Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura.

Implementación de un sistema de automatización para pruebas de campo del rack DeltaV S.I.S.

Paulo Atán Cárdenas

200752224

Cartago, Junio de 2013.

TEC | Tecnológico de Costa Rica

INSTITUTO TECNOLÓGICO DE COSTA RICA
ESCUELA DE INGENIERIA ELECTRÓNICA

PROYECTO DE GRADUACIÓN

TRIBUNAL EVALUADOR

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Ing. Marvin Hernández Cisneros
Profesor asesor

Ing. Juan Carlos Jiménez Robles
Profesor lector

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica

Cartago, junio 25, 2013

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía, he procedido a indicar las fuentes mediante las respectivas citas bibliográficas.

En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Cartago, 25 de Junio de 2013.

Lugar, fecha

Paulo Atán C.

Firma del autor

Paulo Atán Cárdenas.

Nombre completo del autor

Céd: *503690738*

RESUMEN

Este trabajo surge de la necesidad de la empresa Emerson Electric de contar con un sistema que realice pruebas de campo que, determinen el funcionamiento adecuado del equipo destinado a prevenir fallos en los procesos de control de las entidades clientes de dicha compañía.

Hasta el momento, la empresa ha contado con métodos de prueba realizados a través del software de programación DeltaV y una serie de componentes analógicos, que ocasionan que se emplee largos periodos de tiempo para confirmar la operación correcta del sistema DeltaV S.I.S.

En consecuencia, se desarrolló un sistema de automatización basado en la plataforma Arduino, capaz de generar y recibir señales eléctricas que permitieran poner a revisión la funcionalidad de los modos de operación del DeltaV Logic Solver en una cantidad de tiempo menor al invertido en las pruebas originales, y que evitara la utilización y conexión de diferentes componentes al equipo.

Este sistema contó con comunicación vía protocolo Ethernet, con el fin de que cualquier computadora perteneciente a la red de la empresa tuviera acceso a la configuración y prueba del rack; por medio de una interfaz de usuario ejecutada en un navegador web.

Palabras clave**:** *DeltaV, Automatización, Logic Solver, Ethernet, Interfaz de usuario, S.I.S., Arduino, Emerson Electric*.

ABSTRACT

This work arises from the need of Emerson Electric to have a field tests system, which determines the proper operation of equipment designed to prevent control process failures for client entities of the company.

So far, the company has had test methods performed through the DeltaV programming software and a number of analog components, which caused long periods of time in the confirmation of the correct operation of the DeltaV SIS system.

Consequently, I developed an automation system based on the Arduino platform, capable of generating and receiving electrical signals that putted to revision the functionality of DeltaV Logic Solver operating modes, on a shorter amount of time than the invested in the original tests.

This system had communication via Ethernet protocol, so that any computer within the network of the company had access to the rack configuration and testing, through a user interface running on a web browser.

Keywords: *DeltaV, Automation, Logic Solver, Ethernet, User Interface, S.I.S., Arduino, Emerson Electric.*

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

1. Introducción

Actualmente los procesos automatizados son parte vital de la producción industrial, cualquier falla en los controladores del proceso tendría como consecuencia una grave pérdida para la empresa a nivel económico y podría poner en peligro la vida de los trabajadores. Esto implica la necesidad de implementar sistemas computacionales con el fin de reducir riesgos en caso de un fallo en el control del proceso, así como la capacitad de monitorización del estado de los diferentes equipos necesarios para realizarlo. Un sistema instrumentado de seguridad (SIS) consiste en un conjunto de software y hardware diseñado para ser utilizado en sistemas de control crítico de procesos, se le llama sistema instrumentado debido a que existe una serie de instrumentos que forman parte del medio, los cuales son comandados por un control central. Un proceso crítico puede ser identificado como aquel en el que una vez en funcionamiento, si ocurre un problema de operación, el sistema debería entrar en un “estado seguro” para procurar la seguridad y evitar consecuencias a la salud o el ambiente.

En muchas ocasiones el cliente desea tener la posibilidad de que se realice una demostración del sistema diseñado por Emerson, para lo que la compañía presenta dos alternativas: realizar una simulación del sistema por medio de una herramienta de software con el fin de monitorear las respuestas del sistema de acuerdo a estímulos realizados en tiempo real, y realizar las conexiones físicas en el sistema y realizar dicha comprobación por medio de contactores, fuentes analógicas de corriente y luces de indicación.

Debido a que la primera opción conlleva un alto grado de complejidad ingenieril para el cliente y demanda gran cantidad de tiempo; surge la necesidad de implementar un sistema de simulación integral para los Logic Solver Delta V SIS, apropiado para los diferentes posibles escenarios de demostración funcional que demande el cliente.

Dicha aplicación fue basada en la idea de crear un sistema implementado en un lenguaje y plataforma libre llamado Arduino, esto debido a que posee las ventajas de una amplia disponibilidad de módulos y librerías, es eficiente, flexible, fácil de usar y posee licencia gratis. Este microcontrolador se programa mediante el lenguaje de programación y entorno de desarrollo Arduino, los cuales se obtienen de forma gratuita en el sitio oficial.

2. Meta y Objetivos.

2.1. Meta.

Disminuir el tiempo de ejecución de las demostraciones funcionales de los Delta V SIS, por medio de la reducción de las tareas que realiza el usuario.

2.2. Objetivo General.

- Implementar un sistema autónomo para la simulación y demostración en tiempo real de variables físicas externas en Logic Solvers Delta V SIS.

2.3. Objetivos Específicos.

- Diseñar un sistema capaz de realizar peticiones para generar y capturar datos ya sean, digitales o analógicos con protocolo HART para ser enviados por la terminal del puerto 80.

- Diseñar una etapa de acople de datos para establecer una interacción directa con las Entradas/Salidas de los Logic Solvers Delta V SIS.

- Desarrollar una interfaz de software apropiada para la interacción del usuario con el sistema.

- Implementar una comunicación Ethernet entre el sistema de control y el rack, con el fin de integrar un módulo unificado para pruebas a la red local de la empresa.

3. Marco Teórico.

3.1. Sistema Instrumentado de Seguridad (SIS) DeltaV Logic Solver.

Un SIS tiene como función, realizar tareas específicas de control cuando ocurre algún fallo o para mantener la operación segura de un proceso cuando una condición inaceptable o peligrosa ocurre. Los SIS deben ser independientes de otros sistemas de control que controlan el mismo equipo para asegurar que el funcionamiento no esté comprometido [3]. Un SIS se compone de los mismos tipos de elementos de control (incluidos los sensores, solucionadores lógicos, actuadores y otros equipos de control) de un sistema de proceso de control básico (BPCS), sin embargo todos los elementos de control en un SIS están dedicados exclusivamente para el funcionamiento adecuado del SIS.

Figura 1. Rack de S.I.S. DeltaV Logic Solvers.

El DeltaV Logic Solver es un SIS utilizado por la empresa Emerson para el desarrollo de sistemas de control automático distribuido de acuerdo con la necesidad del cliente. El logic solver DeltaV SIS cuenta con una arquitectura modular única basada en SLS (Smart Logic Solver). Sus canales entrada/salida son configurables por medio de software permitiendo flexibilidad para la implementación de funciones instrumentadas de seguridad, en donde por cada Logic Solver se pueden configurar 15 señales entrada/salida ya sea como entrada

discreta (DI), salida discreta (DO), entrada analógica (AI) o salida de dos estados con protocolo HART (AO).

A continuación se muestran las formas de conexión y operación de los canales del DeltaV logic solver [4]:

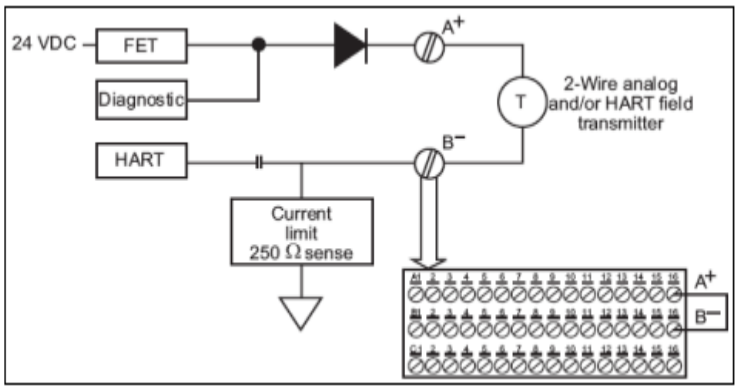


Figura 2. Diagrama de cableado para una entrada analógica.

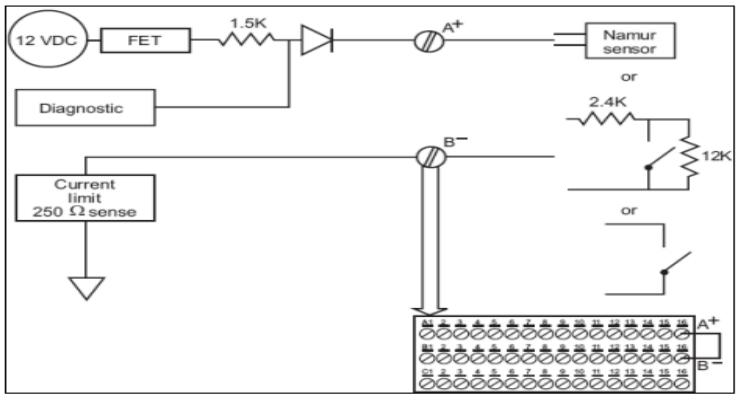


Figura 3. Diagrama de cableado para una entrada discreta.

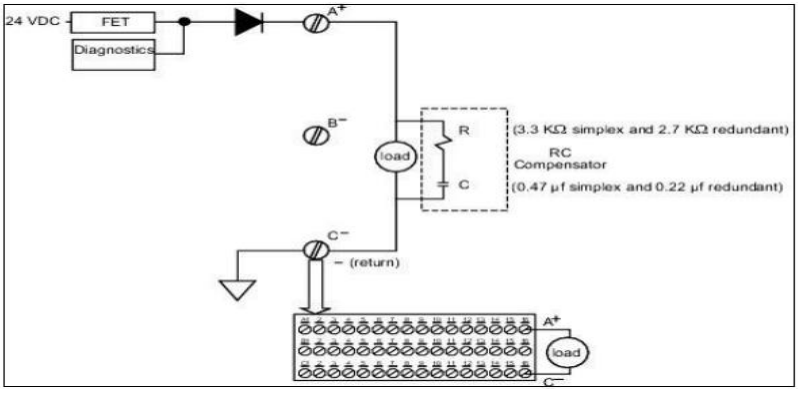


Figura 4. Diagrama de cableado para una salida discreta.

A su vez, este sistema utiliza inteligencia predictiva para aumentar la disponibilidad de la totalidad del sistema instrumentado de seguridad, esto le ayuda al usuario a proteger sus activos de forma fiable y mejorar el rendimiento de su planta. Los estándares de seguridad insisten en que los sistemas de control y seguridad deben de ser implementados de forma separada, esto con el fin de remover cualquier posibilidad de un fallo común afectando ambas capas de protección. Por otro lado, los usuarios finales necesitan una plataforma que integre sistema de configuración, mantenimiento y operación. El sistema DeltaV SIS presenta una solución única a este problema; implementando funciones de seguridad con hardware dedicado, software y redes manteniéndose de forma inadvertida integrado en las estaciones de trabajo.

3.2 Software DeltaV 11.3.

Es el programa utilizado para construir los sistemas de control que gobernarán los Logic Solver. Para usarlo en una computadora se necesita una licencia otorgada por la empresa (Dongle), así como la ejecución de dicho software dentro de una máquina virtual, en este caso: VMware Player.

En este simulador (Apéndice A.4) se pueden establecer bloques en los cuales se configuran los modos de funcionamiento y valor de cada canal: ya sea como entrada o salida y como modo analógico o discreto respectivamente. Además, detecta los valores que se envían a cada canal del Logic Solver y los imprime en pantalla [6].

A continuación se muestra el ambiente que presenta el software cuando está siendo utilizado.

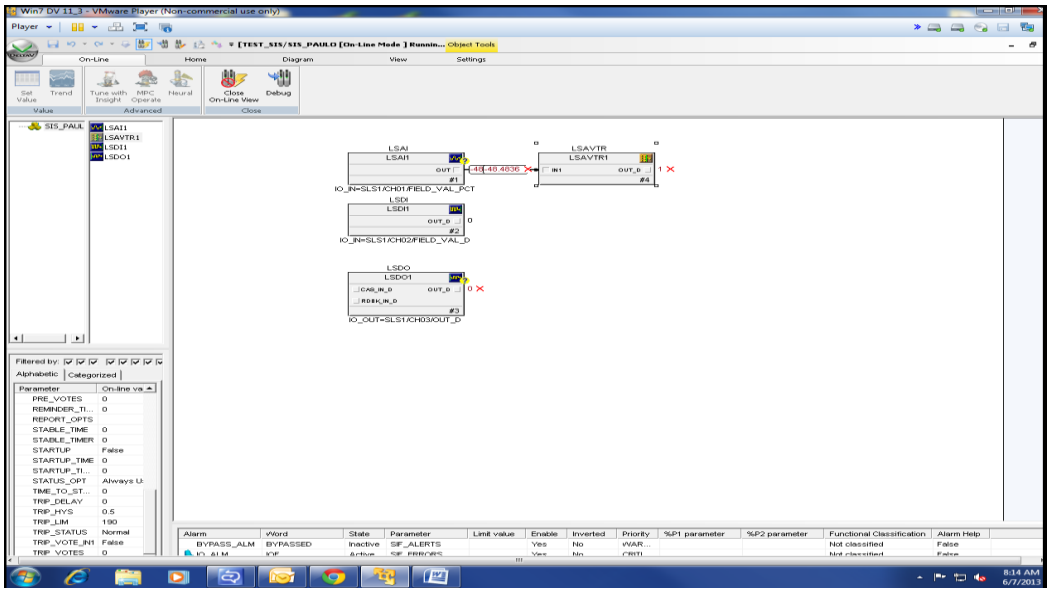


Figura 5. Interfaz del DeltaV 11.3.

3.3. Arduino Mega 2560 R3.

El Arduino Mega es una placa microcontrolador basada ATmega2560. Tiene 54 entradas/salidas digitales de las cuales 15 proporcionan salida PWM, 16 entradas analogicas, 4 UARTS (puertos serie por hardware), un cristal oscilador de 16MHz, conexión USB, entrada de corriente, conector ICSP y botón de Reset. Puede ser alimentado vía la conexión USB o con una fuente de alimentación externa.

Las fuentes de alimentación externas pueden ser tanto un transformador o una batería, el transformador se conecta usando un conector macho de 2.1mm con centro positivo en el conector hembra de la placa, los cables de la batería se conectan a los pines Gnd y Vin en los conectores de alimentación. La placa puede trabajar con una alimentación externa de entre 6 a 20 voltios, si la tensión suministrada es inferior a 7V, el pin de 5V puede proporcionar menos de 5 Voltios y la placa puede volverse inestable, si se usan más de 12V los reguladores de tensión se pueden sobrecalentar y dañar la placa. El rango recomendado es de 7 a 12 V.

Los pines de alimentación son los siguientes:

• **Vin:** La entrada de tensión a la placa Arduino cuando se está usando una fuente externa de alimentación. Se puede proporcionar tensión a través de este pin, o, si se está alimentado a través de la conexión de 2.1mm, acceder a ella a través de este pin.

• **5V:** La fuente de tensión usada para alimentar el microcontrolador y otros componentes de la placa. Esta puede provenir de Vin a través de un regulador integrado en la placa, o proporcionada directamente por el USB u otra fuente estabilizada de 5V.

• **3,3V:** Una fuente de tensión a 3.3 voltios generada por el regulador de la placa. La corriente máxima soportada es de 50mA.

• **GND:** Pines de toma a tierra.

El ATmega2560 tiene 256KB de memoria flash para almacenar código, de los cuales 8Kb son usados para el arranque del sistema (bootloader), 8 Kb de memoria SRAM, 4KB de EEPROM que se pueden acceder para leer o escribir con la librería EEPROM.

Cada uno de los 54 pines digitales pueden utilizarse como entradas o como salidas que operan a 5 voltios, usando las funciones pinMode(), digitalWrite(), y digitalRead(), donde el pin puede proporcionar o recibir una intensidad máxima de 40mA y tiene una resistencia interna (desconectada por defecto) de 20-50kOhms.

Además, algunos pines tienen funciones especializadas:

• **Serie 0: 0 (RX) y 1 (TX), Serie 1: 19 (RX) y 18 (TX); Serie 2: 17 (RX) y 16**

(TX); Serie 3: 15 (RX) y 14 (TX): Usados para recibir (RX) y transmitir (TX) datos a través del puerto serie TTL.

• **Interrupciones Externas: 2 (Interrupción 0), 3 (Interrupción 1), 18 (Interrupción 5), 19 (Interrupción 4), 20 (Interrupción 3), y 21 (Interrupción 2)**:

Estos pines se pueden configurar para lanzar una interrupción en un valor LOW (0V), en flancos de subida o bajada (cambio de LOW a HIGH (5V) o viceversa), o en cambios de valor.

• **PWM:** de 2 a 13 y 44 a 46. Proporcionan una salida PWM (Pulse Width Modulation) de 8 bits de resolución (valores de 0 a 255) a través de la función analogWrite().

• **SPI: 50 (SS), 51 (MOSI), 52 (MISO), 53 (SCK):** Estos pines proporcionan comunicación SPI atreves de la librería SPI.

• **LED 13:** Hay un LED integrado en la placa conectado al pin digital 13, cuando este pin tiene un valor alto (5V), el LED se enciende y cuando este tiene un valor bajo (0V) se apaga.

• **TWI: 20 (SDA) y 21 (SCL):** Proporcionan comunicación TWI utilizando la librería Wire.

La placa tiene 16 entradas analógicas, y cada una de ellas proporciona una resolución de 10bits (1024 valores). Por defecto se mide de tierra a 5 voltios, aunque es posible cambiar la cota superior de este rango usando el pin AREF y la función analogReference ().

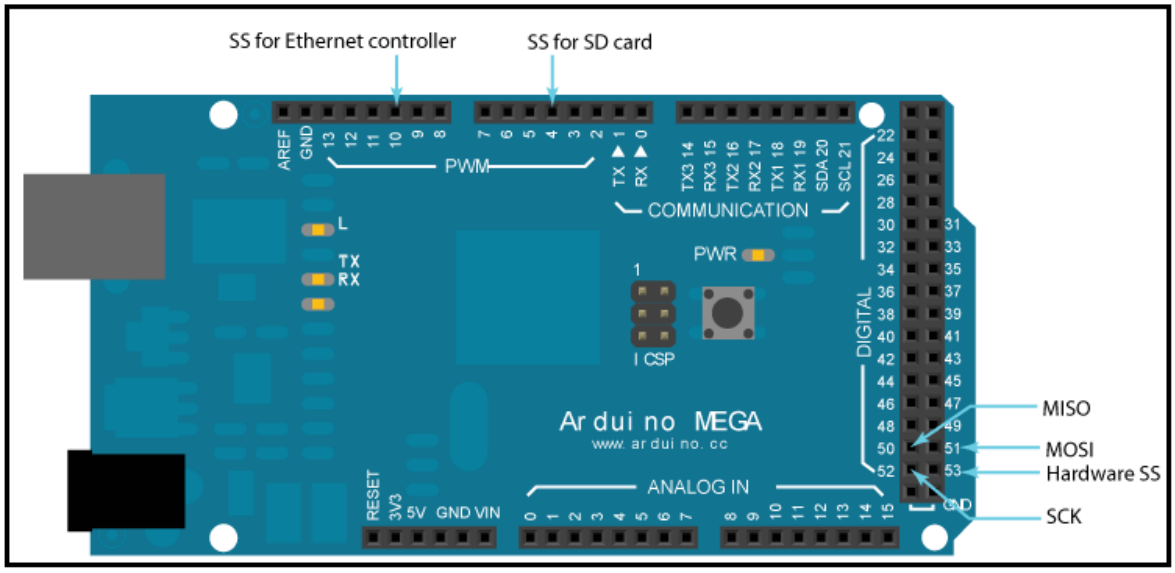


Figura 6. Descripción de pines del Arduino Mega R3.

3.4. Entorno de Desarrollo para Arduino (IDE).

El entorno de Desarrollo Arduino está constituido por un editor de texto para escribir el código, un área de mensajes, una consola de texto, una barra de herramientas con botones para las funciones comunes, y una serie de menús.

Además, permite la conexión con el hardware de Arduino para cargar los programas y comunicarse con ellos. Arduino utiliza para escribir el software lo que denomina "sketch" (programa), que son escritos en el editor de texto. En el área de mensajes se muestra información mientras se cargan los programas y también muestra errores. La consola muestra el texto de salida para el entorno de Arduino incluyendo los mensajes de error completos y otras informaciones. La barra de herramientas permite verificar el proceso de carga, creación, apertura y guardado de programas, y la monitorización serie.

Cuando se sube un "sketch", se está utilizando el "bootloader" de Arduino, un pequeño programa que ha sido cargado en el microcontrolador en su placa. Este permite la subida del código sin utilizar hardware adicional. El "bootloader" está activo durante unos segundos cuando la placa es reiniciada; después se inicia el "sketch" que fue actualizado en el microcontrolador. El "bootloader" produce un parpadeo en el LED de la placa (pin 13) cuando se inicia.

Las librerías proporcionan funcionalidad extra para la utilización de "sketches", por ejemplo: para trabajar con hardware adicional o manipular datos. Para utilizar una librería en un "sketch", se selecciona el menú Sketch > Import Library. Esto inserta una o más sentencias **#include** al principio del "sketch" y compila la librería con su "sketch". Debido a que las librerías se suben a la placa junto con el "sketch", se incrementa la ocupación del espacio disponible en memoria.

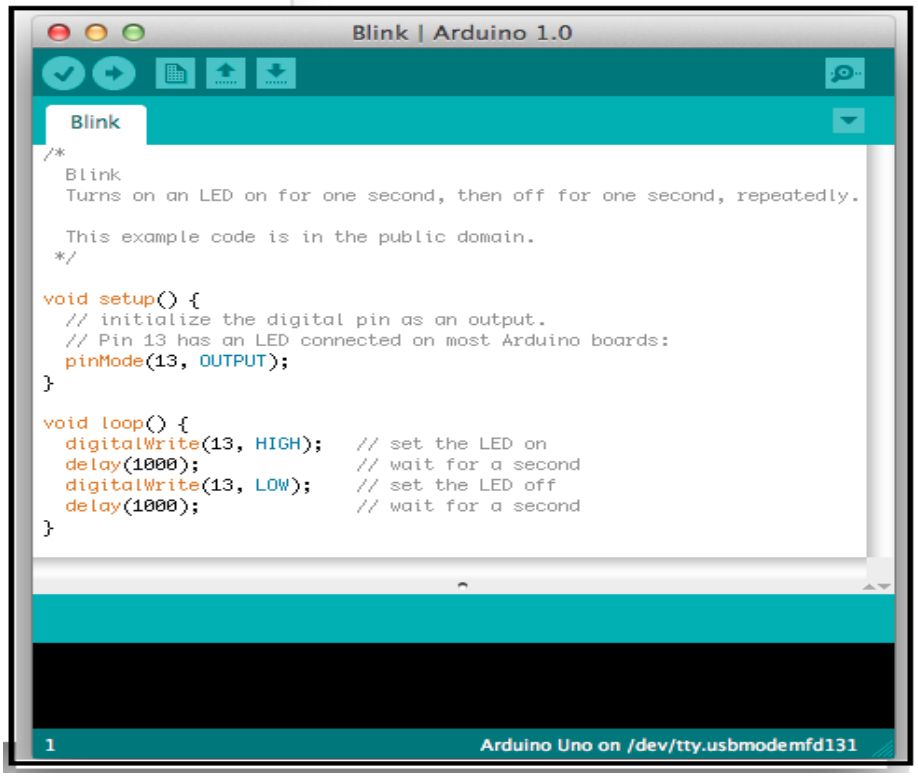


Figura 7. Entorno de desarrollo (IDE) de Arduino.

3.5. Escudo Ethernet para Arduino.

Este escudo permite a una placa Arduino conectarse a internet. El escudo está basado en el chip Ethernet Wiznet W5100, este provee de una pila de red IP capaz de TCP y UDP. Soporta hasta cuatro conexiones de sockets simultáneas y es compatible con las placas Arduino Uno y Mega. Con la librería Ethernet se escriben programas que logran realizar la conexión a internet usando el escudo.

Por medio de la librería se le asigna una dirección MAC e IP utilizando la función Ethernet.begin (). La dirección MAC es un identificador único globalmente para cada dispositivo en particular, esta dirección es incluida con el escudo en las versiones actuales.

También utilizando el protocolo DHCP se le asigna de forma dinámica una dirección IP al dispositivo y opcionalmente se establece una dirección de Gateway y de Subred. El escudo puede servir como un servidor aceptando conexiones entrantes o como un cliente realizando conexiones salientes. Esta librería logra soportar hasta 4 conexiones concurrentes en modo de servidor o cliente o ambas.

Por otra parte, posee un controlador de reset activado por medio de un botón en el escudo; el cual reinicia ambos, el W5100 y la placa Arduino mientras están encendidas.

El escudo contiene un número de LEDs para información:

• **PWR:** indica que la placa y el escudo están alimentadas

• **LINK:** indica la presencia de un enlace de red y parpadea cuando el escudo envía o recibe datos

• **FULLD:** indica que la conexión de red es full duplex

• **100M:** indica la presencia de una conexión de red de 100 Mb/s (de forma opuesta a una de 10Mb/s)

• **RX:** parpadea cuando el escudo recibe datos

• **TX:** parpadea cuando el escudo envía datos

• **COLL:** parpadea cuando se detectan colisiones en la red

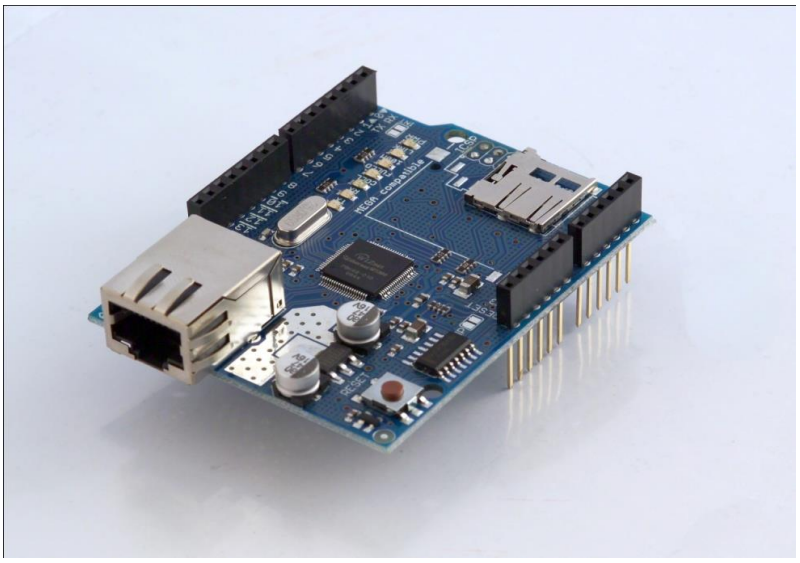


Figura 8. Escudo Ethernet Arduino.

3.6. Modelo TCP/IP

El modelo TCP/IP es la base de Internet, y sirve para comunicar todo tipo de dispositivos, computadoras que utilizan diferentes sistemas operativos, minicomputadoras y computadoras centrales sobre redes de área local (LAN) y área extensa (WAN). Describe un conjunto de guías generales de diseño e implementación de protocolos de red específicos para permitir que un equipo pueda comunicarse en una red. Provee conectividad de extremo a extremo especificando como los datos deberían ser formateados, direccionados, transmitidos, enrutados y recibidos por el destinatario. Este conjunto de protocolos es en referencia a los dos protocolos más importantes que la componen: Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP), que fueron dos de los primeros en definirse, y que son los más utilizados de la familia [10].

Existen tantos protocolos en este conjunto que llegan a ser más de 100 diferentes, entre ellos se encuentra el popular HTTP (HyperText Transfer Protocol), que es el que se utiliza para acceder a las páginas web. EL modelo TCP/IP está compuesto por cuatro capas o niveles, cada nivel se encarga de determinados aspectos de la comunicación y a su vez brinda un servicio específico a la capa superior. Estas capas son: Aplicación, Transporte, Internet y Enlace.

3.6.1. Capa de Aplicación

La capa de aplicación del modelo TCP/IP maneja protocolos de alto nivel, aspectos de representación, codificación y control de diálogo. El modelo TCP/IP

combina todos los aspectos relacionados con las aplicaciones en una sola capa y asegura que estos datos estén correctamente empaquetados antes de que pasen a la capa siguiente. TCP/IP incluye no sólo las especificaciones de Internet y de la capa de transporte, tales como IP y TCP, sino también las especificaciones para aplicaciones comunes, ya que tiene protocolos que soportan la transferencia de archivos, e-mail, y conexión remota, como los siguientes:

• **TELNET (Emulación de terminal):** Telnet tiene la capacidad de acceder de forma remota a otro computador. Permite que el usuario se conecte a un host de Internet y ejecute comandos. El cliente de Telnet recibe el nombre de host local. El servidor de Telnet recibe el nombre de host remoto.

• **HTTP: (protocolo de transferencia de hipertexto):** es el protocolo usado en cada transacción de la World Wide Web. Define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. A la información transmitida se la llama recurso y se la identifica mediante un localizador uniforme de recursos (URL). Una transacción HTTP está formada por un encabezado seguido, opcionalmente, por una línea en blanco y algún dato. El encabezado especificará cosas como la acción requerida del servidor, o el tipo de dato retornado, o el código de estado. Se define 8 métodos que indica la acción que desea que se efectúe sobre el recurso identificado: HEAD, GET, POST, PUT, DELETE, TRACE, OPTIONS y CONNECT.

3.6.2. Capa de Transporte.

La capa de transporte proporciona servicios de transporte desde el host origen hacia el host destino formando una conexión lógica entre los puntos finales de la red. Los protocolos de transporte segmentan y re ensamblan los datos mandados por las capas superiores en el mismo flujo de datos, o conexión lógica entre los extremos. Los servicios de transporte incluyen los siguientes servicios:

• **Modelo TCP (Protocolo de Control de Transmisión):** un protocolo de comunicación orientado a conexión, añade las funciones necesarias para prestar un servicio que permita que la comunicación entre dos sistemas se efectúe libre de errores, sin pérdidas y con seguridad. Este protocolo cuenta con las siguientes características:

i. Establecimiento de operaciones de punta a punta.

ii. Control de flujo proporcionado por ventanas deslizantes.

iii. Confiabilidad proporcionada por los números de secuencia y los acuses de recibo.

3.6.3. Capa de Internet

Esta capa tiene como propósito seleccionar la mejor ruta para enviar paquetes por la red. El protocolo principal que funciona en esta capa es el Protocolo de Internet (IP). La determinación de la mejor ruta y la conmutación de los paquetes ocurren en esta capa. Los protocolos que operan en la capa de internet son:

• **ICMP (Protocolo de mensajes de control en Internet):** suministra capacidades de control y envío de mensajes.

• **IP (Internet Protocol):** proporciona un enrutamiento de paquetes no orientado a conexión de máximo esfuerzo, o sea, no provee ningún mecanismo para determinar si un paquete alcanza o no su destino y únicamente proporciona seguridad de sus cabeceras y no de los datos transmitidos. El IP no se ve afectado por el contenido de los paquetes, sino que busca una ruta de hacia el destino. Las cabeceras IP contienen las direcciones de las máquinas de origen y destino, direcciones que serán usadas por los enrutadores para decidir el tramo de red por el que reenviarán los paquetes.

Funciones del Protocolo IP:

i. Define un paquete y un esquema de direccionamiento.

ii. Transfiere los datos entre la capa Internet y las capas de acceso de red.

iii. Enruta los paquetes hacia los hosts remotos.

3.6.4. Capa de Enlace.

También denominada capa de host de red. Esta es la capa que maneja todos los aspectos que un paquete IP requiere para efectuar un enlace físico real con los medios de la red, incluye los detalles de la tecnología LAN y WAN, define los procedimientos para realizar la interfaz con el hardware de la red y para tener acceso al medio de transmisión.

Son funciones de esta capa: la asignación de direcciones IP a las direcciones físicas, el encapsulamiento de los paquetes IP en tramas. Basándose en el tipo de hardware y la interfaz de la red, la capa de acceso de red definirá la conexión con los medios físicos de la misma.

3.7. El lenguaje HTML

HTML (HyperText Markup Language) es un lenguaje para definir la estructura de documentos para la elaboración de páginas web, que se utilizan para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos. La mayoría de los documentos tienen estructuras comunes (títulos, párrafos, lista.) este lenguaje permite definir la estructura mediante "tags" etiquetas rodeadas por corchetes angulares (<,>). Cualquier cosa que no sea una tag es parte del documento mismo. Este lenguaje no describe la apariencia de un documento sino que ofrece a cada plataforma la información para que le de formato según su capacidad y la de su navegador.

Los documentos escritos en HTML constan del texto mismo del documento y las "tags" que le dan formato. Consta de varios componentes vitales, entre ellos los elementos y sus atributos, tipos de datos y la declaración de tipo de documento.

3.7.1 Estructura del documento HTML

Los "tags" que describen la estructura general de un documento y dan una información sencilla sobre él son: <HTML> <HEAD> <TITLE> <BODY>. Estas tags no afectan a la apariencia del documento y solo interpretan y filtran los archivos HTML.

• **<HTML>:** Limitan el documento e indica que se encuentra escrito en este lenguaje.

• **<HEAD>:** Especifica el prólogo del resto del archivo. Son pocas las "tags" que van dentro de ella, destacando la del título. En head no hay que colocar nada del texto del documento.

• **<TITLE>:** Utilizado por los marcadores del navegador e identificará el contenido de la página. Solo puede haber un título por documento y no caben otras tags dentro de él.

• **<BODY>:** Encierra el cuerpo principal del documento, el contenido.

3.8 Protocolo I²C.

Se trata de, un bus bidireccional que utiliza dos líneas, una de datos serie (SDA) y otra de reloj serie (SCL), que requiere resistencias de polarización a positivo

(RPA). SCL se utiliza para sincronizar todos los datos SDA de las transferencias durante este protocolo.

Las líneas SCL y SDA están conectadas a todos los dispositivos en el bus I²C, y ambas líneas son del tipo drenador abierto; asociadas a un transistor de efecto de campo (o FET), es decir, un estado similar al de colector abierto. Esto significa que el chip puede manejar su salida a BAJO, pero no puede manejar a ALTO. Para que la línea pueda ir a ALTO, se deben proporcionar resistencias de polarización a 5V. Es decir, se necesita una resistencia de la línea SCL a la línea de 5V y otra de la línea SDA a la línea de 5V [11].

Sólo se necesita un conjunto de resistencias de RPA (pull-up) para todo el bus I²C, no son necesarias para cada dispositivo. La alimentación del sistema, debe tener una tierra común y también puede haber una alimentación compartida que, se distribuye entre los distintos dispositivos.

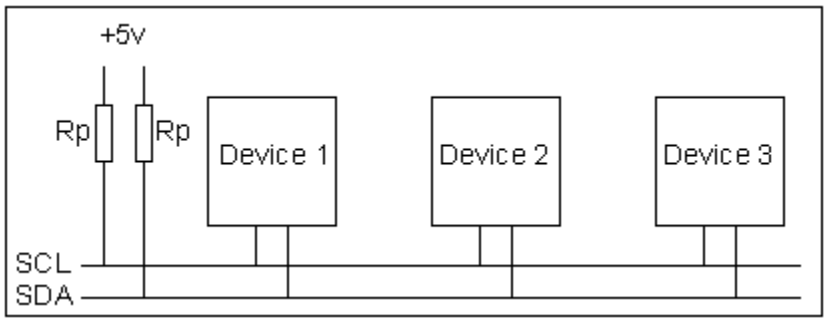


Figura 9. Diagrama de conexión del protocolo I2C.

Por otra parte, los dispositivos en este bus son maestros o esclavos. El maestro, es siempre el dispositivo que maneja la línea de reloj SCL. Los esclavos, son los dispositivos que responden al maestro. Un esclavo no puede iniciar una transferencia a través del bus, sólo un maestro puede hacer esa función. Generalmente son, varios esclavos en el bus, sin embargo, normalmente hay un solo maestro. Es posible tener varios maestros, pero es inusual y no se comentará aquí. Los esclavos, nunca inician una transferencia. Tanto el maestro, como el esclavo puede transferir datos a través del bus, pero la transferencia siempre es controlada por el maestro.

Todas las direcciones bus I²C son de 7 bits o 10 bits. Esto significa que, se pueden tener hasta 128 dispositivos en el bus. El protocolo I²C tiene un diseño de espacio de referencia de 7 bits de direcciones, reservado con 16 direcciones, de modo que finalmente, pueden comunicarse en el mismo bus un máximo de 112 nodos. El número máximo de nodos está limitado por el espacio de direcciones y también

por la capacidad total de los buses de 400 pF, lo que restringe la práctica de comunicación, a distancias de unos pocos metros.

Cuando se envía la dirección de 7 bits, se envía en realidad 8 bits. El bit extra (bit 8) se usa para informar al esclavo si el maestro está escribiendo o leyendo de él. Si el bit 8º es 0, el maestro está escribiendo al esclavo. Si el bit 8º es 1, el maestro está en la lectura del esclavo.

La estructura de la comunicación básica es la siguiente:

1. Inicio (Master).
2. 7 Bits de dirección de esclavo (Master).
3. 1 Bit de RW, 0 es Leer y 1 Escribir. (Master).
4. 1 Bit de Acknowledge (Slave).
5. Byte de dirección de memoria (Master).
6. 1 Bit de Acknowledge (Slave).
7. Byte de datos (Master/Slave (Escritura/Lectura)).
8. 1 Bit de Acknowledge (Slave/Master (Escritura/Lectura)).
9. Parada (Master).

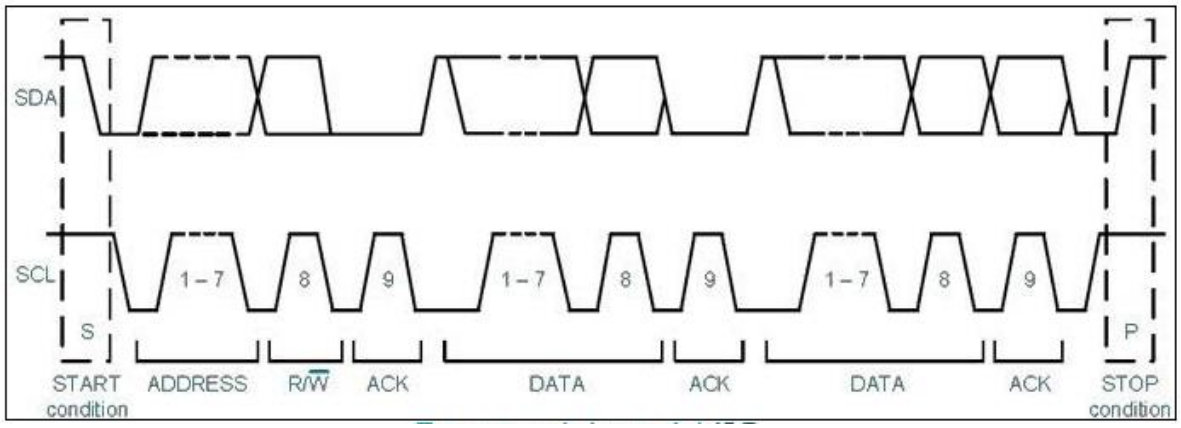


Figura 10. Descripción del protocolo I2C.

4. Procedimiento metodológico.

4.1 Reconocimiento y Definición del problema.

Debido a que en la empresa se realizan sistemas de control automático distribuido, un SIS no puede estar compuesto únicamente por una unidad de Logic Solver. Es por lo que la empresa cuenta con un Rack de Logic Solvers para su uso en pruebas físicas de dichos sistemas; esto tiene como consecuencia que la demostración utilizando conexiones para todo un sistema de seguridad pase por un lento proceso de revisión, necesitando de grandes espacios y tiempo de preparación.

Por otro lado, los ingenieros de la empresa llegan a la conclusión de que los sistemas de simulación por software actualmente utilizados son poco flexibles ya que para cada prueba se necesitan varias fuentes de alimentación, multímetros y una botonera; por lo que no es posible iniciar rutinas de demostración fáciles de implementar por el usuario a la hora de realizar pruebas de campo.

A continuación se muestra el sistema analógico y manual utilizado por la empresa en cada prueba:

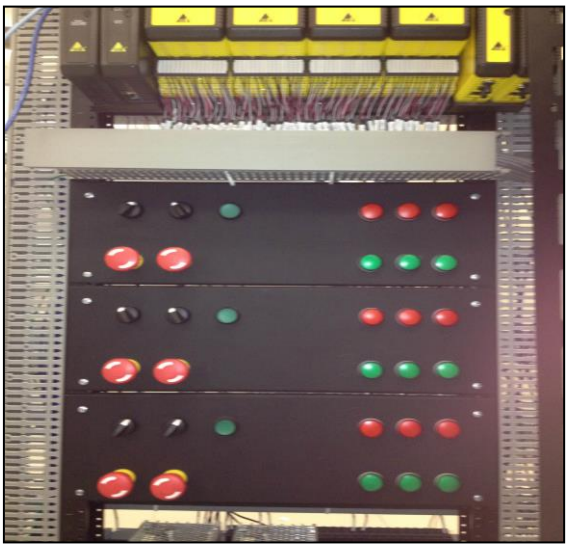


Figura 11. Botonera para pruebas de campo del Rack.

4.2. Obtención y análisis de la información.

La información acerca del rack DeltaV fue proporcionada por la empresa a través de hojas de datos, manuales de usuario y libros de entrenamiento para empleados. Para el análisis de aspectos técnicos nuevos y reafirmación de otros (protocolo HART, etc) se realizaron búsquedas en internet con el fin de tener claras las cualidades y posibles restricciones del sistema.

A partir de la información obtenida, se procedió a formular la idea general de la solución. Esta se basó en la compatibilidad con las características eléctricas (Protocolos HART y Ethernet, cantidad de puertos de entrada y salida) del sistema a mejorar, así como del precio de mercado de microcontroladores comúnmente utilizados para sistemas de automatización y, su flexibilidad y capacidad para crear interfaces amigables con el usuario.

4.3. Evaluación de las alternativas y síntesis de una solución.

Luego del proceso de investigación, se estableció la idea principal de la solución: implementar un sistema de simulación que integre herramientas físicas y de software; destinadas a la demostración de la funcionalidad del Delta V SIS, utilizando una unidad que sea capaz de generar o medir señales eléctricas con el fin de comprobar el funcionamiento en las terminales respectivas. De la misma manera era necesario implementar una interfaz de usuario con el fin de facilitar el uso del sistema planteado, que permitiera la versatilidad en la visualización y manipulación de posibles señales de estimulación.

Con el fin de realizar el proyecto cumpliendo con los objetivos necesarios para calificar con el proyecto de graduación de ingeniería electrónica; se pretendió realizar el sistema de simulación mediante la plataforma de hardware libre Arduino, haciendo uso de sus herramientas de software libre; con el fin de limitar el costo en el área de materiales por parte de la empresa únicamente al hardware, que ya por sí solo, es un microcontrolador barato y popular en el mercado.

De acuerdo con lo anteriormente planteado; a nivel de hardware fue necesario realizar un sistema de acople para medir y escribir señales en los puertos del Logic Solver, estas señales se debieron generar en base a opciones determinadas por un usuario en una terminal por medio de una comunicación entre el computador, el rack y el microcontrolador; donde este último debía interpretar estas señales y enviar o recibir información a los módulos de acople, con el fin de agilizar la demostración de los sistemas y poder visualizar el comportamiento en base a respuestas esperadas.

A nivel de software se necesitó la flexibilidad de escoger el tipo de entrada-salida asociada para un mismo puerto y el establecimiento de una comunicación por medio de una interfaz Ethernet con acceso al puerto 80 para configuración y operación remota, a lograrse a través de un modulo compatible con el microcontrolador, llamado Arduino Ethernet Shield.

Por otro lado, se debió realizar la implementación de rutinas de simulación por medio de la interfaz de usuario con el fin de facilitar el uso de los modos del Logic Solver. Para esto, se escogió el lenguaje HTML en lugar del comúnmente utilizado Visual Basic ya que posee alta compatibilidad con Arduino; gracias a su gran cantidad de librerías.

Para la etapa de acople entre el sistema de Logic Solvers y el sistema automatizado se barajaron 2 alternativas obtenidas del análisis de costo y diseño, en conjunto con la opinión del profesor asesor y del ingeniero supervisor: la primera consistió en la utilización de un sistema de microcontroladores funcionando en paralelo, además del uso de la fuente interna del Logic Solver, potenciómetros digitales y del protocolo S.P.I., así como la creación de tarjetas de circuito impreso (PCB) para empotrar los diferentes tipos de componentes a utilizar en esta etapa (chips, relés, resistencias, capacitores).

La segunda alternativa definió establecer una comunicación I2C entre los Arduinos y desarrollar un control de las salidas a base de PWM, direccionar la comunicación con el puerto 80 con un sólo Ethernet shield y utilizar seguidores y reguladores de tensión para el acople de sistemas, así como no utilizar la fuente interna antes mencionada y eliminar la implementación del PCB.

La segunda solución resultó escogida debido a que permitió abarcar múltiples temáticas en el área de ingeniería electrónica ya que implicó la comunicación entre diversos dispositivos, el desarrollo de sistemas empotrados y el diseño de hardware para cumplir con especificaciones de señales especificas [1].

4.4. Implementación de la solución.

Inicialmente, se contó con un Arduino Mega y un Ethernet shield para establecer la comunicación a través del puerto 80. Para esto, se procedió a realizar pruebas con las librerías para Ethernet del microcontrolador; en donde este funcionó como servidor web. De esa manera se verificó el envío y recepción de datos entre el Arduino y una computadora vía protocolo TCP/IP.

Luego, se procedió a desarrollar el control PWM que permitió medir en los pines con posibilidad de salida analógica una tensión variable de 0V a 5V, con una resolución máxima de hasta 256 pasos (Convertidor analógico-digital Arduino de 8 bits). Además, se consiguió escribir y leer datos de lógica positiva en cualquiera de los pines digitales del microcontrolador.

Después de conseguir la manipulación correcta de datos analógicos y discretos, se comenzó a desarrollar la interfaz de usuario por medio del lenguaje HTML. Para esto se tomó como base las librerías de Arduino, así como tutoriales encontrados en internet sobre la programación de página web [10]; en la cual inicialmente se establecieron botones que enviaban código que al ser detectado por el microcontrolador, este llevó a cabo la escritura o lectura de valores de tensión en el pin indicado por el mismo código.

Posteriormente, se inició a desarrollar la comunicación entre el sistema de automatización y el rack DeltaV SIS. Como el rack está compuesto por 12 Logic Solvers, inicialmente se pensó en implementar la primer opción de las 2 soluciones diseñadas; que consistió en 12 Arduinos, 12 shields y 12 IP's diferentes asignadas a la red local de Emerson Electric y así ejecutar en todos los microcontroladores el mismo programa, pero luego de un nuevo análisis de costos y de la red local, se llegó a la conclusión de llevar a cabo la segunda solución.

A razón de esto, se ideó establecer una comunicación maestro-esclavo para utilizar solo un Ethernet shield y por ende, una IP de la red local. Para lo anterior, se implementó la comunicación entre Arduinos por medio del protocolo I2C; en la cual, al Arduino maestro se le asignó el Shield para establecer comunicación vía puerto 80 con el usuario y a los 12 microcontroladores esclavos se les adjudicó la interacción directa con los Logic Solvers.

Posteriormente, se procedió a realizar el acople de impedancia para el modo analógico del Arduino. Esto se hizo mediante seguidores de tensión cuyas entradas de cada amplificador se conectaron a cada pin con capacidad PWM, y cada salida a cada entrada analógica del Logic Solver. También se implementó el uso de relés para cada entrada discreta y reguladores para cada salida discreta.

Después de comprobar el funcionamiento para un Logic Solver, se reprodujo el mismo procedimiento 3 veces más hasta poner en funcionamiento 3 Logic Solvers. Por último, se realizaron los últimos retoques a la interfaz de usuario para dar un toque estético a dicho programa.

Este proyecto se ha instalado en la sala de ingeniería de la empresa Emerson Electric y se dio a conocer sus resultados mediante una presentación a los ingenieros encargados del área de procesos, sistemas y soluciones (PSS).

Además este informe sirve como manual y guía para los empleados de la empresa que en un futuro interactúen con este sistema.

4.5 Reevaluación y rediseño.

Al pensar en una optimización del sistema automatizado diseñado, se pudo detectar algunos aspectos que podrían tener un diseño diferente; como lo son la etapa de acople y la interfaz de usuario.

En cuanto a la etapa de acople, se llegó a la conclusión de que se podría sustituir el circuito armado en la protoboard por un prototipo impreso (PCB). Esto con el fin de eliminar el cableado a base de conductores de cobre realizado entre los chips, las fuentes de alimentación y los reguladores de tensión.

Por otra parte, al no ser un concepto perteneciente a la ingeniería electrónica; la programación de la interfaz de usuario usando el lenguaje HTML se podría hacer de una manera más elegante o eficiente. Es decir, un experto en el tema podría realizar un ambiente aún más amigable y, que posea comandos de ejecución e interacción más avanzados que los empleados en este proyecto para manipular los datos de entrada o salida.

5. Descripción detallada de la solución.

5.1. Análisis de soluciones y selección final.

El proyecto consistió en realizar un entorno de simulación que fuera capaz de manejar un sistema periférico de medición y establecimiento de estímulos en base a comandos enviados desde una terminal por medio de una red Ethernet, para esto fue necesaria la creación de dicho sistema periférico; el cual debió cumplir con una serie de especificaciones técnicas dadas por el tipo de puerto y el tipo de señal que se estuviera generando o midiendo.

A continuación se enumeran los criterios de diseño necesarios que se siguieron para llevar a implementación física el sistema propuesto:

- Implementación de un sistema de acople para la etapa de potencia entre un microcontrolador y el rack, así como para la generación y medición de señales para el Delta V SIS (15 Entradas/Salidas).

- Implementación de un sistema capaz de recopilar y escribir información necesaria para la demostración. Este sistema se debe comunicar por medio de la red Ethernet local de la empresa al equipo del funcionario que desee realizar la comprobación funcional del DeltaV SIS.

- Diseño de una interfaz de usuario basada en el lenguaje HTML, que controle las pruebas del Delta V SIS mediante un equipo de computación externo, a través de una aplicación desplegada en un navegador web.

Para cumplir con tales lineamientos, se determinó las herramientas de hardware y software que permitieran desarrollar las 2 soluciones que siguieron el diagrama de bloques de la solución general:

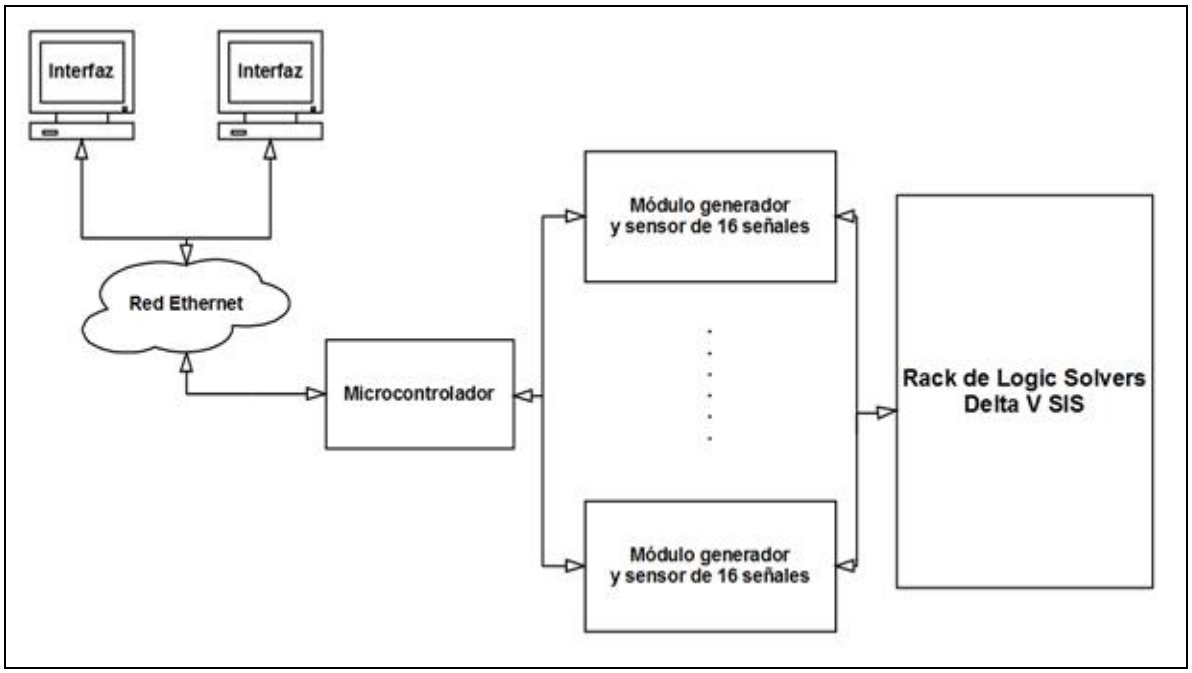


Figura 12. Esquema general del sistema planteado para realizar el sistema de automatización.

La primera solución consistió en un sistema de 12 microcontroladores conectados a la misma red local a través de 12 IP's diferentes y que ejecutaran en paralelo la misma rutina de programación. Es decir, cada arduino con su respectivo Ethernet shield controlaría los canales de un Logic Solver. Cabe destacar que esta alternativa fue desechada por el alto costo económico que

significa adquirir 12 Ethernet Shields, así como equipo adicional que permitiera agregar los Arduinos a la red local de la empresa (Hub, router, cable RJ-45).

La opción elegida como solución final consistió en usar un Arduino que tuviera interacción exclusiva con la interfaz de usuario por medio del Ethernet shield, y que funcionó como maestro con respecto a otros 12 microcontroladores del mismo tipo; por medio del protocolo I2C. A raíz de esto se desarrollaron 2 rutinas de programación, una para el maestro y otra para el esclavo, que a través de sus puertos de entrada-salida, fue el encargado de recibir y enviar las señales eléctricas del o hacia el Logic Solver.

Consecuentemente, se eligió la segunda alternativa debido a que el análisis de costos reflejó que esta opción era más barata, rápida y factible para conseguir los componentes a la hora de su fabricación; ya que el chip de potenciómetros digitales no se encuentra a la venta en Costa Rica. Además de que esta solución desde el punto de vista del criterio de diseño, significó el uso de menos componentes y la no implementación de un tercer protocolo de comunicación (SPI).El diagrama de la solución final se ilustra mediante la siguiente la figura:

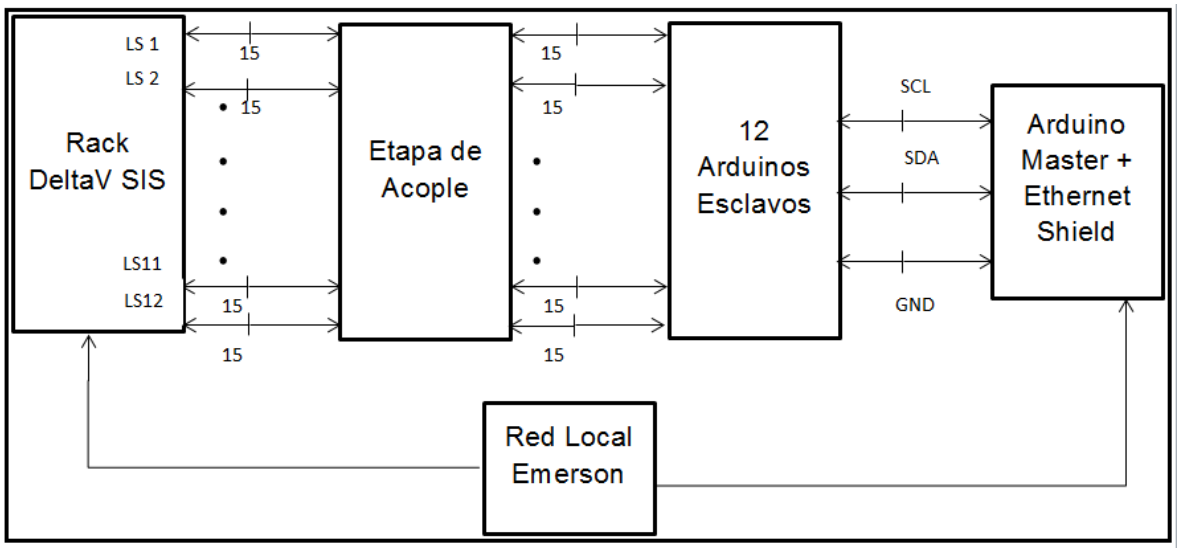


Figura 13. Diagrama modular de la solución final.

5.2. Descripción del Hardware.

Con respecto a la plataforma de hardware, se escogió Arduino debido a que es una tecnología de hardware y software libre [7] , por lo que existen gran cantidad de librerías para protocolos, gran cantidad de pines de entrada-salida, módulos de compatibilidad para aplicaciones de automatización industrial (Shields) y, su precio de adquisición es muy accesible (con respecto a las demás alternativas halladas en el mercado); lo que abarató significativamente el costo final del proyecto para la empresa.

En primer lugar, se utilizó el Arduino Mega 2560 REV 3 y el Ethernet Shield REV 3 ya que tienen una distribución de pines idéntica; por lo que para conectarlos solo se montó el shield sobre el Arduino. Con respecto a la fuente de alimentación de ese módulo, para las pruebas iniciales se usaron los puertos USB de una PC. En estas pruebas iniciales, se desarrolló la comunicación vía Ethernet entre el bloque Arduino-shield y una computadora perteneciente a la red la local de la empresa, en donde sus respectivos puertos se conectaron a través de una interfaz física RJ-45.

Luego, se procedió a realizar mediciones de parámetros eléctricos (niveles lógicos de tensión y señales analógicas, (Apéndice A.3) en los pines de entrada-salida del microcontrolador después de que se enviara un comando desde la PC usando la comunicación Ethernet. Esto con el fin de verificar el control del Arduino desde cualquier lugar perteneciente a la red local.

Por otra parte, como las señales analógicas que utiliza el DeltaV SIS se manipulan mediante porcentajes de rendimiento (0 a 100%) y siguen la forma de 4 a 20mA del protocolo HART; se necesitó implementar una señal que mantuviera las salidas del Arduino entre dicho rango. Para controlar la cantidad de corriente correspondiente a cada valor de porcentaje se hizo uso de la modulación de ancho de pulso (PWM), por medio del convertidor digital-analógico interno del Arduino; el cual tiene una resolución de 8 bits (hasta 256 valores de PWM) que se mapearon a un valor específico de tensión, en donde 5V fue el valor máximo y 1V el mínimo establecidos para generar los valores umbrales de corriente que estableció el protocolo HART.

Para este caso, se hizo uso de la resistencia interna que tiene cada canal del Logic Solver entre el punto B y C; cuyo valor medido fue de 256 Ω. Es decir, se intentó conectar en paralelo los pines del arduino que se destinaron a funciones analógicas con esta carga; para que la tensión cayera directamente de la

resistencia y así convertir la tensión en valores que se encontraran dentro del rango de corrientes predeterminado por Delta V Logic Solver (4 mA a 20 mA).

Sin embargo, la carga que significó la resistencia interna, se trajo abajo la tensión de los pines de salida del microcontrolador; por lo que tuvo que diseñarse un acople de impedancias [8]. Este consistió en conectar un seguidor de tensión entre cada pin del Arduino y el punto B de canal del DeltaV SIS, ya que esta configuración del amplificador operacional posee las características eléctricas necesarias para eliminar dicha falla; como lo son su impedancia de entrada y salida ($Z_{IN} = \infty$, $Z_O = 0$).

Para el modo de entrada discreta se implementó un módulo de relés que fueron controlados por el Arduino con el fin de aislar las terminales del microcontrolador y las del Logic Solver, el cual detectó como nivel bajo un estado de abierto entre sus terminales, y como nivel alto, un cortocircuito entre los canales. En este caso, también se necesitó el acople anteriormente comentado pero que este caso funcionó como un buffer de corriente; para que se pudiera proveer la suficiente cantidad de corriente a cada relé.

En lo que respecta al modo de salida discreta, se implementó un regulador de tensión 7805 con el fin de transformar la fuente de 24V entre los puntos A y C de cada canal del Logic Solver a 5V, valor límite de la lógica digital para los pines de entrada del Arduino. Luego, se conectó cada uno de estos pines a la salida de cada regulador; con lo que así se logró detectar con el microcontrolador valores binarios provenientes del DeltaV. Es necesario señalar que este módulo se implementó para todos los canales de cada Logic Solver.

A continuación se presenta el diagrama de bloques que refleja el funcionamiento eléctrico de la etapa de acople para un sólo canal:

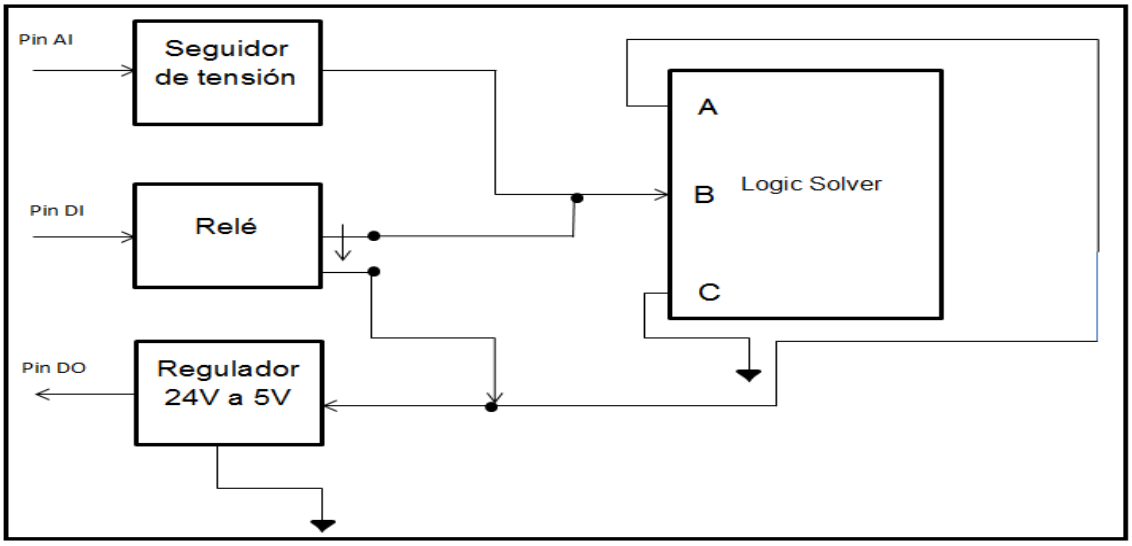


Figura 14. Diseño modular de la etapa de acople para puertos de entrada-salida.

Después de que se realizaron las pruebas de los diferentes modos de operación, se inició con la implementación del protocolo I2C. Esto se realizó mediante la creación de 2 buses de conexión para las señales de datos serie (SDA) y reloj serial (SCL), con el fin de conectar el Arduino maestro a los esclavos a través de esas líneas.

Para verificar la funcionalidad del protocolo, se comenzó a enviar y recibir datos entre el maestro y solo un esclavo. Tras implementar la comunicación con un esclavo, se fue agregando un esclavo y volviendo a ejecutar la prueba sucesivamente, hasta completar el bus con los esclavos. Ya para este momento se cambió la fuente de alimentación USB de los Arduinos, a adaptadores de corriente alterna (CA) a directa (DC) de 9V.

En lo relacionado al mapeo físico de los puertos entrada-salida del Arduino para los respectivos modos de funcionamiento, se sacó provecho de la gran cantidad que este microcontrolador (otra de las razones de su escogencia).Para el modo de entrada analógico se utilizó el grupo de pines del 2 al 13 y 44 al 46, ya que estos están configurados para desplegar señales PWM. El modo de entrada discreta se mapeó a los pines 24 hasta 38 y el modo de salida discreta desde el pin 39 al 53.

Cabe destacar que para este sistema de automatización se implementó solo una puesta a tierra (GND), con el fin de tener la misma referencia en cualquier parte del diseño; esto con el fin de verificar el arribo y envío de datos desde un hardware a otro. Es decir, se unió las referencias de los microcontroladores, el DeltaV SIS, la etapa de acople y las fuentes de alimentación.

5.3. Descripción del Software.

En cuanto al software se refiere, la herramienta de programación de Arduino presenta un entorno amigable con el usuario, usa un lenguaje propio basado en el lenguaje de programación Processing y el cual es altamente compatible con C, C++ y HTML, lo cual genera una gran base de ejemplos disponibles que le permiten al usuario aprender y dominar rápidamente la tecnología Arduino [9].

Para realizar el control del sistema se partió del hecho de que se necesitaba 2 rutinas de programación, una para el maestro y otra para los esclavos. En la rutina del maestro, se implementó la función que inicializó la comunicación por el puerto 80, así como la transmisión y recepción de datos a través del protocolo I2C (Apéndice A.1); en donde cada acción ejecutada por el usuario tuvo su respectivo código, que el maestro se encargó de enviar al esclavo para que este ejecutara cada acción específica.

A continuación se presenta el diagrama de flujo que representa la rutina de programación completa que se ejecutó en el Arduino maestro:

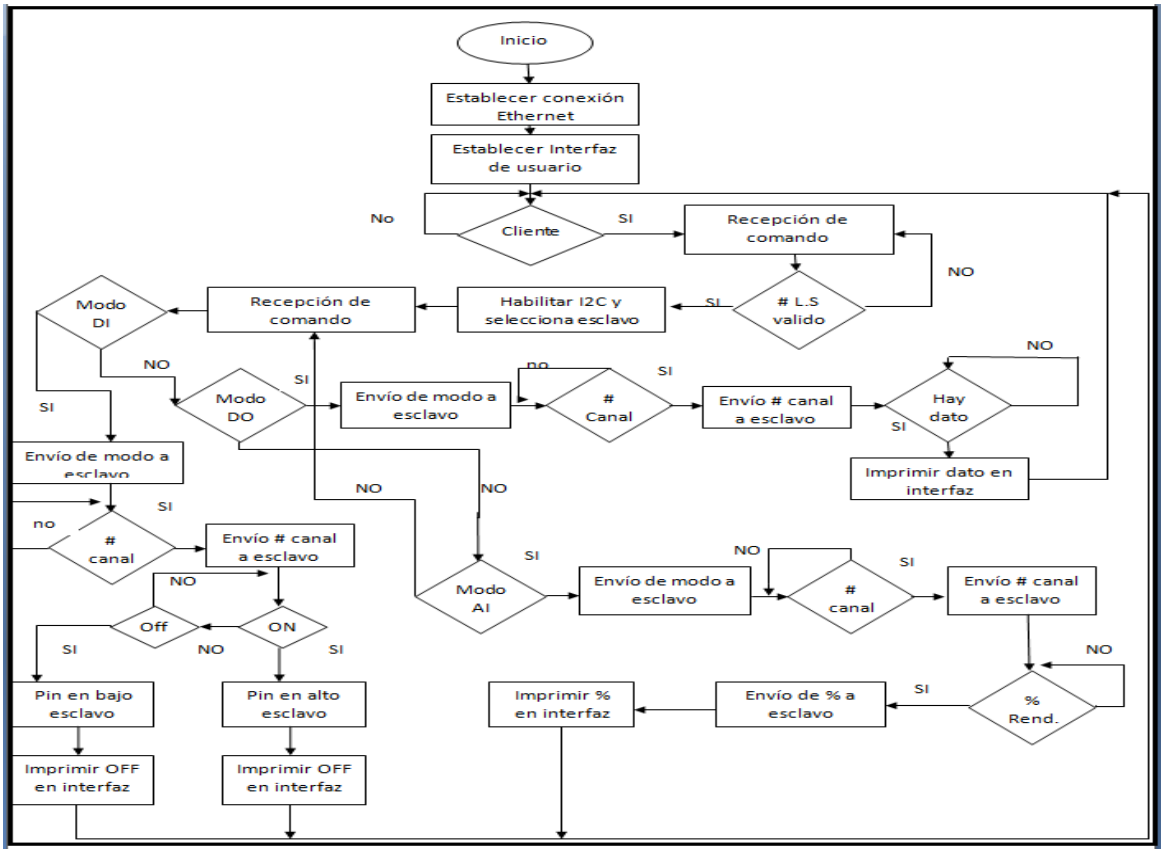


Figura 15. Secuencia lógica de operación del Arduino maestro.

La rutina del esclavo fue la encargada de escribir el valor eléctrico seleccionado en la interfaz, y que se envió a través de la etapa de acople; además leyó el valor proveniente del Logic Solver para luego enviarlo al maestro para que este lo imprimiera en pantalla. En este caso, se detectó una cadena de datos provenientes del botón seleccionado en la interfaz de usuario, que determinó si se escribía o leía un valor en el pin seleccionado o, se desplegaba en la interfaz el dato enviado por el DeltaV SIS.

A continuación se presenta el diagrama de flujo que representa la rutina de programación completa que se ejecutó en el Arduino esclavo:

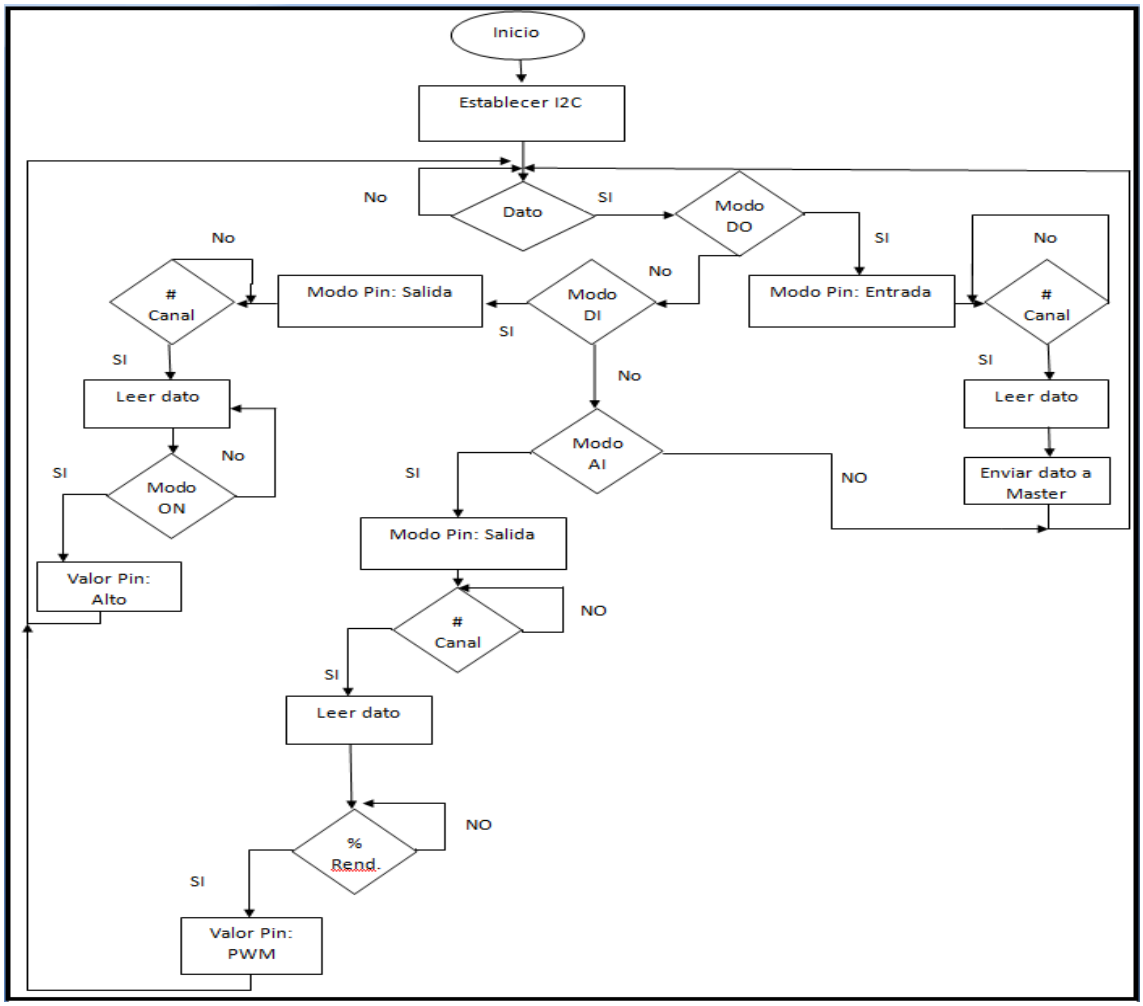


Figura 16. Secuencia lógica de operación del Arduino esclavo.

5.4 .Comunicación con la PC.

Para que la plataforma Arduino pudiera operar en forma de servidor web se tuvo que verificar que la configuración de red fuera adecuada para que el escudo Ethernet funcionara correctamente. Hubo 2 direcciones dentro del protocolo TCP/IP que se debieron establecer correctamente para conectar con éxito el escudo y lograr mostrar los datos que se envían y reciben del servidor web:

• La dirección de control de acceso al medio (MAC) identifica de forma única el escudo Ethernet. Cada dispositivo de red debe tener una dirección MAC diferente, por lo que el escudo tuvo por defecto su propia dirección: byte mac [ ] = {0x90, 0xA2, 0xDA, 0x0D, 0xB1, 0x0F}.

• La dirección de protocolo de internet (IP) se utilizó para identificar algo que se está comunicando a través del puerto 80 y también debe ser única en la red. Las IP se expresan generalmente con puntos separando los bytes, por ejemplo, 10.4.0.30., que fue la dirección que se le asignó al Arduino. En el sketch, se utilizaron comas en lugar de puntos, porque los bytes se almacenaron en una matriz: byte ip [ ] = {10, 4, 0, 30}.La dirección que se utilizó para el escudo de Ethernet se asignó de forma automática por medio del servicio DHCP explicado en el marco teórico.

Las solicitudes y las respuestas de un navegador web usan mensajes en protocolo HTTP; en donde un cliente o un servidor web para responder correctamente, debe interpretar y responder a estas peticiones. La comunicación del sistema implementado entre la aplicación web y la plataforma Arduino utilizó este protocolo.

La petición del Arduino se realizó utilizando el comando Get. Para obtener las modificaciones que se debieron realizar al sistema de control local por la acción de un usuario a través de la aplicación web, se realizó una conexión al servidor cada cierto tiempo, durante este periodo de tiempo se envió el comando get para realizar la petición al servidor.

La aplicación web utilizó el formato Hypertext Markup Language (HTML) para describir la estructura y el contenido de esta. La pantalla presentó una interfaz para que el usuario ingresara los parámetros de la simulación: escogencia de Logic Solver, canales, modos de operación (AI, DI, DO) y porcentajes de prueba de rendimiento. Esta interfaz contuvo un formulario que aceptó los valores numéricos por medio de las entradas de texto; al presionar el botón “aceptar” se activó el evento “Submit” que verificó si los campos de texto se encontraban con datos; para luego enviar un aviso de alerta si hubo alguno vacío. Además, el

botón al ser de tipo “Submit” cargó el código que se encargó de hacer la conexión con la base de datos.

6. Análisis de Resultados.

6.1 Resultados.

A continuación se presentan mediciones de valores eléctricos y porcentajes de rendimiento obtenidos de las pruebas de diferentes modos para los canales de los Logic Solvers. Estos resultados se obtuvieron mediante la prueba del sistema maestro-esclavo, a través de la interfaz de usuario; así como del uso de un multímetro digital Fluke 789 y el software de control DeltaV:

Porcentaje de rendimiento teórico (%)	Porcentaje de rendimiento obtenido (%)	Porcentaje de error (%)
0	0.510	--
10	9.684	3.160
20	19.450	2.750
30	29.789	0.703
40	39.963	0.09
50	50.882	1.764
60	59.856	0.240
70	69.431	0.813
80	79.589	0.513
90	89.888	0.124
100	99.058	0.942

Tabla 1. Porcentajes de rendimiento ingresados y detectados por DeltaV.

Porcentaje de rendimiento (%)	Valor teórico HART (mA)	Valor Obtenido HART (mA)	Porcentaje de error (%)
0	4.00	4,074	1.850
10	5.600	5.679	1.580
20	7.200	7.210	1.111
30	8.800	8.820	0.227
40	10.400	10.351	0.471
50	12.000	11,957	0.358
60	13.600	13.336	1.941
70	15.200	14.945	1.677
80	16.800	16.484	1.880
90	18.400	18.086	1.706
100	20.000	19,531	2.345

Tabla 2. Muestra de porcentajes de rendimiento y corrientes para el modo AI.

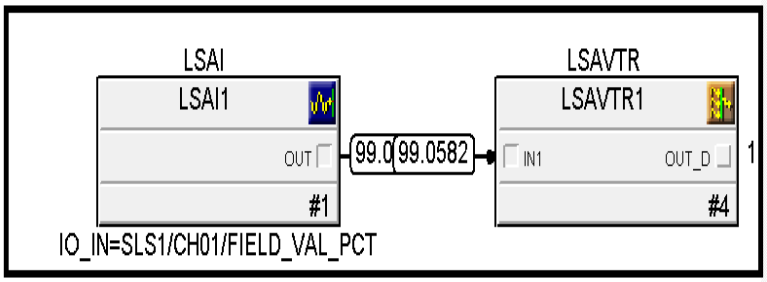


Figura 17. Prueba de entrada analógica para 100% de rendimiento.

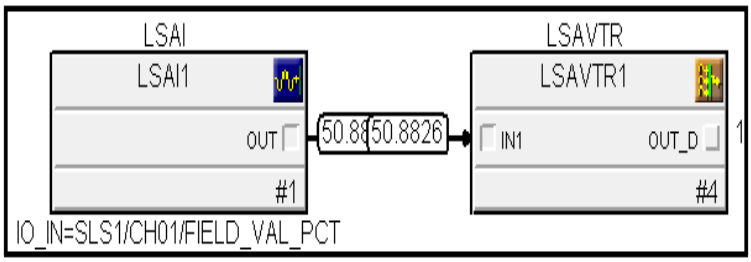


Figura 18. Prueba de entrada analógica para 50% de rendimiento.

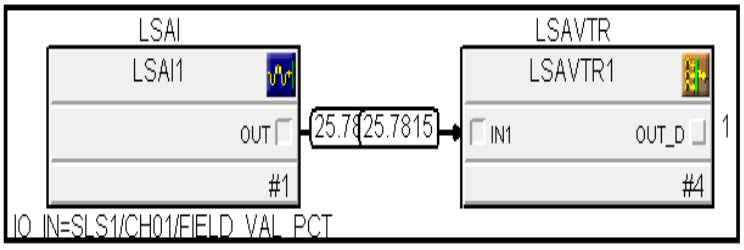


Figura 19. Prueba de entrada analógica para 26% de rendimiento.

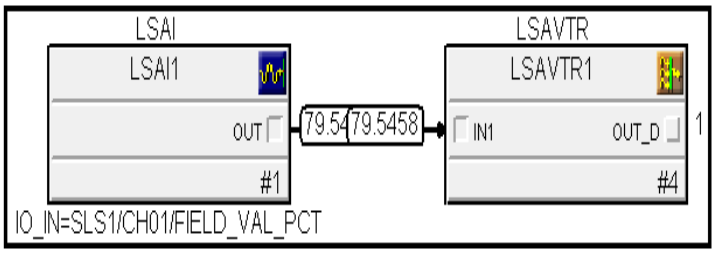


Figura 20. Prueba de entrada analógica para 80% de rendimiento.

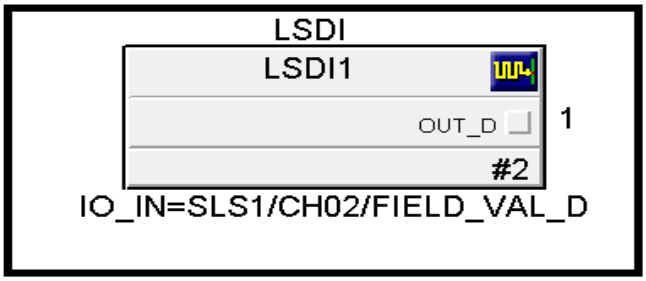


Figura 21. Verificación de Prueba de entrada discreta para un valor de nivel alto.

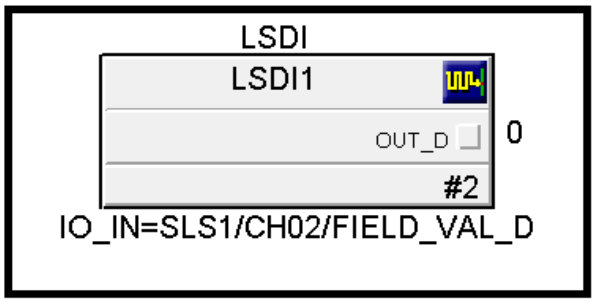


Figura 22. Verificación de Prueba de entrada discreta para un valor de nivel bajo.

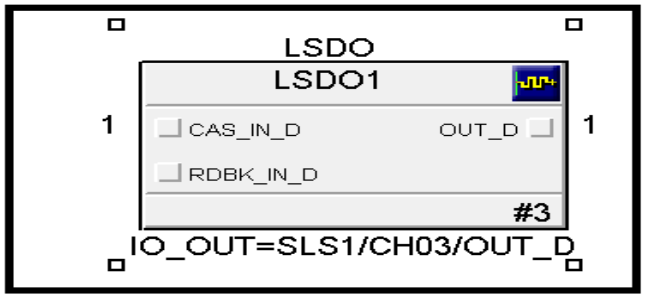


Figura 23. Verificación de Prueba de salida discreta para un valor de nivel alto.

A continuación se presenta la interfaz de usuario usada para comunicar la PC con sistema de automatización:

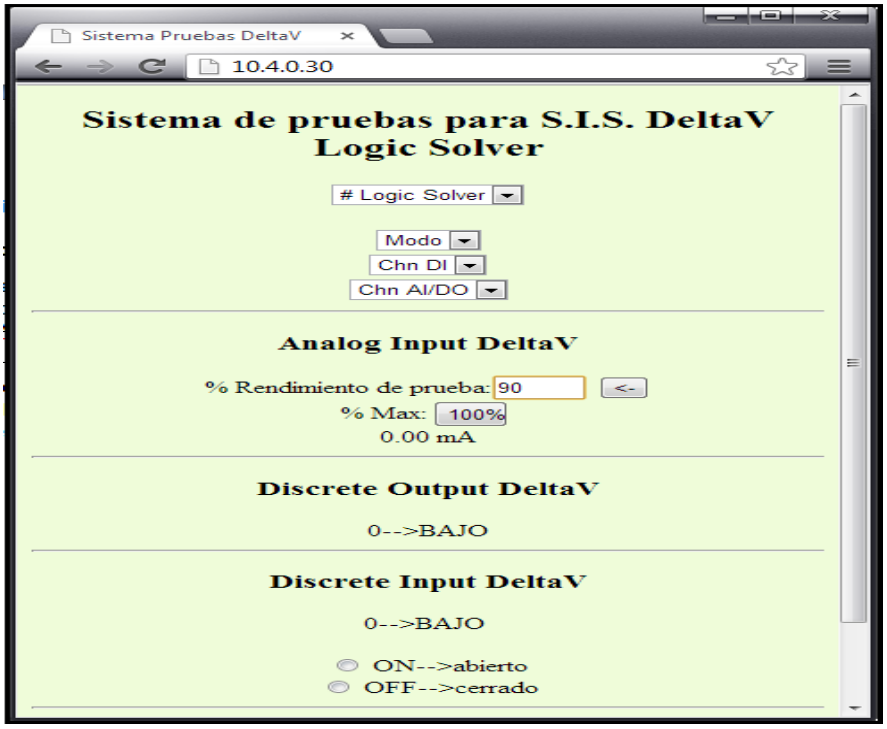


Figura 24. Interfaz de usuario ejecutada en el navegador web.

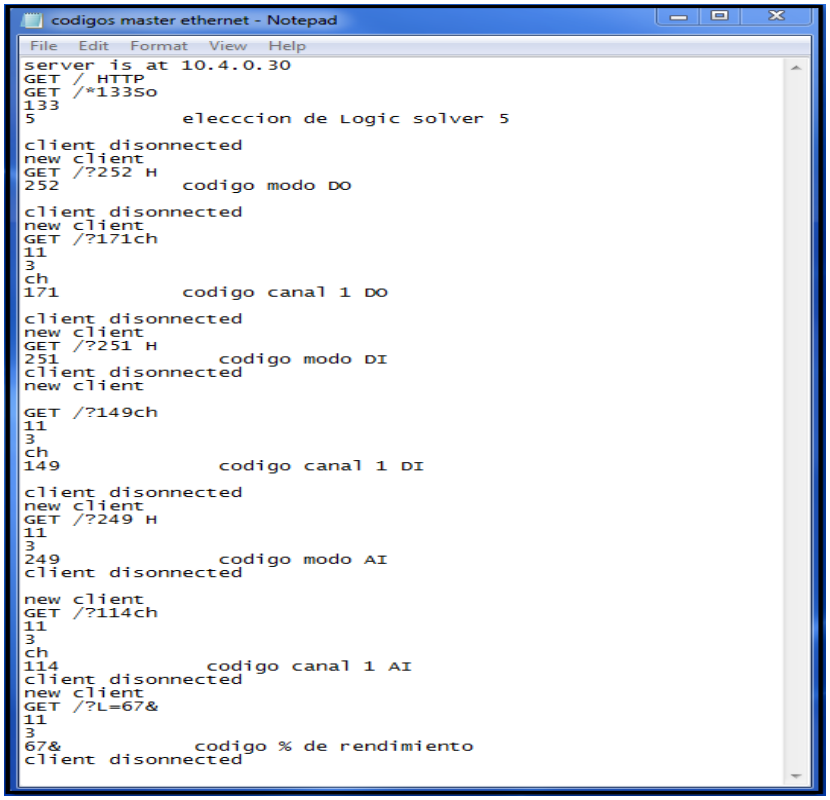


Figura 25. Interpretación del código recibido por maestro vía Ethernet.

6.2 Análisis.

Para comprobar la funcionalidad del sistema creado, se procedió a realizar un análisis desde el punto de vista de los objetivos específicos; los cuales en conjunto dieron forma al objetivo general y por ende, a la meta del proyecto.

Primero, se logró la generación y captura de datos de nivel lógico, así como la implementación de datos que cumplieron con las especificaciones HART para su posterior envío. La confirmación de este objetivo se llevó a cabo mediante la medición de valores eléctricos a través del uso de un multímetro (para las señales tipo HART) y del software DeltaV (para datos de formato digital); después de haber ejecutado alguna petición en la interfaz.

Con base en lo anterior, se satisfizo este primer objetivo específico, el cual se definió como la obtención de una correspondencia de datos medidos con respecto a aquellos establecidos por los protocolos de comunicación del Logic Solver. Esto se respaldó mediante los resultados correspondientes a una muestra de valores pertenecientes a una prueba en modo de entrada analógica, plasmados en la tabla 2, de la cual se pudo constatar que las señales de corriente generadas por el microcontrolador presentaron un porcentaje de error menor al 2,5% con respecto a los valores teóricos. Además, para los modos entrada y salida discreta se tomó como base los valores de salida de los bloques implementados en el software DeltaV (configurados en modos DI y DO respectivamente); los cuales se adjuntaron en la sección anterior (Figuras 21 a 23) y corroboraron la lectura y escritura de valores lógicos altos y bajos, por parte del sistema desarrollado.

Con respecto al segundo objetivo específico, se desarrolló la etapa de acople que sirvió como puente para el tráfico de datos entre el rack y la interfaz web. Es aquí donde se mapeó cada interacción con los botones o cajas de texto de la aplicación, en un código único que se envió al Arduino maestro (servidor) por medio del puerto 80; para que fuera interpretado y activara en un Arduino esclavo (cliente) una acción en particular, definida por el mismo código, que habilitó una sección de la etapa de acople, que dependiendo del modo elegido inicialmente; conectó el pin del arduino con el puerto del logic solver deseado.

La demostración del cumplimento de este objetivo se pudo observar mediante los bloques de prueba del programa DeltaV (figuras 17 a 20), los cuales se usaron para verificar que se llevara a cabo la petición ejecutada en la interfaz. Es así como en la tabla 1 se mostró las pruebas de peticiones de porcentaje de rendimiento ingresadas desde el navegador web y la acción puntual detectada en el programa DeltaV, cuyos valores tuvieron porcentajes de error menores a 3,2%;

por lo que se generó una correspondencia de por lo menos 96,8% en las peticiones del usuario.

El siguiente objetivo fue de la mano con el anterior, ya que para poder ejecutar cada petición por parte del usuario; se necesitó antes la interfaz. Consecuentemente, se implementó una aplicación que contara con los elementos necesarios para ejecutar todas las peticiones requeridas por el usuario (selección de logic solver, modo, canal, valores analógicos o discretos). La demostración de la posibilidad de configurar el sistema de automatización al gusto del usuario se ilustró por medio de la figura 24.

Por lo tanto, se estableció una manipulación total del sistema a través de una interfaz de programación (que reemplazó el sistema de botones físicos y fuentes de corriente utilizado hasta la fecha), y por ende, se cumplió con lograr el 100% de manipulación del rack DeltaV vía PC.

En lo que se refiere al objetivo de la comunicación tipo Ethernet, se implementó una subred de área local a la cual pertenecieran los equipos a desear por la empresa. Es así como al rack DeltaV, al sistema de automatización y las computadoras de prueba se les asignó una dirección IP perteneciente a la subred. Además, para verificar la concordancia en el envío y recepción de datos a través de este protocolo, se utilizó el puerto COM del Arduino para imprimir cada código correspondiente a cada petición realizada por el usuario desde una PC dentro de la subred; petición que fue recibida o transmitida por el mismo Arduino vía puerto 80. En la figura 25, se pudo observar los códigos impresos enviados desde una PC y correspondientes a una función en específico, cumpliendo así con el objetivo de acceder al sistema de pruebas desde una PC, configurada adecuadamente para hacerlo.

Por otro lado, la implementación final tuvo ciertos aspectos que valen la destacar: La cantidad de canales de cada Logic Solver es 16, pero se desarrolló un sistema para 15 canales por Logic Solver. Esto porque cuando se realizó el criterio de diseño correspondiente al hardware, no se encontró ningún microcontrolador con la capacidad de manejar 16 pines analógicos de salida. Arduino fue la posibilidad que mas pines con estas características podía proveer.

También, dentro del modo de operación del Logic Solver se encuetra el de salida analógica o de 2 estados HART. Este modo de operación no se tomó en cuenta debido a la complejidad que presenta este tipo de comunicación y puesta en marcha; es por esto que los ingenieros supervisores del proyecto no tomaron en cuenta este modo dentro de los parámetros a desarrollar en el proyecto. Igualmente, los ingenieros de la empresa han tomado la decisión de implementar

el sistema de automatización en 3 de los 12 Logic Solvers que hay en el rack, al menos en el corto plazo. Esto porque con el sistema desarrollado ya se demuestra el funcionamiento adecuado del Logic Solver y como todos son exactamente iguales; se asume los mismos resultados para los demás.

7. Conclusiones y Recomendaciones.

7.1. Conclusiones.

1. Se puso en evidencia la capacidad y flexibilidad que tiene la plataforma Arduino para interactuar con sistemas de control industrial complejos, como lo es el DeltaV S.I.S.

2. Fue imperativo desarrollar una etapa de acople entre los pines de entrada/salida del Arduino y del Logic Solver, que permitiera una transferencia de datos precisa y un aislamiento del microcontrolador con respecto a tensiones elevadas del DeltaV SIS.

3. Para establecer una comunicación bidireccional a través del protocolo Ethernet entre la PC, el rack DeltaV SIS y el sistema de automatización; se debió asignar a cada uno de ellos, una IP perteneciente a la misma red de área local.

4. La rutina de control maestra implementada en la plataforma Arduino y basada en el protocolo I2C, se diseñó de manera tal que, se contó con gran flexibilidad para agregar más módulos esclavos al sistema de automatización original.

5. El envío y captura de datos vía puerto 80 se realizó a través de las funciones de interacción con el usuario que posee el protocolo HTML, empleadas a través de botones y cajas de texto en la interfaz web.

6. Se tornó relativamente sencillo implementar señales que cumplieron con las especificaciones HART, usando como base la manipulación de la función PWM del Arduino.

7.2 Recomendaciones.

1. Con miras a una implementación más sofisticada, se podría realizar una etapa de acople y potencia unificada por medio de un circuito impreso (PCB) que se conecte al Arduino y así proveer una solución integral y modular a la empresa.

2. Además, si se encuentra en las posibilidades económicas de la empresa, se podría completar el sistema de automatización con los Logic Solvers no tomados en cuenta. Desde el punto de vista de implementación, sólo se necesita adquirir los microcontroladores para programarles la rutina diseñada para el esclavo y replicar la etapa de acople para cada nuevo Logic Solver.

3. Por otra parte, si se quisiera una mejora estética o una versión más sofisticada de la interfaz de usuario; se puede emplear el conocimiento de un diseñador de páginas web, el cual pueda aplicar desarrollar funciones más complejas para la aplicación y que una persona con conocimiento en electrónica no tiene los fundamentos para realizar.

8. Referencias

[1] Castro Gil, M., Díaz Orueta, G., & Mur Pérez, F. (2007). *Comunicaciones industriales: Sistemas distribuidos y aplicaciones.* España: UNED - Universidad Nacional de Educación a Distancia.

[2] Dydek, Z., Annaswamy, A., & Lavresky, E. (2010). *Adaptive control and the NASA x-15-3 flight revised*. Control Systems Magazine, 30(3), 32.

[3] Emerson (2009). *Operations and Security*. Emerson Process Management.

[4] Emerson (2011). *DeltaV SIS Product Data Sheet.* Emerson Process Management.

[5] Emerson (2011). *Modular Safety Concept for Marine & Offshore Applications.* Emerson Process Management.

[6] Emerson (s. f.). *DeltaV SIS Large System Capabilities*. Emerson Process Management.

[7] Miralles, F., & Armelini, G. (2004). *'Linux' y 'software' de código abierto: ¿listos para su empresa?*

[8] Rodríguez Penin, A. (2008). *Comunicaciones* industriales. España: Marcombo.

[9] Rosa, F. d., & Heinz, F. (2009). *Guía práctica sobre software libre: Su selección y aplicación local en américa latina y el caribe*. Uruguay: UNESCO. Office Montevideo and Regional Bureau for Science in Latin America and the Caribbean.

[10] Ruiz Gutierrez, José Manuel. (2013).Implantación de Arduino en las redes Ethernet. Ecuador: Universidad Estatal de Bolivar.

[11] Ruiz Gutierrez, José Manuel. (2012).Manejo y aplicaciones del bus I2C de Arduino. Ecuador: Universidad Estatal de Bolivar.

[12] Saucedo Flores, S., & Rodríguez García, J. L. (1985). Apuntes de control automático de procesos. México: Instituto Politécnico Nacional.

9. Apéndice y Anexos.

9.1 Apéndice.

A.1. Concordancia de datos recibidos y transmitidos por medio del protocolo I2C.

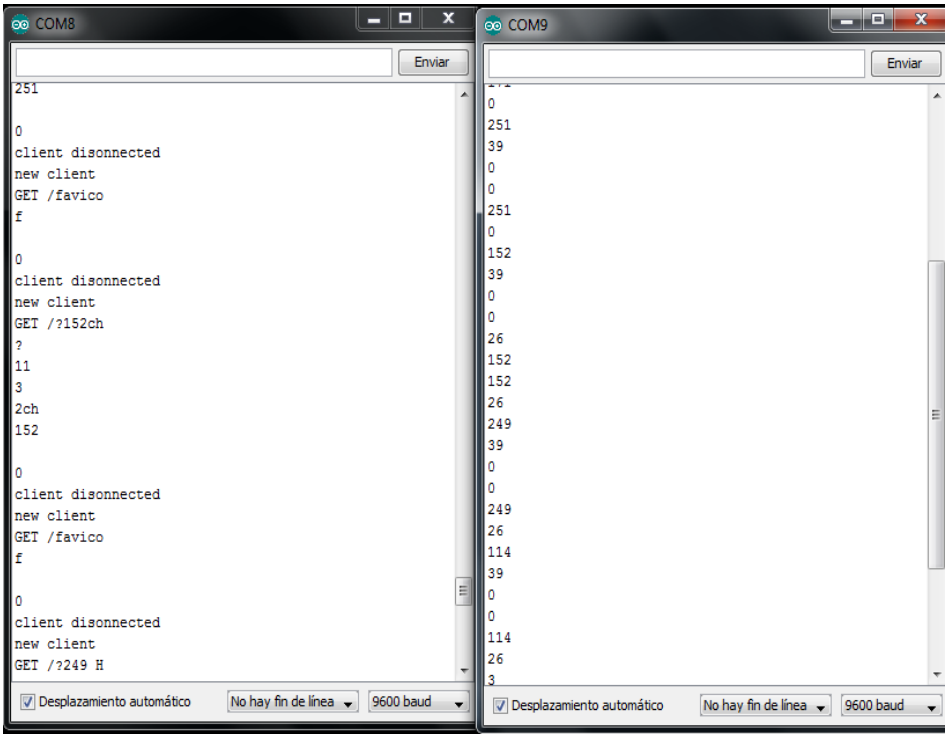


Figura 26. Datos transferidos por el bus I2C.

A.2 Interpretación de códigos recibidos por el Arduino esclavo.

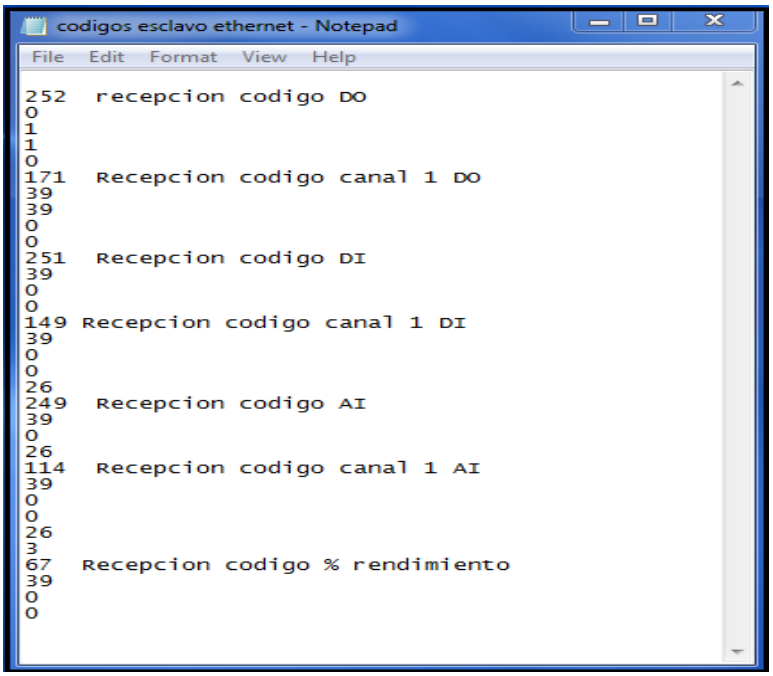


Figura 27. Algunas interpretaciones del código para el Arduino esclavo.

A.3 Medición de parámetros eléctricos de los puertos del Logic Solver.

Modo Canal	Terminales A,B (V)	Terminales A,C (V)	Terminales B,C (V)
Entrada Analógica	22,650	0,000	0,000
Entrada Discreta	12,510	0,000	0,000
Salida Discreta	0,000	22,660	0,000

Tabla 3. Medición de tensión de las combinaciones de canal del Logic Solver.

Terminales	Impedancia de entrada (Ω)
A,B	∞
A,C	∞
B,C	256,100

Tabla 4. Impedancia de entrada de las combinaciones de canal del Logic Solver.

A.4 Diagramas de configuración para la simulación de puertos en DeltaV.

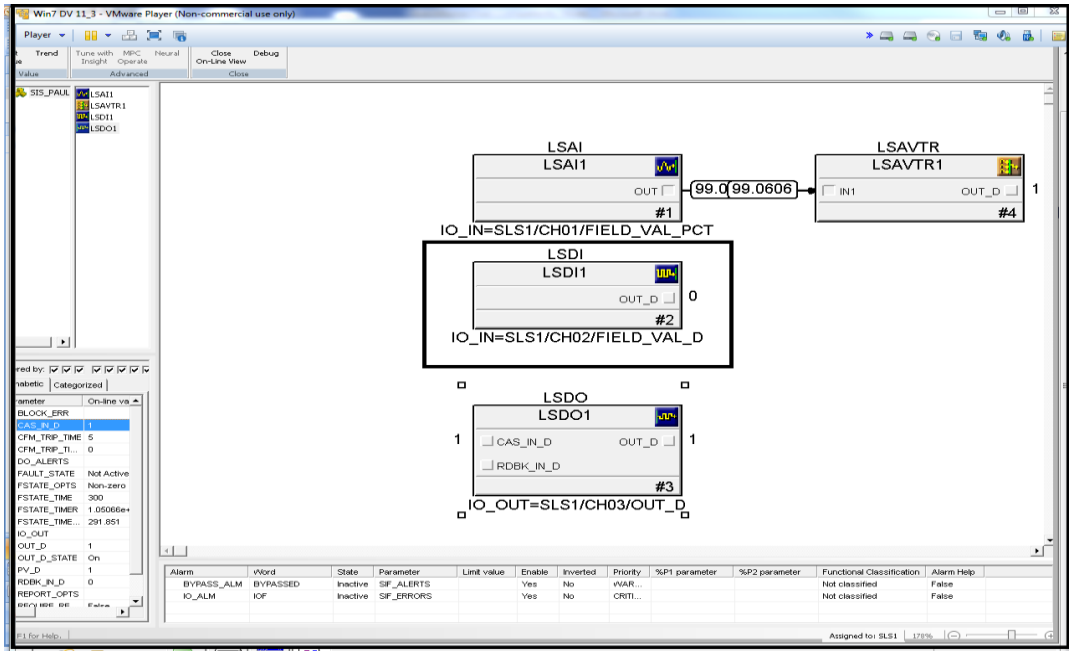


Figura 28. Pruebas de software realizadas en DeltaV.

9.2. Anexos.

A.5. Hoja de datos Arduino Mega 2560.

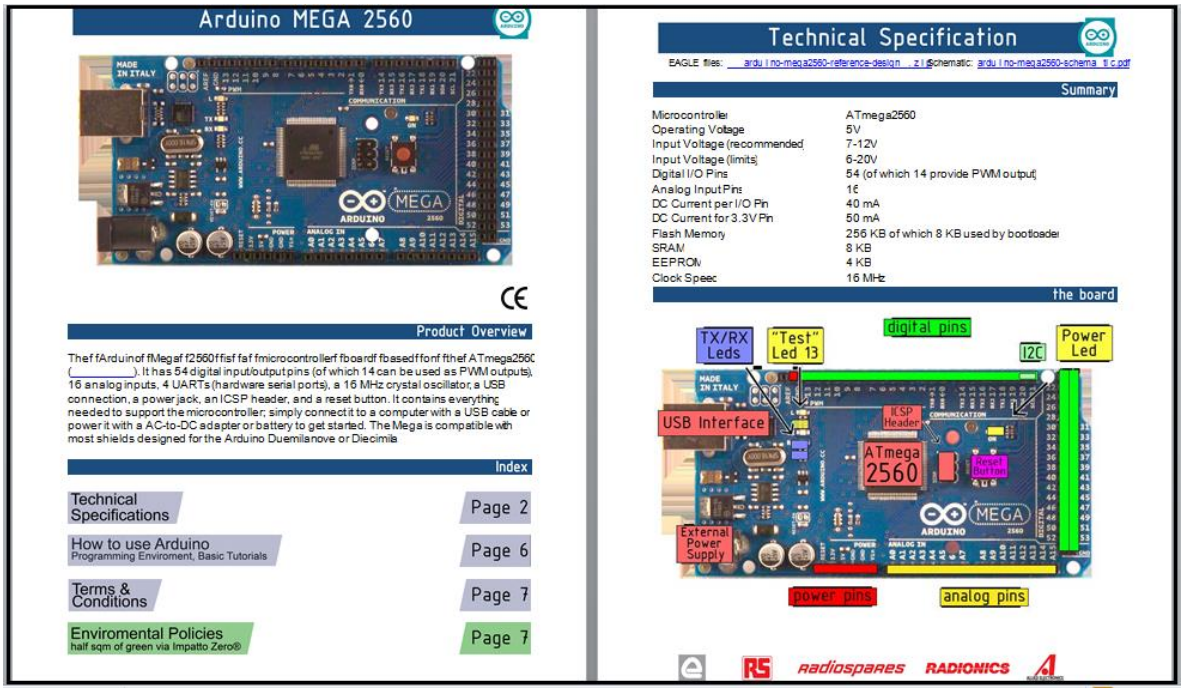


Figura 29. Hoja de datos Arduino Mega 2560.

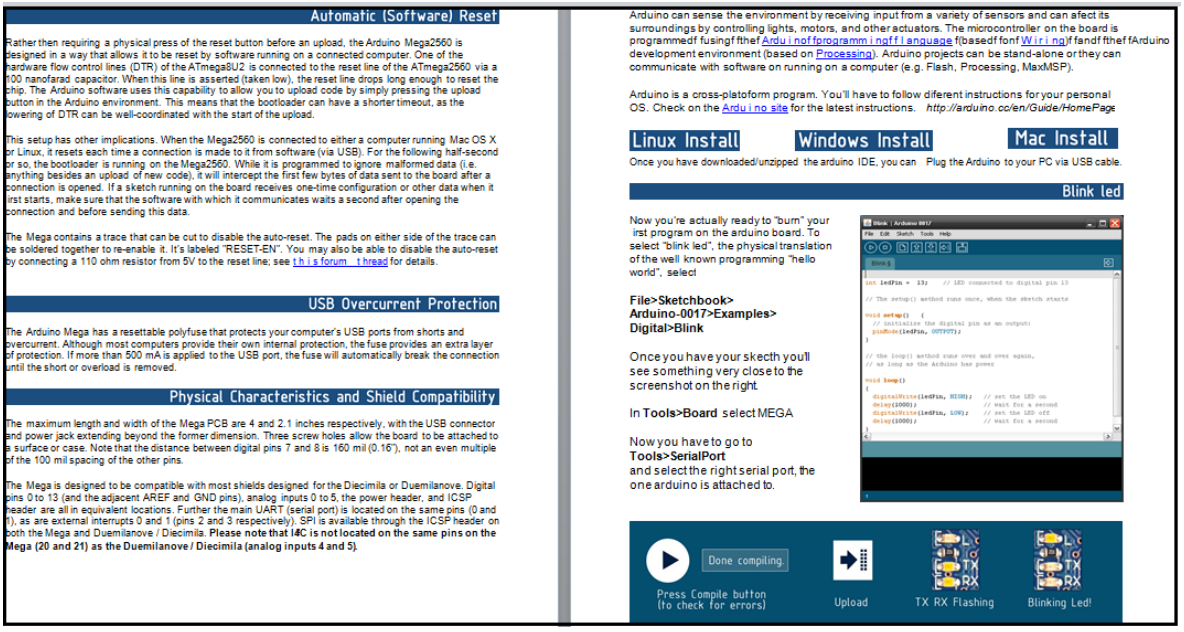


Figura 30. Hoja de datos Arduino Mega 2560.

A.6. Hoja de datos Arduino Ethernet Shield.

Overview

The Arduino Ethernet is a microcontroller board based on the ATmega328 (datasheet). It has

14 digital input/output pins, 6 analog inputs, a 16 MHz crystal oscillator, a RJ45 connection,

a power jack, an ICSP header, and a reset button.

NB: Pins 10, 11, 12 and 13 are reserved for interfacing with the Ethernet module and should not be used otherwise. This reduces the number of available pins to 9, with 4 available as PWM outputs.

An optional Power over Ethernet module can be added to the board as well.

The Ethernet differs from other boards in that it does not have an onboard USB-to-serial driver chip, but has a Wiznet Ethernet interface. This is the same interface found on the Ethernet shield.

An onboard microSD card reader, which can be used to store files for serving over the network, is accessible through the SD Library. Pin 10 is reserved for the Wiznet interface, SS for the SD card is on Pin 4.

The 6-pin serial programming header is compatible with the USB Serial adapter and also with the FTDI USB cables or with Sparkfun and Adafruit FTDI-style basic USB-to-serial breakout boards. It features support for automatic reset, allowing sketches to be uploaded without pressing the reset button on the board. When plugged into a USB to Serial adapter, the Arduino Ethernet is powered from the adapter.

Summary

Microcontroller ATmega328

Operating Voltage 5V

Input Voltage Plug(limits) 6-18V

Input Voltage PoE (limits) 36-57V

Digital I/O Pins 14 (of which 4 provide PWM output)

Arduino Pins reserved:

10 to 13 used for SPI

4 used for SD card

2 W5100 interrupt (when bridged)

Analog Input Pins 6

DC Current per I/O Pin 40 mA

DC Current for 3.3V Pin 50 mA

Flash Memory

32 KB (ATmega328) of which 0.5 KB used by

bootloader

SRAM 2 KB (ATmega328)

EEPROM 1 KB (ATmega328)

Clock Speed 16 MHz

W5100 TCP/IP Embedded Ethernet

Controller

Power Over Ethernet ready Magnetic Jack

Micro SD card, with active voltaje translators

Power

The board can also be powered via an external power supply, an optional Power over

Ethernet (PoE) module, or by using a FTDI cable/USB Serial connector.

External power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

VIN. The input voltage to the Arduino board when it's using an external power source (as

opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

5V. the regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.

3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

GND. Ground pins.

The Power Over Ethernet (PoE) module is designed to extract power from a conventional twisted pair Category 5 Ethernet cable:

IEEE802.3af compliant

Low output ripple and noise (100mVpp)

Input voltage range 36V to 57V

Overload and short-circuit protection

9V Output

High efficiency DC/DC converter: typ 75% @ 50% load

1500V isolation (input to output)

When using the power adapter, power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of

SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Ethernet board can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data.

External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

PWM: 3, 5, 6, 9, and 10. Provide 8-bit PWM output with the analogWrite() function.

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.

LED: 9. There is a built-in LED connected to digital pin 9. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off. On most other arduino boards, this LED is found on pin 13. It is on pin 9 on the Ethernet board because pin 13 is used as part of the SPI connection.

A.7. Hoja de datos Amplificador Operacional 741.

LM741 Operational Amplifier

Check for Samples: LM741

FEATURES

- Overload Protection on the Input and Output
- No Latch-Up When the Common Mode Range is Exceeded

DESCRIPTION

The LM741 series are general purpose operational amplifiers which feature improved performance over industry standards like the LM709. They are direct, plug-in replacements for the 709C, LM201, MC1439 and 748 in most applications.

The amplifiers offer many features which make their application nearly foolproof: overload protection on the input and output, no latch-up when the common mode range is exceeded, as well as freedom from oscillations.

The LM741C is identical to the LM741/LM741A except that the LM741C has their performance ensured over a 0°C to +70°C temperature range, instead of −55°C to +125°C.

Connection Diagrams

LM741H is available per JM38510/10101

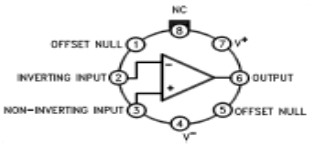


Figure 1. TO-99 Package
See Package Number LMC0008C

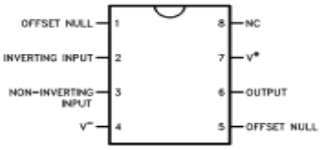


Figure 2. CDIP or PDIP Package
See Package Number NAB0008A, P0008E

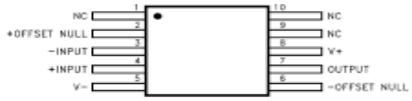


Figure 3. CLGA Package
See Package Number NAD0010A

Figura 31. Hoja de datos Amplificador Operacional 741.

Absolute Maximum Ratings[1][2][3]

	LM741A	LM741	LM741C
Supply Voltage	±22V	±22V	±18V
Power Dissipation [4]	500 mW	500 mW	500 mW
Differential Input Voltage	±30V	±30V	±30V
Input Voltage [5]	±15V	±15V	±15V
Output Short Circuit Duration	Continuous	Continuous	Continuous
Operating Temperature Range	−55°C to +125°C	−55°C to +125°C	0°C to +70°C
Storage Temperature Range	−65°C to +150°C	−65°C to +150°C	−65°C to +150°C
Junction Temperature	150°C	150°C	100°C
Soldering Information			
P0008E-Package (10 seconds)	260°C	260°C	260°C
NAB0008A- or LMC0008C-Package (10 seconds)	300°C	300°C	300°C
M-Package			
Vapor Phase (60 seconds)	215°C	215°C	215°C
Infrared (15 seconds)	215°C	215°C	215°C
ESD Tolerance [6]	400V	400V	400V

(1) "Absolute Maximum Ratings" indicate limits beyond which damage to the device may occur. Operating Ratings indicate conditions for which the device is functional, but do not ensure specific performance limits.
(2) For military specifications see RETS741X for LM741 and RETS741AX for LM741A.
(3) If Military/Aerospace specified devices are required, please contact the TI Sales Office/Distributors for availability and specifications.
(4) For operation at elevated temperatures, these devices must be derated based on thermal resistance, and T_j max. (listed under "Absolute Maximum Ratings"). $T_j = T_A + (\theta_{jA} P_D)$.
(5) For supply voltages less than ±15V, the absolute maximum input voltage is equal to the supply voltage.
(6) Human body model, 1.5 kΩ in series with 100 pF.

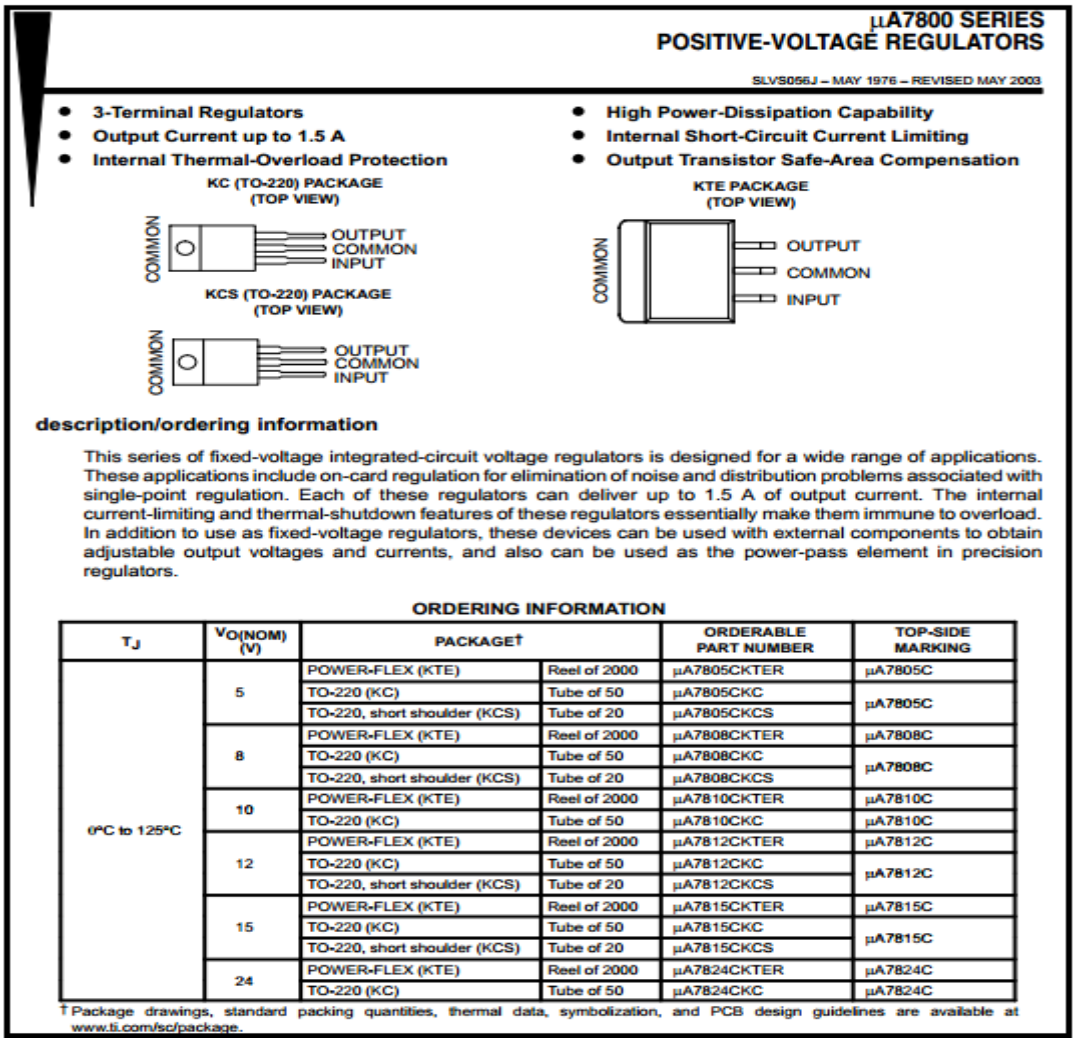
Electrical Characteristics[1]

Parameter	Test Conditions	LM741A			LM741			LM741C			Units
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Input Offset Voltage	T_A = 25°C										
	$R_S \le 10$ kΩ					1.0	5.0		2.0	6.0	mV
	$R_S \le 50$Ω		0.8	3.0							
	$T_{AMIN} \le T_A \le T_{AMAX}$										
	$R_S \le 50$Ω			4.0							mV
	$R_S \le 10$ kΩ						6.0			7.5	
Average Input Offset Voltage Drift				15							µV/°C

(1) Unless otherwise specified, these specifications apply for V_S = ±15V, −55°C ≤ T_A ≤ +125°C (LM741/LM741A). For the LM741C/LM741E, these specifications are limited to 0°C ≤ T_A ≤ +70°C.

Figura 32. Hoja de datos Amplificador Operacional 741.

A.8. Hoja de datos Regulador de tensión 7805.



μA7800 SERIES
POSITIVE-VOLTAGE REGULATORS

SLVS056J – MAY 1976 – REVISED MAY 2003

- 3-Terminal Regulators
- Output Current up to 1.5 A
- Internal Thermal-Overload Protection
- High Power-Dissipation Capability
- Internal Short-Circuit Current Limiting
- Output Transistor Safe-Area Compensation

KC (TO-220) PACKAGE
(TOP VIEW)

KCS (TO-220) PACKAGE
(TOP VIEW)

KTE PACKAGE
(TOP VIEW)

description/ordering information

This series of fixed-voltage integrated-circuit voltage regulators is designed for a wide range of applications. These applications include on-card regulation for elimination of noise and distribution problems associated with single-point regulation. Each of these regulators can deliver up to 1.5 A of output current. The internal current-limiting and thermal-shutdown features of these regulators essentially make them immune to overload. In addition to use as fixed-voltage regulators, these devices can be used with external components to obtain adjustable output voltages and currents, and also can be used as the power-pass element in precision regulators.

ORDERING INFORMATION

T_J	$V_{O(NOM)}$ (V)	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 125°C	5	POWER-FLEX (KTE)	Reel of 2000	μA7805CKTER	μA7805C
		TO-220 (KC)	Tube of 50	μA7805CKC	μA7805C
		TO-220, short shoulder (KCS)	Tube of 20	μA7805CKCS	
	8	POWER-FLEX (KTE)	Reel of 2000	μA7808CKTER	μA7808C
		TO-220 (KC)	Tube of 50	μA7808CKC	μA7808C
		TO-220, short shoulder (KCS)	Tube of 20	μA7808CKCS	
	10	POWER-FLEX (KTE)	Reel of 2000	μA7810CKTER	μA7810C
		TO-220 (KC)	Tube of 50	μA7810CKC	μA7810C
	12	POWER-FLEX (KTE)	Reel of 2000	μA7812CKTER	μA7812C
		TO-220 (KC)	Tube of 50	μA7812CKC	μA7812C
		TO-220, short shoulder (KCS)	Tube of 20	μA7812CKCS	
	15	POWER-FLEX (KTE)	Reel of 2000	μA7815CKTER	μA7815C
		TO-220 (KC)	Tube of 50	μA7815CKC	μA7815C
		TO-220, short shoulder (KCS)	Tube of 20	μA7815CKCS	
	24	POWER-FLEX (KTE)	Reel of 2000	μA7824CKTER	μA7824C
		TO-220 (KC)	Tube of 50	μA7824CKC	μA7824C

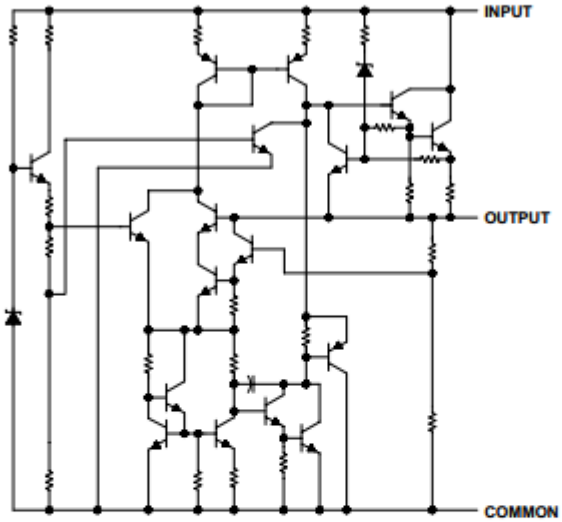
† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

Figura 33. Hoja de datos Regulador de tensión 7805.

µA7800 SERIES
POSITIVE-VOLTAGE REGULATORS

SLVS056J – MAY 1976 – REVISED MAY 2003

schematic



absolute maximum ratings over virtual junction temperature range (unless otherwise noted)†

Input voltage, V_I:	µA7824C	40 V
	All others	35 V
Operating virtual junction temperature, T_J		150°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds		260°C
Storage temperature range, T_{stg}		–65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

package thermal data (see Note 1)

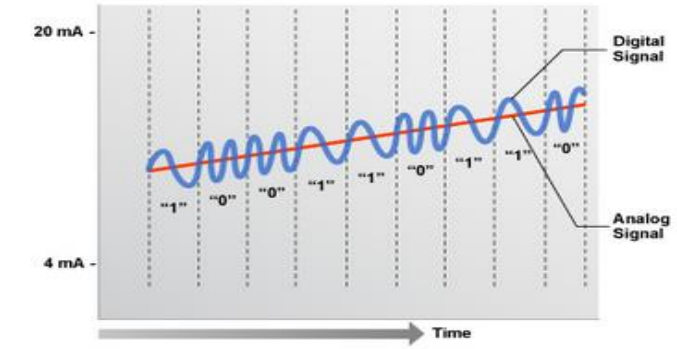
PACKAGE	BOARD	θ_{JC}	θ_{JA}
POWER-FLEX (KTE)	High K, JESD 51-5	3°C/W	23°C/W
TO-220 (KC/KCS)	High K, JESD 51-5	3°C/W	19°C/W

NOTE 1: Maximum power dissipation is a function of $T_J(max)$, θ_{JA}, and T_A. The maximum allowable power dissipation at any allowable ambient temperature is $P_D = (T_J(max) – T_A)/\theta_{JA}$. Operating at the absolute maximum T_J of 150°C can affect reliability.

Figura 34. Hoja de datos Regulador de tensión 7805.

A.9. Hoja de especificación Protocolo HART.

"HART" is an acronym for Highway Addressable Remote Transducer. The HART Protocol makes use of the Bell 202 Frequency Shift Keying (FSK) standard to superimpose digital communication signals at a low level on top of the 4-20mA.



Note: Drawing not to scale

Digital over Analog

Figure 1. Frequency Shift Keying (FSK)

This enables two-way field communication to take place and makes it possible for additional information beyond just the normal process variable to be communicated to/from a smart field instrument. The HART Protocol communicates at 1200 bps without interrupting the 4-20mA signal and allows a host application (master) to get two or more digital updates per second from a smart field device. As the digital FSK signal is phase continuous, there is no interference with the 4-20mA signal.

HART technology is a master/slave protocol, which means that a smart field (slave) device only speaks when spoken to by a master. The HART Protocol can be used in various modes such as point-to-point or multidrop for communicating information to/from smart field instruments and central control or monitoring systems.

HART Communication occurs between two HART-enabled devices, typically a smart field device and a control or monitoring system. Communication occurs using standard instrumentation grade wire and using standard wiring and termination practices.

The HART Protocol provides two simultaneous communication channels: the 4-20mA analog signal and a digital signal. The 4-20mA signal communicates the primary measured value (in the case of a field instrument) using the 4-20mA current loop - the fastest and most reliable industry standard. Additional device information is communicated using a digital signal that is superimposed on the analog signal.

Figura 35. Especificación del protocolo HART.

10. Fórmulas.

10.1. Ley de Ohm:

$$V = I \times R \tag{10.1}$$

10.2. Señal PWM:

$$PWM_{NIVEL} = \frac{255}{5} \times V_{EFF} \tag{10.2}$$

10.3. Seguidor de tensión:

$$V_{ENTRADA} = V_{SALIDA} \tag{10.3}$$

$$Z_{ENTRADA} = \infty \tag{10.4}$$

$$Z_{SALIDA} = 0 \tag{10.5}$$

10.4. Porcentaje de error:

$$\%_{ERROR} = \frac{Valor_{TEORICO} - Valor_{EXPERIMENTAL}}{Valor_{TEORICO}} \times 100 \tag{10.6}$$