UNIVERSIDAD DE ORIENTE

Trabajo final presentado en opción al Título de Máster en Automática

Autor: Lic. Yoelvys Barriento López

Tutores: Dra. María E. Pardo Gómez

Dra. C. Ivón O. Benítez González

Consultante: Ing. Iván Rodríguez Pasín

Santiago de Cuba

2021

UNIVERSIDAD DE ORIENTE | **Facultad de Ingeniería Eléctrica**
Departamento de Ingeniería en Automática

Trabajo final presentado en opción al Título de Máster en Automática

Título: Sistema Informático de Programación de Autómatas Programables sobre tecnología Arduino

Autor: Lic. Yoelvys Barriento López
Tutores: Dra. María E. Pardo Gómez
Dra. C. Ivón O. Benítez González

Consultante: Ing. Iván Rodríguez Pasín

Resumen

En el presente trabajo se propone un sistema informático para la programación de Controladores Lógicos Programables (PLC) sobre tecnología Arduino, el cual facilita considerablemente la programación cumpliendo con la norma IEC 61131-3. El diseño consta de tres elementos principales:

- Uso del Banco de Trabajo del GEB Automation versión Student.
- La traducción y optimización del código generado por GEB Automation para su compatibilidad con el Kernel.
- El Kernel del PLC donde se ejecutan las aplicaciones generadas desde el banco de trabajo en un formato especial.

El GEB Automation utiliza la variedad de los lenguajes que dispone la metodología IEC 61131-3 para desarrollar los programas que se van ejecutar en el PLC. El entorno de traducción, optimización e integración se implementó sobre C++ Builder 10.3 Versión Community, que es un ambiente de desarrollo para plataforma Windows. El Kernel del PLC se amplió con nuevas funciones y bloques funcionales, orientados a facilitar el control de los procesos tecnológicos y la comunicación con los softwares de Supervisión, Control y Adquisición de Datos (SCADA) cumpliendo con la norma IEC 61131-5.

De esta forma, se obtuvo un entorno de programación el cual se independiza del hardware, logrando gran compatibilidad y permitiendo la edición ágil de los programas.

Abstract

In this work, a computer system is proposed for programming Programmable Logic Controllers (PLC) on Arduino technology, which considerably facilitates programming, complying with the IEC 61131-3 standard. The design consists of three main elements:

- Use of the GEB Automation Student version Workbench.
- The translation and optimization of the code generated by GEB Automation for its compatibility with the kernel.
- The Kernel of the PLC where the applications generated from the workbench are executed in a special format.

GEB Automation uses the variety of languages provided by the IEC 61131-3 methodology to develop the programs to be executed in the PLC. The translation, optimization and integration environment was implemented on C ++ Builder 10.3 Version Community, which is a development environment for Windows platform. The PLC Kernel was expanded with new functions and functional blocks, aimed at facilitating the control of technological processes and communication with the Supervision, Control and Data Acquisition (SCADA) software complying with the IEC 61131-5 standard.

In this way, a programming environment was obtained which is independent of the hardware, achieving great compatibility and allowing agile editing of programs.

Índice

Índice de Ilustraciones

Índice de Tablas

INTRODUCCIÓN

El PLC ha supuesto una gran revolución en la automatización industrial. Estos aparatos electrónicos, debido a su facilidad de programación e incremento de eficiencia, han terminado por ser clave en la modernización de las empresas en varias esferas de producción. Los autómatas programables han ido sustituyendo desde los años 60, los antiguos sistemas de control basados en circuitos eléctricos, relés, interruptores y otros componentes eléctricos.

Existen en el mundo importantes fabricantes de PLC: Rockwell Automation, Mitsubishi Electric, Siemens Industry, Perceptron Inc, General Electric, Scheneider Electric, entre otros[1]. Todos ellos suministran la aplicación que permite programarlos utilizando uno o varios lenguajes de programación de PLC establecidos en la norma IEC 61131-3.[2]

La División de Automatización de la empresa SERCONI no se ha quedado atrás en estos adelantos, por ello en el año 1997 participó en el desarrollo del hardware de una nueva versión del PLC cubano NODOREM denominada NOVA en conjunto con el Instituto Central de Investigaciones Digitales, ICID, más adelante se fabricó una nueva versión del PLC NOVA, el Controlador Programable Compacto EROS-PLC, así como el diseño del autómata distribuido EROS-PLC-D en el año 2010, con módulos I/O distribuidos y uso del bus CAN para su comunicación con el procesador central, el cuál fue instalado en la empresa Ernesto Che Guevara de la Unión del Níquel en el 2016 y aún se encuentra trabajando hasta actualidad; pero su producción y generalización se ha visto detenida por la baja disponibilidad de moneda convertible para adquirir los componentes electrónicos para su fabricación.

Ante esta situación, se decidió desarrollar una versión mejorada de los Autómatas Compactos EROSPLC y NOVA, con tecnología Arduino sustentada, en el módulo Atmega 2560 PRO y otra más pequeña con Atmega 328 para, con ambos, cubrir la gama de complejidad, tamaño y necesidad de estos equipos, que experimenta la

industria actualmente en nuestro país, manteniendo los estándares de diseño y producción nacional, aplicando módulos de Arduino de bajo costo y de hardware libre, lo cual permite el uso de todo el software desarrollado para los módulos de Arduino (software libre Open Source)[3], manteniendo el número de I/O digitales en uno de los diseños (similar al EROS-PLC y NOVA) y disminuyendo el tiempo de ejecución del ciclo del autómata.

A su vez, redundaría en una disminución de los costos de producción de los PLC, con similares prestaciones y tecnología moderna, para cubrir una amplia gama de necesidades de automatización para las cuales los autómatas que se importan estarían en muchos casos sobredimensionados.

Sin embargo, ante la mejora de los dispositivos PLC cubanos, se presenta la problemática de limitación de software de programación, teniendo en cuenta que los Arduinos se programan con el lenguaje C o C++ en el entorno de desarrollo integrado (IDE) de Arduino. Aunque existen paquetes de programación alternativos, la mayoría de ellos se alejan de la norma IEC 61131-3, lo cual dificulta que técnicos de planta puedan brindar soporte adecuado a los algoritmos de control de los procesos tecnológicos.

Se determinó usar el IDE GEB Automation[4] en su versión Student de libre acceso que da soporte a la tecnología Arduino y es usado para la programación de PLCs atendiendo a las normas estándares IEC 61131-3, su arquitectura está fundamentada sobre las bases de los Sistemas de Automatización Abierta[5] (*Open Automation*) y está compuesto básicamente por los siguientes elementos:

- Banco de Desarrollo de Aplicaciones (Editor).
- Kernel de los dispositivos ya portados en el IDE.
- Simulación de programa avanzada
- Traductor en código C de las aplicaciones.

El GEB Automation posee un Kernel en su versión Student limitado al controlador ATMEGA 2560 de tecnología Arduino[6], por lo que se hace necesario el desarrollo de un Kernel que pueda soportar los modelos de PLC con tecnología Arduino de desarrollo nacional.

El kernel (o núcleo) es la parte central de un sistema operativo y es el que se encarga de realizar toda la comunicación segura entre el software y el hardware. El desarrollo del Kernel torna el proceso de diseño en una tarea no trivial dado que se carece de:

- Códigos fuentes y documentación de la máquina virtual.
- Algoritmo interno de las funciones y los bloques funcionales.
- Bloques funcionales para los servicios de comunicación con el SCADA.
- Funciones y bloques funcionales para realizar control de procesos.

Luego, una vez desarrollados los algoritmos de control de procesos sobre GEB Automation, disponiendo de un sistema informático que posibilite traducir y optimizar el código C generado desde el GEB, y descargar la aplicación hacia el PLC vía USB, ISP ó UART; los algoritmos pueden ser ejecutados en el PLC por el Kernel desarrollado que es parte de dicho sistema informático.

A partir de lo antes señalado, se evidencia la necesidad de desarrollar una investigación, la cual considera como **problema**, la carencia de un sistema informático que posibilite desarrollar aplicaciones de control de procesos tecnológicos para PLC con tecnología Arduino, usando los lenguajes del estándar IEC 61131-3.

Se plantea como **objeto de la investigación** los sistemas informáticos para la programación de los PLC con tecnología Arduino, definiendo como **campo de acción** el banco de trabajo GEB Automation.

El **objetivo de la investigación** consiste en desarrollar un sistema informático para la programación de PLC con tecnología Arduino.

Para ello se plantea la siguiente **hipótesis:**

Si se cuenta con un sistema informático para la programación de los PLCs producidos nacionalmente sobre tecnología Arduino, se contará con el soporte adecuado para el desarrollo de los algoritmos de control de los procesos tecnológicos.

Para el cumplimiento del objetivo propuesto se han asumido las siguientes **tareas de investigación**:

1. Caracterización desde el punto de vista gnoseológico, histórico y actual los IDE de programación para tecnología Arduino.
2. Describir y caracterizar los elementos a tener en cuenta para desarrollar aplicaciones de control usando los 5 lenguajes del estándar IEC 61131-3 sobre GEB Automation.
3. Desarrollar conjunto de bibliotecas en lenguaje “C” para su ejecución sobre el hardware del PLC que garantice la ejecución de las aplicaciones de control desarrolladas empleando los lenguajes del estándar IEC 61131-3.
4. Desarrollar el Kernel para el funcionamiento interno del PLC.
5. Diseñar una aplicación que se ejecute sobre la PC para la traducción de los códigos C generados con el fin de optimizarlos y adaptarlos a la arquitectura del kernel.
6. Generar el Mapa Modbus para la importación de las variables del proceso desde el SCADA.

Estas tareas han sido desarrolladas utilizando como base las siguientes **técnicas y métodos de investigación**:

1. Análisis de documentos. Para revisión y estudio exhaustivo de la documentación que acompaña al producto GEB Automation.

2. Método histórico-lógico. Para realizar un análisis histórico sobre la evolución de los IDE de programación para tecnología Arduino.

3. Método de análisis-síntesis. A partir de lo reportado en la literatura y de las experiencias previas en el desarrollo de la Programadora de Autómata EROS-PG, se diseñó la estructura del sistema informático. Este método también fue válido, durante el ciclo de desarrollo del Sistema.

Aporte de la investigación:

La empresa podrá contar con una herramienta novedosa en Cuba para la programación de PLC con tecnología Arduino con un mínimo de coste y con el know-how para posteriores desarrollos.

Significación práctica de la investigación:

Al diseñar e implementar el sistema automatizado, permitirá desarrollar aplicaciones distribuidas y con conexión a SCADA usando el subsistema de comunicación incorporado al kernel del PLC, y de las funciones y bloques funcionales C de comunicación en concordancia con la especificación IEC 61131-3

Organización del Trabajo:

Este trabajo se encuentra organizado de la siguiente forma: una introducción general, dos capítulos con sus introducciones y conclusiones parciales, conclusiones generales, recomendaciones, bibliografías y anexos.

En el Capítulo 1, se aborda el Marco Teórico de la Investigación. Se conceptualizan los PLC, las normativas por las que se rigen y lenguajes de programación utilizadas a nivel mundial en la rama. Se realiza caracterización de las plataformas Arduino y se comparan las diferentes aplicaciones de programación de la misma. Además, se

fundamentan las herramientas, las tecnologías y la metodología a utilizar para desarrollar el software.

En el Capítulo 2, se realiza el análisis y diseño del Sistema Informático de Programación de Autómatas Programables sobre tecnología Arduino. Basado en los conceptos y técnicas expuestos en el Capítulo 1, se exponen los resultados de las etapas fundamentales del desarrollo del software, las funcionalidades que brinda esta herramienta, así como las ventajas y posibilidades que esta ofrece para su uso. Se realiza un análisis valorativo de los aspectos económicos, sociales y medioambientales relacionados con la implementación de software.

Aporte Práctico de la Investigación: Obtención de un sistema informático con una interfaz moderna, amigable, que facilita la programación de los autómatas basados en tecnología Arduino usando 4 de los lenguajes de la norma IEC61131-3.

Capítulo 1. SISTEMAS INFORMÁTICOS PARA LA PROGRAMACIÓN DE LOS PLC

INTRODUCCIÓN

En este capítulo se analizan los conceptos fundamentales referidos a los PLC y sus lenguajes de programación, se caracterizan las herramientas para el desarrollo de aplicaciones de control de procesos tecnológicos sobre tecnología Arduino. Se aborda brevemente, la metodología general a aplicar para el desarrollo del software y finalmente, se profundiza en el Entorno de Programación GEB Automation.

1.1. MARCO TEMÁTICO CONTROLADORES LÓGICOS PROGRAMABLES (PLC)

1.1.1. DEFINICIÓN Y CARACTERÍSTICAS PRINCIPALES

Un Controlador Lógico Programable (en adelante PLC, llamado así por sus siglas en inglés), o Autómata Programable, es un sistema electrónico programable diseñado para controlar diversos tipos de máquinas o procesos en tiempo real en un entorno industrial. Está compuesto por un microprocesador o microcontrolador, memorias, interfaces de entradas/salidas y circuitos adicionales. Puede programarse utilizando distintos lenguajes, tanto textuales como gráficos [7].

1.1.2. PROGRAMACIÓN DE UN PLC

Para programar un PLC, se debe comunicar al usuario con el Hardware PLC, mediante un dispositivo que permita escribir y poner a punto el programa que se ejecutará. Este dispositivo se conoce como “consola de programación”, que puede ser un dispositivo con pantalla diseñado específicamente para tal fin, o bien, una computadora de propósito general, corriendo un software con las mismas habilidades (más común en la actualidad).

1.1.3. FUNCIONES DE UN PLC

Un PLC es capaz de:

1. Adquirir datos del proceso por medio de sensores conectados eléctricamente a sus entradas digitales y analógicas.
2. Almacenar datos en memoria.
3. Tomar decisiones mediante un programa en memoria ingresado por el usuario que consiste de instrucciones las cuales implementan funciones específicas tales como lógicas, secuenciales, temporización, contadores y matemáticas.
4. Actuar sobre el proceso mediante sus salidas digitales y analógicas conectadas eléctricamente a actuadores.
5. Comunicarse con otros sistemas externos.

1.1.4. APLICACIONES DE LOS PLC

El PLC es el equipo más utilizado en la automatización industrial y domótica. Es por esto, que en el desarrollo de la práctica de su profesión un Ingeniero en Automatización y Control Industrial, utiliza estos equipos con mucha frecuencia. Existen seis áreas generales de aplicación de PLC:

1. Control secuencial.
2. Control de movimiento.
3. Control de procesos.
4. Monitoreo y supervisión de procesos.
5. Administración de datos.
6. Comunicaciones.

1.1.5. CONTEXTO HISTÓRICO

En 1968 GM Hydramatic (la división de transmisiones automáticas de General Motors) emitió una solicitud de propuestas para un reemplazo electrónico de los sistemas cableados de relés. La propuesta ganadora vino de Bedford Associates[8]. El resultado fue el primer PLC, designado 084 porque era el proyecto de Bedford Associates nº 84. Una de las personas que trabajaron en ese proyecto fue Dick Morley, quien es considerado como el «padre» del PLC[9].

1.1.6. ESTADO ACTUAL DE LOS PLC

Existen muchos fabricantes de PLC en el mercado, desde grandes empresas multinacionales, hasta pequeños desarrollos.

Los entornos de programación de PLC han evolucionado ampliamente desde sus comienzos hasta hoy en día. En el sitio web[10], pueden observarse diferentes entornos de programación de PLC.

En la actualidad, las empresas líderes en el mercado tienden a intentar resolver el problema completo de automatización; esto es, en un mismo entorno proveen programación y simulación tanto del software que contiene el PLC, como de sus módulos de entradas, salidas, de comunicación, conexiones de red entre los dispositivos y Paneles para HMI/SCADA. Así, se pueden resolver muchos problemas de automatización utilizando una única herramienta de software, reduciendo los tiempos de desarrollo y entrenamiento del personal encargado de utilizarlos, sin embargo, debido a que cada fabricante puede tener procesos y objetivos diferentes, la unificación y estandarización de los lenguajes de comunicación en todos los PLC es una tarea que difícilmente podía ser completada, puesto que cada fabricante tenía herramientas y protocolos que solo trabajaban bajo sus estándares y dispositivos, por lo cual en la actualidad hay 5 lenguajes estandarizados de PLC a nivel mundial, los cuales difieren grandemente entre sí.

Es por esto que el software libre también tocó a las puertas de la automatización, la aparición de tecnologías embebidas como arduino[11, 12] en el año 2005 y Raspberry en el 2009, ha cimentado el comienzo de una revolución en el campo de la automatización, no solo en procesos académicos y de aprendizaje, sino también en campos de aplicación en diferentes sectores de la economía. Sin embargo, estos dispositivos embebidos por sí solos pueden llegar a ser muy inestables en un entorno industrial debido al gran ruido al que se exponen los equipos en un ambiente con grandes motores y máquinas que demandan gran cantidad de energía. Es esta la razón por la que empresas como KUNBUS, Industrino, Controlino, Industrial Shields y otras en el mercado, han decidido dar un paso adelante diseñando y construyendo dispositivos de control que prometen fiabilidad para manejar maquinaria de carácter industrial, con placas programables bajo la flexibilidad y economía del software libre.

1.2. NORMA IEC 61131 - CONTROLADORES PROGRAMABLES

Dado que hoy en día hay una gran variedad de fabricantes y cada uno ha definido su propio lenguaje de programación de PLC, se creó un estándar internacional con el ánimo de normalizar todo lo relacionado con esta tecnología, la norma IEC 61131[13], que es el resultado del gran esfuerzo realizado por siete multinacionales, con muchos años de experiencia en el campo de la automatización industrial; logrando el primer paso en la estandarización de los controladores programables y sus periféricos, incluyendo los lenguajes de programación que se deben utilizar.

La finalidad de esta norma es:

1. Definir e identificar las características principales que se refieren a la selección y aplicación de los PLC y sus periféricos.
2. Especificar los requisitos mínimos para las características funcionales, las condiciones de servicio, los aspectos constructivos, la seguridad general y los ensayos aplicables a los PLC y sus periféricos.
3. Definir los lenguajes de programación de uso más corriente, las reglas sintácticas y semánticas, el juego de instrucciones fundamentales, los ensayos y los medios de ampliación y adaptación de los equipos.
4. Dar a los usuarios una información de carácter general y unas directrices de aplicación.
5. Definir las comunicaciones entre los PLC y otros sistemas.

Esta norma, está formada por las siguientes partes:

1. IEC 61131-1. Información general.
2. IEC 61131-2. Especificaciones y ensayos de los equipos.
3. IEC 61131-3. Lenguajes de programación.
4. IEC 61131-4. Guías de usuario.
5. IEC 61131-5. Comunicaciones.
6. IEC 61131-6. Reservada.
7. IEC 61131-7. Fuzzy Control.
8. IEC 61131-8 Guías de programación

1.3. IEC 61131-3 LENGUAJES DE PROGRAMACIÓN.

En la presente Tesis se utiliza la segunda edición de esta parte de la norma, publicada en el año 2003 (IEC 61131-3:2003). La misma especifica los modelos de software, comunicación, programación y cinco lenguajes de programación utilizados para la organización de programas confeccionados mediante dichos lenguajes. En las siguientes secciones se realiza un resumen de estas características[2].

1.3.1. MODELO DE SOFTWARE

El modelo de Software definido en la norma IEC 61131-3 se expone en la figura 1.1.

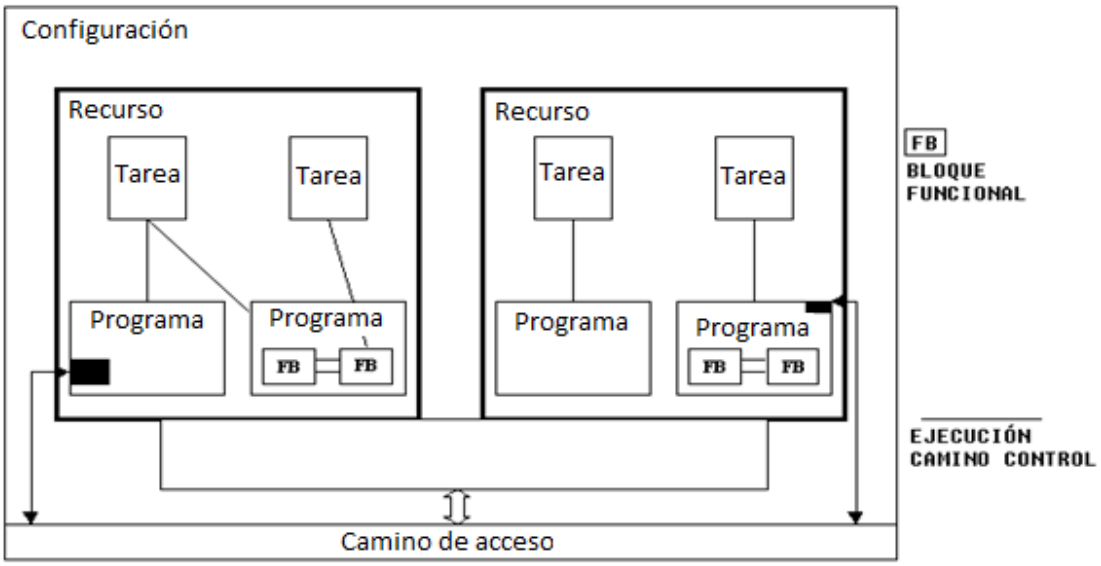

Figura 1.1: Modelo de Software de PLC según norma IEC 61131-3 [14]

Al más alto nivel, el elemento software requerido para solucionar un problema de control particular puede ser formulado como una *configuración*. Una configuración es específica para un tipo de sistema de control, incluyendo las características del hardware: procesadores, direccionamiento de la memoria para los canales de I/O y otras capacidades del sistema.

Dentro de una configuración, se pueden definir uno o más *recursos*. Se puede entender el recurso como un procesador capaz de ejecutar programas IEC. Con un recurso, pueden estar definidas una o más *tareas*. Las tareas controlan la ejecución de un conjunto de programas y/o bloques de función. Cada una de ellos puede ser ejecutado periódicamente o por una señal de disparo especificada, como el cambio de estado de una variable.

Los *programas* están diseñados a partir de un diferente número de elementos de software, escrito en algunos de los distintos lenguajes definidos en IEC 61131-3. Típicamente, un programa es una interacción de *Funciones* y *Bloques Funcionales,* con capacidad para intercambiar datos. Funciones y bloques funcionales son las partes básicas de construcción de un programa, que contienen una declaración de datos y variables y un conjunto de instrucciones.

Comparado esto con un PLC convencional o basado en tecnología Arduino, éste contiene un solo recurso, ejecutando una tarea que controla un único programa de manera cíclica. IEC 61131-3 incluye la posibilidad de disponer de estructuras más complejas como multi-procesamiento y gestión de programas por eventos tales como las características de los sistemas distribuidos o los sistemas de control de tiempo real. IEC 61131-3 está disponible para un amplio rango de aplicaciones, sin tener que conocer otros lenguajes de programación adicionales.

1.3.2. MODELO DE COMUNICACIÓN

Define brevemente cómo se comunican dos configuraciones, mediante Caminos de Acceso, o bien, entre Programas o Bloques de Función. La mayoría de los protocolos actuales derivan de comunicaciones de serie heredadas. Si recordamos el estándar de comunicaciones 485, cuando se conecta un grupo de dispositivos a un cable serie, todos los dispositivos reciben el mismo mensaje y todos los dispositivos deciden si aceptarlo o no.

El protocolo Modbus del cual se hará uso en la presente Tesis es el más popular, alrededor del 90% de PLC usan este protocolo. Es un protocolo abierto, probablemente esta sea la razón por la que todos los PLC admiten Modbus, porque

no requiere de ninguna licencia. Modbus puede certificar un dispositivo para asegurar que dicho dispositivo realmente cumple con todos los estándares, pero no tiene por qué hacerlo realmente, no hay licencias.[15]

1.3.3. MODELO DE PROGRAMACIÓN

El modelo de programación define:

- Tipos de datos
- Literales
- Variables
- Categorías de variables
- Unidades de Organización de Programa

Las Unidades de Organización de Programa, (POU), están formadas por distintas categorías de Declaraciones de Variables, y un cuerpo de programa confeccionado en alguno de los cuatro lenguajes propuestos por la norma. Existen 3 tipos de POU:

- Funciones: IEC 61131-3 especifica funciones estándar y funciones definidas por el usuario (o funciones derivadas); las primeras son, por ejemplo, ADD (suma), ABS (valor absoluto), SQRT (raíz cuadrada), SIN (seno), y COS (coseno), y las restantes, una vez implementadas por el usuario, pueden ser utilizadas dentro de cualquier POU. Las funciones no pueden contener ninguna información de estado interno, es decir, que la invocación de una función con los mismos argumentos (parámetros de entrada) debe suministrar siempre el mismo valor (salida).
- Bloques de Función: Representan funciones de control especializadas. Contienen, datos, instrucciones, y, además, pueden guardar los valores de las variables. Debido a esto, es un bloque altamente reutilizable, ya que permite definir instancias del mismo independientes entre sí, las cuales contienen la información de su estado interno. De la misma forma que en el caso de las Funciones, existen Bloques de Función estándar (por ejemplo, biestables, detección de flancos, contadores, temporizadores, etc.) y el usuario puede definir sus propios. En su interior puede contener llamados a Funciones.

- Programas: Se define como un conjunto lógico de todos los elementos y construcciones del lenguaje de programación, que se requiere para el control de una máquina o proceso mediante el sistema PLC. Un programa puede contener en su interior llamados a Funciones e instancias de Bloques de Función.

El kernel a usar está compuesto por un paquete de bibliotecas desarrolladas en C, que manejan el funcionamiento del PLC y sus subsistemas, que contienen la implementación del conjunto de operadores, funciones y FBs disponibles de acuerdo a las especificaciones IEC 61131-3 e IEC 61131-5.

1.3.4. TIPOS DE LENGUAJES DE PROGRAMACIÓN PARA PLC

Los lenguajes de programación de PLC son símbolos, caracteres y reglas de uso que fueron diseñados para poder tener una comunicación de los usuarios con las máquinas. Gracias a este vínculo, podemos ser capaces de crear un programa con instrucciones para controlar el funcionamiento de cualquier proceso o máquina[16].

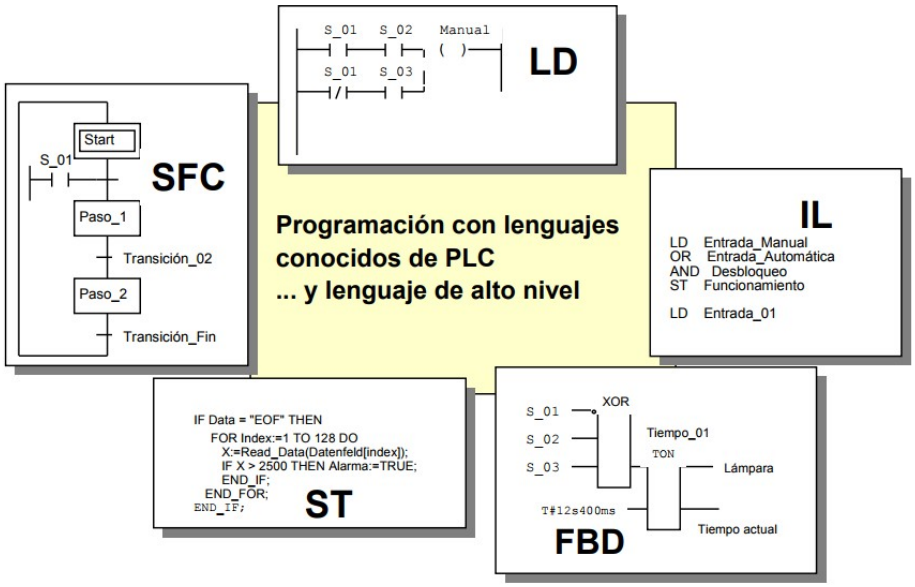

Figura 1.2: Lenguajes de programación PLC[17]

Este tipo de lenguajes, enmarcan diferentes formas graficas en el programa como se verán en resumen en los siguientes apartados.

1.3.4.1. LENGUAJES GRÁFICOS

1.3.4.1.1. Diagrama de escalera (LD)

El Diagrama de escalera, consiste en una recopilación de códigos con información simbólica, estos identifican una instrucción propia del programa que se está ejecutando.

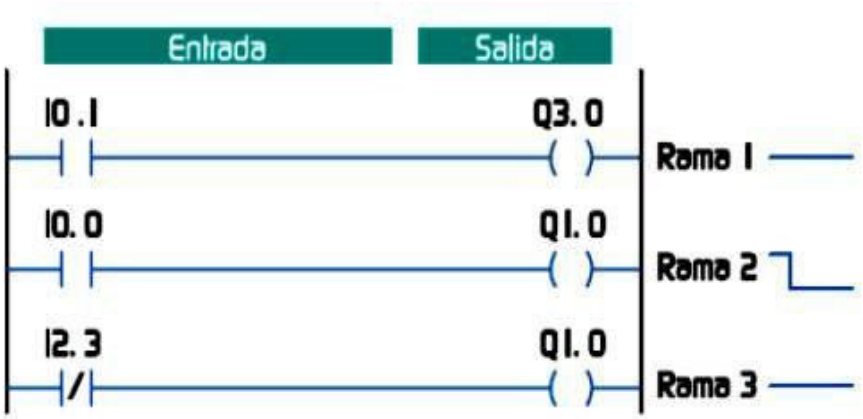

Figura 1.3: Lenguaje de escalera [17]

Este lenguaje se le apoda el nombre de diagrama de contactos, sin embargo, más conocido como escalera, es una recopilación de circuitos conectados en paralelo a través de una línea de corriente directa (cd) o de corriente alterna (ac).

1.3.4.1.2. Diagrama de bloques funcionales (FBD)

El diagrama de bloques funcionales, permite una integración mucho más familiar en el aspecto gráfico donde las características funcionales del programa, se representan mediante bloques lógicos o aritméticos. La programación específica de este lenguaje, permite el procesamiento de datos en paralelo y los diagramas de bloques reemplazan de manera absoluta a la lógica escalera en los PLC más sofisticados.

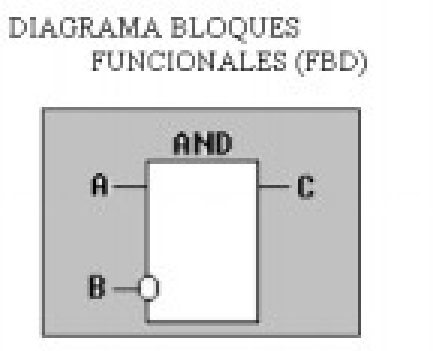

Figura 1.4: Diagrama de bloques funcionales [17]

Otra característica de los (FBD), es su identidad frente a las funciones que posee para ganar potencia en el uso de detección de flancos, contadores, temporizadores, entre otros. Es capaz de ejecutar varias copias como se desee en un mismo bloque funcional; a cada copia se le denomina instancia y lleva asociado un término con estructura de datos que contiene sus variables de entrada, de salida e internas a diferencia del resto de las instancias.

1.3.4.1.3. Grafcet (SFC)

El lenguaje de programación GRAFCET, simplifica la complejidad de extensas líneas de código con la nueva concepción algorítmica de que todo proceso lleva a cabo en una secuencia. Actualmente, este tipo de lenguaje es el más usado por su mayor atención a la necesidad industrial y su simplicidad en un diagrama funcional normalizado. Esto permite realizar un modelo del proceso a automatizar, teniendo como pieza clave las entradas, acciones a realizar y los procesos que actúan sobre las acciones.

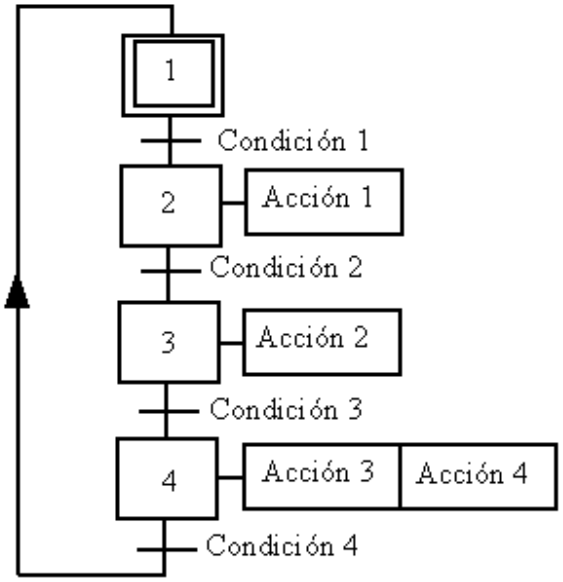

Figura 1.5: Diagrama de bloques funcionales [17]

Como se indicó anteriormente, la estructura SFC se integra de varias etapas nombradas simbólicamente por cajas rectangulares conectadas entre ella, a través de líneas verticales. Cada transición es una condición de valor que se encuentra diferenciada por (true/false), lo cual permite tomar decisiones como apagar o prender.

1.3.4.2. LENGUAJES LITERALES

1.3.4.2.1. Lista de instrucciones (IL)

Se trata de un lenguaje muy parecido a un assembler o ensamblador de bajo nivel, por lo que sus instrucciones son bastante básicas, lo cual lo hace relativamente difícil y engorroso de aplicar. Es el más próximo al verdadero lenguaje de máquina que emplea el PLC durante su operación, por lo que el examen de un programa realizado con este lenguaje proporciona la mejor visión de cómo es realmente la secuencia de operaciones e instrucciones que va ejecutando el autómata.

1.3.4.2.2. Texto estructurado (ST)

Se trata de un lenguaje textual de alto nivel que se presta muy bien para la programación estructurada. Su sintaxis es muy parecida a la de Pascal o C y soporta

una amplia gama de funciones y operadores estandarizados. Lamentablemente, son pocos los autómatas que admiten en la actualidad la programación realizada con este lenguaje. [2]

Dentro de sus beneficios, el texto estructurado se basa en la formulación de tareas de dicho programa. Aquí intervienen programas elaborados con reglas de instrucciones concisas que permiten realizar secuencias de ciclos como: If, while, for, case, etc.

1.4. Protocolo de Comunicaciones MODBUS

Modbus es un protocolo de comunicaciones serie situado en el nivel 7 del Modelo OSI (*Open System Interconnection*), el cual fue desarrollado y publicado por Modicon (ahora conocido como Schneider Electric) para su gama de controladores lógicos programables, también llamados PLCs, en 1979.

Analizando el mercado industrial nos damos cuenta de que, actualmente, el protocolo Modbus es el protocolo de comunicaciones más instaurado en los dispositivos electrónicos industriales, sistemas de telecontrol y monitorización, lo que implica que, tanto a nivel local como a nivel de red, en su versión TCP/IP, seguirá siendo uno de los protocolos de referencias en las llamadas Smart Grids (*Redes inteligentes*), telecontrol, redes de sensores, y un extenso etcétera de sistemas de información que ya empiezan a asomar la cabeza en nuestro día a día.

El objetivo del protocolo Modbus consiste básicamente en la transmisión de información entre distintos equipos electrónicos conectados a su mismo bus. Existiendo en dicho bus un solo dispositivo maestro (Master) y varios equipos esclavos (Slaves) conectados[18].

En sus inicios estaba orientado a una conectividad a través de líneas serie como pueden ser RS-232 o RS-485, pero a lo largo del tiempo han aparecido variantes como es el Modbus TCP, el cual permite el encapsulamiento del Modbus serie en tramas Ethernet TCP/IP de forma sencilla. Esto sucede porque desde un punto de vista de la torre OSI, el protocolo Modbus se ubica en la capa de aplicación.

Las razones por las cuales el uso de Modbus destaca con respecto a otros protocolos de comunicaciones, y así haya conseguido ser el más estandarizado en el sector industrial, se debe a:

- Compatibilidad con dispositivos e instalaciones de vieja construcción: Su funcionalidad es tal que es capaz de funcionar sobre prácticamente la mayoría de medios de comunicación que incluyan cables de par trenzado, fibra óptica, Ethernet, o incluso conectividad inalámbrica como teléfonos celulares y microondas.
- Su implementación es simple: Requiere poco desarrollo debido a que, a diferencia de otros protocolos, se simplifica la estructura de las tramas y en consecuencia el acceso a los datos que no están almacenados en estructuras complejas.
- Es un estándar público y seguro: Esto resulta interesante para los fabricantes, ya que permite que estos puedan desarrollar dispositivos tanto Maestro como Esclavo sin gastos aplicados al protocolo. Este hecho facilita el acceso a la información y estructura del protocolo que, además, es muy básica pero funcional para su objetivo.
- Flexibilidad en el intercambio de información: Maneja bloques de datos sin suponer restricciones, es decir, la transmisión de información no está ligada a algún tipo de datos en concreto. Un ejemplo claro para entender el concepto: si se transmite un dato de 16 bits de información su representación no está sujeta a ninguna restricción, por lo que puede tratarse de un dato tipo Word con signo, un entero sin signo de 16 bits o la parte alta de una representación tipo Float de 32bits, etc. La representación del valor vendrá definida por la especificación que el fabricante dé del dispositivo, lo que permite la representación de un amplio rango de valores.

1.4.1. FUNCIONAMIENTO

El protocolo Modbus siempre funciona con un Maestro y uno o más Esclavos, siendo el maestro quién controla en todo momento el inicio de la comunicación con los esclavos, solicitando información del resto de dispositivos conectados que ejercen

como esclavos y son quienes suministran la información al primero. El esclavo por otro lado se limita a devolver los datos solicitados por el maestro.

Según la especificación, en una misma red puede haber hasta 247 dispositivos esclavos, esto es debido a que en una trama Modbus la dirección del esclavo se representa con un solo Byte, existiendo algunas direcciones reservadas para propósitos específicos como Broadcast.

0	From 1 to 247	From 248 to 255
Broadcast address	Slave individual addresses	Reserved

El nodo maestro no tiene ninguna dirección asignada, únicamente los nodos esclavos deben tener una dirección única en un bus y, a su vez, deben reconocer la dirección 0, que es la reservada como la dirección de difusión.
Las peticiones del nodo maestro a los nodos esclavos pueden ser:

- Modo Unicast: el maestro se dirige a un esclavo concreto e individual y éste, tras recibir y procesar la petición, devuelve una respuesta al maestro.

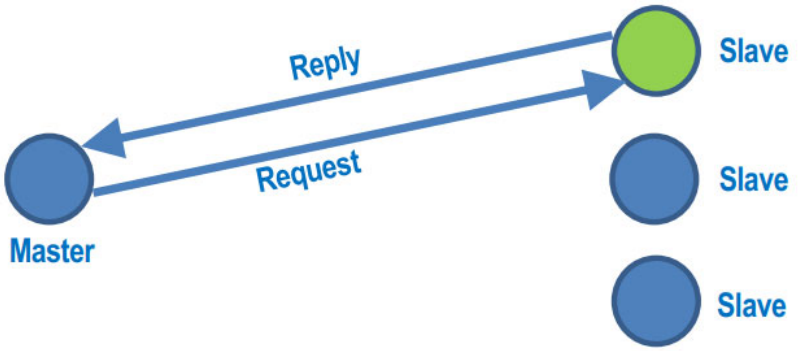

Figura 1.6: Modo Unicast

- Modo Broadcast: el maestro se dirige a todos los esclavos que se encuentren en el mismo bus. La única restricción que tiene es que las peticiones de difusión solo pueden ser de escritura.

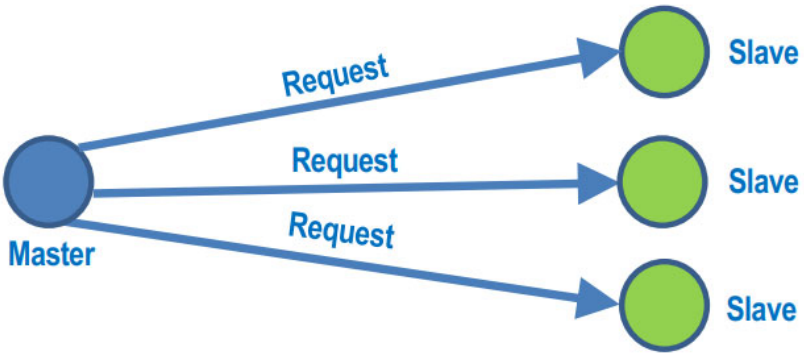

Figura 1.7: Modo Broadcast

En definitiva, su funcionamiento se resume básicamente en: el Maestro pregunta y los Esclavos responden o actúan en función de lo que este ordene.

1.4.2. TIPOS DE OBJETOS

En el protocolo Modbus se distingue entre entradas digitales (discrete input), salidas digitales (coils), registros de entrada (input register) y registros de retención (holding registers). Tanto las entradas como las salidas digitales tienen una longitud de un bit, mientras que los registros, tanto de entrada como de retención, tienen una longitud de dos Bytes.

En la siguiente tabla se recoge tanto el acceso como el tamaño según el tipo de objeto proporcionado por un esclavo:

Tabla 1.1: Tipos de objetos. [3]

Object type	Access	Size
Discrete input	Read-only	1 bit
Coil	Read-Write	1 bit
Input register	Read-only	16 bits
Holding register	Read-Write	16 bits

1.4.3. FORMATO DE LA TRAMA

Los modos de transmisión definen como se envían los paquetes de datos entre maestros y esclavos, y el protocolo Modbus define dos variantes, con diferentes

representaciones de los datos y de algunos detalles del protocolo ligeramente desiguales:

- Modbus RTU: La comunicación se realiza por medio de una representación binaria compacta de los datos. El formato RTU finaliza la trama con una suma de control de redundancia cíclica (CRC).
- Modbus ASCII: Es una representación legible del protocolo, pero menos eficiente. El formato ASCII utiliza una suma de control de redundancia longitudinal (LRC).
- Modbus TCP/IP: En realidad no es una variante más ya que es muy semejante al formato RTU, pero estableciendo la transmisión mediante paquetes TCP/IP a través del puerto 502.

Ambas implementaciones del protocolo, RTU y ASCII, son en serie.

La estructura de trama Modbus es muy simple, siendo uno de los motivos de su éxito junto a ser un protocolo abierto y a no estar orientado a conexión.

Por tanto, la estructura básica de una trama Modbus RTU, tanto de lectura como de escritura, es la que se muestra en la siguiente figura:

	Address	Function Code	DATA	Checksum
Modbus Frame	1 Byte	1 Byte	N Byte	2 Byte

Figura 1.8: Trama Modbus

- Address: Indica la dirección del esclavo que, al ser de 1 Byte, limita el número de esclavos que podemos tener conectados de forma correcta al bus serie Modbus como se indicó anteriormente en el punto 2.1.
- Function Code: En este campo se indica que acción requiere el maestro del esclavo al que va dirigida la trama. Este campo se particulariza dependiendo de si se trata de una trama maestro->esclavo o si por el contrario es esclavo->maestro.

El código de operación puede tomar cualquier valor comprendido entre el 0 y el 127 (el bit de más peso se reserva para indicar error). Cada código se corresponde con una determinada operación. Algunos de estos códigos se consideran estándar y son aceptados e interpretados por igual por todos los dispositivos que dicen ser compatibles con MODBUS, mientras que otros códigos son implementaciones propias de cada fabricante. Es decir que algunos fabricantes realizan implementaciones propias de estos códigos "no estándar".

La relación de funciones implementadas más usadas de este protocolo son las siguientes:

Tabla 1.2: Códigos de función Modbus

Function code	Function name
1	Read coils
2	Read Discrete Inputs
3	Read Multiple Holding Registers
4	Read Input Registers
5	Write Single Coil
6	Write Single Holding Registers
15	Write Multiple Coil
16	Write Multiple Holding Registers
23	Read/Write Multiple Registers

- **Data:** Este campo dependerá tanto en contenido como en longitud de la función que se indique en el campo anterior, así como de si se trata de una trama maestro-esclavo o de respuesta esclavo-maestro.
- **Checksum:** Este campo consta de dos Bytes y sirve para la detección de errores en la trama. El CRC es un código más que frecuente en la detección de errores en redes digitales, sistemas de almacenamiento para la detección

de modificación accidental de los datos o en este caso para comprobar la integridad de los datos en su transmisión por buses de campo.

1.4.4. VERSIONES DEL PROTOCOLO

En la actualidad, existen diversas versiones del protocolo Modbus, para el puerto serie, para el estándar de redes de área local como es Ethernet, e incluso otros protocolos que soportan el conjunto de protocolos TCP/IP de Internet. Esto es debido a que al igual que las redes de comunicaciones entre dispositivos electrónicos han ido evolucionando, han ido apareciendo variantes de este protocolo, en cuyos inicios estaba diseñado para redes sobre líneas serie. En la siguiente imagen recoge la perfecta armonía que existe entre las diferentes versiones, las cuales se explican seguidamente:

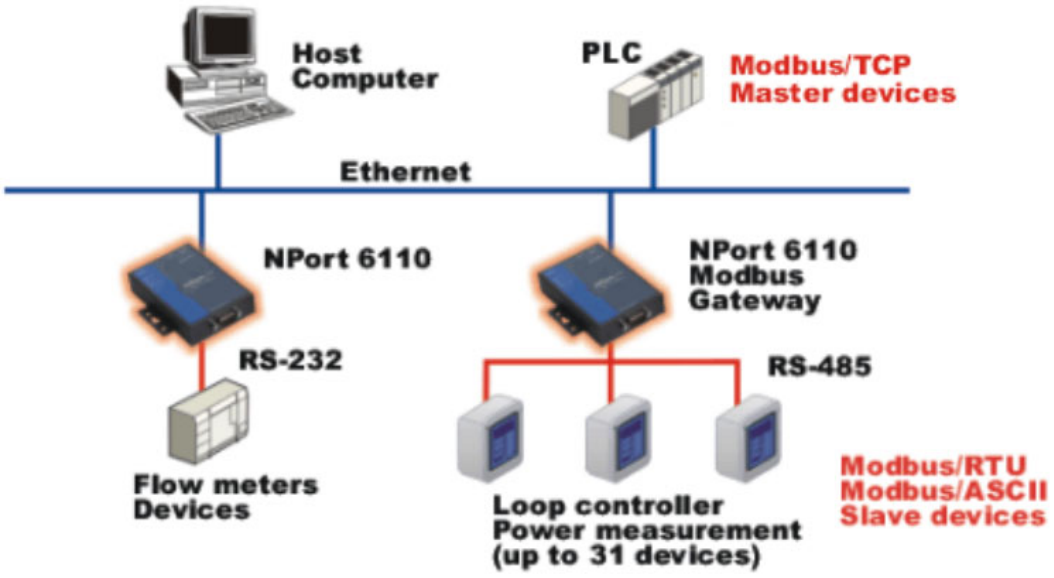

Figura 1.9: Escenario con las diferentes versiones del protocolo[19]

1.4.4.1. MODBUS RTU

Como se ha comentado al inicio del punto 2, Protocolo de comunicaciones Modbus, es la implementación más instaurada para Modbus. Se utiliza en la comunicación serie convencional, usando una representación binaria compacta de los datos. Los mensajes Modbus RTU deben transmitirse continuamente sin oscilaciones entre

caracteres. Permite la comunicación con 16 dispositivos esclavos por canal, a una velocidad de transferencia de hasta 19.2 Kbps.

1.4.4.2. MODBUS ASCII

Esta versión utiliza igualmente la comunicación serie convencional, pero con la particularidad de que hace uso de caracteres ASCII. Al igual que la versión anterior, Modbus RTU, permite la comunicación con 16 dispositivos esclavos por canal, a la misma velocidad de transferencia, hasta 19.2 Kbps.

1.4.4.3. MODBUS TCP/IP

Es la versión que se ha elegido para desarrollar este trabajo y por ello se va a entrar en más detalles sobre sus características y especificaciones [4].

Esta variante es la evolución más fuerte, utilizada y conocida, ya que permite la implementación de este protocolo de comunicaciones sobre redes Ethernet, lo que supone un incremento en cuanto al grado de conectividad. Como se ha comentado anteriormente, es muy parecido al formato RTU, pero en este caso la transmisión se establece a través de paquetes TCP/IP mediante el puerto 502. Esta versión encapsula la trama base del protocolo Modbus en la capa 7 del modelo OSI, capa de aplicación, TCP/IP de forma sencilla.

Por tanto, Modbus-TCP permite que se pueda usar en Internet, objetivo destacable por el que se inició su desarrollo, remitiéndose la especificación del protocolo a la IETF. De este modo, un dispositivo instalado en cualquier parte del mundo podría ser direccionado desde cualquier otra parte del mundo.

También proporciona ventajas muy interesantes tanto para la empresa en cuestión como para el instalador:

- Permite acceder remotamente utilizando un PC y así solucionar los problemas que se presenten, realizar reparaciones o incluso el mantenimiento de los mismos sin desplazamientos, lo que implica una mejora en el servicio con el cliente y una reducción palpable en los costes.

- Permite realizar la gestión de sistemas distribuidos geográficamente mediante el empleo de las tecnologías de Internet/Intranet actualmente disponibles.

Modbus TCP/IP se ha convertido en un estándar industrial debido a su simplicidad, reducido coste, requisitos hardware mínimos y por tratarse de un protocolo abierto, motivo por el cual hay un amplio catálogo de dispositivos Modbus TCP/IP en el mercado.
Gracias a la combinación de una red física versátil y escalable como el estándar universal de interredes TCP/IP y una representación de datos independientes de fabricante, proporciona una red y accesible para el intercambio de datos de proceso.

Esta versión del protocolo Modbus únicamente encapsula una trama Modbus en un segmente TCP, ya que este protocolo de transmisión proporciona un servicio orientado a conexión fiable, es decir, toda consulta espera una respuesta.

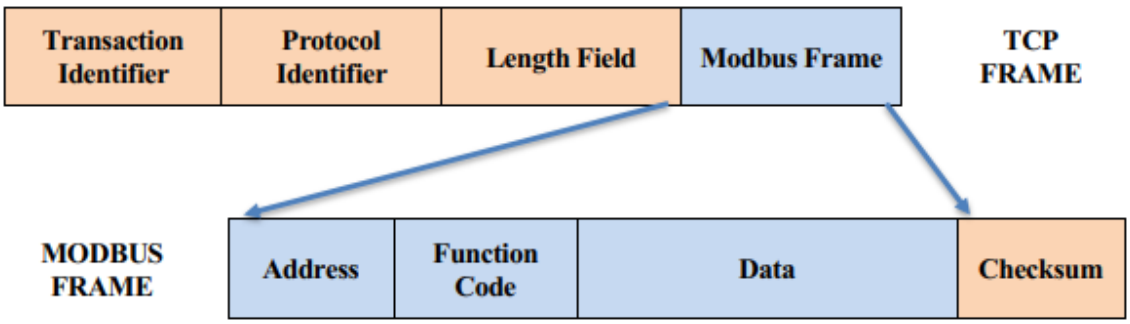

Figura 1.10: Encapsulamiento de la trama Modbus en TCP[19]

Esta técnica de consulta-respuesta encaja perfectamente con la naturaleza Maestro-Esclavo de Modbus, todo esto añadido a las ventajas que también proporcionan las redes Ethernet conmutadas a los usuarios en la industria. Modbus TCP/IP proporciona una solución para la gestión desde unos pocos a decenas de miles de nodos.

En cuanto a las prestaciones de un sistema Modbus TCP/IP depende básicamente de la red y el hardware usado. Si se usa sobre Internet, las prestaciones serán las correspondientes a tiempos de respuesta de Internet, por lo que pueden no ser las deseadas para un sistema de control, pero sí pueden ser suficientes para una

comunicación destinada a la depuración y mantenimiento, reduciendo así el desplazamiento al lugar de la instalación. Si en cambio se usa sobre una Intranet de altas prestaciones con conmutadores Ethernet de alta velocidad, la situación es totalmente diferente.

Poniendo un ejemplo teórico, Modbus TCP/IP transporta datos con una eficiencia alrededor del 60% cuando se transfieren registros en bloque, y poniendo que 10BaseT proporciona unos 1,25 Mbps de datos, la velocidad de transferencia de información útil será:

1,25M / 2 * 60% = 360000 registros por segundo

Tomando 100BaseT la velocidad se incrementa en 10 veces. Todo este ejemplo es suponiendo que se están empleando dispositivos que pueden dar servicio en la red Ethernet aprovechando todo el ancho de banda disponible.

Además, debido al abaratamiento de los ordenadores personales y el desarrollo de redes Ethernet cada vez más rápidas, permite elevar las velocidades de funcionamiento, a diferencia de otros buses que están inherentemente limitados a una sola velocidad.

La comunicación entre dispositivos sobre Modbus TCP/IP es muy sencilla ya que solo requiere una pasarela que convierta el protocolo Modbus a Modbus TCP/IP.

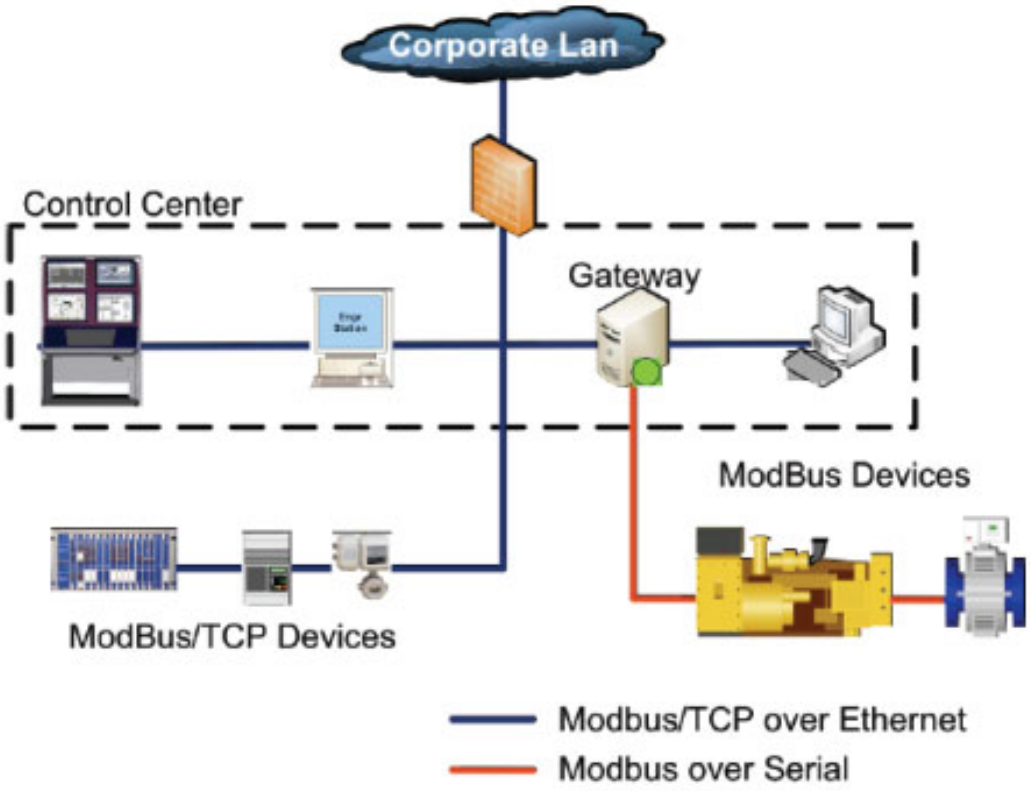

Figura 1.11: Escenario con las diferentes versiones del protocolo[19]

1.5. La Plataforma Arduino

Si bien es cierto existen muchas plataformas de desarrollo de *hardware* abierto, los microcontroladores conocidos como Arduinos se han convertido en el ambiente de prototipado más usado a nivel global[3] por su costo, su variada oferta de controladores y accesorios y su forma de programación basada en C con múltiples bibliotecas de código abierto, aunado a su capacidad de procesamiento. Por ejemplo, si se revisa la especificación técnica para la tarjeta Arduino MEGA[6], se observa que posee un procesador marca ATMEL modelo ATmega 2560[6] que es un procesador de 8 bits tipo RISC que procesa 16 millones de instrucciones por segundo (MIPS) a 16 MHz; una memoria de trabajo *flash* de 256K Bytes, 54 pines configurables como entradas o salidas, 16 entradas analógicas de 0 a 5 voltios con una resolución de 10 bits y frecuencia de muestreo de 10kHz, etc. Con estas especificaciones de procesamiento y capacidad para manejar entradas y salidas digitales, es válido plantear la automatización de pequeña escala con estos dispositivos, es decir, como

sustituto de controladores industriales tales como los relés inteligentes y los nano y micro PLC.

Si se revisan las especificaciones técnicas del micro PLC de última generación Siemens CPU 1212C [20], se ve que posee una memoria de trabajo de 75 K Bytes, procesa 10 MIPS y tiene 8 entradas y 6 salidas digitales, con 2 entradas analógicas, etc. Si se comparara solamente la velocidad de procesamiento, cantidad de memoria y cantidad de entradas y salidas, el Arduino Mega es una opción superior a este PLC particular. Sin embargo, estas no son las únicas variables que se deben analizar; el PLC tiene características a su favor, por ejemplo, las entradas están aisladas opto eléctricamente, posee salidas a relés o a transistor de 24 voltios, tiene carcasas plásticas con algún grado de protección IP, el *software* de programación utiliza lenguajes de programación estandarizados definidos en la norma IEC 61131-3, lo que facilita las modificaciones a los programas de automatización, tiene respaldo del fabricante, certificaciones eléctricas UL, CE, etc., modularidad y expansión, manejo de protocolos industriales, etc.

Dadas algunas debilidades que se le señalan al Arduino, han aparecido versiones de *hardware* robustas, pensadas para ambientes más hostiles, tal es el caso de Ruggeduino[21] que incorpora protección para todas las entradas, montaje para riel DIN; el PLC Arduino, que consiste de *shield* (tarjeta de expiación) para Arduino Uno, en el que las entradas son opto acopladas para señales de 24 voltios DC, salidas a relé, puertos de comunicación RS 485 y ethernet; y el Industrino[22], con carcasa para riel DIN, pensado para la automatización domótica.

1.5.1. ENTORNOS DE PROGRAMACIÓN PLATAFORMA ARDUINO

1.5.1.1. ARDUINO IDE

Arduino IDE [11], permite generar impactos positivos en la implementación de tableros con PLCs. Para programar un Arduino, el lenguaje estándar es C++, aunque es posible programarlo en otros lenguajes. No es un C++ puro sino que es una adaptación que proveniente de avr-libc[23] que provee de una librería de C de alta calidad para usar con GNU Compiler Collection (GCC) en los microcontroladores

AVR de Atmel y muchas funciones específicas para los MCU AVR de Atmel. Con las numerosas bibliotecas de Arduino, se podrá emplear cualquier tipo de proyecto seleccionando en el menú el tipo de placa a utilizar. Además, brinda ventajas de aplicación si se usa Windows, esto con el fin de integrar automáticamente la biblioteca para Arduino 1.8.10. Tiene como desventaja que no soporta la norma IEC 61131-3.

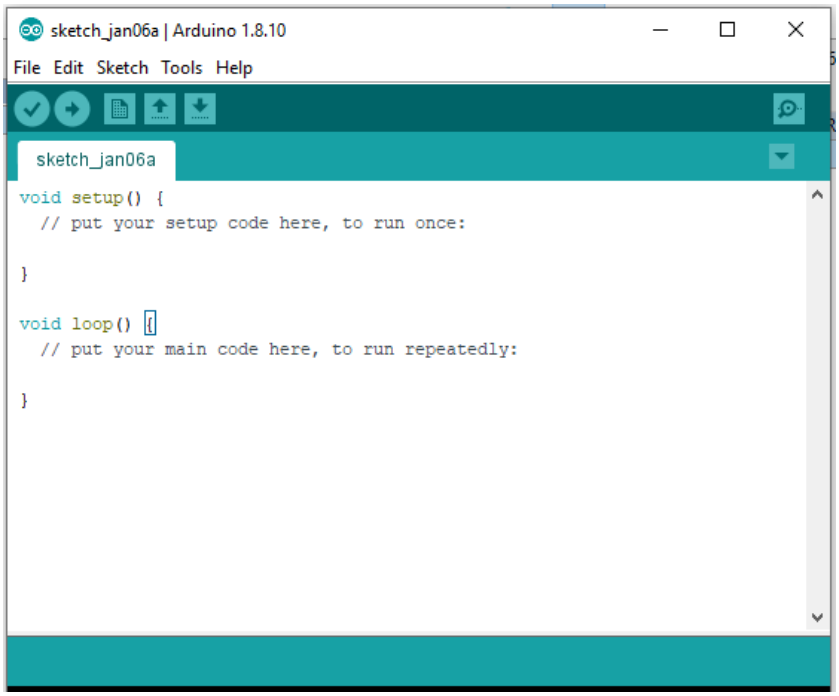

Figura 1.12: Interfaz de programación Arduino [11]

1.5.1.2. LOGI.CAD3

Construida sobre eclipse, es otro de los lenguajes de programación compatibles con los Arduinos. Este lenguaje se basa en el estándar industrial IEC 61131-3 que permite un software para programar microcontroladores y PLC. Ayuda a la fácil programación en ST (texto estructurado) y FBD (Diagrama de bloques de funciones). Al igual que la placa Arduino, es compatible con plataformas Windows, Linux y Mac OS X en la industria. Tiene como desventaja que su versión gratuita solo permite

usar el editor de código ST, por lo que es necesario comprar su licencia, siendo la más económica la FBD-Light que tiene un costo de 474.81 euros[24].

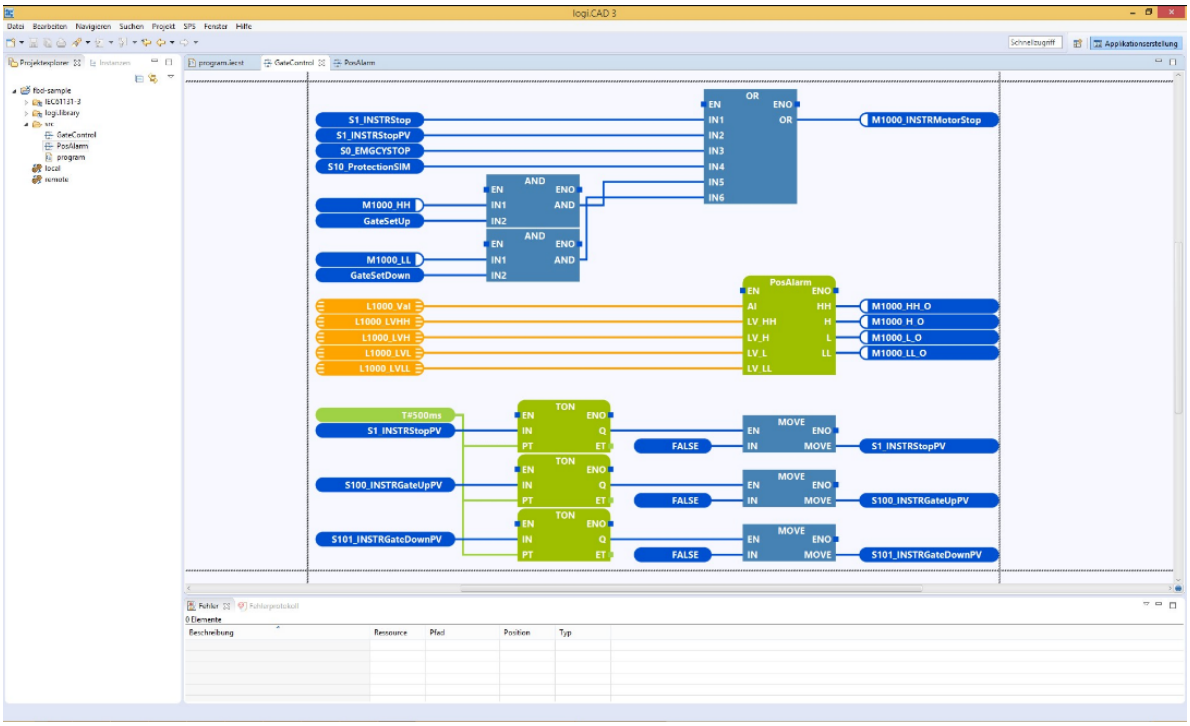

Figura 1.13: Interfaz de programación Logicad3[24]

1.5.1.3. VISUINO

Visuino[25] es un entorno de programación basado en gráficos mayormente visuales para Arduino. Es capaz de diseñar específicas soluciones de automatización industrial e IoT, con solo colocar y enlazar algunos bloques gráficos. De acuerdo con el bloque y la secuencia de los conectores, Visuino entregará el código Arduino deseado. En su versión free solo tiene algunos bloques gráficos básicos.

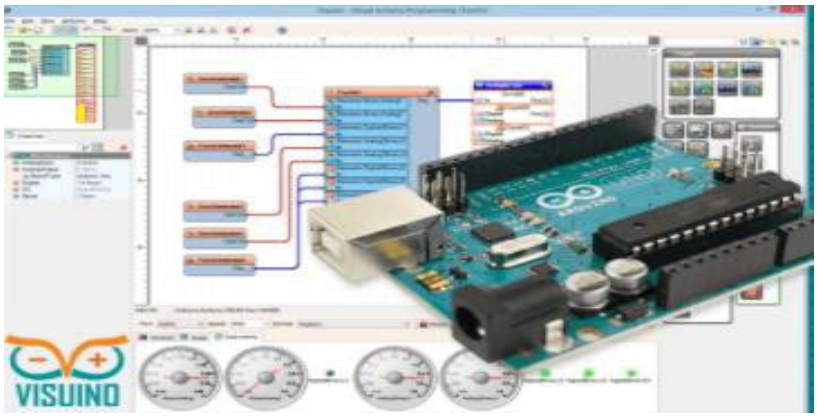

Figura 1.14: Interfaz de programación Visuino[25]

Su fácil entorno, permite una navegación sencilla sobre los diseños robustos. También ayuda por medio de un terminal de comunicación integrado para la ejecución de medidores, instrumentos visuales para monitoreo e inspección en el flujo de datos enviados desde el Arduino. Tiene como desventaja que no soporta la norma IEC 61131-3.

1.5.1.4. EMBRIO

Embrio[26], es un entorno de programación visual que no intenta imitar la programación tradicional, sino que crea su programa a partir de múltiples "agentes", cada uno de los cuales tiene un trabajo y todos se ejecutan al mismo tiempo en paralelo. Los agentes se adhieren entre sí, activando y reprimiendo otros agentes similares a las neuronas del cerebro. Los agentes se implementan agregando y conectando nodos visuales, y puede escribir sus propios nodos personalizados, lo que le brinda todo el poder de la codificación tradicional envuelto en una interfaz gráfica fácil de usar. Trabaja con una conexión en vivo al Arduino para obtener comentarios e interacción en tiempo real, evitando la complicada compilación, carga y ciclo de prueba de la programación tradicional. No soporta la norma IEC 61131-3.

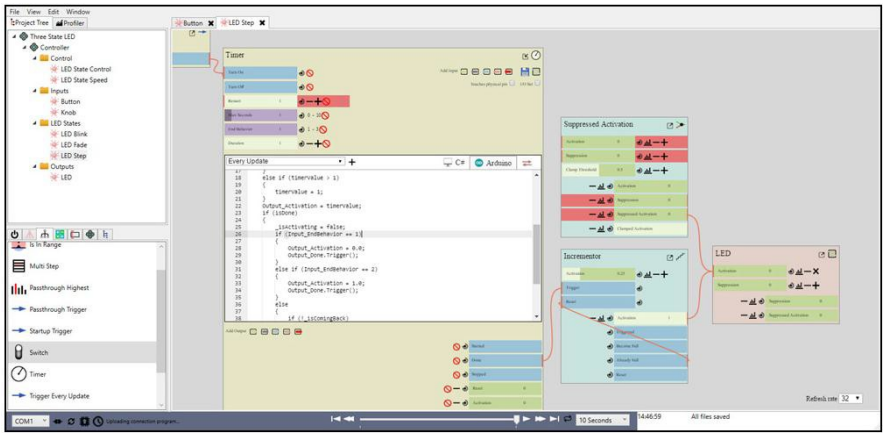

Figura 1.15: Interfaz de programación Embrio[26]

1.5.1.5. LABVIEW

El reconocido compilador Arduino para LabVIEW[27], es una marca de laboratorio virtual en instrumentación para la ingeniería perteneciente al National Instruments. Este tipo de lenguaje es básicamente gráfico y se apoya de iconos en vez de líneas de texto para diseñar aplicaciones. Arduito TM y LabVIEW son compatibles para el compilador que emprenderá el programa que descargará los objetos compatibles con el Arduino. El programa o código guardado se reanudará independiente en el objetivo Arduino.

LabVIEW es una gran herramienta para la automatización industrial en conjunto con los PLCs. Permite implementar la programación de software embebida con programación grafica sin la dependencia del lenguaje C++. No soporta la norma IEC 61131-3.

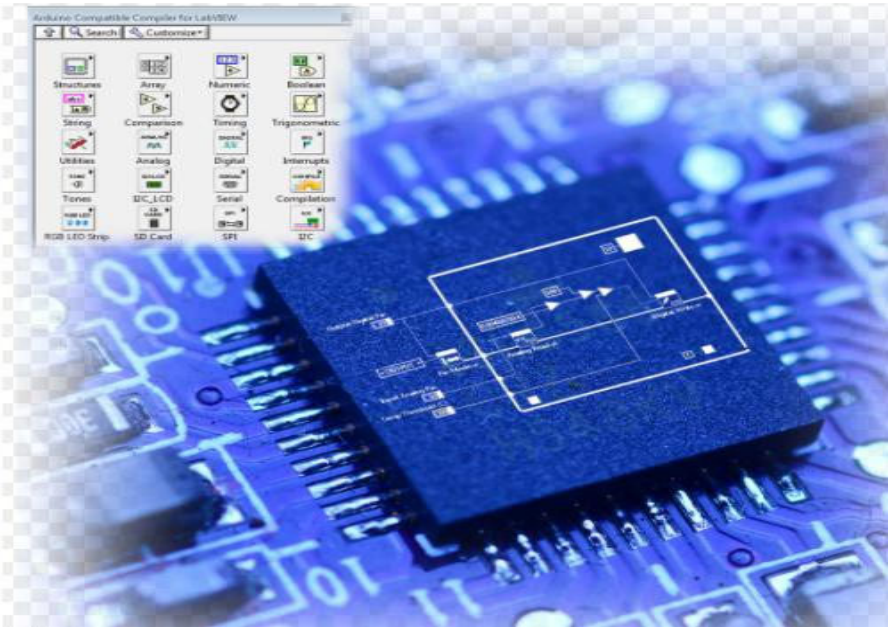

Figura 1.16: *Interfaz de programación LabVIEW[27]*

1.5.1.6. PROGRAMINO

El lenguaje de programación Programino[27], es una alternativa fácil de usar en el entorno IDE de Arduino y Genuino para trabajos aliados con Arduino. Está hecho para múltiples tareas con herramientas disponibles en el editor HTML5 con aplicaciones IoT que facilitan el diseño en la arquitectura de su proyecto.

Programino, está basado en formatos de archivos admitidos por Arduino, C++, encabezados C, HTML, HTML5, JavaScript, CSS, texto. No soporta la norma IEC 61131-3.

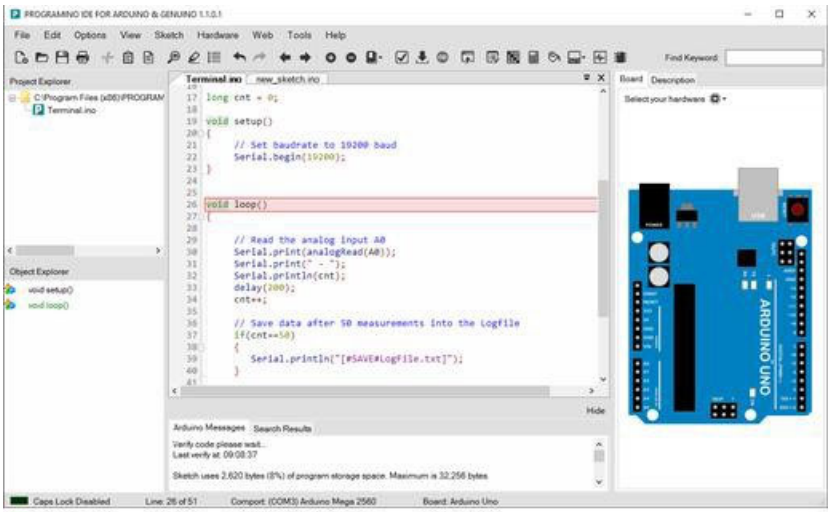

Figura 1.17: *Interfaz de programación Programino[27]*

1.5.1.7. ENTORNO DE PROGRAMACIÓN GEB AUTOMATION

El IDE GEB Automation[4] como se muestra en la Figura 1.15, es un paquete de software de control industrial que nos permite crear proyectos reales locales o distribuidos usando hardware popular como Raspberry Pi y Arduino, es usado además para la programación de PLC atendiendo a las normas estándares IEC 61131-3, su arquitectura está fundamentada sobre las bases de los Sistemas de Automatización Abierta (*Open Automation*) y sus principales elementos son los siguientes:

- Banco de Desarrollo de Aplicaciones (Editor).
- Kernel de los dispositivos ya portados en el IDE
- Simulación de programa avanzada.
- Traductor en código C de las aplicaciones.

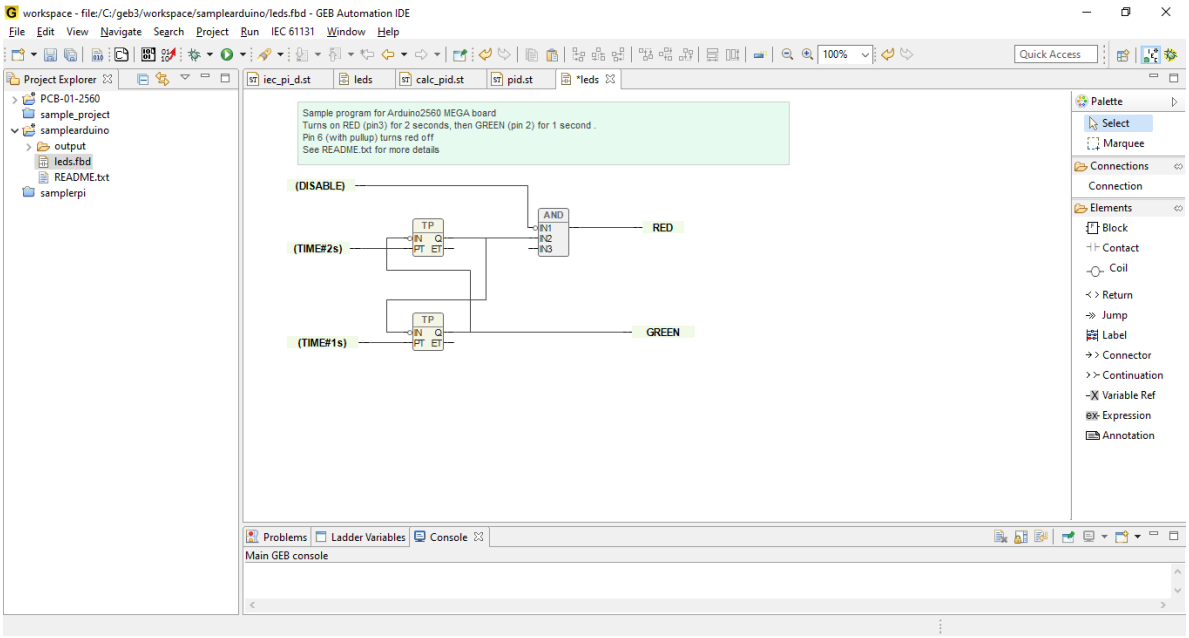

Figura 1.18: *Interfaz de programación GEB Automation[4]*

Banco de Desarrollo de Aplicaciones

El Banco de Desarrollo de Aplicaciones suministra los lenguajes de control LD, ST, IL y FBD del estándar IEC 61131-3, con editores gráficos y de texto profesionales que cumplen con la norma IEC 61131-3, posee opciones de deshacer en los editores textuales y gráficos con integración sencilla con control de revisiones (GIT, CVS, ...)

Kernel de los dispositivos ya portados en el IDE

Es un motor de ejecución, optimizado y muy rápido que ejecuta el código de las aplicaciones desarrolladas en el Banco de Desarrollo y asegura que estas puedan ser usadas sobre la plataforma de hardware sin necesidad de modificación.

Simulación de programa avanzada

El IDE posee un simulador gráfico y textual completamente dentro del IDE que ejecuta uno o varios programas en modo por pasos o continuo, el depurador incluye las características de paso hacia / sobre puntos de interrupción con muestreo de todas las variables, los lenguajes gráficos permiten monitorear el estado de los enlaces incluso cuando se ejecuta en modo continuo, permitiendo configurar manualmente las variables de entrada simuladas y supervisar visualmente las salidas simuladas.

Traductor en código C de las aplicaciones

Es el paso intermedio para el desarrollo y la programación de PLC modernos, genera código ANSI-C estándar altamente legible con proceso de construcción flexible, se adapta al compilador de C desde su dispositivo con un código C eficiente, en velocidad y espacio: requisitos de memoria flash en el dispositivo tan bajos como 20KB.

1.6. ACTUALIDAD CIENTÍFICA INTERNACIONAL

Para el sector industrial, los dispositivos computacionales más comunes para controlar la automatización y otros factores son los PLC. Estos normalmente son diferentes a los otros dispositivos computacionales, ya que están diseñados para las condiciones extremas que se producen en las plantas de fabricación. Esto significa que pueden soportar el polvo, temperaturas más altas y bajas y a humedad, condiciones ambientales que Arduino no está fabricado para soportar.

Los PLC también tienen entradas/salidas (E/S) más extensas para conectarlos a otros sensores y actuadores. Un PLC puede transmitir a otros elementos, incluidos los motores eléctricos, relés magnéticos, sirenas, lámparas indicadoras y mucho más. Esto es algo que el Arduino también puede hacer, pero de forma más limitada, ya que sus entradas análogas solo van desde 0 a 5V y las salidas análogas son de modulación por ancho de pulsos (PWM).

1.6.1. **GALLEGO OLIVARES, LUCÍA. ESTUDIO E IMPLEMENTACION DE UN SISTEMA DE DETECCION DE CAIDAS MEDIANTE EL USO DE UN ACELEROMETRO. 2020.[28]**

La detección de caídas es un área importante de investigación debido a su estrecha relación con la atención médica, especialmente para las personas mayores. Es necesario diseñar un sistema fiable y rápido para minimizar las posibles lesiones una vez que se ha producido la caída, de manera que se ejecute la intervención por parte de los servicios de emergencia lo más rápido posible. El objetivo de este trabajo es desarrollar un sistema fiable que sea capaz de obtener la señal de aceleración cuando el sujeto está llevando a cabo una actividad diaria y ser capaz de identificar

la detección de caídas en tiempo real. En primer lugar, se determinará una posición adecuada del sensor para la adquisición de la señal de aceleración durante la caída. A continuación, se desarrollará el sistema de hardware para la adquisición de datos basados en acelerómetros. El sistema de hardware se compone de un acelerómetro y un microcontrolador alimentado por una batería externa. El microcontrolador es el responsable de leer la señal de salida del acelerómetro por un puerto SPI, y enviarlo al PC a través de una red WIFI. Asimismo, se desarrollará el programa de MATLAB que permita la lectura y análisis de los datos para la detección de caídas. Para ello, se propone en esta tesis determinar los parámetros característicos de la señal de aceleración que permiten hacer la diferencia entre pasos normales del sujeto y caídas.

1.6.2. **SISTEMA DE SUPERVISIÓN LOW-COST CON ALGORITMO DE DIAGNÓSTICO PREDICTIVO DE PUNTOS CALIENTES EN PANELES FOTOVOLTAICOS.2020.[29]**

Este trabajo de final de máster pretende buscar una solución factible y óptima capaz de detectar una anomalía en las curvas de funcionamiento de los paneles solares, con el fin de frenar posibles apariciones de puntos calientes desde su fase inicial, tratando de evitar así que el fallo llegue a ser catastrófico. Para implementar la idea descrita, se hace uso de un dispositivo cuyo software es gratuito, se trata de Arduino UNO, que se le agrega un módulo de expansión de entradas analógicas (29 entradas en total) para conectar los sensores de corriente y de voltaje de hasta 14 paneles fotovoltaicos por cada dispositivo Arduino, y una tarjeta de red para conectar el dispositivo a la red de Internet mediante un cable Ethernet. La supervisión y monitorización se lleva a cabo mediante Codesys, un entorno gratuito de desarrollo para la programación de controladores conforme con el estándar industrial internacional IEC 61131-3.

1.6.3. **GARCÍA REIG, FERNANDO. DISEÑO Y FABRICACIÓN DE UN BRAZO ROBÓTICO DE 6 GDL DE BAJO COSTE BASADO EN ARDUINO. DISS. 2020[30]**

En este Trabajo de Fin de Grado se ha diseñado y fabricado un brazo robótico, tal y como se indica en el título de este. Para ello se ha comenzado por la selección de los componentes, en el que se ha optado por utilizar 6 servos, lo cual le da esos 6

grados de libertad al brazo robótico. Por otra parte, los motores (y el brazo) se controlan mediante dos joysticks. La placa y el lenguaje en el que se ha decidido programar es Arduino. En segundo lugar, se ha procedido a diseñar las partes que forman el brazo en el programa de diseño 3D, Solidworks. Para la fabricación de las piezas que forman el robot, se ha optado por fabricación en impresora 3D. El ensamblaje de estas partes con el resto de los componentes se ha hecho utilizando materiales de uso común y herramientas de alcance general. Se han realizado pruebas tanto con el software como con los componentes, y determinado también la elección o inclusión de nuevos componentes acordes a las necesidades, que han variado desde la idea inicial. En estas pruebas también se ha visto la necesidad de incluir una fuente de alimentación externa más potente para poder mover los 6 servos al mismo tiempo. La finalidad de este proyecto es crear un brazo robótico en su totalidad a partir de conocimientos básicos y semi-avanzados de la robótica, fabricación e impresión 3D, diseño de las partes en 3D y elección de componentes adecuados y comunes de uso extendido.

1.6.4. PROGRAMACIÓN DE LABORATORIOS DE BIOLOGÍA PORTÁTILES ABIERTOS BASADOS EN ARDUINO CON EL LENGUAJE DE PROGRAMACIÓN VISUAL XOD[31]

Es un proyecto cuyo principal objetivo es adaptar la tecnología existente para que personas sin conocimientos técnicos de informática sea capaz de programar y manejar de manera intuitiva y sencilla un laboratorio portátil que pueda estar formado por componentes baratos y que se puedan sustituir por otros parecidos de una forma fácil.

El software XOD está basado en bloques, y permite simular componentes compatibles con Arduino y conectarlos gráficamente entre sí y con otros nodos que se encarguen del control y funcionamiento en un hardware compatible con Arduino. Es gratuito y se pueden crear nuevos nodos para los componentes que se vayan a emplear o utilizar los nodos ya existentes sin modificar o adaptarlos, ya que el código de los nodos es abierto, y el lenguaje que utiliza es C++.

1.6.5. IMPLEMENTACIÓN DEL SISTEMA ELÉCTRICO-ELECTRÓNICO Y SISTEMA DE SOFTWARE DE UNA MÁQUINA CNC[32]

En este artículo se presenta el funcionamiento de una máquina de Control Numérico Computarizado (CNC en inglés Computer Numerical Control) controlado por un firmware (GRBL) gratuito de código abierto que se ejecuta en la tarjeta Arduino Uno R3, de esta manera se describen y justifican los componentes que hacen parte del sistema eléctrico-electrónico y los programas informáticos que constituyen el sistema de software para el mecanizado de placas de circuitos impresos. Las pruebas realizadas registraron un tiempo estimado de 21: 86 en minutos: segundos dividido en los tres procesos del grabado (caminos 13:12, orificios 6:48 y contorno 2:26 dados en minutos: segundos), estos valores pueden variar dependiendo las dimensiones de la PCB requerida, aun así, resulta ser más eficiente en calidad y tiempo comparado con el método convencional de atacado químico.

1.6.6. **DISEÑO Y CONSTRUCCION DE UN SISTEMA AUTOMATIZADO DE CONTROL DE BOMBAS DE AGUA EN UN CULTIVO HIDROPONICO EN EL ENTORNO ARDUINO[33]**

El objetivo del trabajo es desarrollar un sistema automatizado de riego en torno al Arduino para cultivos hidropónicos. El sistema utiliza una placa Arduino ATmega328P a la que se conectan diferentes componentes, tales como un módulo de reloj en tiempo real, pantalla LCD I2c, dos módulos relé de un canal y resistencias eléctricas; con la finalidad de obtener un sistema de micro controlador programable que puede activar y desactivar la bomba de agua de un invernadero en tiempos determinados de acuerdo al desarrollo del cultivo, particularmente en los cultivos de lechugas con diferentes variedades, para ello es necesario la utilización de diferentes técnicas de control y la realización de varias pruebas experimentales con el fin de justar ciertos parámetros de control, especialmente la humedad y temperatura. Este sistema fue instalado en el invernadero de la Comunidad de Mollepata en Ayacucho, dentro del convenio trianual “Inti, la energía que alimenta la Tierra” entre la Universidad Nacional de San Cristóbal de Huamanga (UNSCH) y el Comité Regional ’éducation Pour le Développement International de Lanaudière (CREDIL–JOLIETTE) – Canadá.

La programación se realizó en el entorno de desarrollo integrado (IDE) de Arduino, el cual es una aplicación multiplataforma.

1.7. HERRAMIENTAS Y TECNOLOGÍAS

Las herramientas son un conjunto de programas que son usados por los diseñadores y los programadores para construir sistemas. La utilización de tecnologías permite aplicar conocimientos y habilidades con el objetivo de conseguir una solución que permita resolver un problema determinado y lograr satisfacer las necesidades del proyecto.

1.7.1. LENGUAJE UNIFICADO DE MODELADO (UML)

El Lenguaje Unificado de Modelado (UML) es un lenguaje que se centra en la representación gráfica de un sistema. Ofrece un estándar para representar y modelar la información con la que se trabaja en las fases de análisis y, especialmente, de diseño.

El lenguaje UML tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, hasta la implementación y configuración con los diagramas de despliegue. Es independiente del lenguaje de programación y de las características de los proyectos. UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura de hardware donde se ejecuten.

Permite dimensionar mejor los riesgos de un proyecto en base a tener un mejor rendimiento antes de construir el sistema y facilita la documentación de las decisiones de la arquitectura del proyecto. Además, ofrece un mejor soporte a la planificación y control del proyecto.[34]

1.7.2. HERRAMIENTA CASE

La realización de un nuevo software requiere que las tareas sean organizadas y completadas en forma correcta y eficiente. Las herramientas CASE (Computer Aided Software Engineering) fueron desarrolladas para automatizar esos procesos y facilitar las tareas de coordinación de los eventos que necesitan ser mejoradas en el ciclo de desarrollo de software. Se pueden definir como un conjunto de programas y

ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software.[35]

1.7.3. VISUAL PARADIGM PARA UML V16.2

Visual Paradigm para UML (VP-UML) es una herramienta CASE de diseño UML diseñada para ayudar al desarrollo de software. Ofrece un completo conjunto de herramientas de los equipos de desarrollo de software necesario para la captura de requisitos, la planificación de controles, el modelado de clases y el modelado de datos. Es multiplataforma y está concebida para soportar el ciclo de vida completo del proceso de desarrollo del software. Posee un diseño centrado en casos de uso y enfocado al negocio generando así un software con mayor calidad. Por otra parte, las imágenes y reportes que genera son de muy buena calidad.[36]

1.7.4. LENGUAJE DE PROGRAMACIÓN C

C es un lenguaje de programación de nivel medio con el cual se desarrollan tanto aplicaciones como sistemas operativos a la vez que forma la base de otros lenguajes más actuales como Java, C++ o C#. Es un lenguaje de programación de propósito general que ofrece economía sintáctica, control de flujo y estructuras sencillas y un buen conjunto de operadores. No depende del hardware, por lo que se puede migrar a otros sistemas. Ofrece un control absoluto de todo lo que sucede en el ordenador. Garantiza una organización del trabajo con total libertad. Los programas son producidos de forma rápida y son bastante potentes.[37, 38]

1.7.5. LENGUAJE DE PROGRAMACIÓN C++

C ++ es uno de los lenguajes de programación más usados en los últimos años, completamente orientado a objetos que brinda soporte a las técnicas de desarrollo y a la reutilización de componentes de software. Es una de sus principales características, el alto rendimiento que ofrece. Esto es debido a que puede hacer llamadas directas al sistema operativo, es un lenguaje compilado para cada plataforma, posee gran variedad de parámetros de optimización y se integra de forma directa con el lenguaje ensamblador. Garantiza la compatibilidad con

bibliotecas enriquecidas y estándar de funciones que ayudan a escribir código rápidamente.

Es un lenguaje fuertemente tipado y tiene un conjunto completo de instrucciones de control, permitiendo la agrupación de estas. Por otra parte, la compilación y ejecución de un programa en C++ es mucho más rápida que en la mayoría de lenguajes de programación.[39-42]

1.7.6. ENTORNO DE DESARROLLO INTEGRADO

Un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) es una aplicación de software que proporciona servicios integrales a los programadores de computadoras para el desarrollo de software. [43]

Un IDE normalmente se compone de:

- Un editor de texto.
- Un compilador.
- Un intérprete.
- Herramientas de automatización.
- Un depurador.
- Posibilidad de ofrecer un sistema de control de versiones.
- Factibilidad para ayudar en la construcción de interfaces gráficas de usuario.

1.7.7. ATMEL STUDIO

ATMEL, es una solución de programación directa. Consta de programación para CONTROLLINO sin barreras en el código. Tiene un alto uso potencial en los microcontroladores industriales, con la posibilidad de monitorear sensores de temperatura y a control remoto desde el panel de control. frece un entorno sencillo y fácil de usar para escribir, construir y depurar sus aplicaciones escritas en C / C ++ o código ensamblador. También se conecta sin problemas a los depuradores y kits de desarrollo de Atmel. Posee una amplia biblioteca de código fuente, incluidos controladores y pilas de comunicación.[44]

Figura 1.19: Interfaz de programación Atmel[44]

1.7.8. EMBARCADERO C++BUILDER 10.3.3 COMMUNITY EDITION

C++Builder en su edición Community Edition es un IDE gratuito para crear aplicaciones nativas multiplataforma y pensada para escribir, compilar, depurar y ejecutar programas desarrollados en C++. Proporciona herramientas de desarrollo integradas y de nivel profesional. Incluye un editor de código, herramientas integradas de depuración que te permiten depurar en cualquier dispositivo y un acceso integrado a bases de datos locales populares con datos en tiempo real en tiempo de diseño. Permite el diseño visual de UI con los frameworks VCL y FireMonkey, con soporte para diseño de píxel perfecto y específico para la plataforma. Posee cientos de componentes incluidos para mejorar la aplicación y reducir los ciclos de desarrollo.[45]

1.8. DISEÑO DEL SOFTWARE

El Sistema Informático de Programación de Autómatas Programables, propuesto en la presente tesis, sobre tecnología Arduino tendrá una arquitectura básicamente compuesta por 6 partes como se muestra en la figura 1.17:

- **Banco de Desarrollo de Aplicaciones de GEB Automation:** Editor, compilador y simulador de los lenguajes LD, ST, IL y FBD de la IEC 61131-3, entre otras funcionalidades.

- **Kernel del PLC:** El GEB Automation posee un Kernel en su versión Student limitado al controlador ATMEGA 2560 de tecnología Arduino, por lo que se crea una variante de kernel que pueda soportar los modelos de PLC con tecnología Arduino de desarrollo nacional. Para ello se elaboró un conjunto de bibliotecas en código C utilizando la herramienta Atmel Studio 7.[44]

- **Traductor + Creación asistida del mapa de la red MODBUS:** La aplicación fue desarrollada usando C++Builder en su edición Community Edition[45], encargada de procesar el código "ANSI C" generado por el IDE GEB Automation para optimizarlo y adaptarlo a la estructura del nuevo kernel. En esta fase se adiciona la configuración del mapa de la red MODBUS y un grupo de parámetros generales en forma de estructuras de datos "C", posteriormente se compila este código con las bibliotecas del kernel para obtener el código ejecutable final. Opcionalmente se puede producir código ejecutable para descargarlo directamente en la memoria RAM del PLC o para grabar memorias EPROM.

- **Nuevos Bloques Funcionales para el Control Automático de Procesos:** Conjunto empleado para realizar las acciones de control clásicas como PID, ON-OFF y demás. Para cada uno de estos se suministra una versión "C" optimizada y otra IEC para su simulación en el GEB Automation.

- **Ayuda y Documentación:** Reúne en formato HLP y PDF los detalles del sistema y su forma de uso.
- **Instalador del Sistema:** Posibilita instalar de una manera pilotada todo el sistema abarcando el GEB Automation y las librerías Arduino.

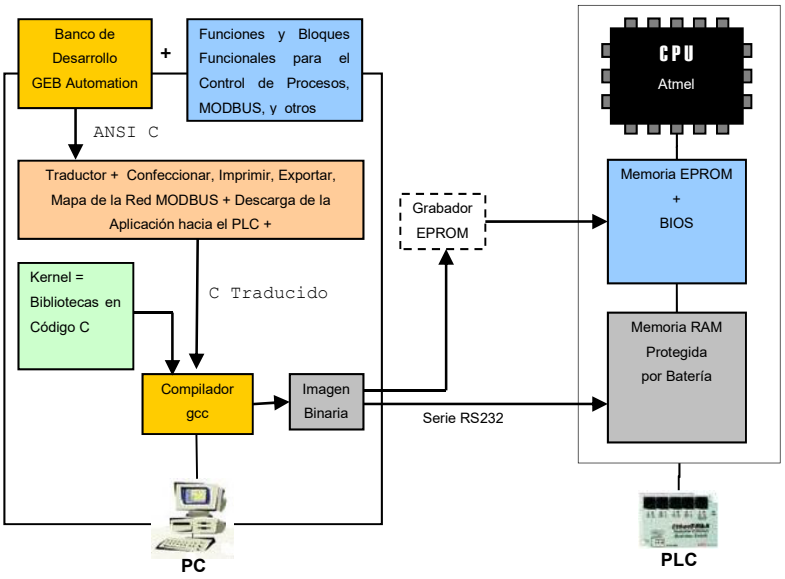

Figura 1.20: Arquitectura del Sistema

1.9. BREVE DESCRIPCIÓN DEL PROCESO DE DESARROLLO DE SOFTWARE

El modelo del Ciclo de Vida del Desarrollo de Software en la División de Automatización, se solapa en el modelo Ciclo de Vida del Proyecto establecido en el Manual del Proyecto (DA-ME-01). Las pautas fundamentales para el desarrollo del software son (DA-PE-002, NC-ISO/IEC 90003:2006, NC-ISO/IEC 12119:2005):

- Determinación y análisis de Requisitos.
- Diseño de la arquitectura.
- Diseño e implementación de prototipos.
- Codificación de los Casos de Uso. Pruebas Unitarias.
- Integración de los Casos de Uso. Pruebas de Integración.
- Pruebas de Sistema. Instalación y Soporte del Software.
- Pruebas de Aceptación y documentación del Software (Manual de Configuración y Operación).

A continuación, se describen brevemente:

Determinación y análisis de requisitos: En esta fase se determinan los requisitos del producto software desde la perspectiva del usuario. Establece los servicios que el sistema debe proporcionar y las restricciones bajo las cuales debe operar. Se especifican las condiciones que determinan qué debe hacer el sistema y cómo debe hacerlo. Los requisitos deben revisarse de acuerdo a la instrucción DA-PE-002.

Diseño de la Arquitectura: Se establecen decisiones acerca de los recursos de implementación y prototipos a emplear. Entre las actividades se relacionan: Definición de entradas y salidas del software; Definición de Caso de Uso; Definición de interfaces, Determinación de Prototipos; Definición de Plataforma y herramientas de programación; Desarrollo de algoritmos de programación para las funcionalidades.

Luego de la implementación de los Prototipos, se realiza la codificación de los Casos de Uso y las Pruebas Unitarias, el cual supone todo el proceso de escribir el código software necesario que hará posible que el sistema finalmente implementado cumpla con las especificaciones establecidas y responda al diseño del sistema descrito, las pruebas tienen como objetivo comprobar que el Caso de Uso, entendido como una unidad funcional de un programa independiente, está correctamente codificado.

Integración de los Casos de Uso. Pruebas de Integración. Se determina la secuencia en que se van a producir e integrar los componentes, tienen el propósito de asegurar que no haya errores de interfaces y de encontrar defectos en el sistema.

1.10. Breve descripción del proceso de evaluación de software

En el año 1994 se produjeron dos series de normas: ISO/IEC 9126 referido al modelo de calidad del producto software y la ISO/IEC 14598, referido a la evaluación de la calidad del producto. Una nueva propuesta de calidad surgió en el año 2000, con la serie de normas ISO/IEC 25000, con la idea de proponer un nuevo marco de referencia para la calidad del producto software, mediante la especificación de requisitos y evaluación de características de calidad. La ISO/IEC 25000 sustituye las normas ISO/IEC 9126 y la ISO/IEC 14598.

La Norma ISO/IEC 25040 define el proceso para llevar a cabo la evaluación del producto software. Dicho proceso consta de un total de cinco actividades: establecer los requisitos de evaluación, especificar la evaluación, diseñar, ejecutar y concluir la evaluación (DA-PE-003) (Ver Figura 1.18).

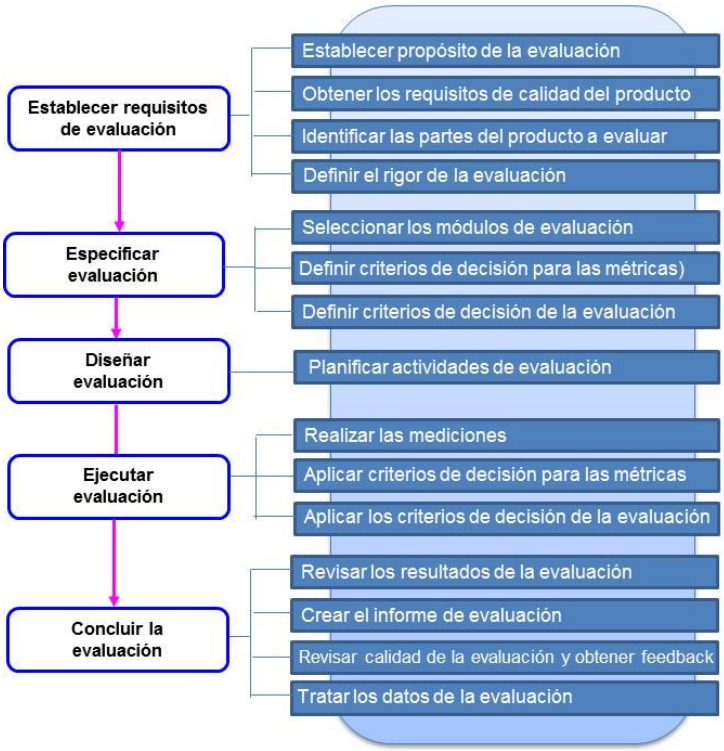

Figura 1.21: Proceso de evaluación del proceso y producto de software Fuente: (DA-PE-003)

CONCLUSIONES

Se expuso la composición del software de los PLCs, formados, principalmente, de un sistema operativo y un entorno de programación. Se caracterizan las herramientas para el desarrollo de aplicaciones de control de procesos tecnológicos para tecnología Arduino. Se aborda brevemente, la metodología general a aplicar para el desarrollo del software y finalmente, se profundiza en el Entorno de Programación GEB Automation.

Capítulo 2. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA INFORMÁTICO DE PROGRAMACIÓN DE AUTÓMATAS PROGRAMABLES SOBRE TECNOLOGÍA ARDUINO

INTRODUCCIÓN

En el presente capitulo se describe el proceso de diseño e implementación del Sistema Informático de Programación de Autómatas Programables sobre tecnología Arduino, se describen las funcionalidades que brinda el sistema y se analizan las prestaciones y beneficios de la aplicación informática desarrollada según normas internacionales creadas al efecto y la experiencia en el uso de la herramienta en diferentes industrias. Por último, se analizan las ventajas económicas, sociales y medioambientales que ofrece la aplicación del sistema informático.

2.1. ESPECIFICACIÓN DE LOS REQUISITOS

Los requisitos describen los servicios (funciones) que se esperan del sistema. Es la capacidad del producto del software para proveer funciones que cumplan con necesidades específicas o implícitas, cuando el software es utilizado bajo ciertas condiciones. Se presentan como sigue: (RG01/DA-PE-002).

- Descripción: Nombre o descripción del requisito.
- Entrada: Descripción de las entradas que el sistema necesita para ejecutar la funcionalidad. Es orientada al usuario, por ejemplo, autenticarse.
- Procesamiento: Descripción de lo que el sistema debe hacer con las entradas que recibe, con sus condiciones y restricciones. Es orientada al desarrollador ¿Qué debe lograr la aplicación?
- Salida: Descripción de las respuestas o nuevo estado del sistema. Es orientado al usuario ¿Qué logró realizar el usuario?

2.1.1. REQUISITOS FUNCIONALES

Los requisitos funcionales se identifican por empezar con la letra RF y un número consecutivo. Además, cada requerimiento está asociado a un módulo que se

identifica de la siguiente manera: el módulo de gestión de Proyectos GEB Automation con la letra P, el módulo de gestión del Mapa Modbus con la letra M, el módulo de Configuración con letra C y el módulo de programación del Firmware con la letra F.

Se mide cualitativamente el riesgo que se corre a la hora de implementar el requerimiento (despreciable, marginal y crítico), y la prioridad que se la asignará a cada uno (baja, media y alta).

En la siguiente tabla están listados los requisitos funcionales del sistema.

Tabla 2.1 Lista de Requisitos Funcionales

ID	Requisitos Funcionales Adecuación	Módulo	Riesgo	Prioridad
RF1	Listar proyectos de GEB Automation	P	Despreciable	Media
RF2	Abrir proyecto GEB Automation	P	Despreciable	Alta
RF3	Mostrar información general	M	Despreciable	Media
RF4	Listar programas	M	Despreciable	Media
RF5	Listar variables booleanas	M	Despreciable	Critico
RF6	Listar variables analógicas	M	Despreciable	Critico
RF7	Listar variables de tiempo	M	Despreciable	Critico
RF8	Listar variables de mensajes	M	Despreciable	Critico
RF9	Mostrar mapa Modbus - Scada	M	Despreciable	Alta
RF10	Agrupar variables según grupo de protocolo Modbus	M	Marginal	Alta
RF11	Configuración del Mapa Modbus	M	Marginal	Alta
RF12	Exportar Mapa Modbus	M	Marginal	Alta
RF13	Editar opciones generales	C	Marginal	Media
RF14	Editar opciones de exportación del mapa Modbus	C	Crítico	Media
RF15	Editar configuración del PLC	C	Crítico	Alta
RF16	Programar PLC	F	Crítico	Alta

RF1 - Descripción: Listar proyectos de GEB Automation

Entrada:

- Directorio donde se encuentra el proyecto GEB Automation

Procesamiento:

- Realizar búsqueda dentro de la carpeta de proyectos de GEB Automation.
- Determinar cuál de los proyectos existente corresponde a dispositivo de tecnología Arduino.
- Mostrar diálogo de apertura de proyectos agrupados por tipo de dispositivo.

Salida:

- Capacidad del producto de ver agrupados por dispositivos los proyectos realizados en GEB Automation con tecnología Arduino

RF2 - Descripción: Abrir proyecto GEB Automation

Entrada:

- Vínculo del proyecto GEB Automation

Procesamiento:

- Obtener código c generado desde proyecto GEB Automation.
- Realizar ingeniería inversa al código c para obtener los programas desarrollados en el editor de programas en GEB Automation.
- Realizar ingeniería inversa al código c para obtener los las variables declaradas en cada uno de los programas.
- Almacenar en base de datos SQlite los programas y variables obtenidos a partir de realizar la ingeniería inversa.
- Optimización del código c.

Salida:

- Importación del proyecto GEB Automation.
- Estructura visual en forma de árbol del proyecto.

RF3 - Descripción: Mostrar información general

Entrada:

- Información almacenada en base de datos SQlite con los programas y variables.

Procesamiento:

- Mostrar características esenciales del proyecto.
- Mostrar fecha de importación del proyecto GEB Automation

Salida:

- Datos generales del proyecto

RF4 - Descripción: Listar programas

Entrada:

- Información almacenada en base de datos SQlite con los programas y variables.

Procesamiento:

- Listar los programas que fueron desarrollados en cualquiera de los lenguajes de la norma IEC 61131-3 en el editor de programas de GEB Automation.

Salida:

- Listado de programas

RF5 - Descripción: Listar variables booleanas

Entrada:

- Información almacenada en base de datos SQlite con los programas y variables.

Procesamiento:

- Listar todas las variables booleanas sin importar en cuál de los programas del proyecto fueron declaradas.

Salida:

- Listado de variables booleanas

RF6 - Descripción: Listar variables analógicas

Entrada:

- Información almacenada en base de datos SQlite con los programas y variables.

Procesamiento:

- Listar todas las variables analógicas sin importar en cuál de los programas del proyecto fueron declaradas.

Salida:

- Listado de variables analógicas

RF7 - Descripción: Listar variables de tiempo

Entrada:

- Información almacenada en base de datos SQlite con los programas y variables.

Procesamiento:

- Listar todas las variables de tiempo sin importar en cuál de los programas del proyecto fueron declaradas.

Salida:

- Listado de variables de tiempo

RF8 - Descripción: Listar variables de mensajes

Entrada:

- Información almacenada en base de datos SQlite con los programas y variables.

Procesamiento:

- Listar todas las variables de mensajes sin importar en cuál de los programas del proyecto fueron declaradas.

Salida:

- Listado de variables de mensajes

RF9 - Descripción: Mostrar mapa Modbus - Scada

Entrada:

- Diccionario de variables.

Procesamiento:

- Muestra una lista de variables, donde a cada una de ellas se le asigna una dirección de forma automática o manual. Cuando se da clic sobre este elemento

en el árbol de información, a la derecha de la ventana aparece la información del mapa en la parte superior. Sus columnas son: Dirección de red, Variable (nombre), Tipo, Sentido, Atributo y Rango. En la parte inferior aparecen las variables disponibles para formar el mapa. En él no se aceptan valores constantes, definiciones, variables locales a un programa o subprograma y variables del tipo mensajes.

Salida:

- Archivo applivm.mbs donde se almacena la configuración realizada al Mapa Modbus.

RF10 - Exportar Mapa Modbus

Entrada:

- Mapa Modbus

Procesamiento:

Se selecciona uno de los dos tipos de exportación existentes para la aplicación, ya sea hacia un fichero o hacia el portapapeles, además permite exportar todas las variables o solamente las seleccionadas.

Salida:

- Fichero de exportación del mapa modbus.
- Mapa modbus guardado en el portapapeles del sistema operativo

RF11 - Descripción: Agrupar variables según grupo de protocolo Modbus

Entrada:

- Mapa Modbus de variables.

Procesamiento:

- La dirección de red de una variable está formada por 2 bytes (4 nibbles); representada en forma hexadecimal, donde un nibble es un dígito hexadecimal. La dirección efectiva de la variable son los tres nibbles menos significativos, y el más significativo representa al grupo.

- Para formar al mapa, ya que se requiere del protocolo Modbus, se necesitan solamente cuatro grupos de variables: Coils (Bobinas), Inputs (Entradas), Input Registers (Registros de entrada) y Holding Registers (Registros internos). Las bobinas son formadas por variables booleanas (discretas) de salida, las entradas por booleanas de entrada, los registros de entradas por analógicas (enteras o reales) de entradas y los registros internos por analógicas de salida, temporizadores y variables internas analógicas.

Salida:

- Variables agrupadas por segmentos de direcciones

RF12 - Descripción: Configuración del Mapa Modbus

Entrada:

- Mapa Modbus de variables.

Procesamiento:

- La configuración del mapa se realiza añadiendo variables disponibles al área del mapa. Esta acción puede ser realizada mediante dos vías:
    1. Dando clic derecho sobre la(s) variable(s) a incluir en dicho mapa, para lo cual aparece un menú contextual con uno o dos elementos; cuando aparece con el elemento “Aplicar cambios” es porque se ha modificado el mapa. Seleccionando “Asignar dirección” la variable pasa a la parte superior de la ventana con una dirección determinada por varios aspectos: que ya exista alguna de su tipo Modbus mapeada (se le asigna una dirección subsiguiente a la última de su tipo, si es booleana se suma 1 sino se suman 2 al valor de la dirección precedente). Si el cuadro combinado “Dirección de Inicio” posee una dirección de inicio diferente de “Todas”, entonces se agrega la variable al mapa, poniéndole como grupo el referenciado por la dirección de inicio antes mencionada; teniendo en cuenta que las variables mapeadas en este grupo sean del mismo tipo Modbus que se agrega o que en el grupo no haya ninguna variable.

2. Arrastrando una (no más) con el puntero del ratón hasta el área del mapa, pudiéndola dejar al final del mapa o insertada entre otra. Aquí se cumple la misma regla de asignación que en el punto anterior.

- Haciendo clic sobre aplicar cambios, todas las modificaciones hechas al mapa son almacenadas para el proyecto activo. De este modo podrá cerrar con seguridad la aplicación, grabar el PLC o imprimir los datos ya configurados para el mapa de dicho proyecto.
- Cada variable mapeada puede ser borrada o se le puede configurar sus propiedades Atributo y Rango. Si se selecciona más de una variable las mismas propiedades se le aplican, por igual, a todas.

Salida:

- Mapa Modbus – Scada actualizado

RF13 - Descripción: Editar opciones generales

Entrada:

- No requiere

Procesamiento:

Configura esencialmente, la interfaz Usuario – Sistema, donde se podrán realizar las siguientes opciones:

- Guardar posición: Guarda tanto la posición de la ventana principal como su tamaño, para que cuando el programa se ejecute nuevamente, se muestre como se vio por última vez.
- Cerrar la Aplicación al finalizar la programación del PLC: Esto se utiliza para que apenas se termine de programar el autómata se cierre el software.
- Variables MODBUS en grupo por defecto: Habilita la posibilidad de utilizar los grupos por defecto, de MODBUS, a la hora de agregar una variable al mapa. Estos grupos son: el segmento 0x (desde 0x0000 a 0x0FFF) para las Bobinas, el 1x (desde 0x1000 a 1FFF) para las Entradas, el 3x (desde 0x3000 a 0x3FFF) para los Registros de entrada y el 4x (desde 0x4000 hasta 0x4FFF) para los Registros internos

- Direcciones en hexadecimal: Muestra las variables listadas en el mapa con su dirección de red en hexadecimal; las variables se mostrarán en decimal, cuando esta caja de selección no esté marcada.

Salida:

- Configuración de la aplicación

RF14 - Descripción: Editar opciones de exportación del mapa Modbus

Entrada:

- No requiere

Procesamiento:

- Configuración de los valores por defecto que va a tener el diálogo de exportación del mapa Modbus. Los valores prefijo y sufijo de las variables, tanto en el nombre, como en el comentario determinan el comienzo y la terminación de estos parámetros para los objetos exportados. Si se tiene una variable cuyo nombre es "anaOutp", un prefijo cuyos caracteres sean "PLC1_" y el sufijo "_TEST"; se mostrará, en el SCADA, un identificador cuyo nombre será "PLC1_anaOutp_TEST".

Salida:

- Configuración de exportación del Mapa Modbus

RF15 - Descripción: Editar configuración del PLC

Entrada:

- No requiere.

Procesamiento:

Configura las opciones usadas durante la programación y el funcionamiento posterior del PLC. Las opciones a configurar son:

- El valor por defecto del tiempo de ciclo: Aquí se especifica el valor del tiempo de ciclo por defecto en milisegundos.

- Al ocurrir errores en tiempo de corrida: Esta opción define la acción a ejecutarse en el PLC al ocurrir una excepción en tiempo de corrida, como por ejemplo una división por cero. Las posibles acciones son:
    1. Deshabilitado: Contabiliza los errores internamente y continua la ejecución normal de la aplicación. Usada generalmente en las aplicaciones finales (aplicaciones ya depuradas, las cuales se van a montar en un proceso).
    2. Disparar perro guardián: Detiene el PLC, deshabilita las interrupciones y espera el disparo del perro guardián desde un lazo infinito.

Salida:

- Configuración del PLC

RF16 - Descripción: Programar PLC

Entrada:

- Proyecto a ejecutar en el autómata.
- Mapa Modbus configurado.

Procesamiento:

Programar el PLC a partir de las entradas. Para ello existen dos formas:

1. Compilación directa para la RAM (Random Access Memory) del autómata.
2. Generación de fichero binario, que contiene el programa que se ejecutará en el autómata, para EPROM, es decir, que usando un grabador de EPROMs se puede generar, en serie, una aplicación para uno o más autómatas sin la necesidad de conectar varios de ellos a su PC.

Salida:

- Autómata programado

2.1.2. REQUISITOS NO FUNCIONALES

Los requisitos no funcionales se identifican por empezar con la letra RNF y un número consecutivo. También se mide cualitativamente el riesgo que se corre a la hora de implementar el requerimiento (despreciable, marginal y crítico), y la prioridad que se la asignará a cada uno (baja, media y alta).

En la siguiente tabla están listados los requisitos no funcionales del sistema.

Tabla 2.2 Lista de Requisitos No Funcionales

ID	Requisito	Riesgo	Prioridad
RNF1	Usabilidad	Crítico	Alta
RNF2	Fiabilidad	Marginal	Alta
RNF3	Eficiencia	Marginal	Media
RNF4	Portabilidad	Despreciable	Baja
RNF5	Documentación de Usuario y Soporte de Ayuda	Marginal	Media
RNF6	Fiabilidad	Marginal	Alta
RNF7	Requisitos legales, derecho de autor y otros	Crítico	Media

RNF1 - Usabilidad

Es la capacidad del producto de software para ser atractivo, entendido, aprendido y utilizado por el usuario bajo condiciones específicas.

Descripción: Interfaz gráfica del Software con el usuario

Entrada:

- Contraseña

Procesamiento:

- La aplicación informática debe garantizar un acceso fácil y rápido, contando con un menú que satisfaga las necesidades de los usuarios.
- La aplicación informática podrá ser usada sólo por especialistas que posean conocimientos avanzados en automática.
- Los botones deben poseer ToolTips que muestren información de la acción que realizan.
- Ante la ocurrencia de algún error, la aplicación informática debe mostrar al usuario el origen de este, además de otras informaciones significativas a modo de ayuda que lo guíen a su solución. También, debe garantizar que el usuario pueda recuperarse lo más rápido posible de algún error cometido durante la realización de alguna operación.

Salida:

- Configuración de la interfaz

Descripción: Navegación

Entrada:

- No requiere contraseña

Procesamiento:

- La navegación de una pantalla a otra no debe requerir más de una secuencia de 3 teclas o clic de ratón.
- Debe proveer un menú jerárquico de navegación el cual es desplegado con un solo clic (tecla) desde cualquier pantalla.
- Las pantallas de procesos deben proveer de botones que permitan volver al despliegue anterior con un solo clic (tecla).
- Navegación entre pantallas.

RNF2 - Fiabilidad

Es la capacidad del producto de software para mantener un nivel especificado de rendimiento cuando es utilizado bajo condiciones específicas. La ausencia de fiabilidad conduce a fallos, es decir, incapacidad de realizar la función que le ha sido encomendada.

Descripción: Capacidad para la tolerancia a fallas

Procesamiento:

- Anulación de operaciones incorrectas y recuperación.
- Disponibilidad y capacidad de recuperación.
  - Fallas de alimentación en el fluido eléctrico de la base.
  - Pérdida de comunicación con el PLC.

Salida:

Sistema con alta disponibilidad, madurez, tolerancia a fallas y capacidad de recuperación.

RNF3 - Eficiencia

Es la capacidad del producto de software para proveer un desempeño adecuado, de acuerdo a la cantidad de recursos utilizados y bajo las condiciones planteadas.

Descripción: Requerimientos de Capacidad

Microprocesador: Pentium III o superior
Memoria RAM: 512 Mb o superior
Adaptador de video: SVGA
Espacio Disco Duro: 0,5 GB o superior
Mouse: PS/2, puerto serie o USB

RNF4 - Portabilidad

Es la capacidad del software para ser trasladado de un entorno a otro, tales como: entornos organizacionales, de hardware o de software.

- El Sistema puede ser adaptado a diferentes entornos XP/Vista/W7/W8/W10) y coexistir con otro software independientemente, en un entorno común, compartiendo recursos comunes, sin aplicar acciones o mecanismos distintos a los proporcionados.
- Puede ser reemplazado fácilmente ante nuevas versiones, para lo que se disponen del instalador y el des instalador.

Los requisitos de portabilidad se adhieren a las siguientes normas:

- NC-ISO/IEC90000:2006. Capítulo 7.2. Procesos relacionados con el cliente. Tópico 7.2.1.1. Requisitos relacionados con el cliente.
- NC-ISO/IEC12119:2005. Capítulo 3. Requisitos de Calidad. Tópico 3.1.8. Afirmaciones referentes a la Portabilidad.

RNF5 - Documentación de Usuario y Soporte de Ayuda

- El usuario tiene la posibilidad de consultar la ayuda en cualquier momento, sin tener que salir de la aplicación, cumpliendo con el requisito de Disponibilidad de la ayuda.

- No impide el uso normal de alguna otra aplicación, cumpliendo con el requisito de ayuda No Obstructiva.
- La ayuda cubre todo el sistema, con Precisión y Detalle.
- Se aplican términos conocidos de la informática y la automatización, cumpliendo con el requisito de Consistencia de la ayuda.
- Disponible Manual del Usuario y Configuración.
- Disponible Guías de Instalación.

RNF6 - Mantenibilidad

Es la capacidad del producto de software para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptación del software a cambios en el entorno y especificaciones de requerimientos funcionales.

- El alcance del mantenimiento del Sistema relativo a correcciones, mejoras, adaptación a cambios del entorno y nuevas especificaciones de requisitos será estipulado en el contrato con el cliente.
- Mediante los registros: RG02/DA-PE-002 Registro de Mejora y RG03/DA-PE-002 Control de versiones, se llevará el control del estado inicial de cada uno de los elementos mantenidos, las partes que han de ser modificadas, recomendaciones para nuevas versiones, el porcentaje de fracasos al reproducir y solucionar las incidencias (causas de fallo del Sistema), densidad de los defectos, garantizando la CAPACIDAD DE SER ANALIZADO y la CAMBIABILIDAD.
- Para asegurar la continuidad del soporte técnico, se dispondrá de copias de respaldo. Las disposiciones en caso de desastres por parte del cliente, deberán establecerse en el contrato.
- Se dispone del código fuente de las versiones para implementar las modificaciones y mejoras, satisfaciendo el requisito de CAMBIABILIDAD.

RNF7 - Legalidad, derecho de autor y otros

- Decreto Ley No. 199 sobre la seguridad y protección de la información Oficial.

- Resolución No. 21/2002 del Ministerio de Ciencia, Tecnología y Medio ambiente sobre el Sistema Nacional de Propiedad Intelectual.
- Certificación del Registro de productores de software del MINCOM.

2.2. ARQUITECTURA DEL SISTEMA PROPUESTO

La arquitectura del sistema se basa en el modelo monolítico, que consiste en crear una aplicación autosuficiente que contenga absolutamente toda la funcionalidad necesaria para realizar la tarea para la cual fue diseñada, sin contar con dependencias externas que complementen su funcionalidad. En este sentido, sus componentes trabajan juntos, compartiendo los mismos recursos y memoria. En pocas palabras, una aplicación monolítica es una unidad cohesiva de código.

Un monolítico podría estar construido como una sola unidad de software o creada a partir de varios módulo o librerías, pero lo que la distingue es que al momento de compilarse se empaqueta como una solo pieza, de tal forma que todos los módulos y librerías se empaquetarán junto con la aplicación principal como se muestra en la Figura 18.

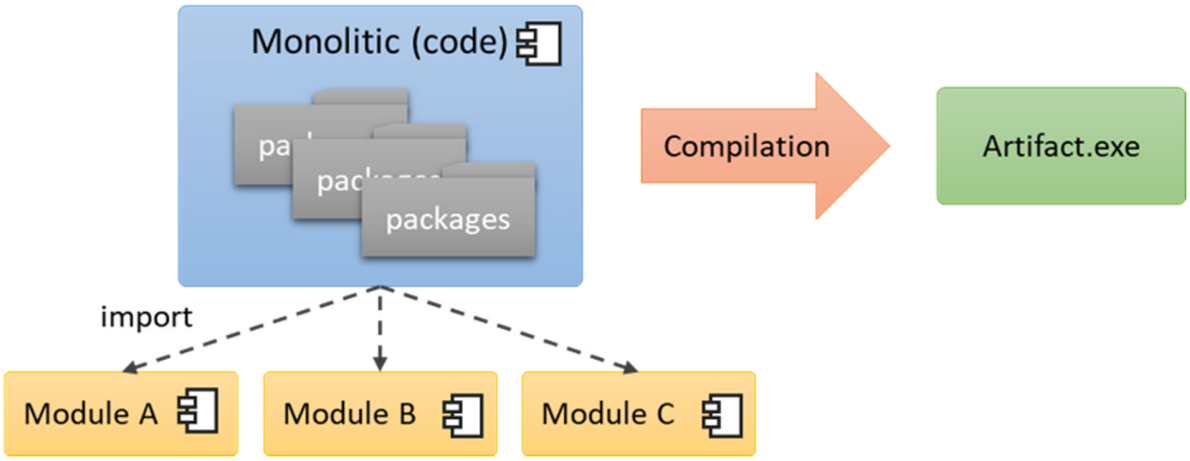

Figura 2.1: Estructura de una aplicación monolítica

Algunas de las ventajas de esta arquitectura son:

- **Fácil de desarrollar**: Debido a que solo existe un componente, es muy fácil para un equipo pequeño de desarrollo iniciar un nuevo proyecto y ponerlo en producción rápidamente.
- **Fácil de escalar:** Solo es necesario instalar la aplicación en varios servidores y ponerlo detrás de un balanceador de cargar.
- **Pocos puntos de fallo:** El hecho de no depender de nadie más, mitiga gran parte de los errores de comunicación, red, integraciones, etc. Prácticamente los errores que pueden salir son por algún bug del programador, pero no por factores ajenos.
- **Autónomo:** Las aplicaciones Monolíticas se caracterizan por ser totalmente autónomas (auto suficientes), lo que les permite funcionar independientemente del resto de aplicaciones.
- **Performance:** Las aplicaciones Monolíticas son significativamente más rápidas debido que todo el procesamiento lo realizan localmente y no requieren consumir procesos distribuidos para completar una tarea.
- **Fácil de probar:** Debido a que es una sola unidad de código, toda la funcionalidad está disponible desde el inicio de la aplicación, por lo que es posible realizar todas las pruebas necesarias sin depender de nada más.

2.3. Prototipo del Sistema Programación de Autómatas Programables sobre tecnología Arduino

Un prototipo en sentido genérico es una implementación parcial pero concreta de un sistema o una parte del mismo que principalmente se crean para explorar cuestiones sobre aspectos muy diversos del sistema durante el desarrollo del mismo.

En la Figura 2.2 se muestra prototipo del diseño elegido de la interfaz principal de la aplicación.

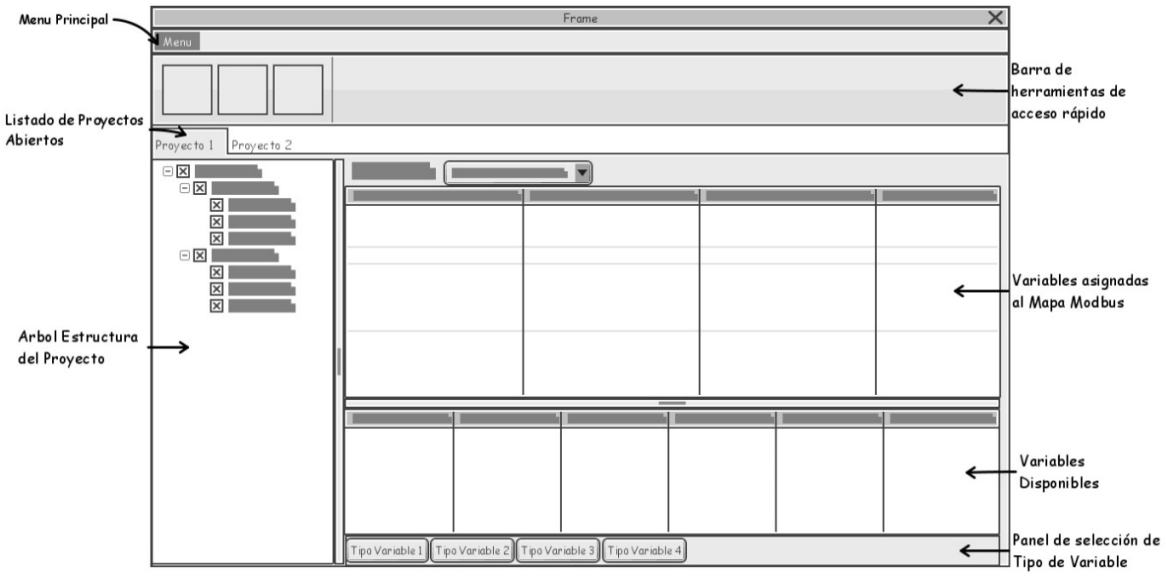

Figura 2.2: Prototipo del Software de Programación

La misma está distribuida en 5 elementos primordiales como son:

2.3.1. **MENÚ PRINCIPAL**

El menú en cualquier tipo de aplicación posee la funcionalidad primordial del acceso a todas las posibilidades que brinda ésta (Ver Figura A.1). A diferencia de una barra de herramientas, este puede estar más cargado de acciones, debido a que el área visual que ocupa es menor; nunca una barra de herramientas tendrá más acciones realizables sobre una aplicación que un menú.

2.3.2. **BARRA DE HERRAMIENTAS DE ACCESO RÁPIDO**

La ventaja del uso de la barra de herramientas, es que su acceso es más rápido. Contiene las acciones fundamentales a realizar en la aplicación: abrir un proyecto, cerrarlo, salir de la aplicación, compilarlo, las opciones de configuración, etcétera (Ver Figura A.2).

2.3.3. **LISTADO DE PROYECTOS ABIERTOS**

La aplicación será tipo MDI (Multiple Document Interface), dando la posibilidad de tener más de un proyecto a la vez abierto para su uso (Ver Figura A.3).

2.3.4. **ÁRBOL DE ESTRUCTURA DEL PROYECTO**

Al abrir un proyecto la información se muestra en forma de un árbol (a la izquierda) donde en sus nodos se representaran la información acerca de los programas de usuarios usados en el proyecto, todas las variables declaradas para el proyecto, ya sean locales o globales: Booleanas, Analógicas, Temporizadores, Mensajes; y el Mapa Modbus – SCADA (Ver Figura A.4).

2.3.5. MAPA MODBUS

Lista de variables del proyecto confeccionado en el GEB Automation para poder utilizarlas en una red MODBUS (Ver Figura A.5)

- **Variables disponibles:** Listado de variables a las cuales no les ha sido asignada dirección dentro del Mapa Modbus.
- **Variables asignadas al Mapa Modbus:** Listado de variables a las cuales ya les fue asignada dirección dentro del Mapa Modbus.
- **Panel de selección de tipos de variables:** Dará opción de filtrado de las variables disponibles por su tipo.

2.4. DESCRIPCIÓN DE LOS CASOS DE USO

El Caso de Uso es la descripción de un conjunto de secuencias de acciones, incluyendo variaciones que un sistema de software realiza para lograr un resultado observable de valor para un actor, es decir, el resultado que es significativo para el rol que un humano, dispositivo de hardware, o aún otro sistema juega con el sistema.

Para el diseño de los casos de uso del sistema se decidió agrupar los requisitos funcionales atendiendo a la acción que cada uno realiza. De esta manera, quedaron agrupados los 16 requisitos funcionales identificados en los 6 casos de uso como se muestran en la Figura 2.8.

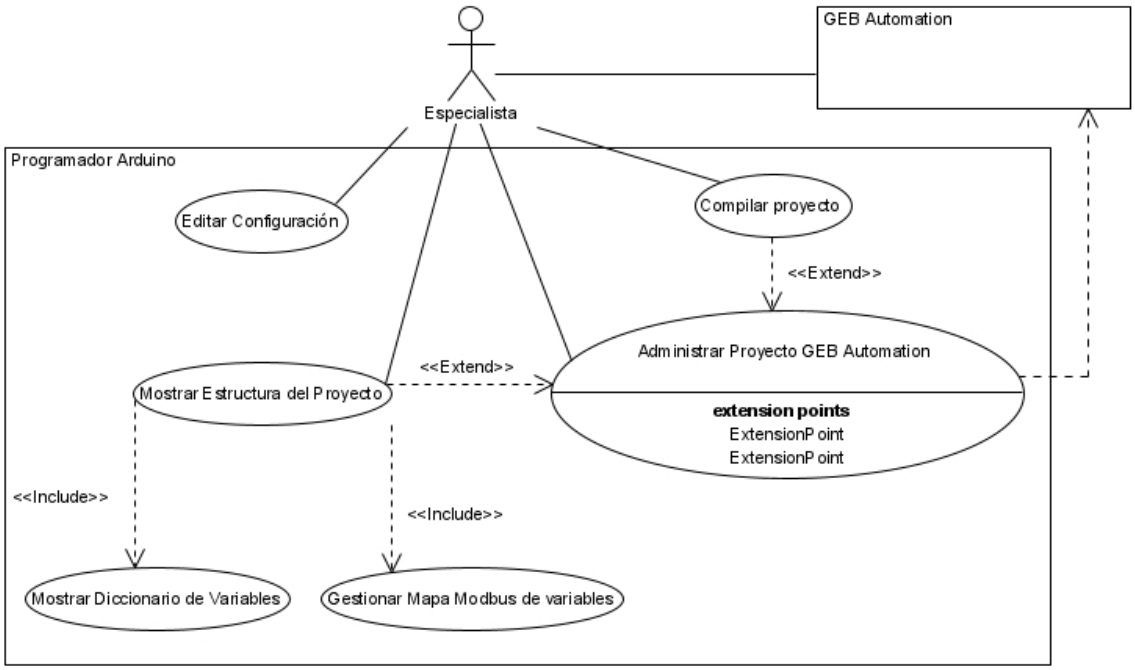

Figura 2.8: Diagrama del Caso de Uso del sistema

Listado de casos de uso

2.4.1. **CU-001. EDITAR CONFIGURACIÓN**

2.4.2. **CU-002. ADMINISTRAR PROYECTO GEB AUTOMATION**

2.4.3. **CU-003. MOSTRAR ESTRUCTURA DEL PROYECTO**

2.4.4. **CU-004. MOSTRAR DICCIONARIO DE VARIABLES**

2.4.5. **CU-005. GESTIONAR MAPA-MODBUS**

2.4.6. **CU-006. COMPILAR PROYECTO**

2.5. VALORACIÓN O CORROBORACIÓN DE LOS RESULTADOS ALCANZADOS

2.5.1 PLAN DE PRUEBAS

- Pruebas Unitarias o de módulo a cada uno de los Casos de Usos. El objetivo es comprobar que el Caso de Uso, entendido como una unidad funcional de un programa independiente, está correctamente codificado.
- Pruebas de integración. Se realizan con el objetivo de asegurar que no hay errores de interfaces y para encontrar defectos en el sistema. Se examinan las

interfases para asegurar que estos componentes individuales son llamados cuando es necesario y que los datos que se transmiten entre dichos componentes son los requeridos.

- Pruebas de Sistema: Este tipo de pruebas tiene como propósito ejercitar profundamente el sistema para verificar que se han integrado adecuadamente todos los elementos del sistema (hardware y software) y que realizan las funciones adecuadas.

Se implementó preliminarmente en el Concentrador Analógico con tecnología Arduino. Este dispositivo está basado en el microchip ATmega328P. Se desarrolló en la División de Automatización. La placa está equipada con 6 pines de entradas analógicas con capacidad de procesar entradas de voltaje de 0 a 10 V o de corriente de 0 a 20 mA y programable con el Arduino IDE. Alimentación con voltaje entre 8 y 28 voltios. Para su comunicación consta de un puerto RS485 con protocolo Modbus RTU y velocidad seleccionable. El mueble, en su parte superior posee 4 indicadores lumínicos que muestra el funcionamiento del equipo. Su fijación es de manera fácil al tener una ranura para rail DIN (Figura 2.9). La información medida en los canales analógicos, se envía por protocolo Modbus al SCADA EROS para supervisión y monitorio.

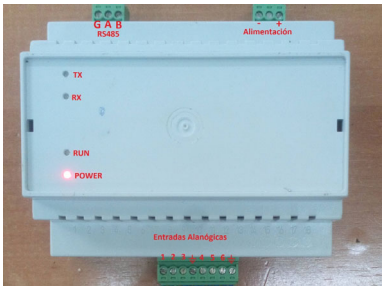

Figura 2.9: Concentrador Analógico con tecnología Arduino

Se utilizó un filtro exponencial EMA (Exponential Moving Average). El Filtrado exponencial EMA es uno de los más empleados en electrónica digital por sus buenos resultados, unidos a una increíblemente sencilla y eficiente implementación. El Filtro EMA consiste en obtener un valor filtrado a partir de una medición mediante la aplicación de la siguiente expresión:

$$A_n = \alpha M + (1 - \alpha) A_{n-1}$$

Siendo An el valor filtrado actual, An-1 el valor filtrado anterior, M es el valor medido de la señal a filtrar, y α es un factor entre 0 y 1.

El resultado de un filtro exponencial EMA es una señal filtrada donde la intensidad de filtrado depende del factor α.

2.5.2 RESULTADOS DE LAS PRUEBAS

Se realizó la evaluación del software mediante el procedimiento DA-PE-003 de la División de Automatización, SERCONI, el cual se basa en las normas ISO/IEC 9126, ISO/IEC 14598 y ISO/IEC 25000 adecuadas al entorno de la organización.

Se construyó una tabla que incluye las columnas: (1) Requisitos funcionales y no funcionales, (2) las características correspondientes a cada requisito, y (3) las especificaciones de los requisitos que básicamente constituyen las métricas utilizadas para la evaluación. Se obtuvo un total de 108 puntos, para una evaluación general (Ev) de 4.3 puntos (0≤Ev≤5), (ver tabla 2.1).

Tabla 2.1. Evaluación (Ev) del CaB-Soporte

Requisitos	**Características**		**Especificaciones**		**Ev**
Funcionales	1	Adecuación	1.1	Listar proyectos de GEB Automation	5
			1.2	Abrir proyecto GEB Automation	5
			1.3	Mostrar información general	5
			1.4	Listar programas	5
			1.5	Listar variables booleanas	5
			1.6	Listar variables analógicas	5
			1.7	Listar variables de tiempo	5
			1.8	Listar variables de mensajes	5
			1.9	Mostrar mapa Modbus - Scada	5
			1.10	Agrupar variables según grupo de protocolo Modbus	5
			1.11	Configuración del Mapa Modbus	5
			1.12	Exportar Mapa Modbus	5
			1.13	Editar opciones generales	5
			1.14	Editar opciones de exportación del mapa Modbus	5
			1.15	Editar configuración del PLC	5
No funcionales	2	Eficiencia	2.1	Requerimientos de Capacidad	5
	3	Usabilidad	3.1	Interfaz gráfica del Software con el usuario	5

			3.2	Navegación	5
	4	Fiabilidad	4.1	Capacidad para la tolerancia a fallas	5
	5	Portabilidad	5.1	Portabilidad (XP/Vista/W7/W8/W10)	5
	6	Documentación y soporte de ayuda	6.1	Documentación y soporte de ayuda	3
	7	Mantenibilidad	7.1	Mantenibilidad	5
	8	Legales y derecho de autor.	8.1	Legales y derecho de autor	0
	9	Estándares aplicables	9.1	NC-ISO/IEC90000:2006 NC-ISO/IEC12119:2005	-
Total					**108**

El requisito de ADECUACIÓN refiere la capacidad del producto para proporcionar el conjunto de funciones apropiadas para ejecutar las tareas de los usuarios. Se incluyeron nuevos requisitos que no disponían en la versión anterior, referido a los dispositivos sobre tecnología Arduino. Se integró la exportación del Mapa MODBUS con el SCADA EROS.

El producto posee la capacidad de ser entendido, aprendido, usado y resultar atractivo para el usuario (USABILIDAD). Es intuitivo, permite al usuario reconocer la adecuación del software para ejecutar las operaciones a través de los menús; es sencillo y amigable, lo que facilita aprender su aplicación y operarlo. La estética de la interfaz es adecuada y coherente con sus funcionalidades, capaz de agradar y satisfacer la interacción con el usuario.

El software incluye protección contra errores de usuario, como es confirmar la operación cumpliendo con los requisitos de SEGURIDAD DE LOS DATOS y FIABILIDAD.

Con respecto al requisito de mantenibilidad, en la División se prevén actividades de gestión de la configuración tales como: registro control de versiones, registros de mejoras, control de cambios, identificación de las recomendaciones para nuevas versiones, identificación de los datos registrados durante la operación. Además, analiza la capacidad de implementación de las modificaciones de las interfaces que puedan requerirse cuando se hacen adiciones o cambios al sistema en cuanto utilizando la variable tiempo.

Por último, el cumplimiento del requisito no funcional orientado al usuario, "documentación y soporte de ayuda" y derechos Legales y derecho de autor, están en el proceso de elaboración, aún no se ha inscrito en el "Registro de productores de software" ni está asociado a una marca que lo identifique en cuanto al logotipo e isotipo.

2.5.3 Análisis valorativo de los aspectos económicos y medioambientales.

Se desarrolló un Sistema Informático para la Programación de Autómatas Programables sobre tecnología Arduino.

Se utilizó en la programación de un Concentrador Analógico que se diseñó y fabricó en la División de Automatización, y se implementó en el cárnico de Santiago de Cuba, Cárnico de Palma Soriano y en el Lácteo de Ciego de Ávila. Mediante dicho concentrador, las mediciones de temperatura se enviaron por protocolo Modbus al SCADA EROS para la supervisión y monitorio.

Actualmente en la División se trabaja en el diseño una serie de autómatas con tecnología Arduino; además, sobre la base de las experiencias en la línea de desarrollo de programadoras de autómatas, se encuentra en etapa conceptualización el EROSCODE31, una versión moderna y superior que va integrar el banco de trabajo con la norma de los lenguajes de programación IEC61131.

Conclusiones

1. Se realizó la conceptualización y diseño de un Sistema Informático para la Programación de Autómatas Programables sobre tecnología Arduino.
2. La evaluación del producto fue favorable, por lo cual se implementó en la programación de la tarjeta específica Concentrador Analógico, para empresas del sector cárnico y lácteo.

CONCLUSIONES

Se desarrolló un Sistema Informático para la Programación de Autómatas Programables sobre tecnología Arduino, sobre la base de la base de la experiencia de la Programadora de Autómatas EROSPG.

El Sistema se evaluó utilizando el procedimiento DA-PE-003 de SERCONI, basado en las normas ISO/IEC 9126, ISO/IEC 14598 y ISO/IEC 25000, resultando Conforme con evaluación de 4,3 puntos, y se implementó en la programación de la tarjeta específica Concentrador Analógico, que se comercializó en las empresas del sector cárnico y lácteo.

RECOMENDACIONES

Desarrollar la programadora de autómatas EROSCODE31, versión moderna y superior que integra el banco de trabajo con la norma de los lenguajes de programación IEC61131, y sea compatible con los autómatas de diversas tecnologías que se desarrollan en la División.

BIBLIOGRAFÍA

[1] I. Metalmecánica. "Proveedores de Controles Lógicos Programables (PLC)." https://www.metalmecanica.com/guia-de-proveedores/Controles-logicos-programables-PLC+8190302 (accessed 15/02/2021.

[2] M. Tiegelkamp and K.-H. John, *IEC 61131-3: Programming industrial automation systems*. Springer, 1995.

[3] "What is Arduino?" https://www.arduino.cc/en/Guide/Introduction (accessed 15/02/2021.

[4] "GEB Automation version Student," 2 Junio 2020. [Online]. Available: https://www.gebautomation.com/student.html.

[5] G. Castro, "Las principales tendencias tecnológicas para la Automatización Industrial," 2017. [Online]. Available: http://www.emb.cl/electroindustria/articulo.mvc?xid=3146&ni=las-principales-tendencias-tecnologicas-para-la-automatizacion-industrial.

[6] "Arduino Mega 2560 Rev3." https://store.arduino.cc/usa/mega-2560-r3 (accessed 15/02/2021.

[7] J. A. Polanco, "Análisis comparativo de los lenguajes de programación de PLC definidos en la norma IEC 61131-3," 2019.

[8] E. Unicrom. "Historia del PLC, Modicon, Modbus." https://unicrom.com/historia-del-plc-modicon-modbus/ (accessed 15/02/2021.

[9] Wikipedia. "Dick Morley - Inventor del PLC." https://es.qaz.wiki/wiki/Dick_Morley (accessed 15/02/2021.

[10] "Softwares de autómata programable," no. 15/02/2021, 2021. [Online]. Available: https://www.directindustry.es/fabricante-industrial/software-automata-programable-109421.html.

[11] https://www.arduino.cc/ (accessed 15/02/2021.

[12] A. Istanbullu and M. Taşçı, "Open source hardware-Arduino: Case study on mechanical engineering students design project," 2019.

[13] J. C. Quezada, E. Flores, A. E. Solís, and V. Quezada, "IEC-61131 Controladores Programables," *Boletín Científico INVESTIGIUM de la Escuela Superior de Tizayuca,* vol. 1, no. 2, 01/05 2016, doi: 10.29057/est.v1i2.1698.

[14] S. Cavalieri and M. Salafia, "Asset Administration Shell for PLC representation based on IEC 61131-3," *IEEE Access,* vol. PP, pp. 1-1, 08/03 2020, doi: 10.1109/ACCESS.2020.3013890.

[15] E. Sothis, "Qué es un PLC y el protocolo más utilizado," 18 Junio 2019. [Online]. Available: https://www.sothis.tech/plc-dispositivo-electronico-o-programmable-logic-controller/.

[16] I. Mecafenix. "Lenguajes para programación de plc." https://www.ingmecafenix.com/automatizacion/lenguajes-programacion-plc/ (accessed 26/02/2021.

[17] A. S. De Castro Rivera, F. A. Canchila Aguirre, and A. A. Anaya Pineda, "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," 2017.
[18] N. I. CORP, "The Modbus Protocol In-Depth," 5 Febrero 2021 2021. [Online]. Available: https://www.ni.com/en-us/innovations/white-papers/14/the-modbus-protocol-in-depth.html.
[19] J. M. M. Rodríguez, "Implementación de una plataforma de automatización de bajo coste," p. 83, 2017.
[20] "Siemens CPU 1212C," no. 15/02/2021. [Online]. Available: https://www.automation24.biz/siemens-cpu-1212c-6es7212-1ae40-0xb0.
[21] Rugged-Circuits. "Ruggeduino-SE "Special Edition"." https://www.rugged-circuits.com/microcontroller-boards/ruggeduino-se-special-edition (accessed 15/02/2021.
[22] "Industrino," no. 15/02/2021, 2018. [Online]. Available: https://industruino.com/.
[23] "The GNU C Library (glibc)." http://www.gnu.org/savannah-checkouts/gnu/libc/libc.html (accessed 15/02/2021.
[24] "LOGI.CAD 3 FBD-LIGHT License - online shop." https://revolution.kunbus.de/shop/en/logicad-3-fbd-light (accessed 15/02/2021.
[25] M. Software. "Visuino - Visual Development for Arduino." https://www.visuino.com/ (accessed.
[26] "Embrio - A visual, real time development tool for the Arduino," no. 15/02/2021, 2020. [Online]. Available: https://www.embrio.io/.
[27] Labview. "Labview." https://www.ni.com/en-us/shop/labview.html (accessed.
[28] L. Gallego Olivares, "ESTUDIO E IMPLEMENTACION DE UN SISTEMA DE DETECCION DE CAIDAS MEDIANTE EL USO DE UN ACELEROMETRO," 2020.
[29] M. Rouin Jarjour, "Sistema de supervisión low-cost con algoritmo de diagnóstico predictivo de puntos calientes en paneles fotovoltaicos," 2021.
[30] F. García Reig, "Diseño y fabricación de un brazo robótico de 6 GDL de bajo coste basado en Arduino," 2020.
[31] S. T. Sánchez, "Programación de Laboratorios de Biología Portátiles Abiertos Basados en Arduino con el Lenguaje de Programación Visual XOD," 2020.
[32] E. D. R. Castrillo, J. J. G. Pabón, and J. R. B. Santaella, "IMPLEMENTACIÓN DEL SISTEMA ELÉCTRICO-ELECTRÓNICO Y SISTEMA DE SOFTWARE DE UNA MÁQUINA CNC," *REVISTA COLOMBIANA DE TECNOLOGIAS DE AVANZADA,* vol. 2, no. 36, pp. 18-26, 2020.
[33] J. L. H. Villar, J. O. García, and S. Y. M. Hiyo, "Diseño y construcción de un sistema automatizado de control de bombas de agua en un cultivo hidropónico en el entorno Arduino, UNSCH–Ayacucho," *Revista ECIPerú Volumen,* vol. 15, no. 1, 2020.
[34] E. H. Orallo, " El Lenguaje Unificado de Modelado (UML)."
[35] F. M. Alfaro, "Herramientas CASE," 1999.
[36] "Visual Paradigm. Sitio oficial." http://www.visual-paradigm.com/ (accessed.
[37] "Observatorio Tecnológico. Introducción a la programación con el lenguaje C," no. 5 de Diciembre de 2019. [Online]. Available: http://recursostic.educacion.es/observatorio/web/ca/software/programacion/745-introduccion-a-la-programacion-con-el-lenguaje-c.

[38] "Aprende a programar. ¿Qué es y para qué sirve C?," no. 7 de Enero de 2020. [Online]. Available: https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=894:ique-es-y-para-que-sirve-c-uso-en-sistemas-operativos-unix-compiladores-familia-lenguajes-c-cu00505f&catid=82&Itemid=210.

[39] "Qué es C++: Características y aplicaciones," no. 15 de Septiembre de 2019. [Online]. Available: https://openwebinars.net/blog/que-es-cpp/. OpenWebinars.

[40] "Lenguajes de programación. C++," no. 15 de Septiembre de 2019. [Online]. Available: https://lenguajesdeprogramacion.net/cpp/.

[41] V. Bidone, M. Maggi, T. Bandera, S. Roccatagliata, and E. Gomez, "C++: Un lenguaje base."

[42] A. G. Carrillo, *Fundamentos de programación en C++*. Delta Publicaciones, 2005.

[43] "El concepto de IDE," 29 de enero de 2020. [Online]. Available: https://www.redhat.com/es/topics/middleware/what-is-ide. RetHat Sitio Oficial.

[44] "Atmel® Studio 7," 20 de Agosto 2020. [Online]. Available: https://microchipdeveloper.com/atstudio:studio7intro.

[45] "Embarcadero Sitio Oficial. C++Builder Community Edition," 15 de Septiembre de 2020. [Online]. Available: https://www.embarcadero.com/es/products/cbuilder/starter.

ANEXOS

ANEXO 1

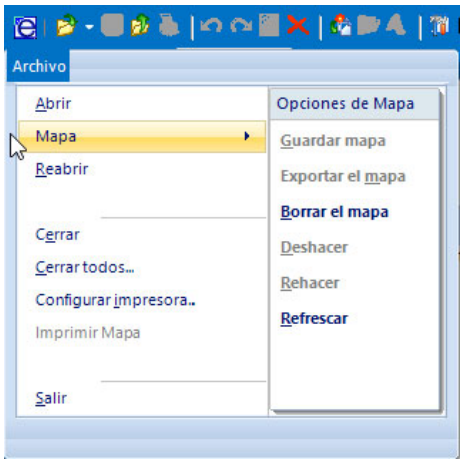

Figura A.1: Menú Principal

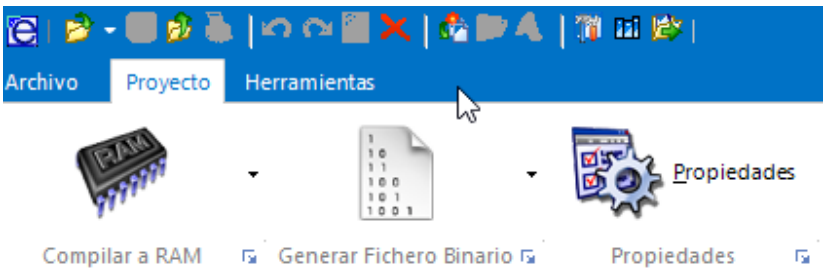

Figura A.2: Barra de Herramientas de Acceso Rápido

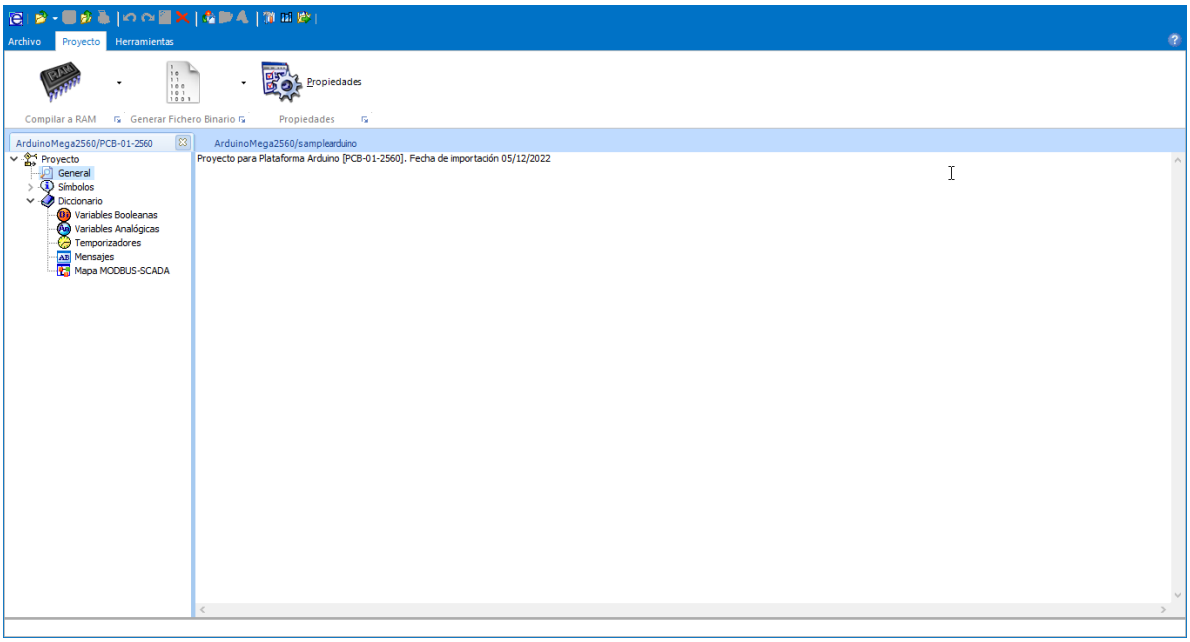

Figura A.3: Ventana de varios Proyectos Abiertos

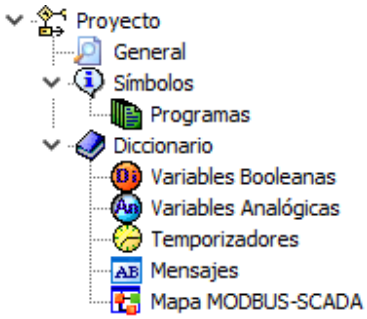

Figura A.4: Estructura de un Proyecto

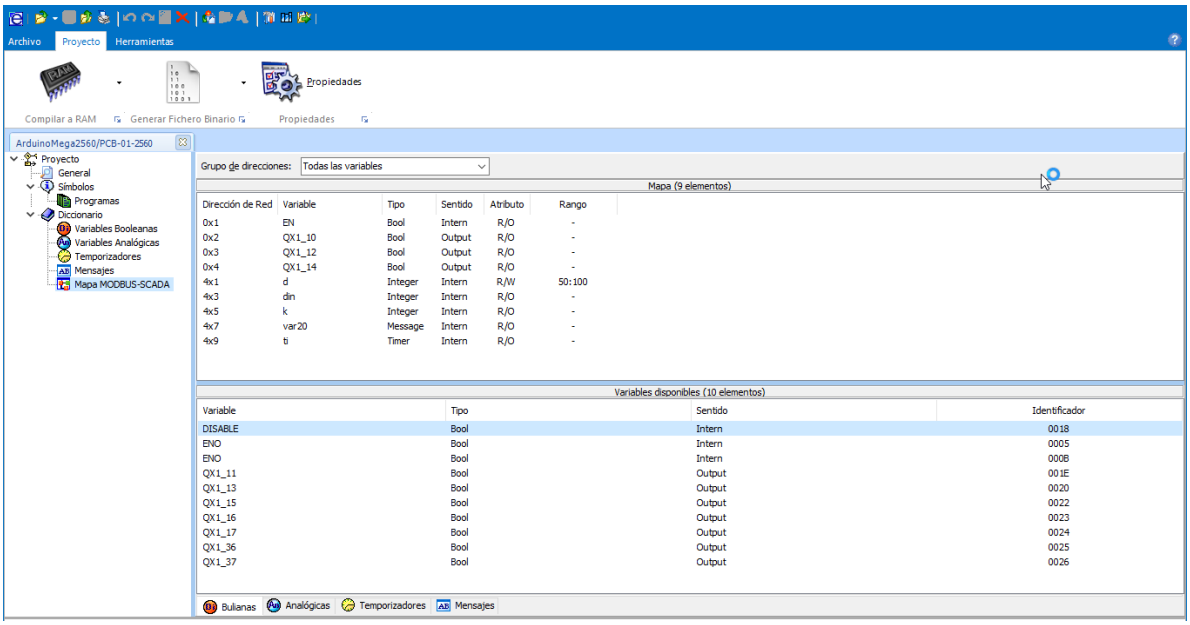

Figura A.5: Variables del Mapa Modbus