Universidad de las Ciencias Informáticas

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

Título:

Capa de acceso inalámbrica para la comunicación de un servidor Modbus basada en Arduino.

Autora:

Jessica Bárbara Monja Vizcaíno

Tutores:

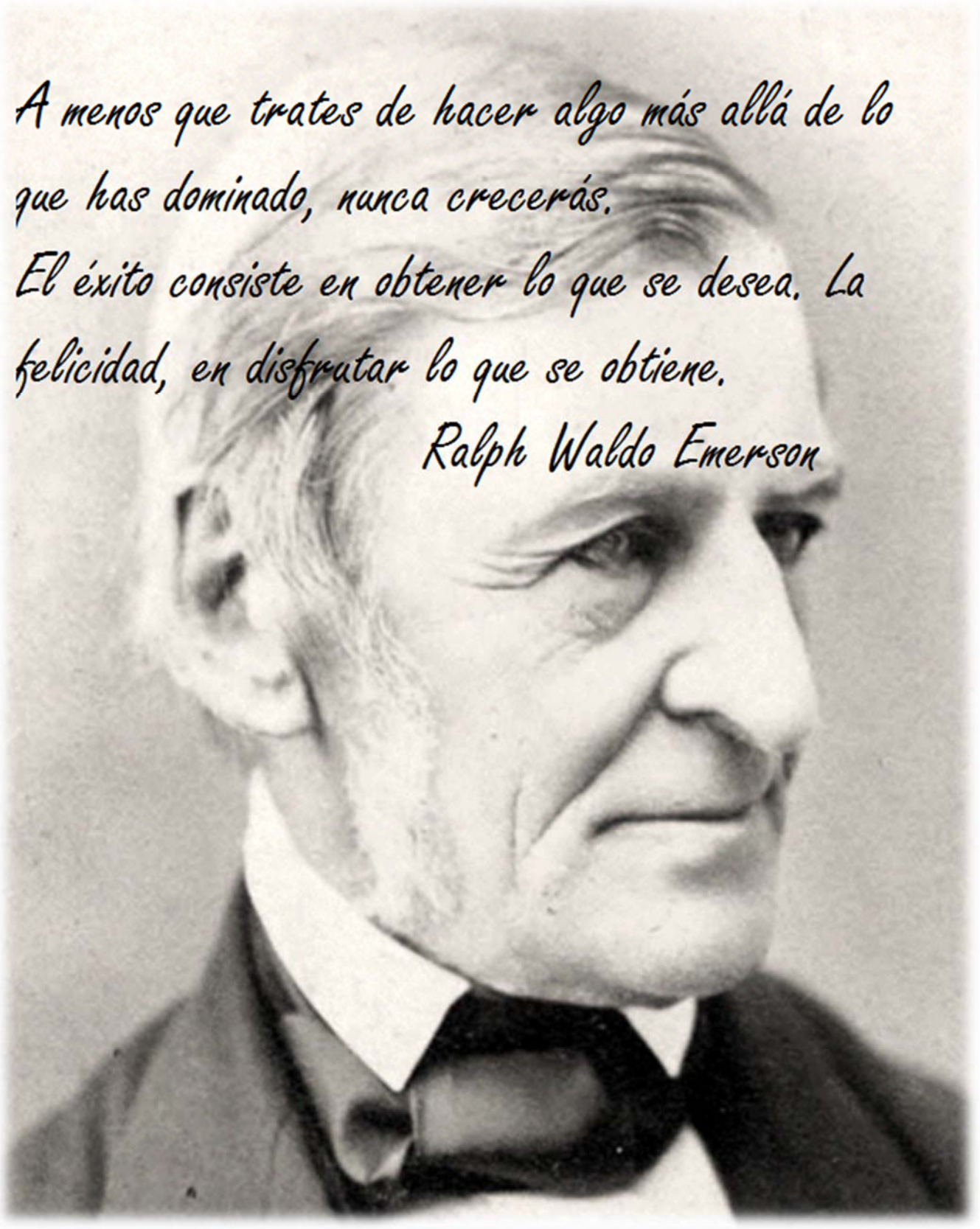
Ing. Julio Alberto Leyva Durán

Ing. Jeiser Medrano Abreu

MSc. Yadira Ramírez Rodríguez

La Habana, junio de 2018

Pensamiento



Agradecimientos

A mi mamá, por ser la mejor mamá de todas, por ser mi papá y toda una familia completa cuando la necesité, por darme la seguridad de que mientras tú existas no hay nada que no pueda hacer, la mujer que sé que aunque yo tenga 60 años me seguirá diciendo cuquita, porque hoy estoy aquí gracias a ti y a la fuerza que me das con ese carisma que nadie puede opacar, por ser mi loquita preferida y que aunque la mayoría del tiempo estemos lejos yo siempre estoy donde estés, y tú siempre estarás donde esté yo. Hoy gracias a ti tu cuquita es ingeniera.

A mi abu, todo el que me conoce aunque sea un poco ha oído hablar de abucusi, la que no duerme cuando no duermo, la que siente dolor cuando me duele y la que se despierta de madrugada para ver si estoy durmiendo bien. Tengo una abuela maravillosa que no cambio por todo el oro del mundo porque hoy aunque se hubiera levantado con la presión a millón y aunque se estuviera muriendo del dolor en sus piernas, nadie le iba a decir que no iba a estar aquí para sentirse la abuela más orgullosa de todas. Abucusi lo logramos…

Yo no hablo mucho de él, no me gusta, pero si este no es el momento no encuentro otro, quiero agradecerle al hombre con los mejores conceptos y principios que conocí, al hombre más integro de todos los tiempos, a mi abuelo. Me acuerdo cuando te dije que me habían dado informática y tú incluso sin saber lo que era me dijiste que iba a ser la mejor informática de Cuba. Yo sé que debes estar por ahí en cualquier lugar de este salón viendo como lo logré y diciendo "salió a mí".

A mi tía, que no sé cómo lo hizo pero todas las mañanas antes de cada prueba me mandaba un mensaje que increíblemente siempre era el mismo "cuqui te deseo mucha suerte, mantén la calma que tu si puedes", por creer en mí incluso cuando yo misma no lo hacía por estar siempre y por ser mi mamá con nacionalidad cubana.

A mis tíos, mis primos, mi inmensa familia por su amor incondicional, a mi papá, a mi chuchi por tener ese carisma que me levanta siempre el ánimo, te quiero mi tatico bello.

En 5 años conoces mucha gente, gente que se vuelven amigos y amigos que se vuelven familia.

Cuando alguien me diga que no existe amistad entre un hombre y una mujer le contaré de ti, Medu, de la persona que me hace reír todo el tiempo aunque el tiempo no sea el mejor, con la que comparto millones de afinidades y podríamos pasarnos un día entero hablando de cualquier bobería que se le ocurriera, por ser más que mi guardaespaldas, por ser el amigo como los que

no existe ya, por tus leches con grumos, por las tarde de frambo, por ser siempre ser el mismo, por ser siempre tú, gracias.

Lianet y Lidia por enseñarme a quererlas como a mi familia, por aguantarme en mis malos ratos y estar ahí cuando la pasaba mal. Por estar en mis malos días, por guardarme secretos hasta hoy, por ser las amigas que todos queremos tener.

Mario Jorge por ser mi apoyo incondicional hasta hoy, por estar cuando nadie está y ser la persona que jamás olvidaré. Anabel por ser mi eterno tormento, Kevin por hacerme sentir siempre que tengo alguien con quien contar y Sergio por enseñarme que hay personas especiales en el mundo y volverme fans número uno de Rihanna.

Gracias a todos.

Dedicatoria

“A mis abuelos, por darme todo el amor del mundo y más, por inculcarme valores que me han hecho mejor persona, y que han logrado hacer de su niña una mujer, y hoy una ingeniera”

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	Firma del Tutor
Jessica B. Monja Vizcaíno	Ing. Julio A. Leyva Durán	MSc. Yadira Ramírez

_____

Firma del Tutor

Ing. Jeiser Medrano Abreu

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.

Yadira Ramírez Rodríguez: Graduada de Ingeniera en Ciencias Informáticas en el año 2007. Máster en Calidad de Software en la UCI en el año 2017. Es profesora auxiliar desde el 2017, ha impartido pregrado de Ingeniería de Software I y II desde el año 2007. A partir del curso 2015-2016 se desempeña como Vicedecana de Formación de la facultad 4.

Jeiser Medrano Abreu: Ingeniero en Ciencias Informáticas desde el año 2014, obtuvo la categoría docente instructor en el año 2017. Se desempeña como profesor de las asignaturas Gestión de Software y Componente Profesional de Ingeniería y Gestión de Software desde el año 2015.

Resumen

El presente trabajo surge por la necesidad de realizar una variante de comunicación de la tarjeta de adquisición de datos con el sistema Arex de manera fácil y de bajo costo debido a que actualmente se establece esta comunicación sobre una red cableada de forma engorrosa y poco estética. Para realizar la propuesta planteada se trazó como objetivo desarrollar una capa de acceso inalámbrico para la comunicación de un servidor Modbus basada en Arduino mediante una red WiFi. Se utilizó AUP como metodología de desarrollo de software en su 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 Unificado de Modelado (UML) en la versión 6.4. Se hace uso del lenguaje de programación C++ y el entorno de desarrollo integrado Arduino en la versión 1.8.5. Se realizó el diseño e implementación de la herramienta y se le aplicaron diferentes pruebas de software permitiendo detectar y corregir la máxima cantidad de errores antes de su entrega al cliente. Finalmente se logró una herramienta que entre sus principales funcionalidades permite la comunicación mediante la red WiFi de la tarjeta de adquisición de datos basada en Arduino con el sistema genérico Arex.

Palabras claves: Arduino, sistema Arex, tarjeta de adquisición de datos, WiFi.

Índice

Índice de figuras

Índice de tablas

Introducción

La adquisición de datos o data acquisition por sus siglas en inglés DAQ es el proceso de medir con una computadora (PC) un fenómeno eléctrico o físico como el voltaje, la temperatura, la presión o el sonido. Un sistema DAQ consiste de sensores, tarjetas de adquisición de datos y una computadora con software programable. Los sistemas DAQ basados en PC aprovechan la potencia del procesamiento, la productividad, la visualización y las habilidades de conectividad de las PCs estándares en la industria proporcionando una solución de medidas más potente, flexible y rentable. (1)

Las tarjetas de adquisición de datos (DAQT por sus siglas en inglés) se encargan de transformar los códigos del mundo real a los códigos digitales, como si se tratara de un traductor que convierte de un lenguaje a otro, con el fin de que el sistema digital (es decir, cualquier computadora o dispositivo electrónico) sea capaz de comprender los signos del analógico. (1)

La medición de un fenómeno físico, como la temperatura de una habitación, la intensidad de una fuente de luz o la fuerza aplicada a un objeto, comienza con un sensor. Un sensor, también llamado un transductor, convierte un fenómeno físico en una señal eléctrica que se puede medir. Dependiendo del tipo de sensor, su salida eléctrica puede ser un voltaje, corriente, resistencia u otro atributo eléctrico que varía con el tiempo. Algunos sensores pueden requerir componentes adicionales y circuitos para producir correctamente una señal que puede ser leída con precisión y con toda seguridad por un dispositivo DAQ. (1)

El Centro de Informática Industrial de la Universidad de las Ciencias Informáticas cuenta con la línea de desarrollo de sistemas embebidos. Éste se dedica al desarrollo de prototipos de dispositivos y programas para los mismos, además de software para dispositivos móviles o para tarjetas de hardware libre como RaspBerry Pi o Arduino.

En la línea de sistemas embebidos se realizó una DAQT basada en la placa de desarrollo Arduino. La tarjeta de adquisición de datos permite desde el sistema de adquisición (en este caso Arex) recolectar el valor de las variables de numerosos sensores, específicamente dedicados a la domótica[1]. Actualmente el acceso del sistema Arex a la DAQT se establece sobre una red cableada, mediante el protocolo Modbus RTU (maestro-esclavo) o TCP (cliente servidor).

Con esta topología de red y según la estructura de los locales, en muchos escenarios cuando se implementa el proceso de instalación de los dispositivos de campos es engorroso debido a que el entramado de cable es costoso por las características de los locales, además en ocasiones para adquirir las señales eléctricas de actuadores y sensores, la distancia de los cables conlleva a pérdidas de señal debido a caída de voltajes por la longitud del mismo y el acceso desde un local hacia otro; surgiendo así la necesidad de pensar en la comunicación inalámbrica.

[1] Domótica: Sistemas capaces de automatizar una vivienda o edificación de cualquier tipo, aportando servicios de gestión energética, seguridad, bienestar y comunicación.

En la actualidad la comunicación inalámbrica se utiliza con frecuencia, debido a su fácil manejo sin la necesidad de un amplio conocimiento. Las tecnologías inalámbricas más utilizadas son WiFi, Bluetooth y ZigBee ya que trabajan en un rango de frecuencia no muy saturado (2). De las tres tecnologías antes mencionadas la WiFi se adecua mejor para redes de propósito general debido a que permite conexiones más rápidas, un rango de distancias mayor y mejores mecanismos de seguridad.

Con estas condiciones, surgió la posibilidad de implementar una variante de comunicación de la tarjeta DAQ con el recolector de Arex de forma inalámbrica y de bajo costo mediante WiFi.

Teniendo en cuenta la **situación problemática** descrita se enuncia el siguiente **problema de la investigación**: ¿Cómo realizar la comunicación de la tarjeta de adquisición de datos con el sistema Arex de manera inalámbrica mediante una red WIFI?

Teniendo como **Objetivo general**: desarrollar una capa de acceso inalámbrico para la comunicación de un servidor Modbus basada en Arduino.

Se determina como **Objeto de Estudio:** la comunicación de un servidor Modbus basada en Arduino y el **campo de acción** se delimita en el acceso inalámbrico de la tarjeta de adquisición de datos con el sistema Arex.

Para dar solución al objetivo general se plantean las siguientes **Tareas de investigación:**

- ✓ Estudio del protocolo de comunicación Modbus.
- ✓ Estudio del estado del arte.
- ✓ Identificación de los requisitos funcionales para el desarrollo de la solución propuesta.
- ✓ Generación de los artefactos relacionados con el análisis y diseño de la solución.
- ✓ Definición de la arquitectura de la solución cumpliendo con las políticas de software libre.
- ✓ Diseño e implementación de la solución propuesta.
- ✓ Validación y corrección de errores de la capa de acceso inalámbrica.

Obteniendo como **posible resultado** una capa de acceso inalámbrica para la comunicación de un servidor Modbus basada en Arduino.

Los métodos de investigación utilizados fueron:

Métodos teóricos

- ✓ Analítico sintético: se realizó el análisis de múltiples documentos acerca de los sistemas similares, las tarjetas de adquisición de datos y las capas de acceso inalámbrico.
- ✓ Análisis histórico lógico: Mediante el mismo se realizó un estudio del estado del arte y las soluciones similares.

Métodos empíricos utilizados son:

- ✓ Experimento: se emplea este método con el propósito de comprobar la funcionalidad de la capa de acceso inalámbrico para la comunicación de un servidor Modbus basada en Arduino.
- ✓ Modelación: se emplea a través de los diagramas de lenguaje de modelado (UML). Se utilizará en la representación de las características y las relaciones entre los objetos de la solución.

- ✓ Observación: se puso en práctica este método para observar el código de la comunicación del sistema Arex con las DAQT mediante cableado que utiliza el Centro de Informática Industrial, para poder a partir de lo antes descrito realizarlo mediante acceso inalámbrico.

El presente trabajo de diploma consta de tres capítulos estructurados de la siguiente forma:

Capítulo I: Fundamentación teórica: se abordan los principales elementos teóricos vinculados a la investigación, se realiza el estudio del estado del arte de dicha investigación y el análisis de las metodologías de desarrollo, tecnologías y herramientas a utilizar.

Capítulo II: Propuesta de solución: se describe la solución propuesta y los principales aspectos relacionados con su diseño. Se describe la arquitectura y los artefactos generados por la metodología seleccionada.

Capítulo III: Implementación y pruebas: se describen los artefactos relacionados con la implementación y las pruebas realizadas con el objetivo de validar su correcto funcionamiento y su correspondencia con los requerimientos especificados.

Capítulo I: Fundamentación teórica

Introducción

En el campo de informática industrial se ha hecho de gran importancia las tarjetas de adquisición de datos basadas en la placa de Arduino siendo muy utilizadas en los proyectos de sistemas embebidos, por esto surgen nuevas formas para facilitar su uso debido a que necesita un entramado extenso de cable para ser conectado al software que lo maneja; y se propone implementar una capa de acceso inalámbrico para la comunicación de un servidor Modbus basada en Arduino.

Para cumplir este objetivo es necesario que sean comprendidos en su totalidad los conceptos que se expondrán en el presente capítulo, además se desarrollará un estudio acerca de las soluciones similares relacionadas estrechamente con la propuesta de solución y las principales tecnologías, metodologías y herramientas que se utilizan.

1.1 Conceptos asociados al dominio del problema

En este epígrafe se exponen algunos conceptos relacionados con la investigación para llegar al entendimiento de la solución propuesta.

1.1.1 Sistema Arex

El software genérico Arex desarrollado por el Centro de Informática Industrial (CEDIN) de la Universidad de Ciencias Informáticas es un sistema para la adquisición, monitoreo y control de datos en procesos de pequeña y mediana complejidad, configurable a cualquier objetivo de automatización en procesos de hasta 1000 variables, a su vez está optimizado para funcionar exitosamente en hardware de bajas prestaciones, y brindar interfaces de usuario incluso en dispositivos móviles. El sistema es capaz de generar seis tipos de alarmas para el monitoreo de variables en los procesos. Su arquitectura distribuida permite su aplicación en espacios pequeños o grandes. Su extensibilidad permite la adición de nuevos tipos de componentes gráficos a mostrar en las interfaces de usuario, además de diversos protocolos para la comunicación con dispositivos físicos. Incorpora además el almacenamiento de datos sobre variables y alarmas, para su visualización en gráficas de tendencia o tiempo real. (3)

Arex es una solución alternativa a problemas de automatización en diferentes áreas de la industria. Tiene una arquitectura modular que funciona bajo el principio Cliente-Servidor. Dentro de las principales funcionalidades que implementa se encuentran la visualización del estado de los elementos automatizados en el entorno (temperatura, presencia, humedad); el control de los mismos de manera manual, o automática por medio de rutinas de control implementadas; la persistencia de la información para análisis sobre los datos del proceso, así como también diferentes gráficas para cada variable. (4)

Arex se puede usar en:

- ✓ Locales inteligentes, donde deban preservarse parámetros específicos de temperatura, humedad, luz, nivel del polvo, etc.
- ✓ Áreas climatizadas.
- ✓ Cuarto de control central.
- ✓ Producción de agua fría y caliente.
- ✓ Bloque energético.
- ✓ Control de válvulas en redes hidráulicas.
- ✓ Domótica de recintos en Hotelería y Turismo.
- ✓ Sistemas de alarmas.
- ✓ Alumbrado de áreas exteriores y públicas.
- ✓ Sistemas de control de lazo abierto o cerrado.
- ✓ Programación de eventos de encendido/apagado.
- ✓ Recolección y almacenamiento de información medible.

Arex está formado por cuatro módulos fundamentales: HMI-Edición, HMI-Ejecución, Recolector y tarjetas de adquisición de datos (figura 1). Se tienen en cuenta aspectos esenciales en este tipo de sistemas, por ejemplo el manejo de puntos y alarmas. La comunicación con las tarjetas de adquisición se realiza mediante las interfaces de red Ethernet y Serie sobre el protocolo Modbus en sus variantes TCP y RTU.

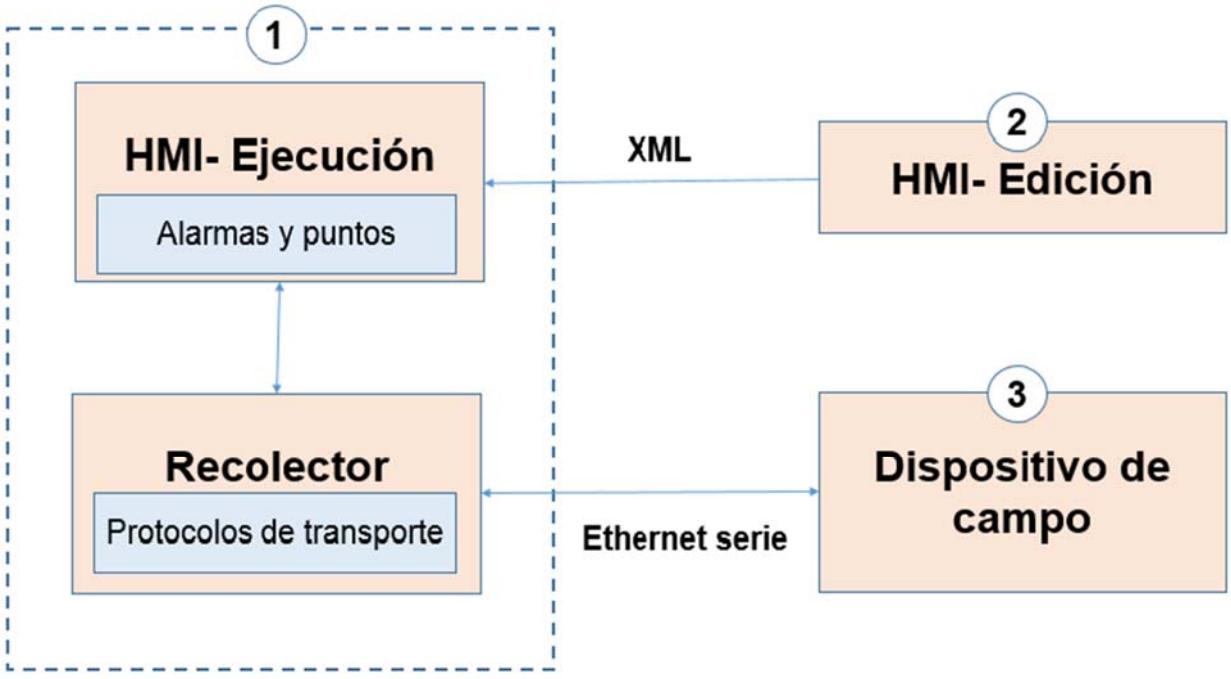


Figura 1: Arquitectura de Arex *(4)*

A continuación se describen brevemente los principales módulos de Arex:

- ✓ Recolector: Este módulo es el encargado de gestionar la comunicación con las tarjetas de adquisición de datos mediante el protocolo Modbus en sus variantes TCP y RTU. El recolector recibe la configuración de los dispositivos además de sus variables y procede mediante un manejador a la creación de bloques de encuesta, actividad que se realiza atendiendo a ciertos parámetros como son: el tipo de dato, el período de muestreo de las variables, el tamaño máximo permitido, la densidad del bloque, entre otros. Este componente de adquisición posee un planificador que permite adicionar y ejecutar tareas de lectura y de escritura sobre las tarjetas de adquisición de datos. La información de los valores de las variables se publica en un servidor de datos.
- ✓ HMI-Edición: Este módulo puede ser ejecutado en una computadora convencional, permite al usuario crear y configurar despliegues que representen lógicamente el ambiente donde va a ser desplegado el sistema. Permite además la configuración de los dispositivos, sus variables asociadas así como las alarmas. Las variables son vinculadas a elementos gráficos que forman parte de los despliegues. Finalmente las configuraciones son exportadas en formato XML.
- ✓ HMI-Ejecución: Permite la representación en tiempo de ejecución de los procesos mediante la actualización de los componentes gráficos ubicados en los despliegues. Permite además la generación de sumarios de puntos y ofrece opciones para el manejo de las alarmas generadas en el sistema. Otra de las funcionalidades soportadas es el envío de comandos de escritura, que posibilita al operador ejercer control sobre los procesos.

Actualmente en el CEDIN, Arex es el software que se utiliza en proyectos de la línea de sistemas embebidos, que tiene la función de monitorizar y controlar los datos desde la tarjeta de adquisición de datos de manera cableada.

1.1.2 Tarjeta de adquisición de datos

Las DAQT sirven para obtener una muestra de una variable física (voltaje, temperatura, nivel de sonido), es decir, toman una señal de un sensor (sistema analógico) y después la adecuan para transformarla en un dato que pueda ser reconocido y registrado por un sistema digital, con el fin de que la pueda leer una computadora y realizar una tarea mediante un software específico. (5)

Las DAQT actúan como la interfaz entre una computadora y señales físicas, es decir, la información recaudada por el sensor se pasa a la DAQT, el cual se encarga de transformar los códigos del mundo real a los códigos digitales, como si se tratara de un traductor que convierte de un lenguaje a otro, con el fin de que el sistema digital (es decir, cualquier computadora o dispositivo electrónico) sea capaz de comprender los signos del analógico. (1)

Los componentes claves para un dispositivo DAQ son:

- ✓ El acondicionamiento de señales, el cual manipula una señal de tal forma que sea apropiada para la entrada a un convertidor analógico digital. Este circuito puede incluir amplificación, atenuación, filtrado y aislamiento. Algunos dispositivos DAQ incluyen acondicionamiento de señales integrados, diseñado para medir tipos específicos de sensores.
- ✓ El convertidor analógico digital o digital analog converter (ADC) se basa en un chip que proporciona una representación digital de una señal analógica en un instante de tiempo. En la práctica, las señales analógicas varían continuamente con el tiempo y un ADC realiza "muestras" periódicas de la señal a una razón predefinida. Estas muestras son transferidas a una PC a través de un bus, donde la señal original es reconstruida desde las muestras en software.
- ✓ El bus de PC que sirve como la interfaz de comunicación entre el dispositivo de adquisición de datos y la PC para pasar instrucciones y datos medidos. Los dispositivos de adquisición de datos se ofrecen en los buses de PC más comunes, incluyendo USB, PCI, PCI Express y Ethernet. (6)

En los últimos años, los dispositivos de adquisición de datos han llegado a estar disponibles para 802.11 WiFi para comunicación inalámbrica.

Figura 2: Esquema de adquisición de datos

1.1.3 Arduino

Arduino es una plataforma de hardware libre, ampliamente conocida porque facilita el diseño de proyectos electrónicos, se caracteriza por su sencillez y bajo costo; además de su versatilidad para ser programada, contando con un software de desarrollo cuyo lenguaje de programación está basada en C++. (7)

Arduino se enfoca en acercar y facilitar el uso de la electrónica y programación de sistemas embebidos en proyectos multidisciplinarios. Toda la plataforma, tanto para sus componentes de hardware como de

software, son liberados con licencia de código abierto que permite libertad de acceso a ellos. El hardware consiste en una placa de circuito impreso con un microcontrolador, puertos digitales y analógicos de entrada/salida, los cuales pueden conectarse a placas de expansión, que amplían las características de funcionamiento de la placa Arduino. Asimismo, posee un puerto de conexión USB desde donde se puede alimentar la placa y establecer comunicación con el computador. La primera placa Arduino fue introducida en 2005, ofreciendo un bajo costo y facilidad de uso para novatos y profesionales; buscaba desarrollar proyectos interactivos con su entorno mediante el uso de actuadores y sensores. (8)

Tabla 1: características técnicas de Arduino

Microcontrolador	Atmega328
Voltaje de operación	5V
Voltaje de entrada (recomendado)	7V- 12V
Voltaje de entrada (límite)	6V- 20V
Pines para entrada – salida digital	14 (6 pueden usarse como salida de PWM)
Pines de entrada analógica	6
Corriente continua por pin IO	40 mA
Corriente continua en el pin 3.3V	50 mA
Memoria flash	32 KB (0.5 KB ocupados por el bootloader)
SRAM	2 KB
EEPROM	1 KB
Frecuencia de reloj	16 MHz

Las ventajas de Arduino son:

- ✓ Barato: Las placas Arduino son relativamente baratas comparadas con otras plataformas microcontroladoras. La versión menos cara del módulo Arduino puede ser ensamblada a mano, e incluso los módulos de Arduino preensamblados cuestan menos de 50$.
- ✓ Multiplataforma: El software de Arduino llamado Genuino se ejecuta en sistemas operativos Windows y GNU/Linux.
- ✓ Limitaciones: Arduino es una compañía que no tiene limitaciones ya que existe una gran variedad de placas las cuales van cubriendo las necesidad que vaya exigiendo el cliente ya sean propias y de compañías ajenas pero que se puedan adaptar. (9)

- ✓ Código abierto y software extensible: El software Arduino está publicado como herramientas de código abierto, disponible para extensión por programadores experimentados.

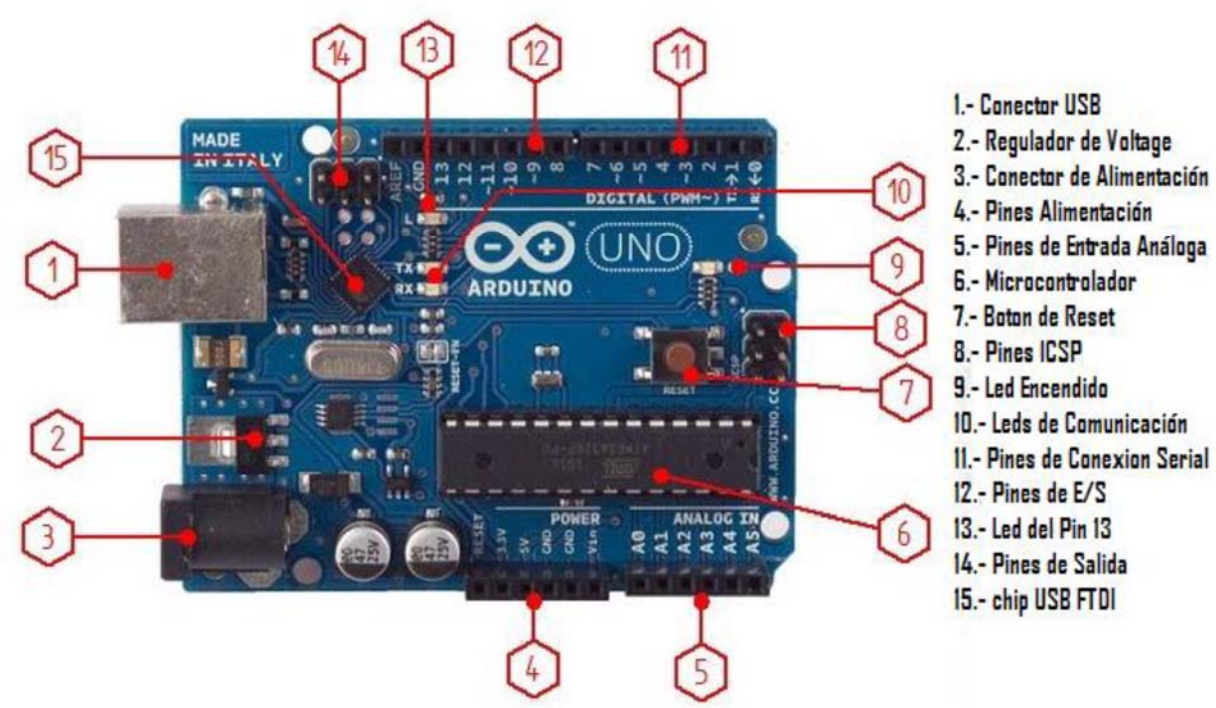


Figura 3: Estructura Hardware de Arduino

El software de Arduino es propio de la compañía Genuino y es un software libre ya que puede ser expandido por medio de librerías de C++ y que se puede descargar de cualquier navegador y puede instalarlo en cualquier Sistema Operativo que desee, lo interesante de este software es su fácil manejo y su programación y es por eso que es de fácil aprendizaje pero también a la vez puede realizar proyectos de avanzada complejidad (2).

El tipo de lenguaje de programación que utiliza es Processing que es muy similar a C++.

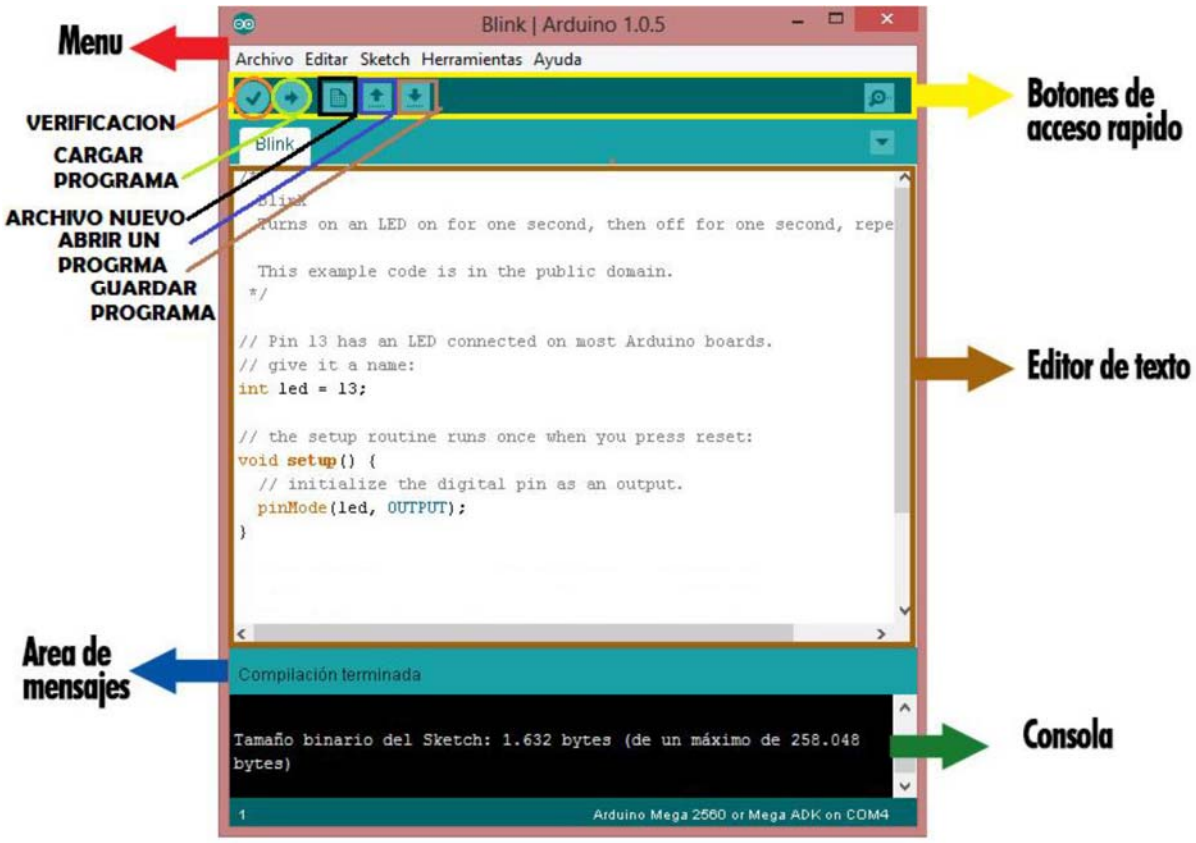


Figura 4: Software de programación Arduino

La tarjeta de adquisición de datos basada en la placa Arduino necesita ser conectada al software genérico, en este caso Arex y para esto se utilizará la comunicación inalámbrica ya que en algunos escenarios se hace engorroso el entramado de cable por la longitud a utilizar y su alto costo.

1.1.4 Comunicaciones inalámbricas

Una red inalámbrica es la interconexión de dispositivos con la capacidad de transmitir información entre ellos, pero sin la utilización de un medio físico. En la actualidad este tipo de comunicación se utiliza con frecuencia, debido a su fácil manejo sin la necesidad de un amplio conocimiento.

Antiguamente usaban módulos de radiofrecuencia, posteriormente aparecieron circuitos transmisores integrados que unificaron las funciones de emisor y receptor a través del uso de bandas de frecuencia y actualmente posee un extensor de campo de aplicación. Las tecnologías inalámbricas más utilizadas son WiFi, Bluetooth y ZigBee ya que trabajan en un rango de frecuencia no muy saturado (2).

1.1.5 Principales tecnologías inalámbricas

- ✓ Bluetooth

Bluetooth es un protocolo de comunicaciones que sirve para la transmisión inalámbrica de datos (fotos, música, contactos…) y voz entre diferentes dispositivos que se encuentran a corta distancia, dentro de un radio de alcance que, generalmente, es de diez metros.

El uso del Bluetooth se ha asociado a los teléfonos móviles, ya que éstos fueron de los primeros dispositivos en incorporar el protocolo. Sin embargo, esta tecnología inalámbrica se encuentra presente, hoy en día, en smartphones, tablets, portátiles, ratones, teclados, impresoras, auriculares, televisores, cámaras digitales, reproductores MP3 o videoconsolas.

La tecnología Bluetooth transmite inalámbricamente datos y voz a través de ondas de radio. Para ello, hace uso de las Redes Inalámbricas de Área Personal (WPAN, por sus siglas en inglés). Al realizarse la transferencia por radiofrecuencia, los dispositivos no tienen la obligación de hallarse alineados; sin embargo, los equipos deben encontrarse dentro de un radio de alcance, que suele ser corto, aunque pueda variar en función del dispositivo.

- ✓ WiFi

WiFi es una abreviación de la marca comercial *Wireless Fidelity* y se trata de un tipo de tecnología de comunicación inalámbrica que permite conectar a internet equipos electrónicos, como por ejemplo, computadoras, Tablet, Smartphone o celulares; WiFi logra esto mediante el uso de radiofrecuencias o infrarrojos para la trasmisión de la información. (10)

Las redes WiFi permiten la conectividad de equipos y dispositivos mediante ondas de radio; existen distintos estándares que se han ido implementando con el paso del tiempo, con el objetivo de mejorar la conectividad y su rendimiento. Poseen características diferentes como la frecuencia que usan, el ancho de banda, la velocidad y el alcance o rango. En los dispositivos casi siempre existe compatibilidad con los estándares anteriores y un adaptador inalámbrico aunque admita varios estándares, siempre va a escoger y usar de ser posible el que más velocidad permita. (11)

Todas las mejoras recientes tratan de evitar la popular frecuencia de la banda de 2.4 GHz debido a que está muy congestionada debido a varios dispositivos que la usan como: equipos de microonda, teléfonos inalámbricos, cámaras de seguridad, entre otros. (11)

Entre las características esenciales de la red WiFi se encuentra:

- ✓ El punto de acceso: dispositivo que permite comunicar todos los elementos de la red con el router. Cada punto de acceso tiene un máximo de 90 metros en entornos cerrados; en lugares abiertos puede ser hasta tres veces superior.
- ✓ Tarjeta de Red Wireless*:* permite al usuario conectarse en su punto de acceso más próximo.
- ✓ Router: permite conectar un punto de acceso a Internet.

Tabla 2: Características de estándares WiFi

Estándar	Velocidad (Teórica)	Velocidad (Práctica)	Frecuencia	Ancho de Banda	Alcance (m)	Detalles	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	Aceptación Internacional	1999
802.11g	54Mbit/s	22Mbit/s	2,4GHz	20MHz	460	Aceptación Internacional	2003
802.11n	600Mbit/s	100Mbit/s	2,4GHz y 5,4 GHz	20/40MHz	820	Mayormente utilizado, se puede configurar su ancho de banda.	2009
802.11ac	6.93Gbit/s	100Mbit/s	5,4GHz	80hasta 160MHz		Nuevo estándar sin interferencia pero menor alcance.	2013
802.11ad	7.13Gbit/s	Hasta6Gbit/s	60GHz	2MHz	300		2012
802.11ah			0,9GHz		1000	Fácil distribución en áreas rurales, menor consumo de energía	2016

Como se muestra en la tabla 1 con el paso de los años los estándares de comunicaciones WiFi han evolucionado notablemente en cuanto a velocidad, alcance y reconocimiento internacional, siendo una solución en la actualidad a problemas existentes en el mundo de las redes y la informática.

- ✓ Zigbee

ZigBee es un estándar de comunicaciones inalámbricas creado por la ZigBee Alliance. ZigBee está basado en el estándar IEEE 802.15.4 de redes inalámbricas de área personal y tiene como objetivo principal extender la vida útil de sus baterías, es decir usando un bajo nivel de envió de datos. Por ello podemos decir que es ideal para las redes domóticas, evitando el exceso de sensores y actuadores. Además que posee un bajo costo para red Wireless de información pequeña, seguro y confiable.

La mayor ventaja de esta tecnología es que reduce el tiempo de espera en la transmisión, es decir, en el envío y recepción de paquetes, por lo que lo convierte en la tecnología ideal para redes de baja tasa de transferencia de datos.

A continuación se presenta en la tabla 3 un resumen de comparación de las tecnologías estudiadas en cuanto a ancho de banda, alcance, potencia y aplicaciones.

Tabla 3: comparativa de tecnologías de comunicación inalámbrica

Tecnología	Ancho de banda	Alcance	Potencia	Aplicaciones
Bluetooth	1-24 Mbps	1m	1mW	Auriculares inalámbrico. Red inalámbrica en espacios reducidos.
		10m	10mW	
		100m	100mW	
WiFi	54-400 Mbps	Hasta 100m		Conexión inalámbrica a internet.
Zigbee	20-250 Kbps	10-100m		Redes inalámbricas de sensores.

Después de estudiar las tecnologías, sus ventajas y desventajas, se seleccionó la tecnología WiFi debido a que:

- ✓ Cualquier dispositivo puede conectarse desde distintos puntos dentro de un espacio bastante amplio.
- ✓ En cualquier parte del mundo se podrá utilizar la tecnología WiFi con una compatibilidad absoluta.
- ✓ Permite conexiones rápidas.

1.1.6 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)[2]; convertido en un protocolo de comunicaciones

[2] PLC: dispositivos electrónicos muy usados en Automatización Industrial. Un PLC controla la lógica de funcionamiento de máquinas, plantas y procesos industriales, procesan y reciben señales digitales y analógicas y pueden aplicar estrategias de control.

estándar en la industria; es además, el que goza de mayor disponibilidad para la conexión de dispositivos electrónicos industriales. (12)

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 permite el control de una red de dispositivos, por ejemplo un sistema de medida de temperatura y humedad, y comunicar los resultados a un ordenador. Modbus también se usa para la conexión de un ordenador de supervisión con una unidad remota (RTU) en sistemas de supervisión, control y adquisición de datos (SCADA). Existen versiones del protocolo Modbus para puerto serie y Ethernet (Modbus/TCP).

Cada dispositivo de la red Modbus posee una dirección única. Cualquier dispositivo puede enviar órdenes Modbus, aunque lo habitual es permitirlo sólo a un dispositivo maestro. Cada comando Modbus contiene la dirección del dispositivo destinatario de la orden. Todos los dispositivos reciben la trama pero sólo el destinatario la ejecuta (salvo un modo especial denominado "Broadcast"). Cada uno de los mensajes incluye información redundante que asegura su integridad en la recepción. Los comandos básicos Modbus permiten controlar un dispositivo RTU para modificar el valor de alguno de sus registros o bien solicitar el contenido de dichos registros. (12)

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). 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). (12)

- ✓ Modbus over Ethernet

Modbus over Ethernet es un tipo de protocolo Modbus diseñado para controlar y supervisar equipos a través de TCP o 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. (13)

- ✓ Modbus over Wireless

Una red Modbus sobre Wireless 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. (14)

1.2 Análisis del estado del arte

El estado del arte es una investigación documental que tiene como objetivo recuperar y trascender el conocimiento acumulado sobre un objeto de estudio específico. Además posibilita la compresión crítica sobre el conocimiento de un fenómeno con el fin de generar nuevos conocimientos y comprensiones.

1.2.1 Estudio de factibilidad de costos

Para demostrar que la comunicación inalámbrica WiFi supone un menor costo que la comunicación cableada utilizada actualmente, se muestra un estudio de los gastos para estas dos comunicaciones.

Tabla 4: Estudio de costos

	Cableado	**Inalámbrico (WiFi)**
Arduino	13.30 cuc	13.30 cuc
Tarjeta Ethernet	9.80 cuc	
Metro de cable (10 metros (m))	(0.25 cuc*m) 2.50 cuc	
Canaletas (10 metros (m))	(1.40 cuc*m) 14 cuc	
Mano de obra (2 personas (p))	(20 cuc*p) 40 cuc	
Switch	21.30 cuc	
Router		21.90 cuc
Computadora	300 cuc	300 cuc
Tarjeta WiFi		10.20 cuc
Access Point		20.10 cuc
Total	**400.90 cuc**	**365.50 cuc**

El estudio anterior se realizó a un local del CEDIN, donde se puede concluir que la comunicación mediante una red inalámbrica reduce el costo en 35.40 cuc con respecto a la red cableada. Teniendo en cuenta que el CEDIN cuenta con seis laboratorios productivos el gasto sería varias veces mayor y ascendería considerablemente llegando a alcanzar un monto de 212.40 cuc. Esto supone una mejora necesaria que responde a las necesidades planteadas en la problemática de la presente investigación.

1.2.2 Estudio de soluciones similares

T7-Pro

LabJack produce un producto de control y adquisición de datos de bajo costo y alta calidad capaz de establecer comunicación inalámbrica (802.11b/g WiFi) llamado T7-Pro. El T7-Pro es un dispositivo de DAQ, que puede controlarse mediante una conexión inalámbrica 802.11b/g; presenta alta calidad y totalmente compatible con Modbus TCP y la mayoría de los programas de SCADA. Contiene:

- ✓ 14 entradas analógicas incorporadas,
- ✓ 84 entradas analógicas con el complemento Mux80,
- ✓ rangos de entrada analógica: ± 10V, ± 1V, ± 0.1V y ± 0.01V,
- ✓ ADC de alta velocidad de 16 bits,
- ✓ ADC de baja velocidad de 24 bits,
- ✓ 10 contadores y
- ✓ 2 salidas analógicas (12 bits, 0-5 V).

Los precios de adquisición para esta tecnología oscilan entre los 480.50 y 592.50 cuc por lo que se evidencia que es muy costoso.

USB-AIO

La USB-AIO ofrece una solución ideal para agregar capacidades de entrada y salida (E/S) analógicas y digitales (A/D) de alta velocidad y fáciles de instalar a cualquier PC o sistema integrado con un puerto USB. Estas unidades son dispositivos USB 2.0, que ofrecen la mayor velocidad disponible en el bus. El USB-AIO16-16F es una tarjeta A/D de resolución de 16 bits capaz de velocidades de hasta 1MHz (en sistemas operativos en tiempo real) para sus 16 entradas analógicas unipolares (16 pseudo-diferencial opcional) u 8 entradas analógicas diferenciales. Cada canal puede configurarse independientemente por software para aceptar 8 rangos de entrada diferentes. Las características adicionales incluyen 16 líneas de E/S digitales y 0, 2 o 4 salidas analógicas. Estas unidades de E / S le brindan al usuario todo lo necesario para comenzar a adquirir, medir, analizar y monitorear en una variedad de aplicaciones. Las tarjetas de adquisición de datos de la serie USB-AIO se pueden utilizar en muchas aplicaciones actuales del mundo real, como medición de precisión, análisis, supervisión y control en innumerables aplicaciones integradas. (15)

Por otra parte los precios de adquisición para la tecnología USB-AIO se encuentran en un rango entre los 418.80 y 984.40 cuc presentando costos considerablemente altos.

Como se plasma en el estudio de factibilidad de costos realizado en el epígrafe anterior el precio de una tarjeta de adquisición de datos basada en Arduino es de solo 13.30 cuc. Teniendo en cuenta esto, se puede apreciar que es ampliamente viable económicamente para el CEDIN la utilización de las DAQT que utilizan Arduino; siendo esta una razón de peso por la que se decide utilizar en dicho centro y en la presente investigación esta tecnología.

1.2.3 Capa de acceso inalámbrico

Una red de área local inalámbrica puede definirse como una red de alcance local que tiene como medio de transmisión el aire. Es una red que 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. Estos enlaces se implementan básicamente a través de tecnologías de microondas y de infrarrojos. (11)

1.2.4 Shield Arduino

Un shield es una placa impresa que se puede conectar en la parte superior de la placa Arduino para ampliar sus capacidades, pudiendo ser introducidas una encima de la otra. Los shields suelen ser diseños bastante simples y en general de código abierto. “No es más que una placa electrónica que puede ser conectada en la parte superior del Arduino y que normalmente permite conectar más placas encima de ella y que incorpora una determinada funcionalidad”. (16)

Arduino Ethernet Shield

Con este tipo de Shield se podrá dotar una conexión LAN permitiéndole obtener internet a una placa Arduino utilizando el conector Rj-45 con el protocolo TCP/IP, su forma es muy semejante al Arduino Uno (17). Sus características son:

- ✓ Microcontrolador Atmega 328
- ✓ Frecuencia 16 MHz
- ✓ 5 voltios
- ✓ 2kb SRAM
- ✓ 1kb de EEPROM
- ✓ 32kb de flash
- ✓ 1 ranura de micro-sd
- ✓ 14 puertos digitales

Arduino WiFi Shield

Este dispositivo realiza el mismo funcionamiento que el shield Ethernet pero su conexión lo hace de forma inalámbrica y así de esta forma se conectaría a una red de acceso de internet por medio de una red WiFi utilizando el estándar 802.11 b/g (17). Tiene como características:

- ✓ Voltaje de 3.3 a 5.4
- ✓ Potencia de transmisión 0-10 dbm
- ✓ Frecuencia de transmisión 2401-2480
- ✓ Canal de transmisión 0-13 canales

1.3 Metodología, tecnologías y herramientas

Para el desarrollo de la propuesta de solución se hace necesaria la definición de las herramientas, tecnologías y metodología que se emplearán, las que estarán en correspondencia con las usadas en el CEDIN.

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 tabla se presenta una comparación entre los dos tipos de metodologías.

Tabla 5: Comparación 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.
- ✓ 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. (18)

1.3.2 Tecnologías y Herramientas

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. (19)

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. (20)

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. (20)

1.3.4 Herramienta 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. (21)

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. (22)

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. (23)

1.3.5 Lenguaje 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. (24)

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++ (25).

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++ (25).

1.3.6 Entorno integrado de desarrollo

Un entorno de desarrollo integrado o *Integrated Development Environment* (IDE), 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. (26)

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.

Normalmente, un IDE está dedicado a un determinado lenguaje de programación. No obstante, las últimas versiones de los IDEs tienden a ser compatibles con varios lenguajes (por ejemplo, Eclipse, NetBeans, Microsoft Visual Studio...) mediante la instalación de plug-ins adicionales (27).

Arduino IDE

Arduino es un entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital. El software de Arduino se ejecuta en diferentes sistemas operativos como puede ser: Windows, Macintosh, GNU/Linux. (28)

1.4 Propuestas de las tecnologías y herramientas a utilizar

- ✓ 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 estructurada de los lenguajes de alto nivel con la flexibilidad del ensamblador.
- ✓ Como entorno de desarrollo Arduino porque es fácil de usar para principiantes, pero suficientemente flexible para que usuarios avanzados puedan aprovecharlo también.

1.5 Conclusiones Parciales

Tomando como punto de partida el estudio realizado previamente sobre los principales conceptos asociados al dominio del problema, se pudo seleccionar la WiFi como tecnología más apropiada para realizar la comunicación inalámbrica. Luego de un análisis del estado del arte de varias tecnologías usadas en dispositivos de control y adquisición de datos se decide la utilización de una DAQT basada en Arduino por su bajo costo de adquisición. La realización de un estudio de factibilidad de costos permitió evidenciar la necesidad para el CEDIN de sustituir las redes cableadas por las comunicaciones inalámbricas vía WiFi para el intercambio de información con un sistema recolector de datos. Por estas razones se decide desarrollar una capa de acceso inalámbrica para la comunicación de un servidor Modbus basado en Arduino.

Todo el proceso será guiado por la metodología de desarrollo de software AUP en su versión UCI, empleando el lenguaje de programación C++ con el entorno de desarrollo Arduino. Además la herramienta Visual Paradigm 8.0 utilizando el lenguaje de modelado UML 6.4 para generar los diferentes artefactos propuestos por la metodología seleccionada.

Capítulo II: Propuesta de solución

Introducción

En este capítulo se expondrán las características principales de la solución a desarrollar. 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 a aplicar y los diagramas de clases que se utilizarán, logrando un mayor entendimiento para el desarrollo del sistema.

2.1 Propuesta del sistema

Para ilustrar la solución que se propone se usará un ejemplo de la vida real. Suponiendo que el Centro de Informática Industrial desea implementar un local inteligente donde mediante un sensor se deba conocer cuando llega al local una persona para así poder ejecutar alguna acción como por ejemplo encender un bombillo. Para esta situación se debe contar en el local con un sensor de presencia conectado a una tarjeta de adquisición de datos basada en Arduino. El sensor será capaz de al llegar un individuo al local transmitirle esta información a la tarjeta de adquisición de datos, esta a su vez traducirá la información en un dato que pueda ser reconocido por un sistema digital. En el local deberá existir una computadora con un software para la adquisición, monitoreo y control de datos; en este caso el sistema Arex, desarrollado por dicho centro. La tarjeta le enviará el dato recolectado al sistema y este lo procesa para ejecutar la acción deseada. Se propone implementar para esto una capa de acceso inalámbrico y de bajo costo mediante el dispositivo WiFi y el protocolo de comunicación *Modbus over Wireless* que es el encargado de las redes inalámbricas.

El firmware de Arduino está conformado por dos capas, la capa de comunicación y el driver Modbus. Para lograr el acceso a la tarjeta de adquisición desde el software genérico Arex se debe establecer una comunicación entre las capas a las que se hizo alusión anteriormente. Por tanto la solución propuesta en la presente investigación se enfocará en establecer dicha comunicación.

En la capa de comunicación se incluirá todas las configuraciones y funciones necesarias para lograr establecer el intercambio de información con un sistema recolector de datos. Para ello se debe configurar el módulo WiFi ESP8266 mediante la utilización de comandos AT. Se establecerá el modo *access point* especificando el SSID y contraseña mediante los que se podrá acceder por vía WiFi a la DAQT basada en Arduino. Por temas de seguridad solo se aceptará una sola conexión. Por último en esta capa se configurará un servidor TCP por el puerto 502 que es el puerto por defecto para el protocolo Modbus.

Por otra parte en la capa de driver Modbus se establecerán las configuraciones necesarias para la utilización del servidor Modbus el cual estará en espera de una solicitud de información por parte de un sistema recolector de datos. Como parte de la implementación del protocolo se desarrollará el mapa de memoria con cuatro variables (*coils, input, input register, holding register*). Luego se pasará a asociar cada dirección de memoria con el valor de un sensor y un actuador para así poder ejecutar dicho

protocolo. La capa de comunicación maneja datos TCP pero para el envío al protocolo realiza una conversión a Serie, de esta manera Modbus lo recibe y realiza nuevamente la conversión a TCP.

Figura 5: Propuesta solución

2.2 Requerimientos del sistema

La especificación de requisitos de software es una descripción completa del comportamiento del sistema que se va a desarrollar. “Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema.” (29)

Para que el desarrollo de software tenga éxito, es esencial comprender perfectamente los requisitos del software. Independientemente de lo bien diseñado o codificado que esté un programa, si se ha analizado y especificado pobremente, no logrará la satisfacción del cliente. “La parte más difícil en la construcción de sistemas de software es decidir precisamente qué construir. Ninguna otra parte del trabajo conceptual es tan ardua como establecer los requisitos técnicos detallados, incluyendo todas las interfaces con humanos, máquinas y otros sistemas. Ninguna otra parte del trabajo puede perjudicar tanto el resultado final si se realiza de forma errónea. Ninguna otra parte es tan difícil de rectificar posteriormente.” (30)

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 (31).

- ✓ **Requerimientos de hardware:** Arduino debe contar con un microcontrolador Atmega328, SRAM de 2 kb y memoria EEPROM 1 kb.

- ✓ **Requerimientos de restricciones del diseño e implementación**: El sistema debe ser desarrollado en el lenguaje de programación C++.

Los **requerimientos funcionales** son declaraciones que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas y de cómo se debe comportar en situaciones particulares. Además describen con detalle la función de éste, sus entradas, sus salidas y excepciones (32)

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 (33).

Tabla 6: Historia de usuario 1

Historia de usuario	
Número:1	**Nombre del requisito:** Establecer comunicación en la tarjeta de adquisición de datos.
Programador: Jessica Bárbara Monja Vizcaíno	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 1
Riesgo en desarrollo: Alto	**Tiempo real:** 1
Observaciones: -	
Descripción: Esta funcionalidad permite configurar el modo de comunicación del Arduino (WiFi).	
Prototipo elemental de interfaz de usuario: N/A	

Tabla 7: Historia de usuario 2

Historia de usuario

Número:2	**Nombre del requisito:** Recibir configuración.
Programador: Jessica Bárbara Monja Vizcaíno	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 3
Riesgo en desarrollo: Alto	**Tiempo real:** 3
Observaciones: Se debe haber establecido la comunicación en la DAQT	
Descripción: Esta funcionalidad permite recibir desde un nodo la información de una nueva configuración enviada por el cliente que contendrá nombre de la tarjeta, valor de los sensores y registro de Modbus.	
Prototipo elemental de interfaz de usuario: N/A	

Tabla 8: Historia de usuario 3

Historia de usuario	
Número:3	**Nombre del requisito:** Enviar configuración.
Programador: Jessica Bárbara Monja Vizcaíno	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 3
Riesgo en desarrollo: Alto	**Tiempo real:** 3
Observaciones: se debe haber establecido comunicación en la DAQT	
Descripción: Esta funcionalidad permite enviar desde el Arduino la información de la configuración.	
Prototipo elemental de interfaz de usuario: N/A	

Tabla 9: Historia de usuario 4

Historia de usuario	
Número:4	**Nombre del requisito:** Modificar configuración.
Programador: Jessica Bárbara Monja Vizcaíno	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 3
Riesgo en desarrollo: Alto	**Tiempo real:** 3
Observaciones: Se debe haber establecido comunicación en la DAQT	
Descripción: Esta funcionalidad será capaz de modificar una configuración en el Arduino cuando el cliente la modifique.	
Prototipo elemental de interfaz de usuario:	

Tabla 10: Historia de usuario 5

Historia de usuario	
Número:5	**Nombre del requisito:** Salvar configuración en memoria EEPROM
Programador: Jessica Bárbara Monja Vizcaíno	**Iteración asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 2
Riesgo en desarrollo: Alto	**Tiempo real:** 2
Observaciones: Se debe haber establecido comunicación en la DAQT	
Descripción: Esta funcionalidad permite salvar la información de una configuración en memoria EEPROM del Arduino.	
Prototipo elemental de interfaz de usuario: N/A	

Tabla 11: Historia de usuario 6

Historia de usuario	
Número:6	**Nombre del requisito:** Cargar configuración de fábrica.
Programador: Jessica Bárbara Monja Vizcaíno	**Iteración asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 3
Riesgo en desarrollo: Alto	**Tiempo real:** 3
Observaciones: Se debe haber establecido comunicación en la DAQT	
Descripción: Esta funcionalidad permite cargar la información de una configuración desde la memoria EEPROM del Arduino.	
Prototipo elemental de interfaz de usuario: N/A	

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 12: Estimación de esfuerzo

Historias de usuario	**Puntos estimados**
Establecer comunicación en la DAQT	2
Recibir configuración	3
Enviar configuración	3
Modificar configuración	3
Salvar configuración en memoria EEPROM	3
Cargar configuración de fábrica	2

2.4 Plan de entregas

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

Tabla 13: Plan de entregas

Plan de entregas				
Capa de acceso inalámbrico para la comunicación de un servidor Modbus basada en Arduino				
Fecha de reunión de planificación		2-12-2017		
Nombre del documentador		Jessica Bárbara Monja Vizcaíno		
Historias de usuario a implementar				
No. HU	**Título**	**Prioridad**	**Fecha en la que se hará entrega**	**Liberación en la que será incluida**
1	Establecer comunicación en la DAQT	alta	26-3-2018	1
2	Recibir configuración	alta	16-4-2018	2
3	Enviar configuración	alta	27-4-2018	2
4	Modificar configuración	alta	6-5-2018	2
5	Salvar configuración en memoria EEPROM	alta	22-5-2018	3
6	Cargar configuración de fábrica	alta	26-5-2018	3

2.5 Plan de iteraciones

Iteración 1

Tabla 14: Iteración 1

Historia de usuario	**Tiempo estimado**	**Iteración asignada**	**Tiempo real**
Establecer comunicación en la DAQT	2	1	2
Recibir configuración	3	1	3
Enviar configuración	3	1	3
Modificar configuración	3	1	3

Iteración 2

Tabla 15: Iteración 2

Historia de usuario	Tiempo estimado	Iteración asignada	Tiempo real
Salvar configuración en memoria EEPROM	3	2	3
Cargar configuración de fábrica	3	2	3

2.6 Modelo Arquitectónico

La arquitectura del software de un programa o sistema de cómputo es la estructura o las estructuras del sistema, que incluyen los componentes del software, las propiedades visibles externamente de esos componentes y las relaciones entre ellos (30).

2.6.1 Estilo arquitectónico

Un estilo arquitectónico define un conjunto de principios que le dan forma o rigen el diseño del sistema a desarrollar. Estos principios van encaminados a cómo interactúan y están estructurados los componentes o módulos del sistema, así como las responsabilidades de cada uno. La arquitectura de un sistema de software casi nunca está atada a un solo estilo arquitectónico, sino que es más bien una combinación de estos estilos los que dan como resultado la arquitectura final del sistema (29).

Para el desarrollo de un software se utilizan de uno a varios estilos arquitectónicos. Cada estilo describe una categoría de sistemas, que abarca:

- ✓ Un conjunto de componentes (por ejemplo, una base de datos, módulos computacionales) que realizan una función requerida por el sistema.
- ✓ Un conjunto de conectores que permiten la comunicación, coordinación y cooperación entre los componentes.
- ✓ Restricciones que definen como se integraran los componentes para formar el sistema.
- ✓ Modelos semánticos que permiten a un diseñador, mediante el análisis de las propiedades conocidas de las partes que lo integran, comprender las propiedades generales de un sistema. (34)

Con el fin de establecer una estructura para todos los componentes del sistema se escoge el estilo arquitectónico Orientado a objeto.

La arquitectura orientada a objetos se basa en que:

- ✓ Los componentes de un sistema encapsulan los datos y las operaciones que se deben realizar para manipular los datos.
- ✓ La comunicación y la coordinación entre componentes se consiguen a través del paso de mensaje.

- ✓ La representación de los datos y sus operaciones primitivas asociadas son encapsuladas en un tipo de dato abstracto u objeto.

Ventajas de la arquitectura orientada a objetos:

- ✓ Como un objeto oculta su representación a sus clientes, es posible cambiar su implementación sin modificar los clientes
- ✓ La integración de un conjunto de rutinas de acceso con los datos que manipulan permite a los diseñadores descomponer los problemas en colecciones de agentes que interactúan.

Desventajas de la arquitectura orientada a objetos:

- ✓ Para que un objeto interactúe con otro (mediante la invocación a un procedimiento) debe conocer la identidad del otro objeto. Luego, cuando la identidad de un objeto cambie es necesario modificar todas las invocaciones a tal objeto.
- ✓ Se pueden presentar efectos laterales: si los objetos A y C usan al objeto B, entonces los efectos de C en B lucen como efectos laterales no esperados en A, y viceversa.

2.6.2 Patrón arquitectónico

Un patrón arquitectónico, al igual que un estilo, impone una transformación en el diseño de una arquitectura. Sin embargo, un patrón difiere de un estilo en varios elementos fundamentales:

- ✓ El alcance de un patrón es menor, ya que se concentra en un aspecto en lugar de hacerlo en toda la arquitectura.
- ✓ Un patrón impone una regla sobre la arquitectura, pues describe la manera en que el software maneja algún aspecto de su funcionalidad al nivel de su infraestructura.
- ✓ Los patrones arquitectónicos tienden a abarcar aspectos específicos del comportamiento dentro del contexto de la arquitectura (35).

Se escogió la arquitectura en capas ya que define cómo organizar el modelo de diseño en capas, que pueden estar físicamente distribuidas, lo cual quiere decir que los componentes de una capa solo pueden hacer referencia a componentes en capas inmediatamente inferiores. Este patrón es importante porque simplifica la compresión la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de los superiores. Además ayuda a identificar que puede reutilizarse, y proporciona una estructura que facilita la toma de decisiones sobre qué partes comprar y qué partes construir.

Los beneficios de trabajar con un sistema dividido en capas son:

- ✓ Se puede entender una capa como un todo, sin considerar las otras.
- ✓ Las capas se pueden sustituir con implementaciones alternativas de los mismos servicios básicos.
- ✓ Se minimizan dependencias entre capas.
- ✓ Las capas posibilitan la estandarización de servicios.

- ✓ Luego de tener una capa construida, puede ser utilizada por muchos servicios de mayor nivel.

La aplicación se divide en 2 capas lógicas distintas, cada una de ellas con un grupo de interfaces perfectamente definidas.

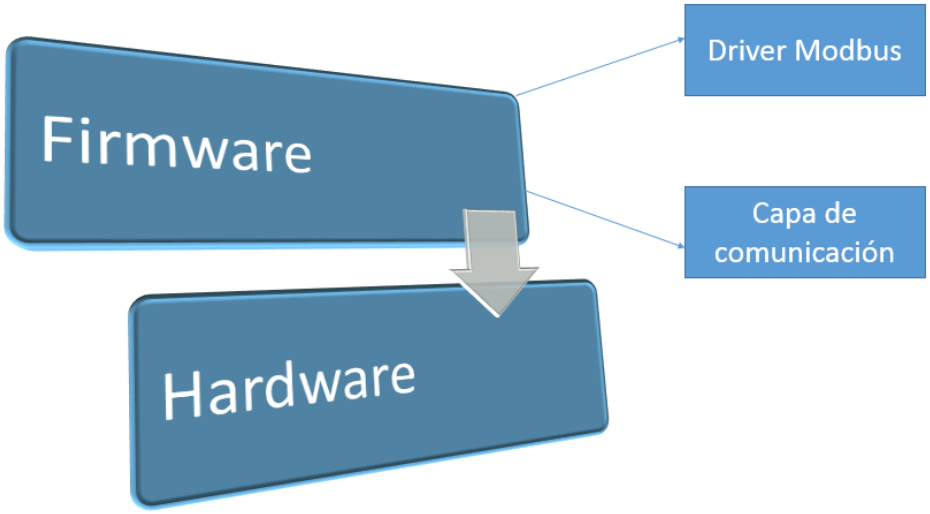


Figura 6: Capas del sistema

2.7 Patrones de diseño

Los patrones de diseño son los que comprimen el conocimiento de experiencias anteriores y pueden utilizarse para crear nuevas soluciones en contextos similares, los patrones tienen en esencia una base empírica, generalmente no son creados sino detectados, por lo que la principal fuente de patrones será tanto la aportación de los expertos como el proceso inductivo de los diseñadores.

Por lo que un patrón de diseño en el desarrollo de software es la descripción de las clases y objetos que se comunicarán entre sí de manera que puedan resolver un problema general de diseño en un contexto particular. En un contexto informático este es similar a conceptos como biblioteca de clases, frameworks, técnicas y herramientas de refactorización o programación extrema (36).

2.7.1 Patrones GRASP

En el diseño de la aplicación se usarán los patrones GRASP (*General Responsibility Assignment Software Patterns*) que son patrones generales de software para asignación de responsabilidades.

- ✓ **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. Asignara 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. Asigna las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible.

- ✓ **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.

El patrón se implementa creando en una clase un método que crea una instancia del objeto sólo si todavía no existe alguna. Para asegurar que la clase no puede ser instanciada nuevamente se regula el alcance del constructor (con atributos como protegido o privado). (37)

2.8 Diagrama de clases

Los diagramas de clases se utilizan para modelar la visión estática de un sistema. Esta visión soporta los requisitos funcionales del sistema, en concreto, los servicios que el sistema debería proporcionar a sus usuarios finales. Normalmente contienen: clases, interfaces y relaciones entre ellas: de asociación, de dependencia y/o de generalización (38).

ConfigurationAPI: Esta es la clase principal que contiene un conjunto de llamadas a las bibliotecas. Además implementa un conjunto de funciones y procedimientos que engloba el sistema.

Collector: Esta clase es la encargada de conectarse a los dispositivos de adquisición de datos y recolectar la información de los mismos.

Sensor: Esta clase hace referencia al hardware que emite una señal para captar un evento en el entorno físico.

Actuador: Esta clase hace referencia al hardware que ejecuta una acción determinada en el entorno físico.

DriverManager: Es la clase que prepara el transporte para el manejo del protocolo Modbus por la red WiFi.

Modbus: Esta clase hace referencia al protocolo Modbus por el cual se realizará la comunicación.

ModbusRTU y ModbusTCP: Son las dos variantes de comunicación del protocolo Modbus.

WiFi: Esta clase hace referencia a la biblioteca que permite configurar una red WiFi en Arduino.

DevicesShield: Esta clase hace referencia al hardware que se inserta en Arduino que permite la comunicación WiFi.

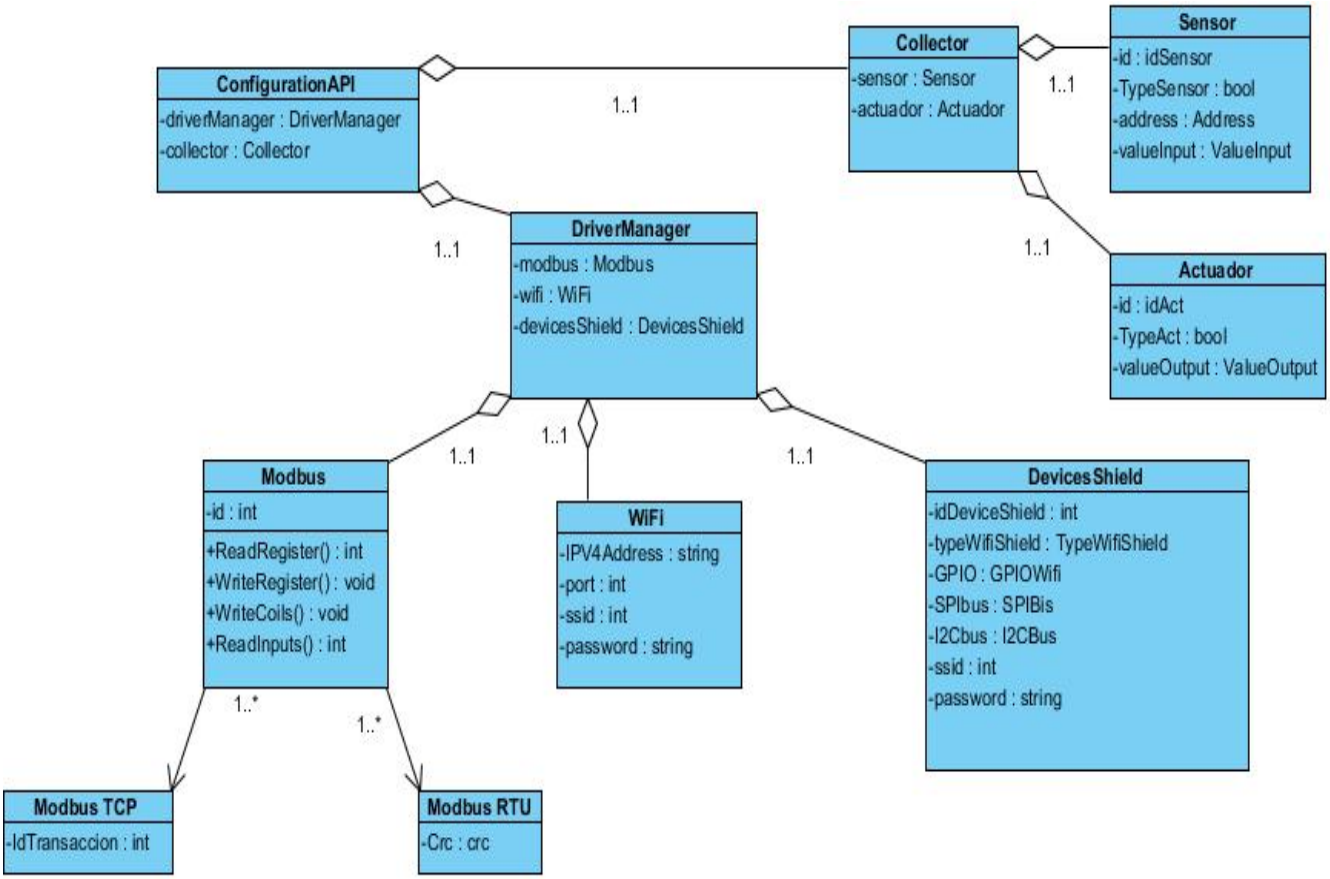


Figura 7: Diagrama de clases

2.9 Conclusiones parciales

En este capítulo quedaron definidas las características de la propuesta de solución para sentar las bases de la comprensión de la misma y de su posterior implementación. Se adoptó el patrón arquitectónico en capas quedando definida la arquitectura en dos capas. La definición de los principales requisitos funcionales y no funcionales con que debe contar la capa de acceso inalámbrica. La generación de las historias de usuario permitió hacer una descripción y especificación de los requerimientos funcionales, así como lograr una planificación en el desarrollo de las mismas. Se definieron los patrones de diseño utilizados, que junto con la selección del patrón arquitectónico seleccionado, hicieron posible 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. Los elementos y artefactos generados permiten dar comienzo a la implementación de la solución propuesta en el trabajo de una forma organizada y garantizando que el proceso se lleve a cabo con la calidad requerida.

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 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 (38).

En la construcción de la solución se emplea la notación *CamelCase y* se aplica a frases o palabras compuestas, utilizándose sus dos variantes *lowerCamelCase* y *UpperCamelCase*. A continuación se muestran algunos ejemplos del empleo de los estándares de codificación utilizados:

- ✓ Los métodos, parámetros y variables se declararon con nombres asociados a la función por la cual fueron creados. En caso de nombres compuestos, la primera palabra comienza con letra inicial minúscula y las demás comienzan con mayúscula, haciendo uso de la notación *lowerCamelCase*.

```
public:
  void configWifi(ESP8266 &wifi, String ssid, String password);
```

Figura 8: Ejemplo de nomenclatura de métodos y parámetros

- ✓ Los nombres de las clases deben comenzar con mayúscula y en caso de estar conformadas por palabras compuestas, la definición debe ser continua sin uso de guiones intermedios y cada palabra debe iniciar con mayúscula siguiendo el estilo *UpperCamelCase*.

```
class ModbusIPOverWifi
{

public:
  ModbusIPOverWifi();
  void configWifi(ESP8266 &wifi, String ssid, String password);

};
```

Figura 9: Ejemplo de la nomenclatura de las clases

3.2 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.

3.2.1 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. (27)

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.

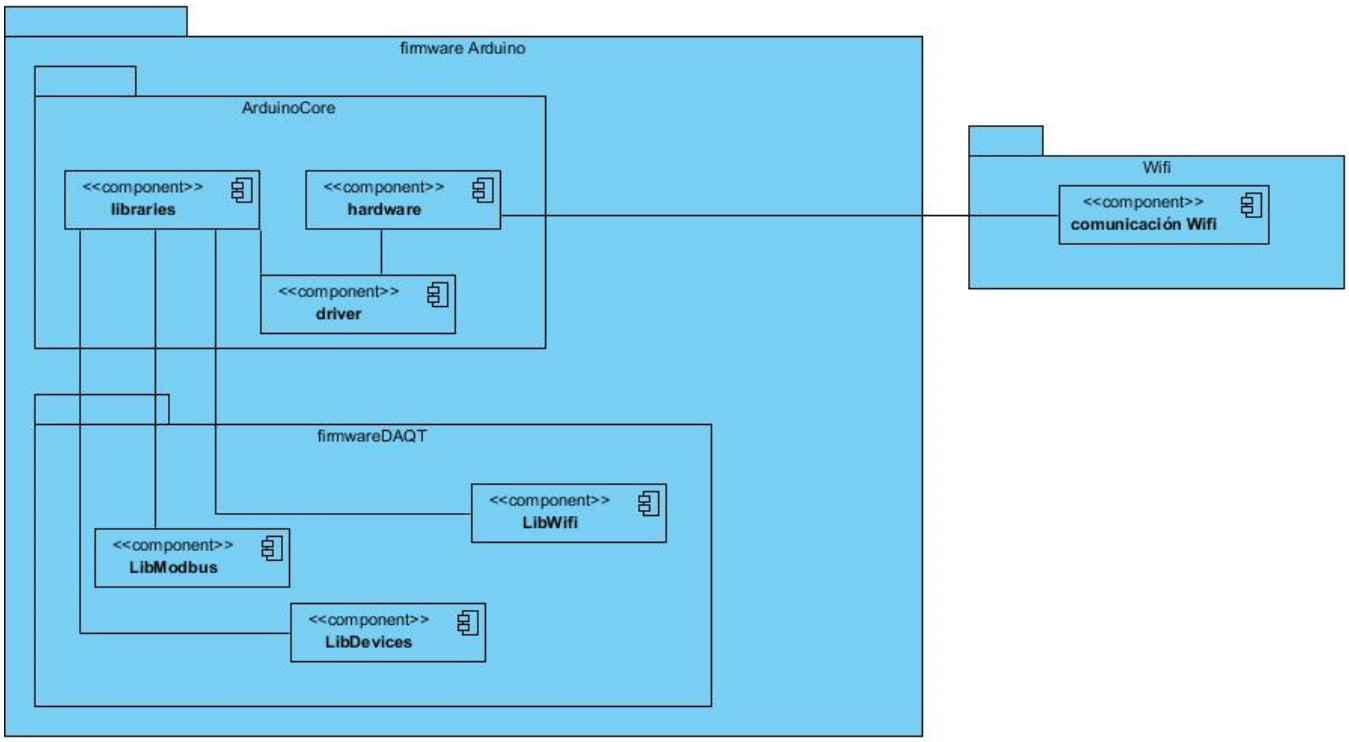


Figura 10: Diagrama de componente

3.2.2 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. (35)

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.

Figura 11: Diagrama de despliegue

3.3 Pruebas de software

Las aplicaciones (en general cualquier mecanismo diseñado e implementado por un humano) son propensas a tener fallos. A veces, pueden contribuir al fracaso de cualquier proyecto de software, e impactar de forma negativa en toda una empresa. Los tiempos de desarrollo, los entornos de programación, las diferencias entre versiones, todo influye para que, incluso con la máxima dedicación, puedan darse fallos que empañen la imagen y a veces la reputación, de una organización. Surge por tanto la necesidad de asegurar en lo posible, la calidad del producto.

Las pruebas de software son las investigaciones empíricas y técnicas cuyo fin es proporcionar información objetiva e independiente sobre la calidad del producto. Esta actividad forma parte del proceso de control de calidad global. Las pruebas son básicamente un conjunto de actividades dentro del desarrollo de software y dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento del proceso de desarrollo (40).

Las pruebas de software son un elemento crítico 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.

3.4 Ambiente de pruebas

Las pruebas se realizaron sobre los ambientes siguientes:

- ✓ Una PC
- ✓ Una tarjeta de adquisición de datos basada en Arduino

3.5 Pruebas de caja blanca

Las pruebas de caja blanca requieren del conocimiento de la estructura interna del programa y son derivadas a partir de las especificaciones internas de diseño o el código. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado.

Estas pruebas deben garantizar como mínimo que:

- ✓ Se ejerciten por lo menos una vez todos los caminos independientes para cada módulo.
- ✓ Se ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsa.
- ✓ Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- ✓ Se ejerciten las estructuras internas de datos para asegurar su validez.

La técnica de caja blanca que se usará es la técnica de camino básico, esta permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico. La idea es derivar casos de pruebas a partir de un conjunto dado de caminos independientes

por los cuales pueden circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye un diagrama de flujo asociado calculando la complejidad ciclomática.

La complejidad ciclomática de un grafo de flujo $V(G)$ establece el número de caminos independientes y se puede calcular de tres formas diferentes, las cuales se definen como:

1. $V(G) = A - N + 2$

Dónde: A es el número de aristas del grafo y N es el número de nodos.

2. $V(G) = P + 1$

Dónde: P es el número de nodos predicado contenido en el grafo G.

3. $V(G) = R$

Dónde: R es el número de regiones.

```
bool ESP8266_TCP::enableTCPServer(int port) {  \\ 1
    clearBuffer();
    write("AT+CIPSERVER=1," + String(port));  \\ 2
    debugPrintln(readData());
    String data = readData();  \\ 3
    debugPrintln(data);
    if(data.equals("no change"))  \\4
        return true;  \\ 5
    data = readData();  \\ 6
    debugPrintln(data);
    return data.equals("OK");  \\ 7
}
```

Figura 12: segmento de código de la clase WiFi

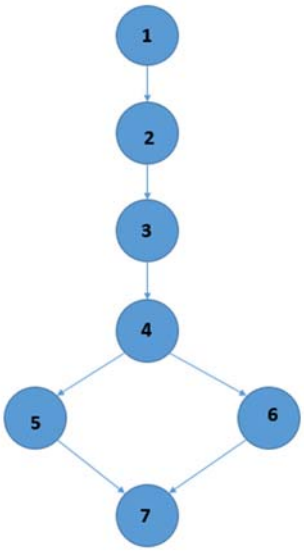


Figura 13: grafo de flujo

Caminos independientes obtenidos:

1- [1,2,3,4,5,7]
2- [1,2,3,4,6,7]

Complejidad ciclomática:

Fórmula 1:

V (G) = (A (Aristas) – N (Nodos)) + 2

V (G) = (7-7) + 2 = 2

Fórmula 2:

V (G) = P (Nodos Predicados) + 1

V (G) = 1+ 1 = 2

Fórmula 3:

V (G) = R = 2

Las pruebas de caja blanca permitieron validar el código y verificar que funcionara como estaba previsto. Estas pruebas mostraron que la aplicación presentaba piezas inútiles de código, caminos innecesarios lo que suponía una pérdida innecesaria de memoria.

3.6 Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las historias de usuario (HU), en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que las HU han sido correctamente implementadas. Las pruebas de aceptación son consideradas como “pruebas de caja negra”, los clientes son responsables de verificar que los resultados de estas pruebas sean correctos.

Las pruebas de aceptación son más importantes que las pruebas unitarias dado que significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente, por esto el cliente es la persona adecuada para diseñar las pruebas de aceptación.

3.7 Diseño de caso de pruebas

Los casos se diseñaron a partir de las historias de usuario de la herramienta de configuración. 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 herramienta de configuración:

Tabla 16: Caso de prueba de aceptación 1

Caso de prueba de aceptación
Historia de usuario: Establecer comunicación en la tarjeta de adquisición de datos.
Responsable: Jessica Bárbara Monja Vizcaíno
Descripción: Esta funcionalidad permite configurar el modo de comunicación del Arduino (Ethernet, Serie, WiFi).
Condiciones de ejecución: La tarjeta se encuentra conectada correctamente a la computadora mediante WiFi.
Resultado esperado: La aplicación se conectó con el dispositivo.
Evaluación de prueba: Prueba satisfactoria.

Tabla 17: Caso de prueba de aceptación 2

Caso de prueba de aceptación
Historia de usuario: Establecer comunicación en la tarjeta de adquisición de datos.
Responsable: Jessica Bárbara Monja Vizcaíno
Descripción: Esta funcionalidad permite configurar el modo de comunicación del Arduino (Ethernet, Serie, WiFi).
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 18: Caso de prueba de aceptación 3

Caso de prueba de aceptación
Historia de usuario: Recibir configuración
Responsable: Jessica Bárbara Monja Vizcaíno
Descripción: Esta funcionalidad permite recibir desde un nodo la información de una nueva configuración.
Condiciones de ejecución: Conexión establecida con el dispositivo.
Resultado esperado: El dispositivo recibió la configuración y esta fue aceptada por el sistema.
Evaluación de prueba: Prueba satisfactoria.

Tabla 19: Caso de prueba de aceptación 4

Caso de prueba de aceptación
Historia de usuario: Recibir configuración
Responsable: Jessica Bárbara Monja Vizcaíno
Descripción: Esta funcionalidad permite recibir desde un nodo la información de una nueva configuración.
Condiciones de ejecución: Conexión establecida con el dispositivo.
Resultado esperado: El dispositivo no recibió la configuración.
Evaluación de prueba: Prueba satisfactoria.

Tabla 20: Caso de prueba de aceptación 5

Caso de prueba de aceptación
Historia de usuario: Enviar configuración
Responsable: Jessica Bárbara Monja Vizcaíno
Descripción: Esta funcionalidad permite enviar desde el Arduino la información de la configuración.
Condiciones de ejecución: Conexión establecida con el dispositivo.
Resultado esperado: El dispositivo envió la configuración y esta fue aceptada por el sistema.
Evaluación de prueba: Prueba satisfactoria.

Tabla 21: Caso de prueba de aceptación 6

Caso de prueba de aceptación
Historia de usuario: Enviar configuración
Responsable: Jessica Bárbara Monja Vizcaíno
Descripción: Esta funcionalidad permite enviar desde el Arduino la información de la configuración.
Condiciones de ejecución: Conexión establecida con el dispositivo.
Resultado esperado: El dispositivo no envió la configuración.
Evaluación de prueba: Prueba satisfactoria.

Tabla 22: Caso de prueba de aceptación 7

Caso de prueba de aceptación
Historia de usuario: Modificar configuración
Responsable: Jessica Bárbara Monja Vizcaíno
Descripción: Esta funcionalidad permite modificar una configuración en el Arduino.
Condiciones de ejecución: Conexión establecida con el dispositivo.
Resultado esperado: Se cambian los valores de configuración de los campos.
Evaluación de prueba: Prueba satisfactoria.

Tabla 23: Caso de prueba de aceptación 8

Caso de prueba de aceptación
Historia de usuario: Modificar configuración
Responsable: Jessica Bárbara Monja Vizcaíno
Descripción: Esta funcionalidad permite modificar una configuración en el Arduino.
Condiciones de ejecución: Conexión establecida con el dispositivo.
Resultado esperado: No se cambian los valores de configuración de los campos.
Evaluación de prueba: Prueba satisfactoria.

Tabla 24: Caso de prueba de aceptación 9

Caso de prueba de aceptación
Historia de usuario: Salvar configuración en memoria
Responsable: Jessica Bárbara Monja Vizcaíno
Descripción: Esta funcionalidad permite salvar la información de una configuración en memoria EEPROM del Arduino.
Condiciones de ejecución: Conexión establecida con el dispositivo.
Resultado esperado: Los bytes salvados sean correctos.

Evaluación de prueba: Prueba satisfactoria.

Tabla 25: Caso de prueba de aceptación 10

Caso de prueba de aceptación
Historia de usuario: Salvar configuración en memoria
Responsable: Jessica Bárbara Monja Vizcaíno
Descripción: Esta funcionalidad permite salvar la información de una configuración en memoria EEPROM del Arduino.
Condiciones de ejecución: Conexión establecida con el dispositivo.
Resultado esperado: Los bytes salvados sean inválidos.
Evaluación de prueba: Prueba satisfactoria.

3.8 Ejecución de los casos de pruebas de aceptación

El proceso de pruebas a cualquier software se realiza a través de iteraciones, donde, a medida que se procede con una nueva iteración deben haberse erradicado los defectos encontrados en la anterior, para garantizar que al final del proceso el producto quede libre de la mayor cantidad de errores posibles y listo para entregar al cliente. (35)

En la primera iteración se detectaron un total de tres no conformidades (Figura 13), de ellas, una para la HU 1 (establecer comunicación en la tarjeta de adquisición de datos), una para la HU 3 (enviar configuración) y una para la HU 5 (salvar configuración en memoria EEPROM). Las no conformidades detectadas se resolvieron en la segunda iteración, donde el cliente especificó que los resultados de los casos de prueba se correspondían con los resultados esperados.

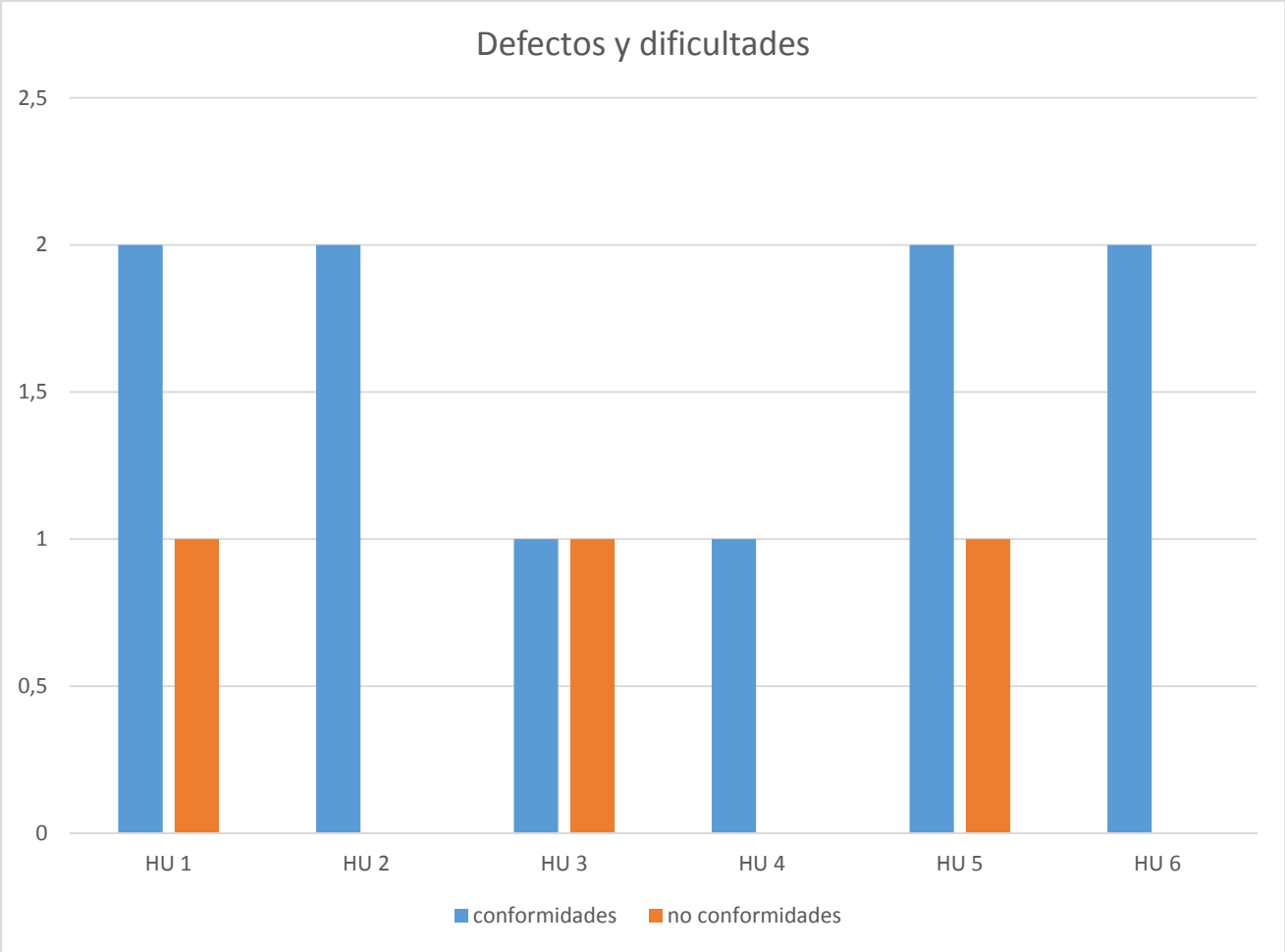


Figura 14: Resumen de defectos y dificultades

3.9 Conclusiones parciales

En este capítulo se registraron las bases necesarias para la implementación de la aplicación. Con la especificación de los estándares de codificación se asegura que la estructura y formato que tendrá el código fuente del sistema sea homogénea. Los diagramas de componente y de despliegue posibilitaron un mejor entendimiento de cómo quedará organizado el sistema final, así como las dependencias y las relaciones físicas entre los componentes del mismo. Por otra parte se realizaron un conjunto de pruebas de aceptación a cada historia de usuario arrojando un conjunto de no conformidades, las que fueron resueltas ofrecieron al cliente conformidad y seguridad ante las funcionalidades del sistema.

Conclusiones generales

- ✓ La recopilación de los referentes teóricos asociados a la comunicación de un servidor Modbus basada en Arduino posibilitó un mejor entendimiento del contexto de la investigación y de la problemática a resolver, así como identificar las tecnologías más adecuadas y menos costosas para permitir la comunicación inalámbrica de una DAQT con un sistema recolector de datos.
- ✓ Se definieron los artefactos necesarios propuestos por la metodología de desarrollo de software AUP-UCI durante la fase de análisis y diseño, los cuales posibilitaron el desarrollo adecuado de la capa de comunicación inalámbrica obtenida como resultado de la investigación.
- ✓ La capa de acceso inalámbrica para la comunicación de un servidor Modbus basada en Arduino brinda la facilidad de configuración de los parámetros de la red WiFi en la DAQT de forma automática; así mismo será capaz de permitir el acceso a la tarjeta de adquisición de datos desde un sistema recolector de datos, estableciendo una comunicación no cableada para el intercambio de información.
- ✓ La correcta ejecución de las pruebas realizadas permitió detectar y corregir varias no conformidades identificadas en dos iteraciones; verificándose de esta forma que el sistema desarrollado cumple con los requisitos propuestos y validándose además que la aplicación cuenta con la calidad requerida.

Recomendaciones

- ✓ Se propone para futuras investigaciones realizar una capa de acceso inalámbrico para la comunicación de un servidor Modbus basado en Arduino mediante el dispositivo bluetooth en su nueva versión 5.0.

Referencias bibliográficas

1. Chicala, Carlos Daniel. adquisición de datos, medir para conocer y controlar. 2012.

2. Soria, A y Acosta, C. pp. 2015.

3. 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.

4. Leyva Duran, Julio Alberto. resumen ejecutivo de Arex. 2016.

5. Cajas Andrade, Raúl Santiago y Campoverde Orozco, Freddy Nestor. diseño e implementación de una tarjeta de adquisición de datos. 2012.

6. Tome Osorio, Edgar Israel. Tarjeta de adquisición de datos (DAQT). 2013.

7. Ruiz Gutierrez, Jose Manuel. Arduino, manual de programación. 2012.

8. Arduino-Home. [En línea] 2018.

9. Monk, Simon. Programming Arduino Next steps. 2014.

10. Gutierrez, Annie y Velarde , cesar. Dispositivos inalámbricos. 2009.

11. López Ortiz, Francisco. El estandar IEEE 802.11. 2010.

12. Gomez, Ruben. capa de acceso a datos síncrona y asíncrona para dispositivos Modbus tcp. 2008.

13. Weis, Olga. Modbus over Ethernet. 2017.

14. Bedoya Meneses, Jessica Pamela. ESTUDIO COMPARATIVO DE LA EFICIENCIA DEL MEDIO DE COMUNICACIÓN ALÁMBRICO E INALÁMBRICO DEL PROTOCOLO MODBUS IMPLEMENTADO EN UN PROCESO MODULAR. 2015.

15. Access I/O Products. inc. [En línea] 2011.

16. Cesar. ciruitos eletronicos,ingenieria electronica. 2017.

17. Padilla, Erica P y Loayza, Jhon H. Diseño e implementacion de una aplicación Android para el control inalámbrico mediante una central basada en Arduino. Riobomba,Ecuador : s.n., 2016.

18. Informáticas, Universidad de las Ciencias. Metodología de desarrollo para la Actividad productiva de la UCI. Carretera a San Antonio Km 2 1⁄2 . Torrens. Boyeros. La Habana. Cuba : s.n., 2014.

19. Mediavilla, Elena. Modelados y herramientas UML. 2016.

20. Larman, Craig. UML y patrones. 2003.

21. Sanchez Franco, Mauricio Alberto y Pavlich-Mariscal, Jaime A. Modeling tool. 2016.

22. Schmuller, J. Aprendiendo UML. 2000.

23. UC3M. Guión Visual Paradigm for UML.Ingeniería del Software. Curso 201 3-2014. 2013.

24. Gallardo López, D. Lenguajes de programación. 2008.

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

26. Suárez Falcón, Yuniel. IDE de programación. 2015.

27. Cabeza, José Luis Comesaña. INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO . 2011-2012.

28. Gutiérrez, JMR. Manual Programación Arduino. 2007.

29. Sommerville, Ian. Ingeniería de Software. 2005.

30. Pressman, Roger S. Ingeniería del Software. 2005.

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

32. Olivera Sosa, Ángel Gabriel. Planificación y modelado. 2010.

33. Gonzalez, Rodriguez. 2013.

34. Fernández Castrillo, Alex. Compilador para el lenguaje de diagrama en escalera. 2014.

35. Laches Hernandez, Rosabel y Leyva Durán, Julio Alberto. Aplicación de configuración para Tarjeta de Adquisición de datos basada en microcontrolador. 2015.

36. Tello, Jesús Cáceres. Patrones de diseño: ejemplo de aplicación en los Generative Learning Object. 2008.

37. Meneses Yero, Mario Alexander. Editor para el lenguaje Lista de Instrucciones. 2015.

38. Vidal Otero, Mari Carmen. Dpto de lenguajes y Sistemas informáticos. 2009.

39. Network, Microsoft Developer. Revisiones de código y estándares de codificación. 2015.

40. Fiestas, Jhonattan. Pruebas para asegurar la calidad del producto de software. 2014.

41. Alejandro Martínez, Raúl Martínez. Guía a Rational Unified Process. España : Escuela Politécnica Superior de Albacete – Universidad de Castilla la Mancha, 2014.

42. Beymar Jimenez Ruiz, Sandra Peña Perz, Yamile Valdez Perez, Aracely J. Aramayo, Iver Salazar Zorrilla. METODOLOGIA DE DESARROLLO DE SOFTWARE-MSF-. Bolivia : s.n., 2012.

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

44. Ken Schwaber, Jeff Sutherland. La Guía Definitiva de Scrum. 2013.

45. Seco, José Antonio González. El lenguaje de programación C#. 2000.

46. Tobar, J y Sandoval, L. 2015. pág. 21.

47. Larman, Craig. UML y patrones. Introducción al análisis y diseño orientado a objetos. 2002.

48. Eclipse IDE. [En línea] 2004. [Citado el: 4 de enero de 2018.] http://www.asociacionaepi.es/eclipse-ide/?print=pdf.