Universidad de las Ciencias Informáticas

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Título:

Driver para acceso a dispositivos Modbus a través de tecnologías Inalámbricas.

Autor:

Brian Estrada Rodríguez

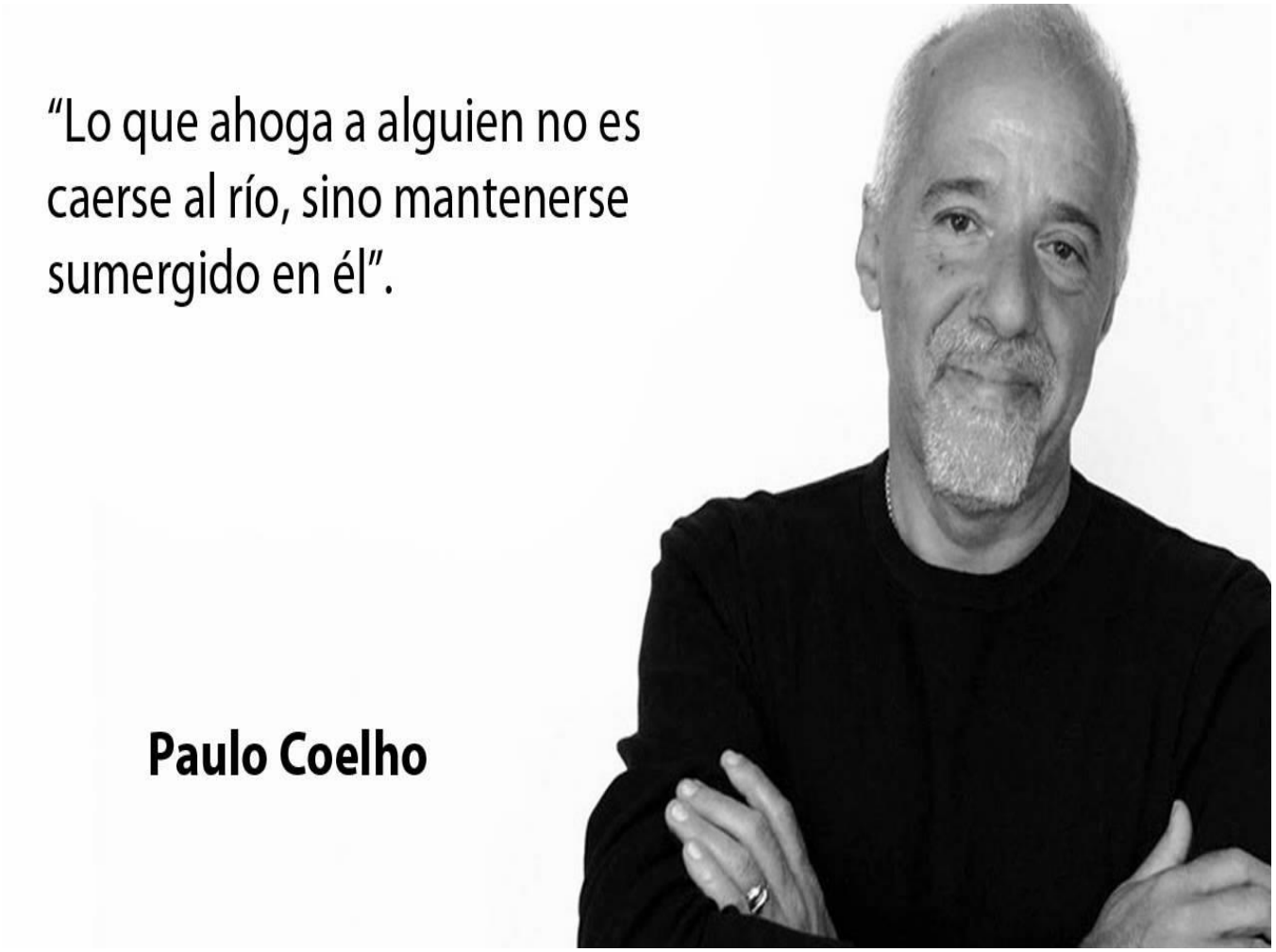
Tutores:

Ing. Julio Alberto Leyva Durán

Msc. Miguel Ángel Socorro Borges

La Habana, junio de 2018

Pensamiento



Agradecimientos

Agradezco:

A mi familia por apoyarme en todo en especial a mis padres María y Ariel dado que su apoyo me es imprescindible.

A mi tutor Julio por ayudarme en el proyecto.

A mis abuelos que me consienten y se preocuparon por mí en todo momento.

A mis compañeros por darme apoyo y no dejarme rendirme ante ninguna situación.

A mis primas las cuales supieron en todo momento mis problemas, ahogaron mis penas y calmaron mi sufrimiento.

A mi hermano por darme la oportunidad de estudiar cuando se necesitaba en ese momento trabajar.

A todos los profesores que de alguna forma influyeron en mi formación como profesional.

Dedicatoria

A mis padres María y Ariel

A mis abuelos

A mis primas y mi hermano

A mi familia

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Firma del Autor Firma del Tutor

Brian Estrada Rodríguez Ing. Julio A. Leyva Durán

_____

Firma del Tutor

Msc. Miguel Ángel Socorro Borges.

Datos del contacto

Julio Alberto Leyva Durán: Ingeniero en Ciencias Informáticas desde 2008, categoría docente Instructor, actualmente Especialista A en Ciencias Informáticas en la línea de Sistemas Embebidos del departamento de aplicaciones del centro CEDIN.

Miguel Ángel Socorro Borges: Ingeniero en Ciencias Informáticas desde 2014, categoría docente Instructor, actualmente Especialista A en Ciencias Informáticas en el proyecto SCADA del departamento de Componentes del centro CEDIN.

Resumen

La presente investigación tiene la taxonomía sobre una variante de comunicación con dispositivos que manejan el protocolo modbus, específicamente para establecer comunicación inalámbrica sobre tarjetas de adquisición de datos (*DAQT*, *Data Acquisition Target* por sus siglas en inglés) de bajo costo para la domótica; debido a que actualmente se establece dicha comunicación sobre una red cableada. La propuesta se enfoca como objetivo desarrollar un mecanismo de acceso inalámbrico en un Cliente Modbus para la recolección de datos de variables de la domótica como son: Humedad, temperatura, luminosidad, apertura de puertas, presencia, etc., de un servidor modbus en un DAQT con comunicación WIFI. Como metodología de desarrollo de software se empleó AUP (versión UCI), para realizar el modelado del sistema se utilizará la herramienta CASE Visual Paradigm Enterprise Edition en la versión 8.0, apoyándose en el Lenguaje de Unificado de Modelado (UML) en la versión 6.4. Se hace uso del lenguaje de programación C++ y el Entorno de Desarrollo Integrado QT-Creator en la versión 5.10. Finalmente se logró un Cliente Modbus que entre sus principales funcionalidades permite la comunicación mediante la red Wifi con un servidor modbus en una DAQT basada en Arduino, para la recolección de datos en la Domótica.

Palabras claves: *Cliente Modbus, Tarjeta de Adquisición de Datos, Wifi, Arduino, Domótica.*

Abstract

The present investigation has the taxonomy on a variant of communication with devices that handle the modbus protocol, specifically to establish wireless communication on data acquisition cards (DAQT, Data Acquisition Target for its acronym in English) for low cost for home automation; because that communication is currently established on a wired network. The proposal aims to develop a wireless access mechanism in a Modbus Client for data collection of home automation variables such as: Humidity, temperature, brightness, opening doors, presence, etc., of a Modbus server in a DAQT with WIFI communication. As a software development methodology, AUP (UCI version) was used. To model the system, the CASE Visual Paradigm Enterprise Edition tool was used in version 8.0, based on the Unified Modeling Language (UML) in version 6.4. It makes use of the C ++ programming language and the Integrated Development Environment QT-Creator in version 5.10. Finally, a Modbus Client was achieved which, among its main functionalities, allows the communication via the Wi-Fi network with a modbus server in a DAQT based on Arduino, for the data collection in the Domotics.

Keywords: Modbus Client, Data Acquisition Card, Wifi, Arduino, Domotics.

Índice

Contenido

Índice de figuras

Índice de tablas

Introducción

La informática con el paso del tiempo ha ocupado un espacio significativo dentro del mundo moderno al crearse sistemas para la gestión óptima de los procesos empresariales, aumentando la producción y logrando una mayor eficiencia. Dicho avance tecnológico a nivel mundial ha provocado una automatización de procesos tanto a nivel de empresa como hogar, ejemplos: los sistemas SCADA (Supervisory Control and Data Acquisition por sus siglas en inglés o en español Supervisión y Control de Adquisición de Datos) surgen debido a la necesidad de automatizar los diferentes procesos que pueden existir para procesos industriales o similares, mientras que en otros procesos más pequeños o de poca complejidad por ejemplo: la domótica o locales inteligentes, se emplean otras soluciones a la medida.

La domótica se centra en mejorar los servicios de un local específico, tales como: el ahorro energético, comunicación, seguridad, el control y automatización de los sistemas de climatización, iluminación, riego, el funcionamiento automatizado de los electrodomésticos. En el mercado existen numerosas maneras de nombrar a esta nueva forma de concebir la automatización en la vivienda en inglés tales como: *Digital Home, Smart Home, iHomes, Home Automation*.

Actualmente existen en el mercado internacional soluciones para la domótica, que abarcan un conjunto de funcionalidades para la automatización, que en su mayoría son altamente costosos debido a los términos de licencia. Dado que Cuba no puede costear dichos proyectos tiene instituciones dedicadas a la producción de software, como la Universidad de las Ciencias Informáticas (UCI). En estos momentos el CEDIN (Centro de Informática Industrial), de la facultad 4 se encuentra en el desarrollo del sistema Arex: proyecto de domótica que permite controlar y visualizar el estado de diferentes sensores y accionar sobre actuadores de un determinado proceso a través de una PC, móvil, o dispositivo de bajo costo como raspberry pi.

El sistema Arex permite la comunicación con dispositivos o tarjetas de adquisición de datos (*DAQT, data acquisition target* por sus siglas en inglés) mediante el protocolo de comunicación modbus en las variantes: RTU (Unidad Terminal Remota o *Remote Terminal Unit*, por sus siglas en inglés) y TCP/IP (Protocolo de Control de Transmisión o *Transmission Control Protocol*, por sus siglas en inglés), empleando para el intercambio de información una infraestructura de red cableada. En muchos escenarios, por la ubicación y edificación de los locales, se dificulta el entramado de cable y la longitud del mismo ha provocado caídas de voltajes y perdida de la señal en los actuadores y sensores; para mitigar dichos problemas se emplea una DAQT con conexión inalámbrica, que permite estar cerca del grupo de instrumentación(sensores y actuadores), encargada de medir las señales físicas del entorno,

evitando perdida de señales eléctricas y a la vez contribuye a la estética del cableado en el local correspondiente.
Dada esta situación polémica surge el **problema científico**: ¿Cómo vincular las tarjetas de adquisición de datos usando tecnologías inalámbricas e integrarlo al sistema Arex?

Para resolver este problema se plantea como **objetivo general**: Desarrollar un Driver para acceso a dispositivos modbus a través de tecnologías inalámbricas.

Para darle solución a dicho problema se propuso abordar como **objeto de estudio**, el protocolo de comunicación modbus sobre una red inalámbrica, y el **campo de acción** se delimita en el intercambio de mensajes modbus TCP/IP con dispositivos DAQT sobre una red WIFI.

Obteniendo como **posible resultado**: La obtención de un manejador que se integre al sistema Arex que permita establecer una comunicación mediante el protocolo modbus con los dispositivos DAQT que posean el módulo WIFI a través de una red inalámbrica.

Para dar cumplimiento al objetivo planteado se tuvieron en cuenta las siguientes **tareas de investigación**:

- Familiarización con los lenguajes de programación C++ y Qt para el desarrollo de la aplicación.
- Familiarización con el proyecto Arex.
- Estudio del Cliente Recolector de Arex.
- Investigación sobre comunicación WI-FI.
- Implementación del manejador o driver con las tecnologías seleccionadas.
- Realización de pruebas prácticas del sistema en el área de aplicación escogida.
- Documentación adecuadamente de todo el proceso.

En la presente investigación se utilizarán los siguientes métodos de investigación:

Métodos Empíricos

Observación: permitió valorar la necesidad de hacer un manejador para el intercambio de información con los dispositivos DAQT.

Métodos Teóricos

Modelación: haciendo uso del mismo se realizarán diagramas o modelos, que permitirán estructurar teóricamente el sistema, facilitando su comprensión.

Capítulo I: Fundamentación teórica

Introducción

El siguiente capítulo hace mención a teorías relacionadas con la domótica, las redes donde se emplea el Protocolo de Control de Transmisión (TCP), las generalidades del protocolo de comunicación Modbus en su variante TCP/IP, aspectos específicos del intercambio de mensajes modbus sobre TCP/IP, características del dispositivos o Tarjeta de Adquisición de Datos (DAQT) de donde se provee los datos. Además tendencias de las tecnologías inalámbricas; y se concluye con tecnologías y metodologías utilizadas para el desarrollo de la solución.

1.1 La Domótica

El término Domótica proviene de la unión de las palabras domus (que significa casa en latín) y tica (de automática, palabra en griego, 'que funciona por sí sola'). La palabra domótica definida por la Real Academia Española es el conjunto de sistemas que automatizan las diferentes instalaciones de una vivienda. (1)

Una vivienda domótica es aquélla que, a través de la tecnología, permite "una mayor calidad de vida, una reducción del trabajo doméstico, un aumento del bienestar y la seguridad de sus habitantes, y una racionalización de los distintos consumos; en ella existen agrupaciones automatizadas de equipos, normalmente asociadas por funciones, que tienen la capacidad de comunicarse interactivamente entre ellas a través de un "bus doméstico multimedia que las integra ". Para ello, la Domótica prevé la ejecución de ciertas funciones, dependiendo de la información captada por sistemas de medida y trasmitidas a través de redes de comunicación de acuerdo con ciertos estándares.

La domótica tiene como objetivo una integración de todos los controles en una unidad centralizada, para su implementación es necesario varios dispositivos que se clasifican de la siguiente manera:

Controlador: Son dispositivos que controlan el sistema dependiendo de los requerimientos recibidos o ya establecidos, estos pueden ser uno o varios.

Actuador: Este dispositivo se encarga de ejecutar la acción requerida por el controlador y así proceder a realizarla ya sea el encendido y apagado de las luces, electrodomésticos, abrir o cerrar persianas, entre otras.

Sensor: Este dispositivo está enfocado a detectar todo tipo de magnitud física la cual se desea tener un reporte de su comportamiento para poder controlarla como el agua, humo, gas, temperatura.

1.1.2 Sistema Arex

Arex es un sistema desarrollado en el CEDIN enfocado a la automatización de procesos de baja y mediana complejidad, que permite una gestión del ahorro de energía, seguridad y confort, a través de la automatización y la aplicación de modernas tecnologías de la información, a un conjunto de locales, hogar, edificio o proceso industrial (1). El sistema en general tiene como objetivos específicos:

- Configurar el proceso (comunicación, dispositivos, variables, alarmas, rutinas de control, despliegues, gráficos, almacenamiento, roles de usuarios).
- Comunicarse con los dispositivos DAQT.
- Recolectar y procesar la información recibida, en forma continua y confiable.
- Representar gráficamente las variables del proceso.
- Monitorizar variables del proceso por medio de alarmas.
- Realizar control manual y automático sobre las variables del proceso.
- Almacenar históricos de variables y alarmas.
- Visualizar históricos de variables y alarmas, y valores en tiempo real.

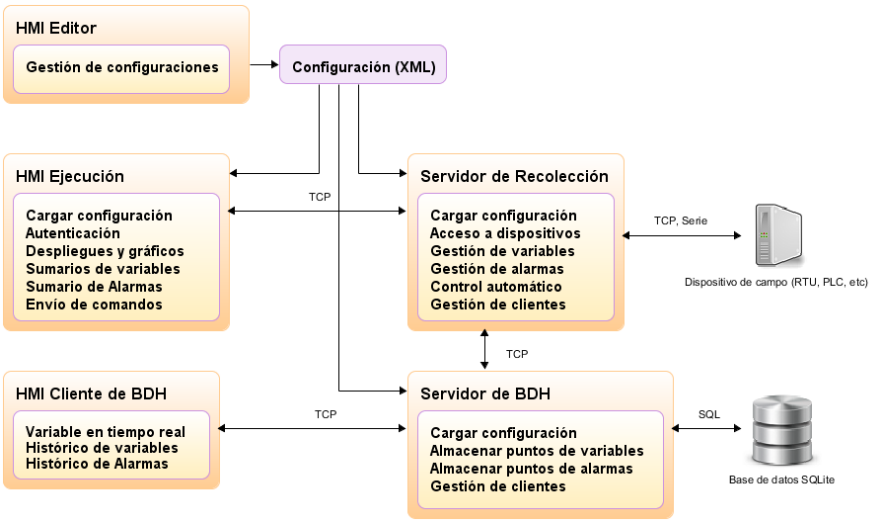


Fig. 1 Arquitectura del Sistema Arex.

HMI Editor: El HMI Editor permite crear y configurar dispositivos variables y alarmas de un proceso, y la manera en que será visualizado el mismo mediante despliegues de componentes gráficos. Se pueden configurar además aspectos imprescindibles como alarmas y rutinas de control automático.

HMI Ejecución: La vista de HMI Ejecución está implementada completamente en lenguaje QML. En la misma, por la interacción con el usuario, se generan solicitudes de: autenticar usuario, visualizar grupo de despliegues, escritura de variables, cambio de estado de alarmas, cerrar sesión de usuario, entre otras.

HMI Cliente de BDH: Este módulo permite la visualización en tiempo real de los puntos de una o más variables, datos históricos de los puntos de una variable dado un intervalo de tiempo, así como la sucesión de estados de alarmas asociadas a una variable. Permite estudiar el comportamiento del proceso monitoreado, para la toma de decisiones por parte de los operadores.

Servidor de BDH: Este módulo permite el almacenamiento de datos recolectados o generados por el Servidor de Recolección (puntos de variables y alarmas), para que puedan ser consultados de inmediato o con posterioridad. Cuando un cliente lo solicita, envía en tiempo real vectores de puntos de variables. Además, para un intervalo de tiempo definido, devuelve el conjunto de puntos generados de una variable o de alarmas asociadas a dicha variable.

Servidor de Recolección: Este módulo es el encargado de gestionar la comunicación con diferentes tipos de dispositivos DAQT, adquirir y modificar datos de dichos dispositivos, generar cambios de variables y alarmas automáticamente, y brindar datos a clientes que lo soliciten. El intercambio de información con los dispositivos DAQT se lleva a cabo mediante interfaces de red como Ethernet y Serie, y a través de protocolos estándar de comunicación y transferencia de datos.

Dispositivos de campo: El sistema Arex tiene implementados manejadores de dispositivos, en forma de librerías, que implementan el protocolo modbus en sus variantes TCP/IP y RTU. Dichos manejadores, haciendo uso de las interfaces de transporte sobre Ethernet y Serie, permiten la comunicación directa con los dispositivos DAQT.

Topología de red: La topología de red que se emplea puede ser de dos variantes: centralizada o distribuida, está en dependencia de los escenarios donde se vaya a desplegar la solución.

- **Arquitectura Centralizada:** En este tipo de arquitectura se tiene una topología de interconexión de tipo estrella. Así el sistema domótico posee un elemento de control centralizado que es el encargado de manejar las señales de los diversos dispositivos; y su vez todos los dispositivos están conectados hacia él.
- **Arquitectura Distribuida:** Los factores que influyen para la utilización de este tipo de arquitectura son los medio de transmisión, la velocidad en las comunicaciones y el tipo de protocolo.

Un ejemplo de una topología centralizada en una red privada de área local, se puede observar en la figura 2.

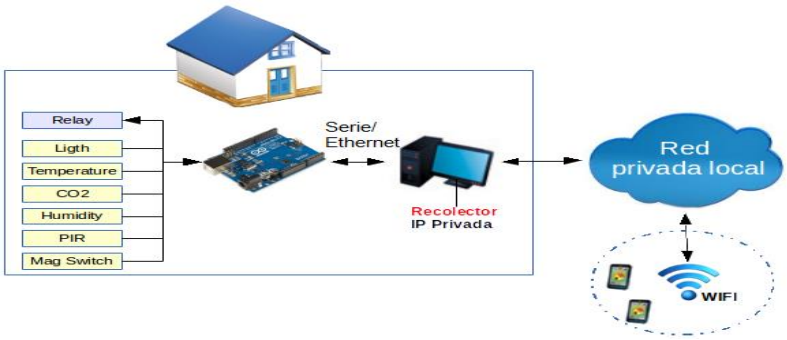


Fig. 2 Despliegue de Arex en una red privada de área local.

1.1.3 Protocolo de Control de Transmisión (TCP)

EL Protocolo de Control de Transmisión (*Transmission Control Protocol*, por sus siglas en inglés) es uno de los protocolos fundamentales en Internet. Fue creado entre los años 1973 -1974 por Vint Cerf y Robert Kahn.

Muchos programas dentro de una red de datos compuesta por ordenadores pueden usar TCP para crear conexiones entre ellos a través de las cuales puede enviarse un flujo de datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto.

TCP está orientado a la conexión en el sentido de que, antes de transmitir datos, los participantes deben establecer la conexión. Todos los datos viajan en segmentos TCP, en donde cada viaje se realiza a través de Internet en un datagrama IP. Los dos protocolos más importantes son TCP e IP (Protocolo de Internet o Internet Protocol, en inglés), de ahí que se denomine también como Conjunto de Protocolos TCP/IP. A continuación se ilustra la pila TCP/IP con sus respectivos niveles:

Fig. 3 Organización de la pila TCP/IP

TCP/IP es la plataforma que permite la comunicación entre diferentes sistemas operativos en diferentes computadoras, ya sea sobre redes de área local (LAN) o redes de área extensa (2).

- **Capa de Aplicación:** La capa de aplicación maneja protocolos de alto nivel, aspectos de representación, codificación y control de diálogo. El modelo TCP/IP combina todos los aspectos relacionados con las aplicaciones en una sola capa y garantiza que estos datos estén correctamente empaquetados para la siguiente capa. La capa de aplicación incluye programas de aplicación que utilizan la red. Es la encargada de pasar la solicitud a la Capa de Transporte cuando un usuario inicia una transferencia de datos. Como ejemplos de estas aplicaciones se tienen al correo electrónico, los programas de transferencia de archivos o a cualquier aplicación que tenga entre sus funcionalidades la de transmitir datos por la red como es el caso de la aplicación que se pretende desarrollar para dar solución a nuestra problemática.
- **Capa de Transporte:** La principal tarea de la capa de transporte es proporcionar la comunicación entre una aplicación y otra. Este tipo de comunicación se conoce frecuentemente como comunicación punto a punto. La capa de transporte regula el flujo de información y puede también proporcionar un transporte confiable, asegurando que los datos lleguen sin errores y en secuencia. Para hacer esto, el software de protocolo de transporte tiene el lado de recepción enviando acuses de recibo de retorno y la parte de envío retransmitiendo los paquetes perdidos.

Debido a que la solución a desarrollar se encuentra enmarcada esencialmente en las capas de aplicación y hará uso de los protocolos de comunicación TCP de la capa de transporte del modelo TCP/IP, en este trabajo, se decidió indagar solamente en estos dos niveles.

1.1.4 Protocolo Modbus

Modbus es un protocolo de comunicaciones situado en el nivel 7 del Modelo OSI, basado en la arquitectura maestro/esclavo (RTU) o cliente/servidor (TCP/IP), diseñado en 1979 por Modicon para su gama de controladores lógicos programables (PLCs); convertido en un protocolo de comunicaciones estándar en la industria; es además, el que goza de mayor disponibilidad para la conexión de dispositivos electrónicos industriales.

Las principales razones por las cuales el uso de Modbus en el entorno industrial se ha impuesto a otros protocolos de comunicaciones son:

- ✓ Se diseñó teniendo en cuenta su uso para aplicaciones industriales.
- ✓ Es público y gratuito.
- ✓ Es fácil de implementar y requiere poco desarrollo.
- ✓ Maneja bloques de datos sin suponer restricciones.

Modbus es un protocolo de solicitud-respuesta implementado usando una relación maestro-esclavo. En una relación maestro-esclavo, la comunicación siempre se produce en pares, un dispositivo debe iniciar una solicitud y luego esperar una respuesta y el dispositivo de inicio (el maestro) es responsable de iniciar cada interacción. Por lo general, el maestro es una interfaz humano-máquina (HMI) o sistema SCADA y el esclavo es un sensor, controlador lógico programable (PLC) o controlador de automatización programable (PAC) o una DAQT.

El contenido de estas solicitudes y respuestas, y las capas de la red a través de las cuales se envían estos mensajes, son definidos por las diferentes capas del protocolo. En la implementación inicial, modbus era un solo protocolo construido en base a serial, por lo que no podía ser dividida en múltiples capas. Con el tiempo, diferentes unidades de datos de aplicación fueron introducidas ya sea para cambiar el formato del paquete utilizado a través de serial o para permitir el uso de redes TCP/IP y UDP. Esto llevó a una separación del protocolo principal, el cual define la unidad de datos de protocolo (PDU) y la capa de red, que define la unidad de datos de aplicación (ADU). (3)

Modelo cliente/servidor en Modbus TCP: El protocolo modbus en su servicio de mensajería provee comunicación de tipo cliente/servidor entre dispositivos conectados a la red Ethernet mediante TCP/IP. Este modelo básicamente está basado en 4 tipos de mensajes. (4)

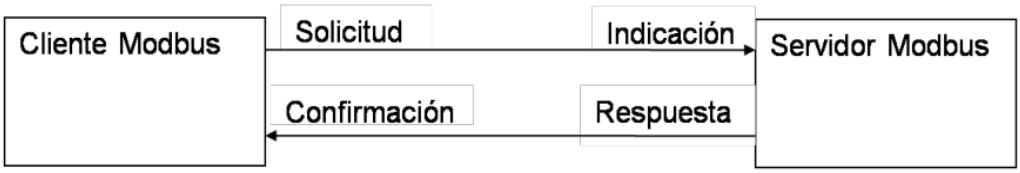


Fig. 4 Modelo cliente/servidor en modbus.

1. **Solicitud Modbus:** es el mensaje enviado por la red por parte del cliente para inicializar la transacción.
2. **Confirmación Modbus**: es el mensaje de solicitud recibido del lado del servidor.
3. **Indicación Modbus**: es el mensaje de respuesta a la solicitud del cliente, enviado por el servidor.
4. **Respuesta Modbus**: es el mensaje de respuesta recibido del lado del cliente.

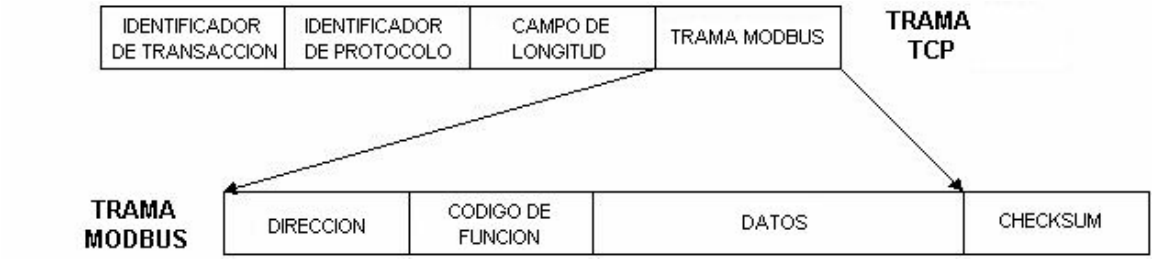


Fig. 5 Encapsulación de las tramas sobre una red TCP/IP.

Tabla. 1 Formato que compone la cabecera MBAP.

Campo	Tamaño	Descripción
Identificador de transacción	2 bytes	Identificador de una trama, ya sea de solicitud o respuesta.
Identificador del protocolo	2 bytes	Es igual a 0 en el caso del protocolo Modbus
Longitud	2 bytes	Numero de bytes que siguen a continuación
Identificador de unidad	1 bytes	Identificador del esclavo remoto a ser encuestado

En la tabla 2 se muestran todas las funciones que define el protocolo Modbus, donde el término "**xxxx**" no es más que la referencia del tipo de registro sobre el cual se efectuará la operación.

Tabla. 2 Funciones de modbus

Código Hex	Descripción de la función asociada
0x01	Leer múltiples bobinas (0**xxxx**).
0x02	Leer múltiples entradas discretas (1**xxxx**).
0x03	Leer múltiples registros de entrada/salida (4**xxxx**).
0x04	Leer múltiples registros de entrada (3**xxxx**).
0x05	Escribir una bobina (0xxxx).
0x06	Escribir un registro de entrada / salida (4**xxxx**).
0x07	Lectura del estado de error
0x08	Acceso a los contadores de diagnóstico

Código Hex	Descripción de la función asociada
0x09	Modos de operación y carga / descarga remota
0x0A	Solicitud de reporte de operación
0x0B	Lectura del contador de eventos
0x0C	Lectura de los eventos de conexión
0x0D	Modos de operación y carga / descarga remota
0x0E	Solicitud de reporte de operación
0x0F	Escritura de múltiples bobinas
0x10	Escritura de múltiples registros de entrada / salida.
0x11	Leer identificación del esclavo
0x12	Modos de operación y carga / descarga remota
0x13	Reseteo de esclavo después de error no resuelto
0x14	Lectura de referencia general
0x15	Escritura de referencia general
0x16	Escritura con máscara de Registros (4**xxxx**).
0x17	Lectura / escritura de Registros (4**xxxx**).
0x18	Lectura de la cola FIFO

Variante de Modbus sobre Ethernet: Modbus sobre Ethernet es un tipo de protocolo Modbus diseñado para controlar y supervisar equipos a través de TCP/IP. Le permite proporcionar acceso compartido a un dispositivo modbus local a través de la red, para que otros usuarios de la red puedan alcanzar su contenido y funcionalidad como si estuviera conectado directamente a sus máquinas. De forma similar, también puede acceder a un Modbus remoto a través de Ethernet, independientemente de su proximidad física al dispositivo. La cantidad de dispositivos con los que puede trabajar a través de la red puede ser de hasta 500, y es posible acceder a todos ellos simultáneamente. (5)

Variante de Modbus sobre redes inalámbricas: Una red Modbus sobre redes inalámbricas se puede configurar sin mucha dificultad, sin embargo en el enlace inalámbrico no reemplazará por completo todo el medio de transmisión alámbrica, pues llegará un instante en que tenga que volver a transmitir por cable hacia un equipo, teniendo un gran campo de aplicación por las ventajas que ofrece como movilidad, reducción considerable del cableado, costos y fácil instalación. (6)

1.1.5 Tecnologías inalámbricas

La tecnología inalámbrica tiene como medio de transmisión el aire. Una red inalámbrica cubre un entorno geográfico limitado, con una velocidad de transferencia de datos relativamente alta, que utiliza ondas electromagnéticas como medio de transmisión de la información que viaja a través del canal inalámbrico enlazando los diferentes equipos o terminales móviles asociados a la red. Muchas tecnologías inalámbricas se emplean en l transmisión de datos, tales como: *ZigBee*, *Bluetooth* o *Wi-Fi*. Cada una de ellas presenta una serie de ventajas e inconvenientes, la elección tecnológica deberá depender de los requisitos de la aplicación en concreto, es decir, se debe hallar una relación de compromiso entre el precio, el consumo de energía y el ancho de banda que es capaz de brindar. A continuación se hace describe las tecnologías ZigBee, Bluetooth o Wi-Fi.

- **Bluetooth:** Es un estándar de comunicaciones inalámbricas basado en radiofrecuencia, de bajo coste y bajo consumo energético. Permite la comunicación entre terminales móviles inteligentes tales como: un teléfono móvil, conexionado de periféricos o dispositivos de audio. Emplea la frecuencia 2,4 GHz y su capacidad máxima de transmisión de hasta 3 Mbps, alcanzando una distancia hasta 100 metros en función de la potencia de emisión que posea el transmisor Bluetooth.
- **ZigBee:** Sistemas de control inalámbrico denominados *ZigBee Alliance*. Se basa en el estándar 802.15.4 definido por el IEEE, posee baja capacidad de transmisión, hasta 250 Kbps, limitado en sistemas microcontroladores de 8 bits. Su principal ventaja es su bajo consumo energético y permite desarrollar sistemas de bajo costo.
- **Wi-Fi:** Es estándar internacional que implementa el nivel físico y el de enlace, sobre un canal inalámbrico. Es un estándar que desde 1997 ha sufrido una constante evolución, encontrando varias versiones, como se muestra en la Tabla 2: Evolución de las tecnologías inalámbricas.

Tabla. 3 Evolución de las tecnologías inalámbricas

Estándar	Velocidad (Teórica)	Velocidad (Práctica)	Frecuencia	Ancho de Banda	Alcance (m)	Año
802.11	2Mbit/s	1Mbit/s	2,4GHz	22MHz	330	1997
802.11ª	54Mbit/s	22Mbit/s	5,4GHz	20MHz	390	1999
802.11b	11Mbit/s	6Mbit/s	2,4GHz	22MHz	460	1999

802.11g	54Mbit/s	22Mbit/s	2,4GHz	20MHz	460	2003
802.11n	600Mbit/s	100Mbit/s	2,4GHzy5,4 GHz	20/40MHz	820	2009
802.11ac	6.93Gbit/s	100Mbit/s	5,4GHz	80hasta 160MHz		2013
802.11ad	7.13Gbit/s	Hasta6Gbit/s	60GHz	2MHz	300	2012
802.11ah			0,9GHz		1000	2016

1.1.6 Dispositivo de adquisición de Datos

Para países subdesarrollados, una solución económica es el desarrollo de dispositivos de adquisición de datos en hardware libre, de bajas prestaciones y bajo coste. Como ejemplo, se incluyen en la

Tabla. 4 Relación de precio de una DAQ basada en Arduino.

Dispositivo Arduino	Interfaz RS232	Interfaz Ethernet	Precio Total
Arduino UNO(10 USD)	6 USD	29 USD	45 USD
Arduino Mega(32 USD)	6 USD	29 USD	67 USD
Arduino UNO(10 USD)	6 USD	-	16 USD
Arduino Mega(32 USD)	-	29 USD	60 USD

propuesta del sistema Arex, prototipos sobre placas Arduino por un costo aproximado de 50.00 USD, y admitiendo una cantidad apreciable de sensores y actuadores.

Característica de la DAQT: La principal característica es que está desarrollada mediante tecnología de hardware libre, se emplea la placa Arduino como base de hardware, la parte lógica corresponde a la implementación de un firmware que gestiona la comunicación mediante el protocolo modbus TCP/IP o RTU. A continuación se ilustra el diseño lógico de la DAQT:

- **Configuración:** Establece la configuración de la tarjeta de forma dinámica.
- **Comunicación:** Establece la comunicación con algún "Cliente Recolector de Datos" por medio de un protocolo industrial.

- **Adquisición:** Recolecta los datos de los manejadores de cada sensor/actuador y los brinda a la capa de comunicación.
- **Manejadores:** Filtra las señales eléctricas de los sensores/actuadores y se las provee a la capa de adquisición.
- **Sensores/Actuadores:** Interfaz de sensores/actuadores que interactúan con la entrada/salida de propósito general o GPIO (*Genera Purpose Input/Output*, por sus sigla en inglés).

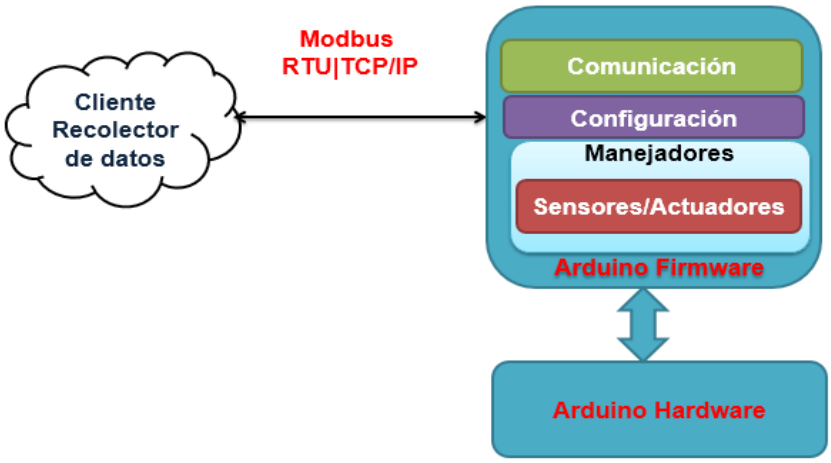


Fig. 6 Diseño lógico del dispositivo DAQT

Modelo de Comunicación inalámbrica en la DAQT: La DAQT tiene implementados manejadores de que provee el mecanismo de comunicación mediante el protocolo modbus en las variante TCP/IP, incluye acceso a la comunicación WIFI a través de una capa inalámbrica. Dicho manejador, hace uso de las interfaces de transporte sobre Ethernet, permitiendo la comunicación directa con los clientes recolectores de datos.

El modelo de comunicación que se emplea actualmente es mediante una red WIFI, para ellos se implementa un DAQT basada en Arduino UNO, con el módulo WIFI ESP8266 con las características principales (7)bajo consumo de energía, contiene una pila TCP/IP con soporte al estándar 802.11 b/g/n y el transceptor trabaja en la frecuencia 2.4 GHz. A continuación se muestra el diagrama esquemático de dicha tarjeta y la conexión con el módulo WIFI:

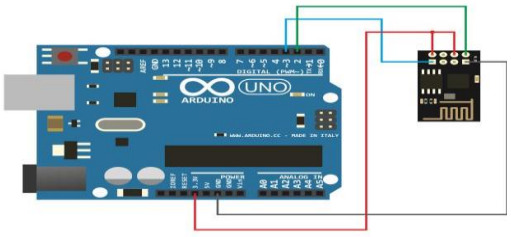


Fig. 7 DAQT basada en Arduino UNO con módulo WIFI

En la DAQT se ejecuta un firmware que gestiona la comunicación inalámbrica y el proceso de mapear una señal que proviene de los sensores y actuadores a registros del protocolo modbus correspondiente. A continuación se ilustra una configuración de sensores y actuadores a los pines de entrada y salida del arduino UNO, y el registro de modbus asociado a cada sensor o actuador.

Tabla. 5 Ejemplo de Registros Modbus en la DAQT.

PIN Arduino UNO	Sensor/Actuador	Registro Modbus
A0(entrada analógica)	LDR (Sensor lumínico).	Input register
A1(entrada analógica)	LM35 (Sensor temperatura).	Input register
A2(entrada analógica)	HIH4000(Sensor humedad)	Input register
A3(entrada analógica)	MQ(Sensor Humo)	Input register
D5(Salida Analógica)	Brillo lámpara (Actuador)	Holding register
D6(Salida Digital)	Relé lámparas (Actuador)	Coil
D7(entrada digital)	Magnético Puerta (Sensor magnético)	Input
D8(entrada digital)	PIR (Sensor de Presencia)	Input

Proceso de comunicación inalámbrica con la DAQT: El proceso comunicación se inicializa cuando se energiza la placa Arduino cargando su bootloader o programa inicial que da paso al proceso de cargar la información de la memoria EEPROM los parámetros de configuración de: sensores/actuadores, los registros de modbus y del componente WIFI ESP8266. Si existe una nueva configuración se procede a realizar la nueva configuración cambiando los valores de los sensores y actuadores en correspondencia de los pines del Arduino y su asociación a los registros de modbus, además de los parámetros del módulo ESP8266 mediante comandos AT que prepara el módulo para establecer la comunicación inalámbrica. Finalmente en un lazo cerrado se procede a ejecutar el proceso de recolección de datos y proveerlos hacia un Cliente Recolector de Datos externo mediante la comunicación WIFI.

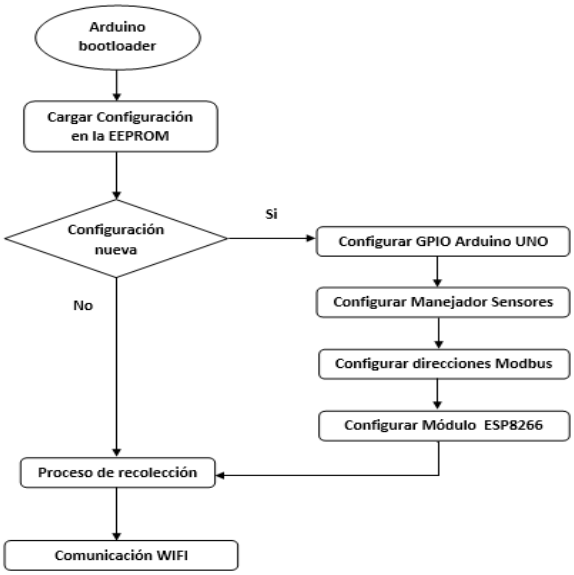


Fig. 8 Proceso de configuración WIFI en la DAQT

1.3 Metodologías y Herramientas

1.3.1 Metodología de desarrollo de software

Las metodologías para el desarrollo de software es un modo sistemático para realizar, gestionar y administrar un proyecto. Seleccionar una buena metodología será transcendental para el éxito de un producto. El papel principal de una metodología es guiar y organizar actividades que conlleven a las metas trazadas.

Para elegir una metodología de desarrollo de software se debe tener en cuenta dos factores fundamentales: el tipo de proyecto que se desea desarrollar y el tiempo que se dispone para desarrollar el mismo. En la actualidad no se puede afirmar que existe una metodología que funcione de manera universal, son más bien concebidas como marcos metodológicos que deben ajustarse a cada organización y tipo de proyecto a desarrollar, por lo que existen dos grandes enfoques de metodologías: las tradicionales y las ágiles. En la siguiente figura se presenta una comparación entre los dos tipos de metodologías, como se muestra en la figura.

Tabla. 6 Comparativa entre las metodologías ágiles y las tradicionales

Metodología ágiles	Metodologías tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo).	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas y normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (menos de 10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y distribuidos.
Pocos artefactos.	Muchos artefactos.
Pocos roles.	Muchos roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Variación de AUP para la UCI (AUP-UCI)

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto exigiéndose así que el proceso sea configurable; se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI debido a que:

- ✓ Esta metodología consiste en centrar la atención en actividades que son esenciales para el desarrollo, no en todas las que forman parte del proyecto. Por lo que el equipo de trabajo no va a leer detalladamente el proceso de documentación, ya que se sabe en lo que se está trabajando.
- ✓ Con el uso de esta metodología se puede utilizar cualquier conjunto de herramientas, pero lo aconsejable son las simples, por ser fáciles de manejar y entender.
- ✓ Es una metodología de fácil adaptación, siempre satisfaciendo las necesidades propias de sus usuarios, por lo que no es necesario comprar una herramienta especial o tomar un curso para poder adaptar un proyecto utilizando AUP-UCI. (8)

1.3.2 Herramientas y Tecnologías

El proceso de desarrollo de software se sustenta en el uso de diferentes herramientas y tecnologías, las cuales, unidas a la metodología seleccionada, conforman el ambiente de desarrollo de un sistema. Por

este motivo se decide estudiar tecnologías y herramientas actuales para seleccionar aquellas que apoyarán el ciclo de vida del desarrollo.

1.3.3 Lenguajes de modelación

Un lenguaje para el modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar un diseño de software orientado a objetos. (9)

UML

El Lenguaje Unificado de Modelado **(UML)** es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, medios y dominios de aplicación. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. (10)

UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código, así como generadores de informes. La especificación de UML no define un proceso estándar, pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. (11)

1.3.4 Herramientas para el diseño

Herramienta CASE (Ingeniería del software asistida por computadoras): Son diversas aplicaciones informáticas o programas informáticos destinadas a aumentar la productividad en el desarrollo de software reduciendo su costo en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. (12)

Visual Paradigm 8.0: *Visual Paradigm para UML es una herramienta CASE que soporta el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. (13)*

Las ventajas que proporciona Visual Paradigm son:

- ✓ Dibujo: facilita el modelado de UML, ya que proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, debido a que la misma se ajusta al estándar soportado por la herramienta.
- ✓ Corrección sintáctica: controla que el modelado con UML sea correcto.

- ✓ Coherencia entre diagramas: al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades.
- ✓ Integración con otras aplicaciones: permite integrarse con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad.
- ✓ Trabajo multiusuario: permite el trabajo en grupo, proporcionando herramientas de compartición de trabajo.
- ✓ Reutilización: facilita la reutilización, ya que dispone de una herramienta centralizada donde se encuentran los modelos utilizados para otros proyectos.
- ✓ Generación de código: permite generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software.
- ✓ Generación de informes: permite generar diversos informes a partir de la información introducida en la herramienta. (14)

1.3.5 Lenguajes de programación

Es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado de un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación. (15)

Lenguaje C++

Bjarnes Stroutstrup diseñó y desarrolló C++ en 1983 buscando un lenguaje con las opciones de programación orientada a objetos. En el año 1995, se incluyeron algunas bibliotecas de funciones al lenguaje C, y con base en ellas, se pudo en 1998 definir el estándar de C++. (16)

C++ es un súper conjunto de C, cualquier compilador de C++ debe ser capaz de compilar un programa en C. De hecho, la mayoría admite tanto código en C como en C++ en un archivo. Por esto, la mayoría de desarrolladores compilan con C++ su código escrito en C, incluso hay quienes, siendo código en C ponen la extensión CPP (extensión de los archivos de código C++) y lo compilan con C++. Algunas personas podrían pensar que entonces C++ desplazó a C, y en algunos aspectos podría ser cierto, pero también es cierto que algunas soluciones a problemas requieren de la estructura simple de C más que la de C++ (16).

1.3.6 Entornos integrados de desarrollo

Un entorno de desarrollo integrado o *Integrated Development Environment o* IDE por sus siglas en inglés, es un entorno de programación que ha sido empaquetado como un programa de aplicación, es

decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. (17)

Un IDE es un tipo de software compuesto por un conjunto de herramientas de programación. En concreto se compone de:

- ✓ Editor de código de programación.
- ✓ Compilador.
- ✓ Intérprete.
- ✓ Depurador.
- ✓ Constructor de interfaz gráfico.

Para el desarrollo de la solución se necesitan herramientas como los entornos de desarrollo integrados. Estos IDE deben tener un nivel de desarrollo y estabilidad alcanzada, documentación existente, flexibilidad y personalización, y como elemento esencial que sean aplicaciones desarrolladas bajo los términos de las licencias que rigen el mundo del software libre.

QT-Creator

Qt es un framework escrito en C++ creado por la compañía Trolltech bajo licencias, pública y propietaria, que permite crear aplicaciones con interfaces gráficas, además de poseer un módulo para la comunicación vía TCP/IP (QtNetwork), posee un mecanismo para el trabajo con objetos, permitiendo adicionar y eliminar propiedades de los mismos en tiempo de ejecución. Qt es totalmente orientado a objetos, fácil de usar, extensible y multiplataforma (soportada en los sistemas operativos Windows, Unix y derivados).

El módulo QtNetwork provee un conjunto de clases que facilitan la programación sobre la red, dentro de este conjunto de clases se encuentran a un alto nivel, pues implementan aplicaciones a nivel de protocolos específicos de red como son el caso de QHttp y QFtp, mientras se encuentran otras de más bajo nivel como QTcpSocket y QUdpSocket. Además contiene otros componentes como QModbus para en trabajo con el protocolo modbus en las variante TCP/IP o RTU. (18)

1.4 Criterios de la selección de las tecnologías y herramientas

El modelado del software se realizará usando los lenguajes UML en su versión 6.4 para especificar, construir, definir de forma gráfica y documentar el diseño de la solución, proporcionándole un soporte concreto a la metodología seleccionada. La herramienta CASE a utilizar para el proceso de modelado será Visual Paradigm Enterprise Edition en su versión 8.0 ya que esta constituye un software de alta eficiencia que permite realizar ingeniería tanto inversa como directa, es de software libre y permite la generación de documentación de forma automática en diferentes formatos. El lenguaje de programación será C++ porque combina la programación orientada a objetos de los lenguajes de alto nivel y es compatible con el framework QT. Como entorno de desarrollo se empleara el QT Creator, es un framework multiplataforma y posee componentes para la comunicación TCP/IP mediante la bibliotecas

QtNetwork y para el protocolo modbus la biblioteca QModbus. Entre las tecnologías inalámbricas descritas en este capítulo se ha decidido elegir el estándar 802.11, conocido comercialmente como WIFI. Se ha decidido utilizar esta tecnología porque es compatible con el módulo en la DAQT: ESP8266.

Capítulo II: Propuesta de solución

Introducción

Este capítulo refleja la solución que se propone para resolver el problema científico identificado. Se realizará la captura de requisitos funcionales y no funcionales y las historias de usuarios correspondientes a ellos. Se describirá la arquitectura, los patrones de diseño y los diagramas de clases que se utilizarán, con el objetivo de proveer un punto de partida para la implementación de la solución.

2.1 Propuesta del sistema

La solución que se propone abarca la creación de un Cliente Modbus para recolectar los valores de las variables correspondiente a cada sensor o actuador, asociados a los registros de modbus en una DAQT mediante conexión WIFI; además se obtendrá la información de la DAQT como: tipo de hardware (Arduino), sensores y actuadores configurado (*input, output, reserved, no used*) en el GPIO del Arduino, registros de modbus(*coil status, input status, holding register, input register*), parámetros del módulo de comunicación ESP8266 tales como: *versión, intensidad de la señal WIFI, modo de transmisión, cantidad máxima de conexiones, local IP, local puerto*. Además de obtener la configuración, puede ser modificada y ser enviada directamente hacia el dispositivo.

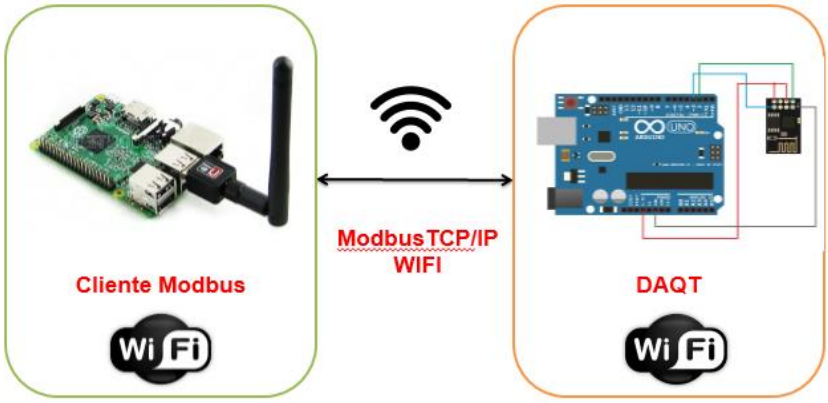


Fig. 9 Diagrama de flujo de datos

2.2 Requerimientos del sistema

Los requerimientos no funcionales especifican las propiedades o cualidades que debe tener la solución a desarrollar. Representan las características que hacen al producto atractivo, usable, rápido o confiable. Estos requisitos pueden marcar la diferencia entre un producto bien aceptado y otro con poca aceptación. (19)

- ✓ **Requerimientos de usabilidad:** podrá ser usado con facilidad por parte del usuario.

- ✓ **Requerimientos de restricciones del diseño e implementación**: El sistema debe ser desarrollado en el lenguaje de programación C++, las interfaces gráficas de usuarios deben ser desarrolladas utilizando el marco de trabajo QT. El sistema debe cumplir con las especificaciones técnicas del hardware de la tarjeta de adquisición de datos.
- ✓ **Requerimientos de Apariencia o interfaz externa**: La interfaz debe ser de fácil comprensión en su funcionamiento permitiendo la utilización del sistema sin mucho entrenamiento, es decir, de fácil uso y con rápida respuesta del sistema. Interfaces uniformes con los mismos colores y diseños. La interfaz debe tener mensajes sin ambigüedades.
- ✓ **Requerimientos de licencias y patentes**: Se deben utilizar herramientas libres.
- ✓ **Requerimientos de Portabilidad**: El sistema debe funcionar en sistemas de la familia GNU/Linux y Windows.
- ✓ **Requerimientos de ayuda y documentación:** El proyecto debe contar con una documentación según la metodología establecida.
- ✓ **Requerimientos de hardware**: Procesador: *1 GHz*. Memoria RAM: *1 GB o superior*. Disco Duro: *8 GB*. Adaptor WIFI: *Wireless Standard: 802.11n IEEE*, *Frequency Range: 2.4 GHz*.

2.3 Historias de usuario

Entre los artefactos que define la metodología seleccionada se encuentran las historias de usuario (HU), las cuales representan una breve descripción del comportamiento del sistema, emplean terminología del cliente sin lenguaje técnico, se realiza una por cada característica principal del sistema, se emplean para hacer estimaciones de tiempo, reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación. Cada HU contiene la estimación de esfuerzo según el orden a realizar. Cada estimación incluye el esfuerzo asociado a la implementación de la HU, la misma se expresa utilizando como medida una puntuación, que es directamente proporcional al esfuerzo. Un punto se considera como una semana ideal de trabajo, donde se trabaje el tiempo planeado sin ningún tipo de interrupción.

Tabla. 7 Historia de usuario 1

Historia de usuario	
Número:1	**Nombre del requisito:** Escanear los dispositivos WIFI.
Programador: Brian Estrada Rodríguez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 1

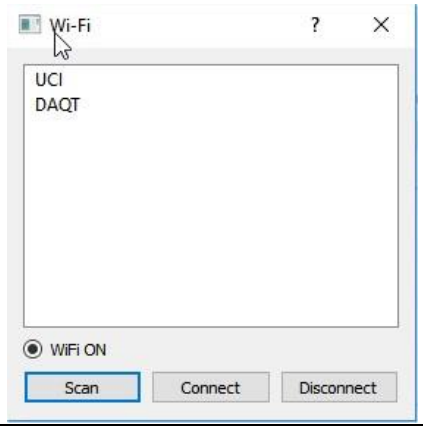
Riesgo en desarrollo: Alto	**Tiempo real:** 1
Observaciones: -	
Descripción: Esta funcionalidad permite escanear los dispositivos al alcance de la WIFI. Mostrando un listado con el nombre de cada señal WIFI y permite seleccionarlas para una posterior conexión.	
Prototipo elemental de interfaz de usuario: Sí. 	

Tabla. 8 Historia de usuario 2

Historia de usuario	
Número:2	**Nombre del requisito:** Conectar con la Tarjeta de Adquisición de Datos (DAQT).
Programador: Brian Estrada Rodríguez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 1
Riesgo en desarrollo: Alto	**Tiempo real:** 1

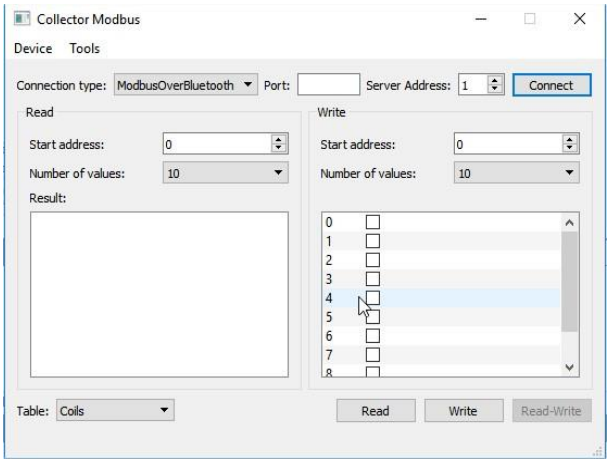
Observaciones: Debe de estar disponible la WIFI en la DAQT.
Descripción: Esta funcionalidad permite conectarse a la WIFI de la Tarjeta de Adquisición de Datos y habilita los campos para una conexión modbus, como también la lectura y escritura de los registros (*Coils, Input, Input Register, Holding Register*) del dispositivo.
Prototipo elemental de interfaz de usuario: Sí. 

Tabla. 9 Historia de usuario 3

Historia de usuario	
Número:3	**Nombre del requisito:** Desconectar la Tarjeta de Adquisición de Datos.
Programador: Brian Estrada Rodríguez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 3
Riesgo en desarrollo: Alto	**Tiempo real:** 3

Observaciones: Se debe haber establecido la comunicación WIFI con la DAQT.
Descripción: Esta funcionalidad permite desconectarse de la WIFI de la Tarjeta de Adquisición de Datos y desactiva los campos de conexión, lectura y escritura del dispositivo.
Prototipo elemental de interfaz de usuario: N/A

Tabla. 10 Historia de usuario 4

Historia de usuario	
Número:4	**Nombre del requisito:** Establecer comunicación con el Servidor Modbus en la DAQT.
Programador: Brian Estrada Rodríguez	**Iteración asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 3
Riesgo en desarrollo: Alto	**Tiempo real:** 3
Observaciones: Se debe haber establecido la comunicación WIFI con la DAQT.	
Descripción: Esta funcionalidad permite establecer la comunicación con el servidor modbus configurado en la DAQT, y otorga los permisos de lectura y escritura sobre los registros del arduino.	
Prototipo elemental de interfaz de usuario: N/A	

Tabla. 11 Historia de usuario 5

Historia de usuario	
Número:5	**Nombre del requisito:** Visualizar datos desde el Servidor Modbus en la DAQT.

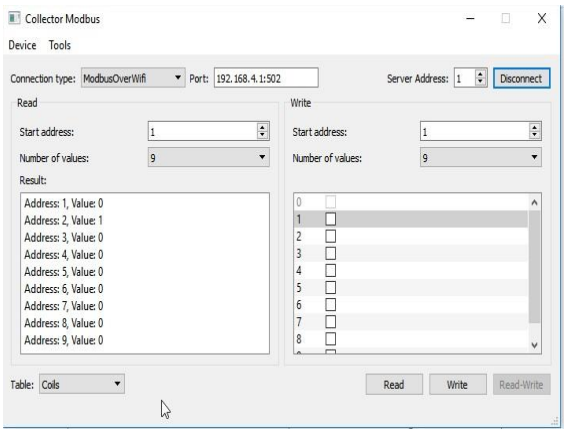
Programador: Brian Estrada Rodríguez	**Iteración asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 2
Riesgo en desarrollo: Alto	**Tiempo real:** 2
Observaciones: Se debe haber establecido la comunicación con el Servidor Modbus en la DAQT.	
Descripción: Esta funcionalidad permite visualizar los valores de los registros de Modbus (*Coils, Input, Input Register, Holding Register*) en el Servidor de la DAQT.	
Prototipo elemental de interfaz de usuario: Sí. 	

Tabla. 12 Historia de usuario 6

Historia de usuario	
Número:6	**Nombre del requisito:** Recibir información de la configuración DAQT.
Programador: Brian Estrada Rodríguez	**Iteración asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 2
Riesgo en desarrollo: Alto	**Tiempo real:** 2

Observaciones: Se debe haber establecido la comunicación con el Servidor Modbus en la DAQT.
Descripción: Esta funcionalidad permite recibir los valores de configuración de la DAQT: Tipo de hardware (Arduino), sensores y actuadores configurado (*input, output, reserved, no used*) en el GPIO del Arduino, registros de modbus (*coil status, input status, holding register, input register*), parámetros del módulo de comunicación ESP8266 tales como: *versión, intensidad de la señal WIFI, modo de transmisión, cantidad máxima de conexiones, local IP, local puerto*).
Prototipo elemental de interfaz de usuario: Sí.

Tabla. 13 Historia de usuario 7

Historia de usuario	
Número:7	**Nombre del requisito:** Enviar información de la configuración DAQT.
Programador: Brian Estrada Rodríguez	**Iteración asignada:** 3
Prioridad: Alta	**Tiempo estimado:** 1
Riesgo en desarrollo: Alto	**Tiempo real:** 1
Observaciones: Se debe haber establecido la comunicación con el Servidor Modbus en la DAQT.	
Descripción: Esta funcionalidad permite enviar los valores de configuración de la DAQT: Tipo de hardware (Arduino), sensores y actuadores configurado (*input, output, reserved, no used*) en el GPIO del Arduino, registros de modbus (*coil status, input status, holding register, input register*), parámetros del módulo de comunicación ESP8266 tales como: *versión, intensidad de la señal WIFI, modo de transmisión, cantidad máxima de conexiones, local IP, local puerto*).	
Prototipo elemental de interfaz de usuario: N/A.	

Tabla. 14 Historia de usuario 8

Historia de usuario	
Número:8	**Nombre del requisito:** Deshabilitar servicio de la WIFI.

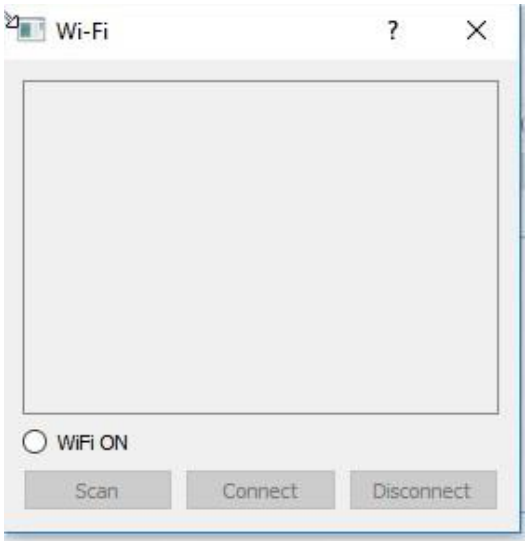
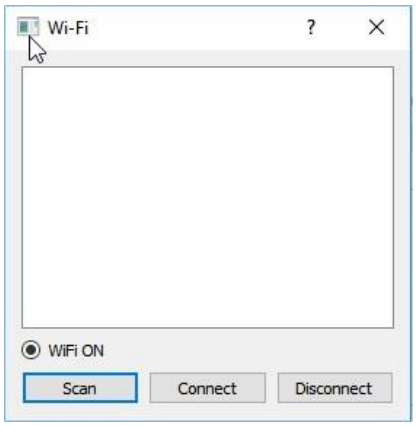
Programador: Brian Estrada Rodríguez	**Iteración asignada:** 3
Prioridad: Alta	**Tiempo estimado:** 1
Riesgo en desarrollo: Alto	**Tiempo real:** 1
Observaciones: Se debe haber habilitado el componente WIFI.	
Descripción: Esta funcionalidad permite apagar en componente WIFI.	
Prototipo elemental de interfaz de usuario: Sí. 	

Tabla. 15 Historia de usuario 9

Historia de usuario	
Número:8	**Nombre del requisito:** Habilitar servicio de la WIFI.
Programador: Brian Estrada Rodríguez	**Iteración asignada:** 3
Prioridad: Alta	**Tiempo estimado:** 1
Riesgo en desarrollo: Alto	**Tiempo real:** 1

Observaciones: Se debe haber deshabilitado el componente WIFI.
Descripción: Esta funcionalidad permite encender en componente WIFI.
Prototipo elemental de interfaz de usuario: Sí. 

2.3.1 Estimación de esfuerzo por historia de usuario

La tabla que se muestra a continuación, muestra la estimación del esfuerzo por historia de usuario (HU), según el orden en que aparecen. Esto se hace con la intención de que los programadores obtengan una estimación de dichas historias en cuanto al nivel de detalle, o sea, para fijar el período de tiempo que se puede tardar en la implementación de cada una. Un punto es considerado una semana ideal de trabajo, donde se trabaja todo el tiempo planeado ininterrumpidamente.

Tabla. 16 Estimación de esfuerzo

Historias de usuario	Puntos estimados
Escanear los dispositivos WIFI.	1
Conectar con la DAQT.	1
Desconectar con la DAQT	1
Establecer comunicación con el Servidor Modbus en la DAQT.	3
Visualizar Datos desde el Servidor Modbus en la DAQT.	2
Recibir información de la configuración DAQT.	2
Enviar información de la configuración DAQT.	1
Deshabilitar Servicio de la WIFI.	1
Habilitar Servicio de la WIFI.	1

2.4 Plan de entregas

El plan de entregas se elabora una vez se culmina la realización de las HU. Aquí se especifica en qué entrega será implementada cada historia de usuario, y se fija una fecha para la culminación de la misma.

Tabla. 17 Plan de entregas

Plan de entregas				
Driver para acceso a dispositivos Modbus a través de tecnologías Inalámbricas				
Fecha de reunión de planificación			4-12-2017	
Nombre del documentador			Brian Estrada Rodríguez	
Historias de usuario a implementar				
No. HU	**Título**	**Prioridad**	**Fecha en la que se hará entrega**	**Liberación en la que será incluida**
1	Escanear los dispositivos WIFI.	Alta	26-3-2018	1
2	Conectar con la DAQT.	Alta	16-4-2018	1
3	Desconectar con la DAQT	Alta	27-4-2018	1
4	Establecer comunicación con el Servidor Modbus en la DAQT.	Alta	6-5-2018	2
5	Visualizar Datos desde el Servidor Modbus en la DAQT.	Alta	22-5-2018	2
6	Recibir información de la configuración DAQT.	Alta	26-5-2018	2
7	Enviar información de la configuración DAQT.	Alta	4-6-2018	3
8	Deshabilitar Servicio de la WIFI.	Alta	11-6-2018	3
9	Habilitar Servicio de la WIFI.	Alta	18-6-2018	3

2.5 Plan de iteraciones

Iteración 1

Tabla. 18 Iteración 1

Historia de usuario	Tiempo estimado	Iteración asignada	Tiempo real
Escanear los dispositivos WIFI.	1	1	1
Conectar con la DAQT.	1	1	1
Desconectar con la DAQT	1	1	1

Iteración 2

Tabla. 19 Iteración 2

Historia de usuario	Tiempo estimado	Iteración asignada	Tiempo real
Establecer comunicación con el Servidor Modbus en la DAQT.	3	2	3
Visualizar Datos desde el Servidor Modbus en la DAQT.	2	2	2
Recibir información de la configuración DAQT.	2	2	2

Iteración 3

Tabla. 20 Iteración 3

Historia de usuario	Tiempo estimado	Iteración asignada	Tiempo real
Enviar información de la configuración DAQT.	1	3	1
Deshabilitar Servicio de la WIFI.	1	3	1
Habilitar Servicio de la WIFI.	1	3	1

2.6 Arquitectura de software

Arquitectura de software La definición oficial que define la IEEE Std 1471-2000 de Arquitectura de Software adoptada también por Microsoft es la siguiente: “La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos el contexto en el que se implantarán y los principios que orientan su diseño y evolución”.

2.6.1 Patrón Arquitectónico

Los patrones arquitectónicos son plantillas que describen los principios estructurales globales que construyen las distintas Arquitecturas de Software viables. Plantean una organización estructural fundamental para un sistema de software, expresando un conjunto de subsistemas predefinidos, especificando responsabilidades y organizando las relaciones entre ellos. La selección de un patrón arquitectónico es además una decisión fundamental de diseño cuando se desarrolla un sistema de software. (20)

El enfoque para el desarrollo de la solución es el patrón de arquitectura en capas, esta arquitectura reside en definir un conjunto de capas que vayan disminuyendo el nivel de abstracción a medida que son más internas. La solución presenta el estilo Arquitectónico de tres capas definido en: Capas de Presentación, Lógica del Cliente y Transporte.

- **Capa Presentación**: En la capa de Presentación es donde el cliente interactúa con la aplicación, esta capa se va a encargar de la interfaz gráfica del manejador.
- **Capa Lógica del Cliente**: Tiene el objetivo de establecer los parámetros de la DAQT como: tipo de hardware (Arduino), sensores y actuadores configurado (*input, output, reserved, no used*) en el GPIO del Arduino, registros de modbus (*coil status, input status, holding register, input register*), parámetros del módulo de comunicación ESP8266 tales como: *versión, intensidad de la señal WIFI, modo de transmisión, cantidad máxima de conexiones, local IP, local puerto.*
- **Capa Transporte**: Tiene el objetivo de establecer la comunicación entre el manejador y el Servidor Modbus en la DAQT. Esta clase controla todos los pasos que define el protocolo para lograr la comunicación satisfactoria, segura y preparada a los fallos que pudieran existir en una conexión inalámbrica.

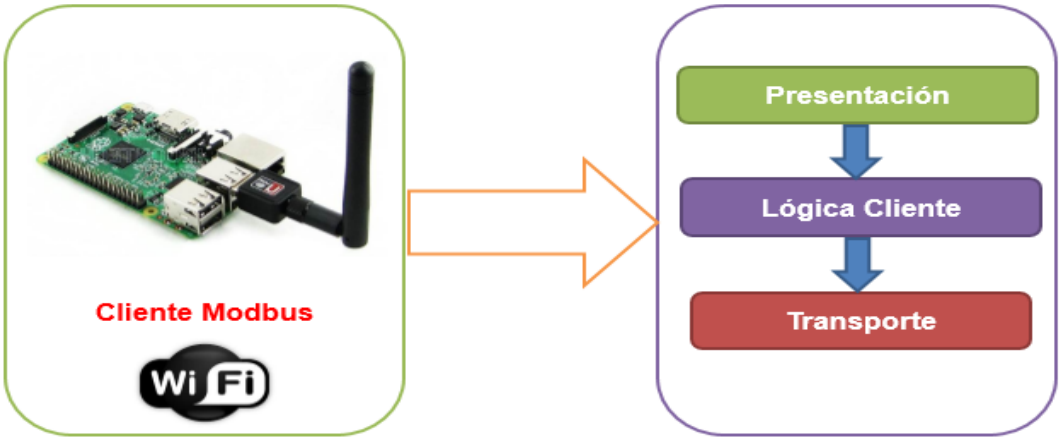


Fig. 10 Capas del sistema

2.7 Patrones de diseño

Los patrones de diseño son las descripciones de los objetos que se comunican y clases que son personalizadas para resolver un problema de diseño en general en un contexto particular. (21)
A continuación se exponen los patrones de diseño seleccionados:

2.7.1 Patrones GRASP

Los Patrones Generales de Asignación de Responsabilidades, por sus siglas en inglés (GRASP), son patrones de diseño que se usan para asignar responsabilidades a una clase.

- **Experto:** Asignar una responsabilidad al más competente en información, la clase cuenta con la información necesaria para cumplir la responsabilidad. Es el principio básico de asignación de responsabilidades que suele utilizarse en el diseño Orientado a Objetos.
- **Alta Cohesión:** Asignar una responsabilidad de modo que la unión se mantenga a gran escala. Asignar a las clases responsabilidades que trabajen sobre una misma área de aplicación y que no tengan mucha complejidad. Mejoran la claridad y facilidad con que se entiende el diseño.
- **Bajo Acoplamiento:** Asignar una responsabilidad para mantener un engranaje pobre. Es un principio que se debe recordar durante las decisiones de diseño. Soporta el diseño de clases más independientes.
- **Controlador:** Asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Asigna la responsabilidad del manejo de mensajes de los eventos de un sistema a una clase.

2.7.2 Patrones GoF

El patrón de diseño Singleton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella.

2.8 Diagrama de clases

Un diagrama de clases de diseño representa las especificaciones de las clases en una aplicación. Entre la información general se encuentran: clases, asociaciones, atributos, métodos, navegabilidad y dependencias. (22)

- **ConfigurationController**: Clase controladora principal, esta es la clase principal que contiene un conjunto de funciones y procedimientos que engloba el sistema.
- **DataCollector**: Esta clase es la encargada de conectarse a los dispositivos de adquisición de datos y recolectar la información de los mismos.
- **SensorManager**: Esta clase hace referencia al sensor físico ubicado en un GPIO de la DAQT.

- **ActuadorManager**: Esta clase hace referencia al actuador físico ubicado en un GPIO de la DAQT.
- **TransportManager**: Es la clase que prepara el transporte para el manejo del protocolo modbus por el transporte TCP/IP sobre la red WIFI.
- **ModbusTCPRegister**: Esta clase hace referencia al banco de memoria modbus de la DAQT.
- **WifiManager**: Esta clase hace referencia a la información del módulo WIFI ESP8266 en el Arduino.
- **HardwareManager**: Esta clase hace referencia al hardware (placa Arduino) que soporta la DAQT.

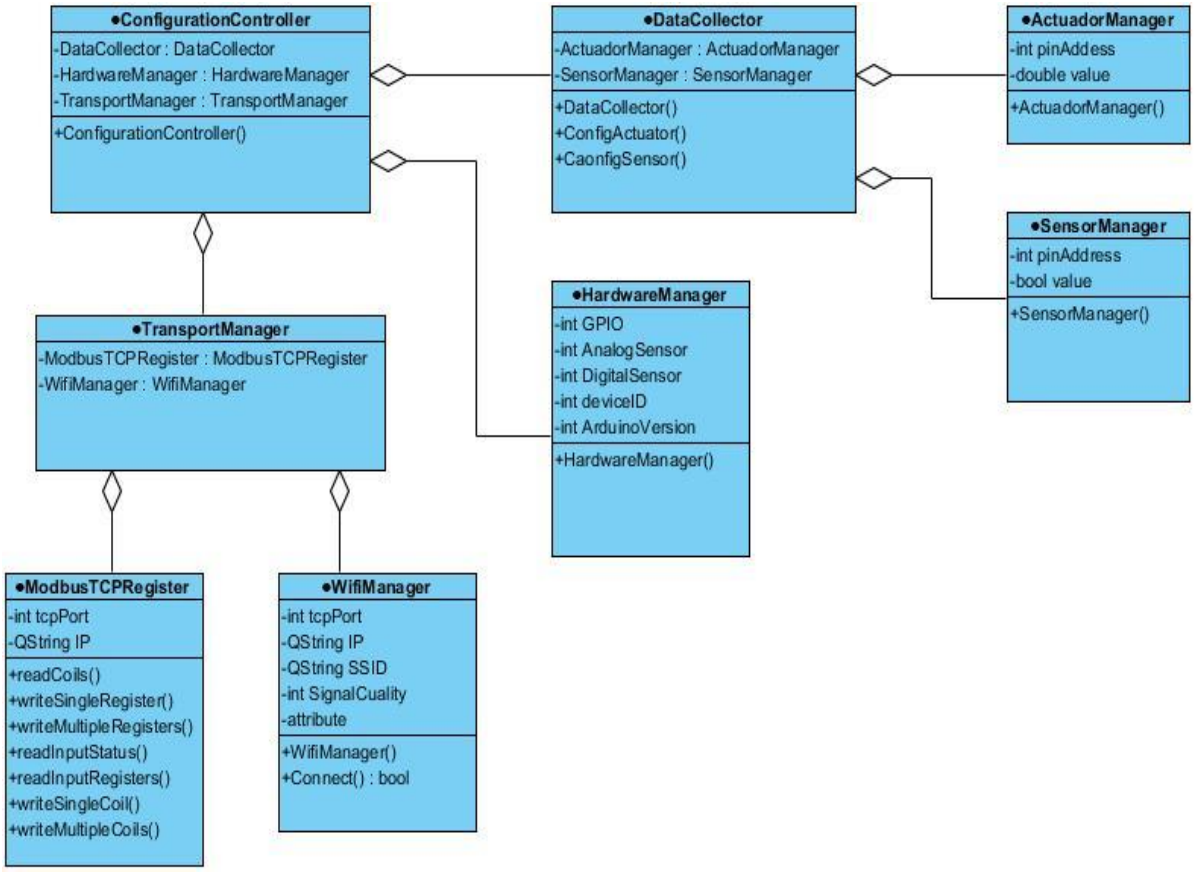


Fig. 11 Diagrama de clases

2.9 Conclusiones parciales

En este capítulo se definieron las características para el desarrollo de la solución, tales como: la adopción del patrón arquitectónico en capas, la definición de los principales requisitos funcionales y no funcionales con que debe contar el manejador. Además, fueron concebidas las historias de usuario para lograr una planificación en el desarrollo de las mismas. También, se definieron los patrones de diseño utilizados para lograr una correcta estructuración de la aplicación. Por último, fueron realizados diferentes diagramas UML, donde se refleja la dependencia entre las capas y las clases que los componen.

Capítulo III: Implementación y pruebas

Luego de haber sido modelado el sistema, especificado los requisitos del software y realizado el análisis y diseño del producto, comienza el flujo de implementación, que tiene como objetivo implementar las clases, probar los componentes desarrollados e integrarlos a un sistema ejecutable.

Con el objetivo de mostrar que la propuesta solución satisface los requerimientos del cliente y encontrar defectos en la aplicación, se hace necesario evaluar la calidad del producto, verificando el correcto funcionamiento de cada uno de los componentes del software.

3.1 Estilo de escritura

El estilo de escritura que se aplica a frases o palabras compuestas es el CamelCase y se utiliza de sus dos variantes el lowerCamelCase que es cuando la primera letra de cada una de las palabras es minúscula.

3.2 Estándares de codificación

Los programadores deben seguir estrictamente un grupo de reglas y condiciones conocidas como el estándar de codificación. En un equipo de programadores todos renuncian a su forma de programar particular y se ajustan a un estándar para el código. De esta forma al final de toda la implementación parece hecha por una misma persona. El ejemplo de esta buena práctica de desarrollo de software es de gran importancia y se traduce en una mayor calidad de desarrollo.

El estándar de codificación conduce a una mayor coherencia entre el código personal y el de los compañeros del equipo, y esto a su vez permite generar un código más fácil de entender que facilita su desarrollo y mantenimiento, además reduce el costo total de las aplicaciones a crear. (23)

A continuación se muestran algunos estándares de codificación utilizados:

Declaraciones: Los métodos y variables se declararon con nombres asociados a la función por la cual fueron creados. En caso de nombres compuestos, el primero se definió en minúscula y los demás comenzando con mayúscula.

```
private:
    Ui::wifi *ui;
    QProcess *p,*c,*d,*e,*s;
    QString wif;
    int count= 0;
};
```

Fig. 12 Declaraciones

Controlador: Sentencia if, else: Se definió para estos dos tipos de sentencias la siguiente estructura:

En el if y el else, se utilizó llaves para abrir y cerrar el bloque de acción (es) que se ejecutan dada(s) su (s) condición (es).

```
void MainWindow::networkUpdate(bool isOnline)
{
    if(isOnline) {
        this->Activar();

    } else {
        this->Desactivar();
    }
    //qDebug()<<"Network is online: " << isOnline;
}
```

Fig. 13 Controlador

Clases: Las clases deben comenzar con mayúscula y en caso de estar conformada por palabras compuestas, la definición debe ser continua y cada palabra debe iniciar con mayúscula siguiendo el estilo determinado.

```
class wifi : public QDialog
{
    Q_OBJECT

public:
    explicit wifi(QWidget *parent = 0);
    ~wifi();
```

Fig. 14 Clases

3.3 Modelo de implementación

El flujo de implementación está fuertemente determinado por el lenguaje de programación y su propósito principal es desarrollar la arquitectura y el sistema como un todo. Describe cómo los elementos del modelo del diseño se implementan en términos de componentes (ficheros de código fuente, archivos ejecutables, librerías, scripts, tablas, bases de datos y documentos) y cómo estos se organizan de acuerdo a los nodos especificados en el modelo de despliegue, generado en flujo de análisis y diseño. Los diagramas de despliegue y los diagramas de componentes conforman lo que se conoce como un modelo de implementación.

3.3.1 Diagrama de componente

Un diagrama de componentes representa como un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Se usan para modelar y documentar cualquier arquitectura de sistema y muestra, además, la organización y las dependencias entre un conjunto de componentes.

A continuación se muestra una representación gráfica del diagrama de componentes para la capa de acceso inalámbrica para la comunicación de un servidor Modbus basada en Arduino.

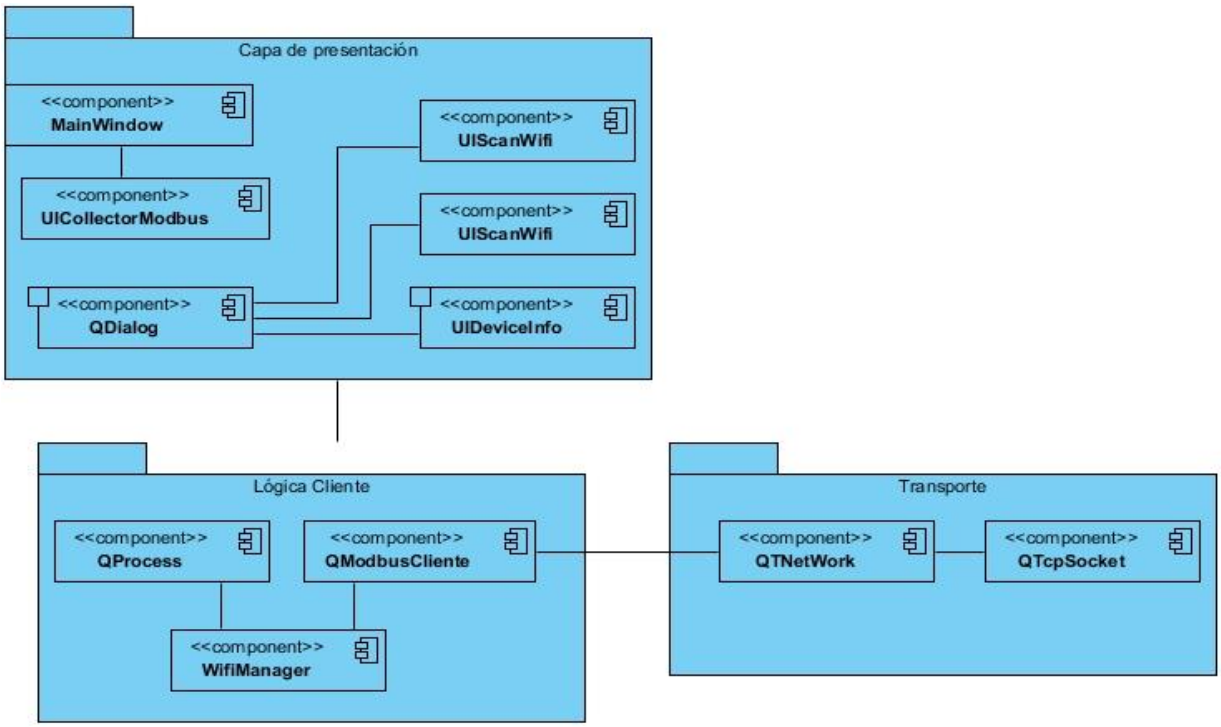


Fig. 15 Diagrama de componente

Descripción de los componentes:

- **Componentes de capa de presentación:** Se encuentran los componentes visuales que tienen la responsabilidad de crear la interfaz visual o GUI para la interacción con el usuario.
- **Componentes de capa de Lógica Cliente:** Se encuentran los principales componentes que encapsulan la lógica del protocolo modbus y el manejo de la WIFI.
- **Componentes de capa de Transporte:** Se encuentran los principales componentes que encapsulan los mecanismos de conexión TCP/IP empleados por el protocolo modbus.

3.3.2 Diagrama de despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final. Este diagrama es útil para ilustrar la arquitectura física de un sistema.

Seguidamente se muestra una representación gráfica donde se evidencia el sistema genérico Arex conectado con la tarjeta de adquisición de datos basada en Arduino mediante una capa de acceso inalámbrico por los protocolos Modbus RTU y TCP.

Fig. 16 Diagrama de despliegue

Descripción de los nodos:

- **Nodo Cliente Recolector de Datos:** Se estará ejecutando un cliente recolector de datos (recolector de Arex o superior) mediante el componente driver o manejador Modbus TCP sobre WIFI, el mismo se ejecutará sobre el hardware de una Raspberry pi.
- **Nodo DAQT**: Este nodo está compuesto por la DAQT con el módulo ESP8266 y se estará ejecutando el firmware para la recolección de datos de los sensores y actuadores.
- La comunicación se establecerá sobre una red inalámbrica WIFI.

3.4 Pruebas de software

Las pruebas de software son el proceso de ejecución de un programa con la intención de descubrir un error. Su objetivo fundamental es diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores haciéndolo con la menor cantidad de tiempo y esfuerzos posible.

Las pruebas de software son un elemento para garantizar el correcto funcionamiento de la aplicación. Entre sus metas se encuentra:

- ✓ Detectar defectos en el software.
- ✓ Verificar la integración adecuada de los componentes.
- ✓ Verificar que todos los requisitos se han implementado correctamente.
- ✓ Identificar y asegurar que los fallos encontrados durante el proceso de prueba, se han corregido antes de entregar el software al cliente.

3.5 Ambiente de pruebas

Las pruebas se realizaron sobre los ambientes siguientes:

- ✓ Una PC con modulo USB WIFI.
- ✓ Una tarjeta de adquisición de datos basada en Arduino UNO con el módulo ESP8266.

3.6 Diseño de caso de pruebas

Los casos se diseñaron a partir de las historias de usuario. Una historia de usuario puede tener tantos casos de prueba como sean necesarios para evaluar su funcionamiento. A continuación se muestran los casos de prueba de aceptación elaborados para la aplicación:

Tabla. 21 Caso de prueba de aceptación 1

Caso de prueba de aceptación
Historia de usuario: Escanear los dispositivos WIFI.
Responsable: Brian Estrada Rodríguez.
Descripción: Esta funcionalidad permite escanear los dispositivos al alcance de la WIFI. Mostrando un listado con el nombre de cada señal y permite seleccionarlas para una posterior conexión.
Condiciones de ejecución: La DAQT se encuentra apareada correctamente a la computadora mediante WIFI.
Resultado esperado: La aplicación se conectó con el dispositivo DAQT.
Evaluación de prueba: Prueba satisfactoria.

Tabla. 22 Caso de prueba de aceptación 2

Caso de prueba de aceptación
Historia de usuario: Conectar con la Tarjeta de Adquisición de Datos (DAQT).
Responsable: Brian Estrada Rodríguez.
Descripción: Esta funcionalidad permite conectarse a la WIFI de la Tarjeta de Adquisición de Datos y habilita los campos para una conexión modbus, como también la lectura y escritura de los registros (*Coils, Input, Input Register, Holding Register*) del dispositivo.
Condiciones de ejecución: La tarjeta se encuentra conectada correctamente a la computadora mediante Wifi.
Resultado esperado: La aplicación no se conectó con el dispositivo.
Evaluación de prueba: Prueba satisfactoria.

Tabla. 23 Caso de prueba de aceptación 3

Caso de prueba de aceptación
Historia de usuario: Desconectar con la Tarjeta de Adquisición de Datos.
Responsable: Brian Estrada Rodríguez.
Descripción: Esta funcionalidad permite desconectarse de la WIFI de la Tarjeta de Adquisición de Datos y desactiva los campos de conexión, lectura y escritura del dispositivo.
Condiciones de ejecución: Se debe haber establecido la comunicación WIFI con la DAQT.
Resultado esperado: El dispositivo se desconectó a la WIFI de la Tarjeta de Adquisición de Datos.
Evaluación de prueba: Prueba satisfactoria.

Tabla. 24 Caso de prueba de aceptación 4

Caso de prueba de aceptación
Historia de usuario: Establecer comunicación con el Servidor Modbus en la DAQT.
Responsable: Brian Estrada Rodríguez.
Descripción: Esta funcionalidad permite establecer la comunicación con el servidor modbus configurado en la DAQT, y otorga los permisos de lectura y escritura sobre los registros del Arduino.
Condiciones de ejecución: Se debe haber establecido la comunicación WIFI con la DAQT.
Resultado esperado: Se estableció la comunicación con el servidor modbus configurado en la DAQT.
Evaluación de prueba: Prueba satisfactoria.

Tabla. 25 Caso de prueba de aceptación 5

Caso de prueba de aceptación
Historia de usuario: Visualizar Datos desde el Servidor Modbus en la DAQT.
Responsable: Brian Estrada Rodríguez.
Descripción: Esta funcionalidad permite visualizar los valores de los registros de Modbus en el Servidor de la DAQT.
Condiciones de ejecución: Se debe haber establecido la comunicación con el Servidor Modbus en la DAQT.
Resultado esperado: Se visualizaron los valores de los registros de Modbus (*coil status, input status, holding register, input register*), en el Servidor de la DAQT.
Evaluación de prueba: Prueba satisfactoria.

Tabla. 26 Caso de prueba de aceptación 6

Caso de prueba de aceptación
Historia de usuario: Recibir información de la configuración DAQT.
Responsable: Brian Estrada Rodríguez.
Descripción: Esta funcionalidad permite recibir los valores de configuración de la DAQT.
Condiciones de ejecución: Se debe haber establecido la comunicación con el Servidor Modbus en la DAQT.
Resultado esperado: Se recibieron los valores de configuración de la DAQT: Tipo de hardware (Arduino), sensores y actuadores configurado (*input, output, reserved, no used*) en el GPIO del Arduino, registros de modbus (*coil status, input status, holding register, input register*), parámetros del módulo de comunicación ESP8266 tales como: *versión, intensidad de la señal WIFI, modo de transmisión, cantidad máxima de conexiones, local IP, local puerto*.

Evaluación de prueba: Prueba satisfactoria.

Tabla. 27 Caso de prueba de aceptación 7

Caso de prueba de aceptación
Historia de usuario: Enviar información de la configuración DAQT.
Responsable: Brian Estrada Rodríguez.
Descripción: Esta funcionalidad permite enviar los valores de configuración de la DAQT.
Condiciones de ejecución: Se debe haber establecido la comunicación con el Servidor Modbus en la DAQT.
Resultado esperado: Se enviaron los valores de configuración de la DAQT: Tipo de hardware (Arduino), sensores y actuadores configurado (*input, output, reserved, no used*) en el GPIO del Arduino, registros de modbus (*coil status, input status, holding register, input register*), parámetros del módulo de comunicación ESP8266 tales como: *versión, intensidad de la señal WIFI, modo de transmisión, cantidad máxima de conexiones, local IP, local puerto*.
Evaluación de prueba: Prueba satisfactoria.

Tabla. 28 Caso de prueba de aceptación 8

Caso de prueba de aceptación
Historia de usuario: Modificar configuración
Responsable: Brian Estrada Rodríguez.
Descripción: Esta funcionalidad permite modificar una configuración en la DAQT.
Condiciones de ejecución: Conexión establecida con el dispositivo.
Resultado esperado: Se modificaron los valores de configuración de los campos: Tipo de hardware (Arduino), sensores y actuadores configurado (*input, output, reserved, no used*) en el GPIO del Arduino, registros de modbus (*coil status, input status, holding register, input register*), parámetros del módulo de comunicación ESP8266 tales como: *versión, intensidad de la señal WIFI, modo de transmisión, cantidad máxima de conexiones, local IP, local puerto*.
Evaluación de prueba: Prueba satisfactoria.

Tabla. 29 Caso de prueba de aceptación 9

Caso de prueba de aceptación
Historia de usuario: Deshabilitar Servicio de la WIFI.

Responsable: Brian Estrada Rodríguez.
Descripción: Esta funcionalidad permite apagar en componente WIFI.
Condiciones de ejecución: Se debe haber habilitado el componente WIFI.
Resultado esperado: Se deshabilitó el servicio de la WIFI.
Evaluación de prueba: Prueba satisfactoria.

Tabla. 30 Caso de prueba de aceptación 10

Caso de prueba de aceptación
Historia de usuario: Habilitar Servicio de la WIFI.
Responsable: Brian Estrada Rodríguez.
Descripción: Esta funcionalidad permite encender en componente WIFI.
Condiciones de ejecución: Se debe haber deshabilitado el componente WIFI.
Resultado esperado: Se habilitó el servicio de la WIFI.
Evaluación de prueba: Prueba satisfactoria.

3.9 Ejecución de los casos de pruebas de aceptación

En la primera iteración se detectaron un total de cinco no conformidades de ellas, una para la HU1: Escanear dispositivo WIFI, una para la HU4: Establecer comunicación con el servidor modbus en la DAQT y una para la HU9: Habilitar Servicio de la WIFI.

Las no conformidades detectadas se resolvieron en la tercera iteración, donde el cliente especificó que los resultados de los casos de prueba se correspondían con los resultados esperados.

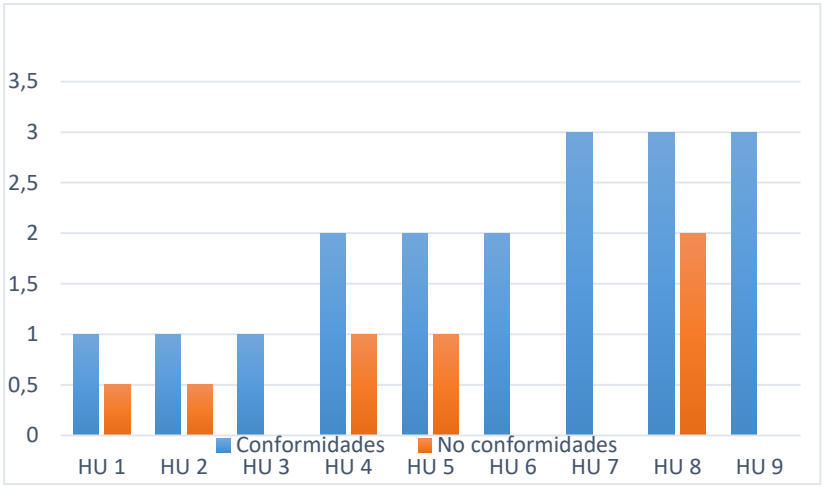


Fig. 17 Resumen de defectos y dificultades

3.10 Conclusiones parciales

En este capítulo se diseñaron las pruebas que se le realizaron a la solución y el ambiente en el que se efectuaron las mismas, permitiendo mejorar en algunos casos el correcto funcionamiento del Manejador Modbus como Cliente sobre WIFI, siempre teniendo en cuenta que algunas pruebas que fueron efectuadas dieron algún error.

Conclusiones generales

- ✓ Como producto final se obtiene un manejador o driver que permite establecer una comunicación mediante el protocolo modbus con los dispositivos DAQT-WIFI a través de una red inalámbrica.
- ✓ El análisis de las metodologías, tecnologías y herramientas posibilitó la selección del ambiente de desarrollo adecuado para el desarrollo del manejador, empleando las potencialidades del framework QT en su versión 5.10.
- ✓ La documentación necesaria para llevar a cabo las actividades de implementación se obtuvo mediante el análisis y diseño del sistema.
- ✓ Las pruebas realizadas permitieron verificar que el sistema desarrollado cumpliera con los requisitos propuestos, validándose además la calidad de la aplicación.

Recomendaciones

- ✓ Se propone para futuras investigaciones realizar un manejador para la comunicación de un servidor modbus mediante el dispositivo bluetooth 5.0.
- ✓ Integrar el Manejador Cliente Modbus/TCP sobre WIFI en la Capa de Recolección de Datos del proyecto Arex.

Referencias bibliográficas

Bibliografía

1. **Viltre, chavez Jordanis, Gali Ramirez, Diana Tahiri y Leyva Duran, Julio Alberto.** Sistema para la adquisición de datos, monitoreo y control en procesos de pequeña y mediana complejidad. 2017.

2. **Barrios Duenas, Joel.** Introducción a TCP/IP. [En línea] http://www.alcancelibre.org/staticpages/index.php/introduccion-tcp-ip.

3. **Gomez, Ruben.** Capa de acceso a datos síncrona y asíncrona para dispositivos Modbus tcp.La Habana, Cuba: Universidad de las Ciencias Informáticas, 2008.

4. **Modbus Messaging Implementation** Guide v1.doc. MODBUS.ORG. [En línea] 1 de Mayo de 2018. [Citado el: 3 de Febrero de 2018] http://www.modbus.org.

5. **Modbus over Ethernet Access Modbus data over IP**. Eltima.com. [En línea] [Citado el: 10 de Marzo de 2018] http://www.eltima.com.

6. **Bedoya Meneses, Jessica Pamela.**Estudio comparativo de la eficiencia del medio de comunicación alambrico e inalambrico del protocolo modbus implementado en un proceso modular. Riobamba,Escuela Superior Pilitecnica de Chimborazo, 2015.

7. **ESP8266 Datasheet.** [En línea] Febrero de 2018. [Citado el: 3 de Marzo de 2018.] https://www.espressif.com.

8. **Tarjeta de adquisición de datos (DAQ)**. prezi.com [En línea] 9 Septiembre 2013. [Citado el: 20 de Marzo de 2018]. www.prezi.com/tajeta-de-adquisicion-de-datos-daq/

9. **CRAIG, L. UML y Patrones Introducción al análisis y diseño orientado a objetos**. 2004.

10. **Arduino.** What is Arduino. Arduino.cc [En línea] 2014. [Citado el: 5 de Abril de 2018]. http://arduino.cc/en/guide/introduction.

11. **Mediavilla, Elena. Modelados y herramientas UML**. [En línea] 2016. [Citado el: 10 de Abril de 2018]. www.ctr.unican.es/

12. **Sanchez Franco, Mauricio Alberto y Pavlich-Mariscal, Jaime A. Modeling tool.** [En línea] 2016. [Citado el: 21 de Abril de 2018]. www.ieeexplore.org/

13. **Schmuller, J. Aprendiendo UML**. [En línea] 2000. [Citado el: 29 de Abril de 2018]. www.u-cursos.cl/

14. **Pressman, Roger S. Cap 09 Ingeniería del Diseño**, Epíg 9.3.2. bibliodoc.uci.cu. [En línea] [Citado el: 5 de mayo de 2018.] http://bibliodoc.uci.cu/pdf/8448111869.pdf.

15. **Gallardo López, D. Lenguajes de programación.** 2008.

16. **Rojas, Alan D. Osorio. C++ Manual teórico-práctico.** 2006.

17. **Suárez Falcón, Yuniel. IDE de programación.** 2015.

18. **QT Creator.** [En línea] [Citado el: 20 de mayo de 2018.] https://wiki.qt.io/

19. **Pérez, Velazco y Mir, Vasoncelo.** Desarrollo de la extensión para la configuración del módulo de adquisición de datos del SCADA Guardián del ALBA. 2010.

20. **Patricio Letelier Torres, Emilio A. Sánchez López. Metodologías Ágiles en el Desarrollo de Software**. España : s.n., 2003.

21. **CAVENAGO, ADRIANA SANDRA ALMEIR and VANINA PEREZ. Arquitectura de Software: Estilos y Patrones***.* San Juan Bosco : s.n., 2007.

22. **CRAIG, L. UML y Patrones Introducción al análisis y diseño orientado a objetos.** 2004.

23. **Vidal Otero, Mari Carmen. Dpto de lenguajes y Sistemas informáticos.** 2003.

24. **UC3M. Guión Visual Paradigm for UML. Ingeniería del Software.** Curso 2013-2014.