UNIVERSIDAD NACIONAL DEL CALLAO

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

"CONTROLADOR LÓGICO PROGRAMABLE DE SALIDA TIPO RELÉ, BASADO EN ARDUINO PARA TRANSMISIÓN DE DATOS ENTRE ETAPAS DE PROCESOS INDUSTRIALES"

SUSTENTACIÓN DE TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

INTEGRANTES:
BACH. CHAFLOQUE MORENO, ALEXIS JHONNY
BACH. PAUCCAR ALCANTARA, LUIS MIGUEL
BACH. RIVERA VARGAS, JOSE LUIS

ASESOR:
DR. ING. RUBIÑOS JIMENEZ SANTIAGO LINDER

Callao, 2019
PERÚ

ACTA PARA LA OBTENCIÓN DEL TITULO PROFESIONAL POR LA MODALIDAD DE TESIS SIN CICLO DE TESIS

A LOS 11 DIAS DEL MES DE ABRIL DEL 2019 SIENDO LAS 11:00 HORAS SE REUNIO EL JURADO EXAMINADOR DE LA FACULTAD DE INGENIERIA ELÉCTRICA Y ELECTRONICA CONFORMADO POR LOS SIGUIENTES DOCENTES ORDINARIOS DE LA UNIVERSIDAD DEL CALLAO.
(RES. DECANAL Nº 053-2019-DFIEE)

- Dr. Ing. NICANOR RAÚL BENITES SARAVIA — PRESIDENTE
- Mg. Ing. Wilbert CHAVEZ IRAZABAL — SECRETARIO
- Mg. Ing. JORGE ELIAS HOSCOSO SANCHEZ — VOCAL

CON EL FIN DE DAR INICIO A LA EXPOSICION DE LOS TESISTAS BACHILLERES EN INGENIERIA ELECTRONICA, QUIENES HABIENDO CUMPLIDO CON LOS REQUISITOS ESTABLECIDOS EN LA NORMATIVA, SUSTENTARAN LA TESIS TITULADA: " CONTROLADOR LOGICO PROGRAMABLE DE SALIDA TIPO RELÉ, BASADO EN ARDUINO PARA TRANSMISION DE DATOS ENTRE ETAPAS DE PROCESOS INDUSTRIALES"

CON EL QUORUM REGLAMENTARIO DE LEY SE DIO INICIO A LA EXPOSICIÓN, CONSIDERANDO LO ESTABLECIDO EN EL REGLAMENTO DE GRADOS Y TITULOS, CORRESPONDIENTE AL OTORGAMIENTO DEL TITULO PROFESIONAL POR LA MODALIDAD DE TESIS SIN CICLO DE TESIS, EFECTUADAS LAS DELIBERACIONES PERTINENTES SE ACORDO: DAR POR Aprobado CON CALIFICATIVO BUENO NOTA: 15 A LOS EXPOSITORES SEÑORES: RIVERA VARGAS, JOSE LUIS; PUCCAR ALCANTARA LUIS MIGUEL, CHAFLOQUE MORENO ALEXIS JHONY, CON LO CUAL SE DIO POR CONCLUIDO LA SESION SIENDO A LAS 12:30 Hrs DEL DIA Y MES EN CURSO.

Mg. JORGE ELIAS HOSCOSO SANCHEZ
VOCAL

Dr. NICANOR RAUL BENITES SARAVIA
PRESIDENTE

Mg. Wilbert CHAVEZ IRAZABAL
SECRETARIO

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

"CONTROLADOR LÓGICO PROGRAMABLE DE SALIDA TIPO RELÉ, BASADO EN ARDUINO PARA TRANSMISION DE DATOS ENTRE ETAPAS DE PROCESOS INDUSTRIALES"

INTEGRANTES:
BACH. CHAFLOQUE MORENO, ALEXIS JHONNY
BACH. PAUCCAR ALCANTARA, LUIS MIGUEL
BACH. RIVERA VARGAS, JOSE LUIS

ASESOR:
DR. ING. RUBIÑOS JIMENEZ SANTIAGO LINDER

CALIFICACIÓN:
15 (QUINCE)

Dr. Ing. BENITES SARAVIA NICANOR RAÚL Presidente de Jurado	Mg. Ing CHAVEZ IRAZABAL WILBERT Secretario de Jurado

Ing. MOSCOSO SANCHEZ JORGE
Vocal de Jurado

Callao, 2019
PERÚ

DEDICATORIA

A mis padres, Elvis y Yolanda, y hermanos, Anthonny y Andree, por el apoyo brindado y hacerme notar que siempre cuento con ellos.

A mi esposa Briyit y a mi hija Briale por ser mi apoyo moral, motivación e inspiración.

A mis amigos Jose y Luis, por su apoyo y conocimientos durante este trabajo.

Alexis Jhonny Chafloque Moreno

A Dios: por permitirme tener la fuerza para terminar mi carrera.

A mis padres Lucio y Carmen: por su esfuerzo en concederme la oportunidad de estudiar y por su constante apoyo a lo largo de mi vida, los quiero y son mi más grande bendición.

A mis hermana y familiares: por sus consejos, paciencia y toda la ayuda que me brindaron para concluir mis estudios.

Y a mis amigos: Alexis y Jose por su valioso apoyo en la tesis, ¡gracias!

Luis Miguel Pauccar Alcantara

Al creador de todas las cosas, el que me ha dado fortaleza en las adversidades; por ello, con toda humildad que de mi corazón puede emanar, dedico primeramente a Dios.

De igual forma, dedico este trabajo a mis padres: Bonifacio y Claudia que me han encolcado conservar los valores y virtudes, los cuales me ha ayudado a salir adelante en los momentos más difíciles.

A mi hermano Edgar en paz descanse, que, durante toda su vida ha luchado por llevar adelante a mi familia, y me ha enseñado a perseverar en todas las circunstancias.

A mis hermanos: Sabino, Efraín, Elizabeth, Audemeo y Jahel, por compartir conmigo los momentos gratificantes, por escucharme, por brindarme siempre sus apoyos incondicionalmente.

A mis amigos, Luis Miguel Pauccar y Alexis Chafloque que, gracias a su apoyo hicieron de esta experiencia una de las más especiales.

Jose Luis Rivera Vargas

AGRADECIMIENTO

Antes que nada, el agradecimiento a nuestros docentes de la Facultad de Ingeniería eléctrica y electrónica de la Universidad Nacional del Callao, porque sus enseñanzas son la base e inspiración de este trabajo de investigación.

Así mismo, agradecer al Instituto de investigación de la Facultad de Ingeniería Eléctrica y Electrónica por permitirnos fortalecer este trabajo de investigación, alcanzando un nivel profesional adecuado.

Los autores

INDICE

TABLA DE CONTENIDOS

1. Contenido de figuras

2. Contenido de Tablas

RESUMEN

Actualmente, en el área de control y automatización, el uso de protocolos de comunicación basados en un controlador lógico programable (PLC), es fundamental, ya sea para grandes industrias o en pequeñas aplicaciones. Dado esto se nos ofertan una gran variedad de marcas y modelos, cada uno con un costo acorde a las características del mismo. Hay diversos factores que causan la elevación del costo de utilización de un PLC, entre ellos tenemos el costo del cable de programación, el elevado precio de los módulos de expansión, entre otros.

Este proyecto nace de la necesidad de reducir el costo de la automatización de un proceso, permitiendo a una mayor cantidad de usuarios utilizar un modelo alternativo de PLC en las aplicaciones que requieran.

El proyecto consiste en adaptar la placa programable "Arduino" para que funcione bajo los mismos parámetros que un PLC comercial, por ejemplo, tensión de entrada (0 – 24VDC), corriente de entrada (4 – 20mA), tensión en la salida (0 – 250VAC; 0 – 30VDC) y corriente en la salida (0 – 10A) esto se logra mediante el uso dispositivos electrónicos básicos y de potencia, incluyendo también aislamiento galvánico para la protección del Arduino, previniendo el traspaso de portadores de carga. Una manera adicional de proteger la placa, es con una carcasa basada en la norma IEC 144.

La esencia de esta nueva placa autómata a diseñar radica en la facilidad del lenguaje de programación "Arduino", la gran cantidad de proyectos ya realizados en este lenguaje, la versatilidad de la placa, el número de entradas y salidas, bajo costo y

facilidad al conseguir cables de programación, además de la gran cantidad de módulos y sensores compatibles con las distintas versiones de la placa. Esto aumenta la cantidad de aplicaciones en las que se podría utilizar este nuevo autómata, manteniendo un bajo costo a comparación de las demás ofertas.

ABSTRACT

Currently, in the area of control and automation, the use of a programmable logic controller (PLC) is essential, whether for large industries or small applications. Given that, we are offered a variety of makes and models, each with a cost according to the characteristics. There are various factors that cause the increase in the cost of using a PLC, among them is the cost of cable programming, the high price of expansion modules, among others.

This project stems from the need to reduce the cost of automating a process, allowing more users to use an alternative model of PLC in applications that require it.

The project is to adapt the programmable board "Arduino" to work under the same parameters as a commercial PLC, for example input voltage (0 - 24VDC), input current (4 - 20mA) output voltage (0 - 250VAC; 0 - 30VDC) and current at the output (0 - 10A) this is achieved by using basic and power electronic devices, also including galvanic isolation for protection Arduino, preventing the transfer of charge carriers. An additional way to protect the board is a housing based on IEC 144 standard.

The essence of this new controller design is the ease of programming language "Arduino" the large number of projects already undertaken in this language, the versatility of the board, the number of inputs and outputs board, low cost and ease the get programming cables, plus the large number of modules and sensors compatible with different versions of the board. This increases the number of

applications that could use this new automaton, maintaining a low cost compared to other bids.

CAPITULO I

I. PLANTEAMIENTO DEL PROBLEMA

1.1. Identificación del Problema

El controlador lógico programable (PLC) es, actualmente, el componente básico en el mundo de la automatización, esto genera que lo sistemas basados en PLC adquieran un costo elevado.

Arduino fue creado para ser utilizado en aplicaciones generales, es un hardware de fácil uso y tiene un costo bajo.

La principal diferencia de estos dos dispositivos, además del precio, radica en los niveles de tensión en los que funcionan. Es conocido que los PLCs generalmente trabajan a tensiones industriales de 24VDC, mientras que Arduino utiliza señales de 5VDC. Entonces, en las entradas, la función de la interfaz será convertir las señales de 24VDC; provenientes de sensores, pulsadores, entre otros; a 5 VDC, mientras que, en la salida, la función de la interfaz será manipular tensiones industriales, sea en corriente alterna o continua, con señales de 5VDC; estas tensiones de salida podrán ser aplicadas a los actuadores, que podrían ser lámparas incandescentes, contactores, electroválvula, entre otros.

1.2. Formulación del problema

A manera de organizar la investigación se plantea el problema general y problemas específicos.

1.2.1. Problema General

¿Es posible implementar un controlador lógico programable, cuyo núcleo sea un Arduino, de bajo costo, a fin de expandir la aplicación de la automatización y para transmisión de datos entre etapas de procesos industriales?

Sí, es viable mediante el uso correcto de la electrónica de potencia y con el análisis adecuado del proceso en el cual será usado este autómata.

1.2.2. Problema específico:

- ¿Es posible adaptar las entradas y salidas del Arduino para funcionar a nivel de tensión industrial?
- ¿Es posible enlazar dos de estos autómatas para transmitir datos sin costo alguno?
- ¿Es posible desarrollar un sistema de supervisión para este sistema?

1.3. Objetivos de la investigación

1.3.1. Objetivo General

Implementar un controlador lógico programable, cuyo núcleo sea un Arduino, de bajo costo, a fin de expandir la aplicación de la automatización y para transmisión de datos entre etapas de procesos industriales.

1.3.2. Objetivos específicos

- Adaptar las entradas y salidas del Arduino mediante circuitos electrónicos de potencia para funcionar a nivel de tensión industrial.

- Enlazar dos de estos autómatas, sin costo alguno, mediante protocolos de comunicación para la transmisión de datos.

- Desarrollar un sistema de supervisión para este sistema mediante protocolos de redes industriales.

1.4. Justificación

La ejecución del presente trabajo de investigación, se justifica de manera:

a. Legal

En el presente trabajo de investigación, por su naturaleza, este rubro no es aplicable, porque no necesita ninguna justificación legal para su estudio.

b. Teórica

Este proyecto nace de la necesidad de reducir el costo de la automatización de un proceso, permitiendo a una mayor cantidad de usuarios utilizar un modelo alternativo de PLC en las aplicaciones que requieran. El proyecto consiste en adaptar la placa programable “Arduino” para que funcione bajo los mismos parámetros que un PLC comercial, por ejemplo, tensión de entrada (0 – 24VDC), corriente de entrada (4 – 20mA), tensión en la salida

(0 – 250VAC; 0 – 30VDC) y corriente en la salida (0 – 10A) esto se logra mediante el uso dispositivos electrónicos básicos y de potencia, incluyendo también aislamiento galvánico para la protección del Arduino, previniendo el traspaso de portadores de carga.

c. Tecnológica

El trabajo tiene justificación tecnológica dado que el desarrollo en la tecnología de Arduino, permite que, hoy en día, se pueda utilizar para entablar comunicación en base a protocolos industriales, tales como ETHERNET, MODBUS, RS232 entre otros.

1.5. Limitaciones y Facilidades

1.5.1. Limitaciones

- Con el fin de facilitar el diseño del autómata y así poder enfocarlo también en la comunicación industrial, solo se utilizarán pines digitales, dejando abierta a un futuro diseño utilizando los pines analógicos.

- Al programar la placa en el lenguaje Arduino, la ejecución del programa, es decir, la lectura de las entradas y la escritura de las salidas, se da en serie (una a continuación de otra), mientras que en un PLC comercial este proceso se da de manera paralela.

- Cuando se programa la placa en KOP (ladder), se omiten algunas funciones especiales, según el software utilizado.

1.5.2. Facilidades

- Menor costo en comparación a los modelos de PLCs de características similares ofertados en la actualidad.

- Mayor cantidad de proyectos realizables debido al número de shields compatibles.

- Sencillez para programar mediante el lenguaje de programación Arduino.

- Compatibilidad con diversos Software usados en ingeniería, por ejemplo, el uso de Arduino junto a LabView, software en el cual la conexión con el Arduino es gratuita.

- Facilidad de comunicación con distintos protocolos.

CAPITULO II

II. FUNDAMENTO TEÓRICO

2.1. Antecedentes de la investigación

A manera de presentación de los antecedentes o datos vinculados, podemos mencionar ciertos desarrollos previos relacionados a esta investigación, dos desarrollados por los autores y uno por otros autores.

Caso_1 "Adaptación de la tarjeta Arduino mediante una interfaz electrónica para ser usada con un PLC y en lazados a LabView"

En el año 2016, uno de los autores de esta tesis, presentó en el congreso llamado "EXPOTRON" llevado a cabo en la Universidad Nacional de San Agustín – UNSA, un proyecto de investigación titulado: **"Adaptación de la tarjeta Arduino mediante una interfaz electrónica para ser usada con un PLC y en lazados a LabView"**, obteniendo el 3er lugar, a continuación, presentamos algunos extractos de dicho trabajo de investigación:

RESUMEN

En la actualidad, es necesario que uno de los principales objetivos de las instituciones educativas, que incluyen en su currícula cursos referidos a Control y Automatización, sea el impulsar la enseñanza de la automatización industrial utilizando Controladores Lógicos Programables (PLC), los cuales son dispositivos que se crearon a mediados del siglo XX, para solventar algunas deficiencias de la lógica cableada.

Ante la dificultad de contar con procesos industriales reales en la mayoría de instituciones educativas, se pretende simular dichos procesos para que sean utilizados por los estudiantes y así implementar el uso del controlador Siemens LOGO. La simulación de las plantas virtuales se lleva a cabo con la ayuda de LabView (Laboratory Virtual Instrumentation Engineering Workbench), el cual es un lenguaje de programación grafico desarrollado por NI (National Instruments) en los años setenta.

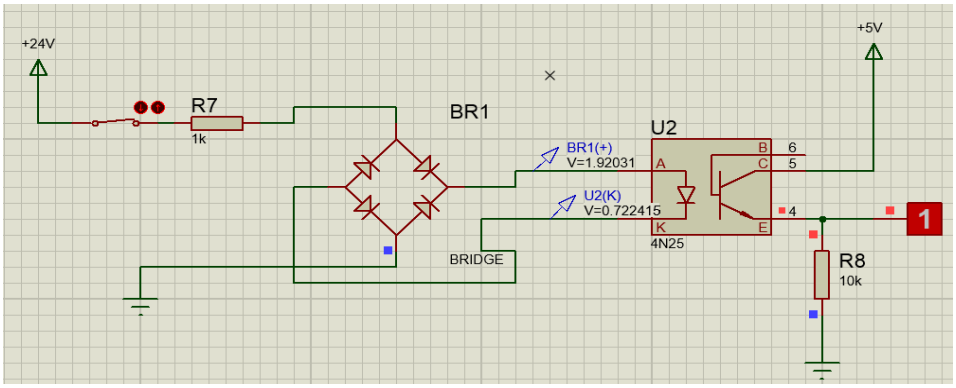
Con la elaboración de este proyecto se procura que las instituciones de enseñanza de Automatización y Control cuenten con suficientes recursos para que sus estudiantes puedan experimentar de la mejor manera el uso de los PLC.

El enlace del PLC con la computadora en la cual se programan las simulaciones será realizado a través de la tarjeta programable Arduino la cual será adaptada para trabajar a un mayor nivel de tensión.

Comunicación PLC – Arduino:

En cuanto al desarrollo del proyecto, se ha analizado las posibles maneras de enlazar el PLC al Arduino, la más óptima, de manera simulada, resulto ser el acople de un puente de diodos junto a circuito de regulación de tensión, para rectificar las salidas en caso sean de corriente alterna y finalmente un optoacoplador con el fin de aislar y proteger al Arduino, en caso las salidas solo sean de corriente continua solo será necesario un circuito STEP-DOWN. El circuito simulado se adjunta a continuación:

Figura N° 2. 1 Circuito de Adaptación de entrada del Arduino



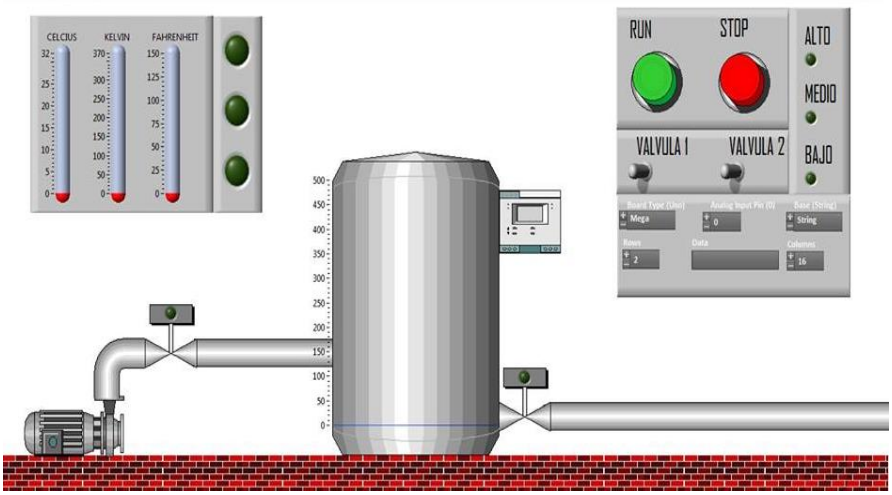
La comunicación PLC – Arduino puede ser reemplazada por códigos de programación, dado que el LabView puede comunicar de manera directa el estado actual de proceso.

Simulación de Plantas:

Se presentan las plantas desarrolladas de manera simulada hasta el momento:

Planta N°1 Monitoreo de temperatura y control de válvulas

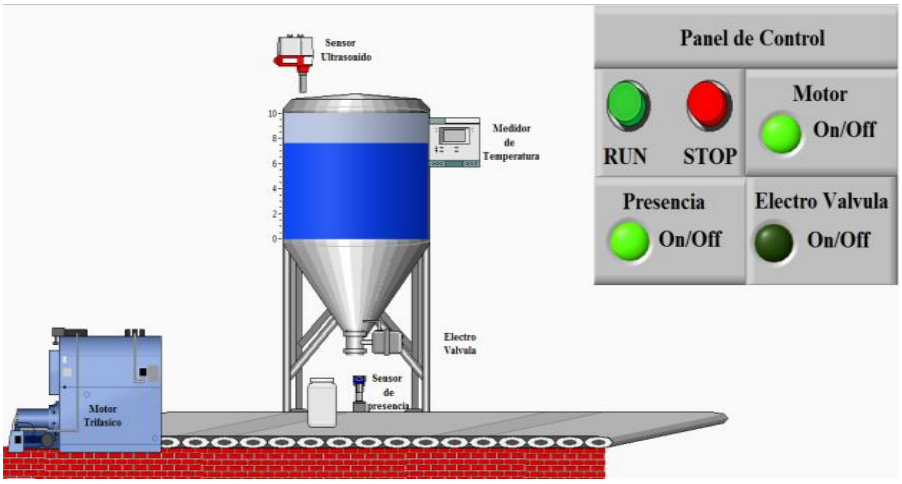
Figura N° 2. 2 Panel Frontal de la planta Numero 1



Este es un proceso industrial simple donde se realizan activaciones de válvulas y monitoreo de temperatura mediante una planta simulada en LabView, pero con elementos en físico para la medición de la tempera, utilizaremos una tarjeta de adquisición de datos (Arduino-Mega2560).

Planta N°2 Llenado de envases

Figura N° 2. 3 Panel Frontal de la planta Numero 2



En este proceso vamos a realizar el llenado de envases para lo cual utilizaremos los siguientes elementos: Motor trifásico, sensor de presencia, sensor ultrasonido y un medidor de temperatura.

La planta será simulada en LabView y contará con un panel de control para monitorear algunas funciones.

Caso_2 "Diseño e implementación de un Controlador Lógico Programable (PLC), de salida tipo relé, basado en Arduino"

En el año 2016, los autores de esta tesis, presentaron en el congreso internacional llamado "CONEIMERA" llevado a cabo en la Universidad Nacional de Pirua – UNP, un proyecto de investigación titulado: **"Diseño e implementación de un Controlador Lógico Programable (PLC), de salida tipo relé, basado en Arduino"**, obteniendo el 2do lugar.

Este último antecedente fue la base principal para el desarrollo de este trabajo, dando otra orientación yendo más allá de su diseño, enfocándose en más aspectos de la automatización.

2.2. Antecedente externo:

Programación de la tarjeta Arduino con el lenguaje de programación industrial LADDER (KOP):

En la presentación del sistema que los autores bautizaron como *PLCmDUINO,* detallaron lo siguiente:

Actualmente la tarjeta Arduino se ha convertido en uno de los sistemas de adquisición de datos y señalización digital más populares, no sólo entre la comunidad estudiantil sino también entre aficionados, técnicos e ingenieros en electrónica, mecatrónica y otras áreas afines. El lenguaje nativo para programar esta tarjeta es una versión de C++ que se llama Processing y en ocasiones, para los que se inician en el estudio de la programación y de las aplicaciones de Arduino, el aprender a programar en este lenguaje puede resultar todo un reto, sobre todo para aquellos que no tienen experiencia previa

en la realización de programas utilizando algún otro lenguaje de programación textual

Por otro lado, para una gran mayoría de nosotros la programación de cualquier dispositivo programable se vuelve más comprensible cuando se utiliza un lenguaje gráfico de programación. Por ejemplo, los lenguajes KOP (Kontaktplan - Esquema de Contactos) y GRAFCET son dos de los lenguajes gráficos que más se utilizan para programar los PLC (Programmable Logic Controller – Controlador Lógico Programable). Estos dispositivos son autómatas programables que se utilizan comúnmente para controlar prácticamente cualquier tipo de proceso en la industria. En particular, el lenguaje KOP o ladder (escalera) es un lenguaje gráfico que es muy similar a los esquemas de la lógica cableada que se utilizan a nivel industrial para representar la conexión eléctrica de dispositivos electro mecánicos que se conectan entre sí para crear automatismos. De hecho, estos esquemas han facilitado desde hace algunas décadas a la fecha, a técnicos e ingenieros, la realización de estrategias de control que se utilizan en la automatización industrial. Incluso, mucho antes de la llegada de los lenguajes de alto nivel de nuestros días.

Para facilitar la programación de la tarjeta Arduino a aquellas personas que no tienen experiencia previa de programación en C++ pero que tienen conocimientos de los lenguajes que se utilizan para programar los PLC, se ha desarrollado un sistema de programación al que he denominado como PLCmDuino el cual permite programar la tarjeta Arduino en lenguaje KOP o

AWL, sin requerir el uso del IDE (Integrated Development Environment - Entorno de Desarrollo Integrado) original de Arduino que utiliza el lenguaje C++ como plataforma de programación.

En el **Anexo 1** mostramos el desarrollo de esta investigación ya que la consideramos relevante para futuras aplicaciones de nuestra investigación.

2.3. Marco Teórico

2.3.1. Arduino:

Arduino es una plataforma de código libre para la creación de prototipos electrónicos, basada en software y hardware flexibles, fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquier interesado en crear entornos u objetos interactivos capaces de transmitir datos de una gran variedad de interruptores y sensores; y controlar multitud de tipos de luces, motores y otros dispositivos físicos.

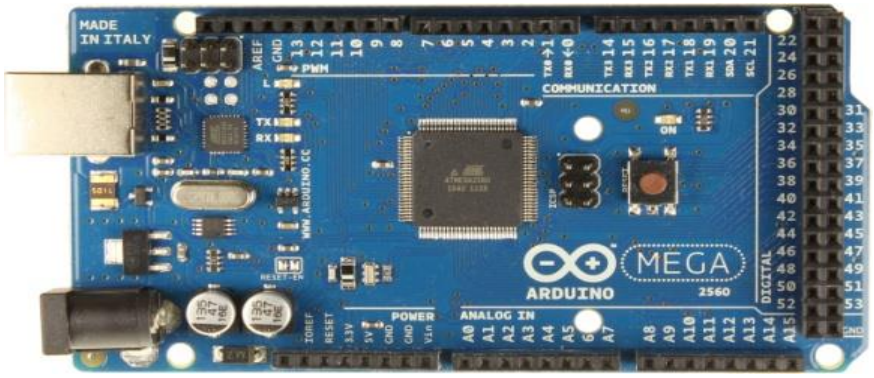
Arduino posee una gran cantidad de shields (módulos) que expanden la cantidad de aplicaciones en las que puede emplearse la placa.

a. Arduino Mega 2560:

Se utiliza Arduino Mega 2560 cuando se requiera una mayor cantidad de pines que el Arduino UNO, además de mayor frecuencia de operación y el tamaño no sea un limitante.

Asimismo, Arduino Mega 2560 es una placa que nos permite interactuar con diversos periféricos y módulos especiales (shields) tales como GSM, GPS, entre otros, en esta investigación hablaremos un poco más acerca del módulo ETHERNET.

Figura N° 2. 4 Vista superior de un Arduino Mega 2560



El Arduino Mega 2560 es una placa electrónica basada en el Atmega2560 . Cuenta con 54 pines digitales de entrada / salida (de las cuales 15 se pueden utilizar como salidas PWM), 16 entradas analógicas, 4 UARTs (puertos serie de hardware), un oscilador de 16MHz, una conexión USB, un conector de alimentación, una cabecera ICSP, y un botón de reinicio. Contiene todo lo necesario para apoyar el microcontrolador; basta con conectarlo a un ordenador con un cable USB o la corriente con un adaptador de CA a CC o una batería para empezar.

Podemos observar la hoja técnica del Arduino Mega 2560 en el **Anexo 2**

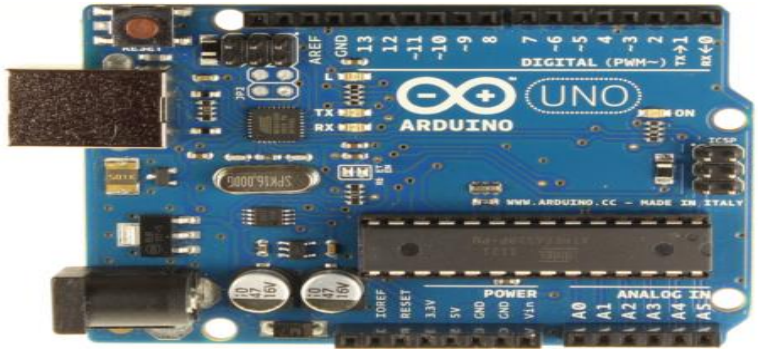
Tabla N° 2. 1 Especificaciones Técnicas del Arduino Mega 2560

Microcontrolador	Atmega 2560
Tensión de Funcionamiento	5V
Voltaje de entrada	6 – 20 V
E/S digitales	54 (de las cuales 15 proporcionan salida PWM)
Pines de entrada Analógica	16
Corriente Continua para E/S	20 mA
Memoria Flash	256KB
SRAM	9KB
EEPROM	4KB
Frecuencia de Operación	16MHz

b. Arduino Uno:

Se utiliza Arduino Uno, dado que a pesar de no contar con la cantidad de pines que posee el Arduino Mega 2560, es compatible con una gran variedad de shields

Figura N° 2. 5 Vista superior de un Arduino UNO



Arduino Uno es una placa electrónica basada en el ATmega328P. Cuenta con 14 pines digitales de entrada / salida (de los cuales 6 se podrán utilizar como salidas PWM), 6 entradas analógicas, un cristal de cuarzo de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP y un botón de reinicio. Contiene todo lo necesario para que el microcontrolador funcione correctamente; basta con conectarlo a un ordenador mediante un cable USB o mediante una fuente de corriente mediante un adaptador de CA a CC o una batería para empezar.

Podemos observar la hoja técnica del Arduino UNO en el **Anex 3.** Cabe resaltar que esta versión es una de las más usadas para iniciar con Arduino.

Tabla N° 2. 2 Especificaciones Técnicas del Arduino Uno

Microcontrolador	Atmega 328P
Tensión de Funcionamiento	5V
Voltaje de entrada	6 – 20 V
E/S digitales	14 (de las cuales 6 proporcionan salida PWM)
Pines de entrada Analógica	6
Corriente Continua para E/S	20 mA
Memoria Flash	32KB
SRAM	2KB
EEPROM	1KB

Frecuencia de Operación	16MHz

c. Arduino Nano:

Utilizaré Arduino Nano porque posee la misma cantidad de pines que el Arduino UNO, pero en menor tamaño y costo, además de haber aplicaciones en las cuales no es necesario utilizar un shield.

Figura N° 2. 6 Vista superior de un Arduino Nano

El Arduino Nano es una pequeña versión de la placa Arduino Uno,por lo tanto, también está basada en el ATmega328 (Arduino Nano3.x) oATmega168 (Arduino Nano 2.x). Tiene más o menos la misma funcionalidad de la Arduino UNO, pero en un paquete diferente. Carece de una sola toma de corriente continua, y funciona con un cable USB Mini-B en lugar de una normal, ofrece la facilidad de ser colocada en una PROTOBOARD, para realizar pruebas.

Podemos observar la hoja técnica del Arduino NANO en el **Anexo 4.**

Tabla N° 2. 3 Especificaciones Técnicas del Arduino Nano

Microcontrolador	Atmega 168 o Atmega 328
Tensión de Funcionamiento	5V
Voltaje de entrada	6 – 20 V
E/S digitales	54 (de las cuales 15 proporcionan salida PWM)
Pines de entrada Analógica	8
Corriente Continua para E/S	40 mA
Memoria Flash	16KB / 32KB
SRAM	9KB
EEPROM	512 bytes / 1KB
Frecuencia de Operación	16MHz

2.3.2. Módulos Ethernet para Arduino:

En la actualidad en el mercado existen diferentes módulos para la conexión a internet de una placa Arduino, en esta parte de la investigación hablaremos de dos:

a. Shield Ethernet para Arduino

El Arduino Ethernet Shield V1 conecta la placa Arduino a Internet en cuestión de minutos. Simplemente conecte este módulo a su placa Arduino, conéctelo a su red con un cable RJ45 y siga unas simples instrucciones para comenzar a controlar su mundo a través de Internet.

El Arduino Ethernet Shield V1 permite que una placa Arduino se conecte a Internet. Se basa en el chip ethernet **Wiznet W5100.** El

Wiznet W5100 proporciona una pila de red (IP) capaz de TCP y UDP. Admite hasta cuatro conexiones de socket simultáneas. Utilice la biblioteca de Ethernet para escribir bocetos que se conectan a Internet utilizando el SHIELD. El shield de Ethernet se conecta a una placa Arduino utilizando cabezales de envoltura de alambre largo que se extienden a través del escudo. Esto mantiene intacto el diseño del pasador y permite apilar otro escudo en la parte superior.

Figura N° 2. 7 Vista superior de un shield Ethernet

Con este shield se abren innumerables opciones para controlar tu Arduino a través de Internet o de la LAN de tu casa. Domótica, automatización, Internet de las cosas (IoT), control y monitoreo remoto, etc, son algunos de los campos donde se puede utilizar este shield. Es compatible con el Arduino Uno y el Mega, además las librerías Ethernet y SD vienen incluidas en el IDE de Arduino, por lo que no hay necesidad de descargarlas.

Esta última versión incluye un slot para tarjetas micro-SD, el cual puede ser empleado para almacenar archivos que podrás poner disponibles a través de la red. En esta revisión del shield también se incluye un controlador de reset, esto es para asegurarse que el módulo W5100 Ethernet inicie correctamente al energizarlo. Revisiones previas de este shield no eran compatibles con el Mega y necesitaban reset manual después de conectarlo.

El Arduino Uno utiliza los pines digitales 11, 12 y 13 (SPI) para comunicarse con este shield. El Mega emplea los pines 50,51 y 52. En ambas tarjetas (Uno y Mega) el pin 10 es empleado para seleccionar el W5100 y el pin 4 para la tarjeta SD. Mientras emplees las funcionalidades de ethernet estos pines no estarán disponibles. Así mismo tomar en cuenta que en el Arduino Mega a pesar de que el pin 53 (SS pin) no es empleado para seleccionar ni el W5100 ni la tarjeta debe ser configurado como una salida ya de otra manera la interfaz SPI no funcionará.

Debido a que el W5100 y la tarjeta SD comparten el mismo bus SPI, solo uno podrá estar activo a la vez. Si estás usando ambos periféricos en tu programa, debes tomar en cuenta esto. Si no estás empleando alguno de estos periféricos debes "deseleccionarlo" explícitamente. Para deshabilitar la tarjeta SD, configura el pin 4 como salida y escríbele HIGH. Para deshabilitar el W5100 configura el pin 10 como una salida en HIGH.

El shield posee un conector RJ45 estándar para ethernet. El botón de reset inicializa tanto el shield como el Arduino. Una gran ventaja de este shield es que es apilable por lo que podrás disponer de todos sus pines en otros shields.

Tabla N° 2. 4 Especificaciones Técnicas del Shield Ethernet

CHIP	W5100
Tensión de Funcionamiento	5VDC
Velocidad Ethernet	10/100 Mbps
Interface	SPI

Asimismo, este *shield Ethernet* es compatible con Arduino Uno, Mega, Leonardo. También incluye un lector MicroSD Card.

Podemos observar su hoja técnica en el **Anexo 5.**

b. Módulo Ethernet ENC28J60

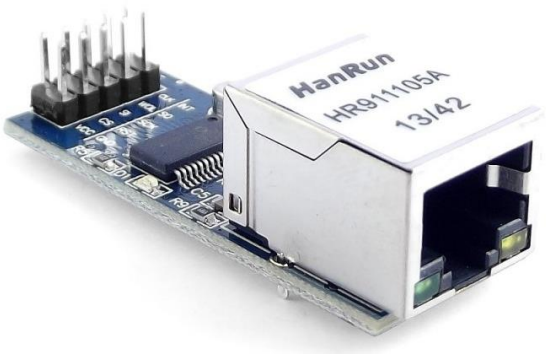
El ENC28J60 es un controlador de Ethernet diseñado para sistemas embebidos fabricado por Microchip Technology Inc. Podemos usar el ENC28J60 junto a un procesador como Arduino para conectar nuestros proyectos de electrónica y robótica con Internet.

El ENC28J60 se controla a través de bus SPI, por lo que la conexión con Arduino es muy sencilla. El ENC28J60 opera a 3.3, pero es tolerante a señales de 5V, por lo que su integración es aún más sencilla.

El ENC28J60 soporta velocidades de 10Mbits/s y los modos Dúplex (Full-Duplex) y Semi-dúplex (Half-Duplex) con detección y corrección automática de la polaridad. El ENC28J60 cumple con las especificaciones IEEE 802.3 10BASE-T.

El ENC28J60 incorpora filtrado de paquetes para limitar el número de paquetes entrantes, un módulo DMA interno para facilitar el flujo de datos y hardware específico para el cálculo de las sumas de control (IP checksums).

Figura N° 2. 8 Modulo Ethernet ENC28J60



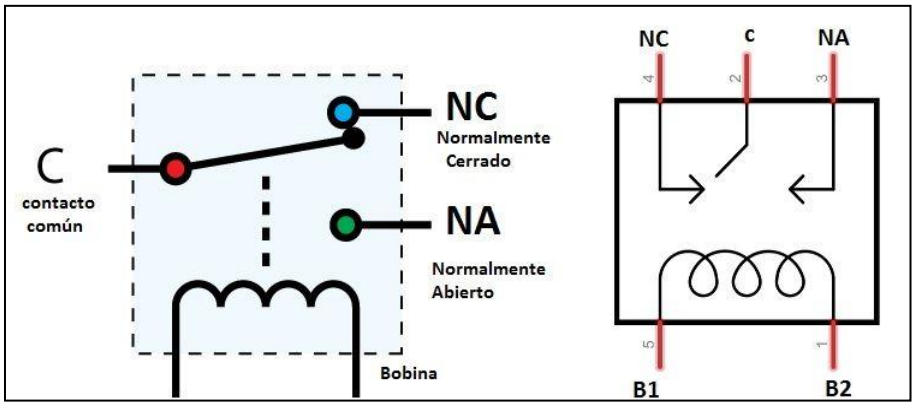
El ENC28J60 es uno de los procesadores más baratos para dotar conectividad a nuestros proyectos, y es más barato que otras alternativas como el W5100.

Sin embargo, el ENC28J60 carece de una pila de TCP/IP por hardware como sí que incluye el W5100. Por tanto, su uso es más complejo y requiere una mayor carga del procesador. Podemos observar su hoja técnica en el **Anexo 6.**

2.3.3. Relé Electromagnético:

El relé o relevador electromagnético funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes.

Figura N° 2. 9 Esquema interno de un relé electromagnético



a. Funcionamiento del Relé.

Vemos que el relé de la figura 7 tiene 2 contactos, una abierto (NC) y otro cerrado (NO) (pueden tener más). **Cuando metemos corriente por la bobina**, esta crea un campo magnético creando **un electroimán que atrae los contactos haciéndolos cambiar de posición**, el que estaba abierto se cierra y el que estaba normalmente cerrado se abre. El contacto que se mueve es el C y es el que hace que cambien de posición los otros dos.

Como ves habrá un circuito que activa la bobina, llamado de control, y otro que será el circuito que activa los elementos de salida a través de los contactos, llamado circuito secundario o de fuerza.

Los relés Pueden tener 1, 2 3 o casi los que queramos contactos de salida y estos puede ser normalmente abiertos o normalmente cerrados (estado normal = estado sin bobina energizada).

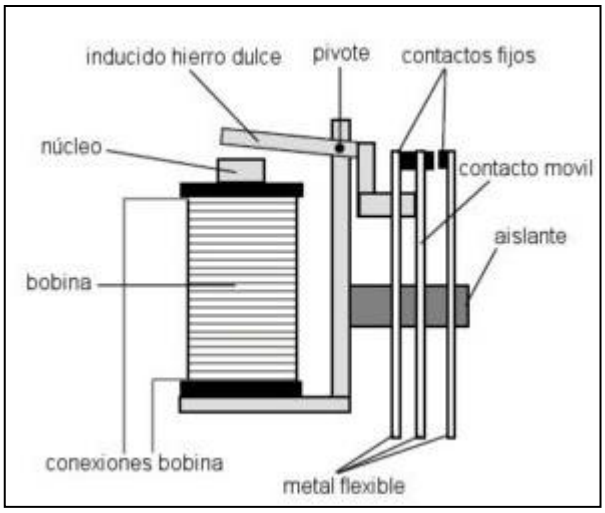
Los relés eléctricos son básicamente interruptores operados eléctricamente que vienen en muchas formas, tamaños y potencias adecuadas para todo tipo de aplicaciones. Los relés también pueden ser relés de potencia, más grandes y utilizados para la tensión mayores o aplicaciones de conmutación de alta corriente. En este caso se llaman Contactores, en lugar de relés. Los contactores son utilizados en aplicaciones industriales, sobre todo en aplicaciones para controlar motores trifásicos.

b. Partes de un relé electromagnético.

El relé electromagnético es uno de los relés más utilizados, las partes principales son:

La bobina - La bobina de este relé es la encargada de generar una corriente inducida en el bobinado crear un campo magnético.

Figura N° 2. 10 Partes de un relé electromagnético



El relé electromagnético es uno de los relés más utilizados, las partes principales son:

La bobina - La bobina de este relé es la encargada de generar una corriente inducida en el bobinado crear un campo magnético.

Conexiones de la bobina - Mediante estas conexiones daremos tensión a la bobina, normalmente serán tensiones de 12 voltios o 24 voltios en corriente continua.

Núcleo - El núcleo está situado en el interior de la bobina y se magnetiza con la intención de atraer la parte metálica llamada hierro inducido.

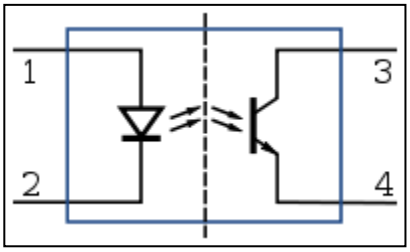
Hierro inducido - El hierro inducido se moverá atraído por el núcleo y provocará la unión de los contactos abiertos.

Contactos abiertos - Los contactos abiertos los utilizaremos para dar tensión al receptor que queramos hacer actuar.

2.3.4. Optoacoplador:

Un optoacoplador, también llamado optoaislador o aislador acoplado ópticamente, es un dispositivo de emisión y recepción que funciona como un interruptor activado mediante la luz emitida por un diodo LED que satura un componente optoelectrónico, normalmente en forma de fototransistor o fototriac. De este modo se combinan en un solo dispositivo semiconductor, un fotoemisor y un fotorreceptor cuya conexión entre ambos es óptica. Estos elementos se encuentran dentro de un encapsulado que por lo general es del tipo DIP. Se suelen utilizar para aislar eléctricamente a dispositivos muy sensibles.

Figura N° 2. 11 Esquema interno de un optoacoplador



a. Funcionamiento de un Optoacoplador:

Activamos una luz y esta luz llega a un detector que genera una tensión de salida, interruptor cerrado. Si no se activa la luz o no le llega la luz al detector, este no genera ninguna tensión de salida, es decir interruptor abierto.

Figura N° 2. 12 Funcionamiento de un optoacoplador

Si combinamos una fuente óptica (generalmente un Led) con algún tipo de detector óptico (generalmente un semiconductor de silicio llamado fototransistor) en un solo encapsulado, el dispositivo resultante es un optoacoplador o interruptor óptico.

Suelen ser elementos que sustituyen a los relés tradicionales. Se suelen utilizar para aislar dos circuitos, uno que trabaja a poca tensión (el del LED), llamado de control y otro a mucha tensión o a una tensión diferente (el del detector) llamado de potencia.

Imagina que con una pequeña tensión activamos el LED del optoacoplador (por ejemplo, a 5V) y la luz que emite el led llega al detector del optoacoplador y activa el detector creando una tensión de salida a 220V.Podemos activar a la salida motores, lámparas, etc. a 220V desde otro sitio en el que solo tenemos 5V, sin riesgo apenas para el que lo activa.

La aplicación principal es en aislamiento entre los circuitos de control y los de potencia. Esto evita que la parte de trabajo (la

del led) no tengan casi riesgos para el que opera en ella, al no tener que trabajar con la parte de alta tensión o intensidad, que estaría separada. Veamos cómo funcionan. Otros usos muy comunes en educación son en coches seguidores de luz.

¿Cómo Funciona?

Tiene una salida de luz (LED emisor) y una entrada de luz, que detecta cuando recibe la luz del LED cuando esta rebota contra alguna superficie (fotodetector). Como ves es similar al transistor, pero en lugar de corriente con luz.

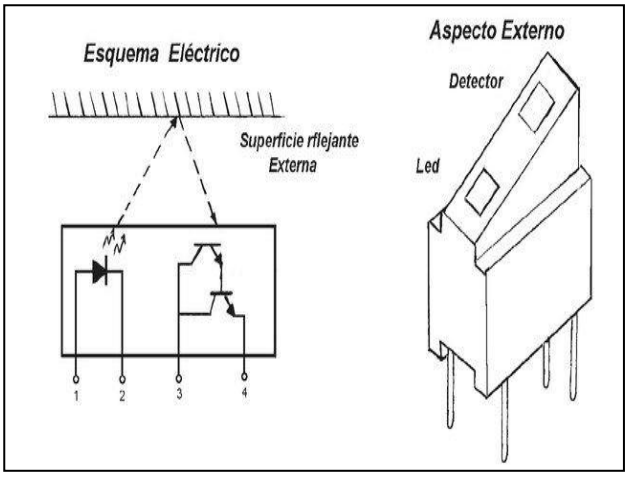
Cuando le llega una señal eléctrica (tensión) a los dos extremos del LED (emisor) este emite una señal luminosa, que recibe el receptor o detector. Este al recibir esta señal luminosa genera en sus bornes (patillas) una tensión eléctrica, que será la tensión de salida.

Como vemos cuando le llega una tensión a la entrada se genera una luz y al recibirla el detector este genera una tensión de salida. Es como un interruptor. Si no llega luz al detector el interruptor estará abierto, si le llega luz del led el interruptor sería cerrado.

OJO podría estar el led encendido, pero no llegarle luz al detector, por ejemplo, porque no rebota en ninguna superficie. El

interruptor estaría abierto porque no se produce tensión a la salida.

Figura N° 2. 13 Aspecto externo de un optoacoplador



Algunos optoacopladores tienen un encapsulado con una cámara de aire para la transmisión de la luz. En este tipo si hay algún objeto dentro de la ranura no llegará luz al detector. También puede ser que no le llegue tensión al led y tampoco tendríamos tensión de salida. Serían los 2 casos posibles.

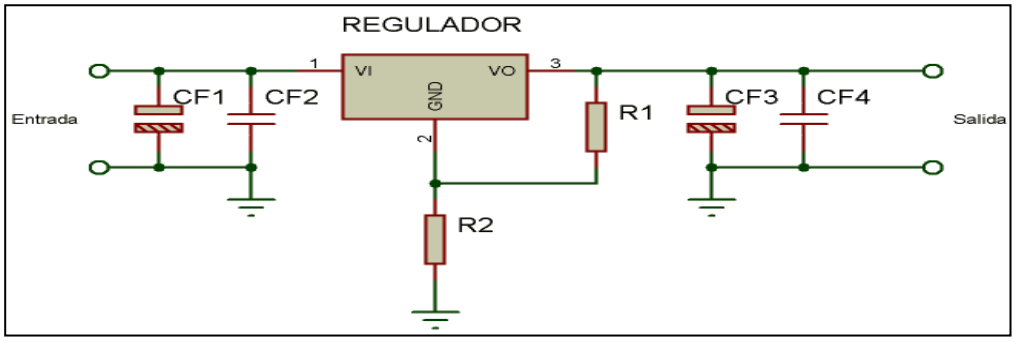
2.3.5. Regulador de Tensión:

Un regulador de tensión o regulador de Voltaje es un dispositivo electrónico diseñado para mantener un nivel de tensión constante.

Los reguladores electrónicos de tensión se encuentran en dispositivos como las fuentes de alimentación de los computadores, donde estabilizan las tensiones de corriente continua usadas por el procesador.

En los alternadores de los automóviles y en las plantas generadoras, los reguladores de tensión controlan la salida de la planta.

Figura N° 2. 14 Circuito esquemático de un regulador de voltaje



a. Función.

La función de un regulador de tensión es proporcionar una tensión estable y bien especificada para alimentar otros circuitos a partir de una fuente de alimentación de entrada de poca calidad; después del amplificador operacional, el regulador de tensión es probablemente el circuito integrado más extensamente usado. Además, deben ser capaces de proporcionar corrientes de salida desde unas cuantas decenas de miliamperios, en el caso de reguladores pequeños, hasta varios amperios, para reguladores grandes.

b. Clasificación

Reguladores en serie o lineales: Controlan la tensión de salida ajustando continuamente la caída de tensión en

un transistor de potencia conectado en serie entre la entrada no regulada y la carga. Puesto que el transistor debe conducir corriente continuamente, opera en su región activa o lineal. Aunque son más sencillos de utilizar que los reguladores de conmutación, tienden a ser muy ineficientes debido a la potencia consumida por el elemento en serie. Su eficiencia es alrededor del 20% y solamente resultan eficaces para baja potencia (< 5 W).

Reguladores de conmutación: Utilizan un transistor de potencia como conmutador de alta frecuencia, de tal manera que la energía se transfiere desde la entrada a la carga en paquetes discretos. Los pulsos de intensidad se convierten después a una corriente continua mediante un filtro inductivo y capacitivo. Puesto que, cuando opera como conmutador, el transistor consume menos potencia que en su región lineal, estos reguladores son más eficientes (hasta el 80%) que los lineales; además, son más pequeños y ligeros.

Estos reguladores se pueden diseñar para operar directamente sobre la tensión de la red rectificada y filtrada, eliminando la necesidad de utilizar transformadores voluminosos. El precio que se paga por estas ventajas es una mayor complejidad del circuito y un mayor ruido de rizado. Los reguladores de conmutación se utilizan especialmente en sistemas digitales, donde a menudo es mucho más importante una alta eficiencia y un peso bajo que un rizado de salida pequeño.

Los reguladores de ajustables de tres terminales: permiten ajustar la tensión de salida a partir de resistencias externas conectadas al terminal denominado ADJUSTMENT o ADJ. Uno de los más utilizados es el LM317 (positivo) y el LM337 (negativo) de la National Semiconductor capaces de proporcionar hasta 1.5 A de corriente de salida, otros ejemplos de estos reguladores son el LM338 de la misma fabrica cuya corriente alcanza hasta 5 A, LT1038 de Linear Technology y LM896 de 10 A de salida

Algunas familias de reguladores de tensión tienen la particularidad de desperdicio de energía al emitir demasiado calor.

c. Circuitos de Protección

Los reguladores están equipados con un circuito de protección cuyo propósito es limitar la corriente del elemento en serie (o incluso anularla). Los circuitos de protección se diseñan para estar inactivos bajo condiciones de operación normal y activarse tan pronto como se intente exceder el correspondiente límite de seguridad. El propósito del circuito de protección contra sobrecarga es evitar que la corriente que circula por el transistor en serie exceda un nivel de seguridad predeterminado, como sucedería, por ejemplo, en el caso de cortocircuitar la salida.

d. Consideraciones

Los reguladores lineales ofrecen simplicidad y bajo costo para potencias por debajo de 25W, además de ello nos brindan una estupenda regulación de línea y de carga (0.1%), pequeños rizados y rápidos tiempos de respuesta que están por debajo de los 20s, todo esto a cambio de la necesidad imperiosa de utilizar transformadores de aislamiento de baja frecuencia (60Hz), de las elevadas pérdidas y del bajo rendimiento (30 a 50 %) y densidad de potencia (0.5W/in3).

Los reguladores conmutados nos brindan un alto rendimiento debido a la condición de conmutación de su transistor de potencia (60-90%), tiene mayor tolerancia a las variaciones de línea, una gran densidad de potencia (15 W/in3), y un tamaño reducido, sin embargo, nos presenta mayor grado de complejidad, tiene grandes interferencias electromagnéticas, regulación de carga y de línea muy pobres y un alto nivel de rizado y ruido.

Los reguladores ajustables son benignos en la posibilidad de obtener una variedad casi infinita de tensiones de salida gracias a que su realimentación y muestreo se realizan externamente. Sin embargo, las corrientes de salida son excesivamente bajas pues están por debajo de 200 mA.

2.3.6. Modelo OSI:

Modelo OSI consiste en un estándar que regula la comunicación entre distintos sistemas.

El modelo OSI abarca una serie de eventos importantes:

- El modo en que los datos se traducen a un formato apropiado para la arquitectura de red que se está utilizando.

- El modo en que las computadoras u otro tipo de dispositivo de la red se comunican. cuando se envíen datos tiene que existir algún tipo de mecanismo que proporcione un canal de comunicación entre el remitente y el destinatario.

- El modo en que los datos se transmiten entre los distintos dispositivos y la forma en que se resuelve la secuenciación y comprobación de errores.

- El modo en que el direccionamiento lógico de los paquetes pasa a convertirse en el direccionamiento físico q proporciona la red.

Está compuesto por siete capas distintas, como se muestra en la **Figura 2.15.**

Figura N° 2. 15 Capas del modelo OSI

Cada capa tiene su función y debe prestar unos servicios que serán usados por la capa superior. Existen dos tipos de comunicación dentro del modelo OSI, la comunicación horizontal entre dos equipos en la misma capa y la comunicación vertical entre una capa y su superior.

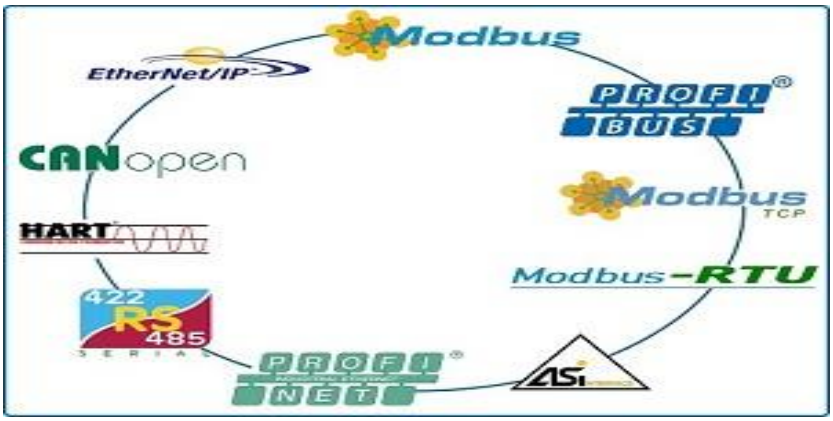
- **Capa Física:** transmite la información entre dos equipos.
- **Capa Enlace de Datos:** estructura la información bajo una trama.
- **Capa de Red:** recibe y envía las tramas.
- **Capa de Transporte:** controla el flujo de comunicaciones.
- **Capa de Sesión:** sincroniza los equipos.
- **Capa de Presentación:** decodifica los datos recibidos.

- **Capa de Aplicación:** transfiere los archivos.

2.3.7. Protocolos de Comunicación:

Para conectar dos partes de una maquinaria es necesario tener una comunicación entre ellas. La comunicación está compuesta de dos partes, una parte física o HARDWARE y una parte de programación o FIRMWARE. El HARDWARE está compuesto por los materiales necesarios para la comunicación, ya sea el cableado, el tipo de conector o el conversor, mientras que el FIRMWARE está compuesto por las instrucciones necesarias o tramas para el funcionamiento de la máquina.

Figura N° 2. 16 Protocolos de comunicación industrial

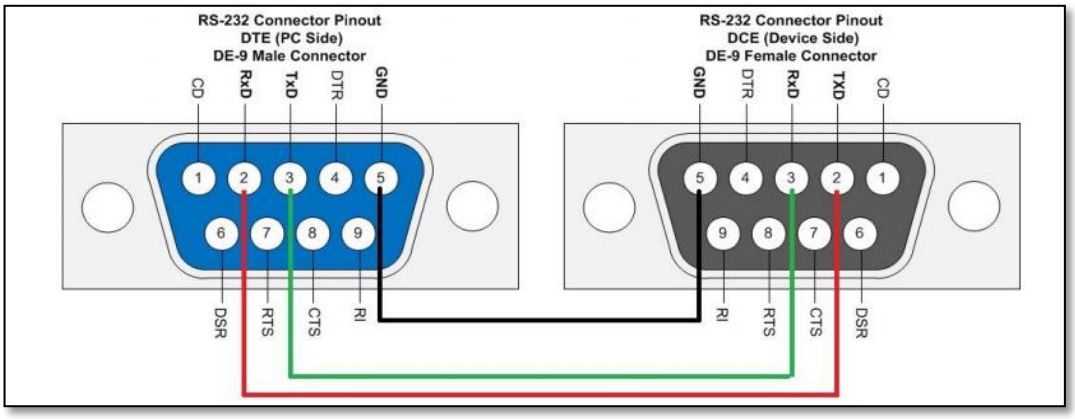


a. Tipos de conexiones en la industria:

- **RS-232.** Este tipo de comunicación fue diseñado para la comunicación punto a punto, donde un equipo hace la función de maestro transmitiendo hacia un equipo esclavo ubicado como máximo a una distancia de 15 metros y a una velocidad máxima de

19,200 bps. El tipo de comunicación está configurado como "single ended" ya que usa la misma masa (GND). Este método es vulnerable al ruido y por ello se emplea en comunicaciones de distancias cortas. En la trasmisión RS232 normalmente las tramas se envían como un conjunto de caracteres ASCII, incluyendo letras, números y caracteres especiales. Esta comunicación consta de la señal de trasmisión TX, recepción RX y tierra GND, En la siguiente imagen se observa el conector DB9 y la asignación de señales, como se muestra en la **Figura 2.17**.

Figura N° 2. 17 Conexión puerto DB-9



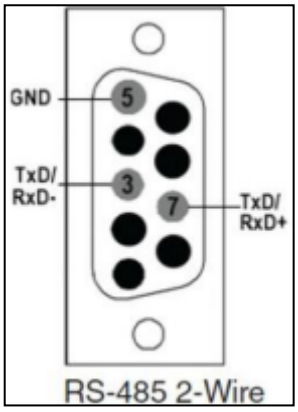
La trasmisión de datos mediante TX y RX se trata de una señal serial bipolar, normalmente entre +10 y -10 volts. Normalmente va acompañado del circuito MAX232 que permite descomponer el carácter ASCII en un byte para su envío, usando el modo inverso para su recepción. Por ejemplo, para la transmisión del numero ASCII “1” se trasmitirá el byte 00110001, Además incluye un bit de

Inicio y un bit de Paro. El bit de inicio (Start bit) se encarga de indicar al receptor cuándo debe empezar a leer la trama mediante un flanco ascendente. Por otro lado, se añade el bit de Paro al final de la trama, para indicar al receptor que ha terminado la trama, esto se efectúa mediante un flanco descendente. Como la comunicación no necesita una señal de sincronización se trata de una transmisión "Asíncrona".

- **RS – 485** Para una comunicación a mayores distancias y velocidades de transmisión se utiliza la norma RS485. Esta norma también permite la trasmisión multipunto, es decir un ordenador central conectado con varias UTR. Este tipo de transmisión se denomina trasmisión diferencial y permite velocidades de hasta 10 Mbps, sobre distancias de hasta 1.3 KM. En este tipo de transmisión se usan dos señales para trasmitir y dos para recibir, además de la tierra, la cual se suele conectar a la malla del cable. En la transmisión se envía la señal de trasmisión y su complemento, mientras que, en el receptor, la señal original se obtiene restando las dos señales, lo cual ayuda a reducir notablemente el ruido generado en la línea, ya que éste se adhiere en ambas líneas por igual, con lo que se consigue cancelarlo al restar ambas señales. Para la comunicación por RS485 se usan 2 señales y el GND y es necesario un protocolo "Half dúplex", ya que las líneas son usadas tanto para la

trasmisión como para la recepción, a continuación, se detalla en la **Figura 2.18**.

Figura N° 2. 18 Conexión puerto RS-485



Todos los dispositivos RS485 deben desconectarse de la red una vez acaben de transmitir la trama para que puedan usarse como recepción. Normalmente, para este fin se usa un circuito temporizador automático habilitado por el flanco ascendente de la señal de trasmisión, aunque en este caso se usará una salida del microcontrolador. El temporizador habilita el circuito trasmisor durante el tiempo que dura el mensaje y lo deshabilita al terminar éste.

- **Ethernet.** Utilizaremos este tipo de comunicación en esta investigación. Allí su importancia.

  Las primeras versiones de Ethernet utilizaban cable coaxial para conectar computadoras en una topología de bus. Cada computadora se conectaba directamente al backbone. Estas

primeras versiones de Ethernet se conocían como Thicknet (10BASE5) y Thinnet (10BASE2) La 10BASE5, o Thicknet, utilizaba un cable coaxial grueso que permitía lograr distancias de cableado de hasta 500 metros antes de que la señal requiriera un repetidor. La 10BASE2, o Thinnet, utilizaba un cable coaxial fino que tenía un diámetro menor y era más flexible que la Thicknet y permitía alcanzar distancias de cableado de 185 metros. Los medios físicos originales de cable coaxial grueso y fino se reemplazaron por categorías iniciales de cables UTP. En comparación con los cables coaxiales, los cables UTP eran más fáciles de utilizar, más livianos y menos costosos. La topología física también se cambió por una topología en estrella utilizando hubs. Los hubs concentran las conexiones. En otras palabras, toman un grupo de nodos y permiten que la red los trate como una sola unidad. Cuando una trama llega a un puerto, se copia a los demás puertos para que todos los segmentos de la LAN reciban la trama. La utilización del hub en esta topología de bus aumentó la confiabilidad de la red, ya que permite que cualquier cable falle sin provocar una interrupción en toda la red. Sin embargo, la repetición de la trama a los demás puertos no solucionó el problema de las colisiones.

Ethernet se tomó como base para la redacción del estándar internacional **IEEE 802.3**, siendo usualmente tomados como

sinónimos. Se diferencian en uno de los campos de la trama de datos. Sin embargo, las tramas Ethernet e IEEE 802.3 pueden coexistir en la misma red.

Figura N° 2. 19 Puerto Ethernet

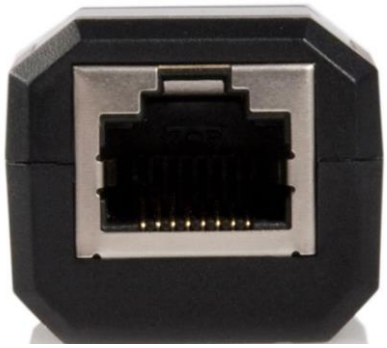
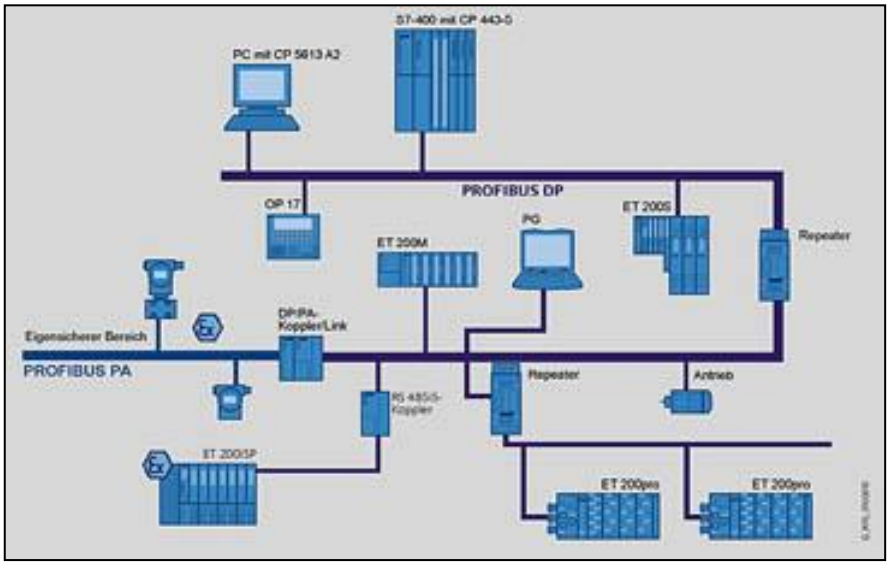


Figura N° 2. 20 Esquema de red Ethernet industrial



- **Versiones de 802.3.** Ethernet se tomó como base para la redacción del estándar internacional **IEEE 802.3**, siendo usualmente tomados como sinónimos. Se diferencian en uno de los campos de la trama de datos. Sin embargo, las tramas

originales Ethernet e IEEE 802.3 pueden coexistir en la misma red.

Tabla N° 2. 5 Versiones de 802.3

Estándar Ethernet	**Fecha**	**Descripción**
Ethernet experimental	1972 (patentado en 1978)	2,85 Mbit/s sobre cable coaxial en topología de bus.
Ethernet II (DIX v2.0)	1982	10 Mbit/s sobre coaxial fino (thinnet) - La trama tiene un campo de tipo de paquete. El protocolo IP usa este formato de trama sobre cualquier medio.
IEEE 802.3	1983	10BASE5 10 Mbit/s sobre coaxial grueso (thicknet). Longitud máxima del segmento 500 metros - Igual que DIX salvo que el campo de Tipo se substituye por la longitud.
802.3a	1985	10BASE2 10 Mbit/s sobre coaxial fino (thinnet o cheapernet). Longitud máxima del segmento 185 metros
802.3b	1985	10BROAD36
802.3c	1985	Especificación de repetidores de 10 Mbit/s
802.3d	1987	FOIRL (Fiber-Optic Inter-Repeater Link) enlace de fibra óptica entre repetidores.
802.3e	1987	1BASE5 o StarLAN
802.3i	1990	10BASE-T 10 Mbit/s sobre par trenzado no blindado (UTP). Longitud máxima del segmento 150 metros.
802.3j	1993	10BASE-F 10 Mbit/s sobre fibra óptica. Longitud máxima del segmento 1000 metros.

802.3u	1995	100BASE-TX, 100BASE-T4, 100BASE-FX Fast Ethernet a 100 Mbit/s con auto-negociación de velocidad.
802.3x	1997	Full Duplex (Transmisión y recepción simultáneos) y control de flujo.
802.3y	1998	100BASE-T2 100 Mbit/s sobre par trenzado no blindado (UTP). Longitud máxima del segmento 100 metros
802.3z	1998	1000BASE-X Ethernet de 1 Gbit/s sobre fibra óptica.
802.3ab	1999	1000BASE-T Ethernet de 1 Gbit/s sobre par trenzado no blindado
802.3ac	1999	Extensión de la trama máxima a 1522 bytes (para permitir las "Q-tag") Las Q-tag incluyen información para 802.1Q VLAN y manejan prioridades según el estándar 802.1p.
802.3ad	2000	Agregación de enlaces paralelos.
802.3ae	2003	Ethernet a 10 Gbit/s ; 10GBASE-SR, 10GBASE-LR
IEEE 802.3af	2003	Alimentación sobre Ethernet (PoE).
802.3ah	2004	Ethernet en la última milla.
802.3ak	2004	10GBASE-CX4 Ethernet a 10 Gbit/s sobre cable bi-axial.
802.3an	2006	10GBASE-T Ethernet a 10 Gbit/s sobre par trenzado no blindado (UTP)
802.3ap	en proceso (draft)	Ethernet de 1 y 10 Gbit/s sobre circuito impreso.
802.3aq	en proceso (draft)	10GBASE-LRM Ethernet a 10 Gbit/s sobre fibra óptica multimodo.

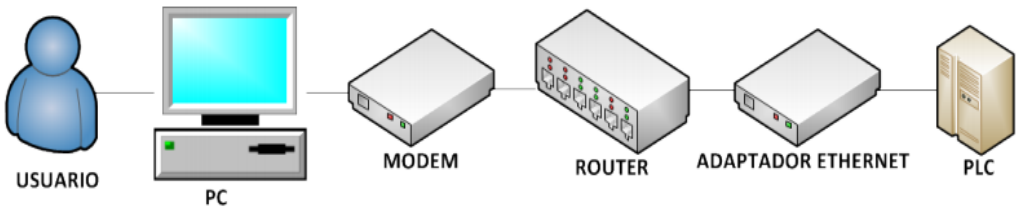
802.3ar	en proceso (draft)	Gestión de Congestión
802.3as	en proceso (draft)	Extensión de la trama

- **¿Por qué utilizar Ethernet?**

La conexión que se desea realizar entre un computador y un PLC se muestra en la figura 21. En esta figura se observa un componente humano que el que observa y controla los movimientos del PLC o simplemente los observa. Para realizar esta actividad el computador que contiene la interfaz de usuario, se conecta a través de su modem a un router compartido, al que también se conecta el adaptador Ethernet que se conecta al PLC para realizar la transmisión de señales. El adaptador Ethernet esencial y corresponde a un módulo incorporado al PLC, el cual tiene entradas y salidas, y es una de los variados tipos de conexiones con las que puede trabajar.

Mostramos una arquitectura típica entre un autómata y una PC para usuario.

Figura N° 2. 21 Conexión PC-PLC mediante Ethernet



b. Protocolos:

- **Profibus.** El protocolo de comunicación Profibus consiste en un bus de campo de alta velocidad para el control de procesos. Éste puede trabajar hasta 12 Mb/sg, garantizado según los estándares EN 50170 y EN 50254. Este tipo de comunicación trabaja con unas ordenes de tipo maestro-esclavo, donde los equipos esclavos se conectan a una línea central sobre el que actúa el maestro. Este protocolo consta de una sincronización asíncrona con una trama de 11 bits, la cual está orientada a caracteres. Además, incluye un bit de espera entre dos tramas, el cual debe de ser un valor lógico alto.

  El protocolo Profibus solo actúa sobre la capa física, de enlace y de aplicación del modelo OSI. Además, es un tipo de comunicación **maestro-esclavo**, lo que consiste en un equipo que inicia una solicitud y posteriormente espera la respuesta, siendo el maestro el encargado de iniciar siempre cada comunicación. Usualmente, el maestro suele ser un sistema **SCADA** mientras que el esclavo suele estar a un nivel inferior pudiendo ser tanto un PLC como un sensor o una unidad de control. Este protocolo también incluye paso de testigo, lo que consiste en que se va pasando un torken o marcador entre los esclavos, y solo aquel que tenga el torken puede comunicar.

Como capa física, lo más utilizado en la industria para este protocolo es un par trenzado con conexión RS485.

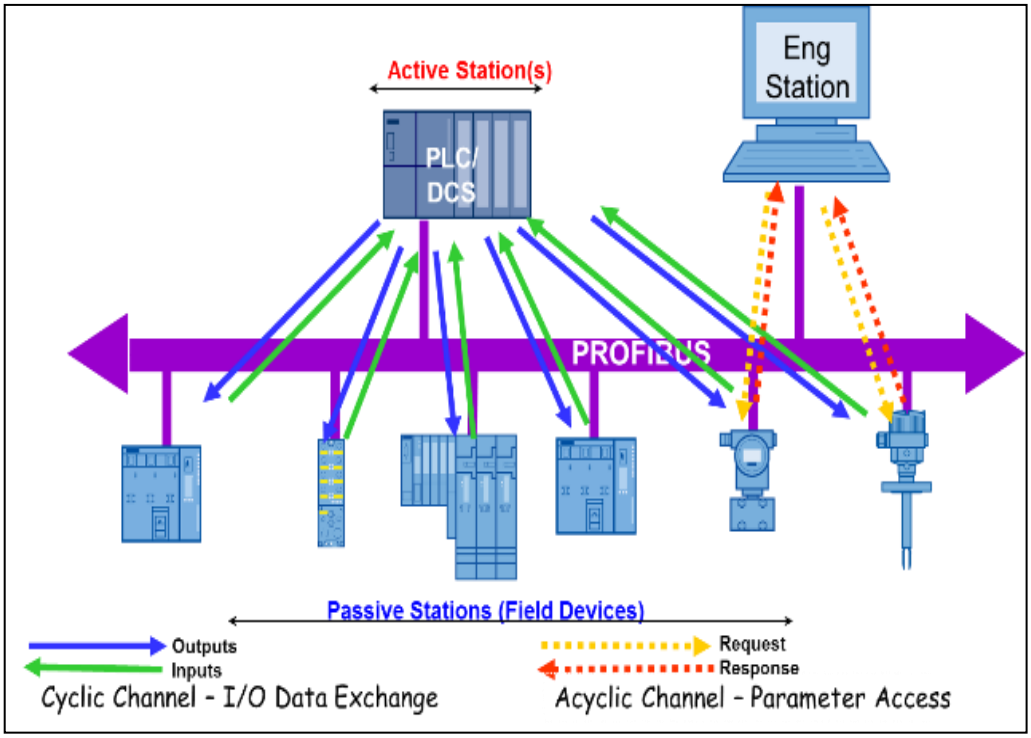
- **Profibus DP:** es un perfil orientado a sensores o actuadores enlazados a procesadores (PLCs) o terminales. Ofrece mayor velocidad y eficiencia que el resto de perfiles. Está diseñado para la comunicación entre los sistemas de automatización y los equipos periféricos, los cuales pueden alcanzar velocidades de transmisión de hasta 12 Mb/sg. Se puede utilizar para la sustitución de los sistemas de comunicación con el estándar de transmisión de 4 a 20 mA.

- **Profibus FMS.** Es un perfil utilizado para la comunicación entre células de proceso o equipos de automatización. Se trata de un perfil especializado en la transmisión de datos comunes en las comunicaciones de entorno industrial. Es utilizado para la comunicación a nivel de control y además utiliza un sistema multimaestro, el cual puede alcanzar velocidades de transmisión de hasta 1.5 Mb/sg.

- **Profibus PA.** Es un perfil orientado para el control de los procesos de automatización. Este perfil permite una transmisión síncrona a una velocidad de 31.2 Kbits/sg, y es utilizado para la tecnología de transmisión en IEC 1158-2 usada en las industrias químicas y petrolíferas. Se puede utilizar para la sustitución de

los sistemas de comunicación con el estándar de transmisión de 4 a 20 mA Permite la comunicación a través de dos hilos simples trenzados y apantallados mediante la capa física RS485.

Figura N° 2. 22 Esquema conexión mediante Profibus

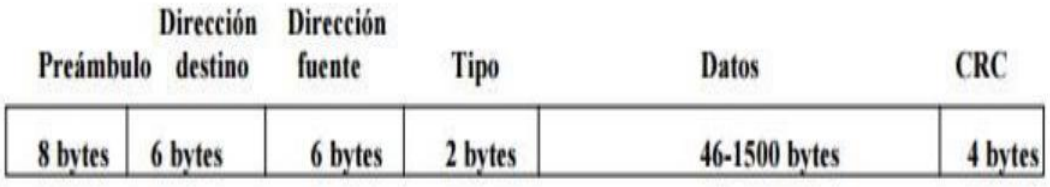


- **Protocolo Ethernet.** Ethernet o IEE 802.3 es uno de los estándares más usados en comunicaciones de Red de Área Local (LAN), donde los equipos están conectados dentro de un área geográfica a través de una red, lo que le permite tener entre 100 y 1000 usuarios y una velocidad de transmisión de entre 10 Mbps y 10 Gbps, dependiendo de la tecnología de transmisión utilizada.

Este protocolo usa un método de transmisión CSMA o acceso múltiple con detección de portadora y colisiones. Lo primero que hace es escuchar y comprobar que no esté transfiriendo otro equipo, si está libre la red, manda la trama y el resto de equipos escuchan la trama pero solo el seleccionado analiza la información. Si dos equipos intentan mandar la trama a la vez, paran de enviar y esperan un tiempo aleatorio antes de volver a intentar enviar la trama.

La trama está compuesta por la dirección del equipo receptor, la dirección del equipo transmisor y el mensaje a transmitir. El mensaje puede variar entre 64 y 1500 bytes, lo que hace que el tiempo de transmisión trabaje entre 50 y 1200 microsegundos. Además, contiene una zona de preámbulo al inicio del envío, compuesto por 64 bits que ayudan a sincronizar con el receptor deseado y una zona de redundancia cíclica o CRC, compuesto por 32 bits donde se comprueba que la trama enviada es correcta, se puede observar la trama de Ethernet en la **Figura 2.23**.

Figura N° 2. 23 . Trama en protocolo Ethernet



- **TCP/IP.** El protocolo TCP/IP se compone de una trama Ethernet auto identificable para diferenciar entre diferentes protocolos. Este conjunto de protocolos se corresponde con el modelo de comunicaciones definido por la norma ISO, que define un sistema de redes de interconexión de sistemas abiertos u OSI, que permite la comunicación entre procesos con diferentes capas, las capas dan servicio a las capas superiores y reciben servicio de capas inferiores.

- **Modbus.** Modbus consiste en un protocolo de solicitud-respuesta entre dos dispositivos. Éste funciona con una relación de maestro-esclavo, ya definido anteriormente. La capa física del protocolo Modbus puede ser a través de RS232, RS422 o RS485.

  Inicialmente, el protocolo Modbus no podía estar dividido en múltiples capas de comunicación, ya que, estaba hecho en base al protocolo serial.

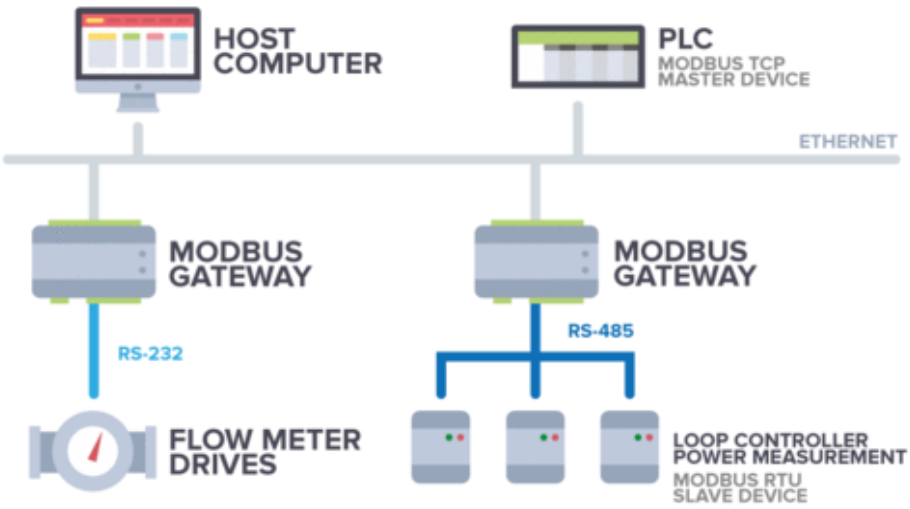
  Actualmente el protocolo Modbus ha evolucionado permitiendo el uso de redes TCP/IP, explicadas previamente en el apartado anterior, o UDP. Este protocolo está dividido en dos partes:

- **PDU.** Es la unidad de datos del protocolo. Los datos enviados a través de Modbus son almacenados en uno de los cuatro bloques de memoria. Dependiendo de la aplicación a usar, estos bancos de memoria determinan el tipo y el acceso de los datos, donde el esclavo tiene acceso a estos datos, mientras que el maestro debe pedir el acceso a estos datos a través de funciones. Cada bloque tiene un espacio máximo de 65.536 celdas, donde cada celda va desde 0 a 65.535, aunque cada elemento de datos esta numerado de 1 a n, siendo n como máximo 65.536. Para simplificar la ubicación de cada bloque de memoria se introdujo un prefijo añadido a la dirección de los datos, este prefijo está compuesto entre 4 y 6 valores (YXXX, YXXXX, YXXXXX), donde Y corresponde con el valor del bloque, mientras que XXXX corresponde con la dirección del registro variando el número de X dependiendo de las celdas ocupadas. Por ejemplo, el prefijo 3025 corresponde al bloque 3 celda 25, por otro lado, el prefijo 165536 corresponde al bloque 1 celda 65536.

- **ADU.** Es la unidad de datos de aplicación. Por otro lado, el equipo maestro debe conocer el límite de la trama por lo que además del prefijo comentado anteriormente, se añade la longitud de la trama y el orden de bytes. Por ejemplo, el prefijo 3025.2H corresponde al bloque 3 celda 25 seguido de una trama

de 2 caracteres donde el byte alto es el primer carácter, así se evita que el equipo maestro tenga que buscar la longitud de la trama. Algunos equipos esclavos pueden almacenar la información invirtiendo el orden de bytes, con lo que es trabajo del equipo maestro saber cómo esta almacenando el equipo esclavo y decodificar la información almacenada. A diferencia del modelo PDU, el modelo ADU sigue un patrón marcado, donde el esclavo primero valida el prefijo, incluyendo el rango de datos, y a continuación ejecuta la orden proveniente del equipo maestro y envía la respuesta necesaria al código.

Figura N° 2. 24 Partes de un sistema MODBUS



2.3.8. SCADA

SCADA, acrónimo de **Supervisory Control And Data Acquisition** (Supervisión, Control y Adquisición de Datos) es un concepto que se

emplea para realizar un software para ordenadores que permite controlar y supervisar procesos industriales a distancia. Facilita retroalimentación en tiempo real con los dispositivos de campo (sensores y actuadores), y controla el proceso automáticamente. Provee de toda la información que se genera en el proceso productivo (supervisión, control calidad, control de producción, almacenamiento de datos, etc.) y permite su gestión e intervención.

La realimentación, también denominada retroalimentación o feedback es, en una organización, el proceso de compartir observaciones, preocupaciones y sugerencias, con la intención de recabar información, a nivel individual o colectivo, para mejorar o modificar diversos aspectos del funcionamiento de una organización. La realimentación tiene que ser bidireccional de modo que la mejora continua sea posible, en el escalafón jerárquico, de arriba para abajo y de abajo para arriba.

En la teoría de control, la realimentación es un proceso por el que una cierta proporción de la señal de salida de un sistema se redirige de nuevo a la entrada. Esto es de uso frecuente para controlar el comportamiento dinámico del sistema. Los ejemplos de la realimentación se pueden encontrar en la mayoría de los sistemas complejos, tales como ingeniería, arquitectura, economía, sociología y biología.

Figura N° 2. 25 Sala de supervisión mediante SCADA

a. Tipos de SCADA

Existen diversos tipos de sistemas SCADA dependiendo del fabricante y sobre todo de la finalidad con que se va a hacer uso del sistema, por ello antes de decidir cuál es el más adecuado hay que tener presente si cumple o no ciertos requisitos:

- Todo sistema debe tener arquitectura abierta, es decir, debe permitir su crecimiento y expansión, así como deben poder adecuarse a las necesidades futuras del proceso y de la planta.

- La programación e instalación no debe presentar mayor dificultad, debe contar con interfaces gráficas que muestren un esquema básico y real del proceso.

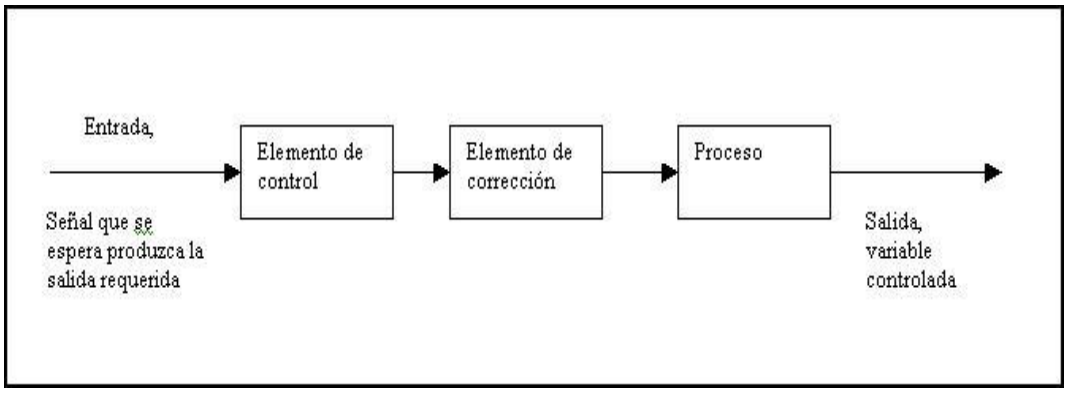
- Deben permitir la adquisición de datos de todo equipo, así como la comunicación a nivel interno y externo (redes locales y de gestión).

- Deben ser programas sencillos de instalar, sin excesivas exigencias de hardware, y fáciles de utilizar, con interfaces amigables para el usuario.

- **Sistemas de Lazo abierto.**

  En los sistemas no realimentados o de lazo abierto no se compara la variable controlada con una entrada de referencia. Cada ajuste de entrada determina una posición de funcionamiento fijo en los elementos de control.

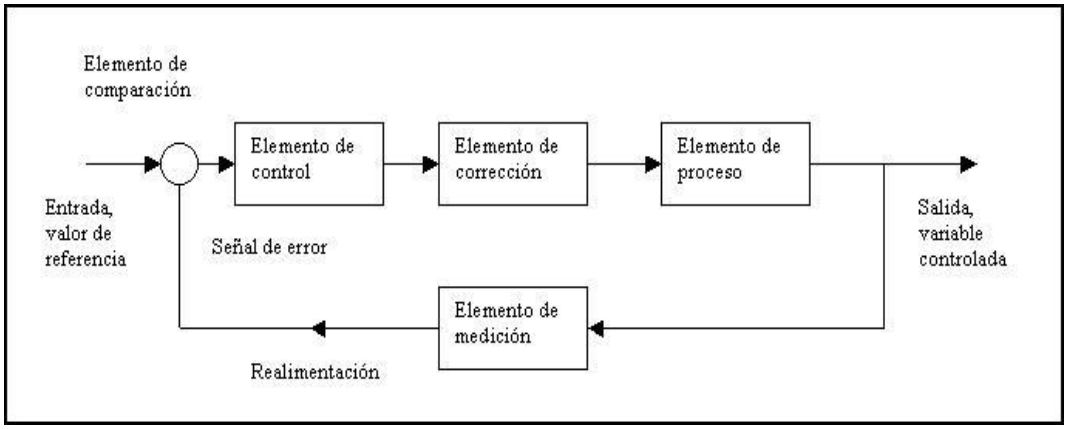
Figura N° 2. 26 Esquema general de sistema de lazo abierto



- **Sistemas de Lazo cerrado**

Los sistemas realimentados o de lazo cerrado funcionan de tal manera que hace que el sistema se realimente, la salida vuelve al principio para que analice la diferencia y en una segunda opción ajuste más, así hasta que el error es 0. Cualquier concepto básico que tenga como naturaleza una cantidad controlada como por ejemplo temperatura, velocidad, presión, caudal, fuerza, posición, y termocuplas, etc. son parámetros de control de lazo cerrado.

Figura N° 2. 27 Esquema general de sistema de lazo cerrado



b. Requisitos básicos

Un software SCADA debe ser capaz de ofrecer al sistema:

- Posibilidad de crear paneles de alarma, que exigen la presencia del operador para reconocer una parada o situación de alarma, con registro de incidencias.

- Generación de datos históricos de la señal de la planta, que pueden ser volcados para su proceso sobre una hoja de cálculo.

- Ejecución de programas, que modifican la ley de control, o incluso anular o modificar las tareas asociadas al autómata, bajo ciertas condiciones.

- Posibilidad de programación numérica, que permite realizar cálculos aritméticos de elevada resolución sobre la CPU del ordenador.

c. Esquema básico

Este esquema es un ejemplo de la aplicación del sistema SCADA en áreas industriales. Estas áreas pueden ser:

- **Monitorizar procesos químicos, físicos o de transporte:** Fundamentalmente en sistemas de suministro de agua, para controlar la generación y distribución de energía eléctrica, de gas o en oleoductos y otros procesos de distribución.

- **Gestión de la producción:** Facilita la programación de la fabricación.

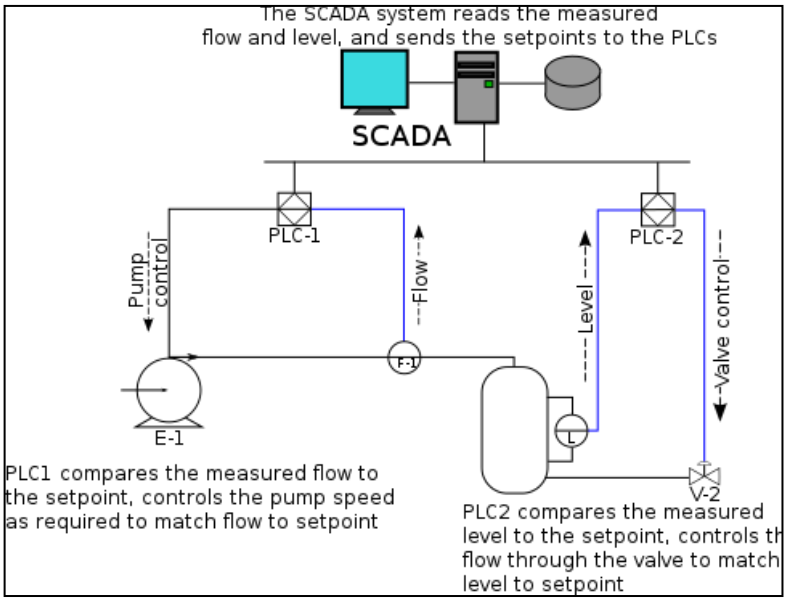
- **Mantenimiento:** Proporciona magnitudes de interés tales para evaluar y determinar modos de fallo, índices de fiabilidad, entre otros).

- **Control de Calidad:** Proporciona de manera automatizada los datos necesarios para calcular índices de estabilidad de la producción CP y CPk, tolerancias, índice de piezas NOK/OK, etc.

- **Administración:** Actualmente pueden enlazarse estos datos del SCADA con un servidor ERP (Sistema de Planificación de Recursos Empresariales), e integrarse como un módulo más.

- **Tratamiento histórico de información:** Se realiza mediante la incorporación de la misma en bases de datos.

Figura N° 2. 28 SCADA en plano P&D



d. Componentes del Sistema.

- **Unidad de Terminal Remota (RTU).** La RTU se conecta al equipo físicamente y lee los datos de estado como los estados abierto/cerrado desde una válvula o un interruptor, lee las medidas

como presión, flujo, voltaje o corriente. Por el equipo el RTU puede enviar señales que pueden controlarlo: abrirlo, cerrarlo, intercambiar la válvula o configurar la velocidad de la bomba, ponerla en marcha, pararla.

La RTU puede leer el estado de los datos digitales o medidas de datos analógicos y envía comandos digitales de salida o puntos de ajuste analógicos.

- **Estación Maestra.** El término "Estación Maestra" se refiere a los servidores y al software responsable para comunicarse con el equipo del campo (RTUs, PLCs, etc) en estos se encuentra el software HMI (Interfaz Humano – Máquina) corriendo para las estaciones de trabajo en el cuarto de control, o en cualquier otro lado. En un sistema SCADA pequeño, la estación maestra puede estar en un solo computador, A gran escala, en los sistemas SCADA la estación maestra puede incluir muchos servidores, aplicaciones de software distribuido, y sitios de recuperación de desastres.

- **Infraestructura y Métodos de Comunicación.** Los sistemas SCADA tienen tradicionalmente una combinación de radios y señales directas seriales o conexiones de módem para conocer los requerimientos de comunicaciones, incluso Ethernet e IP sobre SONET (fibra óptica) es también frecuentemente usada en sitios muy grandes como ferrocarriles y estaciones de energía eléctrica. Es

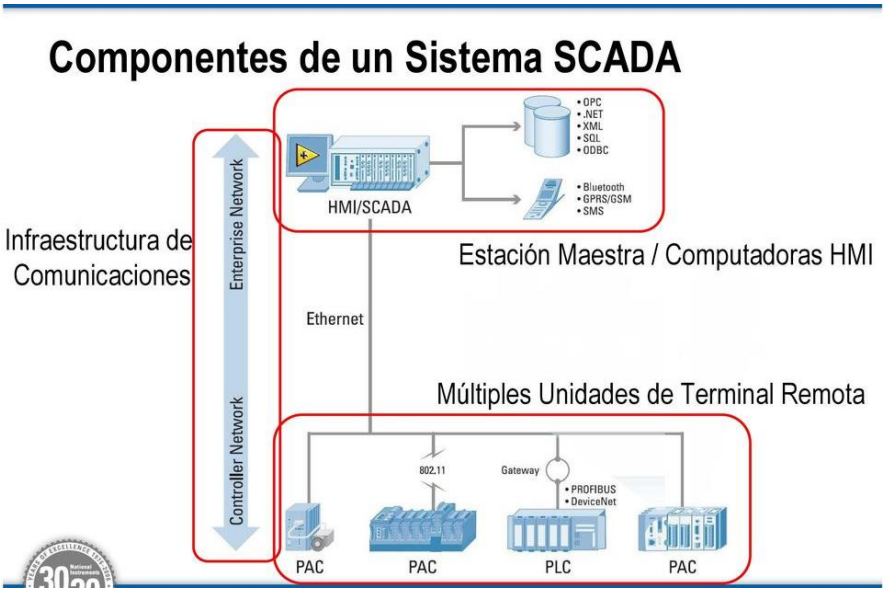
más, los métodos de conexión entre sistemas puede incluso que sea a través de comunicación gíreles, es decir, si se quiere enviar la señal a un teléfono móvil sin necesidad de emplear cables.

e. Requisitos básicos.

Para desarrollar un sistema SCADA es necesario un Entorno de Desarrollo en el cual diseñar, entre otras cosas:

- El aspecto que va a tener el SCADA
- Las funciones y eventos que debe ejecutar cuando se interactúa con su interfaz HMI.
- Las operaciones y cálculos que debe realizar con los datos adquiridos.

Figura N° 2. 29 Componentes de un sistema SCADA



f. Características:

- Tipo de Arquitectura: Centralizada.
- Tipo de control predominante: Supervisor - Lazos de control cerrados por el operador. Adicionalmente: control secuencial y regulatorio.
- Tipos de Variables: Desacopladas.
- Área de acción: Áreas geográficamente distribuidas.
- Unidades de adquisición de datos y control: Remotas, PLCs.
- Medios de comunicación: Radio, satélite, líneas telefónicas, conexión directa, LAN, WAN.
- Base de Datos: Centralizada.

g. Funciones de un Sistema SCADA.

Las funciones básicas de un sistema SCADA son las que se describen a continuación:

- Supervisión Remota de Instalaciones
- Control Remoto de Instalaciones
- Procesamiento de Información

- Presentación de Gráficos Dinámicos
- Generación de Reportes
- Presentación de Alarmas
- Almacenamiento de Información Histórica
- Presentación de Gráficos de Tendencias
- Programación de Eventos

h. Módulos de un Sistema SCADA.

Los módulos o bloques software que permiten las actividades de adquisición, supervisión y control son los siguientes:

- **Configuración:** permite al usuario definir el entorno de trabajo de su SCADA, adaptándolo a la aplicación particular que se desea desarrollar.
- **Interfaz gráfico del operador:** proporciona al operador las funciones de control y supervisión de la planta. El proceso se representa mediante sinópticos gráficos almacenados en el ordenador de proceso y generados desde el editor incorporado en el SCADA o importados desde otra aplicación durante la configuración del paquete.

- **Módulo de proceso:** ejecuta las acciones de mando pre-programadas a partir de los valores actuales de variables leídas.

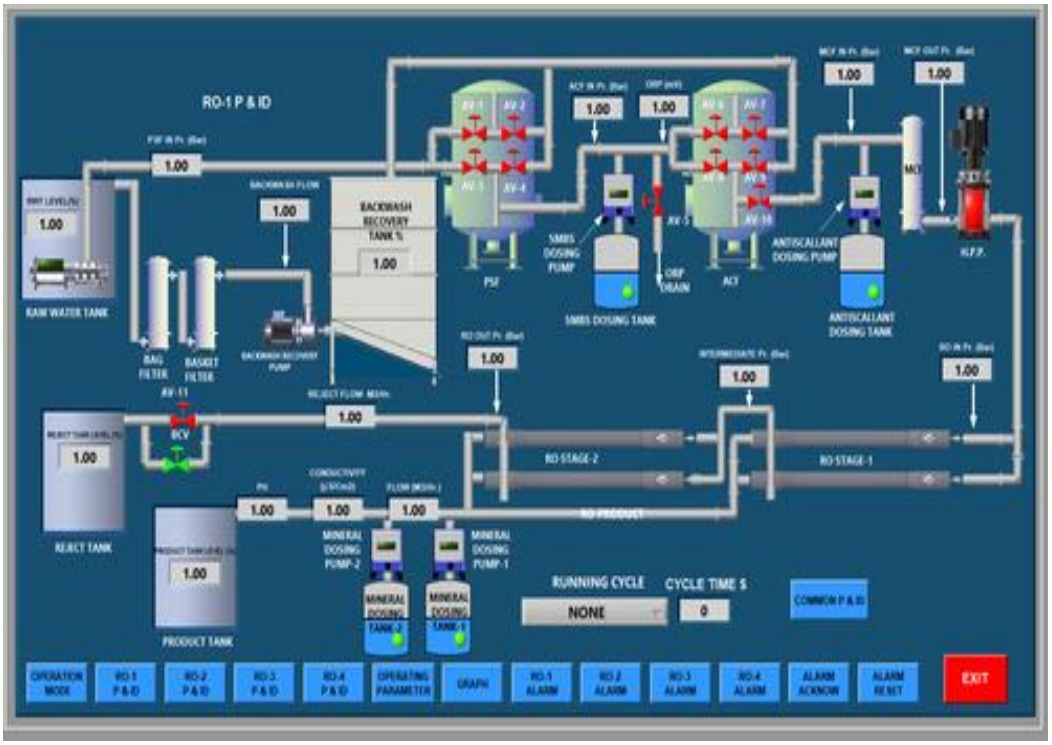
- **Gestión y archivo de datos:** se encarga del almacenamiento y procesado ordenado de los datos, de forma que otra aplicación o dispositivo pueda tener acceso a ellos.

- **Comunicaciones:** se encarga de la transferencia de información entre la planta y la arquitectura hardware que soporta el SCADA, y entre ésta y el resto de elementos informáticos de gestión.

2.3.9. SCADA para Arduino – Acimut Monitoriza

Esta versión de "Acimut Monitoriza for Arduino" es totalmente funcional y libre de todo tipo de restricciones de uso, tanto en cuanto número de variables a controlar como de clientes-puestos de monitorización que la versión comercial sí que tiene. La única "limitación" que tiene es que solo se puede conectar a dispositivos Arduino, lo cual puedo ser visto como una ventaja ya que al ser Arduino tecnología OPEN SPURCE el Scada no tiene costo.

Figura N° 2. 30 Ejemplo de Sistema SCADA



- **Características.**

  - Instalación sencilla e inmediata del producto.

  - Fácil configuración, incluso cuando se trata de una instalación con puestos remotos (WAN) ya que las comunicaciones entre los equipos cliente y el servidor se basan en los estándares de internet (protocolo HTTP).

  - No precisa programación para la creación de proyectos completamente funcionales, basta “pinchar y arrastrar” los objetos SCADA sobre la superficie de los formularios y establecer las propiedades correspondientes para obtener una solución operativa.

- Si se requiere una funcionalidad avanzada que no esté contemplada en los objetos SCADA definidos en Monitoriza no hay problema ya que Monitoriza es extensible mediante programación en C# o VB.Net. También es posible la utilización de librerías de terceros desarrolladas para el .NET Framework de Windows.

- La creación de la interfaz gráfica de usuarios está basada en la tecnología de Windows Forms Designer de Microsoft© lo que facilita de forma notable el diseño.

- Fácil creación de gráficas para representar la tendencia de las variables

- A nivel de proyecto podemos definir usuarios y los permisos asignados a cada uno ellos. Por ejemplo, si solo se tiene permiso de lectura en un determinado formulario o si se tiene acceso total a este.

- Definición inmediata de alarmas.

- Fácil seguimiento de variables.

- Datos en formatos accesibles. Monitoriza permite almacenar las variables que se monitorizan en bases de datos estándar del mercado (Microsoft© SQL Server™, Microsoft© Access™, Oracle®, etc.).

Figura N° 2. 31 Logo de Acimut Monitoriza

- **Partes:**

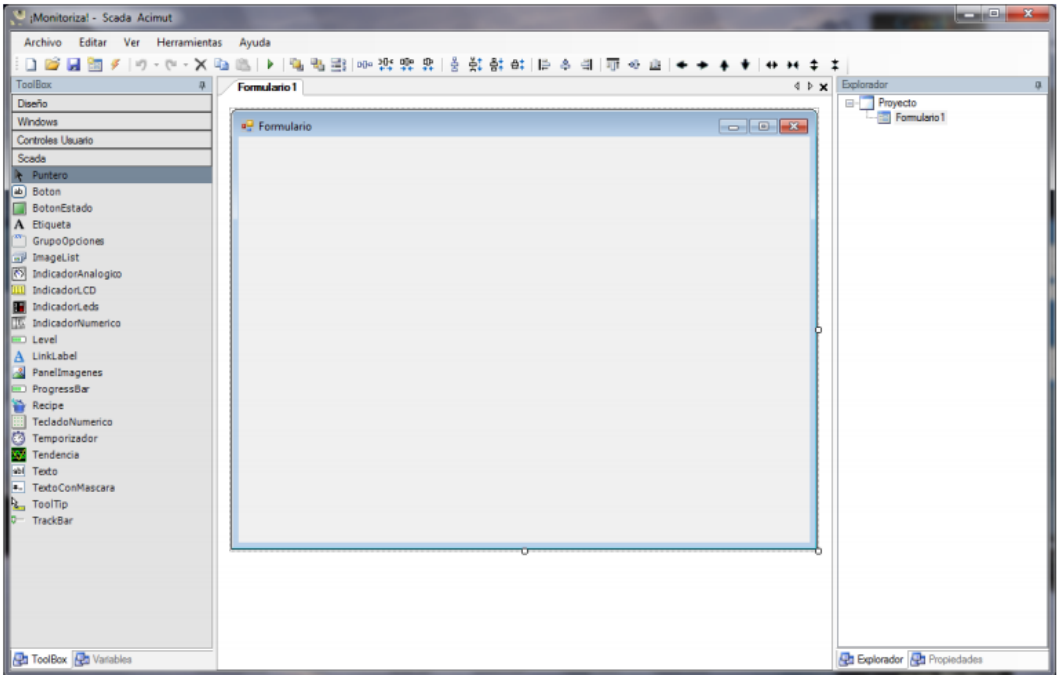
  ACIMUT Monitoriza para Arduino se divide en 3 partes, según su manual, descritas a continuación:

- **Editor.**

  El Editor de Acimut Monitoriza es uno de los tres componentes principales del sistema, con el vamos a crear, diseñar y modificar nuestros proyectos de Scada que luego se ejecutaran a través del Servidor de Comunicaciones y del Cliente Scada. Al crear o modificar un proyecto Scada mediante el Editor podremos definir variables y alarmas, crear formularios para mostrar de forma gráfica los valores de las variables, guardar en base de datos los valores de las variables, mantener un histórico de alarmas, mostrar gráficas de los valores de variables almacenados, escribir variables sobre un

autómata (u otros dispositivos) y gestionar los usuarios que podrán acceder a los recursos del proyecto. La interfaz de usuario del Editor es la que puede verse en la siguiente imagen:

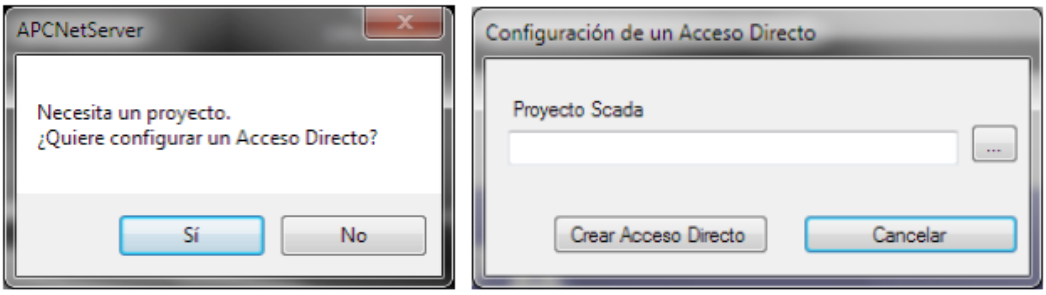
Figura N° 2. 32 Interfaz de Editor de Acimut



- **Servidor**

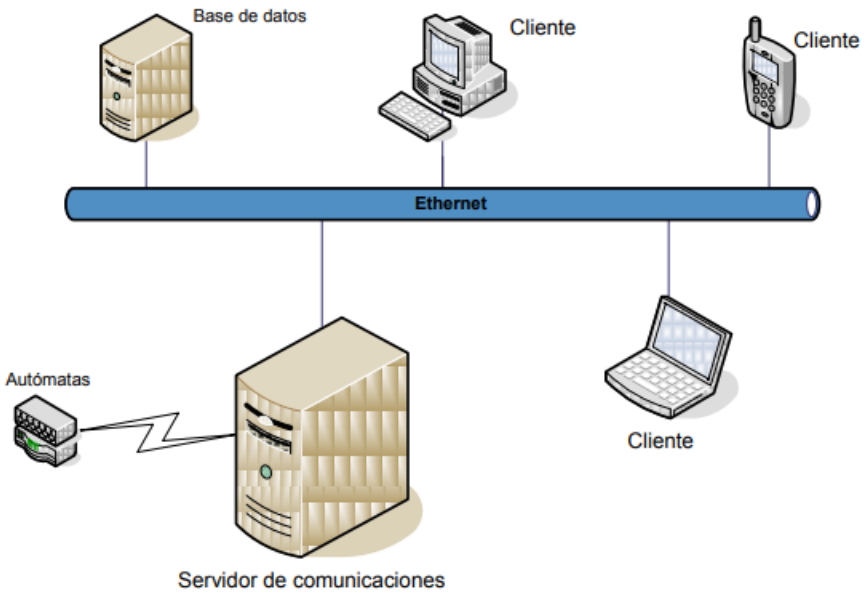
El servidor de Acimut Monitoriza es el encargado de establecer y coordinar las comunicaciones tanto con los dispositivos y autómatas como con las bases de datos. El servidor monitoriza los cambios en las variables definidas en tiempo real e informa a los clientes, o sea, a los puestos de control, para que estos presenten la información a los usuarios del sistema. También se encarga de evaluar los valores de las variables para comprobar si se cumple alguna de las condiciones establecidas para disparar una alarma.

Figura N° 2. 33 Configuración de acceso directo



Cuando iniciamos el servidor lo primero que nos pregunta es si queremos configurar un acceso directo tal como se muestra en la **Figura 2.33**

Figura N° 2. 34 Esquema de comunicaciones en servidor ACIMUT



Esto es así, ya que el servidor necesita saber cuál es el proyecto que queremos ejecutar.

Le tendremos que especificar un proyecto .Scada creado con el Editor de Monitoriza y pulsando el botón Crear Acceso Directo nos creará en el escritorio un enlace al servidor que por ejemplo quedará como:

"C:\Archivos de programa\Acimut\APCNetServer.exe" C:\Documents and Settings\Usuario\Mis documentos\ServidorModBUSTCP.scada

Para finalizar el servidor tenemos que pulsar, con el botón derecho del ratón, sobre el icono del servidor que se encuentra en la barra de notificaciones del escritorio (marcado en rojo en la siguiente figura).

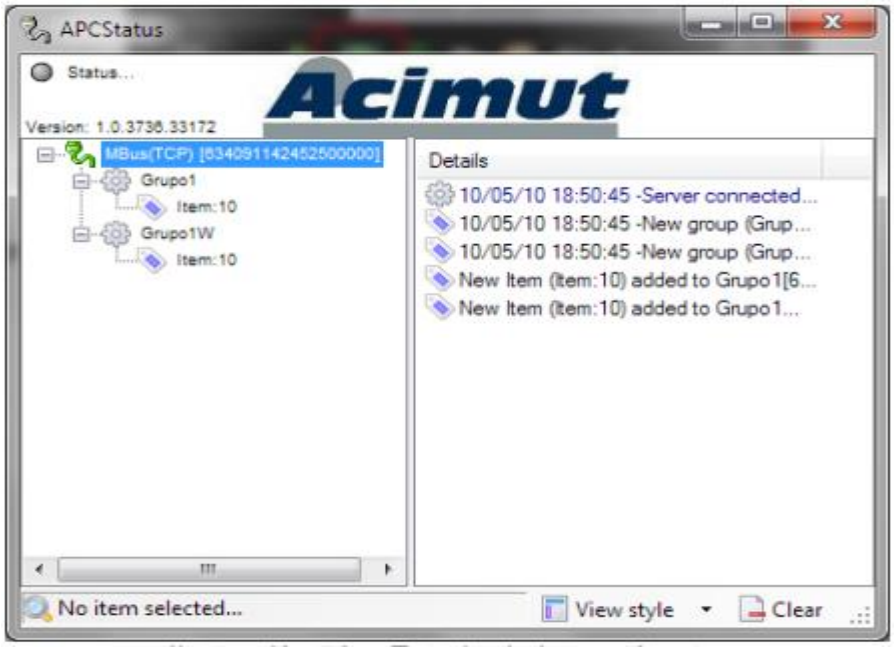
Figura N° 2. 35 Servidor de comunicaciones ACIMUT

Y elegir la opción Salir. Mediante el icono marcado en rojo en la siguiente figura:

Figura N° 2. 36 Icono Control de Autómatas ACIMUT

Podemos ver el estado de comunicación del Servidor con los autómatas para poder comprobar si se está accediendo correctamente a cada uno de los dispositivos definidos.

Figura N° 2. 37 Estados de los Autómatas ACIMUT



- **Cliente.**

  El cliente de Monitoriza debe hacer referencia a un servidor para mostrar la ejecución de un proyecto. El caso más sencillo, es en el que el servidor se encuentra en el mismo ordenador, no precisa de parámetros, pues por defecto se busca un servidor en la máquina local.

  Sin embargo, mostramos aquí los parámetros soportados por el cliente de Monitoriza.

  NombreServidor Puerto Red

  NombreServidor es el nombre o dirección IP del ordenador que ejecuta el servidor.

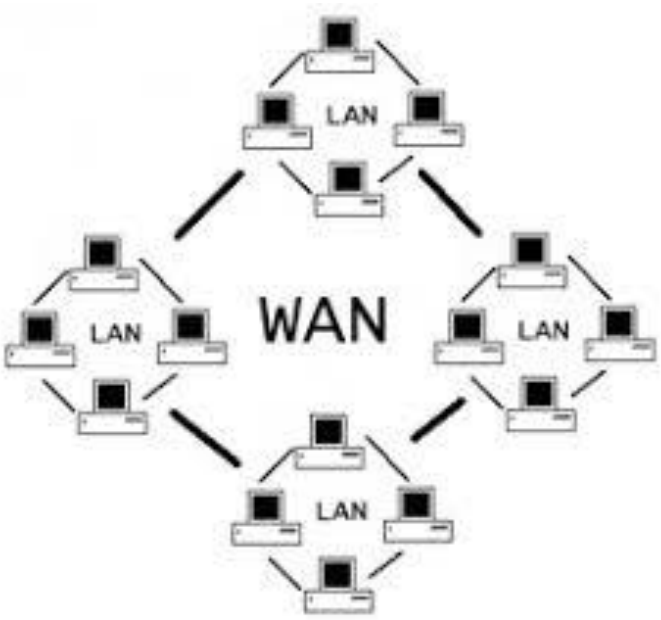
Puerto es el puerto TCP que ejecuta el servidor de Monitoriza, 8800 es el valor por defecto.

Red puede tener dos valores:

WAN indica que se utilizarán comunicaciones a través de Internet (por ejemplo), se utilizará un protocolo HTTP.

LAN indica que se utilizarán comunicaciones en una LAN, se utilizará un protocolo TCP.

Figura N° 2. 38 Relación entre red WAN y LAN



CAPITULO III

III. VARIABLES E HIPÓTESIS:

Definiremos la operacionabilidad de las variables intervinientes en el presente problema objeto de investigación, posibilitando la explicación, demostración y probación de la hipótesis formulada, para ello, se han identificado una variable independiente y una variable dependiente, teniendo la siguiente operacionabilidad:

3.1. Operacionabilidad de las variables:

3.1.1. Variable independiente:

- Aplicación de un controlador lógico programable, de salida tipo relé, basado en Arduino, es una variable independiente "**X**"

3.1.2. Variable dependiente:

- Transmisión de datos entre etapas de procesos industriales, es una variable dependiente "**Y**"

3.2. Hipótesis

3.2.1. Hipótesis principal

Será posible implementar un controlador lógico programable, cuyo núcleo sea un Arduino, de bajo costo, a fin de expandir la aplicación de la automatización y para transmisión de datos entre etapas de procesos industriales mediante el uso de componentes electrónicos.

3.2.2. Hipótesis específicas:

- Será posible adaptar las entradas y salidas del Arduino para funcionar a nivel de tensión industrial, mediante el uso de componentes electrónicos como optoacopladores, relés electromagnéticos, transistores, entre otros.

- Será posible enlazar dos de estos autómatas, sin costo alguno, mediante protocolos de comunicación para la transmisión de datos.

- Será posible desarrollar un sistema de supervisión para este sistema mediante protocolos de redes industriales y presentará un bajo costo frente a los sistemas existentes actualmente en el mercado.

CAPITULO IV

IV. METODOLOGÍA:

4.1. Tipo de investigación:

Temporal subtipo investigación sincrónica

Este trabajo de investigación es de tipo temporal sincrónica puesto que el estudio fue realizado en un pequeño periodo de tiempo de Agosto – diciembre 2016.

Espacial

Este trabajo de investigación es de tipo espacial puesto que hemos tomado como referencia un proceso industrial localizado en Lima.

4.2. Diseño De La Investigación

El inicio de la investigación se dividirá en dos partes, la primera el HARDAWARE y la segunda el SOFTWARE.

4.2.1. Hardware del sistema.

Este proyecto se desarrolló mediante los siguientes pasos y procedimientos:

- Determinación de las especificaciones técnicas de cada componente a utilizar, a fin de saber si es adecuado al objetivo a realizar.

- En base a los objetivos planteados, se procede a investigar sobre la gama de componentes electrónicos que se adecuen a lo deseado y nos permitan un diseño óptimo.

- Desarrollo del diseño preliminar del circuito electrónico mediante el que se adaptará las entradas y salidas.

- Una vez escogidos los componentes electrónicos, se procede a diseñar su distribución en un circuito electrónico de tal manera que se verifique si los componentes escogidos anteriormente se adecuan a lo deseado.

- Simulación del diseño elaborado, con el fin de evaluar la eficacia del diseño elegido y analizar las posibles mejoras.

- Se procederá a simular el circuito diseñado mediante el software de simulación "PROTEUS" y su herramienta "ISIS", a fin de verificar que el diseño cumpla con los parámetros eléctricos deseados.

- Pruebas del diseño elegido en una placa de pruebas en apoyo con módulos comerciales, a fin de analizar los errores y generar correcciones.

- Una vez se realice una simulación exitosa, se implementa el diseño en una placa de pruebas (PROTOBOARD) a fin de

analizar posibles errores de diseño respecto a distribución y cargas estáticas en un entorno real.

- Diseño de la interfaz electrónica de manera para que pueda ser implementada en una tarjeta impresa (PCB).

- Posteriormente, se diseñará la distribución del circuito con la finalidad de ser implementado en una placa PCB, se optará por el diseño de menor dimensión.

- Creación de la interfaz y verificación del funcionamiento.

- Se implementará la placa diseñada anteriormente en una placa PCB, se utilizarán componentes SMD (superficiales), a fin de conseguir el objetivo de ocupar una menor dimensión y conseguir un acabado más profesional.

a. Determinación de Especificaciones técnicas de los componentes a utilizar:

- **Relé Electromagnético:**

  En este proyecto, usaré un conjunto de relés cuya bobina se energiza mediante un nivel de tensión de 5VDC, dado que es el nivel de tensión al que trabaja el Arduino. Cada uno posee un interruptor de dos posiciones, además que por las características eléctricas que posee nos permitirá manipular diversas cargas sin ningún problema.

Figura N° 4. 1 Vista Frontal de un relé electromagnético de 5VDC

Tabla N° 4. 1 Especificaciones técnicas de un relé electromagnético de 5VDC

Tensión de operación	5VDC
Tensión máxima de conmutación AC	250 VAC
Tensión máxima de conmutación DC	30VDC
Intensidad de corriente máxima de conmutación	10A

Podemos observar su hoja técnica en el **Anexo 7.**

- **Optoacoplador:**

  Un optoacoplador, también llamado optoaislador o aislador acoplado ópticamente, es un dispositivo de emisión y recepción que funciona como un interruptor activado mediante la luz emitida por un diodo LED que satura un componente optoelectrónico, normalmente en forma de fototransistor o fototriac. De este modo se combinan en un solo dispositivo

semiconductor, un fotoemisor y un fotorreceptor cuya conexión entre ambos es óptica. Estos elementos se encuentran dentro de un encapsulado que por lo general es del tipo DIP. Se suelen utilizar para aislar eléctricamente a dispositivos muy sensibles.

En este proyecto se usará el optoacoplador 4N25, el cual tiene por componente optoelectrónico a un fototransistor.

Figura N° 4. 2 Vista Superior de un optoacoplador 4N25

Tabla N° 4. 2 Especificaciones Técnicas del optoacoplador 4N25

Corriente máxima del LED	60mA
Voltaje inverso máximo del LED	6V
Voltaje Colector – Emisor máximo	30V
Disipación máxima de potencia en el LED	100mW
Caída de tensión típica en el LED	1.3V

Podemos observar su hoja técnica en el Anexo 8.

- **Regulador de Tensión:**

  Un regulador de tensión o regulador de voltaje es un dispositivo electrónico diseñado para mantener un nivel de tensión constante.

  Los reguladores electrónicos de tensión se encuentran en dispositivos como las fuentes de alimentación de los computadores, donde estabilizan las tensiones de corriente continua usadas por el procesador. En los alternadores de los automóviles y en las plantas generadoras, los reguladores de tensión controlan la salida de la planta.

  La familia LMXXX de regulares de tensión se caracteriza por no desperdiciar energía con una excesiva disipación de calor, además se escoge LM2596 ya que lo encontramos en módulos fabricados comercialmente para Arduino.

Figura N° 4. 3 Vista superior del regulador de Tensión LM2596

Tabla N° 4. 3 Especificaciones Técnicas de LM2596

Voltaje de Entrada	3V – 40V
Voltaje de Salida	1.3V – 35V
Intensidad de Corriente máxima en la salida	3A
Temperatura de trabajo	−65°C - 150°C

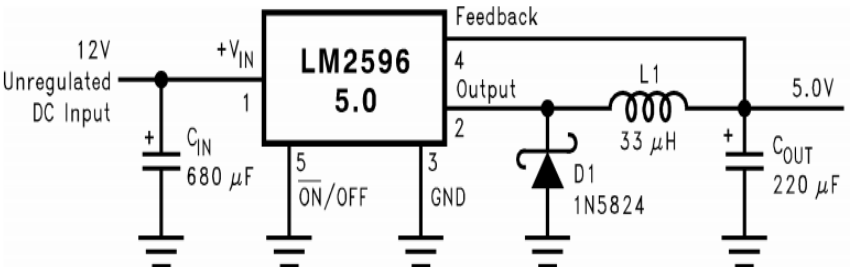
Podemos observar su hoja técnica en el **Anexo 9.**

b. Diseño preliminar del circuito electrónico para cada entrada y salida:

Entradas:

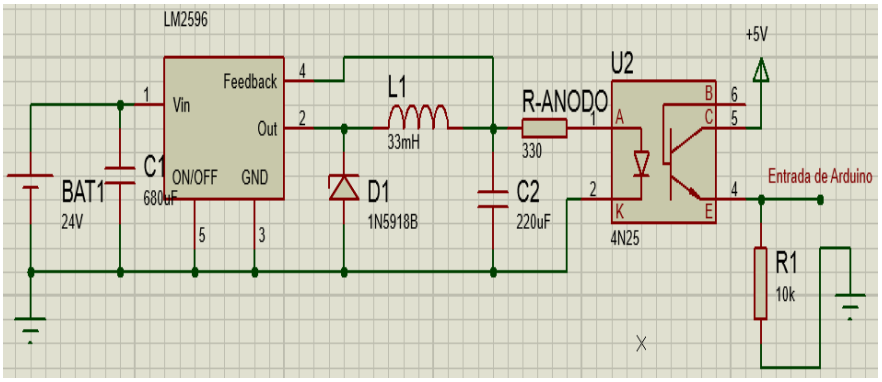
En este proyecto usaremos el regulador de tensión LM2596, implementado en un circuito de tal manera que la salida tenga un nivel de tensión de 5VDC, que es el nivel de tensión al cual trabaja el Arduino, dicho circuito es proporcionado por Texas Instruments, fabricante de este regulador.

Figura N° 4. 4 Circuito Electrónico basado en el LM2596 que proporciona salida de 5VDC



Adicionalmente, a la salida del regulador de tensión se conectará un optoacoplador, con el fin de proteger al Arduino.

Figura N° 4. 5 Simulación de la interfaz electrónica para cada entrada del Arduino



El resistor del Emisor se determinó de tal manera que por el LED del optoacoplador circule una intensidad de corriente de 10mA para que la tensión del LED sea 1,3V. Obtenemos el valor de R-ÁNODO mediante una ecuación de malla.

$$R_{anodo} = \frac{V_{out} - V_{LED}}{I} \quad (1)$$

$$R_{anodo} = \frac{5V - 1{,}3V}{0{,}01A} \quad (2)$$

$$R_{anodo} = 370\Omega \quad (3)$$

El valor comercial más cercano al obtenido es:

$$R_{anodo} = 330\Omega \quad (4)$$

Entonces, al cambiar el valor de la resistencia, la intensidad de corriente que circula por el LED tendrá otro valor.

$$I_{LED} = \frac{V_{out} - V_{LED}}{I_{LED}} \quad (5)$$

$$I_{LED} = \frac{5V - 1{,}3V}{330\Omega} \quad (6)$$

$$I_{LED} = 11.21mA \quad (7)$$

Analizamos el consumo de potencia del resistor con la nueva corriente en (7)

$$P_{R-anodo} = (I_{LED}^2) * R_{anodo} \quad (8)$$

$$P_{R-anodo} = (11{,}21 * 10^{-3})^2 * 330\Omega \quad (9)$$

$$P_{R-anodo} = 41{,}47mW \quad (10)$$

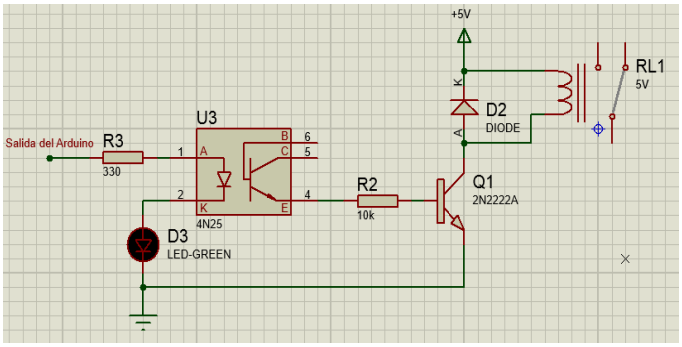
El consumo de potencia del resistor es menor a 0,25 W, por lo cual no hay preocupación alguna en lo que respecta a la potencia.

Salidas:

En las salidas se usarán relés con contactos de dos posiciones, la bobina estará en paralelo a un diodo, a fin de proteger el transistor encargado de la activación del relé, la base del transistor se conectará

a la salida de un optoacoplador mediante una resistencia, el LED del optoacoplador se polariza mediante las salidas del Arduino, dado que este emite señales de 5VDC suficientes para polarizar el LED. Se coloca el optoacoplador a fin de proteger al Arduino. Adicionalmente, se coloca un LED que indique el cambio de estado del relé.

Figura N° 4. 6 Simulación de la interfaz electrónica para cada salida del Arduino



c. Simulación del diseño:

Las simulaciones fueron desarrolladas con la herramienta ISIS de Proteus, de LabCenter, en su versión Proteus 8, dichas simulaciones pudieron corroborar que el diseño elegido es adecuado.

Las imágenes de las simulaciones realizadas se muestran en la figura 9 y la figura 10. Solo fue necesario simular una parte del circuito, dado que en conjunto la placa impresa se basa en la réplica de estos circuitos el número de veces necesarias, según las entradas y salidas que dispongamos en nuestra placa.

d. Pruebas del diseño:

Se realizaron las pruebas mediante el uso de módulos comerciales, relés electromagnéticos de 5VDC en las salidas y un módulo LM2596 en la entrada, se utilizó como alimentación una fuente de 24 VDC de 500mA para energizar al circuito, dado que este nivel de tensión es comúnmente usado a nivel industrial.

Se implementó una aplicación de prueba, en este caso el uso de un circuito de control de una faja transportadora, se incluyó un sensor de presencia y un sensor ultrasonido en las entradas, y un motor monofásico de 220VAC y un pistón de 24 VDC en las salidas. Obteniendo resultados positivos se procede a diseñar la placa PCB, la cual al ser creada será utilizada en una aplicación real.

Figura N° 4. 7 Circuito de prueba del diseño preliminar

e. Diseño preliminar del PCB:

Se muestra el diseño preliminar de la placa impresa destinada a ser usada en las entradas del Arduino, con el fin de adaptarlas para que funcionen con tensiones de 24VDC.

En el diseño se incluyen borneras, además de “espadines”, estos últimos con el fin de que el Arduino pueda ser insertado en dichos pines.

El diseño fue desarrollado en CadSoft EAGLE PCB Design Software, en este diseño se incluyó puentes de diodo, dado que estos estaban pensados anteriormente, mediante pruebas se descartó su funcionalidad, estos serán reemplazados por el regulador de tensión LM2596 el cual ya ha sido probado, obteniendo resultados exitosos.

Reemplazando los puentes de diodo por reguladores de tensión LM2596, además de incluir el diseño de la placa en las salidas, el diseño final de la tarjeta usando un Arduino UNO como núcleo, para la elaboración final se incluyeron librerías que permiten el uso de Arduino, en sus diferentes versiones, para el diseño de la tarjeta.

f. Creación de la interfaz y verificación del funcionamiento.

Una vez concluido el diseño de la placa PCB procedemos a implementarla a manera de pruebas.

Podemos observar el diseño de la placa a utilizar, desarrollado en EAGLE, en el **Anexo 10.**

4.2.2. Software del sistema.

La implementación del software del sistema seguirá la siguiente secuencia:

- Instalación de Arduino
- Desarrollo de proyecto con módulos Ethernet:
- Instalación de las librerías necesarias.
- Diseño de esquema de conexiones de comunicaciones.
- Desarrollo del código de programación.
- Prueba de conexión entre dos Arduino.
- Explicación de posibles extensiones de esta parte del proyecto.
- Desarrollo de proyecto con Scada Acimut Monitoriza:
- Instalación de Acimut Monitoriza.
- Creación de pantallas para SCADA.
- Creación de código Arduino para SCADA.
- Ejecución del programa.

a. **Instalación de Arduino:**

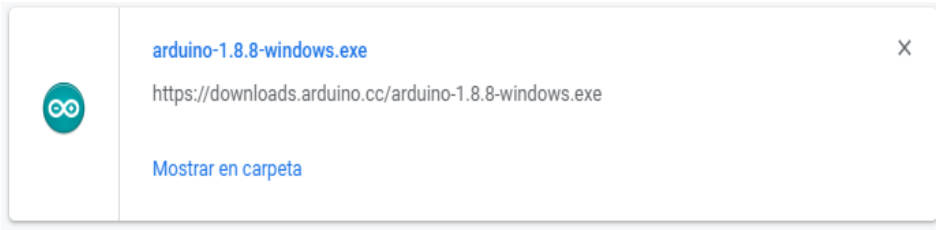
Ingresamos a la página oficial de Arduino , la cual tiene la siguiente dirección web: https://www.arduino.cc/, en la sección de descargas (download) podemos obtener la versión más reciente del IDE de Arduino acorde a las características de la PC en la que se instalara.

Figura N° 4. 8 Ventana de descarga de Arduino

En la actualidad, diciembre del 2018, la versión más reciente es la de Arduino 1.8.8, véase Figura 46.

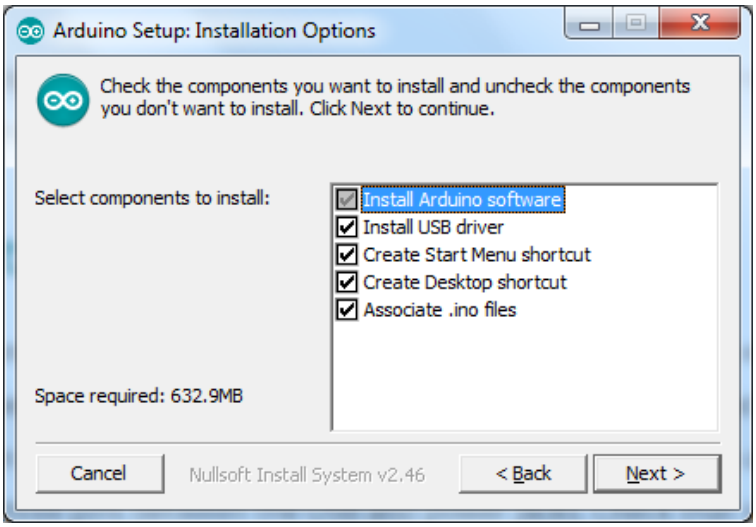
Luego de descargar el instalador adecuado, procedemos a ejecutarlo:

Figura N° 4. 9 Instalador de Arduino



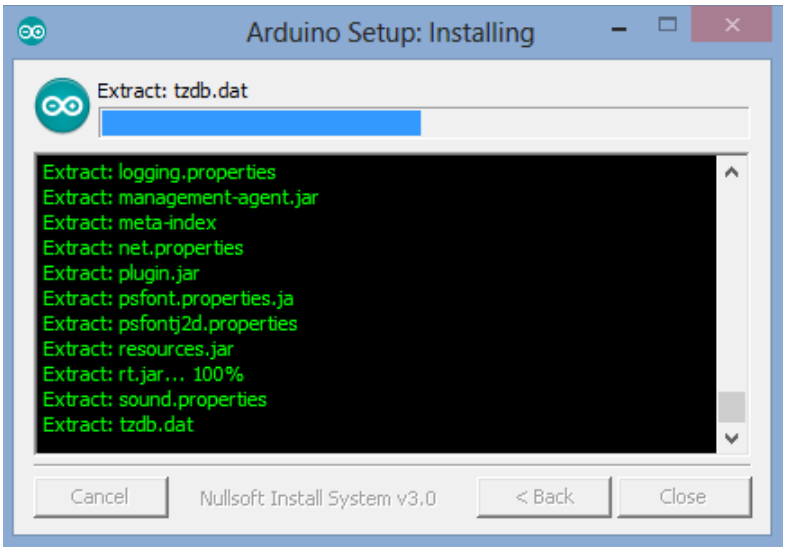
Emergerá una ventana con las opciones de instalación:

Figura N° 4. 10 Opciones de instalación Arduino



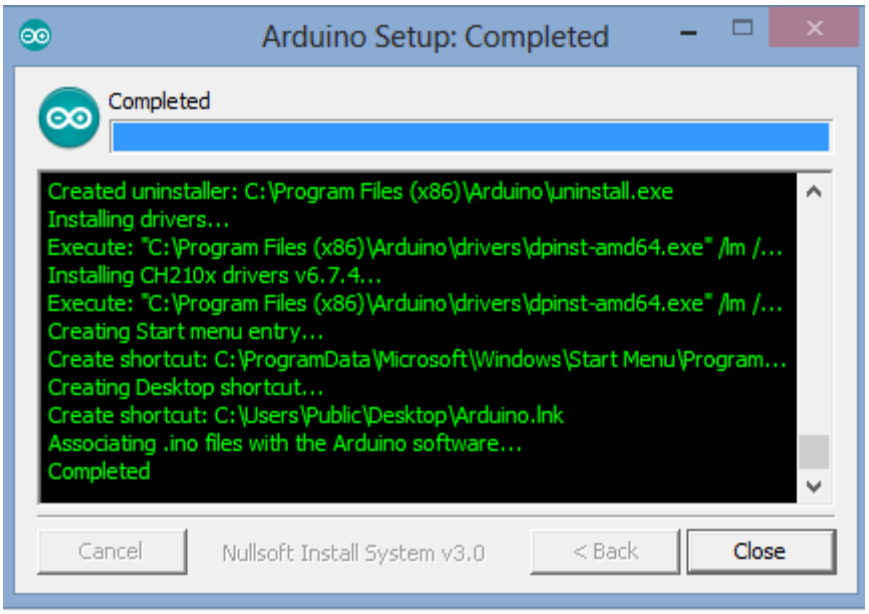
Una vez iniciado el proceso de instalación, se añadirán los archivos en la carpeta de instalación como primer paso.

Figura N° 4. 11 Carga de Instalación de Arduino



Concluida la instalación de los drivers necesarios para el uso de los puertos de la PC con Arduino, la ventana aparecerá de la siguiente forma:

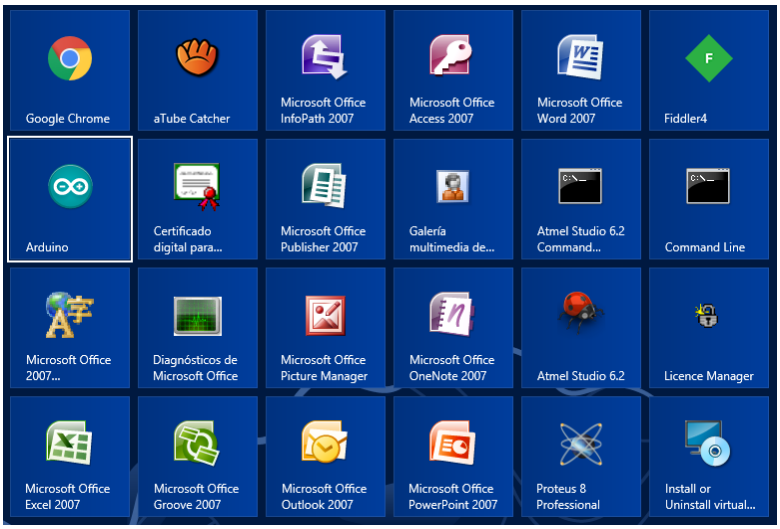
Figura N° 4. 12 Instalación de Arduino completada



Para concluir el proceso de instalación de Arduino procederemos a dar clic izquierdo en el botón CLOSE.

Concluida la instalación, procedemos a ejecutar el programa, para ello buscamos el icono de Arduino en la lista de los programas instalados en nuestra PC:

Figura N° 4. 13 Icono de Arduino en lista de programas instalados



Se ejecutará el programa y emergerá la siguiente pantalla, indicando que se están cargando los recursos necesarios para la ejecución del programa.

Figura N° 4. 14 Pantalla de inicialización de Arduino

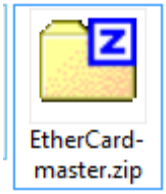
La siguiente pantalla a aparecer es el IDE de Arduino en sí, tiene la apariencia mostrada a continuación, se observa que por defecto aparecen las dos funciones principales: loop (bucle infinito), setup (configuración).

b. Desarrollo de proyecto con módulos Ethernet:

- **Instalación de las librerías necesarias.**

  Bajar la librería a tu PC, se obtiene desde la siguiente dirección web: *https://github.com/jcw/ethercard/archive/master.zip*

Figura N° 4. 15 Librería Ethernet en comprimido

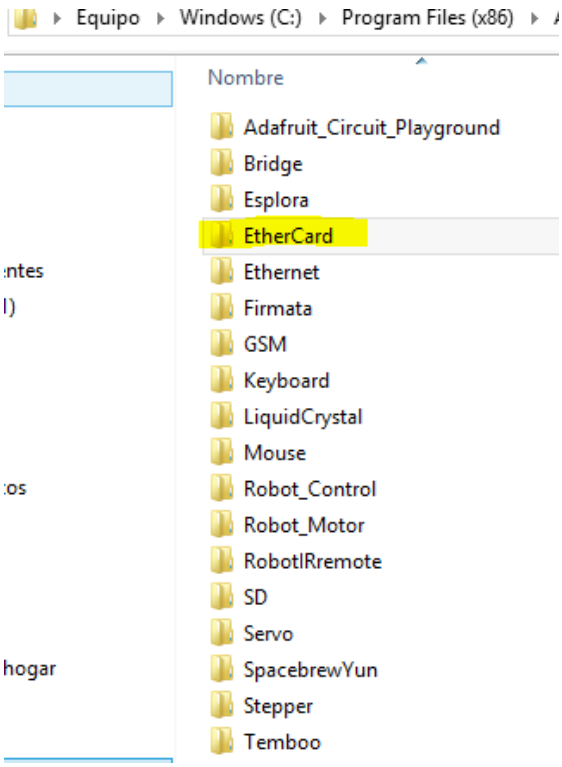


Se descomprime y se renombra la carpeta cómo “EtherCard”

Mover esta carpeta “EtherCard” bajo la carpeta libraries, donde está el ejecutable de Arduino. La cual tendrá una ruta similar a la siguiente:

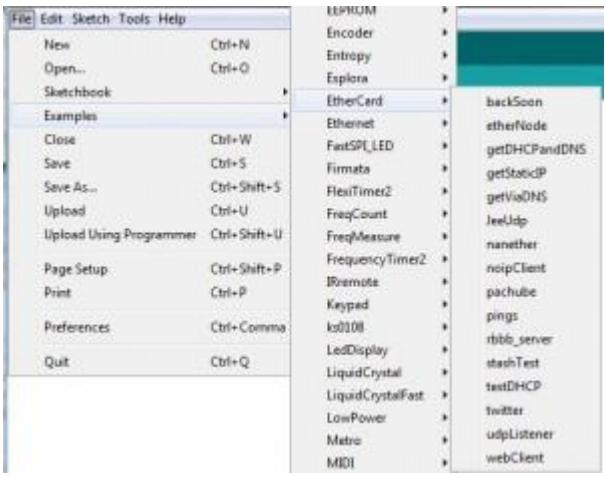
C:\Program Files (x86) \Arduino\libraries

Figura N° 4. 16 Librería copiada en libraries de Arduino



Si estaba abierta, reiniciar la aplicación de Arduino, para asegurarnos que se apliquen los cambios realizados.

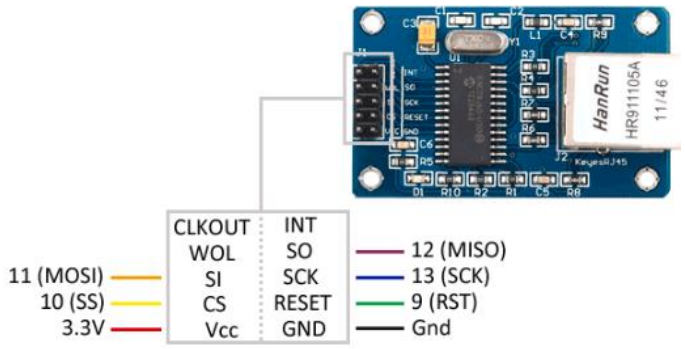
Figura N° 4. 17 Librería en Arduino IDE



- **Diseño de esquema de conexiones de comunicaciones.**

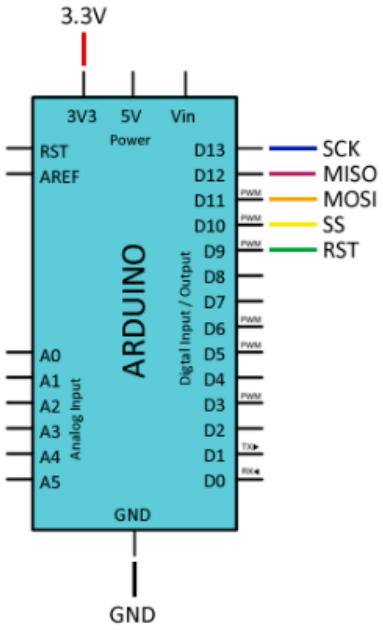
  La conexión de un módulo de Ethernet ENC28J60 es muy sencilla ya que la comunicación se realiza a través del SPI, a continuación, se muestra un esquema de las conexiones a realizar:

Figura N° 4. 18 Distribución de pines de ENC28J60



La conexión en este caso, vista desde Arduino, sería la siguiente:

Figura N° 4. 19 Conexión de ENC28J60 y Arduino



- **Desarrollo del código de programación.**

  Para la correcta conexión entre dos tarjetas Arduino se debe desarrollar dos códigos distintos, uno para cada tarjeta, para una a manera de servidor y otra como cliente.

- **Desarrollo del modo cliente.**

  En esta aplicación Arduino actúa como cliente, es decir, se conecta a una página web para leerla. Leer una página completa y volcarla por el puerto serie es muy lento, y es una de las muestras de las limitaciones de Arduino frente a un ordenador.

  Sin embargo, puede ser útil para Arduino capture información desde un servidor. Por ejemplo, podemos hacer que sincronice la hora, que lea una serie de parámetros de un fichero de texto, que realice una determinada acción si existe un fichero, etc.

  Para mostrar en este ejemplo esta capacidad de lectura de datos desde un servidor en Internet vamos a usar www.pasted.co, una de muchas páginas web que nos permiten añadir un texto para compartirlo con más gente.

  En la página www.pasted.co/2434bc64 he pegado el texto ~1.2.3.4.5~. Los '~' los usaremos como separadores para encontrar el texto deseado '1.2.3.4.5', que simula una serie de parámetros que queremos capturar de un servidor.

El siguiente ejemplo se conecta con esta dirección y realiza la búsqueda del texto 1.2.3.4.5, que muestra por puerto serie. En un ejemplo real emplearíamos estos valores, por ejemplo, para controlar un robot, cambiar los parámetros de medición de una estación, encender o apagar un dispositivo, etc.

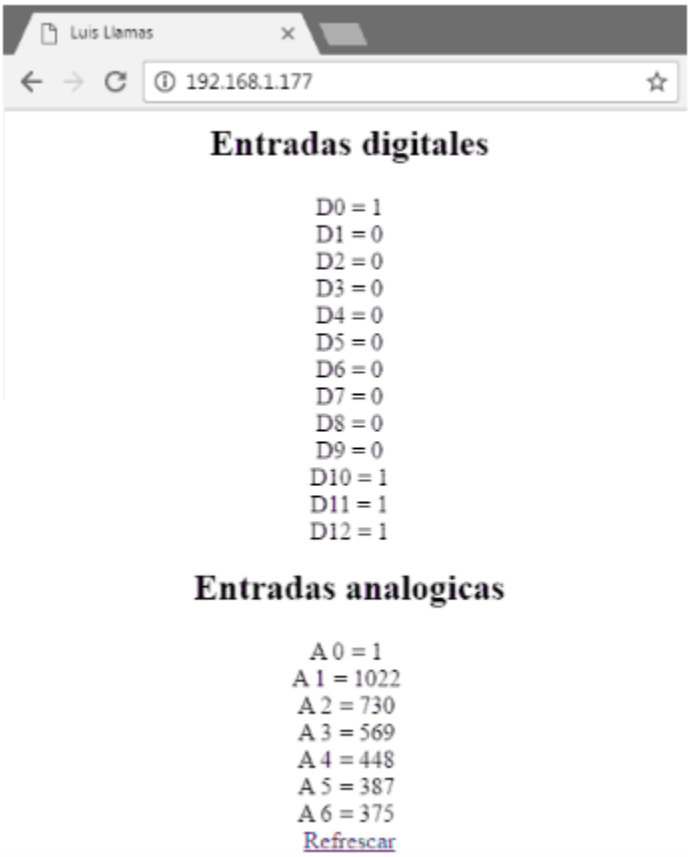
Podemos observar el código de programación para la tarjeta en modo cliente en el **Anexo 11.**

- **Desarrollo del modo servidor.**

  En este tipo de código Arduino actúa como servidor, es decir, devuelve una página web cuando un cliente (un PC, un móvil, otro Arduino…) se conecta a él.

  En este caso, vamos a mostrar una página web con el estado de las entradas digitales y analógicas de Arduino. Para ello, simplemente servimos una cadena de texto en HTML, en la que incluimos los valores de las entradas.

Figura N° 4. 20 Monitorización de señales de Arduino por Ethernet

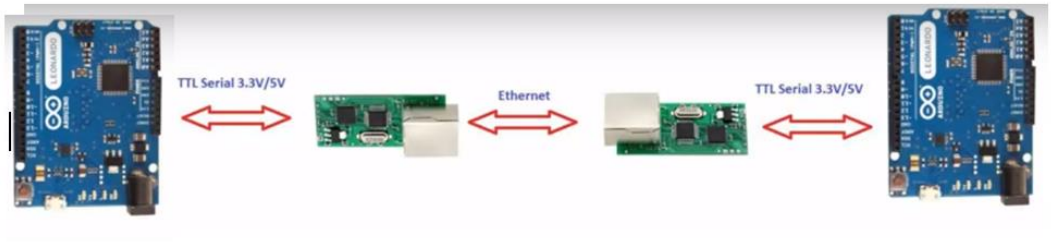


Podemos observar el código de programación para la tarjeta en modo servidor en el **Anexo 12.**

- **Prueba de conexión entre dos Arduino.**

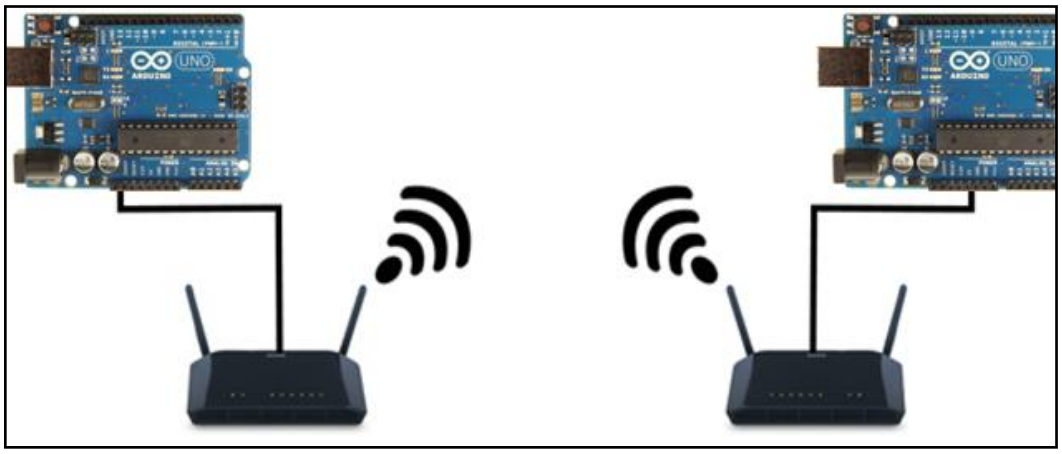
  La conexión entre dos tarjetas Arduino a través de Ethernet puede ser lograda de diversas maneras, pero todas las vías guardan una relación a través de la conexión base entre dos tarjetas de manera general, este esquema es mostrado en la **Figura 4.21**

Figura N° 4. 21 Conexión de dos tarjetas Arduino por Ethernet



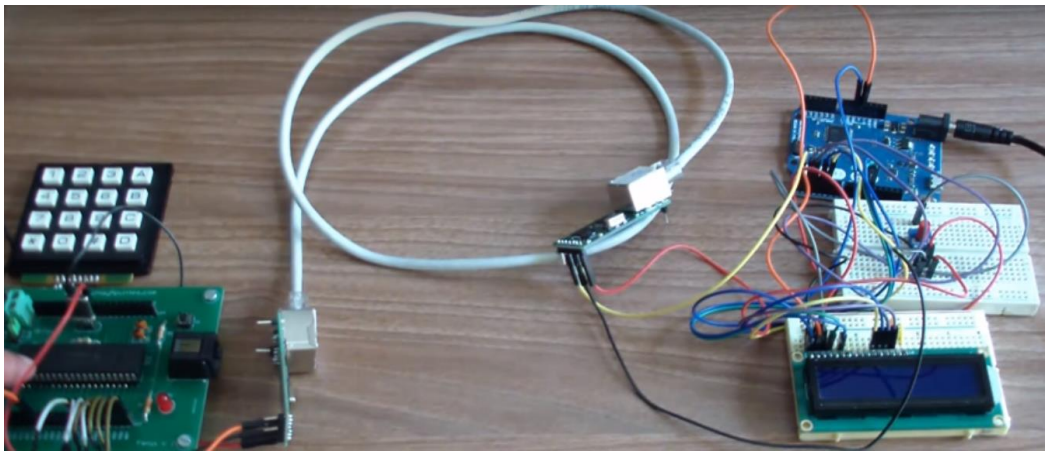
Una de las formas a conectar seria a través de routers y switch, elementos de telecomunicaciones que nos permitirían expandir la distancia de transmisión de datos entre dos controladores:

Figura N° 4. 22 Conexión de dos tarjetas Arduino por Ethernet a larga distancia



A manera de establecer pruebas en la conexión de los dos autómatas, se procede a probar con el módulo ECN28J60 la conexión a través de un cable de red.

Figura N° 4. 23 Conexión de dos tarjetas Arduino por cable Ethernet



La conexión es exitosa, el objetivo de esta implementación es poder transmitir a través de un teclado numérico el dato deseado hacia la pantalla de otro Arduino.

Cabe recordar que para que esta conexión y la transmisión de datos tenga éxito es necesario configurar una tarjeta como servidor y la otra como cliente.

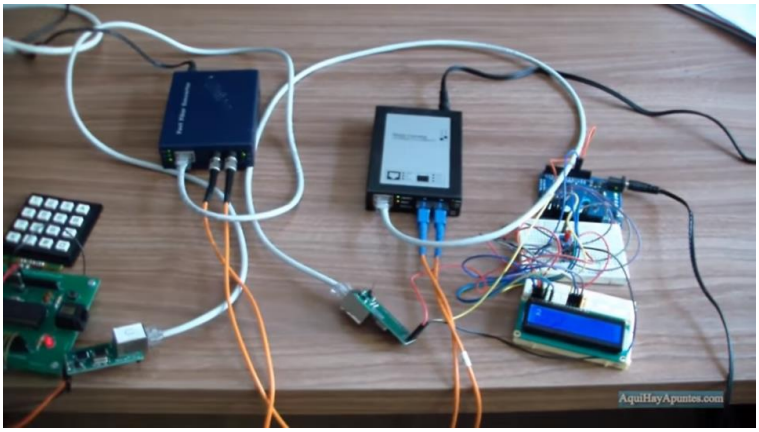
- **Explicación de posibles extensiones de esta parte del proyecto.**

Dado que la conexión previa es exitosa, y actualmente se cuenta con la tecnología necesaria, se somete a prueba la transmisión de los datos a través de fibra óptica.

Como en el caso anterior, una tarjeta es configurada en modo servidor y la otra tarjeta en modo cliente. Para la conexión a través de fibra óptica utilizamos dos conversores, uno de Ethernet a fibra óptica y otro de fibra óptica a Ethernet.

Esta conexión aumenta aún más la posible distancia para la transmisión de datos, acoplando paralelamente la interfaz desarrolla en PCB al sistema de comunicaciones conseguimos conectar dos procesos industriales a manera de maestro-esclavo. La prueba de la transmisión de los datos realizada se muestra en la **Figura N° 4.24**, donde también se muestran los conversores utilizados.

Figura N° 4. 24 Conexión de dos tarjetas Arduino por fibra óptica



c. **Desarrollo de proyecto con Scada Acimut Monitoriza:**

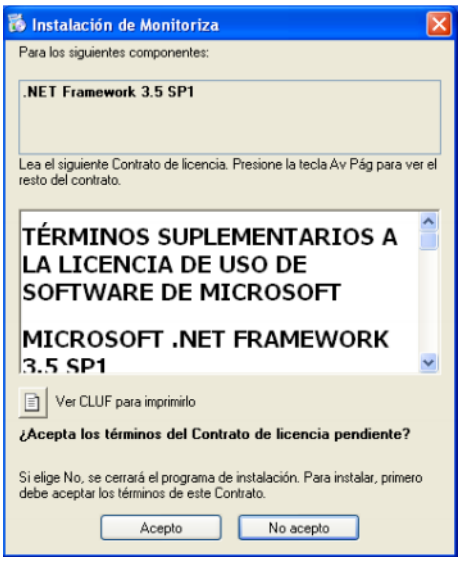
El desarrollo anterior nos muestra la viabilidad de la conexión entre dos tarjetas Arduino, por ende la conexión entre dos procesos industriales distinto a través de la interfaz diseñada, pero existen casos en los cuales se requiere una automatización más avanzada, y es obligatoria la aplicación de nuevas tecnologías, en este caso para la ampliación de las aplicaciones de esta investigación mostraremos cómo es posible diseñar pantallas para un sistema SCADA basado en nuestros controladores, con

el éxito de esta implementación se muestra que, con la correcta arquitectura y conexión, se lograría la conexión no solo entre dos autómatas, sino toda una red industrial basada en Arduino conectados a nuestra interfaz, reduciendo considerablemente los costos de la aplicación de la automatización en diferentes sectores.

- **Instalación de Acimut Monitoriza.**

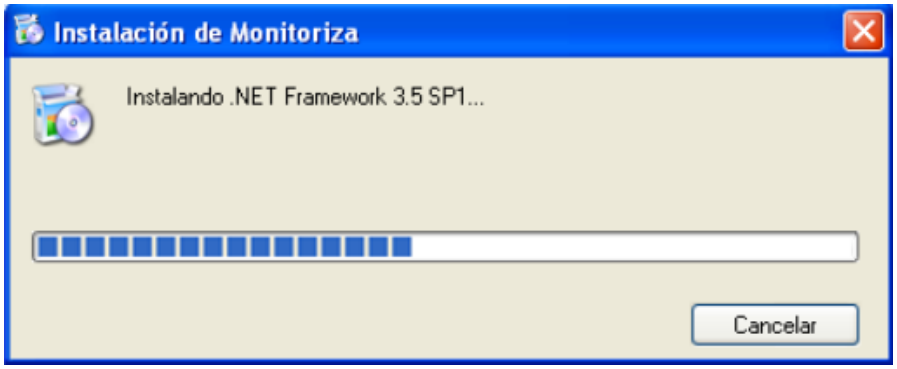
Monitoriza tiene una instalación muy sencilla. Consta de dos ficheros, setup.exe e Instalación Monitoriza.msi. Haciendo un doble clic sobre setup.exe se inicia la instalación. Monitoriza requiere para instalarse el .Net Framework™ 3.5 SP1 de Windows™ en el caso que no se encuentre instalado nos aparecerá el dialogo mostrado en la **figura N° 4.25.**

Figura N° 4. 25 Términos de .NET framework



En ese dialogo, en el que se nos pedirá si aceptamos el Contrato de licencia del .Net Framework de Microsoft. A continuación, procede a descargar desde la web de Microsoft el paquete correspondiente e instalarlo.

Figura N° 4. 26 Instalación de .NET framework

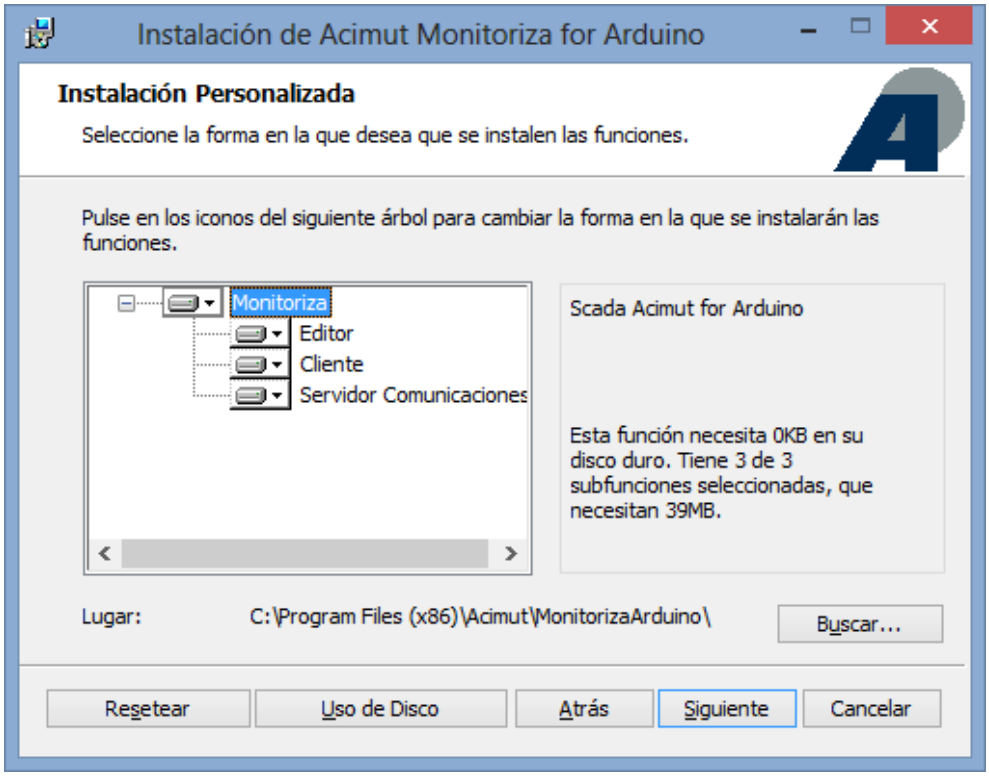


Una vez instalado se inicia la instalación de Acimut Monitoriza propiamente dicha, también se iniciará la instalación en caso ya tengamos previamente instalado el .NET Framework, finalmente para continuar hacemos clic izquierdo en el botón siguiente.

Figura N° 4. 27 Instalación de Acimut Monitoriza

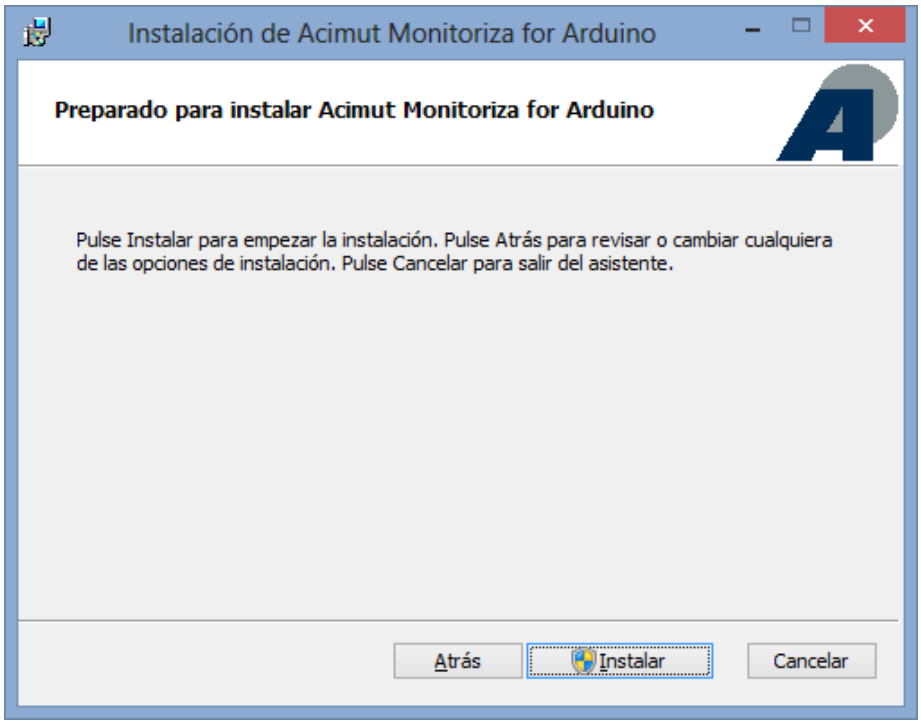
Pulsando sobre el botón Siguiente nos aparece la pantalla de instalación personalizada, en la que seleccionamos los elementos del sistema Acimut Monitoriza que se desea instalar. Estos elementos son el Editor que nos permite crear y modificar nuestros proyectos Scada, el Cliente mediante el cual establecemos el entorno de ejecución de los proyectos Scada y el Servidor de Comunicaciones a través del cual establecemos las comunicaciones tanto con los servidores OPC, como con las bases de datos y los autómatas.

Figura N° 4. 28 . Opciones de Acimut Monitoriza



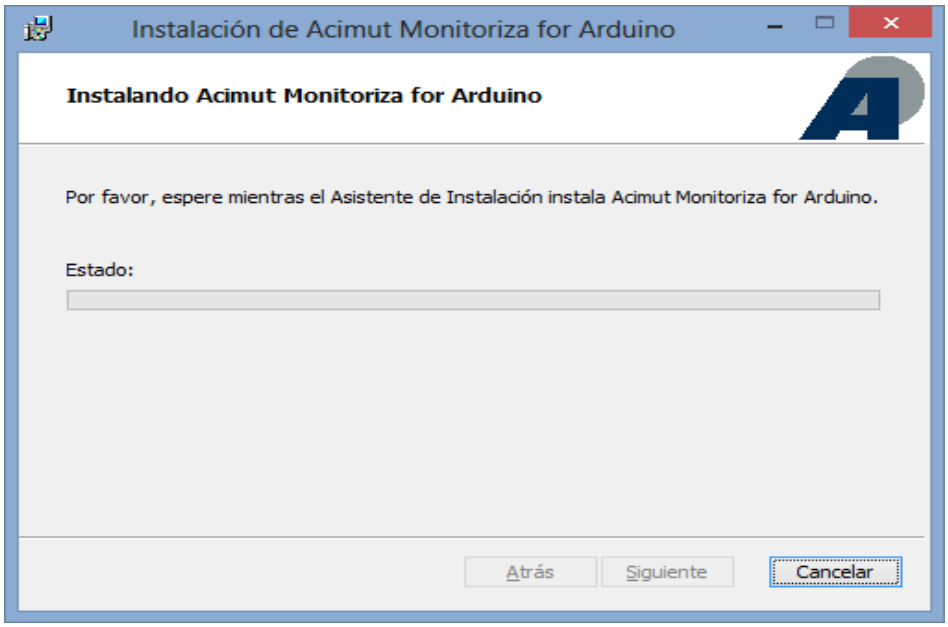
Una vez seleccionadas las funciones requeridas damos clic en el botón siguiente y aparecerá la siguiente pantalla:

Figura N° 4. 29 Permisos para instalar Acimut Monitoriza



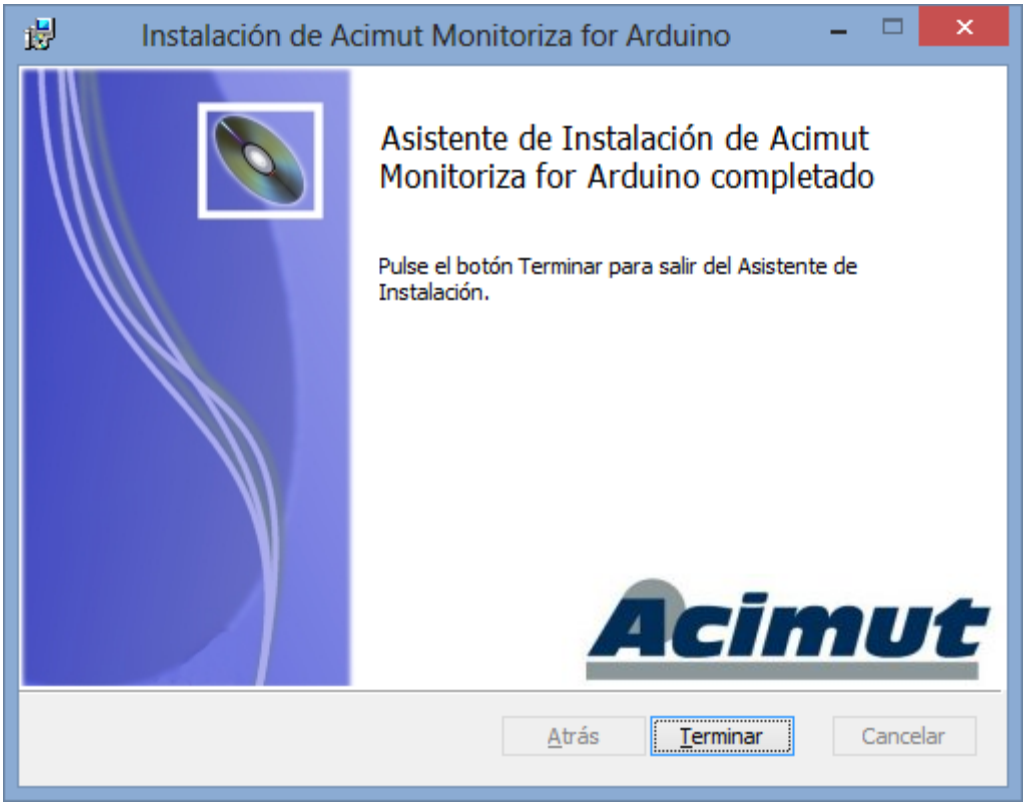
Con permisos de administrador en la PC hacemos clic en el botón Instalar:

Figura N° 4. 30 Proceso de instalación de Acimut Monitoriza



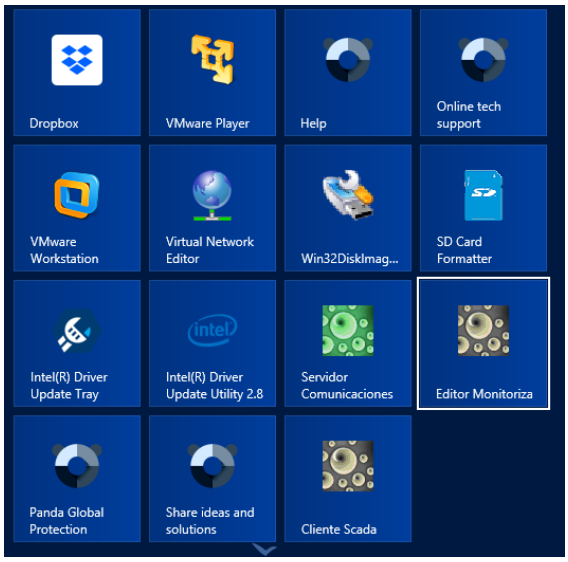
Concluida la instalación de Acimut nos aparecerá la siguiente pantalla, para concluir con la instalación hacemos clic en el botón Terminar.

Figura N° 4. 31 Instalación concluida de Acimut Monitoriza



Para ejecutar el Editor de Acimut, buscamos el icono del programa en nuestra lista de programas instalados:

Figura N° 4. 32 Icono de Acimut Monitoriza en lista de programas instalados.



d. Creación de pantallas para SCADA.

El proyecto a implementar constara en un sistema de transmisión de arena desde un tanque hacia un tanque de almacenamiento.

Se crean las diversas pantallas:

- Pantalla de monitoreo del proceso
- Pantalla de tendencias
- Pantalla de Alarmas

Estas pantallas son creadas de forma estándar en cualquier proceso de implementación de un sistema SCADA, demostrando la compatibilidad del sistema con actuales procesos industriales.

Figura N° 4. 33 Pantalla de monitoreo del proceso

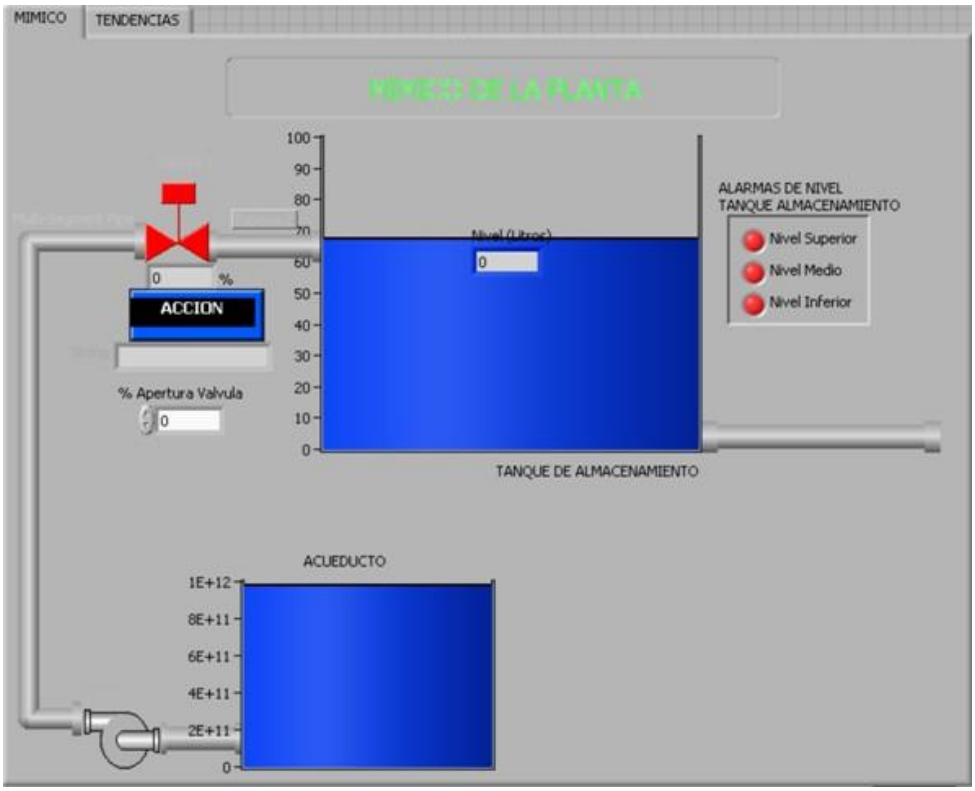


Figura N° 4. 34 Pantalla de tendencias

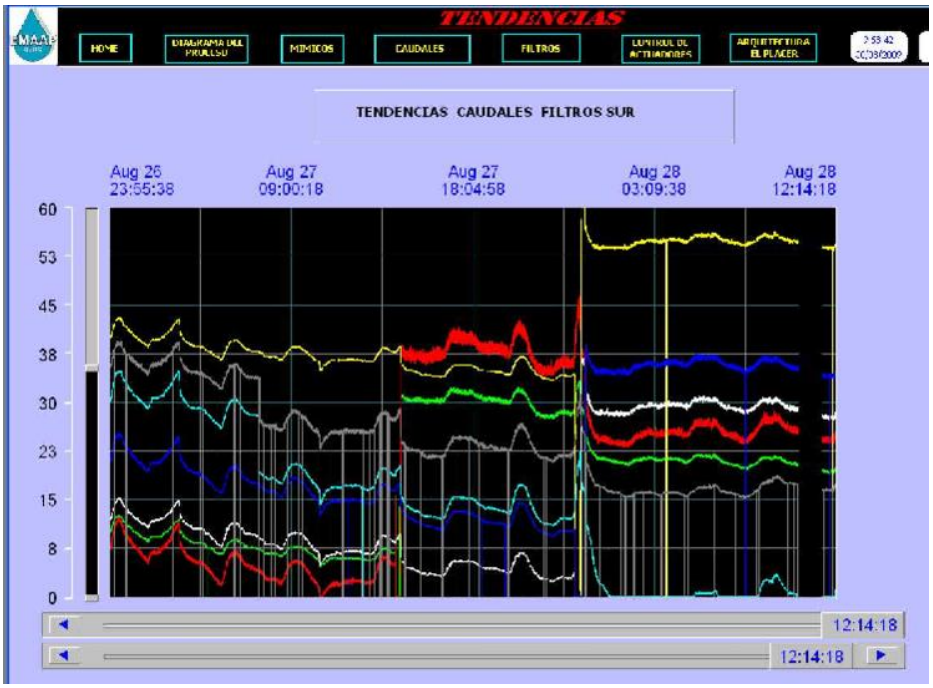
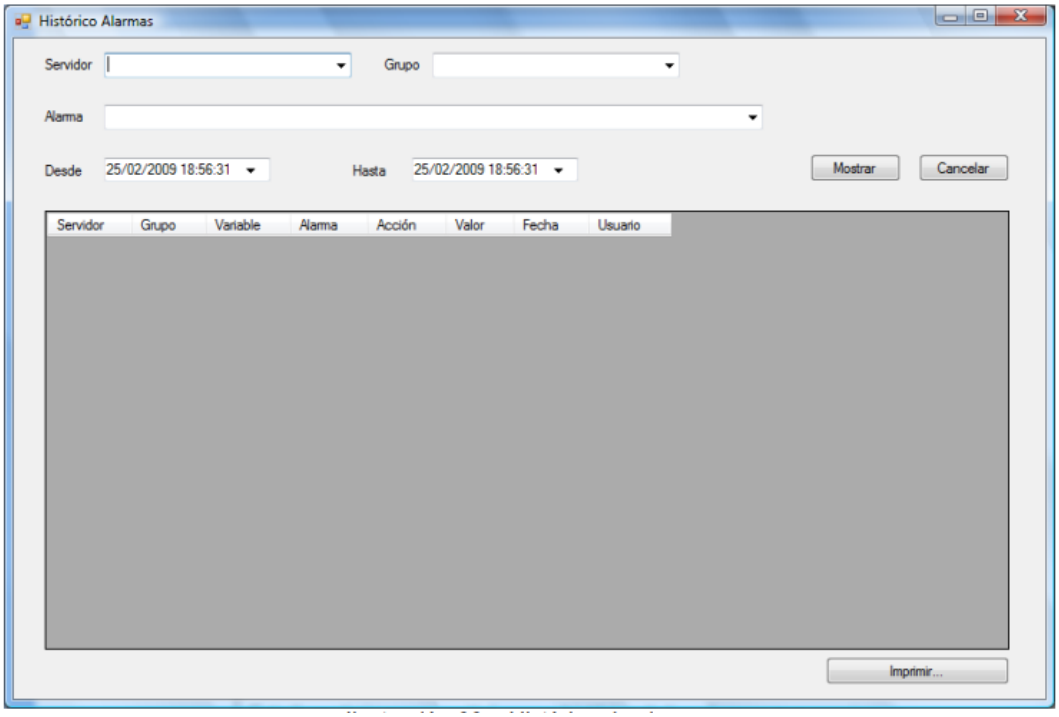


Figura N° 4. 35 Pantalla de histórico de alarmas



e. Creación de código Arduino para SCADA.

Para la ejecución del Scada en conjunto al autómata basado en Arduino es necesario descargar un código base en formato. ino, el código base lo podemos encontrar en la siguiente ruta:

http://www.acimut.com/monitoriza/monitorizaforarduino.html

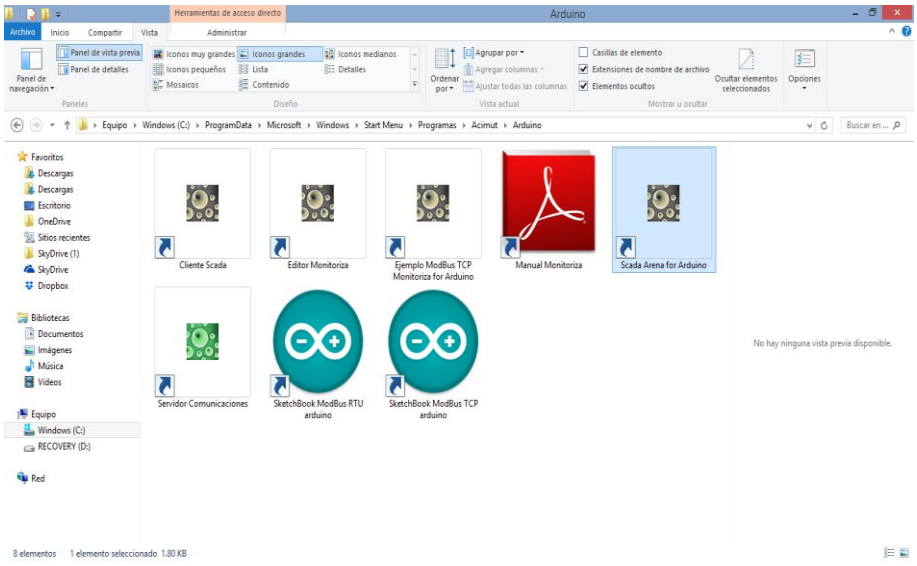
Una vez descargado el código base podemos desarrollar nuestro propio código acorde a las necesidades de nuestra aplicación.

El código para el Scada de nuestra aplicación lo podemos observar en el **Anexo 13.**

f. Ejecución del programa.

Una vez concluido el desarrollo de las pantallas y el programa, podemos pasar a ejecutarlo en cualquier PC que tenga instalado el Acimut Monitoriza, solo mediante la ejecución de un archivo Runtime como el mostrado a continuación:

Figura N° 4. 36 Runtime de sistema SCADA desarrollado



CAPITULO V

V. RESULTADOS

5.1. Ventajas Y Desventajas Del Proyecto.

Ventajas:

- Menor costo en comparación a los modelos de PLCs de características similares ofertados en la actualidad.
- Mayor cantidad de proyectos realizables debido al número de shields compatibles con la versión Arduino Mega 2560.
- Sencillez para programar mediante el lenguaje de programación Arduino.
- Posibilidad de programar en el lenguaje KOP (ladder) mediante el software adecuado, facilitando el aprendizaje de este lenguaje de programación sin la necesidad de adquirir equipos de alto costo.
- Compatibilidad con diversos Software usados en ingeniería, un ejemplo se muestra en el Anexo5, en el cual se muestra el uso de Arduino junto a LabView, software en el cual la conexión con el Arduino es gratuita.

Desventajas:

- Al programar la placa en el lenguaje Arduino, la ejecución del programa, es decir, la lectura de las entradas y la escritura de las

salidas, se da en serie (una a continuación de otra), mientras que en un PLC comercial este proceso se da de manera paralela.

- Cuando se programa la placa en KOP (ladder), se omiten algunas funciones especiales, según el software utilizado.

5.2. Exposición De Resultados.

- Se diseñó con éxito la interfaz adecuada para adaptar al Arduino UNO a los niveles de tensión que opera un PLC comercial.

- Se utilizaron componentes de soldadura superficial (SMD), dado el poco espacio que ocupan en la tarjeta, el precio de cada componente utilizado, junto al costo total, es mostrado en la Tabla6.

- Dada la necesidad de elaborar una tarjeta lo más robusta y óptima posible se opta por delegar la elaboración (impresión y soldadura) a una empresa dedicada a este rubro.

5.3. Discusión De Resultados.

- En este proyecto solo se adaptaron las entradas y salidas digitales del Arduino para la mayor sencillez de la elaboración.

- Solo se elaboró la tarjeta Controluino basada en Arduino UNO, como ejemplo y prueba, para que en un futuro esta placa sea elaborada con versiones más avanzadas de Arduino.

- El diseño es tal que, permite cambiar el Arduino por otro de similares características, en caso el que se encuentre en uso presente fallas o se encuentre defectuoso.

5.4. Factibilidad del proyecto.

Desde el inicio, este proyecto tuvo como fin principal el abaratar costos, como ya se dijo, los precios de cada componente, junto a la elaboración del Hardware, se muestran en la Tabla9, a lo largo de la implementación también se hizo notable el beneficio en costos en cuanto a implementación de Software. Dicho esto, se procederá a hacer un análisis de los costos.

Costos:

A continuación, se muestran los costos, de cada componente electrónico, entre los cuales se incluye al Arduino, según la cantidad utilizada en esta versión de la placa, como el costo de la elaboración del Hardware.

Tabla N° 5. 1 Precios detallados y total de Controluino UNO

Cantidad	Componente	Precio(dólares)
9	Lm2596S	6.02
9	Bobinas	3.096
8	4n25	2.32
4	TL283	2.122
4	Relé	2.424
4	Diodo(1N4004)	0.061
4	2n2222A	3.584
100	Resistencias	0.23

18	Capacitores	1.6
9	Diodo(1n5818)	0.11
1	Arduino Uno + Cable	6
	Elaboración	40
	Total	67.567

Los precios mostrados se basan en productos importados, estos productos son adquiridos desde la página web AliExpress ya que ofrece un envío a domicilio gratuito, haciendo que el precio de nuestra tarjeta sea más económico aún.

Beneficios:

Basándose en la tasa actual de cambio (3.35), obtenemos un precio final de 226.35 Nuevos Soles.

Un PLC comercial, de características similares a Controluino, tiene un costo aproximado de 550 Nuevos Soles.

En cuanto al costo, tenemos una diferencia de 323.65 Nuevos Soles, esto nos genera un gran beneficio económico.

Analizando el Hardware, si bien es cierto, solo nos limitamos a entradas y salidas digitales, la gran diferencia y una de las características más resaltantes de esta tarjeta, es que, nos brinda la posibilidad de poner utilizar diversos modulo, tales como bluetooth, GSM, GPS, entre otros, aumentando así la cantidad de aplicaciones que se pueden elaborar.

CAPITULO VI

VI. CONCLUSIONES:

- Pudimos adaptar las entradas y salidas del Arduino mediante circuitos electrónicos de potencia para funcionar a nivel de tensión industrial sin requerir una gran inversión económica, en un futuro se analizan implementación a nivel de software para ampliar las aplicaciones.

- El sistema SCADA de Acimut Monitoriza nos permitió enlazar dos de estos autómatas, sin costo alguno, mediante protocolos de comunicación para la transmisión de datos entre etapas de procesos industriales, además de entablar una mayor posibilidad al poder comunicar una mayor cantidad, añadir también que pudimos apreciar que mediante los equipos necesarios y la configuración correcta se puede extender la distancia entre los controladores.

- La ventaja que tiene Arduino frente a los núcleos de otros controladores, al ser OPEN SOURCE, permitió que podamos acceder, de manera gratuita, a un sistema de supervisión SCADA de licencia libre sin tener limitación en el uso de tags o variables, cabe resaltar que si se desea aumentar funcionalidades se puede trabajar en el código base de este aplicativo.

CAPITULO VII

VII. RECOMENDACIONES:

- Verificar los sensores y actuadores a utilizar, dado que pueden no ser compatibles con las características eléctricas de Controluino.

- Analizar las características de cada versión de Controluino, con el fin de elegir el adecuado, según la aplicación en la cual vaya a ser utilizado.

- En un futuro, se recomienda la adaptación de los pines analógicos del Arduino, aumentando así la variedad de aplicaciones en las cuales puede ser usado Controluino.

- Al adaptar LabView, junto al Arduino, en este proyecto Controluino, tendremos la posibilidad de manejar plantas industriales de manera virtual, eso genera un gran beneficio, económico, dado que los toolkits para comunicar Arduino a LabView son gratuitos, además de académico, dado que así se amplía el conocimiento de estudiante en cuanto al manejo de equipos industriales e instrumentación.

- Desarrollar un programa en lenguaje de programación industrial LADDER (KOP) supondría una mayor aceptación en el mercado para una futura implementación.

CAPITULO VIII

VIII. REFERENCIAS BIBLIOGRAFICAS:

[1]. (2014) PLC [Online].

Available:http://www.automation.siemens.com/mcms/programmab le-logiccontroller/en/simatic-s7-controller/Pages/Default.aspx

[2]. Página Oficial ARDUINO.

Disponible: http://arduino.cc/es/main/software#.U0X6sPl5N1Q

[3]. Jose Salomon (2013, diciembre 14) TARJETA ARDUINO.

[4]. Ferley Rojas, Yhon Puentes (2013, agosto 14) Arduino y Android una Pareja para Aplicaciones de Ubicuidad. Disponible: http://www.laccei.org/LACCEI2013Cancun/RefereedPapers/RP06

[5]. B. Godin, "Innovation: The history of a category, Working Paper No. 1, Projecton the Intellectual History of Innovation," Paper presented at: Polish Academy of Sciences, Committee for the Science, Warsaw, Poland, pp. 5–47, Diciembre 2008.

[6]. T. Levitt, Innovative Imitation, Harvard Business Review, pp. 63-70, septiembre 1966. [3] W. Kneale, The Idea of Invention, Proceedings of the British Academy, 1955, pp. 85-208.

[7]. H. Barnett, Innovation: the Basis of Cultural Change, 1953.

[8]. J. Schumpeter, The Instability of Capitalism, The Economic Journal, pp. 361-386, septiembre 1928.

[9]. M. Gladwell, The tipping point, 2002.

[10]. R. Dunbar, "Neocortex size as a constraint on group size in primates". Journal of Human Evolution 22 (6), pp 469–493, 1992.

[11]. T. Kelley & J. Littman, The Ten Faces of Innovation: IDEO's Strategies for Defeating the Devil's Advocate and Driving Creativity Throughout Your Organization, 2006.

[12] .G´omez-de-Gabriel, J.M. (2014) Piero Mobile Robot Platform http://gomezdegabriel.com/wordpress/ projects/piero-mobile-robot-platform/

[13]. Lam´ar, K. and Kocsis, A. G. (2013), “Implementation of speed measurement for electrical drives equipped with quadrature encoder in LabView FPGA”, Acta Technica Corviniensis - Bulletin of Engineering.

[14]. Mathwoks, MakerZone Arduino, Raspberry PI and Lego Mindstrorms Resources, http://makerzone.mathworks.com/, Consultado jun. 2014.

[15]. Ogata, K. (1999), “Ingenieria de Control Moderna (Spanish Edition)”, Prentice Hall, pp. 670 - 679.

[16]. Robot Electronics, EMG30 mounting bracket and wheel specification, http://www.robotelectronics.co.uk/htm/emg30.htm, Consultado jun. 2014.

[17]. Rogers J.R. and McVay, R.C. (2012), “Graphical Microcontroller Programming”, IEEE International Conference onTechnologies for

[18]. Practical Robot Applications (TePRA), pp. 48 - 52.

[19]. Seeed Studio, Grove Mega Shield http://www.seeedstudio.com/wiki/Grove - Mega Shield, Consultado jun.2014.

IX. ANEXOS:

- Matriz de consistencia
- Desarrollo de PLC mDUINO
- Hoja técnica de Arduino 2560
- Hoja técnica de Arduino Uno
- Hoja técnica de Arduino Nano
- Hoja técnica de Arduino Shield Ethernet
- Hoja técnica de ENC28J60
- Hoja técnica de Relé 5vdc
- Hoja técnica de 4N25
- Hoja técnica de LM2596
- Placa PCB del controlador
- Código de programación para la tarjeta en modo Cliente
- Código de programación par la tarjeta en modo servidor
- Código de programación para SCADA con la tarjeta de Desarrollo

1. **Matriz de consistencia.**

"CONTROLADOR LÓGICO PROGRAMABLE DE SALIDA TIPO RELÉ, BASADO EN ARDUINO PARA TRANSMISIÓN DE DATOS ENTRE ETAPAS DE PROCESOS INDUSTRIALES"

PROBLEMAS	OBJETIVOS	HIPOTESIS	VARIABLES	TIPOS DE INVESTIGACIÓN
Problema General: ¿Es posible implementar un controlador Lógico Programable, cuyo núcleo sea un Arduino, de bajo costo, a fin de expandir la aplicación de la automatización y para transmisión de datos entre etapas de procesos Industriales? **Problema específico:** • ¿Es posible adaptar las entradas y salidas del Arduino para funcionar a nivel de tensión industrial? • ¿Es posible enlazar dos de estos autómatas para transmitir datos sin costo alguno? • ¿Es posible desarrollar un sistema de supervisión para este sistema?	**Objetivo general:** Implementar un controlador lógico programable, cuyo núcleo sea un Arduino, de bajo costo, a fin de expandir la aplicación de la automatización y para transmisión de datos entre etapas de procesos industriales. **Objetivos específicos:** • Adaptar las entradas y salidas del Arduino mediante circuitos electrónicos de potencia para funcionar a nivel de tensión industrial. • Enlazar dos de estos autómatas, sin costo alguno, mediante protocolos de comunicación para la transmisión de datos. • Desarrollar un sistema de supervisión para este sistema mediante protocolos de redes industriales.	**Hipótesis General.** Un controlador lógico programable, cuyo núcleo sea un Arduino, de bajo costo, a fin de expandir la aplicación de la automatización y para transmisión de datos entre etapas de procesos industriales mediante el uso de componentes electrónicos. **Hipótesis específicas:** • Adaptar las entradas y salidas del Arduino para funcionar a nivel de tensión industrial, mediante el uso de componentes electrónicos como optoacopladores, relés electromagnéticos, transistores, entre otros. • Enlazar dos de estos autómatas, sin costo alguno, mediante protocolos de comunicación para la transmisión de datos. • Desarrollar un sistema de supervisión para este sistema mediante protocolos de redes industriales y presentará un bajo costo frente a los sistemas existentes actualmente en el mercado.	**Variable Independiente.** Aplicación de un controlador lógico programable, de salida tipo relé, basado en Arduino **Variable dependiente.** Transmisión de datos entre etapas de procesos industriales	**Temporal subtipo Investigación Sincrónica.** Este trabajo de investigación es de tipo temporal sincrónica puesto que el estudio fue realizado en un pequeño periodo de tiempo de Agosto - Diciembre 2016. **Espacial:** Este trabajo de investigación es de tipo espacial puesto que hemos tomado como referencia un proceso industrial localizado en Lima

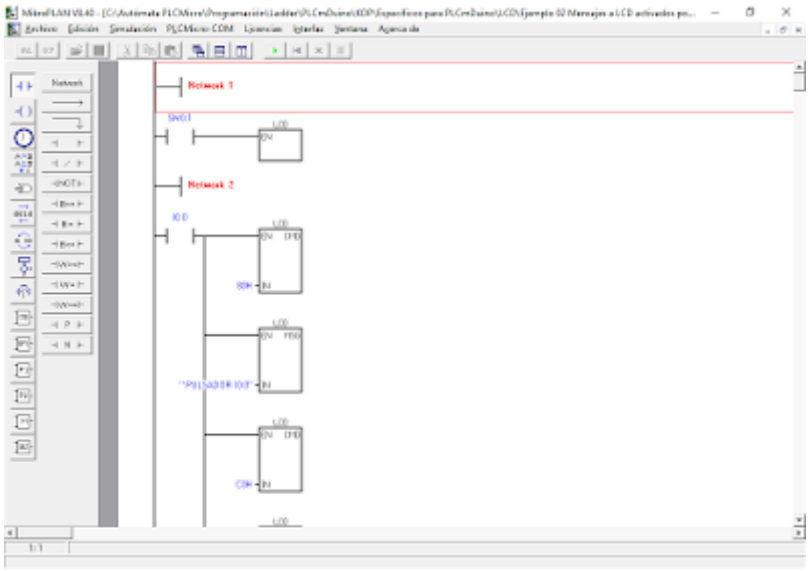
2. Desarrollo de PLCmDUINO

Consta de tres elementos básicos: Software de programación, firmware y una aplicación móvil

Software de programación MikroPLAN

Es una aplicación de escritorio que permite al usuario pueda programar la tarjeta Arduino utilizando diagramas de escalera o de lógica cableada. Incorpora un editor para diagramas en KOP y un compilador que primeramente traduce el esquema realizado por el usuario a instrucciones AWL similares a las que se utilizan en los PLC S7-200 de Siemens. Después, a partir de la lista de instrucciones, se genera una secuencia de códigos numéricos (bytecode) que posteriormente se envían y almacenan en la memoria EEPROM de Arduino.

La programación del chip puede realizarse desde MikroPLAN de forma alámbrica o bien inalámbrica utilizando un enlace Bluetooth. En este caso, la tarjeta Arduino deberá tener conectado un módulo HC-06 o HC-05 para comunicación Bluetooth. En la figura siguiente se muestra una captura de pantalla del editor KOP de MikroPLAN



Con MikroPLAN también puedes crear estrategias de control de movimiento con hasta 3 motores a pasos y hasta 11 servomotores de CD. También puedes programar estrategias de control con PWM. Para acceder a estas características requieres de una licencia de PLCmDUINO en modo extendido.

A partir de esta última entrega de MikroPLAN (24/07/2018) se incluye la primera versión del compilador VHDLDuino, que es una versión simplificada para Arduino del estándar VHDL (Very High Speed Integrated Circuits Hardware Description Language - Lenguaje de Descripción de Hardware para Circuitos Integrados de Muy Alta Velocidad). VHDLDuino te permitirá programar tu Arduino utilizando técnicas de implementación de circuitos digitales que se aplican en los PLD (Programmable Logic Device - Dispositivos Lógicos Programables) o FPGA (Field Programmable Gate Array - Arreglo de Compuertas Programables en Campo)

Firmware o software embebido PLCmDuino

Es el BIOS (Basic Input Output System - Sistema Básico de Entrada y Salida) que se graba en la tarjeta Arduino UNO R3 o MEGA 2560 y que permite cargar programas en KOP o AWL desde MikroPLAN. Su código fuente está escrito en *Processing* y una vez alojado en el chip, PLCmDuino recibe de MikroPLAN la secuencia de códigos que este software generó durante la compilación del diagrama de escalera de la estrategia de control del usuario. El BIOS se comunica con MikroPLAN utilizando el puerto serial con el que se instaló el IDE de Arduino por primera vez en la computadora del usuario. Una vez que todos los códigos se han recibido, PLCmDuino los almacena en un área de la memoria no volátil del chip. Cada uno de estos códigos representan instrucciones en lenguaje AWL compatibles con algunas del juego de instrucciones para los PLC S7-200 de Siemens y están integradas en el mismo código del BIOS. Cuando se corre un programa, PLCmDuino lee cada uno de estos códigos de su memoria; los decodifica y los relaciona con una alguna rutina específica que emula la acción equivalente en AWL.

Básicamente, el dúo MikroPLAN y PLCmDuino funciona de manera similar a Python y una JVM (Java Virtual Machine - Máquina Virtual de Java) la cual interpreta el "bytecode" generado por el primero y luego lo ejecuta en una PC

3. Hoja técnica de Arduino Mega 2560.

Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Power

The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) 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 Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

4. Hoja técnica de Arduino Uno

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V

Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
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

Schematic & Reference Design

EAGLE files: arduino-uno-Rev3-reference-design.zip (NOTE: works with Eagle 6.0 and newer)
Schematic: arduino-uno-Rev3-schematic.pdf
Note: The Arduino reference design can use an Atmega8, 168, or 328, Current models use an ATmega328, but an Atmega8 is shown in the schematic for reference. The pin configuration is identical on all three processors.

Power

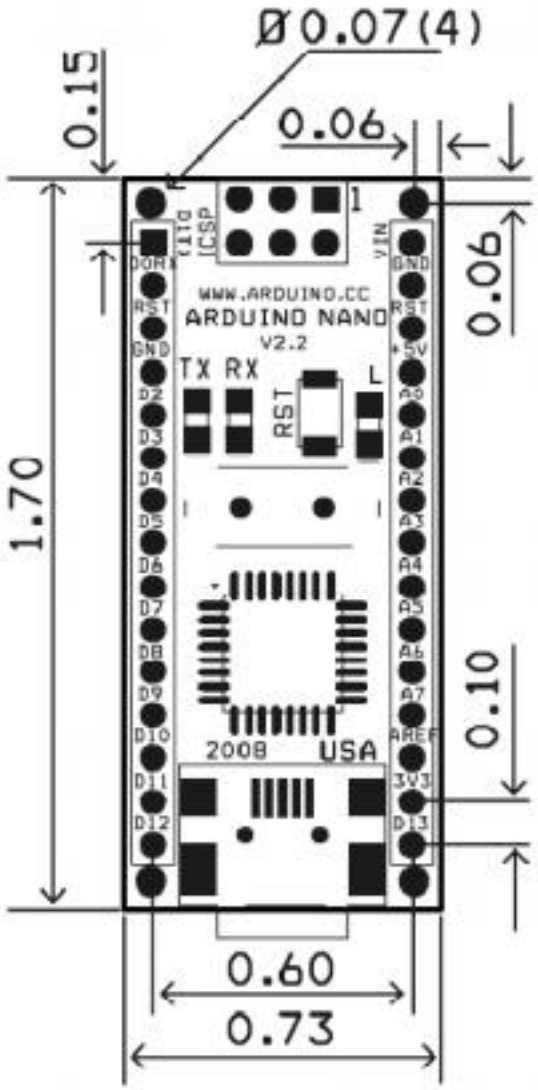
The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.
External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the 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:

5. Hoja técnica de Arduino Nano

Arduino Nano Bill of Material

Item Number	Qty.	Ref. Dest.	Description	Mfg. P/N	MFG	Vendor P/N	Vendor
1	5	C1,C3,C4,C7,C9	Capacitor, 0.1uF 50V 10% Ceramic X7R 0805	C0805C104K5RACTU	Kemet	80-C0805C104K5R	Mouser
2	3	C2,C8,C10	Capacitor, 4.7uF 10V 10% Tantalum Case A	T491A475K010AT	Kemet	80-T491A475K010	Mouser
3	2	C5,C6	Capacitor, 18pF 50V 5% Ceramic NOP/COG 0805	C0805C180J5GACTU	Kemet	80-C0805C180J5G	Mouser
4	1	D1	Diode, Schottky 0.5A 20V	MBR0520LT1G	ONSemi	863-MBR0520LT1G	Mouser
5	1	J1,J2	Headers, 36PS 1 Row	68000-136HLF	FCI	649-68000-136HLF	Mouser
6	1	J4	Connector, Mini-B Recept Rt. Angle	67503-1020	Molex	538-67503-1020	Mouser
7	1	J5	Headers, 72PS 2 Rows	67996-272HLF	FCI	649-67996-272HLF	Mouser
8	1	LD1	LED, Super Bright RED 100mcd 640nm 120degree 0805	APT2012SRCPRV	Kingbright	604-APT2012SRCPRV	Mouser
9	1	LD2	LED, Super Bright GREEN 50mcd 570nm 110degree 0805	APHCM2012CGCK-F01	Kingbright	604-APHCM2012CGCK	Mouser
10	1	LD3	LED, Super Bright ORANGE 160mcd 601nm 110degree 0805	APHCM2012SECK-F01	Kingbright	04-APHCM2012SECK	Mouser
11	1	LD4	LED, Super Bright BLUE 80mcd 470nm 110degree 0805	LTST-C170TBKT	Lite-On Inc	160-1579-1-ND	Digikey
12	1	R1	Resistor Pack, 1K +/-5% 62.5mW 4RES SMD	YC164-JR-071KL	Yageo	YC164J-1.0KCT-ND	Digikey
13	1	R2	Resistor Pack, 68Ω +/-5% 62.5mW 4RES SMD	YC164-JR-07680RL	Yageo	YC164J-68OCT-ND	Digikey
14	1	SW1	Switch, Momentary Tact SPST 150gf 3.0x2.5mm	B3U-1000P	Omron	SW1020CT-ND	Digikey
15	1	U1	IC, Microcontroller RISC 16kB Flash, 0.5kB EEPROM, 23 I/O Pins	ATmega168-20AU	Atmel	556-ATMEGA168-20AU	Mouser
16	1	U2	IC, USB to SERIAL UART 28 Pins SSOP	FT232RL	FTDI	895-FT232RL	Mouser
17	1	U3	IC, Voltage regulator 5V, 500mA SOT-223	UA78M05CDCYRG3	TI	595-UA78M05CDCYRG3	Mouser
18	1	Y1	Cystal, 16MHz +/-20ppm HC-49/US Low Profile	ABL-16.000MHZ-B2	Abracon	815-ABL-16-B2	Mouser

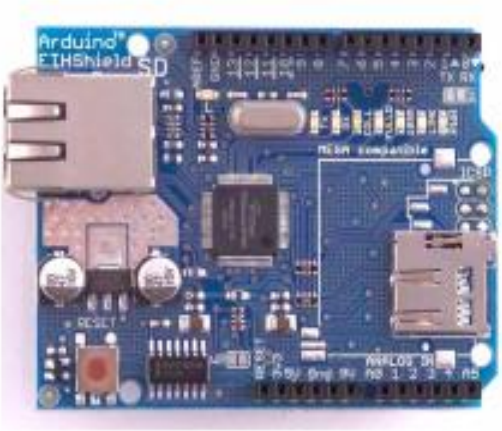
Arduino Nano Mechanical Drawing



ALL DIMENTIONS ARE IN INCHES

6. Hoja técnica de Arduino Shield Ethernet

Arduino Ethernet Shield



Download: arduino-ethernet-shield-05-schematic.pdf, arduino-ethernet-shield-05-reference-design.zip

Download: arduino-ethernet-shield-schematic.pdf, arduino-ethernet-shield-reference-design.zip

The Arduino Ethernet Shield allows an Arduino board to connect to the internet. It is based on the Wiznet W5100 ethernet chip (datasheet). The Wiznet W5100 provides a network (IP) stack capable of both TCP and UDP. It supports up to four simultaneous socket connections. Use the Ethernet library to write sketches which connect to the internet using the shield. The ethernet shield connects to an Arduino board using long wire-wrap headers which extend through the shield. This keeps the pin layout intact and allows another shield to be stacked on top.

The latest revision of the shield adds a micro-SD card slot, which can be used to store files for serving over the network. It is compatible with the Arduino Duemilanove and Mega (using the Ethernet library coming in Arduino 0019). An SD card library is not yet included in the standard Arduino distribution, but the sdfatlib by Bill Greiman works well. See this tutorial from Adafruit Industries for instructions (thanks Limor!).

The latest revision of the shield also includes a reset controller, to ensure that the W5100 Ethernet module is properly reset on power-up. Previous revisions of the shield were not compatible with the Mega and need to be manually reset after power-up. The original revision of the shield contained a full-size SD card slot; this is not supported.

Arduino communicates with both the W5100 and SD card using the SPI bus (through the ICSP header). This is on digital pins 11, 12, and 13 on the Duemilanove and pins 50, 51, and 52 on the Mega. On both boards, pin 10 is used to select the W5100 and pin 4 for the SD card. These pins cannot be used for general i/o. On the Mega, the hardware SS pin, 53, is not used to select either the W5100 or the SD card, but it must be kept as an output or the SPI interface won't work.

Note that because the W5100 and SD card share the SPI bus, only one can be active at a time. If you are using both peripherals in your program, this should be taken care of by the corresponding libraries. If you're not using one of the peripherals in your program, however, you'll need to explicitly deselect it. To do this with the SD card, set pin 4 as an output and write a high to it. For the W5100, set digital pin 10 as a high output.

The shield provides a standard RJ45 ethernet jack.

7. Hoja técnica de ENC28J60

MICROCHIP

ENC28J60

Stand-Alone Ethernet Controller with SPI Interface

Ethernet Controller Features

- IEEE 802.3 compatible Ethernet controller
- Integrated MAC and 10BASE-T PHY
- Supports one 10BASE-T port with automatic polarity detection and correction
- Supports Full and Half-Duplex modes
- Programmable automatic retransmit on collision
- Programmable padding and CRC generation
- Programmable automatic rejection of erroneous packets
- SPI Interface with clock speeds up to 20 MHz

Buffer

- 8-Kbyte transmit/receive packet dual port SRAM
- Configurable transmit/receive buffer size
- Hardware-managed circular receive FIFO
- Byte-wide random and sequential access with auto-increment
- Internal DMA for fast data movement
- Hardware assisted checksum calculation for various network protocols

Medium Access Controller (MAC) Features

- Supports Unicast, Multicast and Broadcast packets
- Programmable receive packet filtering and wake-up host on logical AND or OR of the following:
  - Unicast destination address
  - Multicast address
  - Broadcast address
  - Magic Packet™
  - Group destination addresses as defined by 64-bit hash table
  - Programmable pattern matching of up to 64 bytes at user-defined offset

Physical Layer (PHY) Features

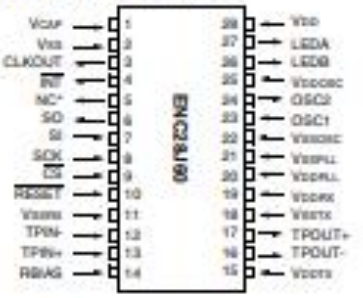
- Loopback mode
- Two programmable LED outputs for LINK, TX, RX, collision and full/half-duplex status

Operational

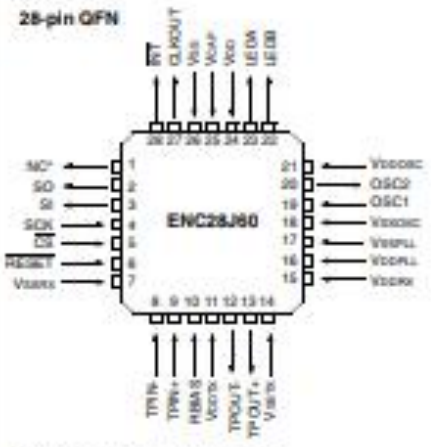
- Six interrupt sources and one interrupt output pin
- 25 MHz clock input requirement
- Clock out pin with programmable prescaler
- Operating voltage of 3.1V to 3.6V (3.3V typical)
- 5V tolerant inputs
- Temperature range: -40°C to +85°C Industrial, 0°C to +70°C Commercial (SSOP only)
- 28-pin SPDIP, SSOP, SOIC, QFN packages

Package Types

28-Pin SPDIP, SSOP, SOIC



28-pin QFN



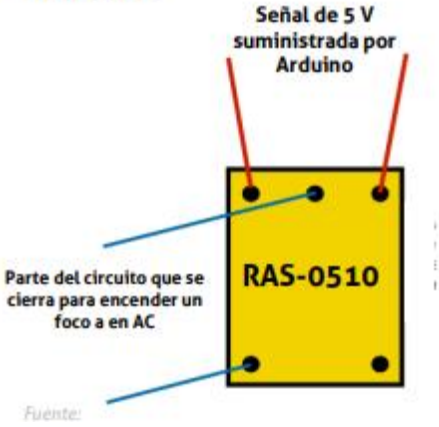
* Reserved pin; always leave disconnected.

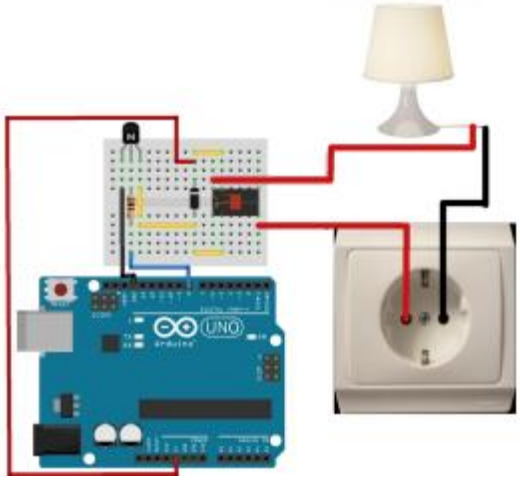
8. Hoja técnica de Relé 5vdc

UNIVERSIDAD PANAMERICANA Campus Bonaterra — Ingenierías UP

Relevador (RAS-0510)

Conexión:



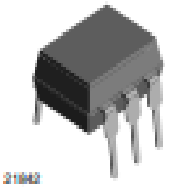


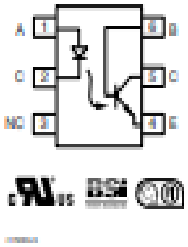
9. Hoja técnica de 4N25

4N25, 4N26, 4N27, 4N28

Vishay Semiconductors

Optocoupler, Phototransistor Output, with Base Connection





FEATURES

- Isolation test voltage 5000 V_{RMS}
- Interfaces with common logic families
- Input-output coupling capacitance < 0.5 pF
- Industry standard dual-in-line 6 pin package
- Compliant to RoHS directive 2002/95/EC and in accordance to WEEE 2002/96/EC

APPLICATIONS

- AC mains detection
- Read relay driving
- Switch mode power supply feedback
- Telephone ring detection
- Logic ground isolation
- Logic coupling with high frequency noise rejection

AGENCY APPROVALS

- UL1577, file no. E52744
- BSI: EN 60065:2002, EN 60950:2000
- FIMKO: EN 60950, EN 60065, EN 60335

DESCRIPTION

The 4N25 family is an industry standard single channel phototransistor coupler. This family includes the 4N25, 4N26, 4N27, 4N28. Each optocoupler consists of gallium arsenide infrared LED and a silicon NPN phototransistor.

ORDER INFORMATION	
PART	REMARKS
4N25	CTR > 20 %, DIP-6
4N26	CTR > 20 %, DIP-6
4N27	CTR > 10 %, DIP-6
4N28	CTR > 10 %, DIP-6

ABSOLUTE MAXIMUM RATINGS [1]				
PARAMETER	TEST CONDITION	SYMBOL	VALUE	UNIT
INPUT				
Reverse voltage		V_R	5	V
Forward current		I_F	80	mA
Surge current	t ≤ 10 µs	I_{FSM}	3	A
Power dissipation		P_{diss}	100	mW
OUTPUT				
Collector emitter breakdown voltage		V_{CEO}	70	V
Emitter base breakdown voltage		V_{EBO}	7	V
Collector current		I_C	50	mA
	t ≤ 1 ms	I_C	100	mA
Power dissipation		P_{diss}	150	mW

10. Hoja técnica de LM2596

LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator

1 Features

- 3.3-V, 5-V, 12-V, and Adjustable Output Versions
- Adjustable Version Output Voltage Range: 1.2-V to 37-V ± 4% Maximum Over Line and Load Conditions
- Available in TO-220 and TO-263 Packages
- 3-A Output Load Current
- Input Voltage Range Up to 40 V
- Requires Only 4 External Components
- Excellent Line and Load Regulation Specifications
- 150-kHz Fixed-Frequency Internal Oscillator
- TTL Shutdown Capability
- Low Power Standby Mode, I_Q, Typically 80 μA
- High Efficiency
- Uses Readily Available Standard Inductors
- Thermal Shutdown and Current-Limit Protection
- Create a Custom Design Using the LM2596 with the WEBENCH Power Designer

2 Applications

- Simple High-Efficiency Step-Down (Buck) Regulator
- On-Card Switching Regulators
- Positive to Negative Converter

3 Description

The LM2596 series of regulators are monolithic integrated circuits that provide all the active functions for a step-down (buck) switching regulator, capable of driving a 3-A load with excellent line and load regulation. These devices are available in fixed output voltages of 3.3 V, 5 V, 12 V, and an adjustable output version.

Requiring a minimum number of external components, these regulators are simple to use and include internal frequency compensation, and a fixed-frequency oscillator.

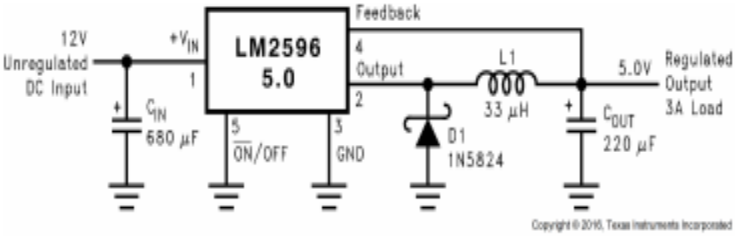
The LM2596 series operates at a switching frequency of 150 kHz, thus allowing smaller sized filter components than what would be required with lower frequency switching regulators. Available in a standard 7-pin TO-220 package with several different lead bend options, and a 7-pin TO-263 surface mount package.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM2596	TO-220 (7)	14.986 mm × 10.16 mm
	TO-263 (7)	10.10 mm × 8.89 mm

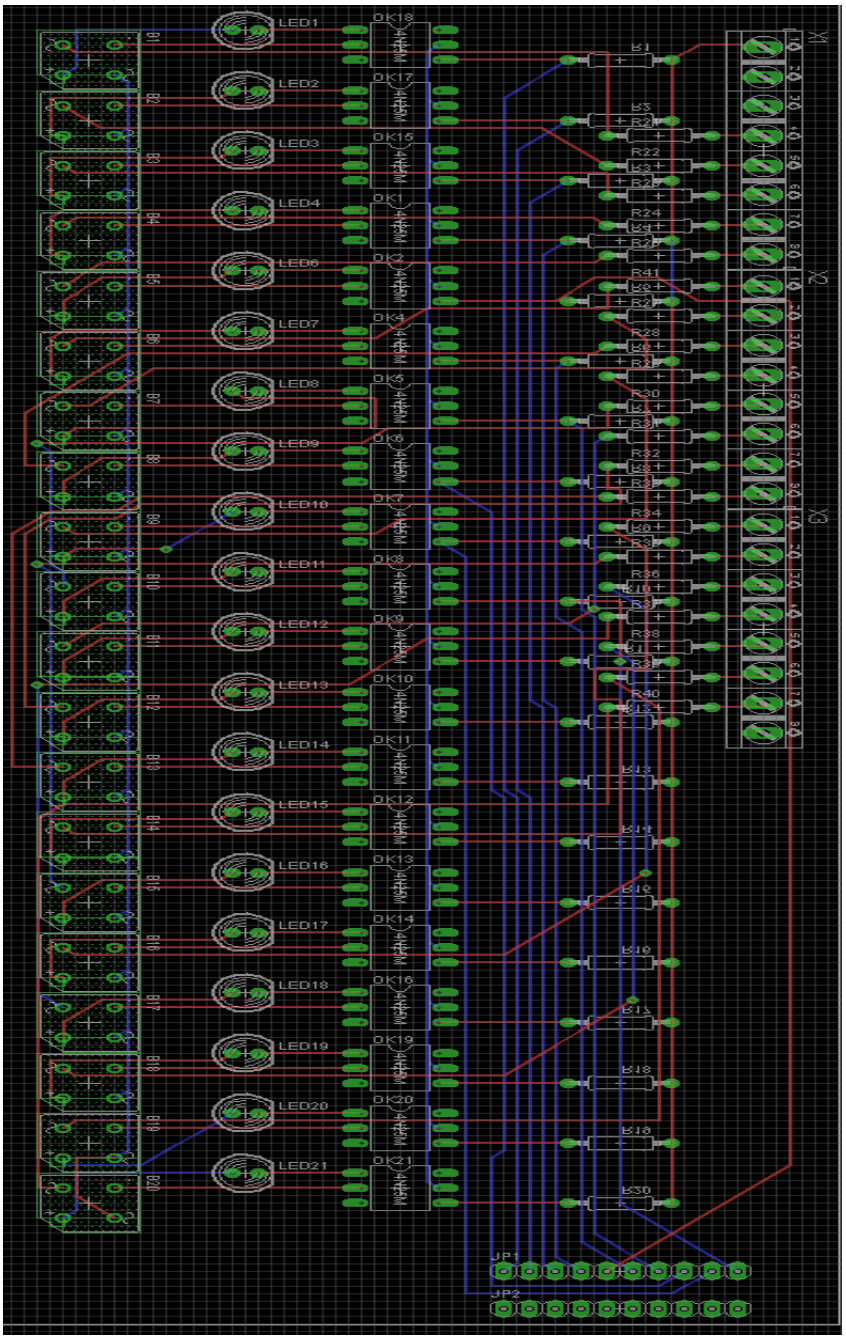
(1) For all available packages, see the orderable addendum at the end of the data sheet.

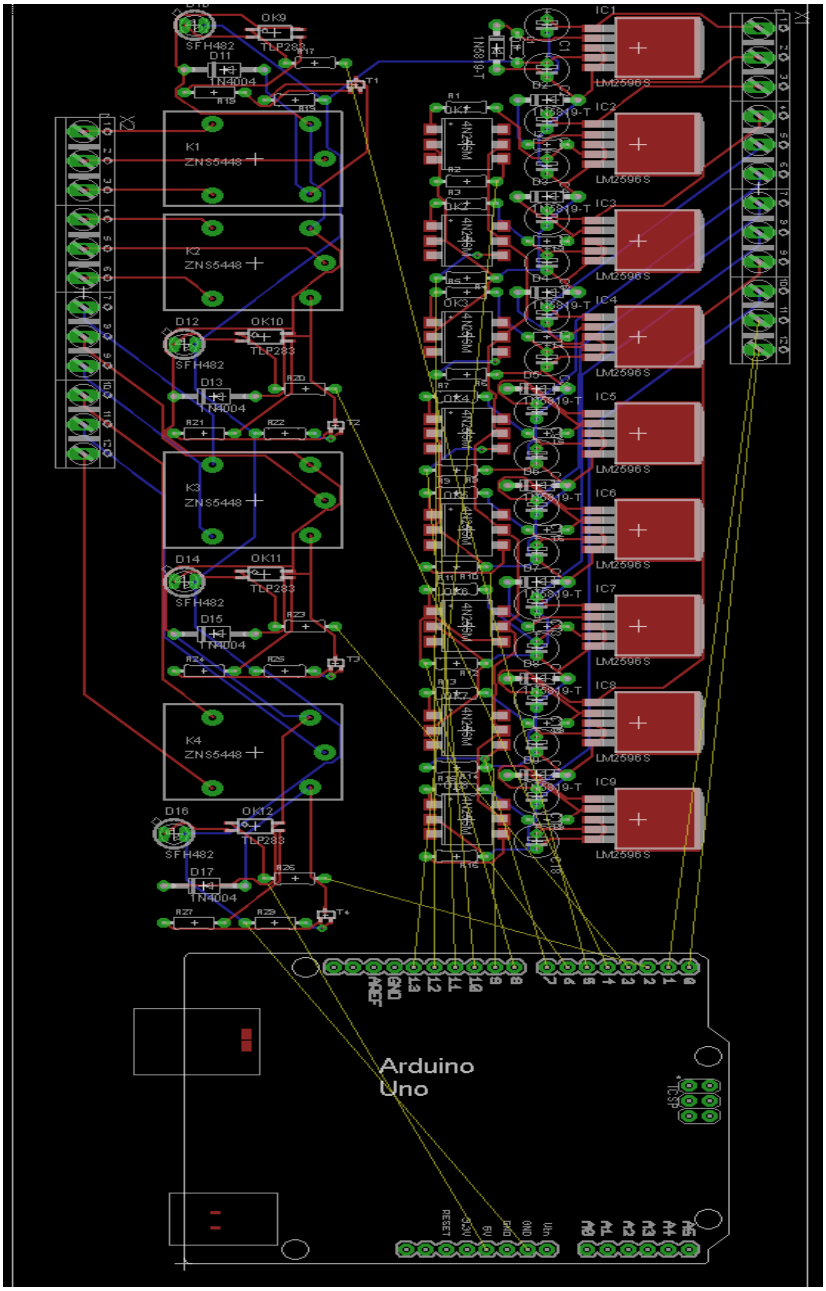
Typical Application



(Fixed Output Voltage Versions)

11. **Placa PCB del controlador**.




Arduino
Uno

12. Código de programación para la tarjeta en modo cliente.

```
#include <EtherCard.h>

static byte mymac[] = { 0xDD, 0xDD, 0xDD, 0x00, 0x01, 0x05 };
static byte myip[] = { 192, 168, 1, 177 };

byte Ethernet::buffer[700];
static uint32_t timer;

const char website[] PROGMEM = "www.pasted.co";
const char dataLocationC[] = "/2434bc64";

// called when the client request is complete
static void my_callback (byte status, word off, word len) {
  Serial.println(">>>");
  Ethernet::buffer[off+300] = 0;
  Serial.print((const char*) Ethernet::buffer + off);
  Serial.println("...");
}

void setup () {
  Serial.begin(57600);
  Serial.println(F("\n[webClient]"));

  if (ether.begin(sizeof Ethernet::buffer, mymac) == 0)
    Serial.println(F("Failed to access Ethernet controller"));
  if (!ether.dhcpSetup())
    Serial.println(F("DHCP failed"));

  ether.printIp("IP:  ", ether.myip);
  ether.printIp("GW:  ", ether.gwip);
  ether.printIp("DNS: ", ether.dnsip);

  if (!ether.dnsLookup(website))
    Serial.println("DNS failed");

  ether.printIp("SRV: ", ether.hisip);
}

void loop () {
  ether.packetLoop(ether.packetReceive());

  if (millis() > timer) {
    timer = millis() + 5000;
    Serial.println();
    Serial.print("<<< REQ ");

    // ether.browseUrl(PSTR(dataLocationC), "", website, my_callback); // No fun
ciona en IDE Standard
```

```
  ether.browseUrl(PSTR("/2434bc64"), "", website, my_callback);  // En entorno I
DE
 }
}
```

13. Código de programación para la tarjeta en modo servidor.

```
#include <EtherCard.h>

static byte mymac[] = { 0xDD, 0xDD, 0xDD, 0x00, 0x01, 0x05 };
static byte myip[] = { 192, 168, 1, 177 };
byte Ethernet::buffer[700];

void setup() {

Serial.begin(9600);

if (!ether.begin(sizeof Ethernet::buffer, mymac, 10))
 Serial.println("No se ha podido acceder a la controlador Ethernet");
else
Serial.println("Controlador Ethernet inicializado");

if (!ether.staticSetup(myip))
 Serial.println("No se pudo establecer la dirección IP");
 Serial.println();
}

static word mainPage()
{
   BufferFiller bfill = ether.tcpOffset();
 bfill.emit_p(PSTR("HTTP/1.0 200 OKrn"
 "Content-Type: text/htmlrnPragma: no-cachernRefresh: 5rnrn"
  "<HTML><head><title>Luis Llamas</title></head>"
 "<body>"
 "<div style='text-align:center;'>"
"<h1>Entradas digitales</h1>"
 "Tiempo transcurrido : $L s"
 "<br /><br />D00: $D"
   "<br /><br />D01: $D"
 "<br /><br />D02: $D"
      "<br /><br />D03: $D"
  "<br /><br />D04: $D"
"<br /><br />D05: $D"
"<br /><br />D06: $D"
   "<br /><br />D07: $D"
  "<br /><br />D08: $D"
 "<br /><br />D09: $D"
 "<br /><br />D10: $D"
"<br /><br />D11: $D"
 "<br /><br />D12: $D"
```

```
   "<br /><br />D13: $D"

"<h1>Entradas analogicas</h1>"
 "<br /><br />AN0: $D"
 "<br /><br />AN1: $D"
 "<br /><br />AN2: $D"
"<br /><br />AN3: $D"
"<br /><br />AN4: $D"
"<br /><br />AN5: $D"
   "<br /><br />AN6: $D"
 "<br /><br />"
 "</body></HTML>"),
   millis() / 1000,
digitalRead(0),
  digitalRead(1),
  digitalRead(2),
 digitalRead(3),
  digitalRead(4),
  digitalRead(5),
 digitalRead(6),
digitalRead(7),
 digitalRead(8),
 digitalRead(9),
 digitalRead(10),
digitalRead(11),
digitalRead(12),
 digitalRead(13),
analogRead(0),
analogRead(1),
analogRead(2),
analogRead(3),
analogRead(4),
analogRead(5),
analogRead(6));

return bfill.position();
}

void loop()
{
// wait for an incoming TCP packet, but ignore its contents
if (ether.packetLoop(ether.packetReceive()))
{
ether.httpServerReply(mainPage());
}
}
```

14. Código de programación para SACAD con la tarjeta de Desarrollo

```
/* Modbus RTU common parameters, the Master MUST use the same parameters

enum {

    COMM_BPS = 19200, /* baud rate */

    MB_SLAVE = 1,          /* modbus slave id */

    PARITY = 'n' /* even parity */

/* slave registers */

enum {

    MB_CTRL,       /* Led control on, off or blink */

    MB_TIME,       /* blink time in milliseconds */

    MB_CNT,       /* counter */

    MB_REGS                /* total number of registers on slave */

};

int regs[MB_REGS];

int ledPin = 13;

unsigned long wdog = 0;        /* watchdog */

unsigned long tprev = 0;        /* previous time*/

int i = 0;

float Counter = 0;
```

```
void setup()
{
	/* configure modbus communication
	 * 19200 bps, 8N1, two-device network */
	configure_mb_slave(COMM_BPS, PARITY, 0);

	pinMode(ledPin, OUTPUT);
}
void loop()
{
	/* check for master requests*/
	if(update_mb_slave(MB_SLAVE, regs, MB_REGS))
		wdog = millis();
	if ((millis() - wdog) > 5000)  {	/* no comms in 5 sec */
		regs[MB_CTRL] = 0;	/* turn off led */
	}
	 Counter = Counter + 0.01;
	if (Counter > 10000) Counter=0;
	regs[2] = (int)Counter;
```

```
  /* the values in regs are set by the modbus master */
switch(regs[MB_CTRL]) {
case 0:
        digitalWrite(ledPin, LOW);
        break;
case 1:
        digitalWrite(ledPin, HIGH);
        break;
default: /* blink */
        if (millis() - tprev > regs[MB_TIME]) {
                if (LOW == digitalRead(ledPin)) {
                        digitalWrite(ledPin, HIGH);
                        /* this is how you change your holding registers
                          so the master can read values from the slave */
                        regs[MB_CNT]++;
                } else {
                        digitalWrite(ledPin, LOW);
                }
                tprev = millis();
        }
```

```
        }
}


/* global variables */
unsigned int Txenpin = 0;       /* Enable transmission pin, used on RS485 networks */
/* enum of supported modbus function codes. If you implement a new one, put its function code here ! */
enum {
        FC_READ_REGS  = 0x03,   //Read contiguous block of holding register
        FC_WRITE_REG  = 0x06,   //Write single holding register
        FC_WRITE_REGS = 0x10,    //Write block of contiguous registers
        FC_READ_DEV_ID = 0x2B  //Read device information
};
/* supported functions. If you implement a new one, put its function code into this array! */
const unsigned char fsupported[] = { FC_READ_REGS, FC_WRITE_REG, FC_WRITE_REGS, FC_READ_DEV_ID };
/* constants */
enum {
        MAX_READ_REGS = 0x7D,
```

```
        MAX_WRITE_REGS = 0x7B,
        MAX_MESSAGE_LENGTH = 256
};
enum {
        RESPONSE_SIZE = 6,
        EXCEPTION_SIZE = 3,
        CHECKSUM_SIZE = 2
};
/* exceptions code */
enum {
        NO_REPLY = -1,
        EXC_FUNC_CODE = 1,
        EXC_ADDR_RANGE = 2,
        EXC_REGS_QUANT = 3,
        EXC_EXECUTE = 4
};
/* positions inside the query/response array */
enum {
        SLAVE = 0,
        FUNC,
```

```
        START_H,
        START_L,
        REGS_H,
        REGS_L,
        BYTE_CNT
};
*
CRC
  INPUTS:
        buf   ->  Array containing message to be sent to controller.
        start ->  Start of loop in crc counter, usually 0.
        cnt   ->  Amount of bytes in message being sent to controller/
 OUTPUTS:
        temp  ->  Returns crc byte for message.
 COMMENTS:
        This routine calculates the crc high and low byte of a message.
        Note that this crc is only used for Modbus, not Modbus+ etc.

*************************************************************************
**********/
```

```
unsigned int crc(unsigned char *buf, unsigned char start,
unsigned char cnt)
{
    unsigned char i, j;
    unsigned temp, temp2, flag;
    temp = 0xFFFF;
    for (i = start; i < cnt; i++) {
        temp = temp ^ buf[i];
        for (j = 1; j <= 8; j++) {
            flag = temp & 0x0001;
            temp = temp >> 1;
            if (flag)
                temp = temp ^ 0xA001;
        }
    }
    /* Reverse byte order. */
    temp2 = temp >> 8;
    temp = (temp << 8) | temp2;
    temp &= 0xFFFF;
    return (temp);
```

```
}
/*
 * Start of the packet of a read_holding_register response
 */
void build_read_packet(unsigned char slave, unsigned char function,
unsigned char count, unsigned char *packet)
{
        packet[SLAVE] = slave;
        packet[FUNC] = function;
        packet[2] = count * 2;
}
* Start of the packet of a preset_multiple_register response
 */
void build_write_packet(unsigned char slave, unsigned char function,
unsigned int start_addr,
unsigned char count,
unsigned char *packet)
{
        packet[SLAVE] = slave;
        packet[FUNC] = function;
```

```
        packet[START_H] = start_addr >> 8;
        packet[START_L] = start_addr & 0x00ff;
        packet[REGS_H] = 0x00;
        packet[REGS_L] = count;
}
/*
 * Start of the packet of a write_single_register response
 */
void build_write_single_packet(unsigned char slave, unsigned char function,
        unsigned int write_addr, unsigned int reg_val, unsigned char* packet)
{
        packet[SLAVE] = slave;
        packet[FUNC] = function;
        packet[START_H] = write_addr >> 8;
        packet[START_L] = write_addr & 0x00ff;
        packet[REGS_H] = reg_val >> 8;
        packet[REGS_L] = reg_val & 0x00ff;
}

/*
```

```
{
        packet[SLAVE] = slave;
        packet[FUNC] = function + 0x80;
        packet[2] = exception;
}



/*****************************************************************
********
 *
 * modbus_query( packet, length)
 *
 * Function to add a checksum to the end of a packet.
 * Please note that the packet array must be at least 2 fields longer than
 * string_length.


void modbus_reply(unsigned char *packet, unsigned char string_length)
{
        int temp_crc;
```

```
	temp_crc = crc(packet, 0, string_length);
	packet[string_length] = temp_crc >> 8;
	string_length++;
	packet[string_length] = temp_crc & 0x00FF;
}

/*****************************************************************
******
 *
 * send_reply( query_string, query_length )
 *
 * Function to send a reply to a modbus master.
 * Returns: total number of characters sent

*****************************************************************
******/

int send_reply(unsigned char *query, unsigned char string_length)
{
	unsigned char i;
```

```
    if (Txenpin > 1) { // set MAX485 to speak mode
        UCSR0A=UCSR0A |(1 << TXC0);
        digitalWrite( Txenpin, HIGH);
        delay(1);
    }
    modbus_reply(query, string_length);
    string_length += 2;
    for (i = 0; i < string_length; i++) {
        Serial.write(query[i]);
    }
    if (Txenpin > 1) {// set MAX485 to listen mode
        while (!(UCSR0A & (1 << TXC0)));
        digitalWrite( Txenpin, LOW);
    }
    return i;           /* it does not mean that the write was succesful, though */
}
/*****************************************************************
******
 *
 *      receive_request( array_for_data )
```

```
*  * Function to monitor for a request from the modbus master.
*
* Returns:      Total number of characters received if OK
* 0 if there is no request
* A negative error code on failure

**************************************************************************
*****/
int receive_request(unsigned char *received_string)
{
      int bytes_received = 0;

      /* FIXME: does Serial.available wait 1.5T or 3.5T before exiting the loop? */
      while (Serial.available()) {
            received_string[bytes_received] = Serial.read();
            bytes_received++;
            if (bytes_received >= MAX_MESSAGE_LENGTH)
                  return NO_REPLY;   /* port error */
      }
```

```
        return (bytes_received);
}
/*****************************************************************
****
 *
 *        modbus_request(slave_id, request_data_array)
 *
 * Function to the correct request is returned and that the checksum
 * is correct.
 *
 * Returns:      string_length if OK
 *               0 if failed
 *               Less than 0 for exception errors
 *
 *        Note: All functions used for sending or receiving data via
 *              modbus return these return values.
 *
*****************************************************************
***/
```

```
int modbus_request(unsigned char slave, unsigned char *data)
{
    int response_length;
    unsigned int crc_calc = 0;
    unsigned int crc_received = 0;
    unsigned char recv_crc_hi;
    unsigned char recv_crc_lo;

    response_length = receive_request(data);

    if (response_length > 0) {
        crc_calc = crc(data, 0, response_length - 2);
        recv_crc_hi = (unsigned) data[response_length - 2];
        recv_crc_lo = (unsigned) data[response_length - 1];
        crc_received = data[response_length - 2];
        crc_received = (unsigned) crc_received << 8;
        crc_received =
            crc_received | (unsigned) data[response_length - 1];

        /********** check CRC of response **********/
```

```
            if (crccalc != crc_received) {
                return NO_REPLY;
            }

            /* check for slave id */
            if (slave != data[SLAVE]) {
                return NO_REPLY;
            }
        }
        return (response_length);
}

/**************************************************************************
****
 *
 *        validate_request(request_data_array, request_length, available_regs)
 *
 * Function to check that the request can be processed by the slave.
 *
 * Returns:      0 if OK
```

```
 *
{
    int i, fcnt = 0;
    unsigned int regs_num = 0;
    unsigned int start_addr = 0;
    unsigned char max_regs_num;

    /* check function code */
    for (i = 0; i < sizeof(fsupported); i++) {
        if (fsupported[i] == data[FUNC]) {
            fcnt = 1;
            break;
        }
    }
    if (0 == fcnt)
        return EXC_FUNC_CODE;

    if (FC_WRITE_REG == data[FUNC]) {
        /* For function write single reg, this is the target reg.*/
        regs_num = ((int) data[START_H] << 8) + (int) data[START_L];
```

```
        if (regs_num >= regs_size)
                return EXC_ADDR_RANGE;
        return 0;
}

if (FC_READ_DEV_ID == data[FUNC]) {
        /* For function device info.*/
        if (data[3] != 1) /* Only Basic device identification */
          return EXC_REGS_QUANT;
        return 0;
}
  /* For functions read/write regs, this is the range. */
regs_num = ((int) data[REGS_H] << 8) + (int) data[REGS_L];

/* check quantity of registers */
if (FC_READ_REGS == data[FUNC])
        max_regs_num = MAX_READ_REGS;
else if (FC_WRITE_REGS == data[FUNC])
        max_regs_num = MAX_WRITE_REGS;
```

```
    if ((regs_num < 1) || (regs_num > max_regs_num))
        return EXC_REGS_QUANT;

    /* check registers range, start address is 0 */
    start_addr = ((int) data[START_H] << 8) + (int) data[START_L];
    if ((start_addr + regs_num) > regs_size)
        return EXC_ADDR_RANGE;

    return 0;           /* OK, no exception */
}

{     int temp;
    unsigned int i;

    for (i = 0; i < query[REGS_L]; i++) {
        /* shift reg hi_byte to temp */
        temp = (int) query[(BYTE_CNT + 1) + i * 2] << 8;
        /* OR with lo_byte          */
        temp = temp | (int) query[(BYTE_CNT + 2) + i * 2];
```

```
            regs[start_addr + i] = temp;
        }
        return i;
}

int preset_multiple_registers(unsigned char slave,
unsigned int start_addr,
unsigned char count,
unsigned char *query,
int *regs)
{
        unsigned char function = FC_WRITE_REGS; /* Preset Multiple Registers */
        int status = 0;
        unsigned char packet[RESPONSE_SIZE + CHECKSUM_SIZE];

        build_write_packet(slave, function, start_addr, count, packet);

        if (write_regs(start_addr, query, regs)) {
            status = send_reply(packet, RESPONSE_SIZE);
        }
```

```
    return (status);
}
{
    unsigned char function = FC_WRITE_REG; /* Function: Write Single Register */
    int status = 0;
    unsigned int reg_val;
    unsigned char packet[RESPONSE_SIZE + CHECKSUM_SIZE];

    reg_val = query[REGS_H] << 8 | query[REGS_L];
    build_write_single_packet(slave, function, write_addr, reg_val, packet);
    regs[write_addr] = (int) reg_val;
/*
    written.start_addr=write_addr;
    written.num_regs=1;
*/
    status = send_reply(packet, RESPONSE_SIZE);

    return (status);
```

```
}


{
    unsigned char function = 0x03;          /* Function 03: Read Holding
Registers */
    int packet_size = 3;
    int status;
    unsigned int i;
    unsigned char packet[MAX_MESSAGE_LENGTH];


    build_read_packet(slave, function, reg_count, packet);


    for (i = start_addr; i < (start_addr + (unsigned int) reg_count);
          i++) {
              packet[packet_size] = regs[i] >> 8;
        packet_size++;
        packet[packet_size] = regs[i] & 0x00FF;
        packet_size++;
    }
```

```
    status = send_reply(packet, packet_size);


    return (status);
}


{
    unsigned char function = 0x2B;          /* Function 2B: Read Deice identification */
    unsigned char MEIType = 0x0E;   /* MEI Type */
    unsigned char ReadDeviceIDCode = 0x01; /* Read device id code */
    unsigned char ConformityLevel = 0x01; /* Conformity level */
    unsigned char MoreFollows = 0x0; /* More Follows */
    unsigned char NextObjectID = 0x0; /* Next Object id */
    unsigned char NumObjects = 0x03; /* Number of objects */
    int packet_size = 3;
    int status;
    unsigned int i;
    unsigned char packet[MAX_MESSAGE_LENGTH];


    packet[SLAVE] = slave;
```

```
        packet[FUNC] = function;
        packet[2] = MEIType;
        packet[3] = ReadDeviceIDCode;
        packet[4] = ConformityLevel;
        packet[5] = MoreFollows;
        packet[6] = NextObjectID;

        unsigned int StartObject = 7;
        packet[StartObject] = NumObjects;
/* Object ID 0 */
        String company ="ARDUINO";
        int length=company.length();
        packet[StartObject+1] = 0;  /* Object id = 0 */
        packet[StartObject+2] = length;
        for (i = 0; i < length; i++ ) {
          packet[StartObject + 3 + i] = company.charAt(i);
        }
        StartObject = StartObject + 2 + length;
/* Object ID 1 */
        String product ="Model";
```

```
        length=product.length();
        packet[StartObject+1] = 1;  /* Object id = 1 */
        packet[StartObject+2] = length;
        for (i = 0; i < length; i++ ) {
          packet[StartObject + 3 + i] = product.charAt(i);
        }
        StartObject = StartObject + 2 + length;
/* Object ID 2 */
        String ver ="V1";
        length=ver.length();
        packet[StartObject+1] = 2;  /* Object id = 2 */
        packet[StartObject+2] = length;
        for (i = 0; i < length; i++ ) {
          packet[StartObject + 3 + i] = ver.charAt(i);
        }
        StartObject = StartObject + 2 + length + 1;

        status = send_reply(packet, StartObject);

        return (status);
```

```
}
void configure_mb_slave(long baud, char parity, char txenpin)
{
	Serial.begin(baud);

	switch (parity) {
	case 'e': // 8E1
		UCSR0C |= ((1<<UPM01) | (1<<UCSZ01) | (1<<UCSZ00));
		//	UCSR0C &= ~((1<<UPM00) | (1<<UCSZ02) | (1<<USBS0));
		break;
	case 'o': // 8O1
		UCSR0C |= ((1<<UPM01) | (1<<UPM00) | (1<<UCSZ01) | (1<<UCSZ00));
		//	UCSR0C &= ~((1<<UCSZ02) | (1<<USBS0));
		break;
	case 'n': // 8N1
		UCSR0C |= ((1<<UCSZ01) | (1<<UCSZ00));
		//	UCSR0C &= ~((1<<UPM01) | (1<<UPM00) | (1<<UCSZ02) | (1<<USBS0));
		break;
```

```
        default:
            break;
        }
        if (txenpin > 1) { // pin 0 & pin 1 are reserved for RX/TX
            Txenpin = txenpin; /* set global variable */
            pinMode(Txenpin, OUTPUT);
            digitalWrite(Txenpin, LOW);
        }

        return;
}

/*
 * update_mb_slave(slave_id, holding_regs_array, number_of_regs)
 *
 * checks if there is any valid request from the modbus master. If there is,
 * performs the action requested
 */

unsigned long Nowdt = 0;
```

```
unsigned int lastBytesReceived;
const unsigned long T35 = 5;

int update_mb_slave(unsigned char slave, int *regs,
unsigned int regs_size)
{       unsigned char query[MAX_MESSAGE_LENGTH];
        unsigned char errpacket[EXCEPTION_SIZE + CHECKSUM_SIZE];
        unsigned int start_addr;
        int exception;
        int length = Serial.available();
        unsigned long now = millis();
        unsigned char MEIType = 0x0E;   /* MEI Type for device identification */

        if (length == 0) {
                lastBytesReceived = 0;
                return 0;
        }
        if (lastBytesReceived != length) {
                lastBytesReceived = length;
                Nowdt = now + T35;
```

```
        return 0;
}
if (now < Nowdt)
        return 0;

lastBytesReceived = 0;

length = modbus_request(slave, query);
if (length < 1)
        return length;

exception = validate_request(query, length, regs_size);
if (exception) {
                build_error_packet(slave, query[FUNC], exception,
                errpacket);
                send_reply(errpacket, EXCEPTION_SIZE);
                return (exception);
}
```

```
Start_addr = ((int) query[START_H] << 8) +
        (int) query[START_L];
switch (query[FUNC]) {
    case FC_READ_REGS:
        return read_holding_registers(slave,
        start_addr,
        query[REGS_L],
        regs);
    break;
    case FC_WRITE_REGS:
        return preset_multiple_registers(slave,
        start_addr,
        query[REGS_L],
        query,
        regs);
    break;
    case FC_WRITE_REG:
        write_single_register(slave,
        start_addr,
```

```
                query,
                regs);
            break;
            case FC_READ_DEV_ID:
                if (query[START_H]==MEIType) {
                    return read_device_id(slave, query, length);
                }
            break;
        }
}
```