo    varchar(200),   -- Esto es solamente para la descripcion.
     Alarm_Digital_Trigger       varchar(50)   NOT NULL  CONSTRAINT "DF_Lista_Variables_Alarm_Digital_Trigger" DEFAULT ('1'),  -- Esto equivale a un umbral.
     Alarm_Digital_Trigger_Titulo varchar(200),   -- Esto es solamente para la descripcion.
     Maximo_Write_Analog         varchar(50),
     Minimo_Write_Analog         varchar(50),
     CONSTRAINT PK_Lista_Variables PRIMARY KEY(
          Id_Variable
       ),
     CONSTRAINT CK_Lista_Variables_Habilitado CHECK(
          Habilitado in ('Si','No')
       ),
     CONSTRAINT CK_Lista_Variables_Tipo_de_Dato CHECK(
          Tipo_de_dato in ('Analogico','Digital')
       ),
     CONSTRAINT CK_Lista_Variables_Alarma_Habilitada CHECK(
          Alarma_Habilitada in ('Si','No')
       ),
     CONSTRAINT CK_Lista_Variables_Alarm_Digital_Trigger CHECK(
          Alarm_Digital_Trigger in ('1','0')
       ),
     CONSTRAINT FK_Lista_Variables_Id_Proceso FOREIGN KEY (
          Id_Proceso
      )REFERENCES Lista_Procesos(
          Id_Proceso
        )
     )
   GO
```

Fig. A.22. Código SQL para la creación de una tabla, la cual contendrá información de todas las instancias de la clase Usuarios. **[*Fuente*: *Propia*]**

ANEXO B
ESPECIFICACIONES TÉCNICAS DEL GATEWAY

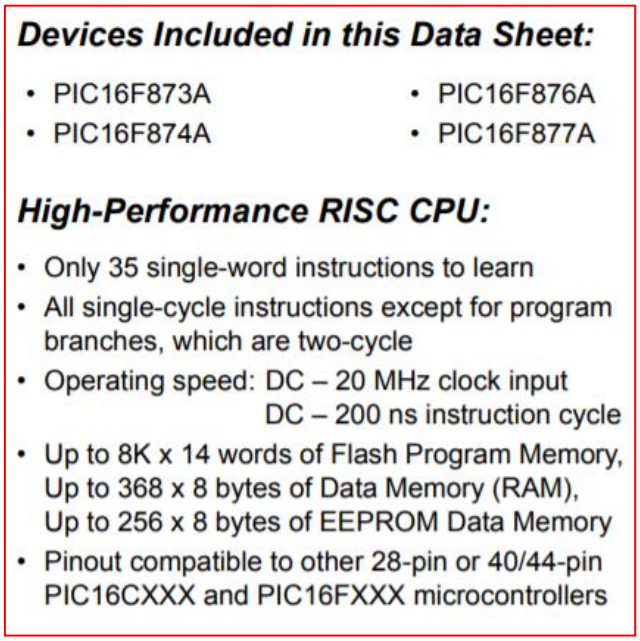
Como se puede observar en la Fig. 3.17 el Gateway está conformado por un microcontrolador PIC 16F877A y un micro-servidor WEB de la marca SitePlayer.

B.1 Especificaciones técnicas del microcontrolador PIC 16F877A

B.1.1 Especificaciones sobre el reloj y las memorias

En la Fig. B.1 se puede visualizar las características del PIC, velocidad de reloj: 20MHz, cantidad de memoria RAM: 368 Bytes y memoria EEPROM: 256 Bytes, información que fue extraída del datasheet del PIC.

Como se puede constatar, sus características son muy pobres, si lo comparamos con los dispositivos que al día de hoy podemos encontrar en el mercado, tales como los Raspberry, que tienen 1.4 GHz de procesador, 1 GB de memoria RAM (lo cual permite trabajar con una cantidad bastante elevada de variables) y además permite adicionar una tarjeta SD de 64GB.



Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

High-Performance RISC CPU:

- Only 35 single-word instructions to learn
- All single-cycle instructions except for program branches, which are two-cycle
- Operating speed: DC – 20 MHz clock input
  DC – 200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM),
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers

Fig. B.1. Especificaciones básicas del PIC 16F877A
[*Fuente*: ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf**]**

B.1.2 Mapeo de la memoria RAM del PIC 16F877A

En la Fig. B.2 se muestra la memoria RAM del PIC, aquí es donde se almacenan las variables temporales que hacen posible la creación de la Trama Modbus, solo 368 bytes podrán ser utilizados para el desarrollo del proyecto.

FIGURE 2-3: PIC16F876A/877A REGISTER FILE MAP

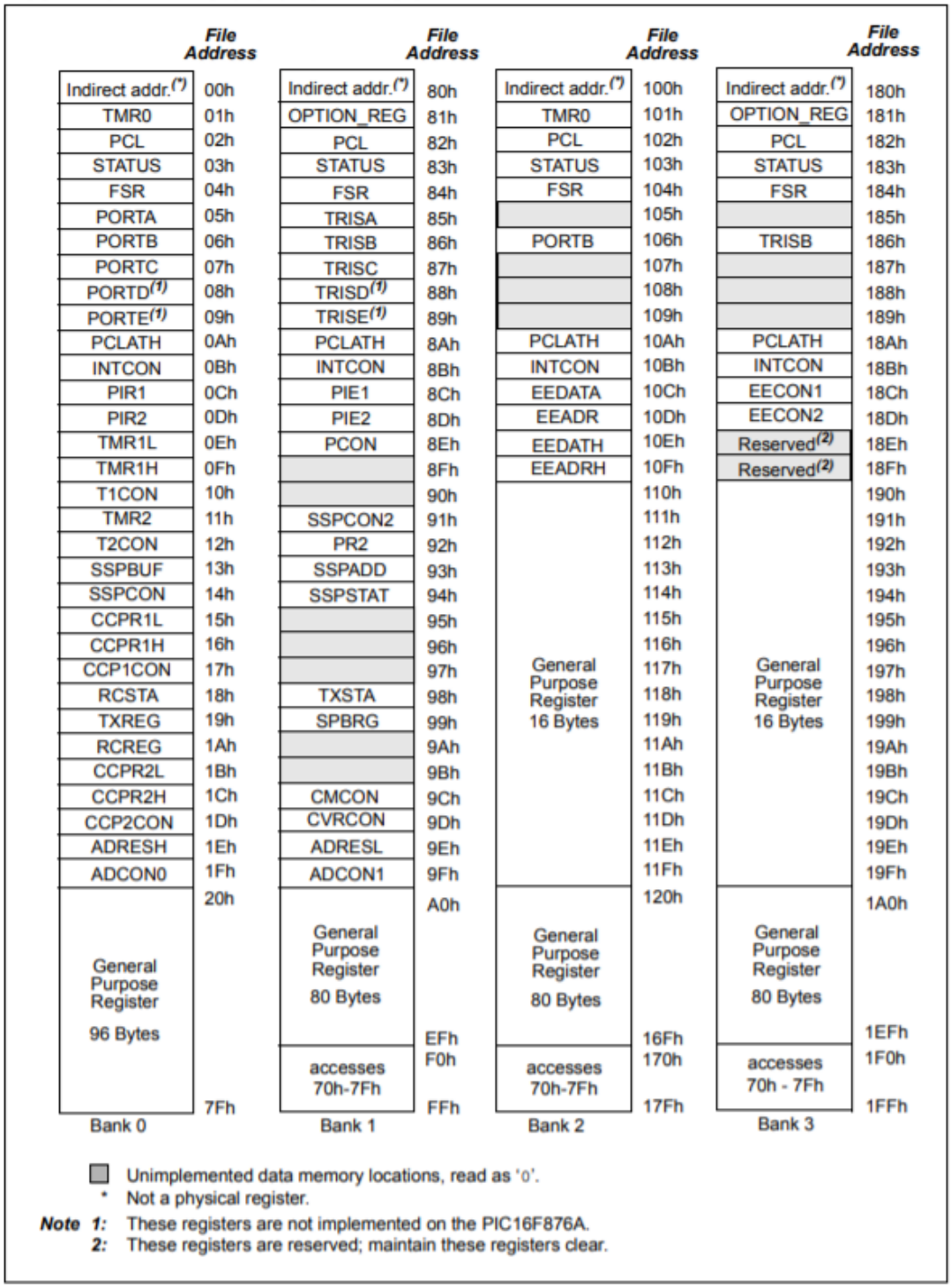


Fig. B.2. Mapa de la memoria RAM del microcontrolador PIC.
[*Fuente*: ww1.microchip.com/downloads/en/DeviceDoc/39582C.pdf]

B.1.3 Especificaciones técnicas del Conversor TTL/RS485 MAX485

Si bien el MAX485 no es un dispositivo que contenga programación, pero juega un rol muy importante ya que es el que lleva a cabo la conversión de voltaje unipolar a voltaje diferencial, lo cual hace posible que un Esclavo Modbus (o RTU) esté ubicado a 1200 metros de distancia del Maestro Modbus (o Gateway), algo que sería imposible con voltaje unipolar TTL. Las siguientes imágenes fueron extraídas del datasheet de MAX485 (Maxim Integrated – "Low-Power Slew-Rate-Limited RS-485/RS-422 Transceivers") [18].

La Fig. B.3 muestra las especificaciones básicas del MAX-485 y la Fig. B.4 muestra cómo se construye una red RS-485 mediante el uso de este dispositivo.

Selection Table

PART NUMBER	HALF/FULL DUPLEX	DATA RATE (Mbps)	SLEW-RATE LIMITED	LOW-POWER SHUTDOWN	RECEIVER/ DRIVER ENABLE	QUIESCENT CURRENT (µA)	NUMBER OF RECEIVERS ON BUS	PIN COUNT
MAX481	Half	2.5	No	Yes	Yes	300	32	8
MAX483	Half	0.25	Yes	Yes	Yes	120	32	8
MAX485	Half	2.5	No	No	Yes	300	32	8
MAX487	Half	0.25	Yes	Yes	Yes	120	128	8
MAX488	Full	0.25	Yes	No	No	120	32	8
MAX489	Full	0.25	Yes	No	Yes	120	32	14
MAX490	Full	2.5	No	No	No	300	32	8
MAX491	Full	2.5	No	No	Yes	300	32	14
MAX1487	Half	2.5	No	No	Yes	230	128	8

Fig. B.3. Especificaciones básicas del MAX-485.
[*Fuente*: https://datasheets.maximintegrated.com/en/ds/MAX1487-MAX491.pdf]

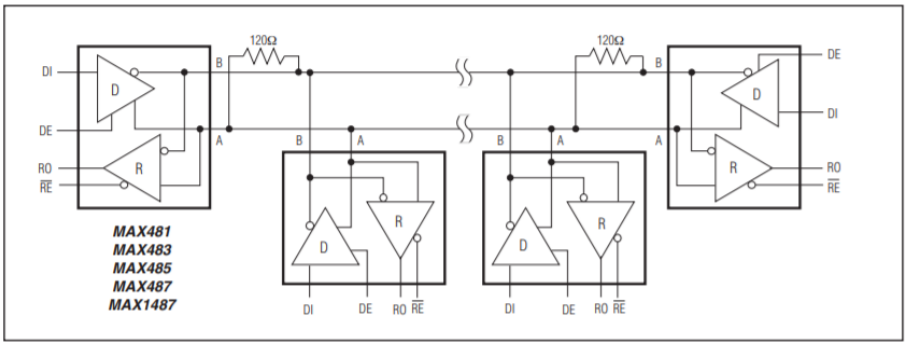


Figure 21. MAX481/MAX483/MAX485/MAX487/MAX1487 Typical Half-Duplex RS-485 Network

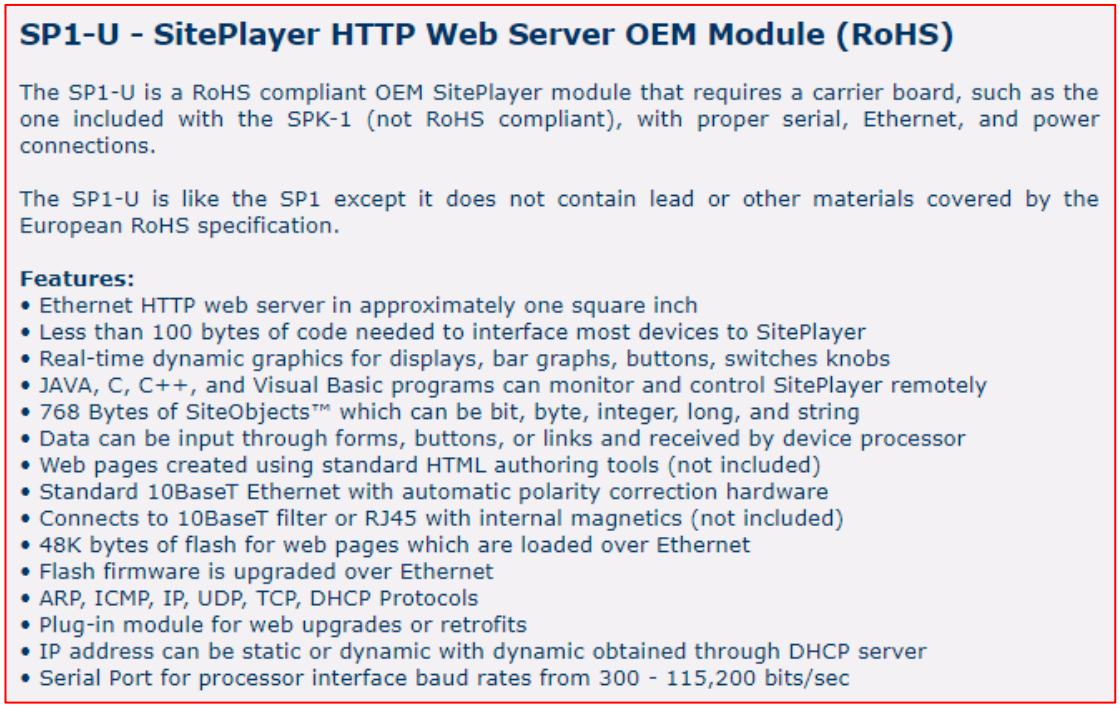
Fig. B.4. Esquema de conexionado del MAX-485 en una red RS-485.
[*Fuente*: https://datasheets.maximintegrated.com/en/ds/MAX1487-MAX491.pdf]

B.2 Especificaciones técnicas del SITEPLAYER.

En la Fig. B.5 se muestran las características del micro-servidor SitePlayer, las características más importantes a tener fueron: 768 bytes de memoria RAM, capacidad para ajustar la polaridad de la conexión Ethernet de forma automática, una velocidad de transmisión 10 Megabits por segundo a través de su puerto Ethernet, y una velocidad de transmisión de hasta 115200 bits por segundos a través de su puerto serial.

Pero al igual que sucedía con microcontrolador PIC, el SitePlayer también tiene características muy pobres si lo comparamos con dispositivos que hay en el mercado al

día de hoy como por ejemplo el Raspberry, el cual tiene una velocidad de transmisión Ethernet de 100 Megabits por segundo y un 1GB RAM.



SP1-U - SitePlayer HTTP Web Server OEM Module (RoHS)

The SP1-U is a RoHS compliant OEM SitePlayer module that requires a carrier board, such as the one included with the SPK-1 (not RoHS compliant), with proper serial, Ethernet, and power connections.

The SP1-U is like the SP1 except it does not contain lead or other materials covered by the European RoHS specification.

Features:
- Ethernet HTTP web server in approximately one square inch
- Less than 100 bytes of code needed to interface most devices to SitePlayer
- Real-time dynamic graphics for displays, bar graphs, buttons, switches knobs
- JAVA, C, C++, and Visual Basic programs can monitor and control SitePlayer remotely
- 768 Bytes of SiteObjects™ which can be bit, byte, integer, long, and string
- Data can be input through forms, buttons, or links and received by device processor
- Web pages created using standard HTML authoring tools (not included)
- Standard 10BaseT Ethernet with automatic polarity correction hardware
- Connects to 10BaseT filter or RJ45 with internal magnetics (not included)
- 48K bytes of flash for web pages which are loaded over Ethernet
- Flash firmware is upgraded over Ethernet
- ARP, ICMP, IP, UDP, TCP, DHCP Protocols
- Plug-in module for web upgrades or retrofits
- IP address can be static or dynamic with dynamic obtained through DHCP server
- Serial Port for processor interface baud rates from 300 - 115,200 bits/sec

Fig. B.5. Especificaciones técnicas del micro-servidor Web SitePlayer.

B.3 Imagen del hardware usado en el proyecto de Tesis

En la imagen Fig. B.6 se muestra todo el hardware que se usó en el proyecto de Tesis, básicamente este consta de lo siguiente:

- Laptop: Donde se almacena el Servidor WEB, el Servidor de Base de Datos, el Servidor de Reportes y el emulador de Modbus Esclavos.
- SitePlayer: Micro-servidor WEB con el cual se lleva la información del Modbus Slave a la red Ethernet, junto con el PIC conforman lo que es el Gateway (o Modbus Master).
- PIC 16F877A: Dispositivo que tienen implementado el protocolo Modbus Master mediante el lenguaje de programación C++, junto con el SitePlayer conforman el Gateway.
- Conversor USB/RS232 y Conversor RS232/RS485: Dispositivos que hacen posible que la Laptop emule a un Modbus Slave.
- Switch Ethernet: Dispositivo que posibilita la comunicación entre el Laptop y el SitePlayer mediante una topología estrella.
- Fuente Regulable: Proporciona 9 VDC para alimentar al SitePlayer y al PIC.
- Cables Ethernet.

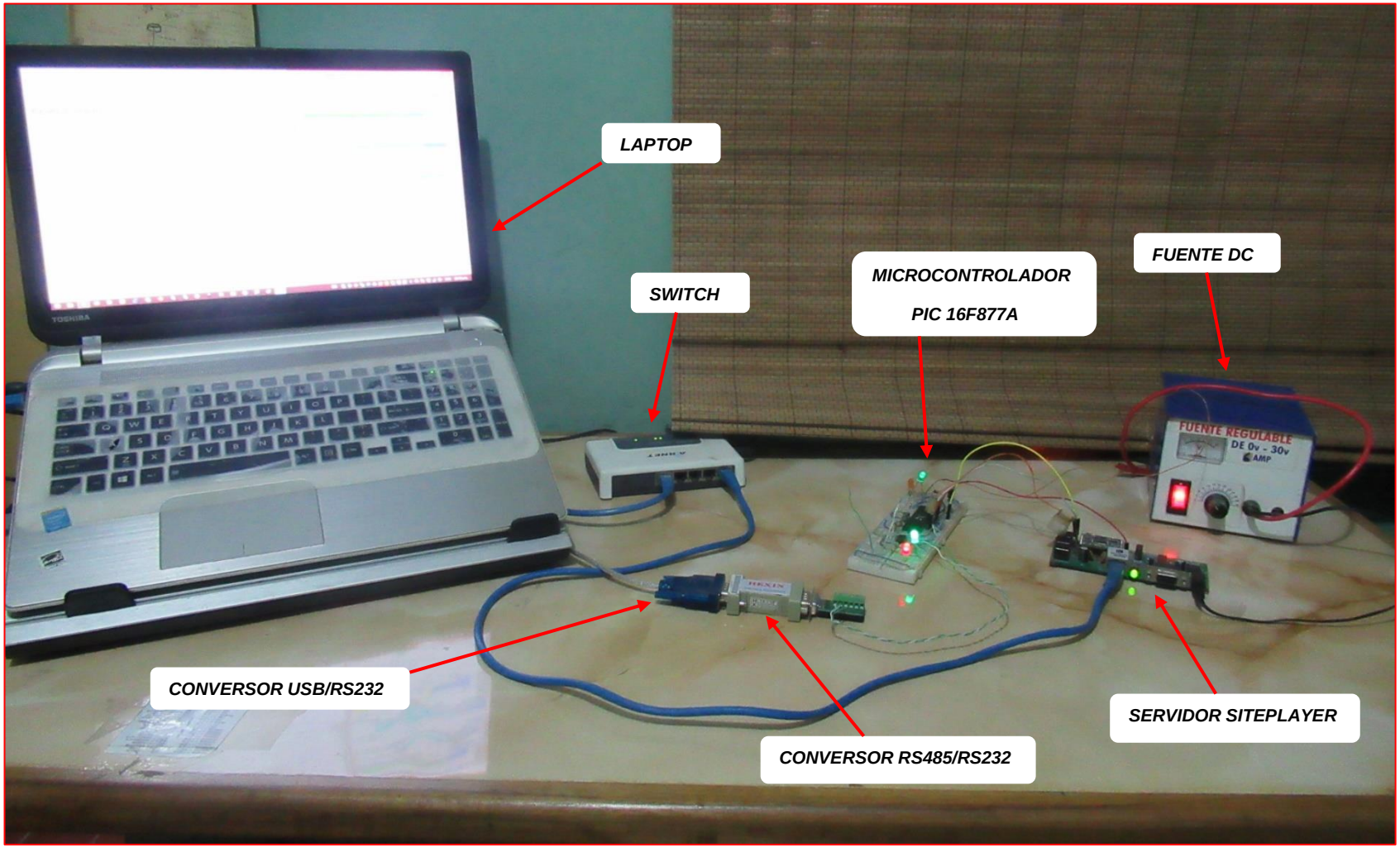


Fig. B.6 Hardware usado en el proyecto de Tesis. [***Fuente**: Propia*]

En las siguientes imágenes se muestran los dispositivos de la figura anterior, pero en más detalle.

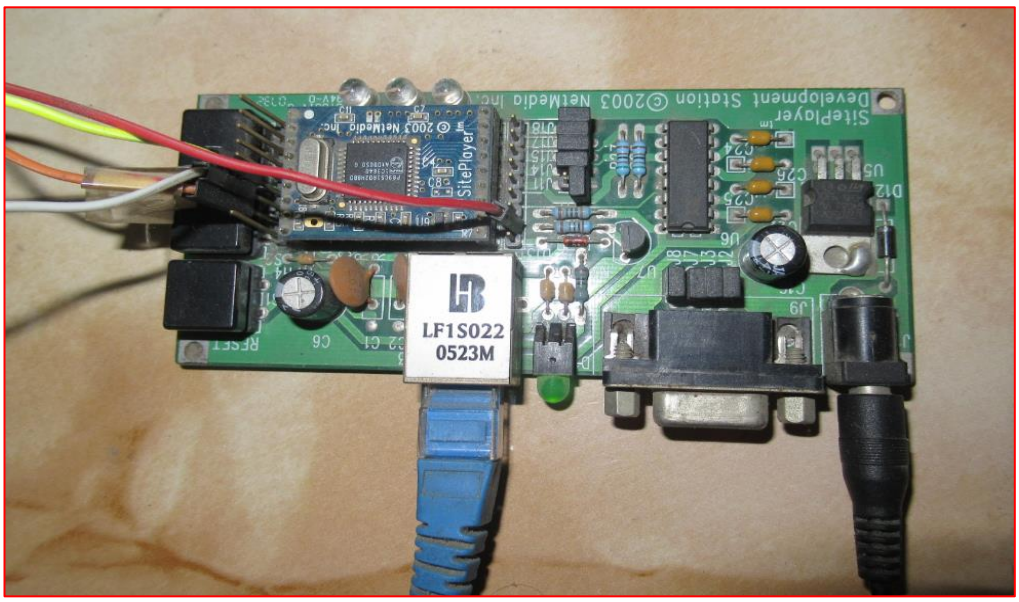


Fig. B.7 Micro Servidor Web SitePlayer. [***Fuente**: Propia*]

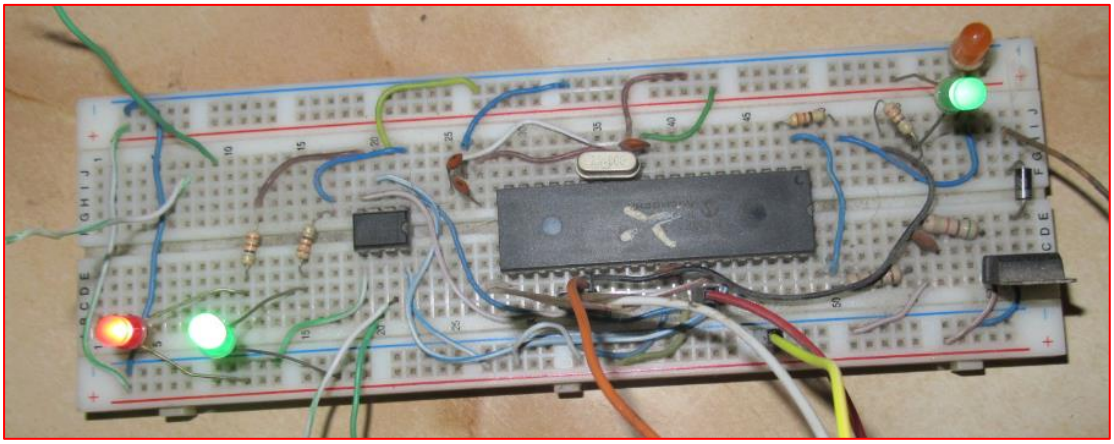


Fig. B.8 Microcontrolador PIC 16F877A. [***Fuente**: Propia*]

Fig. B.9 Conversor USB/RS232 y Conversor RS232/RS485. [***Fuente**: Propia*]