Universidad de las Ciencias Informáticas
Centro de Informática Industrial, Facultad 5

Aplicación de configuración para tarjeta de adquisición de datos basada en microcontrolador AVR

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

Autor: Rosabel Laches Hernández

Tutor: Ing. Julio Alberto Leyva Durán

La Habana, 2015

"La planificación a largo plazo no es pensar en decisiones futuras, sino en el futuro de las decisiones presentes."

Peter Drucker

Dedicatoria

A mi mamá por ser mi guía y mi gran inspiración.

A mis abuelos que se merecen este triunfo porque han sido como mis padres.

A mi papá que aunque no está lo recordaré siempre.

A toda mi familia, que siempre me apoyan y que están muy orgullosos de mi.

Agradecimientos

A mi mamá por dármelo todo, por traerme hasta aquí a base de esfuerzo, sacrificios, ejemplo, humildad, por darme todo el amor del mundo; por deberle todo lo fui, lo que soy y lo que seré, han sido varios años de estudio y sacrificio, pero finalmente este es mi regalo por luchar siempre conmigo.

A mis abuelos por criarme con tanto esfuerzo y amor.

A mi familia en general por tanta preocupación y apoyo.

A mi novio por su cariño, su apoyo y por luchar conmigo todo este tiempo.

A mis amigos Yoenia, Tony y Alejandro que me han brindado su mano cuando la he necesitado.

A todos mis compañeros de grupo y apartamento durante estos cinco años.

A mi tutor por guiarme en el desarrollo de este proyecto.

A todos aquellos que de una forma u otra han contribuido al logro de este sueño.

Declaración de autoría

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

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

Rosabel Laches Hernández	Ing. Julio Alberto Leyva Durán
Autor	Tutor

Resumen

El presente trabajo surge por la necesidad de automatizar el proceso de configuración de los parámetros de ejecución de la tarjeta de adquisición de datos basada en Arduino (Arex DAQ) debido a que el método de configuración actual ocasiona pérdida de tiempo y una curva de aprendizaje alta en programación. Para realizar la propuesta planteada se trazó como objetivo, desarrollar una herramienta que permita la configuración de la tarjeta de adquisición de datos Arex DAQ. Se utilizó AUP como metodología de desarrollo de software en su versión UCI, para realizar el modelado del sistema se utiliza la herramienta CASE Visual Paradigm, apoyándose en el Lenguaje Unificado de Modelado (UML). Se hace uso del lenguaje de programación C++, el entorno de desarrollo integrado QtCreator y el framework de desarrollo QT en su versión 5.3. 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.

Palabras clave: arduino, configuración, tarjeta de adquisición de datos .

Índice general

Índice de figuras

Índice de tablas

Introducción

En la actualidad es difícil encontrar algún proceso que no esté vinculado a las Tecnologías de la Información y las Comunicaciones (TIC), estas han venido a revolucionar la forma de vida de los seres humanos por las ventajas que generan en el uso de las comunicaciones, la información, la industria y los servicios. El desarrollo de la informática desde sus inicios ha ido a la par del desarrollo industrial dotando a las grandes industrias de aplicaciones informáticas para el control de procesos industriales. A medida que estas aplicaciones aumentaron su complejidad se hizo más difícil su configuración, esto ha conllevado al crecimiento en el mercado mundial de las herramientas de configuración elevando el nivel de calidad de las mismas (IBARRA ALMAGUER, 2012).

Muchas de las industrias cubanas no poseen un sistema automático para el control de los procesos de producción. Por otro lado, existen otras industrias que no requieren procesos complejos de automatización o no cuentan con elevados recursos económicos para adquirir equipos que le permitan controlar y supervisar sus procesos industriales (SAN JUAN GUÍA, 2012).

En el ambiente industrial existe la necesidad de medir el comportamiento de un proceso a través de sus variables(valores relacionados con magnitudes y estados que definen el comportamiento de un sistema) (OLIVA LABAUT, 2012), por lo que la Universidad de las Ciencias Informáticas (UCI) propone como solución a los problemas que presentan las industrias cubanas, la creación del sistema de medición Arex, desarrollado en el Centro de Informática Industrial (CEDIN) de la facultad 5. El mismo está basado en sistemas domóticos que tienen como objetivo la automatización de viviendas (edificios, casas o cualquier local cerrado) aportando servicios de gestión energética, confort y comunicación.

El sistema Arex, puede ser ejecutado sobre la tarjeta CID-300/9 desarrollada por el Instituto Central de Investigaciones Digitales (ICID) y para la adquisición de las variables de los sensores se emplea una tarjeta basada en Arduino denominada Arex DAQ, la misma cuenta con tecnología de hardware y software abierto, específicamente como núcleo de procesamiento el Microcontrolador (MCU) Atmega2560, en el proyecto la tarea de configurar los parámetros de ejecución de la tarjeta debe realizarse empleando un lenguaje de bajo nivel recompilando los cambios en el firmware del dispositivo y en caso de ocurrir una modificación en el proceso industrial o en el hardware, se debe realizar nuevamente la recompilación del firmware de manera trabajosa. El método de configuración actual ocasiona pérdida de tiempo y una curva de aprendizaje alta en programación, por lo que se quiere desarrollar una herramienta que automatice la configuración de los parámetros de ejecución de la tarjeta, facilite el entorno apropiado en el que debe operar y permita establecer comunicación con terceros mediante el protocolo modbus.

A partir del análisis de la situación problemática, se define el siguiente **problema de investigación**: ¿Cómo gestionar la configuración de la tarjeta de adquisición de datos Arex DAQ?

Para dar solución al problema planteado, se identifica, como **objeto de estudio**: Mecanismos de configuración de dispositivos de hardware.

Siendo el **objetivo general** de la investigación, desarrollar una herramienta que permita la configuración de la tarjeta de adquisición de datos Arex DAQ.

Teniendo como **campo de acción** del presente trabajo, mecanismos de configuración de tarjeta de adquisición de datos.

Para dar cumplimiento al objetivo se propone el desarrollo de las siguientes **tareas de investigación**:

1. Elaboración del marco teórico a partir del estado del arte del tema.
2. Selección de las tecnologías y herramientas adecuadas para el desarrollo.
    - Lenguaje de programación.
    - Herramienta de modelado.
    - *Framework* de desarrollo.
    - Entorno Integrado de Desarrollo.
    - Lenguaje de modelado.
    - Metodología de desarrollo de software.
3. Definición de la propuesta de solución.
4. Diseño e implementación de la aplicación de configuración, para establecer los parámetros de estados del firmware modbus, comunicándose por alguna de las interfaces UART desde el microcontrolador Atmega2560 y una PC.
5. Diseño de pruebas de software.
    - Pruebas de aceptación.
6. Evaluación de los resultados de las pruebas de software.

Con el cumplimiento de los elementos antes planteados se espera obtener como **posible resultado** una herramienta que permita la configuración de la tarjeta de adquisición de datos Arex DAQ.

En la investigación se combinaron diferentes **métodos de investigación**, los fundamentales se mencionan a continuación:

Métodos Teóricos:

Análisis-síntesis, se utilizan para el estudio de cada una de las características de los parámetros de configuración del protocolo modbus, así como la interrelación de estos para gestionar las configuraciones que deben ser realizadas en la tarjeta de adquisición de datos.

Histórico-lógico, en el estudio de las tendencias históricas y actuales en cuanto a las aplicaciones de configuración basados en comunicación serial.

Métodos Empíricos:

Pruebas al sistema, para la constante realización de pruebas al sistema y determinar si se comporta según los resultados esperados.

Búsqueda bibliográfica, en la búsqueda y localización de referencias bibliográficas relacionadas con los mecanismos de configuración de dispositivos de hardware.

La tesis está estructurada en Introducción, tres capítulos, Conclusiones y Recomendaciones. Los capítulos son los siguientes:

- Capítulo 1. Fundamentación teórica: este capítulo esta dedicado a los principales conceptos asociados a la problemática así como el estado del arte de las tecnologías utilizadas y se examinan los principales métodos, técnicas y herramientas asociadas al tema en cuestión.
- Capítulo 2. Descripción del sistema. Planificación y diseño: el presente capítulo detalla cada uno de los puntos fundamentales dentro del diseño y planificación de la propuesta de solución, tales como la arquitectura del sistema, los patrones de diseño utilizados, las funcionalidades y estructura de clases del sistema desarrollado.
- Capítulo 3. Implementación y prueba: en este capítulo se define el estándar de codificación, el diagrama de paquete, diagrama de despliegue y la elaboración, ejecución y análisis de los resultados de los casos de prueba que evalúan las funcionalidades de la herramienta de configuración.

CAPÍTULO 1

Fundamentación Teórica

En el siguiente capítulo se abordan un grupo de conceptos de vital importancia para el desarrollo de la investigación, destacándose las características de los Sistema de Medición de Datos, las aplicaciones similares a la que se quiere desarrollar, los microcontroladores y algunas características de la plataforma de hardware y software abierto Arduino. Se expone el estudio realizado sobre la comunicación por puerto serie, el protocolo modbus, además se describen las herramientas y metodología utilizadas para resolver el problema antes expuesto, con el objetivo de obtener información que permita establecer el marco de trabajo favorable para el desarrollo de la aplicación de configuración.

1.1. Sistema de Medición de Datos

Un Sistema de Medición de Datos (SMD) es el grupo de instrumentos o calibres, estándar, operaciones, métodos, dispositivos, software, personal, medio ambiente y supuestos utilizados para cuantificar una unidad de medida o valoración determinada al rasgo de la característica medida; proceso completo utilizado para obtener mediciones (REYES AGUILAR, 2007). El objetivo de un SMD es indicar el valor de una variable. Para ello toma del medio la información y la transforma en una cantidad medible a través de un sensor (FERNÁNDEZ, 2010).

Entre los principales componentes de los SMD están los siguientes: (ibíd.)

Elemento primario del sensor: Convierte el valor de la magnitud de interés en un valor de una magnitud de otra naturaleza.

Elemento de conversión de variable: Convierte la magnitud transformada en la señal que puede ser transportada por el sistema.

Elemento manipulador de variable o acondicionador: Transforma la señal que llega a él a señales de formato estándar. Ya que el resto de los elementos que forman el sistema siguen una o algunas normas estandarizadas.

Elemento secundario o transmisor: Se encarga de manejar la comunicación con el resto de los integrantes del sistema. En el caso de las señales digitales requieren de microcontroladores o chips de funciones

específicas.

Línea de transmisión: Es la capa física que soporta la señal de comunicación con los indicadores y/o el sistema de almacenamiento. Su naturaleza puede ser el aire, cables con una composición estándar para las comunicaciones, líneas de alta tensión, el agua u otro líquido.

Elemento presentador de datos o Indicadores: Son los encargados de traducir la señal transmitida a un formato que pueda ser captado por los sentidos humanos. Pueden generar un número, un gráfico, un sonido, un conjunto de sonidos y una luz intermitente.

Elementos de almacenamiento de datos o Sistema de almacenamiento: Son los encargados de almacenar el comportamiento de una o varias magnitudes en el tiempo.

Los SMD nos permiten determinar el comportamiento de alguna magnitud de un objeto y determinar el valor que este posee.

Por otra parte la tarea de registrar las mediciones y procesar los datos recogidos para la visualización se ha convertido en fuente de ocupación para ingenieros. Desde hace varios años el uso de tarjetas de adquisición de datos se ha transformado en la vía mas efectiva de medir las señales y transferir los datos hacia las computadoras. La adquisición de datos (DAQ, siglas en inglés de *Data Acquisition*) no es más que la toma de un conjunto de señales físicas, convertirlas en señales eléctricas y digitalizarlas de manera que se puedan procesar en una computadora (PC) y brindar un mayor manejo de las variables ya sea con fines docentes, científicos, de almacenamiento o control (INSTRUMENTS, 2006a).

Se puede decir entonces que un sistema DAQ no es más que un conjunto de elementos encargados de adquirir diferentes tipos de variables del mundo real y convertirlas a señales que una computadora pueda entender para así poder realizar diferentes operaciones con esa información.

Arex es un sistema de medición porque sirve para medir procesos de baja y mediana complejidad (procesos que incluyen hasta 100 variables), a partir de la adquisición y procesamiento de datos en tiempo real asociados a las variables incidentes dentro del proceso. Arex está diseñado para ser ejecutado en dispositivos con recursos de hardware limitados, por tal motivo no incluye de forma directa la posibilidad de almacenar los datos por largos períodos de tiempo. Sin embargo brinda las interfaces necesarias para incorporar este servicio en caso de que se cuente con el soporte físico para este fin (GÓMEZ VARGAS, 2015).

Con el auge de los sistemas DAQ en diferentes sectores de la vida humana se ha hecho común el desarrollo de aplicaciones para realizar control automatizado de procesos en el ambiente industrial.

1.2. Aplicaciones de configuración

Actualmente se hacen imprescindibles las aplicaciones de configuración tanto en el sector industrial como en los equipos electrodomésticos. La mayor parte de estas aplicaciones son hechas para un software específico por lo tanto existen gran variedad de este tipo de aplicaciones. Haciendo un estudio de las diferentes herramientas de configuración existentes en el mercado se encontraron las siguientes aplicaciones similares a la que se pretende desarrollar:

- La Aplicación de configuración **AppConfig** permite configurar la versión 1 del TETSCADA, este dispositivo proporciona una pasarela (*Gateway*) que sirve como transporte entre los equipos industriales y las aplicaciones de gestión SCADA. El dispositivo se conecta con los terminales de radio TETRA[1] (interfaz RS-232) y con los dispositivos industriales (interfaces RS-232, RS-485 y Etehernet) y es gestionado por un firmware que implementa los mecanismos de comunicación de dichas interfaces. La aplicación de configuración para TETSCADA permite de una forma fácil al usuario la comunicación con el dispositivo, la lectura y modificación de los parámetros de operación, la descarga de un nuevo firmware existente en un archivo binario, la ejecución del firmware existente en el dispositivo, además de la modificación de los parámetros de información del dispositivo tanto como su versión y su número de serie. La aplicación cuenta con otras funcionalidades sin tener comunicación directa con el dispositivo tales como salvar y cargar los parámetros de configuración en y desde un archivo binario, actualizar la configuración con los valores de fábrica (PARRA NÁPOLES, 2011).

- La herramienta de configuración **Apus-PLC** desarrollada para la línea Sistemas Empotrados, tiene como objetivo configurar los parámetros de ejecución del PLC-HMI para que el mismo pueda ser utilizado en diferentes ambientes industriales según las necesidades de cada proceso. Esta herramienta brinda una interfaz para la gestión de los elementos: variables de entrada y salida ya sean digitales o analágicas, puertos de comunicación entre los que se encuentran RS-232 y Ethernet, variables de operación del CPU como son la frecuencia, niveles y la memoria RAM. La aplicación brinda la opción de salvar en archivos XML, local o directamente en el dispositivo las configuraciones realizadas. Además ofrece la posibilidad de cargar estos archivos ya sea de procedencia local o del dispositivo, en caso de ser remota, a través de un protocolo de comunicación (SAN JUAN GUÍA, 2012).

En el estudio de las herramientas existentes similares a la que se pretende desarrollar se detectaron características similares como son:

- Configuración del puerto serie.
- Salvar y cargar la configuración en un archivo.
- Salvar y cargar la configuración desde el mismo dispositivo.

En las industrias existen muchos dispositivos especializados en la adquisición y procesamiento de datos, un ejemplo de estos dispositivos son las tarjetas DAQ que se encargan de transformar las señales obtenidas del medio para que puedan ser entendidas por una PC. Entre los principales componentes de estas tarjetas están los microcontroladores, siendo este un pequeño dispositivo que contiene los elementos fundamentales de una computadora y se encuentra incorporado dentro de la máquina que controla, pero solo puede gobernar una sola tarea debido a sus limitadas prestaciones.

[1] Estándar abierto de comunicaciones definido por el Instituto Europeo de Normas de Telecomunicaciones

1.3. Microcontroladores

Los microcontroladores (MCU) son computadores digitales integrados en un chip que cuentan con un microprocesador o unidad de procesamiento central (CPU), una memoria para almacenar el programa, una memoria para almacenar datos y puertos de entrada/salida. Los microcontroladores son unidades autosuficientes y más económicas. Estos son ampliamente utilizados como el cerebro de una gran variedad de sistemas embebidos que controlan máquinas, componentes de sistemas complejos, como aplicaciones industriales de automatización y robótica, domótica, equipos médicos, sistemas aeroespaciales, e incluso dispositivos de la vida diaria como automóviles, hornos de microondas, teléfonos y televisores (FELIPEZ SÁNCHEZ y CRUZ, 2009).

1.3.1. Arquitectura básica de un microcontrolador

Aunque inicialmente todos los microcontroladores adoptaron la arquitectura clásica de Von Neumann, en el momento presente se impone la arquitectura Harvard. La arquitectura de Von Neumann se caracteriza por disponer de una sola memoria principal donde se almacenan datos e instrucciones de forma indistinta (Figura 1.1). A dicha memoria se accede a través de un sistema de buses único (direcciones, datos y control). La arquitectura Harvard dispone de dos memorias independientes, una que contiene solo instrucciones y otra, solo datos (Figura 1.1). Ambas disponen de sus respectivos buses de acceso y es posible realizar operaciones de acceso (lectura o escritura) simultáneamente en ambas memorias (ibíd.).

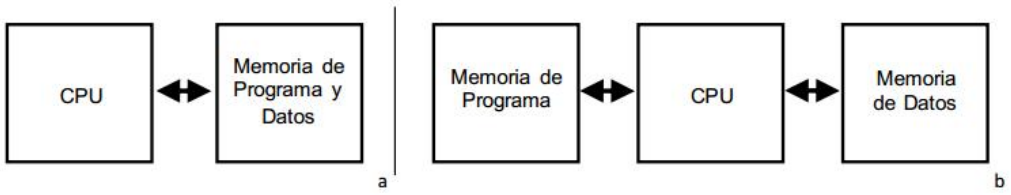

Figura 1.1. Arquitectura de los microcontroladores: a) Von Neumann b) Harvard.

La enorme ventaja que aporta la arquitectura Harvard tiene que ver con los dos tipos de memoria que comúnmente se utilizan en este tipo de sistemas. El programa y las constantes se almacenan en la memoria no volátil mientras que el almacenamiento de datos y variables puede realizarse en la memoria volátil. Esta última pierde su contenido cuando se interrumpe el suministro eléctrico mientras que la memoria no volátil siempre mantiene los datos almacenados, incluso después de desconectar la alimentación eléctrica (ibíd.), además el procesador puede acceder a cada una de ellas de forma simultánea, lo que se traduce en un aumento significativo de la velocidad de procesamiento. Típicamente los sistemas con esta arquitectura pueden ser dos veces más rápidos que sistemas similares con arquitectura Von Neumann (WILMSHURST, 2010).

1.3.2. Microcontroladores Atmel AVR

Los microcontroladores Atmel ofrecen una variada combinación de objetos altamente eficientes e integrados, tecnología comprobada e innovación revolucionaria ideal para los productos conectados e inteligentes de hoy. En la era actual los microcontroladores son una tecnología esencial para alimentar comunicaciones de máquina a máquina (M2M). Con base en décadas de experiencia y liderazgo en este sector Atmel posee arquitecturas comprobadas y optimizadas para bajo consumo de energía, conectividad de alta velocidad, largura de banda de datos ideal y soporte avanzado para interfaces. ATMEL fabrica los microcontroladores de la familia AVR, esta nueva tecnología proporciona todos los beneficios habituales de la arquitectura Computador con Conjunto de Instrucciones Reducidas(RISC por sus siglas en inglés *Reduced Instruction Set Computer*). La característica que identifica a estos microcontroladores de ATMEL es la memoria flash y la memoria de sólo lectura eléctricamente programable y borrable (EEPROM por sus siglas en inglés *Electrica Erasable Programmable Read OnIy Memory*) que incorpora. AVR compite con varias familias de microcontroladores bien establecidas en el mercado, tales como 8051 de Intel, 68HC11 de Motorola y la familia PIC de Microchip. El diseño AVR de ATMEL difiere de los demás microcontroladores por tener mayor cantidad de registros, en este caso tiene 32 registros de 8 bits. Esto hace que la arquitectura AVR sea más fácil de programar a nivel de lenguaje ensamblador y que sea fácil de optimizar con un compilador. El gran conjunto de registros disminuye la dependencia respecto a la memoria, lo cual mejora la velocidad y disminuye las necesidades de almacenamiento de datos (ATMEL, 2014).

Figura 1.2. Microcontrolador ATMEL: ATMEGA2560.

Las tarjetas de adquisición de datos son un elemento importante dentro de los sistemas DAQ por lo que mientras más fácil sea su uso y más ventajas aporte para la creación de pequeños sistemas que permiten automatizar procesos en los hogares mejor será para desarrollar sistemas domóticos de bajo coste. Estas oportunidades se pueden encontrar en la plataforma de hardware y software abierto Arduino, esta placa se ha vuelto muy popular en los ultimos años porque con ella se pueden desarrollar un buen número de proyectos relacionados con los microcontroladores y la electrónica en general.

1.4. Plataforma de hardware y software abierto Arduino

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software-hardware flexibles y fáciles de usar, recibe información del entorno a través de los pines de entrada, de toda

una gama de sensores y puede actuar sobre los dispositivos de salida controlando luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino y el entorno de desarrollo Arduino.

Entre las ventajas que ofrece trabajar con arduino están: (ARDUINO, 2014b)

- **Asequible**: Las placas Arduino son relativamente baratas en comparación con otras plataformas de microcontroladores. La versión menos costosa del módulo Arduino puede ser ensamblado a mano, e incluso los módulos de Arduino premontados cuestan menos de cincuenta euros.
- **Multiplataforma**: El software de Arduino funciona en sistemas operativos Windows, Macintosh OSX y Linux. La mayoría de los entornos para microcontrolador se limitan a Windows.
- **Entorno de programación simple y claro**: El entorno de programación de Arduino es fácil de usar para principiantes y lo suficiente flexible para usuarios avanzados.
- **El software es de código abierto y extensible**: El software de Arduino está publicado bajo una licencia libre y el lenguaje puede ampliarse a través de bibliotecas de C++.
- **El código abierto y el hardware extensible**: Arduino esta basado en el ATMEGA8 y ATmega168 que son microcontroladores de Atmel. Los planos de los módulos están publicados bajo una licencia de Creative Commons (CC, en español equivaldría a: '[Bienes] Comunes Creativos'), por lo que los diseñadores de circuitos experimentados pueden hacer su propia versión del módulo, ampliándolo y mejorándolo. Incluso los usuarios con poca experiencia pueden construir la versión para placa de desarrollo para entender cómo funciona y ahorrar dinero.

1.4.1. Arduino MEGA ADK

Esta placa está basada en el Arduino MEGA2560 pero modificada para permitir su uso con Android ya que dispone de un puerto USB Host incorporado. Cuenta con un microcontrolador ATmega8U2 programado como convertidor serial a USB. Entre las propiedades que presenta el MEGA ADK Arduino se encuentra que puede ser alimentado a través de la conexión USB o con una fuente de alimentación externa, la fuente de alimentación se selecciona automáticamente, además posee una serie de interfaces que permiten la comunicación con un ordenador, otro Arduino u otros microcontroladores. El Atmega2560 ofrece cuatro interfaces UART(son las siglas en inglés de *Universal Asynchronous Receiver-Transmitter*) para lógica transistor a transistor (TTL sigla en inglés de*transistor-transistor logic*), es decir (5V) de comunicación serie. La interfaz de host USB dada por MAX3421E permite al Arduino MEGA ADK conectar e interactuar con cualquier tipo de dispositivo que tenga un puerto USB (ARDUINO, 2014a).

Entre las características que ofrece trabajar con Arduino MEGA ADK están:

- **Microcontrolador**: ATmega2560

- **Voltaje de operación**: 5 V
- **Voltaje de entrada (recomendado)**: 7 - 12 V
- **Voltaje de entrada (límites)**: 6 - 20 V
- **Pines de entrada-salida digital:** 54 (15 de ellos proporcionan salida de modulación por ancho de pulsos (PWM, siglas en inglés de *pulse-width modulation*)
- **Pines de entrada analógica:** 16
- **Corriente por pin de entrada-salida:** 40 mA
- **Corriente para pines de 3.3** V: 50 mA
- **SRAM:** 8 KB
- **EEPROM:** 4 KB
- **Velocidad del reloj:** 16 MHz

Arduino posee dentro de sus características la posibilidad de comunicarse con una computadora a través del puerto serie, por tanto este elemento es un componente fundamental en los proyectos de Arduino y se debe aprender y entender su funcionamiento para poder sacar la mayor utilidad al mismo, el siguiente epígrafe se enfocará en la comunicación serial y sus principales características.

1.5. Comunicación serial

La comunicación serial es un protocolo muy común para comunicación entre dispositivos que se incluye de manera estándar en prácticamente cualquier computadora. La mayoría de las computadoras incluyen dos puertos seriales RS-232. Además, la comunicación serial puede ser utilizada para adquisición de datos si se usa en conjunto con un dispositivo remoto de muestreo. El concepto de comunicación serial es sencillo. El puerto serial envía y recibe bytes de información, un bit a la vez. Aún y cuando esto es más lento que la comunicación en paralelo, que permite la transmisión de un byte completo por vez, este método de comunicación es más sencillo y puede alcanzar mayores distancias. Típicamente, la comunicación serial se utiliza para transmitir datos en formato ASCII (*American Standard Code for Information Interchange*). Debido a que la transmisión es asincrónica, es posible enviar datos por una línea mientras se reciben datos por otra. Las características más importantes de la comunicación serial son la velocidad de transmisión, los bits de datos, los bits de parada, la paridad y control de flujo. Para que dos puertos se puedan comunicar, es necesario que las características sean iguales (INSTRUMENTS, 2006b).

- **Velocidad de transmisión (baud rate):** Indica el número de bits por segundo que se transfieren y se mide en baudios (bauds).

- **Bits de datos:** Se refiere a la cantidad de bits en la transmisión. Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente seria de 8 bits. Las cantidades más comunes de bits por paquete son 5, 6,7 y 8 bits. El número de bits que se envía depende del el tipo de información que se transfiere.

- **Bits de parada:** Usado para indicar el fin de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits. Debido a la manera como se transfiere la información a través de las líneas de comunicación y que cada dispositivo tiene su propio reloj, es posible que los dos dispositivos no estén sincronizados. Por lo tanto, los bits de parada no sólo indican el fin de la transmisión sino además dan un margen de tolerancia para esa diferencia de los relojes. Mientras más bits de parada se usen, mayor sería la tolerancia a la sincronía de los relojes, sin embargo la transmisión sería más lenta.

- **Paridad:** Es una forma sencilla de verificar si hay errores en la transmisión serial. Existen cuatro tipos de paridad: par, impar, marcada y espaciada. Esto permite al dispositivo receptor conocer de antemano el estado de un bit, lo que serviría para determinar si hay ruido que esté afectando de manera negativa la transmisión de los datos, o si los relojes de los dispositivos no están sincronizados.

- **Control de flujo:** fue desarrollado para asegurar que los datos no se pierdan durante la transferencia.

Los protocolos de comunicación en la industria son ampliamente usados porque hacen posible que una Estación Maestra (MTU por sus siglas en inglés) pueda comunicarse con una o varias Estaciones Remotas (RTU por sus siglas en inglés) con el objetivo de obtener datos de campo para la supervisión y control de un proceso.

1.6. Protocolo de Comunicación

El protocolo de comunicación es una estructura definida y concertada de construcción de mensajes y/o datos para ser transmitida entre un transmisor y un receptor a través de un medio físico. La estructura de un protocolo de comunicación debe describir quién envía el mensaje, hacia donde va, los datos e información para verificar que el mensaje que llega está completo. Por lo tanto el transmisor y el receptor de los datos deben manejar el mismo protocolo para que ambos puedan transmitir, recibir e interpretar la información (JOSÉ y FERMÍN, 2008).

1.7. Protocolo de comunicación modbus

Modbus es un protocolo estándar que puede gestionar una comunicación tipo cliente-servidor entre distintos equipos conectados físicamente con un bus serie, este protocolo con el tiempo se ha convertido en unos de los más empleados en las comunicaciones industriales. Las principales razones de ello son la sencillez del protocolo, versatilidad, y que sus especificaciones, gestionadas por la MODBUS Organization,

son de acceso libre y gratuito, este es un protocolo de tipo Petición/Respuesta, por lo que en una transacción de datos se puede identificar al dispositivo que realiza una petición como el cliente o maestro, y al que devuelve la respuesta como el servidor o esclavo de la comunicación. En una red modbus se dispone de un equipo maestro que puede acceder a varios equipos esclavos. Cada esclavo de la red se identifica con una dirección única de dispositivo. Un maestro puede hacer dos tipos de peticiones a un esclavo: para enviar datos a un esclavo y espera su respuesta de confirmación, o para pedir datos a un esclavo y espera su respuesta con los datos. Para intercambiar las peticiones y respuestas, los dispositivos de una red modbus organizan los datos en tramas. Dado que modbus es un protocolo de nivel de aplicación, se requiere utilizarlo sobre una pila de protocolos que resuelva los temas específicos del tipo de red empleada. En la Figura 1.4 se muestran un ejemplo de las tramas de estos dos tipos de modbus (CANDELAS HERÍAS, 2011).

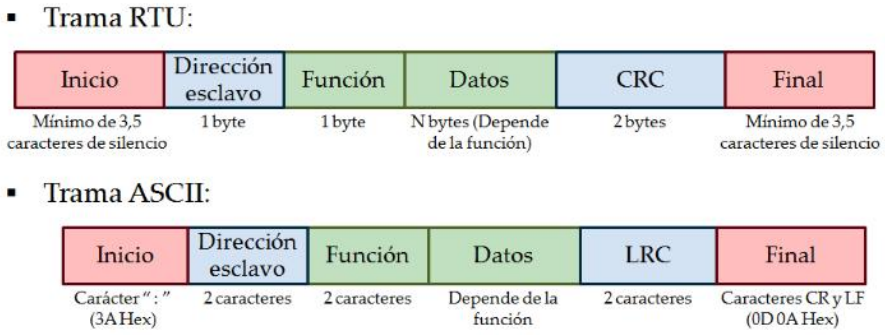

Figura 1.3. Formato de las tramas de modbus RTU y modbus ASCII.

Modbus diferencia cuatro tipos de datos y tiene funciones específicas para ellos con un direccionamiento independiente. Se diferencian entre sí, en cuanto a si son de lectura o de escritura, y si son de tipo Bit o tipo Word (16 bits)(SÁNCHEZ YERA y TORRES GONZÁLEZ, 2013).

Tipo de dato	Lectura/Escritura	Ancho dato	Comentario
Entradas discretas (Discret Input)	Lectura	1 bit	Variables de un bit que el amo puede leer.
Salidas discretas (Discret output)	Lectura /Escritura	1 bit	Variables de un bit que el amo puede escribir y leer
Registros de entrada (Input Register)	Lectura	palabra (16 bits)	Variables de 16 bits que el amo puede leer.
Registros de salida (Holding Register)	Lectura / Escritura	palabra (16 bits)	Variables de 16 bits que el amo puede escribir y leer.

Figura 1.4. Tipo de datos modbus.

Modbus ofrece 24 funciones diferentes. Están caracterizadas por una función asociada a un tamaño de 1 byte y no todos los dispositivos admiten este tipo de función. A continuación se describen algunas de estas.

Código	Acción	Significado
01	Leer Bobinas	Obtiene el estado actual ON/OFF de un grupo de bobinas lógicas.
02	Leer Entradas	Obtiene el estado actual ON/OFF de un grupo de entradas lógicas.
03	Leer Registros	Obtiene el valor binario de uno o más registros de almacenamiento.
04	Leer Registros	Obtiene el valor binario de uno o más registros de entrada.
05	Escribir Bobina	Fuerza el estado de una bobina.
06	Escribir Registro	Escribe el valor binario de un registro de almacenamiento.
15	Escribir Bobinas	Fuerza el estado de un grupo de bobinas.
16	Escribir Registros	Escribe el valor binario de un grupo de registros de almacenamiento.

Figura 1.5. Funciones modbus.

1.8. Herramientas y tecnologías a utilizar

A continuación se describen las tecnologías y herramientas a utilizar. Esta elección responde a múltiples razones, entre las que se encuentra el hecho de que las mismas son exponentes del movimiento de Software Libre lo que contribuye al desarrollo del país y a la generalización de los principios y metas que propone este movimiento ya que su utilización no implica gastos adicionales por concepto de cambio de plataforma ni la adquisición de licencias de software privativos.

1.8.1. Metodología de desarrollo de software

Las metodologías de desarrollo de software son un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en un proyecto. Hoy en día existen numerosas propuestas metodológicas, un ejemplo de ellas son las propuestas tradicionales centradas específicamente en el control del proceso, estas han demostrado ser efectivas en proyectos de gran tamaño. Sin embargo, la experiencia ha demostrado que las metodologías tradicionales no ofrecen una buena solución a proyectos donde el entorno es volátil y donde los requisitos no se conocen con exactitud. Como respuesta a las dificultades presentadas por las metodologías tradicionales, surgen las metodologías ágiles, las cuales tratan de adaptarse a la realidad del software (DELGADO ALÓN y JIMÉNEZ LÓPEZ, 2012).

Algunas Metodologías ágiles son: Programación Extrema (XP), SCRUM, Crystal Clear, Método de desarrollo de sistemas dinámicos (DSDM), Open Unified Process (OpenUP), entre otras. Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto, se decide hacer una variación de la metodología Proceso Unificado ágil (AUP por sus siglas en inglés) es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil la forma de desarrollar aplicaciones de software usando técnicas ágiles y conceptos que se mantienen válidos en RUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI.

A continuación se describe dicha metodología y las variaciones realizadas para adaptarla a las necesidades de desarrollo de software en la UCI (SÁNCHEZ, 2015).

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega una fase de Cierre. Para una mayor comprensión de las fases se explican a continuación.

1. **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

2. **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.

3. **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 8 disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación y la disciplina Despliegue se considera opcional. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto).

1. **Modelado de negocio (opcional):** El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito.

2. **Requisitos:** El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.

3. **Análisis y diseño:** En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea

fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis.

4. **Implementación:** En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.

5. **Pruebas interna:** En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posibles componentes de prueba ejecutables para automatizar las pruebas.

6. **Pruebas de liberación:** Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.

7. **Pruebas de Aceptación:** Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

8. **Despliegue (Opcional):** Constituye la instalación, configuración, adecuación, puesta en marcha de soluciones informáticas y entrenamiento al personal del cliente.

1.8.2. Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML por sus siglas en inglés *Unified Modeling Language*) es un lenguaje estándar para escribir planos de software. UML puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software.

UML es el lenguaje que permite la modelación de sistemas de software con tecnología orientada a objetos, es uno de los más conocidos y utilizados en la actualidad. Se utiliza para definir un sistema en cada una de las etapas por las que tiene que pasar, indica que es lo que supuestamente hará, pero no como lo hará. Incluye aspectos conceptuales tales como procesos de negocio, funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Por lo que es muy útil para comprender el diseño del sistema (GROUP, 2015).

1.8.3. Herramienta de modelado

Hoy día son muy utilizadas las herramientas de Ingeniería Asistida por Computadora (CASE por sus siglas en inglés *Computer Aided Software Engineering*) ya que son diversas aplicaciones informáticas destinadas a automatizar las fases del ciclo de vida de un sistema software.

Luego de un detallado estudio se decidió utilizar para el modelado del sistema la herramienta CASE Visual Paradigm (versión 8.0), ya que esta brinda una respuesta rápida y permite modelar todos los diagramas que se realizarán a lo largo de este trabajo. A continuación se ofrecen más detalles sobre la herramienta escogida.

Visual Paradigm es una herramienta para la creación de diagramas UML que es de gran ayuda en el proceso de desarrollo de software. Especialmente en las fases de análisis y diseño de este proceso, está aplicación ayuda a conseguir un producto de alta calidad. Este sistema puede manejar gran parte de los diagramas definidos en el lenguaje UML, ya sea creándolos manualmente o importándolos a partir de código en C++, Java, Python, Pascal, Delphi, entre otros, lo cual ofrece posibilidades de ingeniería inversa. Permite crear un modelo y generar el código automáticamente en varios lenguajes para ayudar a comenzar la implementación del proyecto. El código generado incluye definiciones de clases con sus métodos y atributos proporcionando rapidez en el proceso inicial de desarrollo y haciéndolo menos propenso a errores (DELGADO ALÓN y JIMÉNEZ LÓPEZ, 2012).

1.8.4. Lenguaje de programación

C++ es un lenguaje de programación diseñado a mediados de los años 80. La intención de su creación fue extender al exitoso lenguaje de programación C con mecanismos que permitieran la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, C++ es un lenguaje híbrido, es posible programar en estilo imperativo o en estilo orientado a objeto (GÓMEZ RUIZ, 2015).

Entre las principales ventajas de C++ como lenguaje de programación orientado a objetos se encuentran:(SAN JUAN GUÍA, 2012)

- C++ es un lenguaje de propósito general, o sea, que con él se puede programar cualquier cosa, desde sistemas operativos y compiladores hasta aplicaciones de bases de datos y procesadores de texto.
- Los programas escritos en C++ tienen la ventaja de que podrán ejecutarse en cualquier máquina y bajo cualquier sistema operativo.
- Es un lenguaje multinivel, es decir, se puede usar para programar directamente el hardware o para crear aplicaciones de desktop.

Se utiliza C++ por las ventajas dichas anteriormente y porque es un lenguaje de código abierto, existe una gran cantidad de documentación y se adecua al desarrollo de las funcionalidades requeridas para el desarrollo de la aplicación.

1.8.5. Framework de desarrollo

Qt es un marco de trabajo (*framework*) escrito en C++ para el desarrollo de aplicaciones de Interfaz Gráfica de Usuario, amplía las posibilidades del lenguaje C++ con una extensa biblioteca de clases, de uso intuitivo y dividida en módulos, en los cuales se puede encontrar desde programación de interfaces gráficas

de usuarios hasta programación de redes, procesamiento de textos enriquecidos, acceso a datos almacenados en varios de los gestores de bases de datos más populares. Incorpora herramientas para escribir aplicaciones robustas y en el menor tiempo posible como el Diseñador Qt (QtDesigner). A esto se le suma su extensa base de datos de ejemplos listos para usar. Qt está disponible bajo licencia LGPL, permitiendo el desarrollo de Software Libre, y si se desea, software comercial no libre, en ambos casos sin vernos obligados al pago de licencias o derechos (VELASCO PÉREZ y VASCONCELO MIR, 2010). Para el desarrollo del presente trabajo se utiliza en su versión 5.3.

1.8.6. Entorno Integrado de Desarrollo

Los Entornos Integrados de Desarrollo (IDE por sus siglas en inglés *Integrated Development Environment*) constituyen programas compuestos por un conjunto de herramientas para los programadores que facilitan la escritura de programas. Pueden dedicarse en exclusivo a un sólo lenguaje de programación o bien, pueden utilizarse para varios. Los IDE facilitan un ambiente de trabajo amigable. Para el desarrollo de la herramienta se decidió usar QtCreator como IDE. QtCreator ofrece un ambiente de desarrollo completo para la creación de aplicaciones Qt. Es una herramienta ligera y multiplataforma, con un enfoque estricto hacia las necesidades de los desarrolladores de aplicaciones Qt(ibíd.). Para el desarrollo del presente trabajo se utiliza en su versión 3.1.2.

1.9. Conclusiones parciales

De manera general en este capítulo se vieron en que consisten los SMD, sus principales componentes, se muestran algunas características de la comunicación serial, los microcontroladores y el protocolo de comunicación modbus, se seleccionaron las herramientas y tecnologías a utilizar. En el mismo, se destaca la selección de AUP-UCI como metodología de desarrollo, la determinación de C++ como lenguaje de programación, de Qt como framework de desarrollo y por último, la selección del Lenguaje Unificado de Modelado junto con la herramienta Visual Paradigm para la modelación de la solución.

CAPÍTULO 2

Descripción del sistema. Planificación y diseño

En el presente capítulo se exponen los elementos que forman parte del modelo de desarrollo del software como son: la propuesta del sistema y una visión general del diseño de la solución.

2.1. Propuesta del sistema

La solución que se propone abarca la creación de una herramienta informática capaz de configurar los parámetros de ejecución de la tarjeta de adquisición de datos y los parámetros de los campos de las funciones de modbus. La herramienta brinda una interfaz para la configuración del puerto serie donde se puede parametrizar los valores: *baud rate, data bits, parity, stop bits y flow control* que va a permitir la comunicación con la tarjeta. Además cuenta con otra interfaz que muestra los pines del dispositivo donde se puedan definir como *input, output, not used*, también se podrán configurar el banco de operación de modbus *coil status, input status, holding register, input register* y las referencias de los registros de modbus. Una vez finalizada la configuración puede ser almacenada de forma local en un fichero o ser enviada directamente hacia el dispositivo (Figura 2.1).

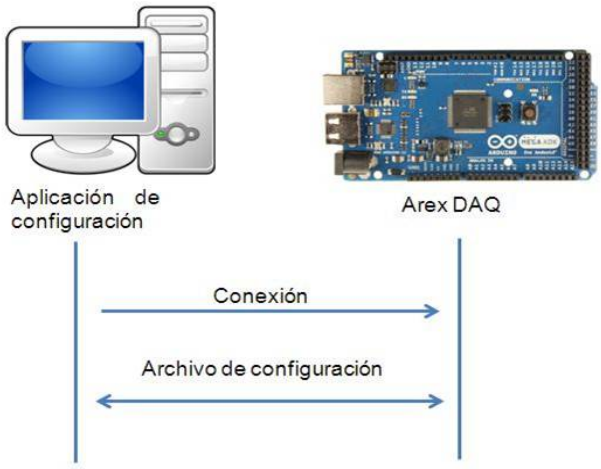

Figura 2.1. Propuesta del sistema

2.2. Requerimientos del sistema

Los requerimientos no funcionales especifican las propiedades o cualidades que debe tener la solución a desarrollar. Representan las características que hacen al producto atractivo, usable, rápido o confiable. Estos requisitos pueden marcar la diferencia entre un producto bien aceptado y otro con poca aceptación (VELASCO PÉREZ y VASCONCELO MIR, 2010).

A continuación se listan estas características:(ibíd.)

Requerimientos de usabilidad

Podrá ser usado con facilidad por parte del usuario.

Requerimientos de restricciones en el diseño e implementación

El sistema debe ser desarrollado en el lenguaje de programación C++.

Las interfaces gráficas de usuarios deben ser desarrolladas utilizando el marco de trabajo Qt.

El sistema debe cumplir con las especificaciones técnicas del hardware de la tarjeta de adquisición de datos .

Requerimientos de Apariencia o interfaz externa

La interfaz debe ser de fácil comprensión en su funcionamiento permitiendo la utilización del sistema sin mucho entrenamiento, es decir, de fácil uso y con rápida respuesta del sistema.

Interfaces uniformes con los mismos colores y diseños.

La interfaz debe tener mensajes sin ambigüedades.

Requerimientos de licencias y patentes

Se deben utilizar herramientas libres.

Requerimientos de Portabilidad

El sistema debe funcionar en sistemas de la familia GNU/Linux y Windows.

Requerimientos de ayuda y documentación

El proyecto debe contar con una documentación según la metodología establecida.

Requerimientos Político culturales

Debe respetar los términos empleados normalmente, por los especialistas en el tema de la esfera que se automatiza.

Requerimientos de hardware

Procesador: 400 MHz.

Memoria RAM: 128 MB o superior.

Disco Duro: 25 MB.

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 (GONZÁLEZ RODRÍGUEZ, 2013).

Se identificarón seis historias de usuario las cuales se detallan a continuación:

Historia de usuario	
Número: 1	**Nombre del requisito:** Establecer comunicación entre la aplicación de configuración y la tarjeta de adquisición de datos.
Programador: Rosabel Laches Hernández.	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 1
Riesgo en Desarrollo: Alto	**Tiempo Real:** 1
Descripción: Permite establecer la comunicación entre la aplicación de configuración y la tarjeta de adquisición de datos para que exista un correcto envió y recibo de datos.	

Tabla 2.1. Historia de usuario 1

Historia de usuario	
Número: 2	**Nombre del requisito:** Recibir la configuración del dispositivo.
Programador: Rosabel Laches Hernández.	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alto	**Tiempo Real:** 3
Descripción: Permite visualizar la configuración actual de la tarjeta de adquisición de datos.	

Tabla 2.2. Historia de usuario 2

Historia de usuario	
Número: 3	**Nombre del requisito:** Modificar los parámetros de configuración del dispositivo.
Programador: Rosabel Laches Hernández.	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alto	**Tiempo Real:** 3
Descripción: Permite especificar los nuevos valores que va a tomar la configuración de la tarjeta de adquisición de datos.	

Tabla 2.3. Historia de usuario 3

Historia de usuario	
Número: 4	**Nombre del requisito:** Modificar parámetros del puerto serie de configuración.
Programador: Rosabel Laches Hernández.	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alto	**Tiempo Real:** 3
Descripción: Permite modificar la configuración del puerto serie existente en un archivo y asi poder actualizarla o simplemente ver que configuración es la existente.	

Tabla 2.4. Historia de usuario 4

Historia de usuario	
Número: 5	**Nombre del requisito:** Enviar la nueva configuración de la tarjeta de adquisición de datos.
Programador: Rosabel Laches Hernández.	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 2
Riesgo en Desarrollo: Alto	**Tiempo Real:** 2
Descripción: Envía la nueva configuración al dispositivo.	

Tabla 2.5. Historia de usuario 5

Historia de usuario	
Número: 6	**Nombre del requisito:** Salvar y cargar la configuración en un archivo.
Programador: Rosabel Laches Hernández.	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 3
Riesgo en Desarrollo: Alto	**Tiempo Real:** 3
Descripción: Permite salvar y cargar la configuración existente en el dispositivo en un archivo.	

Tabla 2.6. Historia de usuario 6

2.4. Plan de iteraciones

Iteración 1

Esta primera iteración tiene como meta implementar las historias de usuarios: establecer comunicación entre la aplicación de configuración y la tarjeta de adquisición de datos, recibir la configuración de la tarjeta de adquisición de datos, modificar los parámetros de configuración del dispositivo y modificar parámetros del puerto serie de configuración. Estas historias de usuario constituyen la estructura básica del sistema. Con la finalización de esta iteración se obtendrá la entrega del sistema con el propósito de mostrar al cliente lo realizado y recibir retroalimentación del mismo.

Iteración 2

En esta iteración se implementan las historias de usuario: enviar la nueva configuración del dispositivo y salvar y cargar la configuración en un archivo. Al finalizar esta iteración, luego de que haya pasado satisfactoriamente por la etapa de pruebas se dispondrá de la aplicación con todas las funcionalidades descritas por el cliente.

2.5. Modelo arquitectónico propuesto

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 (PRESSMAN, 2005).

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 (SOMMERVILLE, 2005).

Un estilo arquitectónico es una transformación impuesta al diseño de todo un sistema. El objetivo es establecer una estructura para los componentes del sistema. Con el objetivo de establecer una estructura para todos los componentes del sistema se escoge el estilo arquitectónico Orientado a objeto. El mismo plantea que los componentes de un sistema encapsulan los datos y las operaciones que deben aplicarse para manipular estos datos. La comunicación y coordinación entre los componentes se consigue mediante el paso de mensajes (PRESSMAN, 2005).

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: 1) el alcance de un patrón es menor, ya que se concentra en un aspecto en lugar de hacerlo en toda la arquitectura; 2) 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 la infraestructura (por ejemplo, concurrencia); 3) los patrones arquitectónicos tienden a abarcar aspectos específicos del comportamiento dentro del contexto de la arquitectura. Los patrones se usan junto con un estilo arquitectónico para determinar la forma de la estructura general de un sistema (ibíd.).

El patrón arquitectónico que representa la estructura de los datos y los componentes de la herramienta de configuración es el Modelo Vista Controlador (MVC por sus siglas en inglés *Model-View-Controller*).

MVC consta de tres tipos de objetos (GAMMA; HELM; JOHNSON y VLISSIDES, 1997).

- **Modelo:** es el objeto de la aplicación, es la representación específica de la información con la cual el sistema opera, es decir, sobre la cual funciona la aplicación. No depende de ninguna vista y de ningún controlador.
- **Vista:** maneja la visualización de la información. Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador:** define la forma en que la interfaz de usuario reacciona a la entrada del usuario, controla el flujo entre la vista y el modelo(los datos). Este responde a eventos, usualmente acciones del usuario

e invoca cambios en el modelo y probablemente en la vista. Es el encargado de realizar la lógica específica del negocio.

Aplicando dicho estilo arquitectónico a la solución propuesta, la arquitectura del software quedo distribuida de la siguiente forma:

Modelo: Clase *SerialConfig* y *CElement*.

Vista: Clase *Visual*, *PasswordLogin* y *Connection*.

Controlador: Clase *Controller*.

Con la siguiente imagen se muestra más claro su funcionamiento:

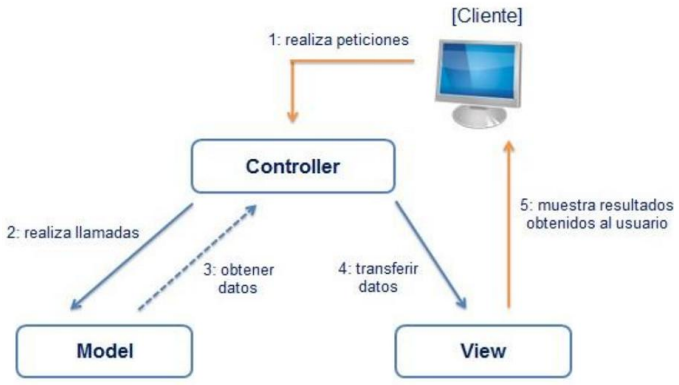

Figura 2.2. Arquitectura Modelo Vista Controlador

Como se muestra en la ilustración el cliente envía una señal llamada Petición, esta es interceptada por el Controlador quien realiza las validaciones necesarias, procesamiento de dichos datos y lógica de negocio asociadas a esa petición del cliente. El Controlador envía datos al Modelo y los obtiene dependiendo de la solicitud del usuario para finalmente enviarlos a la Vista a fin de ser mostrado nuevamente al cliente a través de una respuesta (CLARA HERNÁNDEZ e ISE MORALES, 2014).

2.6. Patrones de Diseño

Los patrones de diseño son las descripciones de los objetos que se comunican y clases que son personalizadas para resolver un problema de diseño en general en un contexto particular (GAMMA; HELM; JOHNSON y VLISSIDES, 1997).

A continuación se exponen los patrones de diseño seleccionados:

2.6.1. Patrones de diseño GoF

Singleton: Con el uso de este patrón se asegura que se cree una sola instancia de una clase y se provee un solo punto de acceso global. Se crea una clase que gestiona una sola instancia de ella misma y cada vez

que se necesite, simplemente se consulta la clase y esta devolverá siempre, la misma instancia.

En el sistema, este patrón se ve reflejado en la clase *Visual* que posee una única instancia del objeto SerialPort.

2.6.2. Patrones de diseño GRASP

Los Patrones Generales de Asignación de Responsabilidades, por sus siglas en inglés (GRASP), son patrones de diseño que se usan para asignar responsabilidades a una clase (SAN JUAN GUÍA, 2012).

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.

Este patrón se puede observar en la clase *CElement* que contiene el dispositivo y es la encargada de crear sus elementos configurables.

Alta Cohesión: Asignar una responsabilidad de modo que la unión se mantenga a gran escala. Asignar a las clases responsabilidades que trabajen sobre una misma área de aplicación y que no tengan mucha complejidad. Mejoran la claridad y facilidad con que se entiende el diseño.

En el sistema, este patrón se ve reflejado mediante las clases *Visual*, *PasswordLogin* y *Connection* de la capa vista que se encargan de visualizar los datos y manejar las interacciones del usuario, la clase *Controller* del Controlador se encarga de realizar un modelo estándar de presentación de los datos y las clases *SerialConfig* y *CElement* del Modelo se encargan de contener los datos.

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.

Se evidencia en la relación que tienen las clases *Visual* y *Controller*, la clase *Visual* no tiene relación con ninguna clase de la capa modelo y solo activa cuando se emite una señal de la clase *Controller*.

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.

Este patrón se evidencia en la clase *Controller*, ya que sirve como intermediario entre la clase *Visual* y *CElemens*. En dependencia del método llamado invoca al algoritmo necesario para cumplir dicha funcionalidad.

2.7. Diagrama de clases

Un diagrama de clases de diseño representa las especificaciones de las clases en una aplicación. Entre la información general se encuentran: clases, asociaciones, atributos, métodos, navegabilidad y dependencias (LARMAN, 2002).

El siguiente diagrama muestra la relación entre las clases pertenecientes al modelo de datos y sus funcionalidades, en este diagrama, la clase *Visual* es la que incluye y organiza a los demás elementos de la capa vista, estos son: el *PasswordLogin* y *Conecction*. En la capa Modelo existe una estructura que comienza por la clase *CElement* que contiene todos los datos de un elemento y la clase *SerialConfig* que contiene todos los datos de la configuración serial. A su vez existe una clase *Controller* que es la encargada de tomar los parámetros introducidos y enviarlos a la capa Modelo, luego, realiza las llamadas en dependencia de la solicitud del cliente y esos datos son enviados a la capa Vista.

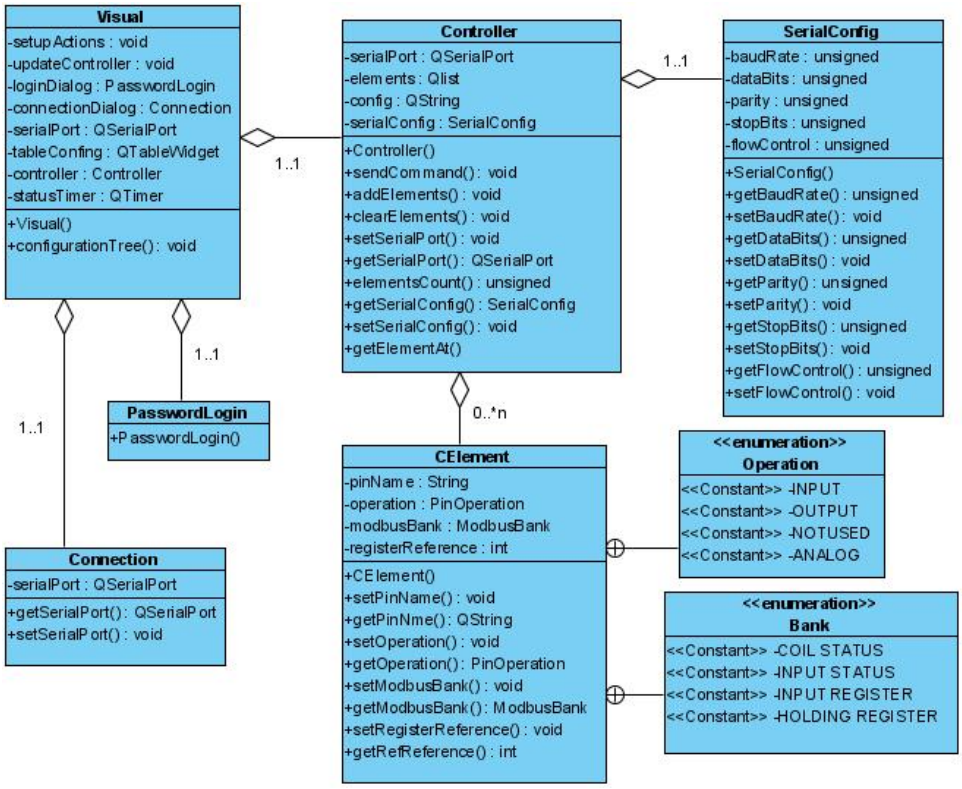

Figura 2.3. Diagrama de clase del modelo de datos

2.8. Conclusiones

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

CAPÍTULO 3

Implementación y pruebas

Este capítulo está dedicado a la implementación y ejecución de casos de prueba que evalúen las funcionalidades de la herramienta de configuración. Se definen los estándares de codificación que posibilitan la organización en la escritura del código en el proceso de desarrollo del software. Se realizan las iteraciones necesarias para cumplir satisfactoriamente los casos de pruebas de aceptación elaborados, obteniéndose finalmente los resultados de las pruebas realizadas.

3.1. Estilo de escritura

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

3.2. Estándares de Codificación

Un estándar de codificación agrupa los aspectos relacionados con la generación del código. Los programadores lo realizan para trabajar de forma coordinada dentro del proyecto, de acuerdo con el tipo de lenguaje que utilizan y las normas que propone el mismo, tratando de lograr un código claro y legible tanto para él como para otro programador (NETWORK, 2015).

A continuación se muestran algunos estándares de codificación que se tuvieron en cuenta para la implementación de la herramienta de configuración haciendo uso del lenguaje C++:

Identación: Se emplearon cuatro espacios como unidad de identación.

```
unsigned Controller::elementsCount()
{
    return elements.length();
}
```

Figura 3.1. Identación

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

```
QString CElement::getPinName() const
{
    return pinName;
}
```

Figura 3.2. Declaraciones

Controlador: Sentencia if, else: Se definió para estos dos tipos de sentencias la siguiente estructura: En el if y el else, se utilizó llaves para abrir y cerrar el bloque de acción (es) que se ejecutan dada(s) su (s) condición (es).

```
CElement *Controller::getElementAt(int index)
{
    if(index >= 0 && index < elements.length())
        return  elements.at(index);
    else
        return 0;
}
```

Figura 3.3. Controlador

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

Figura 3.4. Clase

3.3. Diagrama de paquete

Los paquetes se pueden utilizar para mostrar agrupaciones lógicas de objetos. Cada paquete agrupa un conjunto cohesivo de responsabilidades. Esta es la práctica básica de aplicar la modularidad para dar soporte a la separación de intereses (LARMAN, 2002).

Seguidamente se muestran estas agrupaciones lógicas de objetos y las dependencias entre ellos, evidenciando en el mismo el patrón arquitectónico seleccionado.

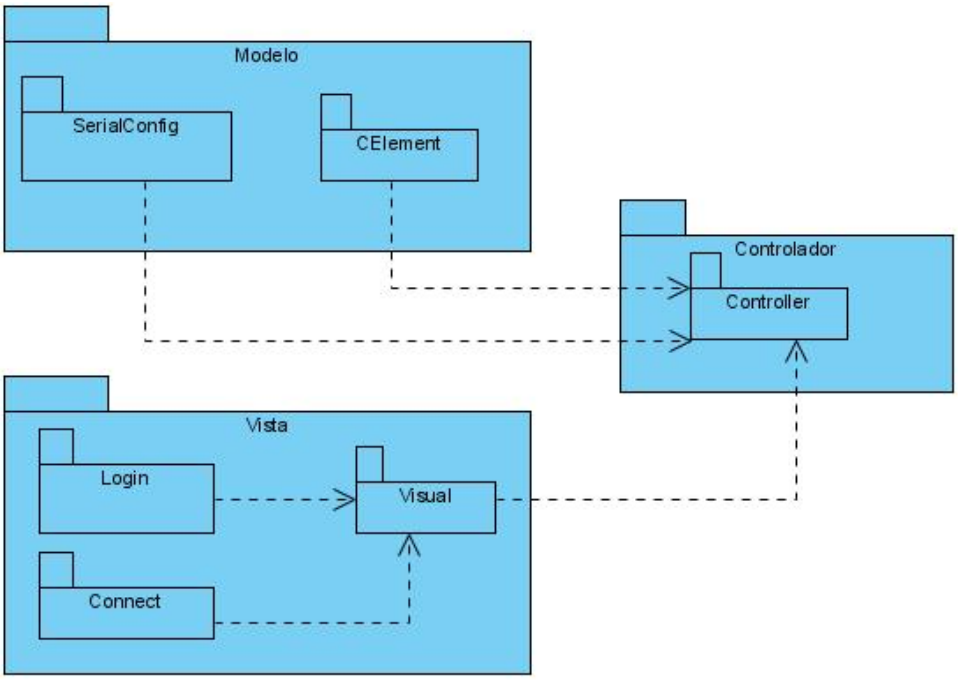

Figura 3.5. Vista de paquetes de la aplicación.

3.4. 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 (ibíd.).

Seguidamente se muestra una representación gráfica donde se encuentra la herramienta de configuración situada en la PC cliente, conectada a través del cable USB al dispositivo.

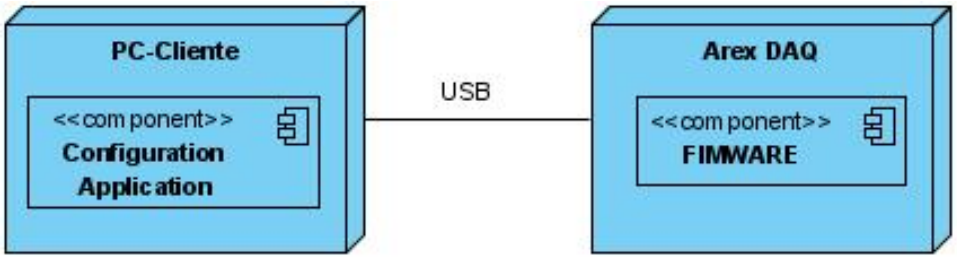

Figura 3.6. Vista de despliegue de la aplicación.

3.5. 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 (FIESTAS, 2014).

Por tanto las pruebas de software son un conjunto de actividades para la verificación del comportamiento de un programa y encontrar los posibles fallos de implementación o usabilidad dentro del software.

El método de prueba que se aplica a la herramienta de configuración es el de caja negra y consiste en que aquel elemento que es estudiando desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno; permitiendo examinar los aspectos funcionales del sistema haciendo mínimo enfoque en la estructura lógica interna del software. Esta técnica aplica pruebas de aceptación que son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Los clientes son responsables de verificar que los resultados de éstas pruebas sean correctos. Asimismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación. Dado que la responsabilidad es grupal, es recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta información (JOSKOWICZ, 2008).

3.6. Ambientes de pruebas

Las pruebas fueron realizadas en una PC con un microprocesador Core I3 a una frecuencia de 3.3 GHz, una memoria RAM de 4 Gb , sobre la distribución 14.04 de Ubuntu como sistema operativo.

3.7. Diseño de casos 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:

Caso de prueba de aceptación
Historia de usuario: Establecer comunicación entre la aplicación de configuración y la tarjeta de adquisición de datos.
Nombre: Establecer la conexión con el dispositivo.
Responsable: Rosabel Laches Hernández.
Descripción: Esta prueba se realiza con el objetivo de verificar que el usuario puede establecer la conexión con el dispositivo cuando se introducen valores válidos.
Condiciones de ejecución: La tarjeta se encuentra conectada correctamente a la computadora mediante el cable USB.
Entradas/Pasos de ejecución: Seleccionar la opción *Open Serial Port* del menú, configurar los parámetros y doy clic en OK.
Resultado esperado: La aplicación se conectó con el dispositivo y se muestra el mensaje *'Serial port is open'* en la barra de estado.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 3.1. Caso de prueba 1

Caso de prueba de aceptación
Historia de usuario: Establecer comunicación entre la aplicación de configuración y la tarjeta de adquisición de datos.
Nombre: Establecer la conexión con el dispositivo.
Responsable: Rosabel Laches Hernández.
Descripción: Esta prueba se realiza con el objetivo de verificar que la herramienta no se puede conectar con el dispositivo.
Condiciones de ejecución: La tarjeta se encuentra conectada correctamente a la computadora mediante el cable USB.
Entradas/Pasos de ejecución: Seleccionar la opción *Open Serial Port* del menú, configurar de forma incorrecta los parámetros del puerto serie y doy clic en OK.
Resultado esperado: La aplicación no se pudo conectar con el dispositivo y se muestra el mensaje *'Serial port is not open'* en la barra de estado.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 3.2. Caso de prueba 2

Caso de prueba de aceptación
Historia de usuario: Recibir la configuración del dispositivo.
Nombre: Leer datos del dispositivo.
Responsable: Rosabel Laches Hernández.
Descripción: Esta prueba se realiza con el objetivo de verificar que el usuario puede leer los datos del dispositivo.
Condiciones de ejecución: Conexión establecida con el dispositivo.
Entradas/Pasos de ejecución: Seleccionar la opción *Read Config* del menú.
Resultado esperado: El dispositivo envió su configuración y esta fue aceptada por el sistema.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 3.3. Caso de prueba 3

Caso de prueba de aceptación
Historia de usuario: Recibir la configuración del dispositivo.
Nombre: Leer datos del dispositivo.
Responsable: Rosabel Laches Hernández.
Descripción: Esta prueba se realiza con el objetivo de verificar que el usuario no pueda leer los datos del dispositivo.
Condiciones de ejecución: Conexión establecida con el dispositivo y el puerto serie no este abierto.
Entradas/Pasos de ejecución: Seleccionar la opción *Read Config* del menú.
Resultado esperado: El dispositivo no puede enviar su configuración y se muestra el mensaje *'Please use Open Serial Port menu option to open serial port connection'*.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 3.4. Caso de prueba 4

Caso de prueba de aceptación
Historia de usuario: Modificar los parámetros de configuración del dispositivo.
Nombre: Actualizar los parámetros de los campos *pinName, pinOperation, modbusBank y registerReference*.
Responsable: Rosabel Laches Hernández.
Descripción: Esta prueba se realiza con el objetivo de verificar que el usuario puede cambiar los valores de los campos de datos del dispositivo.
Condiciones de ejecución: Conexión establecida con el dispositivo.
Entradas/Pasos de ejecución: Seleccionar la opción *Read Config* del menú o la opción *Load Config*.
Resultado esperado: Se cambian los valores de configuración de los campos del dispositivo.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 3.5. Caso de prueba 5

Caso de prueba de aceptación
Historia de usuario: Modificar parámetros del puerto serie de configuración.
Nombre: Actualizar los parámetros de los campos *baudRate, dataBits, stopBits, parity y flowControl*.
Responsable: Rosabel Laches Hernández.
Descripción: Esta prueba se realiza con el objetivo de verificar que el usuario puede cambiar los valores de los campos de datos del puerto serie.
Condiciones de ejecución: Conexión establecida con el dispositivo.
Entradas/Pasos de ejecución: Seleccionar la opción *Read Config* del menú o la opción *Load Config*.
Resultado esperado: Se cambian los valores de configuración de los campos del puerto serie.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 3.6. Caso de prueba 6

Caso de prueba de aceptación
Historia de usuario: Enviar la nueva configuración de la tarjeta de adquisición de datos.
Nombre: Aplicar cambios realizados.
Responsable: Rosabel Laches Hernández.
Descripción: Esta prueba se realiza con el objetivo de verificar que el usuario no puede enviar la nueva configuración de la tarjeta de adquisición de datos.
Condiciones de ejecución: Conexión establecida con el dispositivo mediante el cable USB y el puerto serie no está abierto.
Entradas/Pasos de ejecución: Seleccionar la opción *Appy Changes* del menú.
Resultado esperado: No cambian los valores de configuración en la memoria interna del dispositivo y se muestra el mensaje *'Please use Open Serial Port menu option to open serial port connection'*.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 3.7. Caso de prueba 7

Caso de prueba de aceptación
Historia de usuario: Enviar la nueva configuración de la tarjeta de adquisición de datos.
Nombre: Aplicar cambios realizados.
Responsable: Rosabel Laches Hernández.
Descripción: Esta prueba se realiza con el objetivo de verificar que el usuario puede enviar la nueva configuración de la tarjeta de adquisición de datos.
Condiciones de ejecución: Conexión establecida con el dispositivo mediante el cable USB y el puerto serie este abierto.
Entradas/Pasos de ejecución: Seleccionar la opción *Appy Changes* del menú.
Resultado esperado: Se cambian los valores de configuración en la memoria EEPROM del dispositivo.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 3.8. Caso de prueba 8

Caso de prueba de aceptación
Historia de usuario: Salvar y cargar la configuración en un archivo.
Nombre: Cargar la configuración desde un archivo local.
Responsable: Rosabel Laches Hernández.
Descripción: Esta prueba se realiza con el objetivo de verificar si la funcionalidad que permite cargar la configuración del dispositivo procedente de un archivo local, funciona conforme a lo esperado.
Condiciones de ejecución: Debe existir algún archivo en alguna ubicación del directorio local que contenga la configuración del dispositivo.
Entradas/Pasos de ejecución: Seleccionar del menú la opción *Load Config* y localizar el archivo deseado.
Resultado esperado: Se carga la configuración del dispositivo procedente del archivo seleccionado.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 3.9. Caso de prueba 9

Caso de prueba de aceptación
Historia de usuario: Salvar y cargar la configuración en un archivo.
Nombre: Guardar los valores de configuración en un archivo.
Responsable: Rosabel Laches Hernández.
Descripción: Esta prueba se realiza con el objetivo de verificar si la funcionalidad de salvar la configuración del dispositivo en un archivo, funciona conforme a lo esperado.
Condiciones de ejecución: Capacidad de almacenamiento suficiente para guardar el archivo de configuración.
Entradas/Pasos de ejecución: Seleccionar del menú la opción *Save configuration* y ubicar en algún directorio el archivo creado.
Resultado esperado: Se crea el archivo con la configuración del dispositivo que fue salvada.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 3.10. Caso de prueba 10

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 posible y listo para entregar al cliente. Durante la ejecución de los casos de pruebas de aceptación algunas no arrojaron los resultados esperados. En la primera iteración se detectaron un total de tres no conformidades (Figura 3.6), de ellas, una para la HU 1, una para la HU 3 y una para la HU 6.

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.

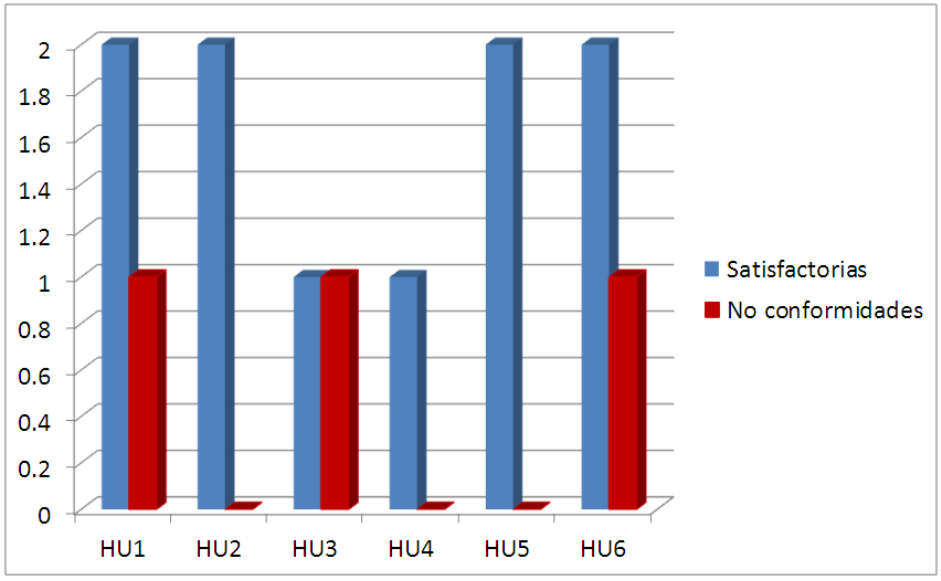

Figura 3.7. Resumen de defectos y dificultades

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 abarca todo lo referente a la estructura que contendrá el desarrollo del sistema para que sea homogénea. Los diagramas de paquete y de despliegue posibilitan un mejor entendimiento de cómo quedará el sistema. Por otra parte se realizaron las pruebas de aceptación a cada historia de usuario, que ofrecerán al cliente conformidad y seguridad ante las funcionalidades del sistema.

Conclusiones

Al término de la investigación se arriba a las siguientes conclusiones:

- Como resultado de todo este proceso de desarrollo se obtuvo una herramienta de configuración que cuenta con funcionalidades básicas que le permiten lograr la correcta configuración de la comunicación .
- Se obtuvo una aplicación multiplataforma desarrollada a partir del uso de tecnologías y herramientas de software libre contribuyendo así a la independencia tecnológica.

Recomendaciones

Al concluir la investigación se proponen, una serie de tareas para ampliar y dar continuidad al trabajo realizado después de analizar los resultados obtenidos en el proceso de investigación y desarrollo de la herramienta de configuración, se recomienda lo siguiente:

- Implementar el módulo de sensores para el área de la domótica en el Arduino.
- Adicionar un módulo para la comunicación por modbus TCP.

Glosario

Android Sistema operativo basado en el núcleo Linux, diseñado principalmente para dispositivos móviles con pantalla táctil. 9

AVR Son microcontroladores que poseen la arquitectura RISC y son desarrollados por la compañía estadounidense Atmel. 8

EEPROM Es un tipo de memoria que puede ser programada, borrada y reprogramada eléctricamente. 8, 10, 35

firmware Conjunto de instrucciones de un programa informático que se encuentra registrado en una memoria ROM, flash o similar. 1, 6

LGPL Es una licencia de software creada por la *Free Software Foundation* que pretende garantizar la libertad de compartir y modificar el software cubierto por ella. 17

UART Dispositivo que controla los puertos y dispositivos serie. Se encuentra integrado en la placa base o en la tarjeta adaptadora del dispositivo. 9

Acrónimos

CEDIN Centro de Informática Industrial. 1

ICID Instituto Central de Investigaciones Digitales. 1

MCU Microcontrolador. 1

SMD Sistema de Medición de Datos. 4

TIC Tecnologías de la Información y las Comunicaciones. 1

UCI Universidad de las Ciencias Informáticas. 1

Referencias bibliográficas

ARDUINO. 2014a. *Arduino MEGA ADK*. 2014. Disponible en: http://www.arduino.cc/en/Main/ArduinoBoardMegaADK.

ARDUINO. 2014b. *What is Arduino?* 2014. Disponible en: http://arduino.cc/en/guide/introduction.

ATMEL. 2014. *Microcontroladores*. 2014. Disponible en: http://www.atmel.com/pt/br/products/microcontrollers/default.aspx.

CANDELAS HERÍAS, Francisco Andrés. 2011. *Comunicación con RS-485 y MODBUS*. 2011.

CLARA HERNÁNDEZ, Anisbel e ISE MORALES, Delís. 2014. *Integración de herramientas de pruebas de seguridad para aplicaciones web en XILEMA- PlatSI*. 2014.

DELGADO ALÓN, Karel y JIMÉNEZ LÓPEZ, Alejandro. 2012. *Módulo HMI para una tarjeta basada en microcontroladores*. 2012.

FELIPEZ SÁNCHEZ, Carlos y CRUZ, Enrique. 2009. *Comunicación entre microcontroladores bajo el protocolo ZIGBEE*. 2009.

FERNÁNDEZ, Dr. Luis M. 2010. *Instrumentación Industrial*. 2010. Disponible en: http://www.icimaf.cu/cursos/luis/ii/Tema2.htm.

FIESTAS, Jhonattan. 2014. *Pruebas para asegurar la calidad del producto software (I)*. 2014. Disponible en: http://blog.elevenpaths.com/2014/09/qa-pruebas-para-asegurar-la-calidad-del.html.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph y VLISSIDES, John. 1997. *Design Patterns CD*. 1997.

GÓMEZ RUIZ, José Antonio. 2015. *Introducción al lenguaje C/C++*. 2015.

GÓMEZ VARGAS, Arianna. 2015. *Plan de Desarrollo de Software*. 2015.

GONZÁLEZ RODRÍGUEZ, Aliannys. 2013. *Aplicación de Configuración Web para Conversores de Protocolos*. 2013.

GROUP, Object Management. 2015. *Introduction to UML*. 2015. Disponible en: http://www.omg.org/gettingstarted/what_is_uml.htm.

IBARRA ALMAGUER, Daniel Iván. 2012. *Aplicación de configuración para la versión 2 de TETSCADA*. 2012.

INSTRUMENTS, Nationals. 2006a. *Adquisición de Datos*. 2006. Disponible en: http://www.ni.com/data-acquisition/what-is/esa/.

INSTRUMENTS, Nationals. 2006b. *Comunicación Serial*. 2006. Disponible en: http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1.

JOSÉ, Víctor y FERMÍN, Luna. 2008. *INGENIERÍA CONCEPTUAL Y BÁSICA DE UN SISTEMA AUTOMATIZADO PARA LA SUPERVISIÓN Y CONTROL DE LAS OPERACIONES DEL SISTEMA DE AMARRE DE UN SOLO PUNTO (MONOBOYA) DEL TERMINAL ORIENTE JOSE (TOJ) DE PDVSA REFINACIÓN*. 2008.

JOSKOWICZ, José. 2008. *Reglas y Prácticas en eXtreme Programming*. 2008.

LARMAN, Craig. 2002. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 2002.

NETWORK, Microsoft Developer. 2015. *Revisiones de código y estándares de codificación*. 2015. Disponible en: https://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx.

OLIVA LABAUT, Manuel Alejandro. 2012. *Firmware para tarjeta de adquisición basada en un microncontrolador STM32*. 2012.

PARRA NÁPOLES, Yoan. 2011. *Pasarela de datos natural sobre estándar TETRA.* 2011.

PRESSMAN, Roger S. 2005. *Ingeniería del software. Un enfoque práctico*. 2005.

REYES AGUILAR, Primitivo. 2007. *Análisis de los Sistemas de Medición (MSA)*. 2007.

SAN JUAN GUÍA, Lauren. 2012. *Herramienta de configuración para el PLC-HMI*. 2012.

SÁNCHEZ YERA, Anier y TORRES GONZÁLEZ, Hainet de los Angeles. 2013. *Conversor de protocolo Modbus TCP a Modbus RTU basado en el microcontrolador STM32F107VCT.* 2013.

SÁNCHEZ, Tamara Rodríguez. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI*. 2015.

SOMMERVILLE, Ian. 2005. *Ingeniería del software.* 2005.

VELASCO PÉREZ, Daimeris y VASCONCELO MIR, Yuniesky Orlando. 2010. *Desarrollo de la extensión para la configuración del módulo de adquisición de datos del SCADA Guardián del ALBA*. 2010.

WILMSHURST, Timothy. 2010. *Designing Embedded Systems with PIC Microcontrollers. Principles and applications*. 2010.

APÉNDICE A

Biblioagrafía consultada

BATISRA VILTRE, José Manuel. 2012. Herramienta para el monitoreo y análisis de los procesos del sistema SCADA Guardián del ALBA. 2012.

TORRES TORRITI, Miguel . 2011. TUTORIAL MICROCONTROLADORES PIC. 2011.

FLORES CORTEZ, Omar Otoniel . 2009. BATALLA DE MICROCONTROLADORES ¿AVR o PIC? . 2009.

FLORES CORTEZ, Omar Otoniel . 2009. BATALLA DE MICROCONTROLADORES ¿AVR o PIC? . 2009.

BLANCHETTE, Jasmin y SUMMERFIELD, Mark. 2006.C++ GUI Programming. 2006