



Facultad 4

Título: Mecanismo de configuración de dispositivos de adquisición de datos basados en Arduino que implementen el protocolo Modbus.

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

Autor: Lisardo Garcia Jane

Tutores: Ing. Jordanis Viltres Chávez

Ing. Alejandro García Santiesteban

Co-Tutor M. Sc. Luis Manuel Vidal Piña

La Habana, junio de 2018

“Año 60 de la Revolución”

Pensamiento

“En tiempos de cambio quienes estén abiertos al aprendizaje se adueñarán del futuro, mientras que aquellos que creen saberlo todo estarán bien equipados para un mundo que ya no existe.”

Eric Hoffer



Declaración de autoría

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis que tiene por título: “Mecanismo de configuración de dispositivos de adquisición de datos basados en Arduino que implementen el protocolo Modbus” 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 junio del 2018.

Lisardo Garcia Jane

Firma del Autor

Ing. Jordanis Viltres Chávez

Firma del Tutor

Ing. Alejandro García Santiesteban

Firma del Tutor

M. Sc. Luis Manuel Vidal Piña

Firma del Co-Tutor

Agradecimientos

A mi mamá por ser mamá y papí y haberme encaminado hasta aquí con mucho orgullo, esmero y dedicación, por las noches de desvelo y preocupación. Gracias a ti mamá por aquietar mis dudas y calmar mis miedos, por enseñarme a soñar. Agradezco a mi tía Pucha por ser mi segunda mamá, por consentirme y al mismo tiempo guiarme por el camino correcto, por el constante apoyo a mi mamá, por ser ejemplo, guía y estar ahí siempre. A mi hermana Lisandra, porque nada se compara con el amor de hermanos, por sacarme las mejores sonrisas en los momentos de enojo, gracias por tu complicidad, porque siempre estas cuando te necesito, porque no me imagino la vida sin tí. A mi Mada, que tanto quiero, por sus ocurrencias que siempre me hacen reír y ser una niña tan especial. A Devorat, que más que amiga, hermana, gracias por las peleas, los consejos, los alones de oreja cuando me costaba estudiar, apoyo en los momentos más difíciles de la carrera y en los buenos también, a tí siempre te tendré presente, te ganaste un lugar muy especial en mi corazón, te convertiste en una amiga para siempre. A mi tutor y amigo Jordanís gracias por estar conmigo en las buenas y malas en estos cinco años y haberme permitido ser parte de tu vida, por ser parte indisoluble de mi existencia. Sabes que estés donde estés podrás contar conmigo.

Agradezco a mi familia por todo el apoyo que me han brindado siempre, no puedo desear una mejor. A mis tutores por las noches de desvelo, la dedicación y entrega. A mis amigos de la universidad que siempre responden "Estoy aquí". A mis profesores, especialmente al profe Omar, que nunca dejó de ser nuestro guía. A mis compañeros de apartamento que durante estos años nos hemos convertido en una familia y con los que siempre podré contar. A mis compañeros de aula por el apoyo en las horas de estudio y los buenos momentos que pasamos juntos que, por mi parte, nunca serán olvidados.

Dedicatoria

A la memoria de mi abuelo...

No existe una única forma de enseñar y de amar, no existe un manual que lo explique, pero justo ahí está la belleza de ser madre. A ti, mamá, dedico este triunfo, porque me enseñaste a volar sin imponerme un camino, por ser una persona íntegra y ejemplar. Por ser creativa, amiga, compañera, confidente. Por ser madre y padre a la vez, porque, aunque sé que es difícil siempre estuviste fuerte, pero también hubo lágrimas y dolor porque ser fuerte duele. Por esas horas de estudio que estuviste a mi lado para que este sueño, que hoy cumplimos los dos, se convirtiera en realidad.

RESUMEN

El proceso de configuración de los parámetros de ejecución de dispositivos de adquisición de datos (DAQ) basados en Arduino¹, que implementen el protocolo industrial de comunicación Modbus, es una tarea que por su alto grado de complejidad durante el proceso de implementación demanda un elevado costo de tiempo. La presente investigación, a partir de la situación problemática identificada y con la utilización de métodos de investigación teóricos y empíricos, así como el estudio de diferentes familias de herramientas de configuración de dispositivos de adquisición con Arduino, agrupadas de esta manera por su gran variedad; describe la implementación de una nueva solución. Se aplicó la “Metodología de desarrollo para la Actividad productiva de la UCI”, permitiendo detallar el diseño e implementación de los componentes. La solución desarrollada permite reducir el tiempo de desarrollo y despliegue de proyectos que hagan uso de uno o varios dispositivos de adquisición de datos.

Palabras claves: arduino, configuración, dispositivo de adquisición de datos, herramientas, modbus.

¹ Plataforma de prototipos electrónica de código abierto (*open-source*) basada en *hardware* y *software* flexibles y fáciles de usar.

Índice de contenido

ÍNDICE DE CONTENIDO

INTRODUCCIÓN	1
CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA	6
1.1 Conceptos asociados al objeto de estudio del problema	6
1.1.1 Dispositivos de Adquisición de Datos	6
1.1.2 Protocolo industrial Modbus	7
1.1.3 Modelo de datos Modbus	8
1.1.4 Comunicación Serial	8
1.1.5 Comunicación TCP	9
1.2 Plataforma Arduino	10
1.3 Herramientas de configuración de DAQ basados en Arduino	11
1.3.1 Entornos de Desarrollo Integrado	12
1.3.2 Herramientas gráficas	13
1.3.3 Herramientas en línea	14
1.3.4 Plugins y Extensiones	14
1.4 Metodología de desarrollo	15
1.5 Herramientas, tecnologías y lenguajes de desarrollo	16
1.5.1 Lenguaje de programación	16
1.5.2 Tecnología de desarrollo	17
1.5.3 Entorno Integrado de Desarrollo	17
1.5.4 Herramienta de modelado	17
1.5.5 Lenguaje de modelado	18
1.5.6 Sistema Gestor de Base de Datos	18
1.5.7 Dependencias	18
1.6 Conclusiones parciales del capítulo	19

Índice de contenido

CAPÍTULO 2 ANÁLISIS Y DISEÑO	20
2.1 Modelo de dominio	20
2.2 Descripción de las clases de dominio	20
2.3 Especificación de requisitos.....	21
2.3.1 Requisitos funcionales.....	22
2.3.2 Requisitos no funcionales	22
2.4 Propuesta de solución	23
2.5 Historias de usuario	24
2.6 Descripción de las HU	25
2.7 Descripción de la arquitectura.....	28
2.7.1 Patrón arquitectónico.....	28
2.8 Patrones <i>GRASP</i>	30
2.9 Patrones <i>GoF</i>	31
2.10 Modelo de diseño	31
2.10.1 Diagrama de paquetes.....	32
2.10.2 Diagrama de clases	32
2.10.3 Modelo de despliegue.....	33
2.11 Validación del diseño	34
2.12 Conclusiones parciales del capítulo	43
CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS AL SISTEMA	44
3.1 Estándar de codificación.....	44
3.2 Pruebas al sistema	46
3.2.1 Pruebas de Caja Negra	47
3.2.2 Partición de Equivalencia.....	47
3.3 Conclusiones parciales del capítulo	58

Índice de contenido

CONCLUSIONES GENERALES	60
RECOMENDACIONES	61
REFERENCIAS BIBLIOGRAFICAS	62
ANEXOS.....	67

Índice de tablas

ÍNDICE DE TABLAS

Tabla 1. Tipos de datos Modbus.	8
Tabla 2. HU1 Crear un nuevo proyecto de configuración.	25
Tabla 3. HU7 Modificar los datos configurados de un sensor.	26
Tabla 4. HU17 Subir código Arduino a la placa conectada.	27
Tabla 5. Tamaño operacional de clase (TOC).	35
Tabla 6. Rango de valores para la evaluación técnica del TOC.	35
Tabla 7. Relación entre clases (RC).	39
Tabla 8. Rango de valores para la evaluación técnica de RC.	39
Tabla 9. Caso de prueba “Crear en un nuevo proyecto de configuración”.	47
Tabla 10. Caso de prueba “Adicionar sensor”.	48
Tabla 11. Caso de prueba “Modificar los datos configurados de un sensor”.	50
Tabla 12. Caso de prueba “Subir código Arduino a la placa conectada”.	52
Tabla 13. Resultados de la primera iteración de pruebas.	54
Tabla 14. Resultados de la segunda iteración de pruebas.	55
Tabla 15. Resultados de la tercera iteración de pruebas.	57

Índice de imágenes

ÍNDICE DE IMÁGENES

Figura 1. Modelo de Dominio.	20
Figura 2. Esquema de funcionamiento del patrón MVC.	29
Figura 3. Diagrama de paquetes.	32
Figura 4. Diagrama de clases.	33
Figura 5. Diagrama de Despliegue.	34
Figura 6. Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.	37
Figura 7. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad	37
Figura 8. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de implementación.	38
Figura 9. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.	38
Figura 10. Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.	40
Figura 11. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.	41
Figura 12. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de mantenimiento.	41
Figura 13. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de pruebas.	42
Figura 14. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.	42
Figura 15. Indentación con 4 espacios.	45
Figura 16. Líneas largas fraccionadas después de la coma.	45
Figura 17. Declaraciones.	46
Figura 18. Sentencias	46

INTRODUCCIÓN

Desde el surgimiento de las Tecnologías de la Información y las Comunicaciones (TIC), es creciente la utilización de las mismas en todos los ámbitos de las sociedades para mejorar la calidad de vida, aumentando entre otros aspectos, el confort y la seguridad. Hoy en día es difícil encontrar un proceso que no esté vinculado a éstas, pues han venido a revolucionar la forma de vida de los humanos por las ventajas que presentan en los diferentes sectores de la sociedad, entre ellos el sector industrial, donde los procesos industriales se han visto beneficiados con la implantación de aplicaciones informáticas para el control de los mismos (Almaguer 2012).

Si bien puede parecer una declaración arriesgada, hay que tener en cuenta que Internet es una de las creaciones más importantes de toda la historia de la humanidad y con ella, el surgimiento del Internet de las Cosas (IoT²) (Evans 2011). Elementos como el *hardware*, sensores, actuadores³, dispositivos que controlan los sistemas y otros sistemas de comunicación alojados en los objetos, son los que conforman el IoT. Los dispositivos y redes proporcionan conectividad física realizando aportes a la domótica, esta última como parte esencial en el IoT, lo que ayuda en la gestión inteligente de hogares y edificios (González 2015).

La domótica constituye un conjunto de tecnologías aplicadas al control y la automatización inteligente de la vivienda, que permite una gestión eficiente del uso de la energía, aportando además, seguridad y confort, y al mismo tiempo comunicación entre el usuario y el sistema (CEDOM 2017). Consiste en introducir infotecnología⁴ en los hogares para mejorar la calidad de vida de las personas y ampliar sus posibilidades de comunicación a partir de la automatización de diferentes procesos (Pesce 2014).

Existen en el mercado una gran variedad de dispositivos orientados a las tareas de adquisición de datos y monitoreo de variables físicas ambientales. Estos dispositivos son capaces de trabajar en forma distribuida, generando gran variedad de topologías de redes para comunicarse entre ellos mediante protocolos de comunicación estándares como: RS-485, HART, Profibus, *Ethernet*, Modbus entre otros (Cázarez-Ayala, Castillo-Meza y Fonseca-Beltrán 2012). Entre estos se encuentran los dispositivos de adquisición de datos, que se encargan, entre otras cosas, de convertir las señales que reciben del entorno en que se encuentran de analógicas a digitales (A/D) o digitales a analógicas (D/A).

2 *Internet of Things*.

3 Dispositivo capaz de ejecutar algún cambio físico dada una señal eléctrica.

4 Término general que describe cualquier tecnología que ayuda a producir, manipular, almacenar, comunicar, y/o esparcir información.

Introducción

En el Centro de Informática Industrial (CEDIN), de la Universidad de las Ciencias Informáticas (UCI), se desarrollan aplicaciones orientadas a la Domótica; entre ellas el Sistema Domótico, el cual está compuesto por varios módulos: Editor, Recolector, Visualizador y Base de datos Históricas. El módulo Recolector es el encargado de comunicarse con dispositivos de adquisición de datos mediante el protocolo Modbus, recolectar la información obtenida por los sensores conectados a estos, accionar los actuadores correspondientes, y por último brindar acceso a los clientes que lo necesiten.

En el Sistema Domótico del CEDIN se hace uso de dispositivos de adquisición de datos implementados con placas Arduino utilizando el protocolo industrial Modbus. A estos dispositivos se conectan un conjunto de sensores, actuadores, y algún módulo de *hardware* para la comunicación con aplicaciones externas si es necesario; a cada uno de estos les corresponde una porción en el código fuente que conforma el *firmware*⁵, que se embebe en la placa Arduino utilizando algún entorno de desarrollo integrado (*IDE*, por sus siglas en inglés); el único utilizado es el Arduino *IDE*.

La implementación del protocolo de comunicación Modbus, y el uso de algunos sensores, requieren de la utilización de algunas bibliotecas de clases en lenguaje Arduino, que se le deben incorporar al Arduino *IDE* como extensiones. La creación de objetos, la configuración de los mismos, y la llamada de sus métodos no siguen un estándar y requieren de personal experimentado. En cuanto a la biblioteca utilizada para Modbus, esta permite la configuración de un Servidor Modbus, los bloques de memoria Modbus, y la lectura y escritura de los mismos desde código Arduino (que hay que escribir) o debido a una solicitud recibida desde una aplicación externa. Para algunos sensores es necesario realizar un ajuste del valor obtenido, antes de enviarlo al en un registro de un bloque de memoria; y de igual manera cuando se quiere enviar una señal a un actuador, a veces es requerido hacer un ajuste del valor leído de un registro antes de escribirlo en el pin donde se conecta el actuador. Una vez finalizada la implementación de un dispositivo de adquisición de datos (DAQ) con Arduino, se necesita probarla mediante alguna aplicación externa que haga de Cliente Modbus, comprobando que la placa responde las solicitudes de lectura o escritura en los bloques de memoria, que cada sensor se lee apropiadamente y que los actuadores reciben la señal necesaria en el momento esperado. Agregarle a todo esto que el código Arduino obtenido para estos dispositivos no sigue actualmente un estándar de codificación, ni es necesariamente óptimo.

En resumen, la implementación de DAQ con Arduino requiere:

⁵ Programa informático que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo.

Introducción

- Experiencia en lenguaje Arduino.
- Conocimiento de cómo incorporar y utilizar bibliotecas implementadas por terceros para Modbus y disímiles sensores.
- Conocimiento de cómo calibrar el valor obtenido por un sensor determinado o la señal enviada a un actuador.
- Detalles técnicos de como conectar los sensores y actuadores a la placa Arduino.
- Detalles técnicos de qué pines son usados por algún módulo de *hardware* adicional que sea necesario conectar a la placa para soportar la variante de comunicación Modbus a implementar.
- Considerable tiempo para realizar implementaciones variadas, que pueden estar sujetas a errores humanos o errores funcionales difíciles de detectar.
- Conocimiento del uso de aplicaciones externas para realizar pruebas funcionales al dispositivo para depurar errores de implementación.

A partir del análisis de la situación problemática anteriormente planteada, se define el siguiente **problema de investigación**: ¿cómo realizar la configuración de dispositivos de adquisición de datos sobre plataformas Arduino que implementen el protocolo Modbus?

Se identifica, como **objeto de estudio**: las herramientas o métodos de configuración y programación de dispositivos de adquisición de datos sobre plataformas Arduino.

Para dar solución al problema planteado, se define como **objetivo general** de la investigación: desarrollar una herramienta que permita configurar dispositivos de adquisición de datos sobre plataformas Arduino que implementen el protocolo Modbus.

Se reconoce como **campo de acción** del presente trabajo: las herramientas de configuración de dispositivos de adquisición de datos sobre plataformas Arduino, que implementen el protocolo Modbus.

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

1. Descripción del estado del arte sobre el tema de investigación.
2. Caracterización del protocolo Modbus.
3. Caracterización de las plataformas Arduino y sus protocolos de comunicación.
4. Elección de los estándares, protocolos de comunicación, metodologías, herramientas y tecnologías a usar para el desarrollo de la solución propuesta.
5. Generación de los artefactos relacionados con el análisis y diseño de la solución propuesta.

Introducción

6. Definición de la arquitectura del sistema.
7. Implementación de la solución propuesta.
8. Pruebas y corrección de errores de la solución propuesta.

Con el objetivo de alcanzar la solución deseada y de facilitar el desarrollo de la investigación se combinaron diferentes **métodos de investigación**, los fundamentales se mencionan a continuación:

Métodos Teóricos:

- **Analítico-Sintético:** este método es usado para realizar un análisis de libros, sitios web, documentos electrónicos y otras fuentes bibliográficas que relacionan elementos relevantes para la investigación como las características de los dispositivos de adquisición de datos, el protocolo de comunicación Modbus y los principales elementos de la plataforma Arduino.
- **Histórico-Lógico:** este método es empleado en la investigación para realizar una sistematización de las herramientas o métodos de configuración o programación de dispositivos de adquisición de datos sobre plataformas Arduino.

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 presente investigación está estructurada como se presenta a continuación:

- **Capítulo 1.** Fundamentación teórica: este capítulo está dedicado a los principales conceptos asociados a la problemática, se realiza un análisis de la metodología de desarrollo de *software*, 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.** Análisis y Diseño: el 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 y estructura de clases del sistema desarrollado, también en el capítulo se describen los requisitos funcionales y no funcionales del sistema.
- **Capítulo 3.** Implementación y prueba: en este capítulo se presenta los estándares de codificación utilizados para el desarrollo de la solución. Se describen las pruebas realizadas al sistema con el

Introducción

objetivo de validar que funcione de manera correcta y cumpla con las especificaciones planteadas. Además, se describen las no conformidades detectadas al realizar las pruebas.

Capítulo 1. Fundamentación teórica

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se aborda un grupo de conceptos de vital importancia para el desarrollo de la investigación, destacándose las características de los diferentes dispositivos de adquisición de datos, las herramientas similares a la que se pretende desarrollar, y algunas características de la plataforma Arduino. Se expone el estudio realizado sobre la comunicación por puerto Serie y el protocolo de control de transmisión (*TCP*), así como las principales características del protocolo industrial Modbus. Se describen las herramientas y metodología que serán 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 una solución aplicable al problema científico planteado.

1.1 Conceptos asociados al objeto de estudio del problema

1.1.1 *Dispositivos de Adquisición de Datos*

El acelerado ritmo de competitividad de las industrias es uno de los principales factores que provocan la rápida evolución y crecimiento de las tecnologías electrónicas y las comunicaciones. La creciente necesidad de desarrollar nuevos productos con la mejor calidad, en forma rápida y más económica para la medición, monitoreo y el control de los diferentes procesos que se llevan a cabo en el sector industrial, trajo consigo la aparición en el mercado de los DAQ, los cuales están extensamente difundidos en este sector por sus amplias funcionalidades.

La adquisición de datos es la utilización de dispositivos para capturar datos, almacenarlos, procesarlos y presentarlos y de alguna forma que puedan ser manipulados por un ordenador u otro dispositivo electrónico. Los DAQ están ampliamente difundidos no solo en la industria sino también en otros sectores de impacto en la sociedad como la salud y la educación. Existen en el mercado una gran cantidad de dispositivos para la adquisición de datos, todos ellos con diferentes fines y características, pero con igual importancia dentro del Sistema de Adquisición de Datos (SAD) para el que fueron diseñados. Los SAD además están compuestos por sensores (temperatura, presión, humedad) y una *PC* que se usa para programar estos dispositivos (CO2 Measurement Specialists 2015).

Un DAQ es el *hardware* que funciona como interfaz entre las señales físicas del medio y las computadoras a través de los diferentes sensores. Sirven para obtener una muestra de una variable física y transformarla en un dato que pueda ser reconocido y registrado por un sistema digital. Pueden ser utilizados en todo tipo de industrias y centros de investigación científica, ya que facilitan la realización de una serie de mediciones y el control sobre las condiciones de trabajo.

Capítulo 1. Fundamentación teórica

Los DAQ están compuestos por tres elementos claves, en primer lugar, el **acondicionador de señales** que es el encargado de manipular la señal que se recibe del medio y que sea apropiada para un **convertidor analógico-digital** que es el chip que brinda una representación digital de la señal recibida, las cuales son enviadas a la *PC* a través de un **bus** que sirve como interfaz de comunicación entre los DAQ y la *PC* para la transferencia de datos e instrucciones. Todos estos componentes son controlados por un *software* embebido llamado *firmware* que comprende rutinas de instrucciones almacenados en la memoria ROM de los dispositivos y define la lógica del funcionamiento de un DAQ.

El *firmware* de un DAQ puede ser escrito en varios lenguajes, utilizando *Assambler* (que es un lenguaje de bajo nivel) o C, C++, Basic o Forth. Se puede obtener un *firmware* completamente funcional usando herramientas como RTOS⁶ o Arduino *IDE* si se quiere utilizar placas Arduino. Teniendo en cuenta las características de un DAQ, se plantea la posibilidad del uso de Arduino y el protocolo industrial Modbus como una opción para la recolección de datos, al ser esta tecnología de fácil adquisición, por ser bajo en costos, además de disponer de pines de entrada y/o salida digitales y analógicos, y variados medios de comunicación (Serial, *TCP*, Bluetooth, WIFI, etc.)

1.1.2 Protocolo industrial Modbus

Modbus es un protocolo de comunicación desarrollado por la empresa Modicon. Fue diseñado como un protocolo de solicitud y respuesta con un modelo flexible de datos y funciones; características que son parte de la razón por la que hoy en día aún sigue en uso, además cuenta con un modelo de datos estructurado en forma de bloques que le dan robustez al protocolo y forman parte de los diferentes tipos de tramas que usa Modbus para el envío y recepción de datos, utilizando cualquiera de los canales de comunicación implementados por Modbus. En términos simples, es un método utilizado para transmitir información entre dispositivos electrónicos. Originalmente implementado como un protocolo para transferir datos en una capa serial, este se ha expandido para incluir implementaciones a través de los canales de comunicación Serial, *TCP/IP*, UDP⁷ entre otros.

Modbus es un protocolo abierto, lo que significa que es gratuito para que los fabricantes lo incorporen a sus equipos sin necesidad de pago alguno. Se ha convertido en un protocolo de comunicaciones estándar en la industria, y ahora es el medio más frecuentemente usado para conectar dispositivos electrónicos

⁶ Sistema operativo de tiempo real

⁷ *User Datagram Protocol* es un protocolo del nivel de transporte basado en el intercambio de datos.

Capítulo 1. Fundamentación teórica

industriales. Está basado en la arquitectura maestro/esclavo con acceso al medio controlado por el maestro, diseñado para trabajar con PLC⁸.

1.1.3 Modelo de datos Modbus

Modbus basa su modelo de datos en una serie de bloques, cada uno con sus propias características. En la siguiente tabla se describe el comportamiento de cada uno.

Tabla 1. Tipos de datos Modbus.

Bloque de memoria	Tipo de dato	Acceso de Maestro	Descripción
Coils	Booleano	Lectura/Escritura	Registros de 1 <i>bit</i> que pueden ser leídos y escritos por el maestro
Input status	Booleano	Solo Lectura	Registros de 1 <i>bit</i> que solo pueden ser leídos por el maestro
Holding Registers	Palabra sin signo	Lectura/Escritura	Registros de 16 <i>bits</i> que pueden ser leídos y escritos por el maestro
Input Registers	Palabra sin signo	Solo Lectura	Registros de 16 <i>bits</i> que solo pueden ser leídos por el maestro

1.1.4 Comunicación Serial

La comunicación serial es un protocolo muy común para la comunicación entre dispositivos, es incluido de manera estándar en prácticamente cualquier computadora. Este tipo de comunicación es comúnmente utilizado por varios dispositivos para instrumentación, y también para la adquisición de datos si se usa en conjunto con un dispositivo remoto de muestreo (National Instruments 2006).

Típicamente, la comunicación serial se utiliza para transmitir datos en formato ASCII y puede ser de manera sincrónica o asincrónica. Usualmente la transmisión se realiza de forma asincrónica, ya que es posible enviar y recibir datos al mismo tiempo. Las características más importantes de la comunicación serial son la velocidad de transmisión, los *bits* de datos, los *bits* de parada, y la paridad, por lo que para que dos puertos se puedan comunicar, es necesario que estas características sean iguales.

⁸ Controladores Lógicos Programables, del inglés Programmable Logic Controller,

Capítulo 1. Fundamentación teórica

A continuación, se relacionan las características de la comunicación serial antes mencionadas (National Instruments 2006):

Velocidad de transmisión (*baud rate*): indica el número de *bits* que se transfieren por segundo, y se mide en baudios (*bauds*). Es posible utilizar velocidades de transmisión altas, pero se reduciría la distancia máxima posible entre los dispositivos, por lo que para lograr altas velocidades los dispositivos deben estar uno junto al otro.

Bits de datos: se refiere a la cantidad de *bits* en la transmisión. El número de *bits* que se envía depende en el tipo de información que se transfiere. El número de *bits* depende del protocolo que se seleccione.

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, es posible que los 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. Mientras más *bits* de parada se usen, mayor será la tolerancia a la sincronía, sin embargo, la transmisión será 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. La opción de no usar paridad alguna también está disponible. Permite al dispositivo receptor conocer de antemano el estado de un *bit*, lo que ayuda a determinar si hay ruido que esté afectando de manera negativa la transmisión de los datos, o verificar el estado de sincronización de los dispositivos.

1.1.5 Comunicación TCP

TCP es un protocolo de comunicación orientado a conexión considerablemente extendido en Internet. Es usado para lograr una mejor comunicación por diferentes aplicaciones de red como ftp, telnet, acceso Web, etc. La función principal del nivel de transporte dentro de la arquitectura de protocolos *TCP/IP* es la de permitir la comunicación entre dos aplicaciones o dispositivos de forma económica y fiable.

Con el uso del protocolo *TCP*, las aplicaciones pueden comunicarse en forma segura, gracias al sistema de acuse de recibo del mencionado protocolo. *TCP* establece una arquitectura cliente/servidor facilitando la comunicación de forma bidireccional. Además, está apto para controlar la velocidad de los datos ya que es posible enviar y recibir mensajes de tamaño variable. Es uno de los protocolos de comunicación más importantes, su nombre representa al conjunto de protocolos que conforman la arquitectura formada por cinco niveles o capas, estas son: aplicación, transporte, internet, físico y red (Estrada Corona 2004):

Capítulo 1. Fundamentación teórica

Entre las principales características de *TCP* están: (Naranjo 2016)

- Los datos que envíe una aplicación a otra en otra máquina llegarán seguros (recuperación ante pérdidas).
- Si la aplicación envía varios bloques de información, éstos llegarán en el mismo orden en que se enviaron (mantiene el orden de secuencia).
- Antes de poder enviar datos, hay que establecer una conexión.
- Especificar cuáles son las aplicaciones y cuáles son las máquinas que realizarán la comunicación (orientado a conexión).
- Ambos extremos de la conexión pueden enviar información al otro extremo (dúplex completo).
- Intenta no congestionar la red.

1.2 Plataforma Arduino

Arduino es una plataforma de código abierto y *hardware* libre, formado por una placa con un sencillo microcontrolador y un entorno de desarrollo para realizar sus programas, que posibilita recibir información de varios elementos externos y accionar sobre otros.

Las placas Arduino están habilitadas para leer valores de diferentes sensores y convertir valores en una salida útil que pueda ser usada para diferentes fines. El crecimiento de Arduino ha dado al traste al surgimiento alrededor de esta plataforma tan flexible y adaptable a varios entornos toda una comunidad de fabricantes (estudiantes, aficionados, artistas, programadores y profesionales). Sus contribuciones se han añadido a una increíble cantidad de conocimiento accesible que puede ser de gran ayuda para principiantes y expertos por igual.

Arduino sólo no constituye un elemento relevante en ninguna investigación, su importancia radica en la conexión a éste de objetos capaces de interactuar entre sí, con algún mecanismo mediante el cual estos dispositivos puedan percibir el mundo que los rodea o ejecutar órdenes. Además, es importante tener en cuenta los mecanismos de comunicación que soporta para intercambiar información, entre ellos la comunicación Serial y *TCP* que son dos de los protocolos más usados para la comunicación entre dispositivos y aplicaciones.

El gran éxito de esta plataforma en parte se debe a que permite programar sin tener grandes conocimientos de programación en general y brinda unas herramientas sencillas y específicas para programar los microcontroladores que suministra en sus placas. Arduino brinda una amplia gama de posibilidades para

Capítulo 1. Fundamentación teórica

realizar disímiles aplicaciones y resolver muchísimos problemas facilitando en gran medida la cotidianidad del hogar y el trabajo.

Arduino ofrece algunas ventajas sobre otros sistemas entre las que se encuentran su capacidad de facilitar el trabajo con microcontroladores y además (Arduino 2017):

1. **Económico:** las placas Arduino son relativamente económicas en comparación con otras plataformas de microcontroladores.
2. **Multiplataforma:** el *software* Arduino *IDE* se ejecuta en sistemas operativos Windows, Mac OS X y *Linux* convirtiéndose en una ventaja sobre la mayoría de los sistemas de microcontroladores que están limitados a Windows.
3. **Entorno de programación sencillo y claro:** Arduino (*IDE*) es fácil de usar para principiantes, sin embargo, es lo suficientemente flexible como para que los usuarios avanzados puedan aprovecharlo también.
4. **Software de código abierto y extensible:** el *software* Arduino se publica como herramientas de código abierto, disponibles para extensión por programadores experimentados. El lenguaje puede expandirse a través de bibliotecas C++ que son añadidas al *IDE* de manera conveniente según lo que se desea implementar.
5. **Hardware de código abierto y extensible:** los planos de los módulos Arduino se publican bajo una licencia de *Creative Commons*⁹, por lo que los diseñadores experimentados de circuitos pueden hacer su propia versión del módulo, ampliarlo y mejorarlo. Incluso los usuarios relativamente inexpertos pueden construir la versión del módulo para comprender cómo funciona.

1.3 Herramientas de configuración de DAQ basados en Arduino

Luego de definir y describir los principales conceptos relacionados con la presente investigación científica, se hace un estudio acerca de las diferentes herramientas de configuración de DAQ basados en Arduino, atendiendo particularmente la forma en la que realizan la configuración, agrupándolas en tres familias: Entorno de Desarrollo Integrado, Herramientas Gráficas, Herramientas en línea y *Plugins* y Extensiones:

⁹ En español significa Comunes Creativos es una organización que permite usar y compartir tanto la creatividad como el conocimiento a través de una serie de instrumentos jurídicos de carácter gratuito.

Capítulo 1. Fundamentación teórica

1.3.1 Entornos de Desarrollo Integrado

Existen muchas herramientas para programar y configurar DAQ usando Arduino como tecnología. Entre las más conocidas están:

- **Arduino IDE:** es el *IDE* oficial de Arduino, 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 (GUI). Además, incorpora las herramientas para cargar el programa ya compilado en la memoria flash del *hardware* (Arduino 2015).
- **Programino:** es completamente compatible con el *software* de Arduino *IDE*, constituye una alternativa fácil y cómoda al *IDE* para la plataforma Arduino. Tiene todo lo que se necesita para programar un Arduino *Sketch*. No es una herramienta libre de pago por lo que para su adquisición se debe pagar una licencia (Rivera et al. 2017).
- **ArduIDE:** es un *IDE* para la plataforma de prototipos de electrónica de Arduino de código abierto basado en Qt¹⁰. Utiliza la base del *software* original de Arduino, pero siendo éste, más rápido transformándola en una herramienta mucho más cómoda y completa (Peres y Lazaro 2012).
- **CodeBlocks Arduino IDE:** es un editor de código fuente, herramientas de construcción y un depurador. Se trata de un entorno de desarrollo gratuito compatible con varios compiladores distintos, que puede utilizarse en diversos sistemas operativos. Gracias a los numerosos *plugins*¹¹ y opciones, es plenamente configurable (Triana Barreda 2017).

La forma habitual de trabajar es recurrir al propio entorno de desarrollo de Arduino, por su sencillez y facilidad de uso. Estas herramientas facilitan la forma de escribir código y subirlo a una placa Arduino. Algunas de ellas son multiplataforma ya que están disponibles para *Windows*, *Mac OS X* y *Linux*. Estas, entre otras funcionalidades, permiten detectar automáticamente la placa conectada, visualizar valores recibidos del medio a través del monitor serie. La comunicación con el Arduino se realiza por medio del canal Serie mediante cable *USB*¹². Además de las antes mencionadas existen un grupo de *Plugins* o Bibliotecas como *Plugin* de Arduino para NetBeans¹³, *Plugin* de Arduino para Eclipse, *Visual Studio Code Extension for Arduino* que integradas a otros *IDE* de desarrollo es posible usar estos para implementar y configurar un DAQ. A pesar de las ventajas que brindan, no proporcionan un entorno gráfico capaz de

10 Qt es un framework multiplataforma orientado a objetos

11 es una aplicación (o programa informático) que se relaciona con otra para agregarle una función nueva

12 Bus Universal en Serie, del inglés *Universal Serial Bus*.

13 NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java.

Capítulo 1. Fundamentación teórica

configurar placas Arduino sin utilizar lenguajes de programación, además se debe tener conocimiento de las bibliotecas de terceros necesarias para lograr las configuraciones ineludibles para obtener un DAQ y dónde ubicarlas para que funcionen, así como lo complejo y costoso en tiempo que representa implementar un DAQ pudiendo ser introducidos errores humanos a la hora de programar (Arduino 2017).

1.3.2 Herramientas gráficas

- **Visuino:** es el último *software* innovador de Mitov Software¹⁴. Proporciona un entorno de programación visual que le permite programar sus placas Arduino. Los componentes que se encuentran en este *software* representan componentes de *hardware* posibilitando crear y diseñar fácilmente sus programas con solo arrastrarlos y soltarlos.
- **Arduviz:** es un entorno de desarrollo integrado de programación visual que tiene características como la generación de código y un entorno de desarrollo independiente. El lenguaje utilizado es basado en el enfoque de programación visual y lleva por nombre “lenguaje de programación visual” (Pratomo y Perdana 2017).
- **Ardublock:** se trata de una utilidad gráfica cuya misión es generar código compatible con el entorno *IDE* Arduino, es gratuita, posee una colección de bloques funcionales básicos que facilitan la comprensión de la programación (Candel Navas 2013).
- **S4A (Scratch for Arduino¹⁵):** se trata de un proyecto de Citilab¹⁶ y tiene el aval de estar realizada en el entorno *Scratch*, que es uno de los más conocidos y poderosos en cuanto a programación gráfica se refiere desarrollado en el MIT¹⁷ y escrito en lenguaje *Smalltalk* (Candel Navas 2013).

Esta familia de herramientas tiene una característica muy particular con respecto a las otras y es la posibilidad de escribir un programa para placas Arduino de forma clara sin hacer uso de lenguajes de programación. La mayoría de estas se basan en la programación en bloques o estructuras que se van usando en la medida de las necesidades del usuario. Algunas son soluciones libres y otras no, como Visuino que es un *software* comercial para configurar DAQ. Estas herramientas presentan como inconveniente que no son compatibles con todas las placas que soporta Arduino *IDE* y además algunas de ellas dependen del mismo para la compilación y subida de código a las placas. Tampoco son capaces de configurar los

¹⁴ Empresa de *software* especializado en el desarrollo de *software* de alto rendimiento y soluciones de flujo de datos.

¹⁵ *Scratch* para Arduino.

¹⁶ Es un laboratorio ciudadano digital de Europa, un centro de investigación e innovación sobre la Internet social

¹⁷ Instituto Tecnológico de Massachusetts, en inglés Massachusetts Institute of Technology

Capítulo 1. Fundamentación teórica

parámetros necesarios para la comunicación a través del protocolo Modbus (Helguero 2011) (Roblogs 2018).

1.3.3 Herramientas en línea

- **Fritzing:** fomenta un ecosistema creativo que permite a los usuarios documentar sus prototipos, compartirlos con otros, enseñar productos electrónicos y diseñar placas de circuito impreso profesionales (Knörig, Wettach y Cohen 2009).
- **Tinkercad for Circuits:** es una sencilla herramienta de diseño y modelado 2D y 3D basada en navegador que todos pueden usar. *Tinkercad* permite a los usuarios diseñar en cuestión de minutos todo lo que puedan imaginar. Entre sus aplicaciones se encuentra el diseño y simulación de circuitos para la plataforma Arduino. Permite programar usando lenguaje de programación visual en bloques o un editor de texto (Autodesk 2018).

Estas herramientas brindan la posibilidad de realizar esquemas eléctricos en proyectos con Arduino. Disponen de bibliotecas con la mayoría de componentes, incluido los propios Arduinos, placas de conexiones, *led*¹⁸, motores, *displays*¹⁹, etc. Además permiten diseñar *PCB*²⁰ personalizados y un sinnúmero de opciones demostrando la utilidad de estas herramientas. Ofrecen disponibilidad para cualquier dispositivo o sistema operativo siempre y cuando tengan conectividad a la red. Estas al ser herramientas web no brindan la posibilidad de subir el código creado a las placas Arduino, por lo que dependen de Arduino *IDE* para ello (Zaragoza MakerSpace 2017) (Crespo 2017).

1.3.4 Plugins y Extensiones

- **Data Acquisition Toolbox:** Una variante para obtener un DAQ con arduino completamente funcional y poder usar los datos adquiridos para análisis es usando el *software Matlab*. En este se debe realizar la configuración del puerto por el cual ocurrirá la comunicación serie con el Arduino y también se debe tener en cuenta los valores que serán capturados, los cuales son mostrados en forma de figuras o diagramas dinámicos que varían dependiendo de los valores sensados.
- **LIFA²¹ de LabView:** haciendo uso de este paquete se logra convertir una placa Arduino en un DAQ permitiendo conectar componentes visuales y valores recibidos de la lectura de diferentes variables

18 Diodo emisor de luz, del inglés Light-emitting diode.

19 Visualizador.

20 Placa de Circuito Impreso.

21 Interfaz, del inglés Labview Interface for Arduino.

Capítulo 1. Fundamentación teórica

desde el dispositivo de adquisición. El paquete LIFA incluye un *sketch*²² para placas Arduino sobre el cual se pueden realizar configuraciones de diferentes periféricos conectados. La configuración de la placa se realiza haciendo uso del Arduino *IDE* con el *sketch* proporcionado por el paquete LIFA.

Estas extensiones permiten enviar datos a través de canales de salida analógicos y digitales proporcionados por el *hardware* de adquisición de datos. Puede configurar el *hardware* de adquisición de datos y leer los datos en MATLAB para un análisis inmediato. Pese a las ventajas que presentan, estas tienen como inconvenientes que no son compatibles con todas las placas Arduino que soporta Arduino IDE y al constituir extensiones es necesario el uso de otras herramientas, al mismo tiempo que dependen del IDE de Arduino para enviar el código a la placa.

Luego del estudio realizado sobre las diferentes herramientas que pueden ser usadas para la configuración de DAQ basados en Arduino, agrupadas en familias, se llega a la conclusión de que en sentido general:

- No brindan un mecanismo para configurar este tipo de dispositivos que implementen Modbus como protocolo de comunicación.
- Para el uso de algunas de las herramientas es necesario el pago de licencias.
- La mayoría de las herramientas a excepción de los *IDE* no son capaces de compilar el código que generan y tampoco lo suben a las placas Arduino.

1.4 Metodología de desarrollo

Una metodología describe cómo llevar a cabo el proceso de desarrollo de *software* definiendo “quién” debe hacer, “qué”, “cuándo” y “cómo” debe hacerlo, imponiendo un proceso disciplinado con el fin de aumentar la calidad del *software* que se produce en todas y cada una de sus fases de desarrollo. Una metodología está formada por un conjunto de pasos y procedimientos dividiendo un proyecto en etapas, tareas que se llevan a cabo en cada etapa, las técnicas, herramientas que deben aplicarse, las restricciones que deben tenerse en cuenta, así como llevar a cabo el control y la gestión de un proyecto. Constituye un medio de estandarización que cubre por completo el proceso de desarrollo de *software*. Posibilita verificar trabajos realizados y realizar la corrección de los errores detectados y además provee un lenguaje común entre los analistas, programadores, clientes y usuarios.

²² Es el nombre que Arduino usa para un programa.

Capítulo 1. Fundamentación teórica

Para el desarrollo de la solución planteada se empleará la “Metodología de desarrollo para la Actividad productiva de la UCI”, debido a que está definida por la universidad como el documento rector de la actividad productiva en los proyectos de desarrollo.

La “Metodología de desarrollo para la Actividad productiva de la UCI” describe, de una manera simple y fácil de entender, la forma de desarrollar aplicaciones de *software* usando técnicas ágiles. Entre las técnicas ágiles que utiliza se encuentra el Modelado ágil que permite encapsular los requisitos funcionales en Historias de Usuario (HU) o en Descripción de requisitos por procesos o Casos de Uso. Para la solución propuesta se hará uso de las HU para encapsular los requisitos funcionales debido a que es la técnica usada por el proyecto Sistema Domótico para realizar esta tarea. Esta metodología se apoya en el Modelo CMMI²³ que es un modelo de calidad que guía a una entidad desarrolladora a aplicar mejores prácticas que se centran en el desarrollo de productos de calidad. En la “Metodología de desarrollo para la Actividad productiva de la UCI” se definen tres fases (Inicio, Ejecución y Cierre), las cuales serán aplicadas para guiar el desarrollo del presente trabajo (Sánchez 2015).

1.5 Herramientas, tecnologías y lenguajes de desarrollo

A continuación, se describen las tecnologías y herramientas a utilizar. Las mismas son pilares importantes en el desarrollo de *software* libre, razón por la cual fueron seleccionadas.

1.5.1 Lenguaje de programación

C++ es un lenguaje de programación orientado a objetos, fue creado a mediados de 1980 por Bjarne Stroustrup²⁴. Además, se considera un lenguaje de nivel intermedio porque encapsula características de lenguaje de bajo y alto nivel. C++ es ampliamente utilizado, de hecho, es muy común en aplicaciones de tipo cliente-servidor, firmware incorporado y controladores de *software*. Además, este lenguaje de programación implica el uso de operadores tales como aritmética, comparación, lógica y manipulación de *bits*. Permite la sobrecarga de operadores, convirtiéndolo en un lenguaje altamente atractivo para los programadores (Tale 2016).

Las principales características del Lenguaje C++ son (Ruiz 2013):

- Lenguaje imperativo y estructurado (permite el uso de subrutinas y estructuras de control).

²³ Integración de Modelos de Madurez de Capacidades, del inglés Capability Maturity Model Integration es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software.

²⁴ Científico de la computación y catedrático de Ciencias de la Computación en la Universidad A&M de Texas. Ha destacado por desarrollar el lenguaje de programación C++.

Capítulo 1. Fundamentación teórica

- Amigable, flexible y muy potente para el programador.
- Eficiente.
- Portable.
- Compilado.

1.5.2 Tecnología de desarrollo

Qt es un *framework*²⁵ completo con herramientas diseñadas para simplificar la creación de aplicaciones e interfaces de usuario para plataformas de escritorio, integradas y móviles. Tiene una API²⁶ intuitiva para programación en C++ y similar a *JavaScript*²⁷ con *Qt Quick*²⁸ para una rápida creación de interfaces de usuario (Qt 5.10 2017).

Constituye una plataforma de código abierto muy difundida en estos momentos, Qt incorpora un estilo de programación a C++, que le hace menos vulnerable a los fallos de programación, sobre todo en el manejo de memoria (Colectivo de autores 2010). Para el desarrollo de la solución propuesta se utiliza la versión 5.7.

1.5.3 Entorno Integrado de Desarrollo

Los entornos integrados de desarrollo constituyen programas compuestos por un conjunto de herramientas para los programadores que facilitan la escritura de programas. Para llevar a cabo la implementación de la solución propuesta se decidió usar *QtCreator* como *IDE* de desarrollo en su versión 4.0.2. Éste 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 (Mir y Pérez 2010).

1.5.4 Herramienta de modelado

Hoy en día existen herramientas *CASE*²⁹ que soportan *UML*³⁰ como lenguaje de modelado. Su utilización en el desarrollo de *software* aumenta en gran medida la productividad y reduce costos en cuanto a tiempo y dinero. *Visual Paradigm* es una de estas herramientas usada para el desarrollo de *software* y gestión de empresas, que ofrece todas las características que demanda la arquitectura empresarial, la gestión de

25 Marco de trabajo.

26 Interfaz de Programación de Aplicaciones en inglés Application Programming Interface.

27 Lenguaje de programación orientado a objetos que se usa generalmente del lado del cliente.

28 Framework de desarrollado incorporado dentro del framework Qt.

29 Ingeniería de *Software* Asistida por Computadora, del inglés *Computer Aided Software Engineering*.

30 Lenguaje Unificado de Modelado del inglés *Unified Modeling Language*.

Capítulo 1. Fundamentación teórica

proyectos, el desarrollo de *software* y la colaboración en equipo en una solución única. Es multiplataforma ya que está disponible para tanto para Windows como para *Linux*, cabe mencionar que cuenta con una versión pagada y una *open source*³¹ (Visual Paradigm 2017).

1.5.5 Lenguaje de modelado

El Lenguaje Unificado de Modelado cuenta con una notación estándar y semánticas esenciales para el modelado de un sistema orientado a objetos. Es un lenguaje gráfico, que tiene como fin especificar y documentar un sistema de *software* de un modo estándar, incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema (José Enrique González Cornejo 2017).

1.5.6 Sistema Gestor de Base de Datos

SQLite es un motor de base de datos *SQL* autónomo, de alta confiabilidad, integrado, con todas las funciones y dominio público. Es la base de datos más implementada del mundo con más aplicaciones de las que podemos contar, incluidos varios proyectos de alto perfil (*SQLite* 2018). Se hará uso de la herramienta cliente *DB Browser for SQLite* por ser una herramienta flexible, amigable y de código abierto para crear, diseñar y editar archivos de bases de datos compatibles con *SQLite* (*sqlitebrowser* 2018).

1.5.7 Dependencias

- **Toolchain de Arduino IDE**³²: constituye una serie de herramientas de *software* que trabajando juntas consiguen alcanzar un objetivo. Puede ser visto en tres fases o herramientas (Hernández 2017):
 - 1) **El IDE de Arduino**: su función es la de procesar texto, mostrar información sobre errores y aportar ayudas a la hora de programar. Se utiliza C++ como lenguaje de programación.
 - 2) **El compilador**: se encarga de convertir el código escrito en C++ a lenguaje de máquina para que sea fácilmente entendido por los microcontroladores de las placas Arduino. Reordena el código y busca errores.
 - 3) **AVRdude**: es un programa que con ayuda de un *software* muy básico llamado *Bootloader*, carga dentro de la memoria del microcontrolador el código escrito e interpretado a través del puerto *USB*.
- **Arduino-MK**: es un paquete instalable que brinda la posibilidad de compilar código Arduino desde la interfaz de línea de comandos. Brinda la posibilidad de sustituir el *IDE* de Arduino para la

³¹ Código abierto.

³² Cadena de herramientas de Arduino.

Capítulo 1. Fundamentación teórica

compilación de código. Proporciona un *Makefile* genérico que se ajusta a la sintaxis “*make*” de GNU³³ pero que puede ser perfectamente configurable para propósitos específicos (Mike 2017).

- **Biblioteca Style³⁴**: esta es una biblioteca usada para dar estilo a todos los elementos en una interfaz de usuario. Está compuesta por varios estilos que definen la entidad marcatoria de cada línea de desarrollo de *software* existente en la UCI. En el caso de la propuesta de solución se aplicará la marca de XEDRO por ser la línea de productos que desarrolla el centro CEDIN.

1.6 Conclusiones parciales del capítulo

Una vez realizado el estudio sobre los principales aspectos de este capítulo, conceptos asociados y el estudio del arte sobre las herramientas similares se puede concluir que:

- Luego de agrupar y estudiar las diferentes familias de herramientas se determinó que no es factible la utilización de alguna de las aplicaciones o sistemas ya existentes para dar solución al problema planteado en la presente investigación.
- Como metodología de desarrollo se seleccionó la “Metodología de desarrollo para la Actividad productiva de la UCI” por las grandes ventajas que brinda para los proyectos productivos que se desarrollan en la universidad, ya que con el uso de esta metodología se logró hablar un lenguaje común en todo lo que respecta a roles, disciplinas, fases y productos de trabajo.
- Analizadas las metodologías, herramientas y tecnologías que actualmente favorecen a la evolución del desarrollo de productos informáticos se decidió continuar con la utilización de las herramientas usadas por el centro CEDIN para el desarrollo de sus productos.

³³ Es un sistema operativo de tipo *Unix* desarrollado por y para el Proyecto *GNU*, y auspiciado por la *Free Software Foundation*.

³⁴ Estilo.

CAPÍTULO 2 ANÁLISIS Y DISEÑO

El presente capítulo tiene como objetivo reflejar las actividades que se realizaron en el proceso de análisis y diseño de la propuesta de solución. En el mismo se exponen los artefactos más importantes que describen el funcionamiento del sistema, tales como especificaciones de requisitos que rigen el desarrollo de la solución, detallando la información del análisis y del diseño de la solución en cuestión.

2.1 Modelo de dominio

El modelo de dominio captura los conceptos (tipos de objetos) más importantes del contexto del sistema desde el punto de vista del problema, no de la solución, reflejando “cosas” que existen o eventos que suceden en el entorno en que trabaja el sistema. Se representa a través de un diagrama de clases con asociaciones entre ellas, pero sin cualificar y se pueden incluir atributos en las clases (de forma conceptual) (Sierra y López 2014).

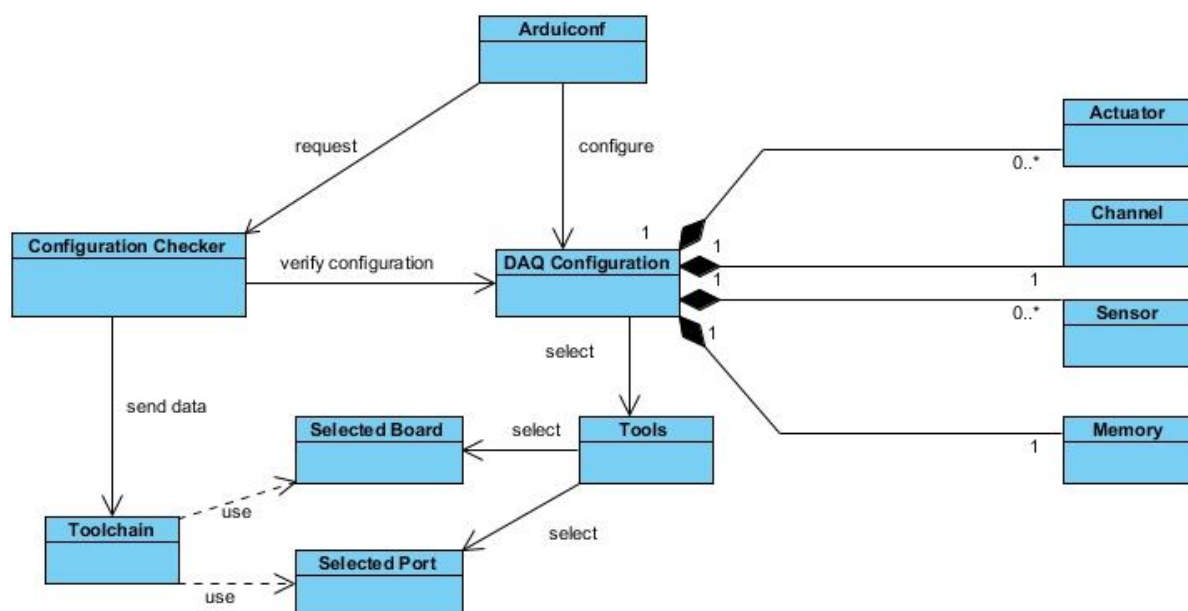


Figura 1. Modelo de Dominio.

Fuente: elaboración propia.

2.2 Descripción de las clases de dominio

Arduiconf: representa la interfaz del *software*, permite realizar las configuraciones de los bloques de memoria Modbus, los sensores, actuadores, el canal de comunicación y muestra un listado de los pines de la placa.

Capítulo 2. Análisis y diseño

DAQ Configuration: es la clase encargada de manejar los actuadores, los sensores y los canales de comunicación para lograr la configuración y que sea subida con éxito a la placa Arduino.

Actuator: se encarga de configurar los actuadores atendiendo al tipo, el registro y la fórmula de conversión usada.

Sensor: se encarga de configurar los diferentes sensores según su tipo, el modelo, el pin al cual están conectados y el registro Modbus.

Channel: es la encargada de configurar los canales de comunicación Serial, TCP o WIFI³⁵, cada uno con sus respectivas características.

Memory: se encarga de configurar la cantidad de bloques de memoria Modbus de cada tipo (*Coils, Input Status, Input Register, Holding Register*).

Configuration Checker: es la encargada de velar porque todas las configuraciones estén correctas y no exista ningún formulario con campos vacíos.

Toolchain: su función es compilar el código recibido y subirlo a la placa Arduino.

Tools: Se encarga de mostrar las placas y puertos disponibles que puedan ser seleccionados.

Selected Board: representa la selección de la placa a la cual será enviada la configuración realizada que debe estar conectada a un puerto *USB*.

Selected Port: representa la selección del puerto en el cual está conectada la placa seleccionada.

2.3 Especificación de requisitos

Durante el proceso de especificación de requisitos es realizada una descripción completa y detallada de las funcionalidades que debe tener el sistema a implementar, definiendo los requisitos funcionales y no funcionales del mismo. Constituye una de las actividades más complejas de la ingeniería de requisitos, por lo complicado que puede resultar el proceso de traspasar las necesidades y expectativas del usuario del futuro sistema al analista y este a su vez al equipo de desarrollo (Martínez y Méndez 2015).

³⁵ Es una tecnología que permite la interconexión inalámbrica de dispositivos electrónicos.

Capítulo 2. Análisis y diseño

2.3.1 Requisitos funcionales

Durante el proceso de obtención de requisitos fueron detectados los siguientes requisitos funcionales los cuales fueron descritos a través de Historias de Usuario.

- RF 1 Crear un nuevo Proyecto de Configuración.
- RF 2 Cargar un Proyecto de configuración desde un archivo *xml*.
- RF 3 Salvar en un archivo *xml* un Proyecto de configuración.
- RF 4 Definir la cantidad de registros de cada tipo de bloque de memoria Modbus.
- RF 5 Adicionar un nuevo sensor.
- RF 6 Listar sensores configurados.
- RF 7 Modificar los datos configurados de un sensor.
- RF 8 Eliminar sensores de la lista.
- RF 9 Adicionar un nuevo actuador.
- RF 10 Listar actuadores configurados.
- RF 11 Modificar los datos configurados de un Actuador.
- RF 12 Eliminar actuadores de la lista.
- RF 13 Verificar configuración.
- RF 14 Listar pines de entrada/salida, con toda la información útil sobre ellos.
- RF 15 Elegir y configurar canal de comunicación (Serie, *TCP* o *WIFI*).
- RF 16 Elegir modelo de placa Arduino y puerto donde está conectada.
- RF 17 Subirlo código Arduino a la placa conectada.

2.3.2 Requisitos no funcionales

Los Requisitos No Funcionales (RNF) se refieren a los requisitos relacionados con la calidad del *software*, especifican propiedades del sistema, como restricciones del entorno o la implementación, rendimiento, dependencia de plataformas, facilidad de mantenimiento, extensibilidad y fiabilidad (Sanchidrian 2014).

- **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 cumpliendo con la estrategia marcaria del centro de desarrollo.
- La interfaz debe tener mensajes sin ambigüedades.

Capítulo 2. Análisis y diseño

- **Usabilidad:**
 - La herramienta debe proporcionar mensajes de error que sean informativos y orientados a usuario final.
 - El sistema debe poseer interfaces gráficas bien formadas.
 - Podrá ser usado con facilidad por parte del usuario.
- **Portabilidad:** la aplicación deberá funcionar en cualquier distribución de *Debian* y *Ubuntu*, pues al ser instalada despliega en el sistema operativo las dependencias necesarias para su funcionamiento.
- **Escalabilidad:** la propuesta de solución del sistema a crear debe estar en capacidad de permitir en el futuro el desarrollo de nuevas funcionalidades, así como modificar o eliminar funcionalidades ya existentes.
- **Adecuación funcional:** el desarrollo de la herramienta de configuración está guiado por las necesidades expresadas por parte de los proveedores de requisitos, dándole cumplimiento a sus especificaciones.

2.4 Propuesta de solución

La solución que se propone pretende llevar por nombre *Arduiconf* y abarca la creación de una herramienta informática capaz de configurar los parámetros de ejecución de un dispositivo de adquisición de datos basado en Arduino que haga uso del protocolo de comunicación Modbus, brindando una interfaz para la configuración de dichos parámetros sencilla y fácil de usar para los usuarios. Estará compuesta por varias interfaces de usuarios, que contienen formularios simples para la configuración de los bloques de memoria Modbus (*Coil*, *Input Status*, *Holding Register* e *Input Register*), la configuración de los sensores y actuadores conectados a la placa Arduino, así como la variante del canal de comunicación Modbus a ejecutar (*TCP*, *Serial*, *WiFi*). Además, será posible visualizar el listado de los pines disponibles en la placa y el uso que se le da a cada uno.

Una vez realizada la configuración será posible guardarla en un archivo con extensión *xml*, el cual puede ser salvado mientras se edita o puede ser cargado uno ya existente. Además, la herramienta posibilitará la subida de dicha configuración a la placa Arduino previamente conectada a un puerto serie, haciendo uso del *toolchain* de Arduino, siendo posible compilar el código generado para cualquiera de las placas soportadas por Arduino *IDE*.

Para la generación de algunos formularios, actualizar el listado de pines de la placa Arduino y la generación código Arduino se hará uso de una base de datos constituida por tres tablas:

Capítulo 2. Análisis y diseño

- Una tabla “*Sensor*” que contiene toda la información necesaria relacionada con diferentes tipos de sensores.
- Una tabla “*Channel*” que especifica los canales de comunicación, sus dependencias para su correcto funcionamiento, entre otros datos.
- Una tabla “*Board*” que, entre otros datos, contiene la información referente a los pines de cada placa, la cual puede ser visualizada en la pestaña “Pines”.

2.5 Historias de usuario

Entre los artefactos que define la metodología seleccionada se encuentran las HU. Se utilizan para especificar los requisitos de las aplicaciones *software* en las metodologías ágiles (*SCRUM, XP, FDD, ASD, AUP, LD*, etc.). Las HU son tarjetas en donde el interesado describe brevemente (con el fin de que sean dinámicas y flexibles) las características que el sistema debe poseer, sean requisitos funcionales o no funcionales (Villamizar Suaza 2014).

Las HU quedan estructuradas de la siguiente manera:

Nombre del requisito: nombre descriptivo de la HU.

Prioridad: grado de prioridad que le asigna el cliente a la HU en dependencia de sus necesidades. Puede tomar los valores de: alta, media o baja.

Estimación: unidades de tiempo estimadas por el equipo de desarrollo para darle cumplimiento a la HU.

Iteración: número de la iteración en la cual será implementada la HU.

Descripción: descripción simple que brinda el cliente sobre lo que debe hacer la funcionalidad en cuestión.

Adicionalmente a cada HU se le asigna un número para facilitar su identificación por parte del equipo de desarrollo y se le puede incluir alguna información adicional que pueda ser útil para la comprensión de la misma. Uno de los beneficios de las HU es que pueden ser escritas en diferentes niveles de detalle. Es posible escribir historias que cubren múltiples funcionalidades.

El autor del trabajo selecciona las HU como técnica de descripción de requisitos por ser una de las técnicas más sencillas y completas, además de seguir la línea de trabajo del proyecto Sistema Domótico para realizar esta tarea.

Capítulo 2. Análisis y diseño

2.6 Descripción de las HU

Durante el diseño de la propuesta de solución se identificaron 17 HU que responden a las diferentes funcionalidades solicitadas por el cliente y presentan una descripción para que el equipo de desarrollo conozca su posterior implementación. A continuación, se describen algunas de las HU identificadas, las restantes pueden ser consultadas en los anexos del 1 al 14.

Tabla 2. HU1 Crear un nuevo proyecto de configuración.

HU1 Crear un nuevo proyecto de configuración	
Número: HU 1	Nombre del requisito: Crear en un nuevo proyecto de configuración.
Programador: Lisardo Garcia Jane	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 3 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 48 horas
Descripción: Al hacer clic en el menú "Archivo" y luego en la opción "Nuevo" o en el icono "Nuevo" de la barra de Herramientas, se debe mostrar un mensaje de confirmación para salvar los últimos cambios realizados en el proyecto sobre el cual se está trabajando. Si escoge "Aceptar" se deben guardar los datos configurados en un archivo <i>xml</i> y mostrar la interfaz con todos los campos reseteados a su valor inicial. Si por el contrario escoge "Cancelar" se pierden las configuraciones realizadas y se reinician todos los campos.	
Observaciones: NA	
Prototipo de Interfaz:	

Capítulo 2. Análisis y diseño

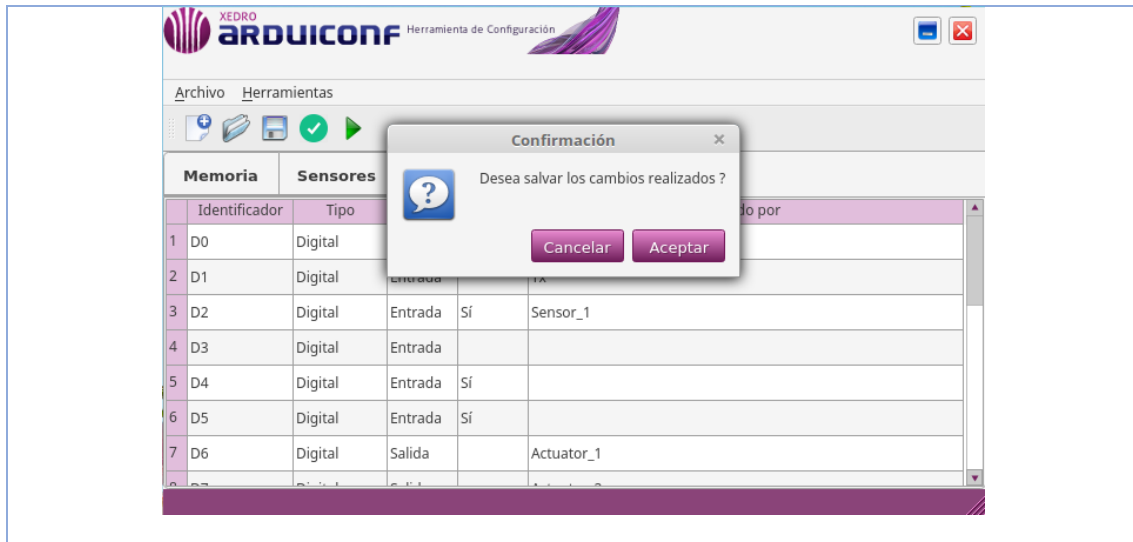


Tabla 3. HU7 Modificar los datos configurados de un sensor.

Número: HU 7	Nombre del requisito: Modificar los datos configurados de un sensor.
Programador: Lisardo Garcia Jane	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 7 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 96 horas
Descripción: Al seleccionar la pestaña "Sensores", se debe mostrar un listado de sensores. Para modificar los datos de alguno de ellos debe estar previamente seleccionado y luego se podrá editar al presionar el botón "Editar" o al dar clic derecho sobre el sensor y escoger la opción "Editar sensor". En ambos casos se mostrará una ventana de edición, la cual permitirá modificar los datos del sensor seleccionado. Una vez modificados se presiona el botón "Aceptar" y se deben mostrar los datos actualizados en el listado.	
Observaciones: Debe existir al menos un sensor listado.	
Prototipo de Interfaz:	

Capítulo 2. Análisis y diseño

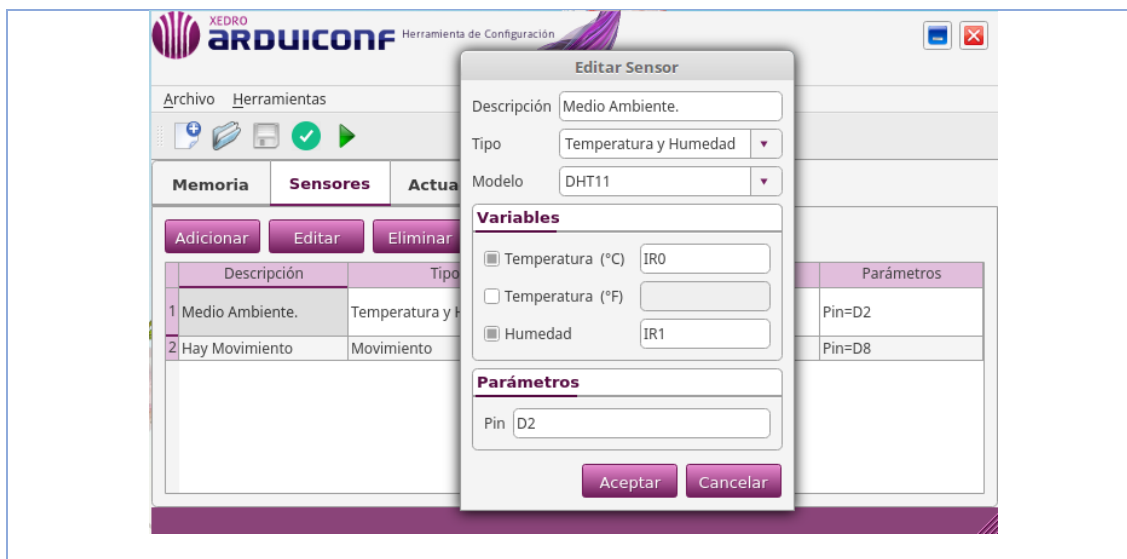


Tabla 4. HU17 Subir código Arduino a la placa conectada.

Número: HU 17	Nombre del requisito: Subir código Arduino a la placa conectada.
Programador: Lisardo Garcia Jane	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 7 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 96 horas
Descripción: Al hacer clic en el botón "Subir Código" se debe verificar si existe algún error en las configuraciones realizadas y si existe alguna placa Arduino conectada. Si no existe errores se debe enviar la configuración realizada a la placa Arduino, la cual debe haber sido seleccionada previamente, en caso contrario se debe mostrar un archivo de texto notificando los errores encontrados en la configuración.	
Observaciones: Debe tener al menos una placa Arduino conectada.	
Prototipo de Interfaz:	

Capítulo 2. Análisis y diseño



2.7 Descripción de la arquitectura

El diseño arquitectónico define la relación entre los elementos estructurales principales del *software*, los patrones de diseño que se pueden utilizar para lograr los requisitos que se han definido para el sistema, y las restricciones que afectan a la manera en que se pueden aplicar los patrones de diseño arquitectónicos, mientras que la arquitectura de *software* describe la estructura del sistema, la cual comprende los componentes del *software*, las propiedades de esos componentes visibles externamente, y las relaciones entre ellos. (Pressman 2005).

2.7.1 Patrón arquitectónico

Atendiendo a las propiedades generales de la aplicación y los requerimientos no funcionales se decidió seleccionar el patrón Modelo-Vista-Controlador³⁶ (MVC) el cual separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distinto.

Modelo: representación específica del dominio de la información sobre la cual funciona la aplicación. La lógica de dominio añade significado a los datos, que además en el caso particular de la aplicación *Arduiconf*, utiliza un mecanismo de almacenamiento (base de datos). En este caso la capa de modelo está compuesta por las clases *Project*, *Sensor*, *Actuator* y *Channel*, las cuales contienen los métodos y funciones necesarias para manejar los datos que son enviados de la interfaz al modelo, así como los que se cargan del modelo

36 Patrón de arquitectura de *software* Model-View-Controller

Capítulo 2. Análisis y diseño

a la interfaz, además, estas clases se encargan del acceso a los datos almacenados en la base de datos a través de consultas que se realizan a la misma. Estas consultas son útiles tanto para la generación de código como para mostrar determinada información en la vista.

Vista: presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario. Esta capa está compuesta por las clases *ActuatorsTableWidget*, *ActuatorWindow*, *FormWindow*, *MainWindow*, *SensorsTableWidget* y *SensorWindow*. Estas clases interactúan tanto con los elementos del modelo como con los elementos controladores, pues estas realizan consultas del estado de elementos del modelo y hacen uso de las diferentes funcionalidades de las clases controladoras necesarias para lograr el fin propuesto.

Controlador: responde a eventos, usualmente acciones del usuario, e invoca cambios en el modelo y probablemente en la vista. En este caso las clases controladoras son *CodeGenerator*, *DataBaseManager*, *XmlReader* y *XmlWriter*. Estas clases responden a eventos que se ejecutan sobre la interfaz, así como también interactúa con el modelo. Fundamentalmente realizan consultas a la base de datos ya sea para generar el código Arduino como para mostrar elementos en los formularios de la interfaz. Además, realizan el manejo del archivo *xml* a la hora de salvarlo o de cargarlo en la vista.



Figura 2. Esquema de funcionamiento del patrón MVC.

Fuente: elaboración propia.

Capítulo 2. Análisis y diseño

2.8 Patrones GRASP³⁷

Los patrones *GRASP* codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Describen los principios fundamentales de la asignación de responsabilidades a objetos expresados en forma de patrones (Grey y Viltres 2015).

Los Patrones *GRASP* para Asignar Responsabilidades constituyen una colección de principios de diseño orientados a objetos que guían la asignación de responsabilidades sobre los objetos, tienen como objetivo esencial ayudar a entender el diseño de éstos, aplicando el razonamiento de diseño de una manera metódica, racional y explicable (Trellini 2015).

Experto: asigna una responsabilidad a la clase que tiene la información necesaria para cumplirla. Este patrón está presente en la clase *FormValidator* que contiene las funciones necesarias para validar cada uno de los campos editables en los formularios.

Controlador: su uso garantiza que los procesos del dominio sean manejados por la capa lógica del negocio y no por la interfaz. Este patrón se manifiesta en la clase *CodeGenerator* la cual sirve de intermediaria entre las clases *MainWindow* (interfaz) y la clase *Project* (modelo).

Alta Cohesión: la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un gran trabajo. Asigna una responsabilidad a la clase de modo que la cohesión siga siendo alta (Marcello y Astudillo 2018). Por ejemplo, la vista *MainWindow* accede a los datos de la clase controladora *CodeGenerator* través de una instancia.

Bajo Acoplamiento: impulsa la asignación de responsabilidades de manera que su localización no incremente el acoplamiento hasta un nivel que nos lleve a los resultados negativos que puede producir un acoplamiento alto. Es el grado de independencia entre los módulos o clases. Un buen diseño se caracteriza por un acoplamiento mínimo, es decir, módulos o clases tan independientes los unos a los otros como sea posible (Gómez, O y N 2011). Este patrón se manifiesta, por ejemplo, en la clase *SensorWindow* que no tiene relación con ninguna clase de la capa modelo y solo se encarga de construir la interfaz para la adición y edición de sensores.

³⁷ General Responsibility Assignment *Software Principles* (or *Patterns*)

Capítulo 2. Análisis y diseño

2.9 Patrones GoF³⁸

- **Estructurales:**
 - **Fachada (*Facade*):** proporciona una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. La “fachada” satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas. Dentro del sistema, este patrón se utiliza para que el usuario acceda a la gran mayoría de las interfaces a través de la interfaz principal de la herramienta. Este patrón especifica que puede darse el caso de vistas a las que se deba acceder a través de otras vistas (Landaburo 2013).
- **Creacionales:**
 - **Singleton:** es quizás el más sencillo de los patrones que se presentan en el catálogo del GoF. Es también uno de los patrones más conocidos y utilizados (Welicki 2018). Dentro de la herramienta su propósito es asegurar que sólo exista una instancia de la clase “*DataBaseManager*” y proporciona un punto de acceso global a esta instancia.
- **De comportamiento** (Tedeschi 2015):
 - **Observer (Observador):** define una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él. Este patrón se manifiesta en la clase “*MainWindow*” específicamente a la hora de actualizar el listado de los pines de la placa seleccionada.
 - **Iterator (Iterador):** proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna. Se evidencia en la clase “*XmlWriter*” la cual se encarga de iterar sobre los sensores y actuadores listados en la interfaz principal.

2.10 Modelo de diseño

La fase de diseño expande y detalla los modelos de análisis tomando en cuenta todas las implicaciones y restricciones técnicas. El propósito del diseño es especificar una solución que trabaje y pueda ser fácilmente convertida en código fuente y construir una arquitectura simple y fácilmente extensible (Navarro 2004). A continuación, se describen los elementos del diseño necesarios para un mejor entendimiento del funcionamiento y estructura de la herramienta de configuración.

38 Pandilla de los Cuatro, del inglés *Gang of Four*.

Capítulo 2. Análisis y diseño

2.10.1 Diagrama de paquetes

Un paquete es un mecanismo utilizado para agrupar elementos de UML, permitiendo organizar los elementos modelados y facilitando de esta forma el manejo de los modelos de un sistema complejo. Los paquetes pueden ser simples estructuras conceptuales o pueden estar reflejados en la implementación. Se pueden utilizar para plantear la arquitectura del sistema a nivel macro (Morales 2013).

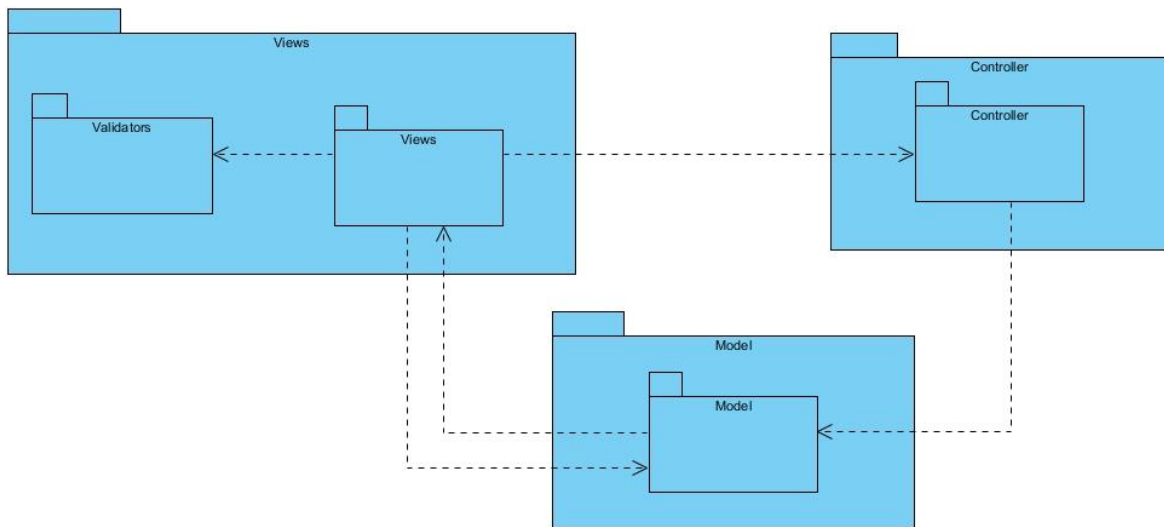


Figura 3. Diagrama de paquetes.

Fuente: elaboración propia

2.10.2 Diagrama de clases

Los diagramas de clases *UML* representan las clases de un programa o sistema computacional, con sus atributos y métodos, junto con las asociaciones entre dichas clases. Además, un diagrama de clases *UML* permite etiquetar clases y las asociaciones entre clases mediante el uso de estereotipos (Vidal-Silva et al. 2018). A continuación, se presenta el diagrama de clases el cual refleja la relación entre las diferentes clases del negocio.

Capítulo 2. Análisis y diseño

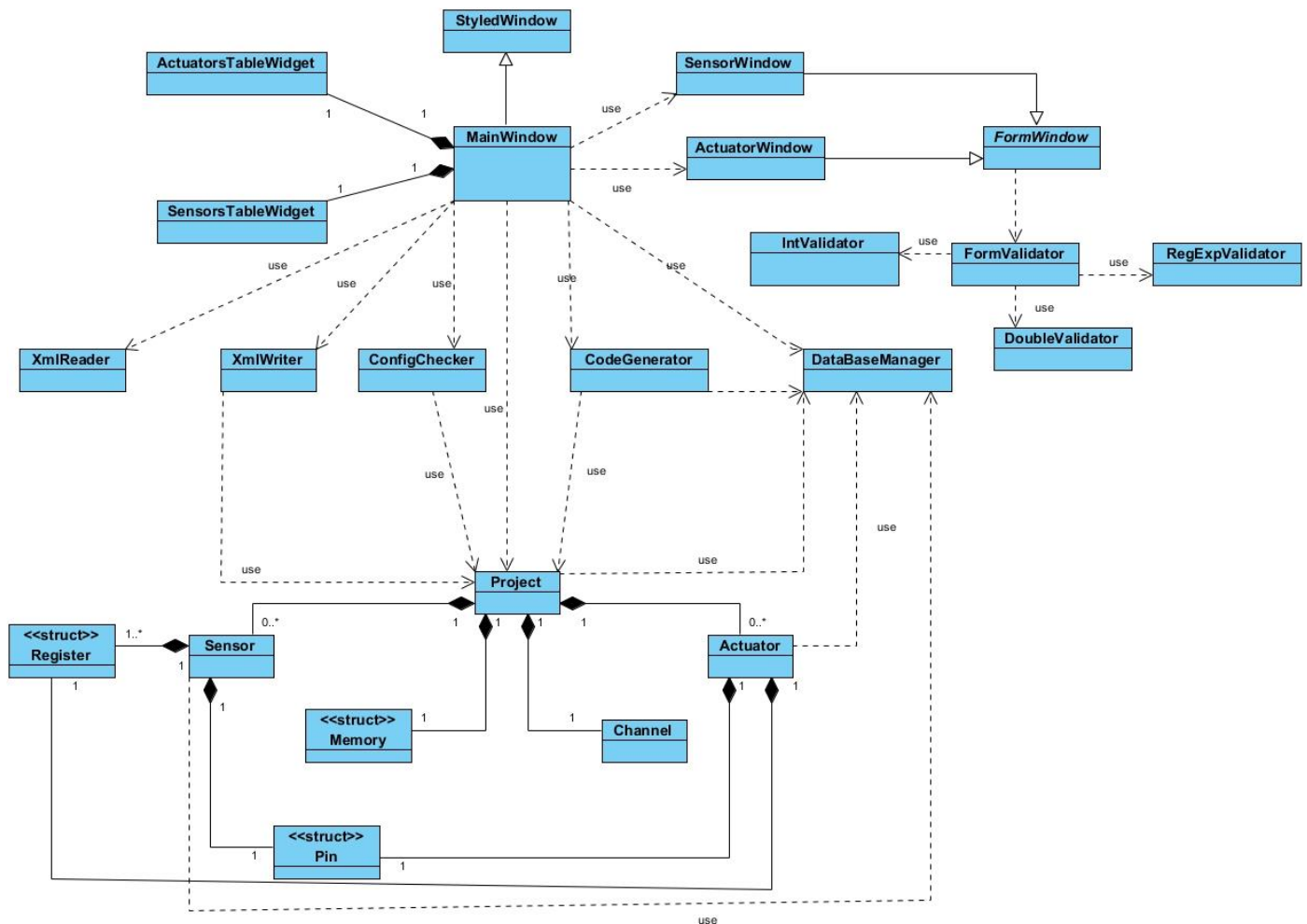


Figura 4. Diagrama de clases.

Fuente: elaboración propia.

2.10.3 Modelo de despliegue

El modelo de despliegue es parte de los diagramas que ayuda a comprender la arquitectura de un sistema. Su propósito es el de distribuir el sistema, asignando componentes ejecutables a nodos en el diagrama de despliegue. Es la forma de mostrar la configuración de nodos de procesamientos en tiempo de ejecución y los componentes que en ellos residen. Estos nodos forman la topología de *hardware* sobre el que se ejecuta el sistema. Este diagrama se preocupa principalmente de la distribución, entrega e instalación de las partes que constituye el sistema físico (Ramirez 2009).

A continuación, se presenta el diagrama de despliegue de la aplicación *Arduiconf*. Está representado por dos componentes principales, una *PC* sobre la cual se ejecuta la aplicación y una base de datos que es

Capítulo 2. Análisis y diseño

usada por la aplicación y se encuentra embebida en el paquete de instalación de la herramienta, siendo colocada en la *PC* en el momento de la instalación de esta en la computadora. Como segundo componente está la placa Arduino, quien se conecta por medio de un cable *USB* a un puerto serie de la computadora, siendo por ese mismo canal que es enviada la configuración realizada en la aplicación.

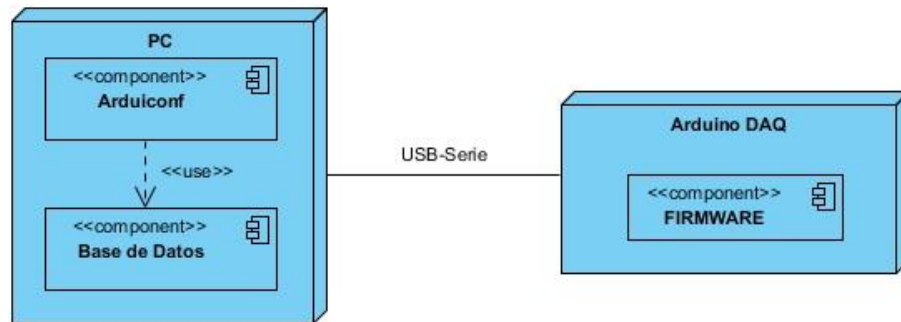


Figura 5. Diagrama de Despliegue.

Fuente: elaboración propia.

2.11 Validación del diseño

Una métrica es un instrumento que cuantifica un criterio y persigue comprender mejor la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo realizado al nivel del proyecto.

Para la evaluación de la calidad del diseño propuesto para la solución se hizo un estudio de las métricas básicas inspiradas en la calidad del diseño orientado a objeto, en el mismo se abarcan atributos de calidad que permiten medir la calidad del diseño propuesto. Dentro de estos se encuentran (Sifontes y Avila 2015):

- **Responsabilidad:** consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- **Complejidad de implementación:** consiste en el grado de dificultad que implica la implementación de un diseño de clases determinado.
- **Reutilización:** consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de *software*.
- **Acoplamiento:** consiste en el grado de dependencia o interconexión de una clase o estructura de clase con otras, está muy ligada a la característica de Reutilización.

Capítulo 2. Análisis y diseño

- **Complejidad del mantenimiento:** consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de *software*. Puede influir indirecta, pero fuertemente en los costos y la planificación del proyecto.
- **Cantidad de pruebas:** consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad del producto diseñado.

Las métricas concebidas como instrumento para evaluar la calidad del diseño y su relación con los atributos de calidad definidos son las siguientes:

TOC (Tamaño Operacional de Clase)

Se refiere al número de métodos pertenecientes a una clase. La siguiente tabla muestra los atributos que forman parte de esta métrica y el modo en que se afectan.

Tabla 5. Tamaño operacional de clase (TOC).

Atributo que afecta	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de Implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución en el grado de reutilización de la clase.

Esta métrica está determinada por los atributos: Responsabilidad, Complejidad de implementación y la Reutilización, existiendo una relación directa con los dos primeros e inversa con el último antes mencionado.

La tabla que se muestra a continuación contiene el rango de valores para la evaluación técnica de los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización). La variable "Prom" indica el promedio de operaciones por cada clase.

Tabla 6. Rango de valores para la evaluación técnica del TOC.

Atributos	Categoría	Criterio
Responsabilidad	Baja	\leq Prom
	Media	Entre Prom y $2 \cdot$ Prom

Capítulo 2. Análisis y diseño

	Alta	$> 2^* \text{ Prom}$
Complejidad de Implementación	Baja	$\leq \text{ Prom}$
	Media	Entre Prom y 2^* Prom
	Alta	$> 2^* \text{ Prom}$
Reutilización	Baja	$\leq \text{ Prom}$
	Media	Entre Prom y 2^* Prom
	Alta	$> 2^* \text{ Prom}$

La siguiente tabla muestra los umbrales de la métrica TOC.

Tamaño operacional de clase	Criterio
Pequeño	$\leq \text{ Prom}$
Medio	Entre Prom y 2^* Prom
Grande	$> 2^* \text{ Prom}$

Resultados del instrumento de evaluación de la métrica TOC.

Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Capítulo 2. Análisis y diseño

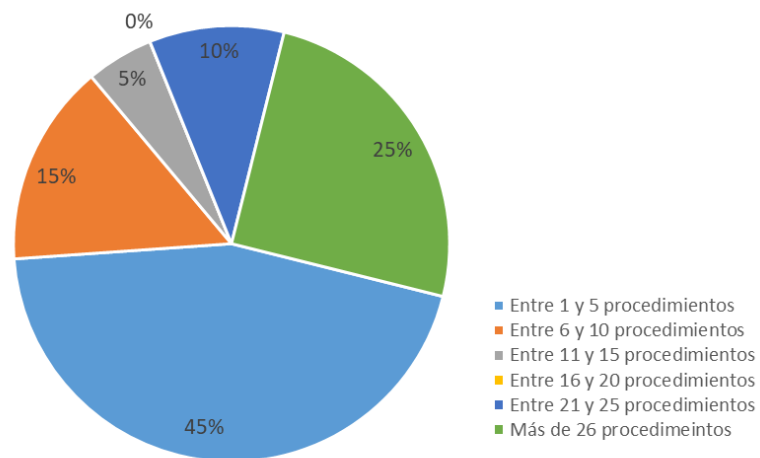


Figura 6. Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

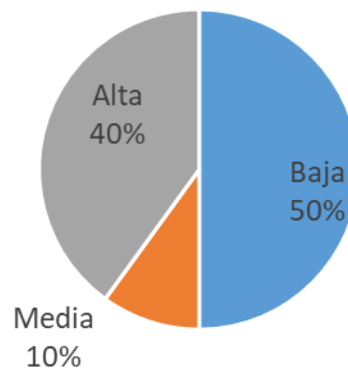


Figura 7. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de implementación.

Capítulo 2. Análisis y diseño

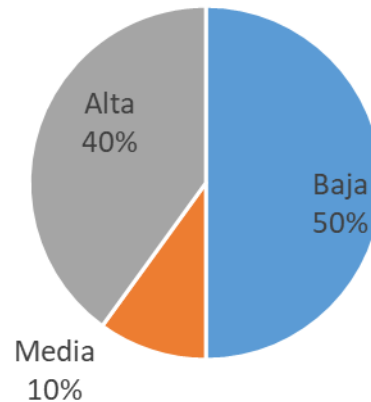


Figura 8. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de implementación.

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización

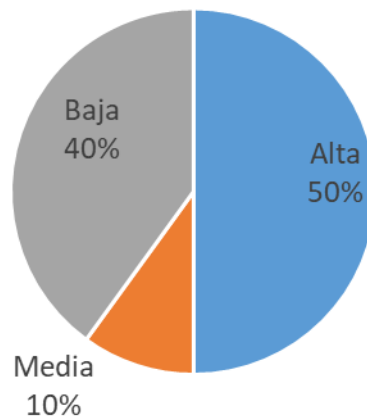


Figura 9. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto para el sistema es simple y tiene una calidad aceptable, teniendo en cuenta que la mayoría de las clases (65%) posee menos cantidad de operaciones que la media registrada en las mediciones. Los atributos de calidad se encuentran en un nivel satisfactorio en el 60% de las clases, de manera que se puede observar cómo se fomenta la Reutilización (elemento clave en el proceso de

Capítulo 2. Análisis y diseño

desarrollo de *software*) y cómo están reducidas en menor grado la Responsabilidad y la Complejidad de implementación.

RC (Relaciones entre Clases)

Esta métrica está dada por el número de relaciones de uso de una clase. La siguiente tabla muestra los atributos pertenecientes a esta métrica y el modo en que se afectan.

Tabla 7. Relación entre clases (RC).

Atributo que afecta	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad del mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.

Esta métrica está determinada por los atributos: Acoplamiento, Complejidad de mantenimiento, Cantidad de pruebas y Reutilización, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado.

La siguiente tabla muestra el Rango de valores para la evaluación técnica de los atributos de calidad (Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de Pruebas) relacionados con la métrica RC. La variable Prom indica el promedio de relaciones entre clases.

Tabla 8. Rango de valores para la evaluación técnica de RC.

Atributos	Categoría	Criterio
Acoplamiento	Baja	\leq Prom
	Media	Entre Prom y $2 * Prom$
	Alta	$> 2 * Prom$
	Baja	\leq Prom

Capítulo 2. Análisis y diseño

Complejidad del mantenimiento	Media	Entre Prom y 2*Prom
	Alta	> 2* Prom
Cantidad de pruebas	Baja	<= Prom
	Media	Entre Prom y 2*Prom
	Alta	> 2* Prom
Reutilización	Baja	<= Prom
	Media	Entre Prom y 2*Prom
	Alta	> 2* Prom

Resultados del instrumento de evaluación de la métrica Relaciones entre Clases

Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

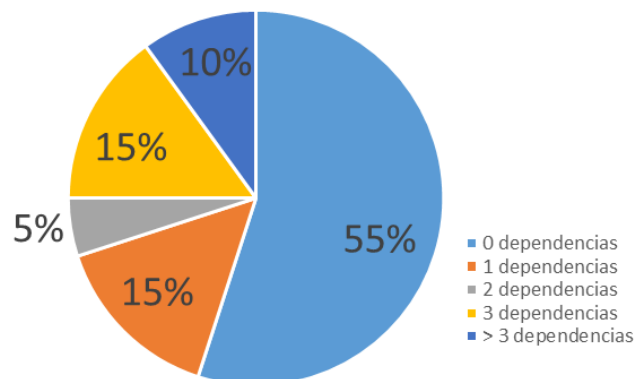


Figura 10. Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

Capítulo 2. Análisis y diseño

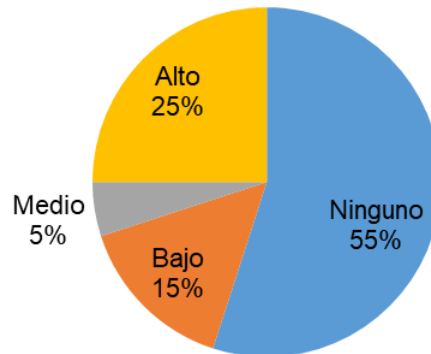


Figura 11. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de mantenimiento.

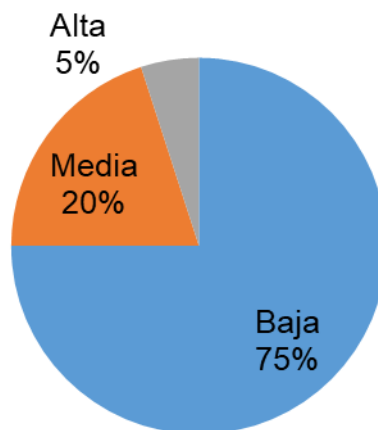


Figura 12. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de mantenimiento.

Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de pruebas.

Capítulo 2. Análisis y diseño

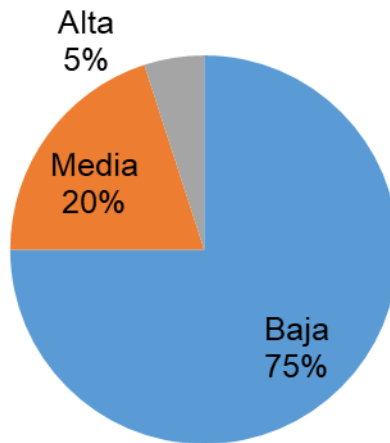


Figura 13. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de pruebas.

Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

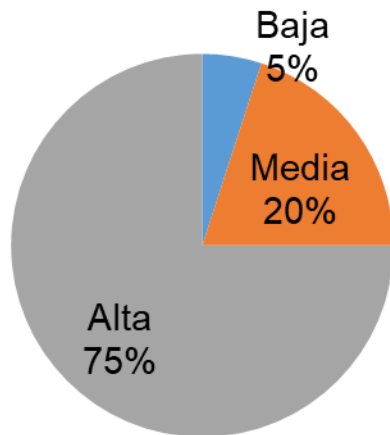


Figura 14. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto para el sistema es simple y tiene una calidad aceptable, teniendo en cuenta que la mayoría de las clases (75%) poseen 2 o menos dependencias respecto a otras. Los atributos de calidad se encuentran en un nivel satisfactorio, en el 75% de las clases el grado de acoplamiento es mínimo, la Complejidad de mantenimiento, la Cantidad de pruebas y la Reutilización se comportan favorablemente para un 95% de las clases.

Capítulo 2. Análisis y diseño

2.12 Conclusiones parciales del capítulo

Luego de realizado el análisis, diseño y durante el desarrollo de la propuesta de solución, se llega a la conclusión de que:

- Se logró modelar la solución de manera general haciendo uso de los artefactos generados por la metodología seleccionada en su escenario 4, llegando a un mejor entendimiento de la herramienta a desarrollar.
- Fueron descritos los requisitos funcionales del sistema haciendo uso de las HU, los cuales serán de gran utilidad para resolver el problema planteado en la investigación y lograr cumplir el objetivo propuesto.
- Se proyectó el diseño de la arquitectura del sistema apoyándose en patrones de diseño *GRASP* y *GoF*, lo que permitió la correcta estructuración del sistema, así como una mejor comprensión de sus componentes y funcionamiento.

Capítulo 3. Implementación y pruebas

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS AL SISTEMA

En el presente capítulo, una vez concluida la fase de análisis y diseño de la propuesta de solución se procede a la implementación de las clases 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* y se determina si las funcionalidades implementadas cumplen con las características establecidas y con las descripciones de las HU anteriormente expuestas, realizando las iteraciones necesarias para cumplir satisfactoriamente los casos de pruebas de aceptación elaborados.

3.1 Estándar de codificación

La forma de escribir código es propia de cada programador. De la forma usada depende la facilidad para entender el código y retomar ciertas partes realizadas por otros integrantes, así como la depuración de las mismas. Se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia pueda ser fácilmente corregido o modificado por otras personas (Marín 2004).

La implementación del sistema fue desarrollada utilizando el idioma inglés para los nombres de constantes, variables, estructuras, funciones, clases y métodos de clases. Como estilo de codificación se utilizó *CamelCase*³⁹. En este caso para los nombres de estructuras, clases y enumeradores se utilizó la variante *UpperCamelCase* con la primera letra de cada palabra en mayúscula, en el caso de los métodos o funciones y variables se utiliza *lowerCamelCase* siendo la primera letra minúscula y la primera letra de las palabras siguientes en mayúsculas.

A continuación, se muestran algunos estilos de codificación que se tuvieron en cuenta para la implementación de la herramienta de configuración, basados en el estándar propuesto por *Google* para las aplicaciones desarrolladas en C++:

Indentación: fueron empleados cuatro espacios como unidades de Indentación de bloques de sentencia.

³⁹ Estilo de escritura que se aplica a frases o palabras compuestas.

Capítulo 3. Implementación y pruebas

```
QStringList boardsNames = (&Singleton<DataBaseManager>::instance())->getBoardsNames();  
for(int i = 0; i < boardsNames.length(); i++)  
{  
    QAction *boardNameAction = new QAction(boardsNames[i], this);  
    boardsMenu->addAction(boardNameAction);  
  
    actionList.append(boardNameAction);  
  
    if (i == 0)  
    {  
        currentBoardAction = boardNameAction;  
        currentBoardAction->setIcon(QIcon(":/cedin-domotic-arduiconf/rightArrow.png"));  
    }  
  
    connect(boardNameAction, SIGNAL(triggered(bool)),  
            this, SLOT(updateSelectedBoard()));  
}
```

Figura 15. Indentación con 4 espacios.

Líneas Largas: las líneas de código largas se fraccionaron atendiendo al principio de fraccionar luego de una coma.

```
connect(addActuatorButton, SIGNAL(clicked(bool)),  
        this, SLOT(showActuatorWindowToCreate()));  
connect(editSensorButton, SIGNAL(clicked(bool)),  
        this, SLOT(showSensorWindowToEdit()));
```

Figura 16. Líneas largas fraccionadas después de la coma.

Declaraciones: los métodos y variables se declararon con nombres asociados a la función por la cual fueron creados.

Capítulo 3. Implementación y pruebas

```
QString Actuator::getName() const
{
    return name;
}

void Actuator::setName(const QString &value)
{
    name = value;
}
```

Figura 17. Declaraciones.

Sentencias: todo el código está compuesto por sentencias simples o complejas indistintamente. En el caso de las sentencias compuestas (*if*, *while*, *do while*, *for*, *switch*) están compuestas por llaves, aunque solo tengan una sentencia evitando la introducción de errores si se añaden otras sentencias posteriormente.

```
void FormWindow::ResetGeometry()
{
    QRect visibleRect = this->visibleRegion().boundingRect();

    if (defaultWindowSize == QSize(-1, -1))
    {
        defaultWindowSize = visibleRect.size();
    }
    else
    {
        this->setGeometry(visibleRect.x(), visibleRect.y(), defaultWindowSize.width(), defaultWindowSize.height());
    }

    switch (type)
    {
        case Type::FixedWidth:
            this->setFixedWidth(defaultWindowSize.width());
            break;
        case Type::FixedHeight:
            this->setFixedHeight(defaultWindowSize.height());
            break;
        case Type::FixedSize:
            this->setFixedSize(defaultWindowSize);
            break;
    }
}
```

Figura 18. Sentencias

3.2 Pruebas al sistema

Los sistemas de *software* hoy en día son parte importante e integral en la gran mayoría de las actividades diarias, es por ello que se debe tener en cuenta que los sistemas o aplicaciones son creadas, desarrolladas e implementadas por seres humanos y, por ende, en cualquiera de sus etapas de creación se puede presentar una equivocación que puede llevar a defectos en las aplicaciones. Las pruebas son necesarias

Capítulo 3. Implementación y pruebas

porque con ellas se puede ayudar a reducir los riesgos en las aplicaciones y lograr de esta manera que se identifiquen los defectos antes de que se ejecuten (Paz 2016).

Una vez concluida la disciplina de implementación y con el objetivo de validar el correcto funcionamiento de los requisitos implementados se realizan pruebas de caja negra y aceptación.

3.2.1 Pruebas de Caja Negra

Las pruebas de caja negra son una forma de derivar y seleccionar condiciones, datos y casos de prueba a partir de los requisitos del sistema. Las pruebas de caja negra no utilizan ninguna información interna de los componentes de *software* o sistemas que se van a probar, sino que consideran el comportamiento del *software* desde el punto de vista de un observador externo (Como los usuarios del sistema). Son utilizadas para realizar pruebas funcionales, basadas en las funciones o características del sistema y su interacción con otros sistemas o componentes. Las funciones del *software* son descritas en los documentos de especificación de requisitos y en las HU (pmoinformatica.com 2016).

3.2.2 Partición de Equivalencia

Esta es una técnica de prueba de Caja Negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de estos casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada la cual regularmente es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica (Pressman 2005). A continuación, se muestran algunos de los casos de prueba empleados para validar el correcto funcionamiento del sistema, el resto puede ser consultado en los anexos del 15 al 25:

Tabla 9. Caso de prueba “Crear en un nuevo proyecto de configuración”.

Escenario 1.1 Crear proyecto de configuración	
Descripción	Se intenta crear un nuevo proyecto de configuración, sin configurar nada aún.
Variables	NA
Respuesta del sistema	Se reinician los campos de todas las pestañas: -La pestaña Memoria muestra los bloques de memoria en cero. -La pestaña Sensores no lista ningún sensor. -La pestaña Actuadores no lista ningún actuador.

Capítulo 3. Implementación y pruebas

	<p>-La pestaña Canal, muestra el canal "TCP" elegido y sin configurar.</p> <p>-La pestaña Pines muestra los pines de la placa Arduino seleccionada, con la columna "Usado por" casi vacía, mostrando solo información de los pines Rx y Tx y los usados por el canal de comunicación.</p>
Flujo central	Desde el ambiente de edición, en la barra de Menú, se presiona el menú "Archivo", se despliegan varias opciones y se selecciona la opción "Nuevo". / Desde el ambiente de edición en la barra de Herramientas se presiona el botón "Nuevo".
Escenario 1.2 Crear proyecto de configuración / Aceptar	
Descripción	Se intenta crear un nuevo proyecto de configuración, guardando los cambios del que se está actualmente configurando.
Variable	NA
Respuesta del sistema	Se guardan los cambios del proyecto que se estaba configurando. Se reinician los campos de todas las pestañas.
Flujo central	Desde el ambiente de edición, en la barra de Menú, se presiona el menú "Archivo", se despliegan varias opciones y se selecciona la opción "Nuevo". Se muestra con un mensaje de confirmación y se presiona el botón "Aceptar". / Desde el ambiente de edición en la barra de Herramientas al presionar el botón "Nuevo". Se muestra un mensaje de confirmación y se da clic en la opción "Aceptar".
Escenario 1.3 Crear proyecto de configuración / Cancelar	
Descripción	Se intenta crear un nuevo proyecto de configuración sin guardar los cambios del que se está actualmente configurando.
Variable	NA
Respuesta del sistema	No se guardan los cambios del proyecto que se estaba configurando. Se reinician los campos de todas las pestañas.
Flujo central	Desde el ambiente de edición, en la barra de Menú, se presiona el menú "Archivo", se despliegan varias opciones y se selecciona la opción "Nuevo". Se muestra con un mensaje de confirmación y se presiona el botón "Cancelar". / Desde el ambiente de edición en la barra de Herramientas al presionar el botón "Nuevo". Se muestra un mensaje de confirmación y se da clic en la opción "Cancelar".

Tabla 10. Caso de prueba "Adicionar sensor".

Escenario 1.1 Adicionar sensor / Opción "Adicionar".

Capítulo 3. Implementación y pruebas

Descripción	Se intenta adicionar un nuevo sensor.		
Descripción		V	Sensor de Temperatura de la Habitación A.
Tipo		V	(Selecciona de la lista desplegable)
Modelo		V	(Selecciona de la lista desplegable)
Variabes	Temperatura (°C)	V	HR1
	Temperatura (°F)	V	CL3
Parámetros	Calibración	V	1.5
Respuesta del sistema	Adiciona el sensor creado a la lista de sensores.		
Flujo central	Desde el ambiente de edición se selecciona la pestaña "Sensores", se presiona el botón "Adicionar". Se muestra en una ventana los campos: Descripción, Tipo, Modelo, Variabes y Parámetros, se llenan dichos campos y se presiona el botón "Adicionar". / Desde el ambiente de edición se selecciona la pestaña "Sensores", se da clic derecho sobre la lista de sensores y se selecciona la opción "Adicionar un sensor". Se muestra en una ventana los campos: Descripción, Tipo, Modelo, Variabes y Parámetros, se llenan dichos campos y se presiona el botón "Adicionar".		
Escenario 1.2 Adicionar sensor / Opción "Cancelar".			
Descripción	Se intenta adicionar un nuevo sensor.		
Descripción		V	Sensor de Temperatura de la Habitación A.
Tipo		V	(Selecciona de la lista desplegable)
Modelo		V	(Selecciona de la lista desplegable)
Variabes	Temperatura (°C)	V	IR5
	Temperatura (°F)	V	IS3
Parámetros	Calibración	V	1.5
Respuesta del sistema	Cierra la ventana y vuelve al ambiente de edición.		
Flujo central	Desde el ambiente de edición se selecciona la pestaña "Sensores", se presiona el botón "Adicionar". Se muestra en una ventana los campos: Descripción, Tipo, Modelo, Variabes y Parámetros, se llenan o no dichos campos y se presiona el botón "Cancelar". / Desde el ambiente de edición se selecciona la pestaña "Sensores", se da clic derecho sobre la lista de sensores y se selecciona la opción "Adicionar un sensor". Se muestra en una ventana los campos: Descripción, Tipo, Modelo,		

Capítulo 3. Implementación y pruebas

	Variables y Parámetros, se llenan o no dichos campos y se presiona el botón "Cancelar".		
Escenario 1.3 Adicionar sensor / Campos vacíos.			
Descripción	Se intenta adicionar un nuevo sensor habiendo seleccionado una o más variables.		
Descripción		V	Sensor de Temperatura de la Habitación A.
Tipo		V	(Selecciona de la lista desplegable)
Modelo		V	(Selecciona de la lista desplegable)
Variables	Temperatura (°C)	V	(Vacío)
	Temperatura (°F)	V	(Vacío)
Parámetros	Calibración	V	1.5
Respuesta del sistema	Muestra los campos vacíos resaltados en rojo y con un <i>ToolTip</i> que explica el error en cada uno. No se activa el botón "Adicionar".		
Flujo central	Desde el ambiente de edición se selecciona la pestaña "Sensores", se presiona el botón "Adicionar". Se muestra en una ventana los campos: Descripción, Tipo, Modelo, Variables y Parámetros; se escoge una o más variables a configurar, y no se les especifica el registro donde se guardará su valor. / Desde el ambiente de edición se selecciona la pestaña "Sensores", se da clic derecho sobre la lista de sensores y se selecciona la opción "Adicionar un sensor". Se muestra en una ventana los campos: Descripción, Tipo, Modelo, Variables y Parámetros; se escoge una o más variables a configurar, y no se les especifica el registro donde se guardará su valor.		

Tabla 11. Caso de prueba "Modificar los datos configurados de un sensor".

Escenario 1.1 Modificar sensor / Opción "Aceptar".			
Descripción	Se intenta modificar un sensor creado.		
Descripción		V	Sensor de Temperatura de la Habitación A.
Tipo		V	(Selecciona de la lista desplegable)
Modelo		V	(Selecciona de la lista desplegable)
Variables	Temperatura (°C)	V	HR5

Capítulo 3. Implementación y pruebas

	Temperatura (°F)	V	CL6
Parámetros	Calibración	V	1.5
Respuesta del sistema	Modifica los datos del sensor. Muestra los cambios realizados en la lista de sensores.		
Flujo central	Desde el ambiente de edición se selecciona la pestaña "Sensores", se selecciona un sensor ya creado y se presiona el botón "Editar". Se muestra en una ventana los campos: Descripción, Tipo, Modelo, Variables y Parámetros, se modifican los campos y se presiona el botón "Aceptar". / Desde el ambiente de edición se selecciona la pestaña "Sensores", se selecciona el sensor, se da clic derecho sobre el mismo y se selecciona la opción "Editar el sensor seleccionado". Se muestra en una ventana los campos: Descripción, Tipo, Modelo, Variables y Parámetros; se modifican dichos campos y se presiona el botón "Aceptar".		
Escenario 1.2 Modificar sensor / Opción "Cancelar".			
Descripción	Se intenta modificar un sensor creado.		
Descripción		V	Sensor de Temperatura de la Habitación A.
Tipo		V	(Selecciona de la lista desplegable)
Modelo		V	(Selecciona de la lista desplegable)
Variables	Temperatura (°C)	V	IR5
	Temperatura (°F)	V	IS3
Parámetros	Calibración	V	1.5
Respuesta del sistema	No modifica los valores, cierra la ventana y vuelve al ambiente de edición.		
Flujo central	Desde el ambiente de edición se selecciona la pestaña "Sensores", se selecciona un sensor y se presiona el botón "Editar". Se muestra en una ventana los campos: Descripción, Tipo, Modelo, Variables y Parámetros, se modifican o no dichos campos y se presiona el botón "Cancelar" / Desde el ambiente de edición se selecciona la pestaña "Sensores", se selecciona un sensor, se da clic derecho sobre el mismo y se selecciona la opción "Editar el sensor seleccionado". Se muestra en una ventana los campos: Descripción, Tipo, Modelo, Variables y Parámetros; se modifican o no dichos campos y se presiona el botón "Cancelar".		
Escenario 1.3 Modificar sensor / Campos vacíos.			
Descripción	Se intenta modificar un sensor dejando campos vacíos.		

Capítulo 3. Implementación y pruebas

Descripción		V	Sensor de Temperatura de la Habitación A.
Tipo		V	(Selecciona de la lista desplegable)
Modelo		V	(Selecciona de la lista desplegable)
Variables	Temperatura (°C)	V	(Vacío)
	Temperatura (°F)	V	(Vacío)
Parámetros	Calibración	V	1.5
Respuesta del sistema	Desactiva el botón "Aceptar".		
Flujo central	Desde el ambiente de edición se selecciona la pestaña "Sensores", se presiona el botón "Editar". Se muestra en una ventana los campos: Descripción, Tipo, Modelo, Variables y Parámetros; se escoge una o más variables a configurar, y no se les especifica el registro donde se guardará su valor. / Desde el ambiente de edición se selecciona la pestaña "Sensores", se selecciona un sensor, se da clic derecho sobre el mismo y se selecciona la opción "Editar el sensor seleccionado". Se muestra en una ventana los campos: Descripción, Tipo, Modelo, Variables y Parámetros; se escoge una o más variables a configurar, y no se les especifica el registro donde se guardará su valor.		

Tabla 12. Caso de prueba "Subir código Arduino a la placa conectada".

Escenario 1.1 Subir código Arduino / No hay errores en la configuración.	
Descripción	Se intenta generar, compilar y subir el código Arduino a la placa conectada, no existiendo errores en la configuración.
Variables	NA
Respuesta del sistema	Genera código Arduino en el archivo "Code.ino". Sube el código compilado a la placa Arduino conectada. El Sistema muestra en la barra de estado el progreso de la operación.
Flujo central	Desde el ambiente de edición, en la barra de Menú se da clic en "Herramientas" y se selecciona la opción "Subir Código". El Sistema verifica que haya una placa Arduino conectada y si la hay. El Sistema verifica errores en la configuración y no detecta ninguno. / Desde el ambiente de edición, en la barra de Herramientas se selecciona la opción "Subir Código". El Sistema verifica que haya una placa Arduino conectada y si la hay. El Sistema verifica errores en la configuración y no detecta ninguno.

Capítulo 3. Implementación y pruebas

Escenario 1.2 Subir código Arduino / Hay errores en la configuración.	
Descripción	Se intenta generar, compilar y subir el código Arduino a la placa conectada, existiendo errores en la configuración.
Variable	NA
Respuesta del sistema	No se genera código Arduino. No se sube el código compilado a la placa Arduino conectada. Se muestra la lista de errores en la configuración que el usuario debe corregir.
Flujo central	Desde el ambiente de edición, en la barra de Menú se presiona el menú "Herramientas" y se selecciona la opción "Subir Código". El Sistema verifica que haya una placa Arduino conectada y si la hay. El Sistema verifica errores en la configuración, detecta al menos uno y muestra un editor de texto con los errores detectados. / Desde el ambiente de edición, en la barra de Herramientas se presiona el botón "Subir Código". El Sistema verifica que haya una placa Arduino conectada y si la hay. El Sistema verifica errores en la configuración, detecta al menos uno y muestra un editor de texto con los errores detectados.
Escenario 1.3 Subir código Arduino / No hay placa Arduino conectada.	
Descripción	Se intenta generar, compilar y subir el código Arduino sin una placa conectada.
Variable	NA
Respuesta del sistema	Se notifica al usuario que conecte primero una placa Arduino.
Flujo central	Desde el ambiente de edición, en la barra de Menú se presiona el menú "Herramientas" y se selecciona la opción "Subir Código". El Sistema verifica que haya una placa Arduino conectada, pero detecta que no la hay y muestra un mensaje de notificación. En la ventana del mensaje de notificación se presiona el botón "Aceptar". / Desde el ambiente de edición, en la barra de Herramientas se presiona el botón "Subir Código". El Sistema verifica que haya una placa Arduino conectada, pero detecta que no la hay y muestra un mensaje de notificación. En la ventana del mensaje de notificación se presiona el botón "Aceptar".

En el proceso de pruebas de caja negra a la aplicación "Arduicon" se aplicaron tres iteraciones de prueba con el objetivo de verificar su correcto funcionamiento y cumplimiento con las funcionalidades. Se obtuvo un total de ocho no conformidades las cuales se explican a continuación:

Capítulo 3. Implementación y pruebas

Iteración 1:

Durante la primera iteración de pruebas a la aplicación “*Arduicon*”, se detectaron cinco no conformidades, las cuales se describen a continuación:

Tabla 13. Resultados de la primera iteración de pruebas.

Funcionalidad	Resultado	Satisfactorio/No satisfactorio
Crear un nuevo Proyecto de Configuración.	Se creó un nuevo proyecto de configuración.	Satisfactorio
Cargar un Proyecto de configuración desde un archivo <i>xml</i> .	Se cargó un proyecto de configuración existente.	Satisfactorio
Salvar en un archivo <i>xml</i> / un Proyecto de configuración.	Se salvó en un archivo con extensión <i>xml</i> un proyecto de configuración.	Satisfactorio
Definir la cantidad de registros de cada tipo de bloque de memoria Modbus.	Se definieron los registros de cada tipo de bloque de memoria Modbus.	Satisfactorio
Adicionar un nuevo sensor.	Se agregó un sensor con el pin definido pero sin registro para almacenar los valores censados.	No satisfactorio
Listar Sensores configurados.	Se mostró el listado de los sensores configurados en la pestaña “Sensores”.	Satisfactorio
Modificar los datos configurados de un sensor.	Al seleccionar la opción editar de la pestaña “Sensores” no se cargan los datos configurados para el sensor seleccionado.	No satisfactorio
Eliminar sensores de la lista.	Se eliminaron todos los sensores seleccionados.	Satisfactorio
Adicionar un nuevo actuador.	Se adicionó un actuador a la lista de actuadores con el pin, registro de memoria y conversión configurados.	Satisfactorio
Listar Actuadores configurados.	Se mostró el listado de actuadores en la pestaña “Actuadores”.	Satisfactorio
Modificar los datos configurados de un Actuador.	Al seleccionar la opción editar de la pestaña “Actuadores” no se cargan los datos	No Satisfactorio

Capítulo 3. Implementación y pruebas

	configurados para el actuador seleccionado.	
Eliminar un Actuador de la lista.	No se eliminó ninguno de los actuadores seleccionados.	No Satisfactorio
Verificar configuración.	Se verificaron las configuraciones realizadas.	Satisfactorio
Listar pines de entrada/salida, con toda la información útil sobre ellos.	Se mostró en la pestaña “Pines” todos los pines según la placa seleccionada con toda la información útil sobre ellos.	Satisfactorio
Elegir y configurar canal de comunicación (Serie o TCP).	Al seleccionar el canal de comunicación “Serie” no se mostraron los campos configurables para este canal.	No Satisfactorio
Elegir modelo de placa Arduino y puerto donde está conectada.	Se eligió el modelo de la placa Arduino y el puerto donde está conectada.	Satisfactorio
Subir código Arduino a la placa conectada.	Se compiló y subió el código Arduino generado a la placa Arduino conectada.	Satisfactorio

Iteración 2

Durante la segunda iteración de pruebas fueron corregidas las no conformidades de la iteración anterior, sin embargo, en esta iteración se detectaron tres nuevas no conformidades, las cuales se describen a continuación:

Tabla 14. Resultados de la segunda iteración de pruebas.

Funcionalidad	Resultado	Satisfactorio/No satisfactorio
Crear un nuevo Proyecto de Configuración.	Se creó un nuevo proyecto de configuración.	Satisfactorio
Cargar un Proyecto de configuración desde un archivo <i>xml</i> .	Se cargó un proyecto de configuración existente.	Satisfactorio
Salvar en un archivo <i>xml</i> un Proyecto de configuración.	Se salvó en un archivo con extensión <i>xml</i> un proyecto de configuración.	Satisfactorio
Definir la cantidad de registros de cada tipo de bloque de memoria Modbus.	Se definieron los registros de cada tipo de bloque de memoria Modbus.	Satisfactorio

Capítulo 3. Implementación y pruebas

Adicionar un nuevo sensor.	Se agregó un sensor con el pin definido y registro para almacenar los valores censados.	Satisfactorio
Listar Sensores configurados.	Se mostró el listado de los sensores configurados en la pestaña "Sensores".	Satisfactorio
Modificar los datos configurados de un sensor.	Al realizar cambios en un sensor configurado no se modificaron los cambios en el listado de sensores.	No satisfactorio
Eliminar sensores de la lista.	Se eliminaron todos los sensores seleccionados.	Satisfactorio
Adicionar un nuevo actuador.	Se adicionó un actuador a la lista de actuadores con el pin, registro de memoria y conversión configurados.	Satisfactorio
Listar Actuadores configurados.	Se mostró el listado de actuadores en la pestaña "Actuadores".	Satisfactorio
Modificar los datos configurados de un Actuador.	No se modificaron todos los campos configurables de un actuador.	No satisfactorio
Eliminar un Actuador de la lista.	Se eliminaron los actuadores seleccionados.	No Satisfactorio
Verificar configuración.	Se verificaron las configuraciones realizadas .	Satisfactorio
Listar pines de entrada/salida, con toda la información útil sobre ellos.	Se mostró en la pestaña "Pines" todos los pines según la placa seleccionada con toda la información útil sobre ellos.	Satisfactorio
Elegir y configurar canal de comunicación (Serie o TCP).	Al seleccionar el canal de comunicación "Serie" se mostraron los campos configurables para este canal, pero no se oculta el combobox para seleccionar el tipo de <i>hardware</i> de <i>Ethernet</i> .	No satisfactorio
Elegir modelo de placa Arduino y puerto donde está conectada.	Se eligió el modelo de la placa Arduino y el puerto donde está conectada.	Satisfactorio
Subir código Arduino a la placa conectada.	Se compiló y subió el código Arduino generado a la placa Arduino conectada.	Satisfactorio

Capítulo 3. Implementación y pruebas

Iteración 3

Durante la tercera iteración de pruebas fueron corregidas las no conformidades detectadas en la iteración anterior, cumpliendo correctamente con todas las funcionalidades especificadas para la aplicación, las cuales se describen a continuación:

Tabla 15. Resultados de la tercera iteración de pruebas.

Funcionalidad	Resultado	Satisfactorio/No satisfactorio
Crear un nuevo Proyecto de Configuración.	Se creó un nuevo proyecto de configuración.	Satisfactorio
Cargar un Proyecto de configuración desde un archivo <i>xml</i> .	Se cargó un proyecto de configuración existente.	Satisfactorio
Salvar en un archivo <i>xml</i> un Proyecto de configuración.	Se salvó en un archivo con extensión <i>xml</i> un proyecto de configuración.	Satisfactorio
Definir la cantidad de registros de cada tipo de bloque de memoria Modbus.	Se definieron los registros de cada tipo de bloque de memoria Modbus.	Satisfactorio
Adicionar un nuevo sensor.	Se agregó un sensor con el pin definido y registro para almacenar los valores censados.	Satisfactorio
Listar Sensores configurados.	Se mostró el listado de los sensores configurados en la pestaña "Sensores".	Satisfactorio
Modificar los datos configurados de un sensor.	Al realizar cambios en un sensor configurado se modificaron los cambios en el listado de sensores.	Satisfactorio
Eliminar sensores de la lista.	Se eliminaron todos los sensores seleccionados.	Satisfactorio
Adicionar un nuevo actuador.	Se adicionó un actuador a la lista de actuadores con el pin, registro de memoria y conversión configurados.	Satisfactorio
Listar Actuadores configurados.	Se mostró el listado de actuadores en la pestaña "Actuadores".	Satisfactorio
Modificar los datos configurados de un Actuador.	Se modificaron todos los campos configurables de un actuador.	Satisfactorio

Capítulo 3. Implementación y pruebas

Eliminar un Actuador de la lista.	Se eliminaron los actuadores seleccionados.	No Satisfactorio
Verificar configuración.	Se verificaron las configuraciones realizadas.	Satisfactorio
Listar pines de entrada/salida, con toda la información útil sobre ellos.	Se mostró en la pestaña "Pines" todos los pines según la placa seleccionada con toda la información útil sobre ellos.	Satisfactorio
Elegir y configurar canal de comunicación (Serie o TCP).	Al seleccionar el canal de comunicación "Serie" se mostraron los campos configurables para este canal quedando oculto el combobox para seleccionar el tipo de <i>hardware</i> de <i>Ethernet</i> .	Satisfactorio
Elegir modelo de placa Arduino y puerto donde está conectada.	Se eligió el modelo de la placa Arduino y el puerto donde está conectada.	Satisfactorio
Subir código Arduino a la placa conectada.	Se compiló y subió el código Arduino generado a la placa Arduino conectada.	Satisfactorio

Como resultado de las pruebas de caja negra se realizó un total de tres iteraciones, donde se obtuvieron ocho no conformidades, las cuales fueron solucionadas en su totalidad, demostrando que la aplicación cumple con los requerimientos especificados.

Otro elemento que se hace necesario mencionar durante el proceso de validación del módulo desarrollado fue la realización de pruebas de aceptación, concluidas con la entrega por parte del cliente de un Acta de aceptación, en la que deja constancia de que el sistema que fue desarrollado cumple con las especificaciones técnicas y funcionales que fueron solicitadas al momento de realizarse la solicitud de servicio.

3.3 Conclusiones parciales del capítulo

Una vez culminada la etapa de implementación y haberle realizado las pruebas de caja negra a la herramienta de configuración, se puede concluir que:

- Se especificaron distintos estándares de código, con el objetivo de facilitar la legibilidad del código, en el desarrollo de la propuesta de solución.

Capítulo 3. Implementación y pruebas

- Se desarrollaron diferentes casos de prueba que permitieron darle cumplimiento a las HU especificadas (en el Capítulo 2), cumpliendo de manera satisfactoria las pruebas de caja negra realizadas.

Conclusiones Generales

CONCLUSIONES GENERALES

Al finalizar esta investigación se llega a la conclusión de que se le dio solución al problema de investigación cumpliendo con el objetivo propuesto al principio de la misma pues:

- Se obtuvo una herramienta que permite configurar dispositivos de adquisición de datos basados en Arduino que implementan el protocolo Modbus con una interfaz sencilla y de fácil utilización.
- Se obtuvo una aplicación desarrollada a partir del uso de tecnologías y herramientas de *software* libre, contribuyendo así a la independencia tecnológica.
- La solución fue sometida a un proceso de pruebas de funcionamiento guiado por casos de pruebas, donde la realización de las mismas corroboró la solidez del sistema, cumpliendo con las características pedidas por el cliente.

RECOMENDACIONES

Se recomienda:

- Poblar la base de datos sobre la que trabaja la herramienta para que soporte una mayor cantidad de sensores, placas Arduino y hardware para la comunicación.
- Incorporar en la base de datos información visual y textual de los pines de cada sensor, y una herramienta de búsqueda de la misma como parte de la ayuda brindada por la aplicación al usuario.
- Incluir en la solución desarrollada la utilización del protocolo de comunicación *bluetooth*.

Referencias Bibliográficas

REFERENCIAS BIBLIOGRÁFICAS

- ALMAGUER, D.I.I., 2012. *Aplicación de configuración para la versión 2 de TETSCADA*. S.l.: s.n.
- ARDUINO, 2017. Arduino. [en línea]. Disponible en: <https://www.arduino.cc/en/Guide/Introduction>.
- ARDUINO, S.A., 2015. Arduino LLC. [en línea]. [Consulta: 8 junio 2018]. Disponible en: http://scholar.googleusercontent.com/scholar?q=cache:Xb6-sCWKOc8J:scholar.google.com/+Arduino+IDE&hl=es&as_sdt=0,5.
- AUTODESK, I., 2018. Circuits on Tinkercad. *Tinkercad* [en línea]. [Consulta: 9 junio 2018]. Disponible en: <https://www.tinkercad.com/circuits>.
- CANDEL NAVAS, R., 2013. Plataforma docente para la enseñanza de las TIC, basada en la maqueta de un puente colgante. [en línea], [Consulta: 8 junio 2018]. Disponible en: <http://repositorio.upct.es/handle/10317/3212>.
- CÁZAREZ-AYALA, G., CASTILLO-MEZA, H. y FONSECA-BELTRÁN, J., 2012. UNIDAD DE ADQUISICIÓN DE DATOS Y MEDICIÓN BASADA EN PROTOCOLO DE COMUNICACIÓN WiFi. . S.l.:
- CEDOM, 2017. CEDOM Asociación Española de Domótica e Inmótica. [en línea]. S.l.: Disponible en: <http://www.cedom.es/sobre-domotica/que-es-domotica>.
- CO2 MEASUREMENT SPECIALISTS, 2015. DATA ACQUISITION SYSTEM (DAS) Software User Manual. [en línea]. [Consulta: 20 noviembre 2017]. Disponible en: <http://co2meters.com/Documentation/Manuals/Manual-DAS.pdf>.
- COLECTIVO DE AUTORES, 2010. *APRENDA Qt4 DESDE HOY MISMO*. S.l.: s.n.
- CRESPO, E., 2017. Simuladores Arduino. *Aprendiendo Arduino* [en línea]. [Consulta: 21 mayo 2018]. Disponible en: <https://aprendiendoarduino.wordpress.com/2017/06/18/simuladores-arduino-2/>.
- ESTRADA CORONA, L.A.E., 2004. PROTOCOLOS TCP/IP DE INTERNET. ,
- EVANS, D., 2011. Internet de las cosas Cómo la próxima evolución de Internet lo cambia todo. [en línea]. S.l.: [Consulta: 19 enero 2018]. Disponible en: https://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&ved=0ahUKEwiGnJKAt eTYAhUITd8KHTnJDkEQFgg7MAM&url=http%3A%2F%2Frcci.uci.cu%2F%3Fjournal%3Drcci%26page%3Darticle%26op%3Ddownload%26path%255B%255D%3D88%26path%255B%255D%3D76&usq=AOvVaw1S_hTpTxXWAX3ZaPeutM_L.
- GÓMEZ, G.L.G., O, J.F.A. y N, D.A.M., 2011. Una ontología para la representación de conceptos de diseño de software. *Avances en Sistemas e Informática*, vol. 8, no. 3, pp. 103-110. ISSN 1909-0056.
- GONZÁLES, S.M., 2015. Entendiendo el internet de las cosas. [en línea], [Consulta: 29 marzo 2018]. Disponible en: http://revistas.tec.ac.cr/index.php/investiga_tec/article/viewFile/2381/2169.

Referencias Bibliográficas

- GREY, L.W.G. y VILTRES, Y.M., 2015. Proceso de mejora del Sistema de Gestión de Proyectos para Cuba y Venezuela. *Campus Virtuales*, vol. 3, no. 2, pp. 16-22. ISSN 2255-1514.
- HELGUERO, A., 2011. Serie: Herramientas Gráficas para la programación de Arduino. [en línea], [Consulta: 21 mayo 2018]. Disponible en: http://www.academia.edu/6287537/Serie_Herramientas_Gr%C3%A1ficas_para_la_programaci%C3%B3n_de_Arduino.
- HERNÁNDEZ, D.V.L., 2017. Curso de Arduino aprende a programar desde cero. *Programar fácil con Arduino* [en línea]. [Consulta: 30 mayo 2018]. Disponible en: <https://programarfacil.com/blog/arduino-blog/curso-de-arduino/>.
- JOSÉ ENRIQUE GONZÁLEZ CORNEJO, D., 2017. El Lenguaje de Modelado Unificado (UML). [en línea]. [Consulta: 5 diciembre 2017]. Disponible en: <http://www.docirs.com/uml.htm>.
- KNÖRIG, A., WETTACH, R. y COHEN, J., 2009. Fritzing: A Tool for Advancing Electronic Prototyping for Designers. *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction* [en línea]. New York, NY, USA: ACM, pp. 351–358. [Consulta: 8 junio 2018]. ISBN 978-1-60558-493-5. DOI 10.1145/1517664.1517735. Disponible en: <http://doi.acm.org/10.1145/1517664.1517735>.
- LANDABURO, J., 2013. *Sistema informático para el ensamblaje y edición de secuencias de ácidos nucleicos*. Investigación. Universidad de las Ciencias Informáticas: s.n.
- MARCELLO, V. y ASTUDILLO, H., 2018. Patrones de Diseño. [en línea]. [Consulta: 20 febrero 2018]. Disponible en: http://www.academia.edu/4726489/Patrones_de_Dise%C3%B1o_Contento_Patrones_GRASP_Experto_Creador_Bajo_Acoplamiento.
- MARIN, D., 2004. 2 Estándares de codificación. [en línea]. [Consulta: 28 marzo 2018]. Disponible en: <http://www.aspl.es/fact/files/aspl-fact/estandares-node2.html>.
- MARTÍNEZ, E.V. y MÉNDEZ, R.F., 2015. Extensiones de Visual Paradigm para la generación de productos de trabajo de apoyo a la Especificación de requisitos de software. [en línea], [Consulta: 8 junio 2018]. Disponible en: <http://repositorio.uci.cu/jspui/handle/123456789/7303>.
- MIKE, 2017. Arduino Hacking from the Command Line. *Raspberry VI* [en línea]. [Consulta: 29 mayo 2018]. Disponible en: <http://www.raspberrypi.org/stories/arduino-cli.html>.
- MIR, Y.O.V. y PÉREZ, D.V., 2010. *Desarrollo de la extensión para la configuración del módulo de adquisición de datos del SCADA Guardián del ALBA*. S.l.: s.n.
- MORALES, E., 2013. INSTANCIAS UML.docx. *Scribd* [en línea]. [Consulta: 31 mayo 2018]. Disponible en: <https://es.scribd.com/document/181939502/INSTANCIAS-UML-docx>.
- NARANJO, F.J., 2016. Conceptos básicos de redes TCP/IP. . S.l.: s.n.,

Referencias Bibliográficas

- NATIONAL INSTRUMENTS, 2006. Comunicación Serial con Control de Flujo por Software - National Instruments. [en línea]. [Consulta: 14 marzo 2018]. Disponible en: <http://digital.ni.com/public.nsf/allkb/FF390A4A51F5171E862575D800538186>.
- NAVARRO, J.M., 2004. *Diseño del Sistema de Tarjeta de Crédito Con UML* [en línea]. Investigativa. Universidad Nacional Mayor de San Marcos: s.n. Disponible en: http://sisbib.unmsm.edu.pe/bibvirtualdata/Tesis/Basic/mendoza_nj/Cap5.pdf.
- PAZ, M.J.A.M., 2016. Análisis del proceso de pruebas de calidad de software. [en línea]. S.I.: Disponible en: <http://dx.doi.org/10.16925/in.v12i20.1482>.
- PERES, M. y LAZARO, S.D., 2012. Arduide (A Qt-based Arduino IDE) - mupuf.org. [en línea]. [Consulta: 8 junio 2018]. Disponible en: <http://www.mupuf.org/project/arduide.html>.
- PESCE, M.Á., 2014. La Domótica con un enfoque antropotécnico. [en línea]. S.I.: [Consulta: 17 mayo 2018]. Disponible en: http://retys.com/wp-content/uploads/2015/07/La_domotica_con_un_enfoque_antropotecnico.pdf.
- PMOINFORMATICA.COM, P. por, 2016. Pruebas de caja negra ISTQB. [en línea]. [Consulta: 23 abril 2018]. Disponible en: <http://www.pmoinformatica.com/2016/04/pruebas-caja-negra-istqb.html>.
- PRATOMO, A.B. y PERDANA, R.S., 2017. Arduviz, a visual programming IDE for arduino. *2017 International Conference on Data and Software Engineering (ICoDSE)*. S.I.: s.n., pp. 1-6. DOI 10.1109/ICoDSE.2017.8285871.
- PRESSMAN, R.S., 2005. *Ingeniería del software un enfoque práctico*. S.I.: s.n.
- QT 5.10, 2017. Qt 5.10. [en línea]. [Consulta: 14 marzo 2018]. Disponible en: <http://doc.qt.io/qt-5/>.
- RAMIREZ, E.A., 2009. *Diagrama de Despliegue* [en línea]. 2009. S.I.: s.n. [Consulta: 6 marzo 2018]. Disponible en: <https://www.google.com/cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=18&ved=0ahUKEwjMzozJptjZAhXJTd8KHQmOB-IQFgh2MBE&url=http%3A%2F%2Fvirtual.usalesiana.edu.bo%2Fweb%2Fpractica%2Farchiv%2Fdespli2.doc&usg=AOvVaw1wTgWE0I5RNUWR31umLzuB>.
- RIVERA, D.C., S, A., AGUIRRE, C., A, F., PINEDA, A. y A, A., 2017. Diseño e implementación de un módulo de entrenamiento de automatización y control utilizando PLC Controllino, programado en Lenguaje C, para actividades prácticas en los laboratorios de electrónica de la Universidad Cooperativa de Colombia sede Santa Marta. [en línea], [Consulta: 8 junio 2018]. Disponible en: <http://repository.ucc.edu.co/handle/ucc/1839>.
- ROBLOGS, 2018. 5 entornos gráficos para Arduino – robologs. [en línea]. [Consulta: 21 mayo 2018]. Disponible en: <https://robologs.net/2013/11/05/5-entornos-graficos-para-arduino/>.
- RUIZ, J.A.G., 2013. *Introducción al Lenguaje C / C++*. S.I.: s.n.

Referencias Bibliográficas

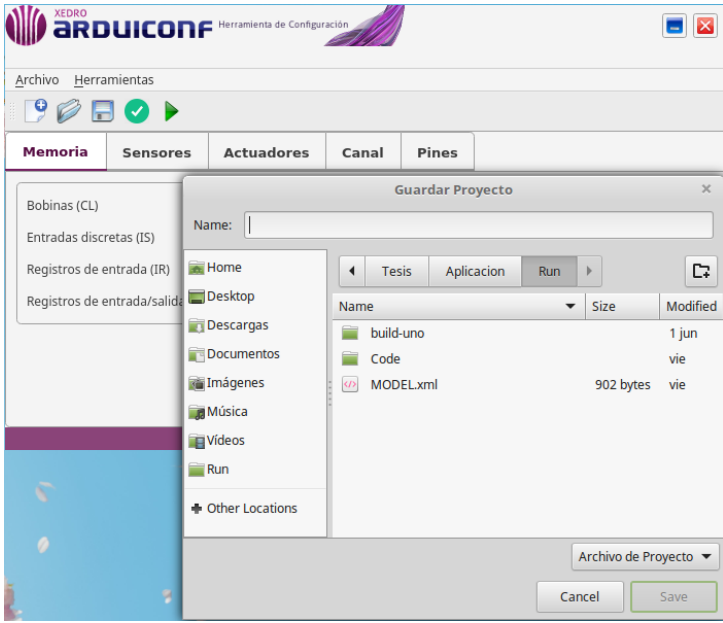
- SÁNCHEZ, T.R., 2015. *Metodología de desarrollo para la Actividad productiva de la UCI*. Universidad de las Ciencias Informáticas: s.n.
- SANCHIDRIAN, E.P., 2014. *Método para determinar la complejidad de los requisitos funcionales de software* [en línea]. La Habana: Universidad de las Ciencias Informáticas. [Consulta: 7 junio 2018]. Disponible en: https://repositorio_institucional.uci.cu/jspui/bitstream/ident/8552/1/Daimara%20Mustelier%20Sanchidrian-TM.pdf.
- SIERRA, M. y LÓPEZ, P., 2014. INGENIERÍA DEL SOFTWARE I Modelado de Requisitos. [en línea]. S.I.: Disponible en: <https://www.ctr.unican.es/asignaturas/is1/is1-p03-trans.pdf>.
- SIFONTES, R.S. y AVILA, Y.F., 2015. *Desarrollo del Módulo Procesamiento de Audiovisuales para el sistema XABAL Arkheia 2.1 para la OAHCE*. La Habana: Universidad de las Ciencias Informáticas.
- SQLITE, 2018. SQLite Home Page. [en línea]. [Consulta: 31 mayo 2018]. Disponible en: <https://www.sqlite.org/index.html>.
- SQLITEBROWSER, 2018. DB Browser for SQLite The Official home of the DB Browser for SQLite. [en línea]. [Consulta: 10 abril 2018]. Disponible en: <http://sqlitedbbrowser.org/>.
- TALE, S., 2016. *C++: The Ultimate Beginners Guide to C++ Programming*. USA: CreateSpace Independent Publishing Platform. ISBN 978-1-5407-4212-4.
- TEDESCHI, N., 2015. ¿Qué es un patrón de diseño? ,
- TRELLINI, L. A., 2015. Principios de Diseño GRASP Diseñando objetos con responsabilidad. [en línea]. S.I.: Disponible en: <http://cs.uns.edu.ar/~atrellini/ayds/downloads/Clases/18.%20AyDS%20-%20Principios%20GRASP.pdf>.
- TRIANA BARREDA, E., 2017. Desarrollo de un equipo para la realización de prácticas de Fundamentos de Informática utilizando CODEBLOCKS y la plataforma Arduino. [en línea], [Consulta: 8 junio 2018]. Disponible en: <http://uvadoc.uva.es:80/handle/10324/23418>.
- VIDAL-SILVA, C.L., PHAM, T.T., VILLARROEL, R.H. y PHILOMINRAJ, A., 2018. Integración de Modelos de Análisis y Diseño de Interface de Punto de Unión JPI en la Búsqueda de un Desarrollo Modular de Software Orientado a Aspectos. *Información Tecnológica*, vol. 29, no. 1.
- VILLAMIZAR SUAZA, K., 2014. *Definición de equivalencias entre historias de usuario y especificaciones en UN-LENCEP para el desarrollo ágil de software* [en línea]. S.I.: s.n. Disponible en: <http://www.bdigital.unal.edu.co/11631/1/1128431389.2014.pdf>.
- VISUAL PARADIGM, 2017. Visual Paradigm - UML, Agile, PMBOK, TOGAF, BPMN and More! [en línea]. [Consulta: 6 diciembre 2017]. Disponible en: <https://www.visual-paradigm.com/features/>.
- WELICKI, L., 2018. El Patrón Singleton. [en línea]. [Consulta: 10 abril 2018]. Disponible en: <https://msdn.microsoft.com/es-es/library/bb972272.aspx>.

Referencias Bibliográficas

ZARAGOZA MAKERSPACE, 2017. Arduino online - Como usar ArduBlockly. *Zaragoza MakerSpace* [en línea]. [Consulta: 21 mayo 2018]. Disponible en: <https://zaragozmakerspace.com/index.php/arduino-online-como-usar-ardublockly/>.

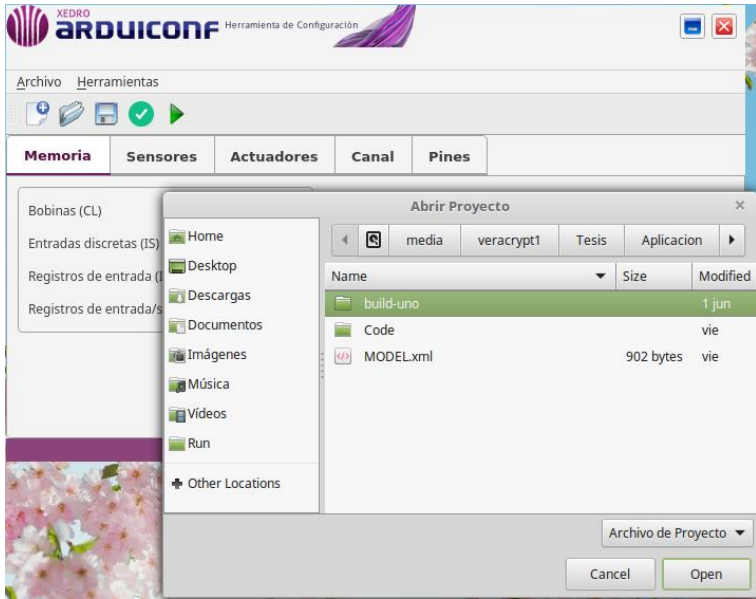
ANEXOS

Anexo 1 Descripción de la HU 2 Salvar desde un archivo *xml* un Proyecto de configuración.

Número: HU 2	Nombre del requisito: Salvar en un archivo <i>xml</i> un Proyecto de configuración.
Programador: Lisardo Garcia Jane	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 3 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 24 horas
<p>Descripción: Al hacer clic en el menú “Archivo” y seleccionar la opción “Guardar” o en el icono “Guardar” en la barra de Herramientas, si el proyecto actual no ha sido previamente guardado se abre una ventana donde se debe especificar la ruta donde será guardado el proyecto de configuración con extensión <i>xml</i>. Una vez seleccionada la ruta donde estará el Proyecto se hace clic en el botón “Aceptar” y se muestra en la aplicación dando la posibilidad de continuar editando las configuraciones antes realizadas. Si el proyecto ya había sido guardado anteriormente simplemente se guardan los cambios y se puede continuar la edición de la configuración.</p>	
Observaciones: NA	
Prototipo de Interfaz:	
	

Anexos


Anexo 2 Descripción de la HU 3 Cargar en un archivo *xml* un Proyecto de configuración.

Anexo 2 Descripción de la HU 3 Cargar en un archivo <i>xml</i> un Proyecto de configuración.	
Número: HU 3	Nombre del requisito: Cargar en un archivo <i>xml</i> un Proyecto de configuración.
Programador: Lisardo Garcia Jane	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 3 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 48 horas
Descripción: Al hacer clic en el botón “Archivo” o en el icono “Cargar Proyecto” se abre una ventana donde se debe especificar la ruta donde se encuentra guardado el proyecto de configuración con la extensión de archivo <i>xml</i> . Una vez seleccionado el archivo de configuración se hace clic en el botón “Aceptar” y se muestran en la aplicación las configuraciones cargadas del archivo antes seleccionado.	
Observaciones: NA	
Prototipo de Interfaz:	
	

Anexo 3 Descripción de la HU 4 Definir la cantidad de registros de cada tipo de Tabla de Memoria.

Anexo 3 Descripción de la HU 4 Definir la cantidad de registros de cada tipo de Tabla de Memoria.	
Número: HU 4	Nombre del requisito: Definir la cantidad de registros de cada tipo de Tabla de Memoria.

Anexos

Programador: Lisardo Garcia Jane	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 7 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 96 horas
<p>Descripción: Al seleccionar la pestaña “Memoria” se muestran en dicha pestaña los campos “Coils”, “Input Status”, “Input Register”, “Holding Register” correspondiente a cada tipo de tabla de memoria Modbus que se puede configurar y los correspondientes <i>SpinBox</i> donde se define la cantidad de entradas para cada tabla de memoria.</p>	
Observaciones: NA	
Prototipo de Interfaz:	
	

Anexo 4 Descripción de la HU 5 Adicionar un nuevo sensor.

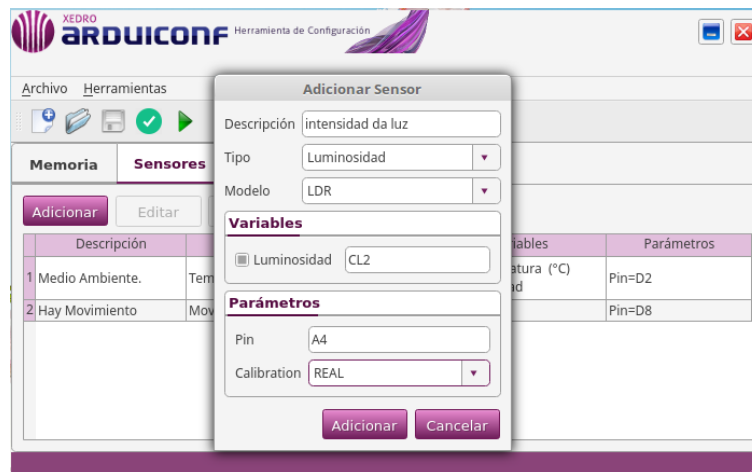
Número: HU 5	Nombre del requisito: Adicionar un nuevo sensor.
Programador: Lisardo Garcia Jane	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 7 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 96 horas
<p>Descripción: Al hacer clic en la pestaña “Sensores” y luego en el botón “Adicionar” o dar clic derecho en el área en blanco de dicha pestaña y presionar sobre la opción “Insertar un sensor”, se muestra una ventana que contiene los parámetros necesarios para configurar un sensor, dígame “Tipo”, “Modelo”, “Pin” y un conjunto de variables y parámetros que se configuran</p>	

Anexos

teniendo en cuenta el tipo de sensor seleccionado. Una vez llenados los campos si están correctamente escritos se debe habilitar el botón “Aceptar” y al presionar sobre este se debe adicionar el sensor a la lista de sensores en la pestaña “Sensores”. En caso de que exista algún error en los campos editables, el botón “Aceptar” nunca se habilitará.

Observaciones: NA

Prototipo de Interfaz:



Anexo 5 Descripción de la HU 6 Listar Sensores configurados.

HU 6 Listar Sensores configurados	
Número: HU 6	Nombre del requisito: Listar Sensores configurados.
Programador: Lisardo Garcia Jane	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 7 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 96 horas
Descripción: Al seleccionar la pestaña “Sensores” se debe mostrar un listado con los sensores que hayan sido configurados.	
Observaciones: Debe haber sido configurado al menos un sensor.	
Prototipo de Interfaz:	

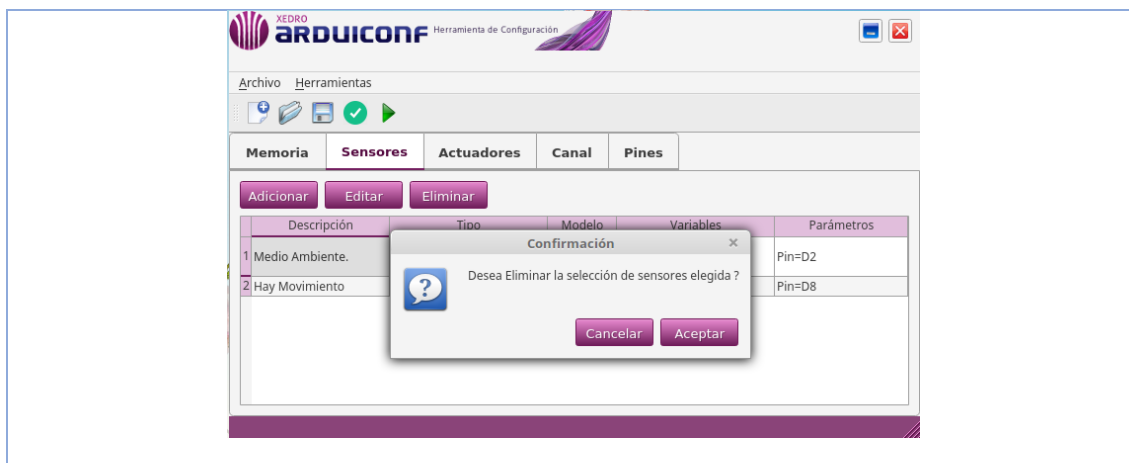
Anexos



Anexo 6 Descripción de la HU 8 Eliminar sensores de la lista.

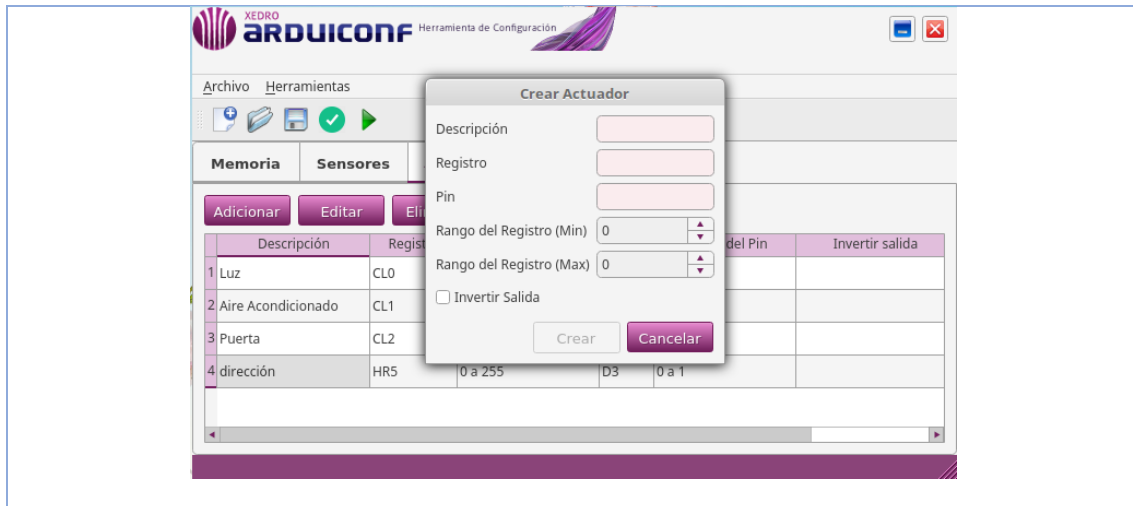
Número: HU 8	Nombre del requisito: Eliminar sensores de la lista.
Programador: Lisardo Garcia Jane	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 7 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 96 horas
<p>Descripción: Al seleccionar la pestaña “Sensores” se debe revelar un listado de sensores configurados. Al seleccionar la primera columna de uno o varios sensores se debe habilitar el botón “Eliminar”, al presionar sobre el mismo o dar clic derecho sobre los sensores seleccionando la opción “Eliminar el o los sensores seleccionados” se debe mostrar un mensaje de confirmación para asegurarnos que se eliminará el o los sensores seleccionados. Al presionar sobre el botón “Aceptar” se eliminan de la lista de sensores configurados el o los sensores seleccionados.</p>	
<p>Observaciones: Debe existir al menos un sensor configurado.</p>	
<p>Prototipo de Interfaz:</p>	

Anexos



Anexo 7 Descripción de la HU 9 Adicionar un nuevo actuador.

Número: HU 9	Nombre del requisito: Adicionar un nuevo actuador y definir a qué pin será conectado, qué registro de memoria se utilizará para accionarlo y qué conversión se utilizará para convertir el valor del registro en señal de salida por el pin.
Programador: Lisardo Garcia Jane	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 7 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 96 horas
Descripción: Al hacer clic en la pestaña “Actuadores” y luego en el botón “Agregar” o dar clic derecho en el área en blanco de dicha pestaña y presionar sobre la opción “Insertar un actuador”, se debe mostrar una ventana que contiene los parámetros necesarios para configurar un actuador, dígame “Registro”, “Pin” y el valor máximo y mínimo del registro. Una vez llenados los campos si están correctamente escritos se habilita el botón “Aceptar” y al presionar sobre éste se debe adicionar el actuador a la lista de actuadores en la pestaña “Actuadores”. En caso de que exista algún error en los campos editables, el botón “Aceptar” nunca se habilitará.	
Observaciones: NA	
Prototipo de Interfaz:	



Anexo 8 Descripción de la HU 10 Listar Actuadores configurados.

Número: HU 10	Nombre del requisito: Listar Actuadores configurados.
Programador: Lisardo Garcia Jane	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 7 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 40 horas
Descripción: Al seleccionar la pestaña "Actuadores" se debe mostrar un listado con los actuadores que hayan sido configurados.	
Observaciones: Debe haber sido configurado al menos un actuador.	
Prototipo de Interfaz:	

Anexos



Anexo 9 Descripción de la HU 11 Modificar los datos configurados de un Actuador.

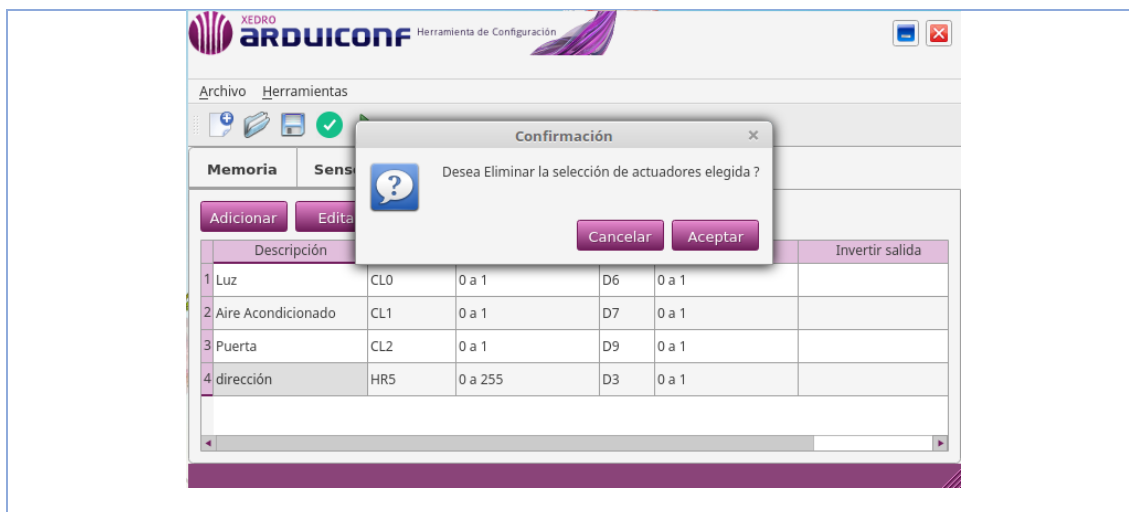
Número: HU 11	Nombre del requisito: Modificar los datos configurados de un Actuador.
Programador: Lisardo Garcia Jane	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 7 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 96 horas
Descripción: Al seleccionar la pestaña “Actuadores”, se debe mostrar un listado de actuadores configurados. Para modificar los datos de alguno de ellos debe estar previamente seleccionado y luego se podrá editar al presionar el botón “Editar” o al dar clic derecho sobre el actuador y escoger la opción “Editar actuador”. En ambos casos se debe mostrar una ventana de edición, la cual debe permitir modificar los datos del actuador seleccionado. Una vez modificados se presiona el botón “Aceptar” y se debe mostrar los datos actualizados en el listado.	
Observaciones: Debe existir al menos un actuador configurado.	
Prototipo de Interfaz:	

Anexos



Anexo 10 Descripción de la HU 12 Eliminar actuadores de la lista.

Número: HU 12	Nombre del requisito: Eliminar actuadores de la lista.
Programador: Lisardo Garcia Jane	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 7 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 96 horas
Descripción: Al seleccionar la pestaña “Actuadores” se debe visualizar un listado de actuadores configurados. Al seleccionar la primera columna de uno o varios actuadores se debe habilitar el botón “Eliminar”, al presionar sobre el mismo o dar clic derecho sobre los actuadores seleccionando la opción “Eliminar el o los actuadores seleccionados” se debe mostrar un mensaje de confirmación para asegurarnos que se eliminará el o los actuadores seleccionados. Al presionar sobre el botón “Aceptar” se debe eliminar de la lista de actuadores configurados el o los actuadores seleccionados.	
Observaciones: Debe existir al menos un actuador configurado.	
Prototipo de Interfaz:	




Anexo 11 Descripción de la HU 13 Verificar código.

Número: HU 13	Nombre del requisito: Verificar configuración.
Programador: Lisardo Garcia Jane	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 7 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 96 horas
Descripción: Una vez realizadas las configuraciones se presiona clic en el botón “Verificar configuración” y se debe mostrar una ventana informativa especificando si encontró o no errores en dichas configuraciones.	
Observaciones: Debe haber seleccionado una placa Arduino.	
Prototipo de Interfaz: NA	

Anexo 12 Descripción de la HU 14 Listar pines de entrada/salida, con toda la información útil sobre ellos.

Número: HU 14	Nombre del requisito: Listar pines de entrada/salida, con toda la información útil sobre ellos.
Programador: Lisardo Garcia Jane	Iteración Asignada: 1

Anexos

Prioridad: Alta	Tiempo Estimado: 5 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 72 horas
Descripción: Una vez seleccionada la placa arduino que será usada en el menú “Herramientas” >> “Seleccionar placa”, se selecciona la pestaña “Pines”, se debe mostrar un listado de pines en correspondencia con la placa seleccionada. Además, se debe reflejar el estado de los pines que ya han sido configurados, si están siendo usados por un <i>shield</i> de <i>Ethernet</i> , si son <i>PWM</i> o no y el tipo (Digital o Analógico) de pin. Si se selecciona otra placa esta información se debe actualizar.	
Observaciones: Debe haber seleccionado una placa Arduino.	
Prototipo de Interfaz:	
	

Anexo 13 Descripción de la HU 14 Elegir y configurar canal de comunicación (Serial, TCP o WIFI).

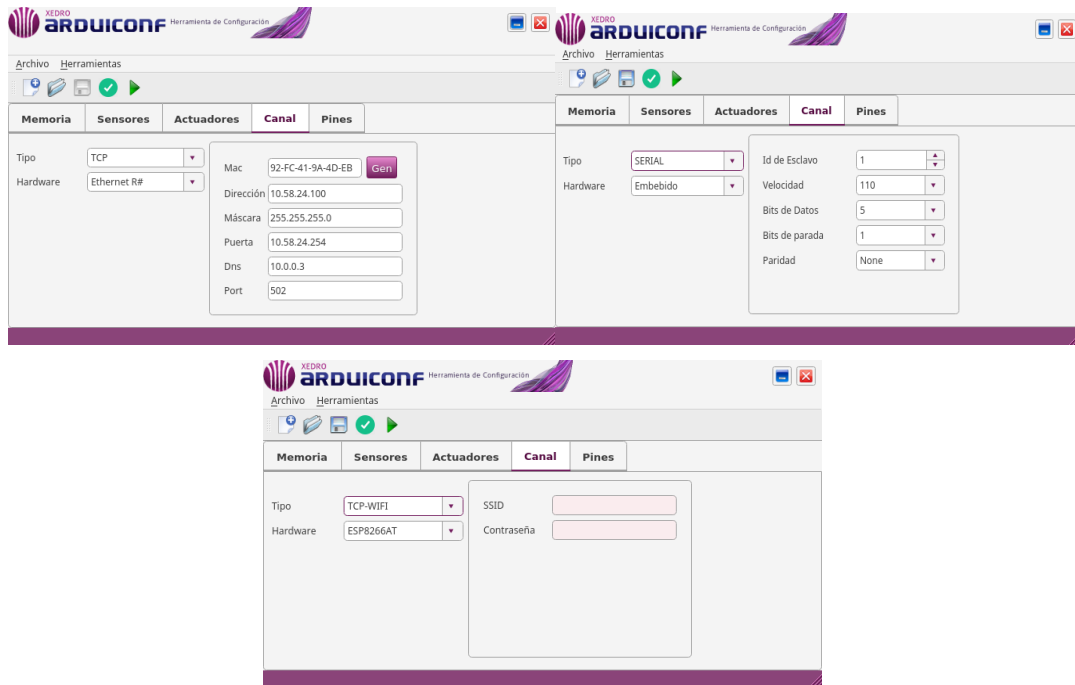
Número: HU 15	Nombre del requisito: Elegir y configurar canal de comunicación (Serial, <i>TCP</i> o <i>WIFI</i>).
Programador: Lisardo Garcia Jane	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 7 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 96 horas
Descripción: Al seleccionar la pestaña “Canal” se deben mostrar los campos para configurar el canal de comunicación (Serie, <i>TCP</i> o <i>WIFI</i>) que seleccione el usuario. Si selecciona “ <i>TCP</i> ” se debe seleccionar el tipo de “ <i>Hardware</i> ” y se debe llenar en la parte derecha de la pestaña	

Anexos

los parámetros “Mac”, “IP”, “Mascara”, “Puerta”, “DNS”, “Puerto”. En caso de seleccionar comunicación tipo “Serie” igualmente se debe seleccionar el tipo de “Hardware” y en la parte derecha de la pestaña se debe visualizar los cuantificadores necesarios para configurar dicho canal que pueden ser modificados según las necesidades del usuario. En caso de seleccionar “WIFI” al igual que los otros canales se debe seleccionar el tipo de “Hardware” y configurar los parámetros “SSID” y “Password”⁴⁰.

Observaciones: Debe uno de los canales de comunicación.

Prototipo de Interfaz:



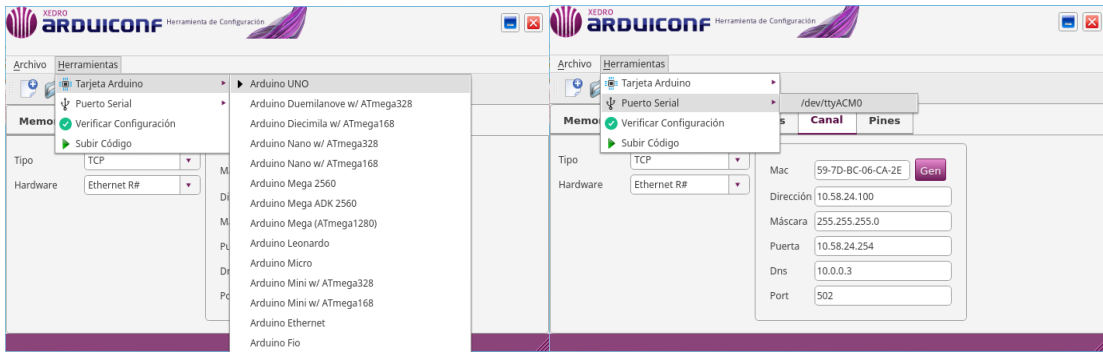
Anexo 14 Descripción de la HU 15

Elegir modelo de placa Arduino y puerto donde está conectada.

Número: HU 16	Nombre del requisito: Elegir modelo de placa Arduino y puerto donde está conectada.
Programador: Lisardo Garcia Jane	Iteración Asignada: 1

⁴⁰ Contraseña

Anexos

Prioridad: Alta	Tiempo Estimado: 7 días
Riesgo en desarrollo: Fallos eléctricos	Tiempo real: 96 horas
<p>Descripción: Al seleccionar en el menú “Herramientas” la opción “Seleccionar placa” se debe mostrar un listado con todas las placas soportadas por la aplicación “Arduiconf” y se elige la placa Arduino según las necesidades del usuario. Luego desde el menú “Herramientas” se selecciona la opción “Seleccionar puerto” y se debe mostrar los puertos donde exista una placa Arduino conectada.</p>	
<p>Observaciones: Debe haber al menos una placa Arduino conectada.</p>	
<p>Prototipo de Interfaz:</p> 	

Anexo 15 Prueba de caja negra sobre la HU 2 Salvar en un archivo xml un proyecto de configuración.

Escenario 1.1 Salvar un proyecto de configuración.	
Descripción	Se intentan salvar los cambios realizados sobre un proyecto de configuración.
Variable	NA
Respuesta del sistema	Se guardan todos los cambios realizados en un proyecto de configuración.
Flujo central	Desde el ambiente de edición en la barra de Menú se presiona el menú "Archivo", se despliegan varias opciones y se selecciona la opción "Salvar". / Desde el ambiente de edición en la barra de Herramientas se presiona el botón "Salvar".
Escenario 1.2 Salvar un proyecto de configuración / Opción "Aceptar".	
Descripción	Se intentan salvar los cambios realizados sobre un proyecto de configuración.

Variable	NA
Respuesta del sistema	Se guarden todos los cambios realizados en un proyecto de configuración.
Flujo central	Desde el ambiente de edición en la barra de Menú se presiona el menú "Archivo" y se selecciona la opción "Salvar". Se muestra una ventana donde se especifica la ruta y el nombre del proyecto de configuración que se desea salvar. Se presiona sobre el botón "Aceptar". / Desde el ambiente de edición en la barra de Herramientas se presiona el botón "Salvar". Se muestra una ventana donde se especifica la ruta y el nombre del proyecto de configuración que se desea salvar. Se presiona sobre el botón "Aceptar".
Escenario 1.3 Salvar un proyecto de configuración / Opción "Cancelar".	
Descripción	Se intentan salvar los cambios realizados sobre un proyecto de configuración.
Variable	NA
Respuesta del sistema	Se cierra la ventana y se vuelve al ambiente de edición sin haber guardado ninguno de los cambios realizados.
Flujo central	Desde el ambiente de edición en la barra de Menú se presiona el menú "Archivo" y se selecciona la opción "Salvar". Se muestra una ventana donde se especifica la ruta y el nombre del proyecto de configuración que se desea salvar. Se presiona sobre el botón "Cancelar". / Desde el ambiente de edición en la barra de Herramientas se presiona el botón "Salvar". Se muestra una ventana donde se especifica la ruta y el nombre del proyecto de configuración que se desea salvar. Se presiona sobre el botón "Cancelar".

Anexo 15 Prueba de caja negra sobre la HU 3 Cargar en un archivo xml un proyecto de configuración.

Escenario 1.1 Cargar un proyecto de configuración / Opción "Abrir".	
Descripción	Se intenta cargar un proyecto de configuración existente desde un archivo con extensión xml.
Variables	NA
Respuesta del sistema	Se cargan en la interfaz las configuraciones presentes en el archivo xml cargado.
Flujo central	Desde el ambiente de edición en la barra de Menú al seleccionar el menú "Archivo" se despliegan varias opciones y se selecciona la opción "Abrir". Se muestra una

Anexos

	ventana donde se selecciona el archivo xml a cargar. Se presiona el botón "Abrir". / Desde el ambiente de edición en la barra de Herramientas al presionar sobre el botón "Abrir". Se muestra una ventana donde se selecciona el archivo xml a cargar. Se presiona el botón "Abrir".
Escenario 1.2 Cargar un proyecto de configuración / Opción "Cancelar"	
Descripción	Se intenta cargar un proyecto de configuración existente desde un archivo con extensión xml.
Variable	NA
Respuesta del sistema	No se carga el archivo xml y no hay cambios en la interfaz.
Flujo central	Desde el ambiente de edición en la barra de Menú al seleccionar el menú "Archivo" se despliegan varias opciones y se selecciona la opción "Abrir". Se muestra una ventana donde se selecciona el archivo con extensión xml a cargar. Se escoge la opción "Cancelar". / Desde el ambiente de edición en la barra de Herramientas se se presiona sobre el botón "Abrir". Se muestra una ventana donde se selecciona el archivo con extensión xml a cargar. Se escoge la opción "Cancelar".

Anexo 16 Prueba de caja negra sobre la HU 4 Definir la cantidad de registros de cada tipo de Tabla de Memoria.

Escenario 1.1 Definir la cantidad de registros de cada tipo de tabla de memoria.			
Descripción	Se intenta definir la cantidad de registros de cada tipo (<i>Coil</i> , <i>Input Status</i> , <i>Input Register</i> y <i>Holding Register</i>) de tabla de memoria Modbus.		
Coils	V	0	
Input Status	V	2	
Input Register	V	3	
Holding Register	V	8	
Respuesta del sistema	Permanece definida la cantidad de registros para cada tipo de tabla de memoria Modbus.		

Anexos

Flujo central	Desde el ambiente de edición en la pestaña "Memoria" se especifica en cada SpinBox la cantidad de registros <u>Coils</u> , <u>Input Status</u> , <u>Input Register</u> y <u>Holding Register</u> .
----------------------	--

Anexo 17 Prueba de caja negra sobre la HU 6

Listar Sensores configurados.

Escenario 1.1 Listar sensores configurados.	
Descripción	Se intentan listar los sensores configurados.
Variables	NA
Respuesta del sistema	Se muestra un listado con los sensores configurados con los campos: Descripción, Tipo, Modelo, Variables y Parámetros.
Flujo central	Desde el ambiente de edición se selecciona la pestaña "Sensores".

Anexo 18 Prueba de caja negra sobre la HU 8

Eliminar sensores de la lista.

Escenario 1.1 Eliminar sensores de la lista / Opción "Aceptar".	
Descripción	Se intenta eliminar uno o varios sensores seleccionados.
Variables	NA
Respuesta del sistema	Se elimina el sensor o los sensores seleccionados de la lista de sensores.
Flujo central	Desde el ambiente de edición al seleccionar la pestaña "Sensores" se muestra un listado de sensores configurados. Al seleccionar la primera celda de uno o varios sensores de la lista, se habilita en dicha pestaña el botón "Eliminar". Al presionar sobre dicho botón se muestra un mensaje de confirmación y se presiona el botón "Aceptar". / Desde el ambiente de edición al seleccionar la pestaña "Sensores" se muestra un listado de sensores configurados. Al seleccionar la primera celda de uno o varios sensores de la lista, y da click derecho sobre la tabla, se muestran varias opciones y se selecciona "Eliminar el o los sensores seleccionados". Al presionar sobre dicha opción se muestra un mensaje de confirmación y se presiona el botón "Aceptar".
Escenario 1.2 Eliminar sensores de la lista / Opción "Cancelar".	

Anexos

Descripción	Se intenta eliminar uno o varios sensores seleccionados.
Variables	NA
Respuesta del sistema	No se elimina el sensor seleccionado.
Flujo central	Desde el ambiente de edición al seleccionar la pestaña "Sensores" se muestra un listado de sensores configurados. Al seleccionar la primera celda de uno o varios sensores de la lista, se habilita en dicha pestaña el botón "Eliminar". Al presionar sobre dicho botón se muestra un mensaje de confirmación y se presiona el botón "Cancelar". / Desde el ambiente de edición al seleccionar la pestaña "Sensores" se muestra un listado de sensores configurados. Al seleccionar la primera celda de uno o varios sensores de la lista, y da click derecho sobre la tabla, se muestran varias opciones y se selecciona "Eliminar el o los sensores seleccionados". Al presionar sobre dicha opción se muestra un mensaje de confirmación y se presiona el botón "Cancelar".

Anexo 19 Prueba de caja negra sobre la HU 9 Adicionar un actuador.

Escenario 1.1 Adicionar un actuador / Opción "Crear".			
Descripción	Se intenta adicionar un nuevo actuador.		
Descripción		V	Prendedor de Luces
Registro		V	CL3
Pin		V	D5
Rango del registro	Min	V	0 (Campo no editable)
	Max	V	1 (Campo no editable)
Invertir Salida		V	(Checkbox seleccionado)
Respuesta del sistema	Adiciona el actuador creado a la lista de actuadores.		
Flujo central	Desde el ambiente de edición al seleccionar la pestaña "Actuadores", se presiona el botón "Adicionar". Se muestra en una ventana los campos: Descripción, Registro, Pin, Rango del registro (Min), Rango del registro (Max) e Invertir Salida, se llenan dichos campos y se presiona el botón "Adicionar". / Desde el ambiente de edición al seleccionar la pestaña "Actuadores", se da click derecho sobre la lista de actuadores y se selecciona la opción "Adicionar un actuador". Se muestra en una ventana los		

Anexos

	campos: Descripción, Registro, Pin, Rango del Registro (Min), Rango del Registro (Max) e Invertir Salida, se llenan dichos campos y se presiona el botón "Adicionar".			
Escenario 1.2 Adicionar actuador / Opción "Cancelar".				
Descripción	Se intenta adicionar un nuevo actuador.			
Descripción		V	Regulador de intensidad de las Luces	
Registro		V	HR5	
Pin		V	D4	
Rango del registro	Min	V	10	
	Max	V	30	
Invertir Salida		V	(Checkbox no seleccionado)	
Respuesta del sistema	Cierra la ventana y vuelve al ambiente de edición.			
Flujo central	Desde el ambiente de edición al seleccionar la pestaña "Actuadores", se presiona el botón "Adicionar". Se muestra en una ventana los campos: Descripción, Registro, Pin, Rango del registro (Min), Rango del registro (Max) e Invertir Salida, se llenan dichos campos y se presiona el botón "Cancelar". / Desde el ambiente de edición al seleccionar la pestaña "Actuadores", se da click derecho sobre la lista de sensores y se selecciona la opción "Adicionar un actuador". Se muestra en una ventana los campos: Descripción, Registro, Pin, Rango del Registro (Min), Rango del Registro (Max) e Invertir Salida, se llenan dichos campos y se presiona el botón "Cancelar".			
Escenario 1.3 Adicionar actuador / Campos vacíos.				
Descripción	Se intenta adicionar un nuevo actuador.			
Descripción		V	Regulador de intensidad de las Luces	V
Registro		I	(Campo vacío)	V
Pin		V	D4	I
Rango del registro	Min	V	0 (Campo no editable)	V
	Max	V	0 (Campo no editable)	V
Invertir Salida		V	(Checkbox no seleccionado)	V

Anexos

Respuesta del sistema	Muestra los campos vacíos resaltados en rojo y con un <i>ToolTip</i> que explica el error en cada uno. No se activa el botón "Aceptar".
Flujo central	Desde el ambiente de edición al seleccionar la pestaña "Actuadores", se presiona el botón "Adicionar". Se muestra en una ventana los campos: Descripción, Registro, Pin, Rango del registro (Min), Rango del registro (Max) e Invertir Salida, no se llenan el campo Registro ni el campo Pin. / Desde el ambiente de edición al seleccionar la pestaña "Actuadores", se da click derecho sobre la lista de actuadores y se selecciona la opción "Adicionar un actuador". Se muestra en una ventana los campos: Descripción, Registro, Pin, Rango del Registro (Min), Rango del Registro (Max) e Invertir Salida, no se llenan el campo Registro ni el campo Pin.

Anexo 20 Prueba de caja negra sobre la HU 10

Listar actuadores configurados.

Escenario EC 1.1 Listar actuadores configurados.	
Descripción	Se intenta listar los actuadores configurados.
Variable	NA
Respuesta del sistema	Se muestra un listado con los actuadores configurados con los campos: Descripción, Registro, Rango del registro, Pin, Rango del pin, Invertir salida.
Flujo central	Desde el ambiente de edición se selecciona la pestaña "Actuadores".

Anexo 21 Prueba de caja negra sobre la HU 11

Modificar los datos configurados de un Actuador.

Escenario 1.1 Modificar los datos configurados de un Actuador / Opción "Aceptar".			
Descripción	Se intenta modificar los datos configurados de un Actuador creado.		
Descripción		V	Prendedor de Luces
Registro		V	CL7
Pin		V	D3
Rango del registro	Min	V	0 (Campo no editable)
	Max	V	1 (Campo no editable)
Invertir Salida		V	(Checkbox seleccionado)

Anexos

Respuesta del sistema	Modifica los datos del actuador seleccionado.		
Flujo central	Desde el ambiente de edición se escoge la pestaña "Actuadores", se selecciona un actuador ya creado y se presiona el botón "Editar". Se muestra en una ventana los campos: Descripción, Registro, Pin, Rango del Registro (Min), Rango del Registro (Max) e Invertir Salida, se modifican los campos y se presiona sobre el botón "Aceptar". / Desde el ambiente de edición se presiona sobre la pestaña "Actuadores", se da click derecho sobre uno de los actuadores y se selecciona la opción "Editar el actuador seleccionado", se muestra en una ventana los campos: Descripción, Registro, Pin, Rango del Registro (Min), Rango del Registro (Max) e Invertir Salida, se modifican los campos y se presiona sobre el botón "Aceptar".		
Escenario 1.2 Modificar los datos configurados de un Actuador / Opción "Cancelar".			
Descripción	Se intenta modificar los datos configurados de un Actuador creado.		
Descripción		V	Regulador de intensidad de las Luces
Registro		V	HR5
Pin		V	D4
Rango del registro	Min	V	(Campo editable)
	Max	V	(Campo editable)
Invertir Salida		V	(Checkbox no seleccionado)
Respuesta del sistema	No modifica los valores, cierra la ventana y vuelve al ambiente de edición.		
Flujo central	Desde el ambiente de edición se escoge la pestaña "Actuadores", se selecciona un actuador ya creado y se presiona el botón "Editar". Se muestra en una ventana los campos: Descripción, Registro, Pin, Rango del Registro (Min), Rango del Registro (Max) e Invertir Salida, se modifican o no los campos y se presiona sobre el botón "Cancelar". / Desde el ambiente de edición se presiona sobre la pestaña "Actuadores", se da click derecho sobre uno de los actuadores y se selecciona la opción "Editar el actuador seleccionado", se muestra en una ventana los campos: Descripción, Registro, Pin, Rango del registro (Min), Rango del registro (Max) e Invertir Salida, se modifican o no los campos y se presiona sobre el botón "Cancelar".		
Escenario 1.3 Modificar los datos configurados de un Actuador / Campos vacíos.			
Descripción	Se intenta modificar los datos configurados de un Actuador creado.		

Anexos

Descripción		V	Regulador de intensidad de las Luces	V	Regulador de intensidad de las Luces
Registro		I	(Campo vacío)	V	HR6
Pin		V	D4	I	(Campo vacío)
Rango del registro	Min	V	0 (Campo no Editable)	V	0
	Max	V	0 (Campo no Editable)	V	10
Invertir Salida		V	(Checkbox no seleccionado)	V	(Checkbox no seleccionado)
Respuesta del sistema	Muestra los campos vacíos resaltados en rojo y con un <i>ToolTip</i> que explica el error en cada uno. No se activa el botón "Aceptar".				
Flujo central	Desde el ambiente de edición se escoge la pestaña "Actuadores", se selecciona un actuador ya creado y se presiona el botón "Editar". Se muestra en una ventana los campos: Descripción, Registro, Pin, Rango del registro (Min), Rango del registro (Max) e Invertir Salida, no se llena el campo Registro ni el campo Pin. / Desde el ambiente de edición se presiona sobre la pestaña "Actuadores", se da click derecho sobre uno de los actuadores y se selecciona la opción "Editar el actuador seleccionado", se muestra en una ventana los campos: Descripción, Registro, Pin, Rango del Registro (Min), Rango del Registro (Max) e Invertir Salida, no se llena el campo Registro ni el campo Pin.				

Anexo 22 Prueba de caja negra sobre la HU 12

Eliminar actuadores de la lista.

Escenario 1.1 Eliminar uno o varios actuadores de la lista / Opción "Aceptar".	
Descripción	Se intenta eliminar uno o varios actuadores seleccionados.
Variabes	NA
Respuesta del sistema	Se elimina el actuador o actuadores seleccionados.
Flujo central	Desde el ambiente de edición al seleccionar la pestaña "Actuadores" se muestra un listado de actuadores configurados. Al seleccionar uno o varios de los actuadores de la lista se habilita en dicha pestaña el botón "Eliminar". Al presionar sobre dicho botón se muestra un mensaje de confirmación y se presiona el botón "Aceptar". / Desde el ambiente de edición al seleccionar la pestaña "Actuadores" se muestra un listado de

	actuadores configurados. Al seleccionar uno o varios de los actuadores de la lista y dar click derecho sobre estos se muestran varias opciones y se selecciona "Eliminar el o los actuadores seleccionados". Al presionar sobre dicha opción se muestra un mensaje de confirmación y se presiona el botón "Aceptar".
Escenario 1.2 Eliminar uno o varios actuadores de la lista / Opción "Cancelar".	
Descripción	Se intenta eliminar uno o varios actuadores seleccionados.
Variables	NA
Respuesta del sistema	No se elimina el actuador o actuadores seleccionados.
Flujo central	Desde el ambiente de edición al seleccionar la pestaña "Actuadores" se muestra un listado de actuadores configurados. Al seleccionar uno o varios de los actuadores de la lista se habilita en dicha pestaña el botón "Eliminar". Al presionar sobre dicho botón se muestra un mensaje de confirmación y se presiona el botón "Cancelar". / Desde el ambiente de edición al seleccionar la pestaña "Actuadores" se muestra un listado de actuadores configurados. Al seleccionar uno o varios de los actuadores de la lista y dar click derecho sobre estos se muestran varias opciones y se selecciona "Eliminar el o los actuadores seleccionados". Al presionar sobre dicha opción se muestra un mensaje de confirmación y se presiona el botón "Cancelar".

**Anexo 22 Prueba de caja negra sobre la HU 13
Verificar configuración.**

Escenario 1.1 Verificar configuración / No hay errores en la configuración.	
Descripción	Se intenta verificar la configuración realizada.
Variables	NA
Respuesta del sistema	Notifica que no existen errores en la configuración.
Flujo central	Desde el ambiente de edición en la barra de Menú al presionar sobre el menú "Herramientas" se despliegan varias opciones y se selecciona "Verificar Configuración". / Desde el ambiente de edición en la barra de Herramientas al presionar sobre el botón "Verificar Configuración".
Escenario 1.2 Verificar configuración / Errores en la configuración.	

Anexos

Descripción	Se intenta verificar la configuración realizada.
Variables	NA
Respuesta del sistema	Se muestra un archivo de texto notificando los errores detectados.
Flujo central	Desde el ambiente de edición en la barra de Menú al presionar sobre el menú "Herramientas" se despliegan varias opciones y se selecciona "Verificar Configuración". / Desde el ambiente de edición en la barra de Herramientas al presionar sobre el botón "Verificar Configuración".

Anexo 23 Prueba de caja negra sobre la HU 14

Listar pines de entrada/salida, con toda la información útil sobre ellos.

Escenario 1.1 Listar pines de entrada/salida.	
Descripción	Se intenta listar el estado de los pines teniendo en cuenta la placa seleccionada y la configuración realizada.
Variables	NA
Respuesta del sistema	Se muestra el listado de los pines de la placa seleccionada con los campos: Identificador, Tipo, E/S, PWM y Usado por.
Flujo central	Desde el ambiente de edición se selecciona la pestaña "Pines".

Anexo 24 Prueba de caja negra sobre la HU 15

Elegir y configurar canal de comunicación (Serial, TCP o WIFI).

SC 1 Configurar canal de comunicación TCP		
Escenario 1.1 Elegir y configurar el canal de comunicación / TCP.		
Descripción	Se intenta elegir y configurar el canal de comunicación TCP.	
Tipo	V	TCP
Hardware	V	Ethernet R#
Mac	V	0A-B2-D2-34-17-2F
IP	V	10.58.20.6
Máscara	V	255.255.255.0
Puerta	V	10.58.20.254
DNS	V	10.0.0.3
Puerto	V	8082

Respuesta del sistema	Se configuran los parámetros de ejecución definidos para el canal de comunicación TCP.		
Flujo central	Desde el ambiente de edición se selecciona la pestaña "Canal", a la izquierda de esta pestaña se selecciona el Tipo de canal "TCP" y el Tipo de Hardware a utilizar por el mismo. A la derecha de la pestaña se configuran los parámetros de ejecución del canal seleccionado.		
Escenario 1.2 Elegir y configurar el canal de comunicación TCP / Campos vacíos.			
Descripción	Se intenta elegir y configurar el canal de comunicación TCP.		
Tipo	V	TCP	
Hardware	V	Ethernet R#	
Mac	I	(Campo vacío)	
IP	I	(Campo vacío)	
Máscara	I	(Campo vacío)	
Puerta	I	(Campo vacío)	
DNS	I	(Campo vacío)	
Puerto	I	(Campo vacío)	
Respuesta del sistema	Muestra los campos vacíos resaltados en rojo y con un <i>ToolTip</i> que explica el error en cada uno.		
Flujo central	Desde el ambiente de edición se selecciona la pestaña "Canal", a la izquierda de esta pestaña se selecciona el Tipo de canal "TCP" y el Tipo de Hardware a utilizar por el mismo. A la derecha de la pestaña se configuran los parámetros de ejecución del canal seleccionado.		
SC 2 Configurar canal de comunicación Serial			
Escenario 1.1 Elegir y configurar el canal de comunicación Serial.			
Descripción	Se intenta elegir y configurar el canal de comunicación Serial.		
Tipo	V	Serial	
Hardware	V	Embebido	
ID de Esclavo	V	1	
Velocidad	V	110	
Bit de Datos	V	5	
Bits de Parada	V	1	
Paridad	V	None	

Anexos

Respuesta del sistema	Se configuran los parámetros de ejecución definidos para el canal de comunicación Serial.		
Flujo central	Desde el ambiente de edición se selecciona la pestaña "Canal", a la izquierda de esta pestaña se selecciona el Tipo de canal "Serial" y el Tipo de Hardware a utilizar por el mismo. A la derecha de la pestaña se configuran los parámetros de ejecución del canal seleccionado.		
SC 3 Configurar canal de comunicación WIFI			
Escenario 1.1 Elegir y configurar el canal de comunicación WIFI.			
Descripción	Se intenta elegir y configurar el canal de comunicación WIFI.		
Tipo	V	WIFI	
Hardware	V	ESP8266AT	
SSID	V	nombre	
Contraseña	V	contraseña	
Respuesta del sistema	Se configuran los parámetros de ejecución definidos para el canal de comunicación WIFI.		
Flujo central	Desde el ambiente de edición se selecciona la pestaña "Canal", a la izquierda de esta pestaña se selecciona el Tipo de canal "Wifi" y el Tipo de Hardware a utilizar por el mismo. A la derecha de la pestaña se configuran los parámetros de ejecución del canal seleccionado.		
Escenario 1.2 Elegir y configurar el canal de comunicación WIFI / Capos vacíos.			
Descripción	Se intenta elegir y configurar el canal de comunicación WIFI.		
Tipo	V	WIFI	
Hardware	V	ESP8266AT	
SSID	I	(Campo vacío)	
Password	I	(Campo vacío)	
Respuesta del sistema	Muestra los campos vacíos resaltados en rojo y con un <i>ToolTip</i> que explica el error en cada uno.		
Flujo central	Desde el ambiente de edición se selecciona la pestaña "Canal", a la izquierda de esta pestaña se selecciona el Tipo de canal "Wifi" y el Tipo de Hardware a utilizar por el mismo. A la derecha de la pestaña se configuran los parámetros de ejecución del canal seleccionado.		

Anexo 25 Prueba de caja negra sobre la HU 16 Elegir modelo de placa Arduino y puerto donde está conectada.

Anexos

Escenario 1.1 Elegir modelo de placa Arduino y puerto donde está conectada.	
Descripción	Se intenta elegir el modelo de la placa Arduino y el puerto donde está conectada.
Variables	NA
Respuesta del sistema	Se actualiza en la pestaña "Pines" el listado de pines.
Flujo central	Desde el ambiente de edición en el menú "Herramientas" se selecciona la opción "Seleccionar placa" y se selecciona el modelo de la placa Arduino. Luego en el menú "Herramientas" se selecciona la opción "Seleccionar puerto" y se selecciona algún puerto donde está conectada alguna placa Arduino.