ESCUELA POLITÉCNICA DEL EJÉRCITO

EXTENSIÓN LATACUNGA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA ELECTRÓNICA E INSTRUMENTACIÓN

"DISEÑO E IMPLEMENTACIÓN DE SOFTWARE Y HARDWARE DE UN REGISTRADOR DE VARIABLES ELÉCTRICAS CON COMUNICACIONES ETHERNET BASADO EN TECNOLOGÍA ARDUINO Y SISTEMA DE SUPERVISIÓN HMI"

AUTOR:

GUSTAVO DE JESÚS LÓPEZ FRÍAS

TESIS PRESENTADA COMO REQUISITO PREVIO A LA OBTENCIÓN DEL GRADO DE:

INGENIERO EN ELECTRÓNICA E INSTRUMENTACIÓN

LATACUNGA, ENERO DEL 2013

DECLARACIÓN DE RESPONSABILIDAD

Yo, Gustavo de Jesús López Frías

DECLARO QUE:

El proyecto de grado titulado "DISEÑO E IMPLEMENTACIÓN DE SOFTWARE Y HARDWARE DE UN REGISTRADOR DE VARIABLES ELÉCTRICAS CON COMUNICACIONES ETHERNET BASADO EN TECNOLOGÍA ARDUINO Y SISTEMA DE SUPERVISIÓN HMI", ha sido desarrollado con base a una exhaustiva investigación, respetando derechos intelectuales de terceros, conforme las citas que constan al pie de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de mi autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance científico del proyecto de grado en mención.

Latacunga, Enero del 2013.

_____

Gustavo de Jesús López Frías

C.I. 1803199718

CERTIFICADO DE TUTORÍA

Ing. David Rivas (DIRECTOR DE TESIS)

Ing. Fabricio Pérez (CODIRECTOR DE TESIS)

CERTIFICAN

Que el trabajo titulado “DISEÑO E IMPLEMENTACIÓN DE SOFTWARE Y HARDWARE DE UN REGISTRADOR DE VARIABLES ELÉCTRICAS CON COMUNICACIONES ETHERNET BASADO EN TECNOLOGÍA ARDUINO Y SISTEMA DE SUPERVISIÓN HMI” realizado por el Sr. Gustavo López, ha sido guiado y revisado periódicamente y cumple normas estatuarias establecidas por la ESPE, en el Reglamento de Estudiantes de la Escuela Politécnica del Ejército.

Debido a que constituye un trabajo de excelente contenido científico y aplicable para el desarrollo profesional, si recomiendan su publicación.

El mencionado trabajo consta de un documento empastado y un disco compacto en el cual contiene los archivos en formato portátil de Acrobat (pdf). Se autoriza al Sr. Gustavo López que lo entregue a la Ing. Nancy Guerrón, en su calidad de Directora de la Carrera.

Latacunga, Enero del 2013

_____	_____
Ing. David Rivas	Ing. Fabricio Pérez
DIRECTOR	CODIRECTOR

CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado en su totalidad por el Sr. Gustavo de Jesús López Frías, bajo nuestra supervisión.

Latacunga, Enero del 2013

_____

Ing. David Rivas

DIRECTOR DEL PROYECTO

_____

Ing. Fabricio Pérez

CODIRECTOR DEL PROYECTO

_____

Ing. Nancy Guerrón

DIRECTORA DE CARRERA

_____

Dr. Rodrigo Vaca

SECRETARIO ACADÉMICO

AUTORIZACIÓN DE PUBLICACIÓN

Yo, Gustavo de Jesús López Frías

Autorizo a la Escuela Politécnica del Ejército la publicación, en la biblioteca virtual de la Institución del trabajo "DISEÑO E IMPLEMENTACIÓN DE SOFTWARE Y HARDWARE DE UN REGISTRADOR DE VARIABLES ELÉCTRICAS CON COMUNICACIONES ETHERNET BASADO EN TECNOLOGÍA ARDUINO Y SISTEMA DE SUPERVISIÓN HMI" cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y autoría.

Latacunga, Enero del 2013

_____

Gustavo de Jesús López Frías

C.I. 1803199718

DEDICATORIA

Dedico este proyecto de tesis a Dios por acompañarme en cada paso de mi vida y por haber puesto en mi camino a aquellas personas que han sido mi soporte para alcanzar esta meta.

A mi madre Cecilia, por darme la vida, amarme mucho, confiar en mí, y por darme una carrera, todo esto te lo debo a ti.

A mi abuela Olga, por quererme, compartir momentos significativos conmigo y apoyarme siempre.

A mi novia María José, por ser fuente de fuerza y aliento en los momentos difíciles.

A todos mis amigos, Darío, Marco, Daniela, Fabián, Sebastián, Alex, Ernesto, David, Fabricio, Irma, Omar, José, por compartir los buenos y malos momentos.

AGRADECIMIENTO

A mi madre, por brindarme su apoyo incondicional y enseñarme a perseverar a través de sus sabios consejos.

Al Ing. David Rivas, director de tesis y al Ing. Fabricio Pérez, codirector de tesis, por sus valiosas guías y asesoramientos para la realización de este proyecto de grado.

A todas aquellas personas que ayudaron directa e indirectamente en la realización de la misma.

ÍNDICE GENERAL

ANEXOS

A. Circuitos esquemáticos de las tarjetas Arduino.

B. Circuitos Impresos.

C. Programación de la tarjeta Arduino Nano.

D. Programación del registrador de variables eléctricas remoto.

E. Programación del registrador de variables eléctricas con supervisión HMI diseñada en LabVIEW

F. Programación del registrador de variables eléctricas remoto con monitoreo a través de un navegador web.

G. Circuitos esquemáticos del registrador de variables eléctricas.

H. Hojas técnicas

I. Manual de Usuario.

ÍNDICE DE TABLAS

CAPÍTULO I

FUNDAMENTOS TEÓRICOS

CAPÍTULO II

PRUEBAS Y RESULTADOS

DISEÑO DEL REGISTRADOR DE VARIABLES ELÉCTRICAS

CAPÍTULO III

PRUEBAS Y RESULTADOS

ÍNDICE DE FIGURAS

CAPÍTULO I

FUNDAMENTOS TEÓRICOS

CAPÍTULO II

PRUEBAS Y RESULTADOS

DISEÑO DEL REGISTRADOR DE VARIABLES ELÉCTRICAS

CAPÍTULO III

PRUEBAS Y RESULTADOS

RESUMEN

El presente proyecto se trata del diseño e implementación de software y hardware de un registrador de variables eléctricas el cual permite obtener información precisa del consumo de energía eléctrica de una vivienda. Además transmite los datos almacenados mediante comunicación Ethernet.

Las tarjetas utilizadas para el desarrollo del registrador son de tecnología ARDUINO las cuales se basan en la filosofía de software y hardware abierto; el hardware del registrador se lo realizó con las siguientes tarjetas: Arduino UNO SMD, Arduino Nano v3.0 y Arduino Ethernet Shield.

La tarjeta Arduino UNO SMD es la encargada de la adquisición de las señales analógicas proporcionadas por los sensores de corriente y del sensor de voltaje, la tarjeta Arduino Nano v3.0 es la que presenta las mediciones realizadas en una pantalla LCD 20x4; y, por último la tarjeta Arduino Ethernet Shield es la que transmite los datos procesados utilizando el protocolo Ethernet.

El registrador puede presentar la información almacenada de tres maneras: a) lectura de una tarjeta de memoria SD, b) descarga de información por un navegador de internet digitando la dirección IP asignada a la tarjeta Arduino; y, c) interfaz desarrollada en LabVIEW. El dispositivo diseñado almacena la información en tiempo real.

El desarrollo de este registrador de variables eléctricas permitirá en un futuro el estudio del comportamiento de la carga en viviendas rurales, lo cual facilitará el desarrollo de múltiples proyectos de generación eléctrica utilizando energías renovables.

ABSTRACT

This project is about the design of software and hardware of an electrical variables logger which allows to obtain precise information from electrical power consumption of a home. Also transmits data stored through Ethernet communication.

The boards used for the development logger are ARDUINO technology which are based on the philosophy of open hardware and software, the logger hardware was made with the following boards: Arduino UNO SMD, Arduino Nano v3.0 and Arduino Ethernet Shield.

Arduino UNO SMD board is in charge for the acquisition of the analog signals provided by the current sensors and voltage sensor, Arduino Nano v3.0 board shows the measurements made in a LCD screen 20x4, and finally Arduino Ethernet Shield board transmits processed data using the Ethernet protocol.

The logger can display the data stored in three ways: a) reading of a SD memory card, b) download of information by a web browser typing the IP address assigned to the Arduino board, and c) interface developed in LabVIEW, The designed device stores information in real time.

The development of this electrical variables logger will allow in the future perform a study the behavior of the load in rural households, which will facilitate the development of multiple power generation projects using renewable energy.

CAPÍTULO 1

FUNDAMENTOS TEÓRICOS

1.1. IMPORTANCIA DEL PROYECTO

El costo de los registradores de variables eléctricas es alto ya que poseen software y hardware propietario, es por ese motivo que este proyecto presenta un registrador de bajo costo ya que el software y el hardware de Arduino son de arquitectura abierta.

La implementación del registrador de variables eléctricas con transmisión de datos a través del protocolo Ethernet permite realizar un monitoreo de dichas variables en cualquier lugar con tan solo acceder a la IP asignada a la tarjeta Arduino.

En Ecuador el desarrollo de tecnología es escaso, por lo que es conveniente fomentar la investigación en nuevos dispositivos para realizar aplicaciones de uso común que permitan cubrir una necesidad en la sociedad ecuatoriana.

1.2. VARIABLES ELÉCTRICAS

1.2.1. DIFERENCIA DE POTENCIAL[1] (*v*)

La diferencia de potencial o tensión *v* entre dos puntos de un campo eléctrico es, por definición, el trabajo necesario para desplazar la unidad de carga eléctrica positiva de un punto al otro en contra o a favor de las fuerzas del campo. En el sistema MKS, la unidad de diferencia de

[1] EDMINISTER, JOSEPH E. “Circuitos Eléctricos” Pág. 2.

potencial es el voltio (V) y corresponde al trabajo de 1 julio (J) al desplazar 1 culombio (C) de carga de uno al otro punto, es decir, 1 V = 1 J/C.

Si entre dos puntos existe una diferencia de potencial *v* (d.d.p.), el trabajo necesario para desplazar una carga *q* será *qv* y la carga se moverá del punto de mayor al de menor potencial.

Un agente o dispositivo, tal como una batería o un generador, posee una fuerza electromotriz (f.e.m.) si es capaz de suministrar a una carga eléctrica la energía suficiente para hacerla circular por él, del terminal de menor al de mayor potencial. La f.e.m. se mide por la d.d.p. en bornes del generador cuando no suministra corriente eléctrica, es decir, en circuito abierto.

[2]Cuando separamos cargas positivas y negativas, tenemos que gastar energía en el proceso. La tensión es la energía por unidad de carga creada por la separación. Expresamos este cociente en forma diferencial como

$$v = \frac{dw}{dq} \qquad \text{Ec. 1.1}$$

Dónde:

v = tensión en voltios,

w = energía en julios,

q = carga en culombios.

[2] NILSSON, JAMES W.; RIEDEL, SUSAN A. “Circuitos Eléctricos” Pág. 14.

1.2.2. CORRIENTE ELÉCTRICA[3] (*i*)

Todo cuerpo con electrones libres capaces de moverse entre los átomos de la red cristalina del mismo se llama conductor. Una de las causas que originan este movimiento es la aplicación al conductor de una diferencia de potencial.

Cuando de un punto a otro de un conductor se desplaza una o más cargas eléctricas diremos que circula por él una corriente eléctrica. Si la carga se transfiere a una velocidad de 1 culombio por segundo (C/s) la corriente por el conductor tiene una intensidad de 1 amperio (A), es decir, 1 A = 1 C/s. En general, la intensidad de corriente instantánea *i* en un conductor es

$$i = \frac{dq}{dt} \qquad \text{Ec. 1.2}$$

Dónde:

i = corriente en amperios,

q = carga en culombios,

t = tiempo en segundos.

Por convenio, se ha establecido como sentido positivo de la intensidad de la corriente eléctrica el opuesto al del movimiento de los electrones (figura1.1).

[3] EDMINISTER, JOSEPH E. “Circuitos Eléctricos” Pág. 2.

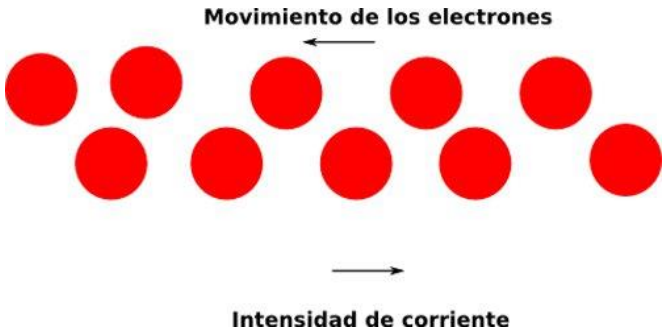


Figura 1.1: *Sentido de la corriente eléctrica.*

1.2.3. POTENCIA[4] (*p*)

La potencia eléctrica *p* se define por el producto de la diferencia de potencial o tensión aplicada *v* y la intensidad de corriente *i* a que da lugar. La unidad de potencia es el vatio (W), de manera que 1 W = 1 V*A. Matemáticamente se escribe:

$$i(W) = p(V) * i(A) \qquad \text{Ec. 1.3}$$

[5]La potencia es la tasa temporal de gasto o de absorción de energía, por lo tanto, la energía por unidad de tiempo se expresa en forma de derivada, de la manera siguiente:

$$p = \frac{dw}{dt} \qquad \text{Ec.1.4}$$

Dónde:

[4] EDMINISTER, JOSEPH E. “Circuitos Eléctricos” Pág. 3.
[5] NILSSON, JAMES W.; RIEDEL, SUSAN A. “Circuitos Eléctricos” Pág. 17.

p = potencia en vatios,

w = energía en julios,

t = tiempo en segundos.

El signo algebraico de la potencia se basa en el movimiento de las cargas a través de caídas e incrementos de tensión. A medida que las cargas se mueven a través de una caída de tensión, pierden energía, mientras que si se mueven a través de un incremento de tensión, ganan energía. La figura 1.2 resume la relación existente entre las referencias de polaridad para la tensión y la corriente y la ecuación que define la potencia.

Ahora podemos enunciar la regla para interpretar el signo algebraico de la potencia:

Si la potencia es positiva (es decir, si p > 0), se estará entregando potencia al circuito contenido dentro de recuadro. Si la potencia es negativa (es decir, si p < 0), se estará extrayendo potencia del circuito contenido en el recuadro.

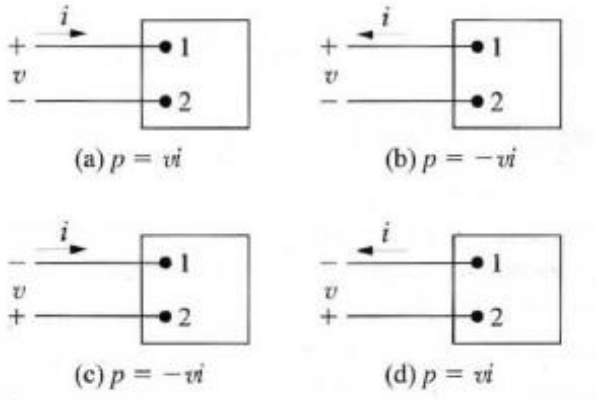


Figura 1.2: *Referencia de polaridad y expresión de la potencia.*

1.2.4. ENERGÍA[6]

Los cálculos de potencia y energía son también importantes en el análisis de circuitos. Una de las razones es que, aunque la tensión y la corriente son variables útiles en el análisis y diseño de sistemas eléctricos, la salida útil del sistema es a menudo de naturaleza no eléctrica, y esta salida puede expresarse convenientemente en términos de potencia o energía.

Como la potencia *p* es la variación de energía transferida en la unidad de tiempo, siendo W la energía total suministrada durante un intervalo de tiempo dado.

$$p = \frac{dw}{dt} \; de \; donde \; W = \int_{t1}^{t2} p \; dt \qquad \text{Ec. 1.5}$$

La unidad de energía es el julio: 1 J = 1 W*s.

1.3. ARDUINO[7]

Arduino es una plataforma de electrónica abierta para la creación de prototipos basados en software y hardware abierto. Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede interactuar con todo aquello que le rodea controlando luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectarse a un ordenador, si bien tienen la posibilidad de

[6] EDMINISTER, JOSEPH E. “Circuitos Eléctricos” Pág. 3.
[7] ARDUINO. Arduino. [Documento HTML]. http://www.arduino.cc/es.

hacerlo y comunicar con diferentes tipos de software por ejemplo: Flash, Processing, MaxMSP.

Las placas pueden ser hechas a mano o compradas montadas de fábrica; el software puede ser descargado de forma gratuita. Los ficheros de diseño de referencia (CAD) están disponibles bajo una licencia abierta.

1.4. ARDUINO UNO SMD[8]

1.4.1. INFORMACIÓN GENERAL

Arduino UNO es una placa electrónica basada en el microcontrolador ATmega 328. Cuenta con 14 pines digitales que pueden ser entradas o salidas de los cuales 6 pueden ser usados como salidas PWM, 6 entradas analógicas, un cristal oscilador de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP y un botón de reinicio. La placa electrónica tiene todo lo necesario para proporcionar soporte al microcontrolador, basta con conectar la placa a una computadora con un cable USB o a un adaptador AC-DC o a una batería para empezar.

La tarjeta Arduino UNO SMD difiere de todas las placas anteriores ya que no utiliza el chip FTDI USB serie. Por el contrario cuenta con el ATmega16U2 programado como un conversor de USB a serie.

“Uno” significa uno en italiano y se llama así por motivos de la próxima versión de Arduino 1.0. La tarjeta Uno y la versión 1.0 serán la versión de referencia de Arduino para seguir adelante. La tarjeta Uno es la última en una serie de placas USB de Arduino, y el modelo de referencia para la plataforma Arduino.

[8] ARDUINO. Arduino UNO. [Documento HTML]. http://arduino.cc/en/Main/ArduinoBoardUno.

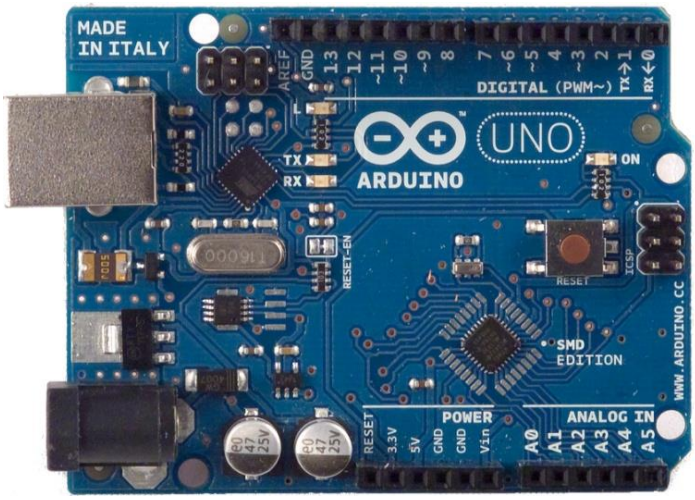


Figura 1.3: *Arduino UNO SMD.*

1.4.2. ESPECIFICACIONES

Tabla 1.1: *Especificaciones de la tarjeta Arduino UNO SMD.*

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	50mA
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

1.4.3. ALIMENTACIÓN

La tarjeta Arduino UNO puede ser alimentada a través de una conexión USB o con una fuente de alimentación externa. La fuente de alimentación es seleccionada automáticamente.

La fuente de alimentación externa (no USB) puede provenir de un adaptador AC-DC o una batería. El adaptador debe tener un conector de 2.1mm cuyo conector central debe ser positivo para alimentar a la placa adecuadamente. Si es a través de una batería, esta puede ser conectada en los pines Gnd y Vin con un conector de alimentación.

La placa puede operar con una fuente externa de 6 a 20 voltios. Si es alimentada con un voltaje menor a 7V el pin de 5V podría proporcionar menos de cinco voltios y además la placa puede ser inestable. Si se utiliza un voltaje de alimentación mayor a 12V, el regulador de voltaje se puede calentar y dañar la placa. Por lo tanto el rango recomendado es de 7 a 12 voltios.

Los pines de alimentación son los siguientes:

a) VIN: Es el pin del voltaje de entrada a la tarjeta Arduino cuando se utiliza una fuente de alimentación externa (opuesta a los 5 voltios de la conexión USB u otra fuente de alimentación regulada). Se puede suministrar tensión a través de este pin, o mediante el enchufe de alimentación de la placa.

b) 5V: Este pin genera cinco voltios regulados por el regulador de la placa. La placa puede ser alimentada con energía eléctrica ya sea a partir de la entrada de alimentación (7–12V), el conector USB (5V) o mediante el pin VIN de la placa (7-12V). El suministro de tensión a través de los pines de 5V o 3.3V no pasan por el regulador y pueden dañar la placa.

c) 3.3V: Este pin proporciona 3.3 voltios generados por el regulador de la placa. El consumo máximo de corriente es de 50 mA.

d) GND: Pines de tierra.

1.4.4. MEMORIA

El ATmega328 tiene 32 MB de memoria (0.5 KB son utilizados para el gestor de arranque). También dispone de 2 KB de SRAM y 1 KB de memoria EEPROM (que puede ser leído y escrito con la librería EEPROM).

1.4.5. ENTRADAS Y SALIDAS

Cada uno de los 14 pines digitales en la tarjeta Arduino Uno pueden ser utilizados como entradas o salidas, usando las funciones pinMode(), digitalWrite() y digitalRead(). Los 14 pines operan a 5 voltios. Cada pin puede proporcionar o recibir un máximo de 40 mA y tienen una resistencia pull-up interna (desconectado por defecto) de 20-50 kΩ. Además, algunos pines tienen funciones específicas:

a) Serial: 0 (RX) y 1 (TX). Estos pines son utilizados para recibir (RX) y transmitir (TX) datos serie TTL. Estos pines están conectados a los puntos correspondientes del chip serial ATmega8U2 USB-to-TTL.

b) Interrupciones externas: 2 y 3. Estos pines se pueden configurar para disparar una interrupción en un valor bajo, un borde ascendente o descendente o un cambio en el valor. Para realizar esto se utiliza la función attachInterrupt().

c) PWM: 3, 5, 6, 9, 10 y 11. Proporcionan 8 bits de salida PWM con la función analogWrite().

d) SPI: 10 (SS), 11(MOSI), 12 (MISO), 13 (SCK). Estos pines permiten la comunicación SPI utilizando la librería SPI.

e) LED: 13. Hay un LED incorporado conectado al pin digital 13. Cuando el pin esta en ALTO, el LED está encendido, cuando el pin esta en BAJO, se apaga el LED.

La tarjeta Arduino UNO tiene 6 entradas analógicas, etiquetadas desde A0 a A5, cada una de las cuales proporcionan 10 bits de resolución (es decir 1024 valores diferentes). Por defecto miden de 0 a 5 voltios, aunque es posible cambiar el extremo superior de su rango con el pin AREF y la función analogReference(). Además, algunos de estos pines tienen funciones específicas:

a) TWI: A4 o pin SDA y A5 o pin SCL. Soporta la comunicación I2C usando la librería Wire.

b) AREF. Referencia de voltaje para las entradas analógicas usando la función analogReference().

c) RESET: Provoca un valor BAJO en la línea que reinicia al microcontrolador. Normalmente es utilizado para añadir un botón de reinicio en la placa.

1.4.6. COMUNICACIONES

La tarjeta Arduino UNO tiene una serie de facilidades para comunicarse con un ordenador a diferencia de otras tarjetas Arduino u otros microcontroladores. El ATmega328 ofrece una comunicación serial UART

TTL (5V), la cual está disponible en los pines digitales 0 (RX) y 1 (TX). Un ATmega16U2 en la placa que proporciona los canales de comunicación serie a través de USB y aparece como un puerto virtual COM con el software en el ordenador. El firmware 16U2 utiliza el estándar de los controladores USB COM, por lo que no son necesarios controladores externos. Sin embargo, en Windows, se requiere un archivo con el sufijo ".inf". El software de Arduino incluye un monitor serie que permite visualizar los datos de texto enviados desde la tarjeta Arduino. Los leds RX y TX en la placa parpadean cuando se están transmitiendo datos a través del chip USB-to-serial y la conexión USB a la computadora (pero no parpadean para la comunicación serie de los pines 0 y 1).

La librería SoftwareSerial permite la comunicación serie en cualquiera de los pines digitales de la tarjeta Arduino UNO.

Además el ATmega328 también soporta la comunicación I2C y SPI. El software de Arduino incluye una librería Wire para simplificar el uso del bus I2C. Para la comunicación SPI, se debe utilizar la librería SPI.

1.4.7. PROGRAMACIÓN

La tarjeta Arduino UNO puede ser programado con el software de Arduino. Seleccione "Arduino UNO del menú Herramientas> Placas".

El ATmega328 en la tarjeta Arduino UNO viene pre-grabado con un gestor de arranque que le permite cargar un nuevo código sin el uso de un programador de hardware externo. Se comunica utilizando el protocolo original STK500 (referencia, archivos de cabecera C).

Además se puede pasar por alto el gestor de arranque y programar el microcontrolador a través d la cabecera ICSP (In Circuit Serial Programming).

El código fuente del firmware del ATmega16U2 (o 8U2 en las placas REV1 y REV2) está disponible. El ATmega16U2/8U2 está cargado con un gestor de arranque DFU, que puede ser activado por:

a) En las placas Rev1: Conectando un puente de soldadura en la parte posterior de la placa (cerca del mapa de Italia) y luego reiniciar el 8U2.

b) En las placas Rev2 y posteriores: Hay una resistencia pulling en la línea a tierra del 8U2/16U2 HWB, por lo que es más fácil poner en modo DFU.

También se puede utilizar el software FLIP de Atmel (Windows) o el programador DFU (Mac OS X y Linux) para cargar un nuevo firmware. O puede utilizarse la cabecera ISP con un programador externo para sobrescribir el gestor de arranque DFU.

1.4.8. REINICIO AUTOMÁTICO (SOFTWARE)

En lugar de requerir un pulso físico del botón de reinicio antes de que se cargue el sketch en la placa, la tarjeta Arduino UNO está diseñada de tal manera que puede ser reiniciada mediante el software que se ejecuta cuando está conectada a un computador. Una de las líneas de control de flujo de hardware (DTR) del ATmega8U2/16U2 está conectada a la línea de reinicio del ATmega328 a través de un condensador de 100 nano faradios. Cuando se asegura esta línea, la línea de reinicio cae lo suficiente como para restablecer el chip. El software de Arduino utiliza esta capacidad de la tarjeta para cargar el código con sólo pulsar el botón “UPLOAD” en el entorno de Arduino. Esto significa que el gestor de arranque puede tener un tiempo de espera más corta, como la reducción de DTR puede ser bien coordinada con el inicio de la carga.

Esta configuración tiene otras implicaciones. Cuando la tarjeta Arduino UNO se conecta a un ordenador con Mac OS X o Linux, se pone a cero cada vez que se realiza una conexión a ella desde el software (a través de USB). Para el siguiente medio segundo más o menos, el gestor de arranque se está ejecutando en la tarjeta Arduino UNO. La tarjeta está programada para ignorar los datos con formato incorrecto interceptando los primeros bytes de datos enviados a la placa después de que una conexión se abre. Si un sketch se está ejecutando en la placa recibe un tiempo de configuración u otros datos cuando se inicia por primera vez, asegúrese de que el software con el que se comunica espere un segundo después de abrir la conexión y antes de enviar estos datos.

La tarjeta Arduino UNO contiene una traza que se puede cortar para deshabilitar el reinicio automático. Las almohadillas en cada lado de la traza pueden ser soldadas entre sí para volver a habilitar la misma. Esto está marcado como "RESET-EN". También se puede deshabilitar el reinicio automático mediante la conexión de una resistencia de 110 ohmios alimentada con 5V a la línea de reinicio.

1.4.9. PROTECCIÓN USB CONTRA SOBRE CORRIENTE

La tarjeta Arduino UNO tiene un poli fusible reajustable que protege a los puertos USB del computador de cortos y de sobre corriente. Aunque la mayoría de las computadoras ofrecen una protección interna, el fusible proporciona una capa adicional de protección. Si hay más de 500 mA en el puerto USB, el fusible automáticamente corta la conexión hasta que el cortocircuito o sobrecarga sea eliminado.

1.4.10. CARACTERÍSTICAS FÍSICAS

El largo y ancho de la PBC de la tarjeta Arduino UNO son 2.7x2.1 pulgadas respectivamente, con el conector USB y conector de alimentación se extiende más allá de la dimensión anterior. Cuatro orificios de tornillos permiten que la tarjeta sea sujetada a una superficie o caja. Obsérvese que la distancia entre los pines digitales 7 y 8 es de 160 milésimas de pulgada (0.16"), no un múltiplo par de la separación de 100 milésimas de los otros pines.

1.5. ARDUINO NANO[9]

1.5.1. INFORMACIÓN GENERAL

La tarjeta Arduino Nano es una placa pequeña, completa basada en el ATmega328 (Arduino Nano 3.0) o ATmega168 (Arduino Nano 2.x). Tiene más o menos la misma funcionalidad de la tarjeta Arduino Duemilanove, pero en un paquete diferente. Le falta sólo un conector de alimentación DC y funciona con un cable USB tipo mini B en lugar de uno estándar.

La tarjeta Arduino Nano fue diseñada y está siendo producida por Gravitech.

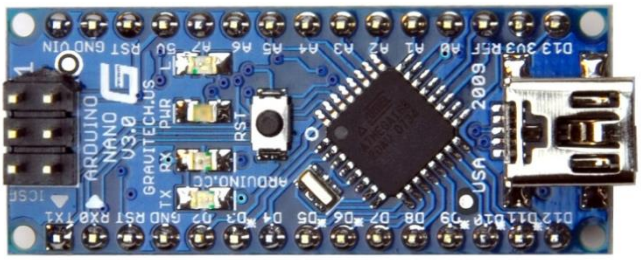


Figura 1.4: *Arduino Nano.*

[9] ARDUINO. Arduino Nano. [Documento HTML]. http://arduino.cc/en/Main/ArduinoBoardNano.

1.5.2. ESPECIFICACIONES

Tabla 1.2: *Especificaciones de la tarjeta Arduino Nano.*

Microcontroller	Atmel ATmega168 or ATmega328
Operating Voltage (logic level)	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	8
DC Current per I/O Pin	40 mA
Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)
Clock Speed	16 MHz

1.5.3. ALIMENTACIÓN

La tarjeta Arduino Nano puede ser alimentada a través de la conexión USB tipo mini B, una fuente de alimentación externa no regulada de 6-20V (pin 30), o una fuente de alimentación externa regulada de 5V (pin 27). La fuente de alimentación se selecciona automáticamente por el voltaje más alto de la fuente.

El chip FTDI FT232RL en el Nano sólo se enciende si la tarjeta está siendo alimentada a través de USB. Como resultado, cuando está

funcionando con una alimentación externa (no USB), la salida de 3.3V (la cual es soportada por el chip FTDI) no está disponible y los leds RX y TX parpadearán si los pines digitales 0 o 1 están en alto.

1.5.4. MEMORIA

El ATmega168 tiene 16 KB de memoria flash para almacenar el código (de los cuales 2 KB se utiliza para el gestor de arranque), el ATmega328 tiene 32 KB, (también con 2 KB utilizados para el gestor de arranque). El ATmega168 tiene 1 KB de SRAM y 512 bytes de EEPROM (puede ser leído y escrito con la librería EEPROM), el ATmega328 tiene 2 KB de SRAM y 1 KB de memoria EEPROM.

1.5.5. ENTRADAS Y SALIDAS

Cada uno de los 14 pines digitales de la tarjeta Arduino Nano puede ser utilizado como una entrada o salida, usando las funciones pinMode(), digitalWrite(), y digitalRead(). Los pines operan a 5 voltios. Cada pin puede proporcionar o recibir una corriente máxima de 40 mA y tienen una resistencia pull-up interna (desconectado por defecto) de 20 a 50 kΩ. Además, algunos pines tienen funciones específicas:

a) Serie: 0 (RX) y 1 (TX). Se utiliza para recibir (RX) y transmitir (TX) datos serie TTL. Estos pines están conectados a los pines correspondientes del chip FTDI USB-to-TTL.

b) Interrupciones externas: 2 y 3. Estos pines pueden ser configurados para disparar una interrupción en un valor bajo, un borde ascendente o descendente o un cambio en el valor. Para realizar esto se utiliza la función attachInterrupt().

c) PWM: 3, 5, 6, 9, 10 y 11. Proporcionar 8 bits de salida PWM con la función analogWrite().

d) SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Estos pines soportan la comunicación SPI, que, a pesar de ser proporcionado por el hardware fundamental, actualmente no está incluido en el lenguaje de Arduino.

e) LED: 13. Hay un LED incorporado conectado al pin digital 13. Cuando el pin esta en ALTO, el LED está encendido, cuando el pin esta en BAJO, se apaga el LED.

La tarjeta Arduino tiene 8 entradas analógicas, cada una de las que proporcionan 10 bits de resolución (es decir 1024 valores diferentes). Por defecto miden desde 0 a 5 voltios, aunque es posible cambiar el extremo superior de su rango utilizando la función analogReference(). Además, algunos de estos pines tienen funciones específicas:

a) I2C: 4 (SDA) y 5 (SCL). Soportan la comunicación I2C (TWI) utilizando la librería Wire.

b) AREF: Voltaje de referencia para las entradas analógicas. Se utiliza la función analogReference().

c) RESET: Provoca un valor BAJO en la línea que reinicia al microcontrolador. Normalmente se utiliza para añadir un botón de reinicio en la placa.

1.5.6. COMUNICACIONES

La tarjeta Arduino Nano tiene una serie de facilidades para comunicarse con un ordenador a diferencia de otras tarjetas Arduino u otros

microcontroladores. El ATmega168 y ATmega328 ofrecen una comunicación serie UART TTL (5V), que está disponible en los pines digitales 0 (RX) y 1 (TX). Un FT232RL FTDI en la placa proporciona los canales de comunicación serie a través de USB y los controladores de FTDI (incluido con el software de Arduino) proporcionan un puerto COM virtual para el software del ordenador. El software de Arduino incluye un monitor serie que permite visualizar los datos de texto que se envía desde y hacia la placa Arduino. Los leds RX y TX en la placa parpadean cuando se están transmitiendo datos a través del chip FTDI y de la conexión USB a la computadora (pero no parpadean para la comunicación serie de los pines 0 y 1).

La biblioteca SoftwareSerial permite la comunicación serie en cualquiera de los pines digitales de la tarjeta Arduino Nano.

El ATmega168 y ATmega328 también soportan I2C (TWI) y la comunicación SPI. El software de Arduino incluye una librería Wire para simplificar el uso del bus I2C.

1.5.7. PROGRAMACIÓN

La tarjeta Arduino Nano puede ser programada con el software de Arduino. Seleccione en el menú Herramientas > Placas "Arduino Diecimila, Duemilanove o Nano w/ATmega168" o "Arduino Duemilanove Nano o w/ATmega328" (de acuerdo con el microcontrolador de la placa).

El ATmega168 o ATmega328 en la Arduino Nano viene pre-grabado con un gestor de arranque que le permite cargar nuevo código a la misma sin el uso de un programador de hardware externo. Se comunica utilizando el protocolo original STK500 (de referencia, archivos de cabecera C).

También se puede pasar por alto el gestor de arranque y programar el microcontrolador a través de la cabecera ICSP.

1.5.8. REINICIO AUTOMÁTICO (SOFTWARE)

En lugar de requerir un pulso físico del botón de reinicio antes de que se cargue el sketch en la placa, La tarjeta Arduino Nano está diseñada de una manera que le permite ser reiniciada mediante el software que se ejecuta cuando está conectada a un ordenador. Una de las líneas de control de flujo de hardware (DTR) del FT232RL está conectado a la línea de reinicio del ATmega168 o ATmega328 a través de un condensador de 100 nano faradios. Cuando se asegura esta línea, la línea de reinicio cae lo suficiente como para restablecer el chip. El software de Arduino utiliza esta capacidad de la tarjeta para cargar el código con sólo pulsar el botón "UPLOAD" en el entorno de Arduino. Esto significa que el gestor de arranque puede tener un tiempo de espera más corto, como la reducción de DTR puede ser bien coordinada con el inicio de la carga.

Esta configuración tiene otras implicaciones. Cuando la tarjeta Arduino Nano se conecta a un ordenador con Mac OS X o Linux, se pone a cero cada vez que se realiza una conexión a ella desde el software (a través de USB). Para el siguiente medio segundo más o menos, el gestor de arranque se ejecuta en la tarjeta Arduino Nano. La tarjeta está programada para ignorar los datos con formato incorrecto interceptando los primeros bytes de datos enviados a la placa después de que una conexión se abre. Si un sketch se está ejecutando en la placa recibe una configuración en tiempo u otros datos cuando se inicia por primera vez, asegúrese de que el software con el que se comunica espere un segundo después de abrir la conexión y antes de enviar estos datos.

1.6. ARDUINO ETHERNET SHIELD[10]

[10] ARDUINO. Arduino Ethernet Shield. [Documento HTML]. http://arduino.cc/en/Main/ArduinoEthernetShield.

1.6.1. INFORMACIÓN GENERAL

La tarjeta Arduino Ethernet Shield conecta a una tarjeta Arduino a internet en cuestión de minutos. Sólo hay que conectar este módulo sobre la placa Arduino, conectarlo a la red con un cable RJ45 (no incluido) y seguir algunas instrucciones sencillas para empezar a controlar su entorno a través de internet. Como siempre con Arduino, todos los elementos de la plataforma de hardware, software y documentación son de libre acceso y de código abierto. Esto significa que se puede aprender exactamente cómo se hacen y utilizar su diseño como punto de partida para realizar aplicaciones específicas.

a) Requiere una tarjeta Arduino.
b) Tensión de 5 V (suministrados desde la tarjeta Arduino).
c) Controlador de Ethernet: W5100 con buffer interno de 16K.
d) Velocidad de conexión: 10/100Mb.
e) Conexión con Arduino mediante el puerto SPI.

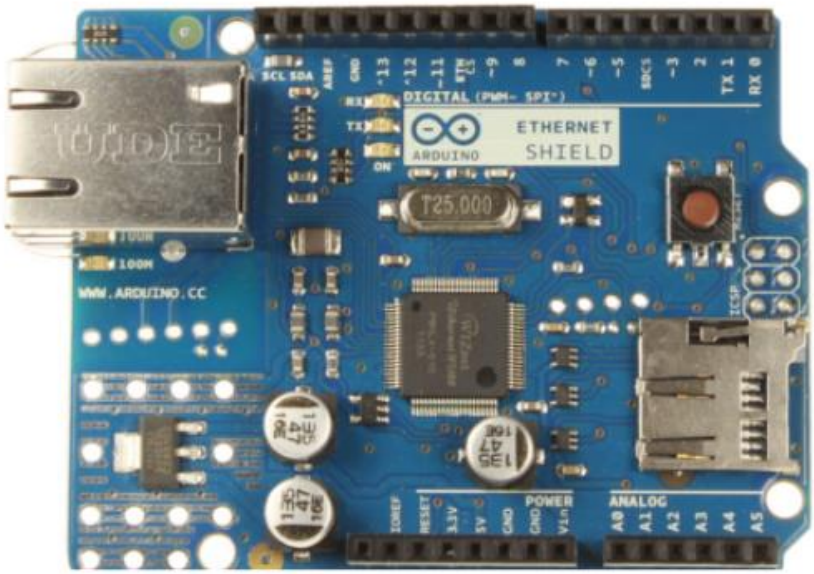


Figura 1.5: *Arduino Ethernet Shield.*

1.6.2. DESCRIPCIÓN

La tarjeta Arduino Ethernet Shield permite a una placa Arduino conectarse a internet. Se basa en el chip para Ethernet Wiznet W5100. El Wiznet W5100 proporciona una red (IP) capaz de contener TCP y UDP. Soporta hasta cuatro conexiones por sockets simultáneas. Utiliza la librería de Ethernet para escribir sketches con conexión a internet usando la tarjeta Arduino Ethernet Shield. El Ethernet Shield se conecta a una placa Arduino con pines largos que se encuentran a través de toda la placa. Esto mantiene intacta la disposición de pines y permite que otro shield pueda ser conectado en la parte superior.

La modificación más reciente de la tarjeta presenta el pin de salida 1.0 sobre la tarjeta Arduino UNO Rev3.

El Ethernet Shield tiene un estándar de conexión RJ-45, con un transformador de línea integrado y alimentación por Ethernet habilitada.

Posee una ranura para tarjetas de memoria micro SD, que puede ser usada para almacenar archivos para proporcionarlos a través de la red. Es compatible con las tarjetas Arduino Uno y Mega (usando la librería Ethernet). Al lector de tarjetas de memoria micro SD a bordo se puede acceder a través de la librería SD. Cuando se trabaja con esta biblioteca, SS es el pin 4. La versión original de la tarjeta Arduino Ethernet Shield tiene una ranura para tarjetas SD de tamaño completo, lo que no es compatible.

El shield también incluye un controlador de reinicio, para asegurarse de que el módulo Ethernet W5100 se reinicie correctamente en el encendido. Las revisiones anteriores del shield no eran compatibles con la tarjeta Arduino Mega y necesitan reiniciarse manualmente después del encendido.

El escudo actual tiene una alimentación a través de Ethernet (PoE), módulo diseñado para extraer alimentación de un cable convencional de par trenzado para Ethernet Categoría 5:

a) Compatible con IEEE802.3af.

b) Bajo rizado de salida y ruido (100mVpp).

c) Rango de voltaje de entrada de 36V a 57V.

d) Protección contra sobrecarga y cortocircuitos.

e) Salida de 9V

f) Alta eficiencia en el convertidor DC/DC: typ 75% a 50% de carga

g) 1500V de aislamiento (entrada a salida)

La alimentación del módulo a través de Ethernet es de hardware propietario, no es hecho por Arduino, es un accesorio de terceros.

El escudo no viene con el módulo PoE (Power over Ethernet) incorporado, es un componente separado que debe ser añadido.

Arduino se comunica tanto con el W5100 y la tarjeta de memoria SD utilizando el bus SPI (a través de la cabecera ICSP). Esto es en los pines digitales 11, 12, y 13 en el Duemilanove y los pines 50, 51, y 52 en la tarjeta Arduino Mega. En ambas tarjetas, el pin 10 se utiliza para seleccionar el W5100 y el pin 4 para la tarjeta SD. Estos pines no se pueden utilizar para entradas y salidas. En la tarjeta Arduino Mega, el pin SS, 53, no se utiliza para seleccionar el W5100 o la tarjeta de memoria SD, pero debe mantenerse como una salida o la interfaz SPI no funcionará.

Nótese que debido a que el W5100 y la tarjeta de memoria SD comparten el bus SPI, sólo uno puede estar activo a la vez. Si está utilizando ambos periféricos en su programa, estos deben ser atendidos por las librerías correspondientes. Si usted no está utilizando uno de los periféricos en su programa, tendrá que de forma explícita anular su selección. Para hacer esto con la tarjeta de memoria SD, ajuste el pin 4 como salida y escriba en alto a la misma. Para el W5100, ajuste el pin 10 como salida y escriba en alta a la misma.

El shield provee un conector estándar de Ethernet RJ45.

El botón de reinicio sobre el shield reinicia tanto el W5100 y la placa Arduino.

El shield contiene los siguientes leds informativos:

a) PWR: indica que la tarjeta y el shield están encendidos.

b) LINK: indica la presencia de un enlace de red y parpadea cuando el shield transmite o recibe datos.

c) FULLD: indica que la conexión de red es full dúplex.

d) 100M: indica la presencia de una conexión de red 100 Mb/s (en lugar de 10 Mb/s).

e) RX: parpadea cuando el shield recibe datos.

f) TX: parpadea cuando el shield envía datos.

g) COLL: parpadea cuando se detectan colisiones de red.

El jumper soldado en la marca "INT" puede conectarse para permitir que la tarjeta Arduino reciba interrupciones por notificación de eventos desde

el W5100, pero esto no es soportado por la librería Ethernet. El jumper conecta el pin INT del W5100 al pin digital 2 de la Arduino.

1.7. RELOJ EN TIEMPO REAL (RTC)[11]

RTC es acrónimo de Real Time Clock o reloj en tiempo real, el mismo que se ha constituido en una herramienta de mucha utilidad para sistemas que usan aplicaciones de temporización industrial o en sistemas en los cuales se debe detectar eventos y la hora en la que ocurrieron. En la figura 1.6 se muestra físicamente al temporizador RTC.

Figura 1.6: *Temporizador RTC (DS1307)*

El DS1307 Real-Time-Clock Serie, es un dispositivo de bajo consumo de energía, completo con código binario decimal (BCD), reloj/calendario más 56 bytes de NVRAM, las direcciones y datos son transferidos a través de 2 hilos serie, bus bidireccional. El reloj/calendario provee información de segundos, minutos, horas, día, fecha, mes y año. El final de fecha de mes se ajusta automáticamente durante meses menores de 31 días, incluyendo correcciones para el año bisiesto. El reloj funciona en cualquier formato de 24 horas o en 12 horas con indicador AM/PM.

[11] HISPAVILA. DS1307. [Documento PDF]. http://www.hispavila.com/3ds/atmega/descargas/rtc_ds1307.pdf.

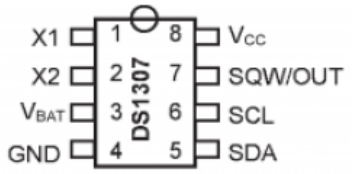


Figura 1.7: *Distribución de Pines (DS1307).*

El DS1307 tiene incorporado un circuito de sensor de tensión que detecta fallas de energía y cambia automáticamente al suministro de batería de respaldo.

Algunas de sus características principales son:

- El DS1307 es un RTC serial que procesa la información de los segundos, minutos, horas, día del mes, día de la semana, mes y año.

- El ajuste para la duración de los meses, incluso en los años bisiestos, es realizado por el propio circuito y es válido hasta el año 2100.

- Contiene 56 bytes de NVRAM (memoria RAM no volátil) para almacenamiento de datos.

- Permite operación en modo de 24 y 12 horas.

- Al presentar fallos en la alimentación (Vcc) el DS1307 cambia automáticamente al modo de operación con batería para no perder su configuración. Su consumo en esta condición, es menor de 500nA.

- Puede generar una señal de onda cuadrada de frecuencia programable.

- Rango de temperatura industrial opcional: -40 ^{o}C a +85 ^{o}C.
- Interface serie I2C.
- Detector automático de fallo de energía y circuito de conmutación.
- El pin 7 es una salida de colector abierto, que oscila a 1Hz. Esto permite la colocación de un led como indicador de segundos en aplicaciones de reloj.
- Los pines X1 y X2 son utilizados para la conexión de un cristal de cuarzo estándar de 32.768 kHz.

1.8. PANTALLA DE CRISTAL LIQUIDO (LCD)[12]

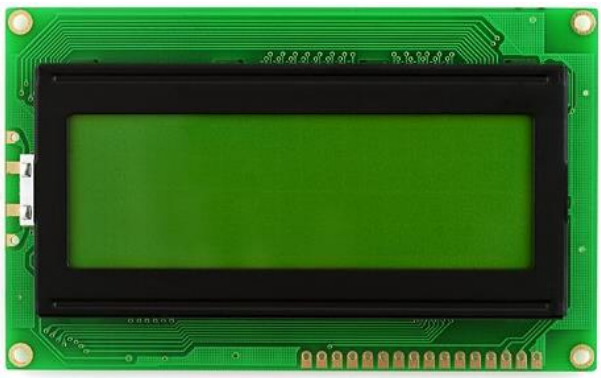


Figura 1.8: *Pantalla de Cristal Líquido – LCD 20x4*

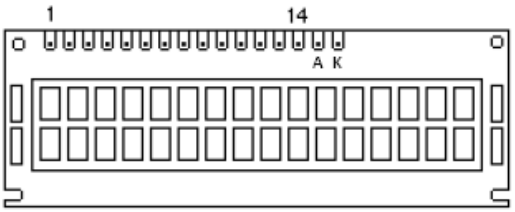
Una pantalla de cristal líquido permite visualizar ciertos caracteres. Para funcionar correctamente una pantalla LCD debe de estar conectada a un circuito impreso en el que estén integrados los controladores del display y

[12] ANGULO U., JOSÉ M.; ANGULO M., IGNACIO "Microcontroladores PIC Diseño práctico de aplicaciones" Pág. 261.

los pines para la conexión del display. Sobre el circuito impreso se encuentra el LCD en sí, rodeado por una estructura metálica que lo protege (Figura 1.8). En total se pueden visualizar 4 líneas de 20 caracteres cada una, es decir, 4x20=80 caracteres.

Tiene un consumo de energía menor a 5mA y son ideales para dispositivos que requieran una visualización pequeña o media. En la tabla 1.3 se muestra la asignación de pines en una pantalla LCD 20x4.

Tabla 1.3: *Descripción de pines del LCD*



Pin No	Name	Description
1	Vss	GND
2	Vdd	+5v
3	Vo	Contrast Control
4	RS	Register Select
5	R/W	Read/Write
6	E	Enable *(Strobe)*
7	D0	Data *LSB*
8	D1	Data
9	D2	Data
10	D3	Data
11	D4	Data
12	D5	Data
13	D6	Data
14	D7	Data *MSB*

- Los pines 1 y 2, son utilizados para la alimentación del módulo LCD. La tensión utilizada es de 5 voltios.

- El pin 3 se utiliza para ajustar el contraste de la pantalla LCD. Por medio de un potenciómetro se regula la intensidad de los

caracteres, a mayor tensión mayor intensidad. Se suele utilizar un potenciómetro de unos 10 o 20 kΩ, que regulara la misma tensión que se utiliza para la alimentación.

- El pin 4 se utiliza para indicar al bus de datos si la información que le llega es una instrucción o por el contrario es un carácter. Si RS=0 indicara que en el bus de datos hay presente una instrucción, y si RS=1, indicará que tiene un carácter alfanumérico.

- El pin 5 es el de escritura o lectura. Si está a 0 el módulo escribe en la pantalla el dato que haya en el bus de datos, y si está a 1 leeremos lo que hay en el bus de datos.

- El pin 6 es el encargado para que el módulo LCD funcione correctamente, o por el contrario no acepte órdenes de funcionamiento. Cuando E=0 no se podrá utilizar el display y cuando E=1 se podrá transferir datos y realizar las demás operaciones.

- Los pines del 7 al 14 son los del bus de datos.

1.9. BUS I2C[13]

El acrónimo I2C o I^2C es un bus de comunicaciones serie, fue diseñado por Philips al inicio de la década de 1980. Su nombre viene de Inter Integrated Circuit. Este bus tiene una velocidad de transmisión de 100Kbits por segundo en el modo estándar, también permite velocidades de 3.4 Mbit/s. Es un bus usado en la industria, principalmente para

[13] FERNÁNDEZ C., ANTONIO M.: Bus I2C. [Documento PDF]. http://www.uco.es/~el1mofer/Docs/IntPerif/Bus%20I2C.pdf.

comunicar microcontroladores y sus periféricos en sistemas integrados y generalizando más para poder comunicar circuitos integrados entre sí, que normalmente residen en un mismo circuito impreso.

La principal característica de I^2C es que utiliza dos líneas para transmitir la información: una para los datos y por otra la señal de reloj. También es necesaria una tercera línea, pero esta sólo es la referencia. Como suelen comunicarse circuitos en una misma placa que comparten una misma tierra esta tercera línea no suele ser necesaria. En la figura 1.9 se muestra un esquema de conexión del bus I^2C.

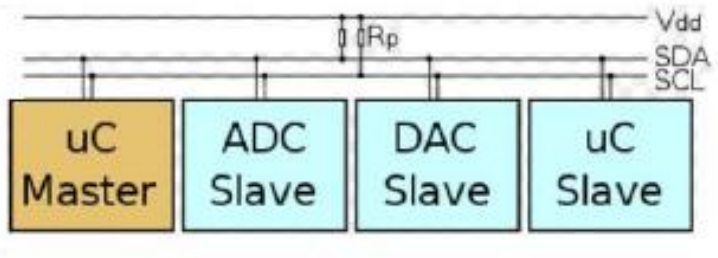


Figura 1.9: *Bus I2C.*

Las líneas SDA y SCL son de drenado abierto, por lo que necesitan resistencias de pull-up.

Los dispositivos conectados al bus I^2C tienen una dirección única para cada uno. También pueden ser maestros o esclavos. El dispositivo maestro inicia la transferencia de datos y además genera la señal de reloj.

1.9.1. PROTOCOLO DEL BUS I2C

El bus I2C sólo define dos señales, además del común:

- SDA: Es la línea de datos serie (Serial Data, en inglés), semibidireccional. Eléctricamente se trata de una señal a colector o drenado abierto. Es gobernada por el emisor, sea este un maestro o un esclavo.

- SCL: Es la señal de sincronía (reloj serie, o Serial Clock en inglés). Eléctricamente se trata de una señal a colector o drenado abierto. En un esclavo se trata de una entrada, mientras que en un maestro es una salida. Un maestro, además de generar la señal de sincronía suele tener la capacidad de evaluar su estado; el motivo es poder implementar un mecanismo de adaptación de velocidades. Esta señal es gobernada única y exclusivamente por el maestro; un esclavo sólo puede retenerla para forzar al maestro a ralentizar su funcionamiento.

Esta particularidad física de que las salidas de los excitadores I2C hayan de ser a colector o drenador abierto no es casual sino que resulta de vital importancia para que a este bus con tan sólo una señal de datos y otra de sincronía se le pueda dotar de todas sus características funcionales. Recuerde que un dispositivo lógico con este tipo de salida permite realizar la función producto lógico con una simple conexión eléctrica. Evidentemente, un enlace I2C necesitará sendas resistencias de elevación en las respectivas líneas SDA y SCL. De esta manera un excitador I2C realmente sólo gobierna el estado 0 lógico en las líneas I2C, mientras que el estado 1 lógico no es suministrado por el excitador directamente sino por medio de la oportuna resistencia de elevación. Así, el concepto de aislamiento del bus se consigue con tal sólo hacer que el excitador del nodo que se desea aislar ofrezca a su salida el estado alto (1 lógico), al que le corresponde una elevadísima impedancia de salida, equivalente a un aislamiento eléctrico con respecto al bus.

1.10. BUS SPI[14]

Es un estándar establecido por Motorola que utiliza un bus de 4 líneas para interconectar dispositivos periféricos de baja y media velocidad. La comunicación se realiza siguiendo un modelo maestro/esclavo donde el maestro selecciona al esclavo y comienza el proceso de transmisión/recepción de información. SPI constituye un bus full duplex, es decir, que se puede enviar y recibir información de manera simultánea, lo cual, eleva la tasa de transferencia de los datos. En este popular estándar no existe ningún medio de direccionamiento de los dispositivos esclavos y mucho menos de reconocimiento (Acknowledgement) de la recepción de los datos, así que, no existe forma alguna para que el dispositivo maestro pueda detactar la presencia de un esclavo o establecer un medio de control del flujo de datos. La figura 1.10 indica la conexión típica de un grupo de dispositivos en un bus SPI.

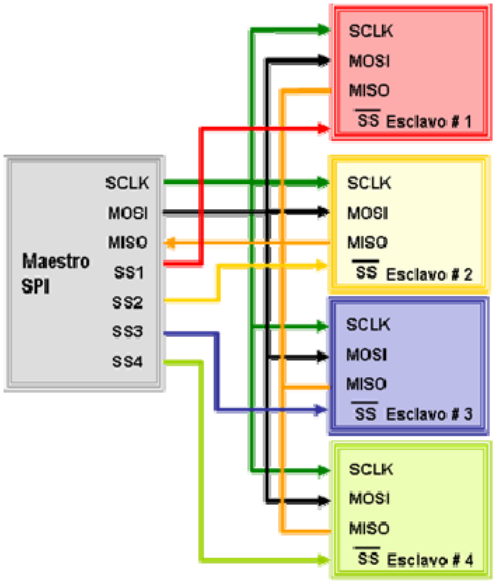


Figura 1.10: *Conexión de varios dispositivos SPI.*

[14] SISTEMA DE DESARROLLO PARA MC, MODELO MDK-B28. Comunicación Serial Sincrónica (SPI). [Documento PDF]. http://proyecto-test-hm1.googlecode.com/files/Cap_No_05.pdf.

Los nombres de las señales son definidos dentro del estándar como:

a) SCLK reloj del bus,
b) MOSI (Master Output Slave Input) salida de datos del maestro y entrada de datos a los esclavos,
c) MISO (Master Input Slave Output) salida de datos de los esclavos y entrada de datos al maestro; y,
d) SS (Slave Select) Habilitación del esclavo por parte del maestro.

Dependiendo del número de esclavos presentes en el bus, se requerirán más o menos líneas de selección para estos, cada una de ellas será nombrada como SS1, SS2, SS3, etc. Este medio de selección permite conectar varios esclavos de manera simple.

A continuación podemos ver cómo se realiza la entrada y la salida de datos en el modelo SPI:

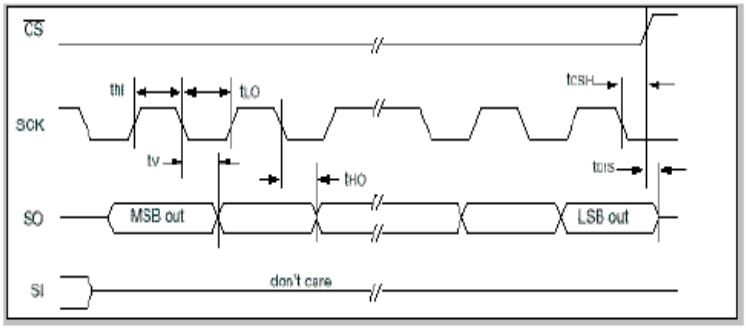


Figura 1.11: En*trada de datos SPI.*

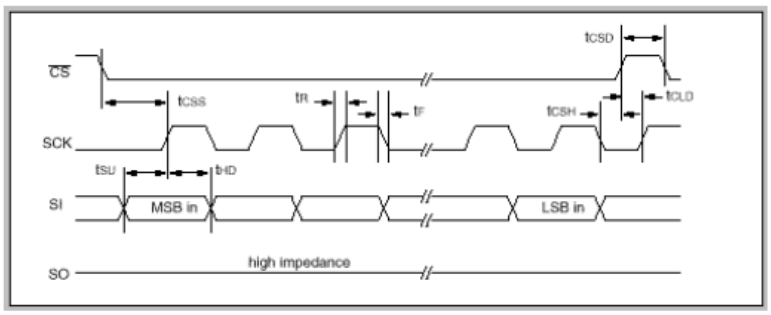


Figura 1.12: *Salida de datos SPI.*

1.10.1. VENTAJAS

- Comunicación Full Duplex.
- Mayor velocidad de transmisión que con I^2C o SMBus.
- Protocolo flexible en que se puede tener control absoluto sobre los bits transmitidos o no está limitado a la transferencia de bloques de 8 bits o elección del tamaño de la trama de bits, de su significado y propósito.
- Su implementación en hardware es extremadamente simple y consume menos energía que I^2C.
- Usa muchos menos terminales en cada chip/conector que una interfaz paralelo equivalente.
- Como mucho una única señal específica para cada cliente, las demás señales pueden ser compartidas.

1.10.2. DESVENTAJAS

- Utiliza más pines de cada chip que I^2C, incluso en la variante de 3 hilos.
- El direccionamiento se hace mediante líneas específicas.
- No hay control de flujo por hardware.
- No hay señal de aislamiento.

- El servidor podría estar enviando información sin que estuviese conectado a ningún cliente y no se daría cuenta de nada.

- No permite fácilmente tener varios servidores conectados al bus.

- Sólo funciona en distancias cortas.

1.11. USB[15]

El Universal Serial Bus, es un puerto que sirve para conectar periféricos a un ordenador. Fue creado en 1996 por siete empresas: IBM, Intel, Northern Telecom, Compaq, Microsoft, Digital Equipment Corporation y NEC. Un sistema USB tiene un diseño asimétrico, que consiste en un solo servidor y múltiples dispositivos conectados en una estructura de árbol utilizando concentradores especiales. Se pueden conectar hasta 127 dispositivos a un solo servidor, pero la suma debe incluir a los concentradores también, así que el total de dispositivos realmente usables es algo menor.

El diseño del USB tenía en mente eliminar la necesidad de adquirir tarjetas separadas para poner en los puertos de bus USA o PCI, y mejorar las capacidades de plug-and-play permitiendo a esos dispositivos ser conectados o desconectados al sistema sin necesidad de reiniciar. Sin embargo, en aplicaciones donde se necesita ancho de banda para grandes transferencias de datos, o si se necesita una latencia baja, los buses PCI o PCI express salen ganando. Igualmente sucede si la aplicación requiere de robustez industrial. A favor del bus USB, cabe decir que cuando se conecta un nuevo dispositivo, el servidor lo enumera y

[15] ESCUDERO G.: USB. [Documento PDF].
http://www.inti.gov.ar/electronicaeinformatica/informatica/pdf/usb.pdf.

agrega el software necesario para que pueda funcionar (esto depende del sistema operativo que esté usando).

Algunos dispositivos requieren una potencia mínima, así que se pueden conectar varios sin necesitar fuentes de alimentación extra. La gran mayoría de los concentradores incluyen fuentes de alimentación que brindan energía a los dispositivos conectados a ellos, pero algunos dispositivos consumen tanta energía que necesitan su propia fuente de alimentación. Los concentradores con fuente de alimentación pueden proporcionarle corriente eléctrica a otros dispositivos sin quitarle corriente al resto de la conexión (dentro de ciertos límites).

1.11.1. VELOCIDAD DE TRANSMISIÓN

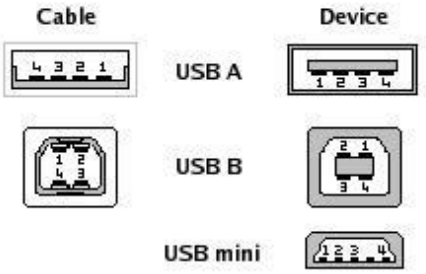
Según la velocidad se clasifican en cuatro tipos:

- Baja velocidad (1.0): Tasa de transferencia de hasta 1.5 Mbps (192 KB/s). Utilizado en su mayor parte por dispositivos de interfaz humana como los teclados, los ratones, las cámaras web, etc.

- Velocidad completa (1.1): Tasa de transferencia de hasta 12 Mbps (1.5 MB/s). Estos dispositivos dividen el ancho de banda de la conexión USB entre ellos, basados en un algoritmo de impedancias LIFO.

- Alta velocidad (2.0): Tasa de transferencia de hasta 480 Mbps (60 MB/s). Está presente casi en el 99% de los PC actuales. El cable USB 2.0 dispone de cuatro líneas, un par para datos, una de corriente y una de toma de tierra (figura 1.13).

- Súper alta velocidad (3.0): Tiene una tasa de transferencia de hasta 4.8 Gbps (600 MB/s). Esta especificación es diez veces más veloz que la anterior 2.0 y se lanzó a mediados del 2009 por Intel.



Pin	Signal	Color	Description
1	VCC		+5V
2	D-		Data -
3	D+		Data +
4	GND		Ground

Figura 1.13: *Cables internos que forman al cable USB.*

1.11.2. COMPATIBILIDAD Y CONECTORES

El estándar USB especifica tolerancias mecánicas relativamente amplias para sus conectores, intentando maximizarla compatibilidad entre los conectores fabricados por la compañía. El estándar USB define tamaños para el área alrededor del conector de un dispositivo para evitar el bloqueo de un puerto adyacente por el dispositivo en cuestión.

Las especificaciones USB 1.0, 1.1 y 2.0 definen dos tipos de conectores para conectar al dispositivo al servidor: A y B.

Una extensión del USB llamada “USB-On-The-Go” permite a un puerto actuar como servidor o como dispositivo, esto se determina por qué lado

del conector está conectado el aparato. Incluso después de que el cable está conectado y las unidades se están comunicando, las 2 unidades pueden cambiar de papel bajo el control de un programa.

Figura 1.14: *Diferentes tipo de conectores USB.*

1.12. ETHERNET[16]

De todos los esquemas utilizados para organizar redes de área local, Ethernet es la que tiene mayor aceptación. Aunque es uno de los esquemas más antiguos, ha mejorado con el tiempo y su tasa de transmisión aumentó de su valor original de 10 Mb/s a 100 Mb/s y en la actualidad, hasta 1 Gb/s.

Xerox dio origen a Ethernet, con la participación posterior de Intel y Digital Equipment Corporation. Alrededor de 1980, cuando el IEEE empezó a redactar normas para redes de área local, Ethernet ya era de facto una

[16] ROY BLAKE. “Sistemas Electrónicos de Comunicación” Pág. 373.

norma. Por consiguiente, aunque se dice algunas veces que Ethernet sigue la norma 802.3 del IEEE, ¡es más exacto decir que la norma IEEE sigue a Ethernet! En la actualidad, la norma Ethernet está apoyada por muchos vendedores.

Un sistema estándar y “clásico” de Ethernet utiliza un cable coaxial, especial, de 50 Ω “Ethernet grueso” (Thick Ethernet), en una configuración de bus. Se permite hasta 100 nodos en una longitud de 500 m. Los datos se transmiten por el bus mediante el código Manchester a una tasa de 10 megabits por segundo. Este sistema recibe comúnmente el nombre de 10Base5, lo cual significa 10Mb/s, banda base, 500 m de longitud. Los transmisores-receptores (Transceptores) instalados a lo largo del bus se conectan con cada nodo mediante cables separados (Figura 1.15).

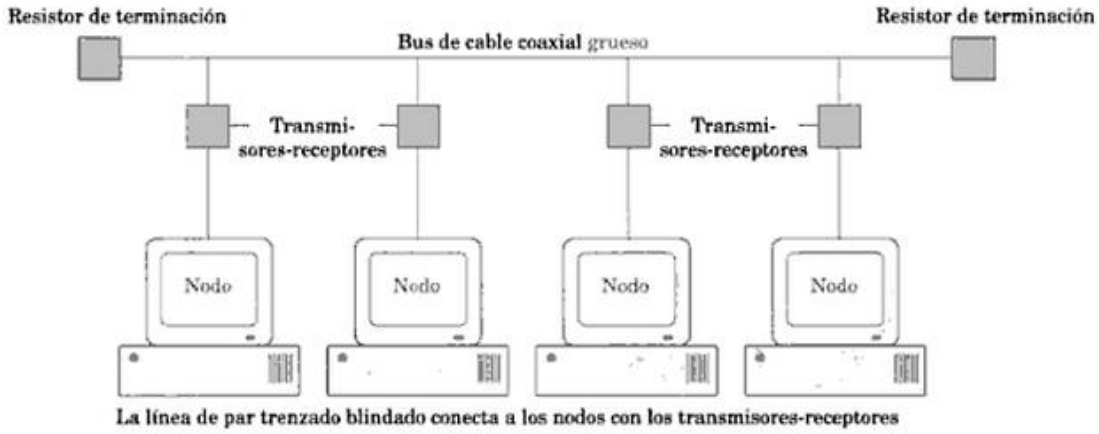


Figura 1.15: *Red “Ethernet Gruesa” (Thick Ethernet).*

El cable Ethernet grueso se reserva en la atualidad para los tendidos de cable de estructura principal (backbone) que conecta a grupos (conglomerados o clusters) de computadora. Las redes pequeñas y grupos de nodos en las redes más grandes utilizan “Ethernet delgada”

(Thin Ethernet, en términos coloquiales conocida como “cheapernet”), en la que se usa el cable coaxial común, barato, RG-58/U. en las normas IEEE, a esta instalación se la llama 10Base2 (10 Mb/s, banda base, longitud de 200 metros). Aunque la norma especifica una longitud máxima de 200 metros, en la práctica la longitud puede ampliarse hasta casi 300 sin que surjan problemas. El cable se conecta a las computadoras mediante conectores en T BNC ordinarios como se ilustra en la figura 1.16.

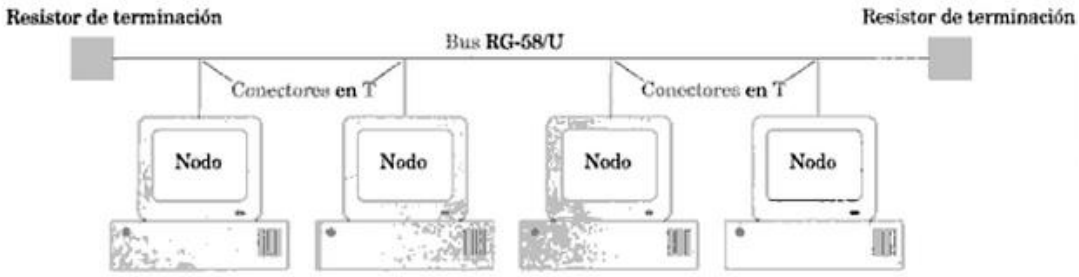


Figura 1.16: *Red “Ethernet Delgada” (Thin Ethernet).*

La Ethernet también se emplea con la línea de par trenzado, como la que se utiliza para la telefonía de voz. Este sistema de 10BaseT (T para trenzado) se restringe a una longitud de 100 m, con dos pares de alambres. Aunque lógicamente se trata de un sistema de bus, la conexión física es en estrella con un concentrador de cableado central. El concentrador desconecta un nodo defectuoso o uno con cableado en corto circuito.

También hay versiones del estándar para tasas de transmisión de dotos más altas (100MB/s) en cable de par trenzado y cable de fibra óptica. Debido a la dificultad de construir conectores en T de fibra óptica, en las redes de fibra siempre se utiliza una configuración física en estrella. La

red Ethernet puede operar en distancias más grandes con amplificación adicional (hasta una distancia de 2.8 Km en ciertas condiciones).

La razón pora la limitación en la longitud total de una LAN ethernet se encuentra en el protocolo CSMA/CD utilizado con la Ethernet.

1.13. DIRECCIÓN IP[17]

Una dirección IP es un número que identifica de manera lógica y jerárquica a una interfaz de un dispositivo dentro de una red que utilice el protocolo IP (Internet Protocol), que corresponde al nivel de red del protocolo TCP/IP.

A través de internet, los ordenadores se conectan entre sí mediante sus respectivas direcciones IP. Sin embargo, a los seres humanos nos es más cómodo utilizar otra notación más fácil de recordar y utilizar, como los nombres de dominio; la traducción entre unos y otros se resuelve mediante los servidores de nombres de dominio DNS.

En su versión 6.55, una dirección IP se implementa con un número de 32 bits que suele ser mostrado en cuatro grupos de números decimales de 8 bits (IPv4). Cada uno de esos números se mueve en un rango de 0 a 255 (expresado en decimal), o de 0 a FF (en hexadecimal), o de 0 a 11111111 (en binario). Las direcciones IP se pueden expresar como números de notación decimal: se dividen los 32 bits de la dirección en cuatro octetos. El valor decimal de cada octeto puede ser entre 0 y 255 (el número binario de 8 bits más alto es 11111111 y esos bits, de derecha a izquierda, tienen valores decimales de 1, 2, 4, 8, 16, 32, 64 y 128, lo que suma 255 en total).

[17] PROFERAMON. Dirección IP. [Documento PDF]. http://www.proferamon.com/documents/adreces_ip.pdf.

En la expresión de direcciones IPv4 en decimal se separa cada octeto por un carácter “.”. Cada uno de estos octetos puede estar comprendido entre 0 y 255, salvo algunas excepciones. Los ceros iniciales, si los hubiera, se pueden obviar.

Una dirección IP fija es una IP asignada por el usuario de manera manual, que es lo que se realizará en este proyecto.

1.14. DIRECCIÓN MAC[18]

En redes de computadoras la dirección MAC (Media Access Control Address o dirección de control de acceso al medio) es un identificador de 48 bits (6 bytes) que corresponde de forma única a una tarjeta o interfaz de red. Es individual, cada dispositivo tiene su propia dirección MAC determinada y configurada por el IEEE (los últimos 24 bits) y el fabricante (los primeros 24 bits) utilizando el OUI. La mayoría de los protocolos que trabajan en la capa 2 del modelo OSI usan una de las tres numeraciones manejadas por el IEEE: MAC-48, EUI-48 y EUI-64 las cuales han sido diseñadas para ser identificadas globalmente únicos. No todos los protocolos de comunicación usan direcciones MAC, y no todos los protocolos requieren identificadores globalmente únicos.

Las direcciones MAC son únicas a nivel mundial, puesto que son escritas directamente, en forma binaria, en el hardware en su momento de fabricación. Debido a esto, las direcciones MAC son a veces llamadas “Direcciones Quemadas Dentro” (BIA, por las siglas Burned-in Address).

La dirección Mac es un número único de 48 bits asignado a cada tarjeta de red. Se conoce también como la dirección física en cuanto identifica dispositivos de red.

[18] TELEFÓNICA. Dirección MAC. [Documento PDF]. http://www.telefonica.net/web2/fja/mac-ip.pdf.

La dirección MAC es utilizada en varias tecnologías entre las que se incluye:

a) Ethernet.

b) 802.5 o redes en anillo a 4 Mbps o 16 Mbps Token Ring.

c) 802.11 redes inalámbricas (WIFI).

d) ATM

En este proyecto se asignará la dirección MAC de una manera directa.

1.15. DIFERENCIAS ENTRE UDP Y TCP[19]

En general, las diferencias en cómo entregan los datos UDP y TCP son similares a las diferencias entre una llamada telefónica y una tarjeta postal. TCP funciona como una llamada telefónica, ya que comprueba que el destino este disponible y preparado para la comunicación. UDP funciona como una tarjeta postal: los mensajes son más pequeños y la entrega es probable, pero no siempre está garantizada.

Normalmente, utilizan UDP los programas que transmiten pequeñas cantidades de datos a la vez o que tienen requisitos de tiempo real. En estas situaciones, las capacidades de carga pequeña y multidifusión de UDP (por ejemplo, un datagrama, muchos destinatarios) resultan más apropiadas que TCP.

UDP es notablemente diferente de los servicios y características que proporciona TCP. En la tabla 1.4 se comparan las diferencias en el modo

[19] LÓPEZ Q., JUAN A.: Diferencias entre UDP y TCP. [Documento PDF]. http://dis.um.es/~lopezquesada/documentos/IES_0506/RAL_0506/doc/UT10.pdf.

de administrar la comunicación TCP/IP según se utilice UDP o TCP para el transporte de datos.

Tabla 1.4: *Diferencias entre UDP y TCP.*

UDP	TCP
Servicio sin conexión; no establece una sesión entre los hosts.	Servicio orientado a la conexión; se establece una sesión entre los hosts.
UDP no garantiza ni confirma la entrega, y no secuencia los datos.	TCP garantiza la entrega mediante el uso de confirmaciones y la entrega secuenciada de datos.
Los programas que utilizan UDP son responsables de proporcionar la confiabilidad necesaria para el transporte de datos.	Los programas que utilizan TCP proporcionan la seguridad del transporte de datos confiable.
UDP es rápido, tiene requisitos de carga pequeños y puede admitir la comunicación punto a punto y de un punto a varios puntos.	TCP es más lento, tiene requisitos de carga mayores y sólo admite la comunicación punto a punto.

UDP y TCP utilizan puertos para identificar las comunicaciones para cada programa TCP/IP.

El campo de puerto tiene una longitud de 16 bits, lo que permite un rango que va desde 0 a 65535, pero no todos los puertos son de libre uso por lo que se detalla algunas normas sobre ellos:

- El puerto 0 es un puerto reservado, pero es un puerto permitido si el emisor no espera recibir mensajes del receptor.

- Los puertos que van del 1 al 1023 reciben el nombre de puertos “bien conocidos”, y en sistemas operativos tipo Unix es necesario acceder como súper usuario para enlazarse con uno de estos puertos.

- Los puertos que van desde 1024 al 49151 son los llamados puertos “registrados” y son de libre utilización.

- Los puertos del 49152 al 65535 son puertos “efímeros”, de tipo temporal y se utilizan sobre todo por los clientes al conectarse con el servidor.

1.16. SOFTWARE DE INTRUMENTACIÓN VIRTUAL[20]

El software de instrumentación transforma al PC de un hardware de adquisición y acondicionamiento de señales en un sistema completo de adquisición y generación de señales, análisis, procesamiento y visualización de datos, es decir, es el último y tal vez uno de los más importantes ingredientes para realizar una aplicación de instrumentación virtual. A la hora de elegir el software adecuado se debe considerar entre otras cosas:

- Compatibilidad con el Hardware de adquisición y acondicionamiento.

- Funcionalidad expresada en drivers para manejar un determinado hardware.

[20] INELE. Instrumentación Virtual. [Documento PDF]. http://www.inele.ufro.cl/apuntes/Instrumentacion_y_Control/Ivan_Velazquez/Catedra/Capitulo%207.%20Instrumentacion%20Virtual.pdf.

- Sistema operativo bajo el cual opera.

- Potencia y flexibilidad.

- Dificultad y complejidad en la programación.

De acuerdo a las características antes mencionadas, el usuario podrá implementar y desarrollar sistemas a la medida de sus necesidades, entregándole una poderosa herramienta para realizar expansiones, modificaciones y generación de nuevos sistemas y aplicaciones en instrumentación, control, monitoreo y automatización de procesos industriales.

Los procesos de automatización para la adquisición y control de instrumentos es una tarea difícil. La inherente dificultad se debe a que los procesos pueden ser confusos. Una de las herramientas disponibles hoy en día es el lenguaje de programación gráfica o lenguaje “G” que ofrece LabVIEW, el cual es útil en aplicaciones de control e instrumentación, ya que es un lenguaje de programación de alto nivel que cuenta con funciones para adquisición y procesamiento de datos.

LabVIEW facilita la tarea de adquisición, análisis y presentación de datos; por lo tanto, solo es necesario el enfoque en el problema original de adquisición o medición. La filosofía de programación del lenguaje “G”, toma como base estructurada de un instrumento tradicional, el cual cuenta con un “panel frontal” (controles, botones e interruptores) para configurar el proceso de medición e indicadores para desplegar el valor medido: detrás del panel frontal se tiene componentes electrónicos que desarrollan la función del instrumento, tales como la conversión de una cantidad física en una señal eléctrica para posteriormente convertirlo a un valor numérico.

Un Instrumento Virtual (VI) es un programa diseñado, en LabVIEW en este caso, para que tenga las mismas características de un instrumento tradicional. En particular, un VI tiene un *"*panel frontal*"* desplegado en la pantalla de la computadora y este opera mediante el teclado o el mouse; el programa o código fuente, representa el ensamble de componentes electrónicos que desarrollan la función del VI; en LabVIEW es llamado "Diagrama de Bloques". El cual se construye uniendo bloques (funciones) mediante líneas que llevan el flujo de datos.

Un instrumento virtual es un módulo software que simula el panel frontal de un instrumento que antes hemos conectado y, apoyándose en elementos hardware accesible por el ordenador (tarjetas de adquisición, tarjetas DSP, instrumentos accesibles vía GPIB, VXI, RS-232), realiza una serie de medidas como si se tratase de un instrumento real.

De este modo cuando se ejecuta un programa que funciona como instrumento virtual o VI (virtual instrument), el usuario ve en la pantalla de su computador un panel cuya función es idéntica a la de un instrumento físico, facilitando la visualización y el control del aparato. A partir de los datos reflejados en el panel frontal, el VI debe actuar recogiendo o generando señales, como lo haría su homologo físico.

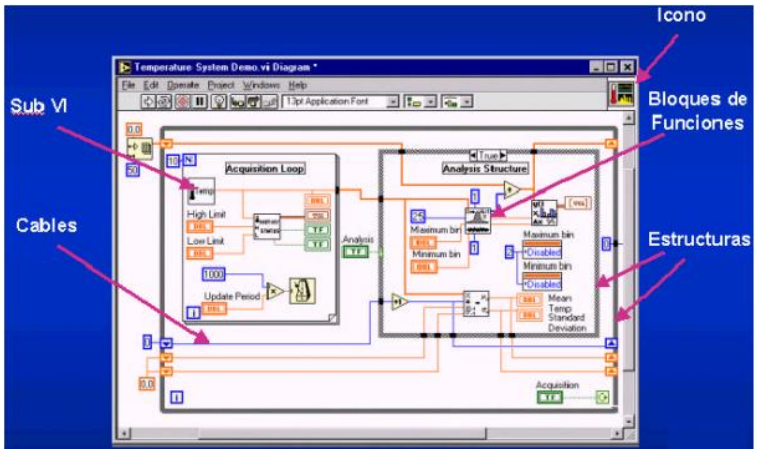


Figura 1.17: *Aplicación desarrollada en LabVIEW.*

CAPÍTULO 2

DISEÑO DEL REGISTRADOR DE VARIABLES ELÉCTRICAS

2.1. DISEÑO DEL HARDWARE

2.1.1. HARDWARE DEL REGISTRADOR

Para tener una concepción general del hardware empleado en el diseño del registrador de variables eléctricas, se lo ha dividido en varias partes otorgando una mejor visión del mismo. En la figura 2.1 se indica el diagrama de bloques del hardware del registrador.

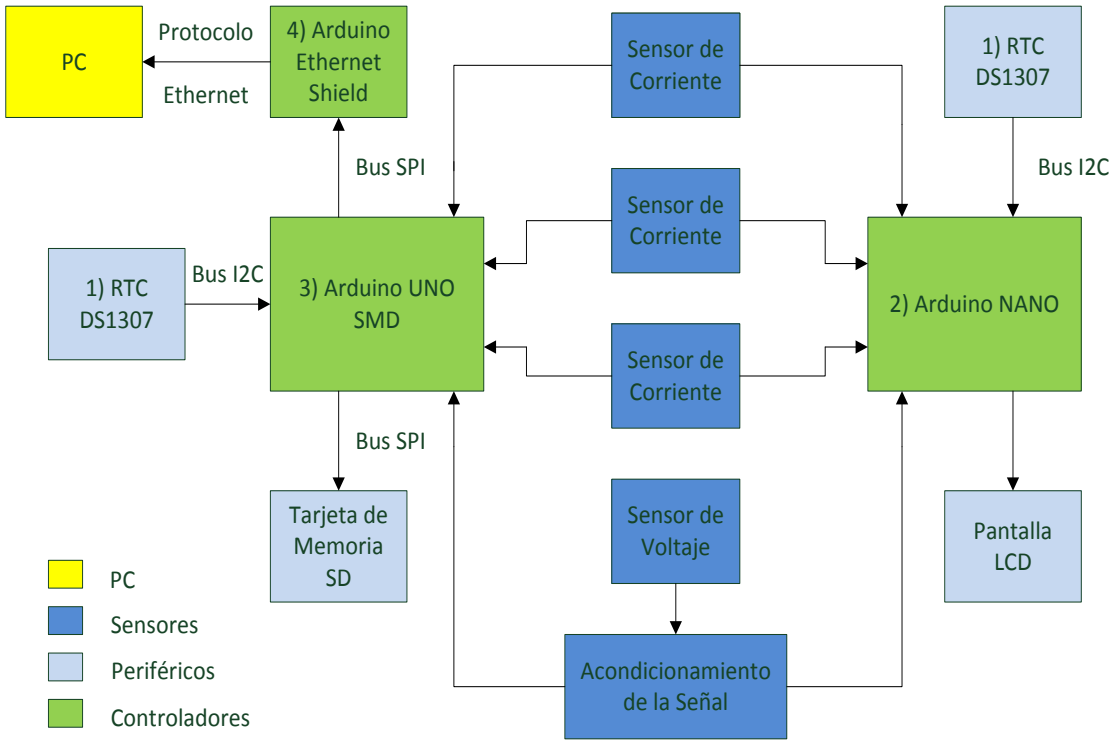


Figura 2.1: *Diagrama de bloques del hardware.*

De acuerdo a la figura anterior el hardware consta de las siguientes partes:

1) Reloj de tiempo Real (RTC): Es un integrado diseñado para dar la hora y fecha sin perder su información.

2) Tarjeta Arduino Nano: Realiza la lectura de las mediciones de los sensores, separa los argumentos, controla fecha y hora del RTC y permite al usuario la visualización de la hora, fecha y de las mediciones realizadas en una pantalla LCD 20x4.

3) Tarjeta Arduino UNO: Realiza la lectura de las mediciones de los sensores cada cierto intervalo (5 segundos), separa los argumentos, crea un archivo con extensión ".CSV" en la tarjeta de memoria SD, controla hora y fecha del RTC, envía datos a la tarjeta de memoria SD y envía datos a la tarjeta Arduino Ethernet Shield.

4) Tarjeta Arduino Ethernet Shield: Permite enviar a través de una conexión Ethernet los datos adquiridos en la tarjeta Arduino UNO.

2.1.2. SISTEMA DE ADQUISICIÓN DE DATOS

Un sistema de adquisición de datos es un equipo que permite tomar señales físicas del entorno y convertirlas en datos para llevarlas a un computador donde se puede procesar y presentar dichas señales.

La mayor parte de sistemas de adquisición de datos hoy en día son basados en computadores o en algún tipo de microprocesador que tenga un conversor Analógico-Digital. En la figura 2.2 se muestran los bloques que componen un sistema de adquisición de datos:

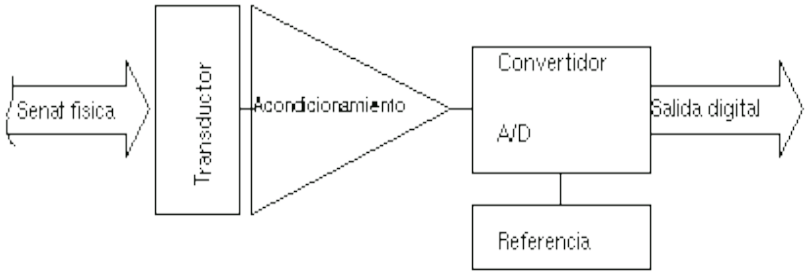


Figura 2.2: *Esquema de bloques de un sistema de adquisición de datos.*

2.1.3. SENSORES

Sensor es el elemento que se encuentra en contacto directo con la magnitud física que se va a evaluar. Estos dispositivos pueden transformar las magnitudes físicas o químicas en una señal de salida normalizada (normalmente tensión o corriente) las cuales pueden ser procesadas por el sistema de adquisición de datos.

Para la implementación y desarrollo del presente trabajo se emplearon sensores de voltaje y corriente.

2.1.3.1. SENSOR DE VOLTAJE

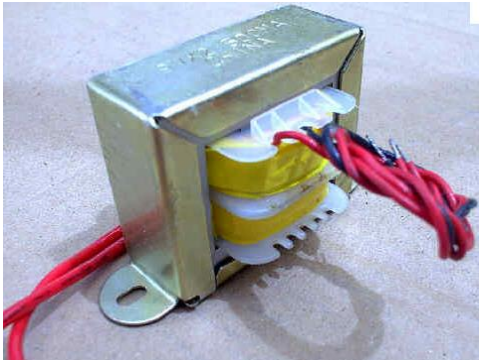


Figura 2.3: *Transformador de tensión.*

Para adquirir la señal de voltaje se seleccionó un transformador de tensión con las siguientes características:

- Voltaje de entrada: 220V/110V
- Voltaje de salida: 12V – 0V – 12V
- Frecuencia: 50/60 Hz.
- Corriente: 300mA.

El transformador presenta un comportamiento muy aceptable en cuanto a linealidad y repetitividad, permitiendo construir un dispositivo de adquisición con la mayor precisión posible. Otra de sus ventajas es que la señal de salida del sensor es una de las menores que ofrece el mercado para este tipo de transformación, lo cual garantiza que protegerá la etapa de adquisición, no solo por la potencia a la salida, sino también por su capacidad de aislar la etapa de adquisición de la red eléctrica.

2.1.3.2. SENSOR DE CORRIENTE

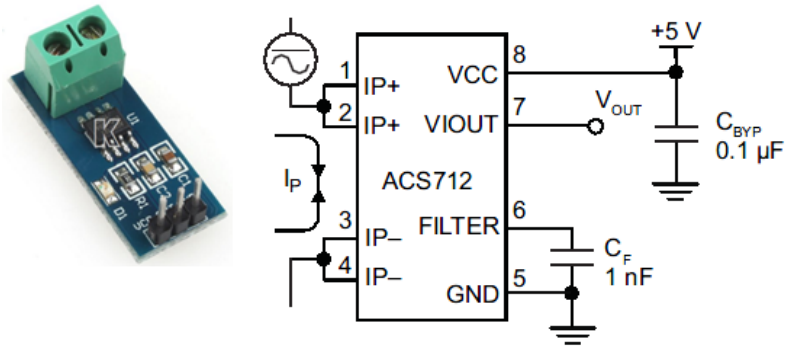


Figura 2.4: *Sensor de corriente ACS712 20A.*

El sensor de corriente seleccionado para la implementación del registrador fue el sensor de efecto hall ACS712 20A ya que no presenta

un error de repetitividad alto, además puede sensar corrientes menores a los 50 mA, mide corriente alterna y continua, y aparte ofrece aislamiento eléctrico al resto del sistema de adquisición separándolo de la red eléctrica.

Las características del sensor de corriente ACS712 20A son las siguientes:

- Pequeño y compacto.
- Error total de salida del ±1.5% a T_A=25ºC
- Rango de medida desde -20 a 20 Aca.
- Voltaje de alimentación 5Vcd.
- Corriente de alimentación 10mA.
- El ancho de banda del dispositivo se ajusta mediante el pin de filtro.
- Ancho de banda 80 kHz.
- 1.2 mΩ de resistencia conductora interna.
- Voltaje offset de salida extremadamente estable.
- Sensibilidad de salida 100mV/A.
- Voltaje de salida a 0A=2.5Vcd.

2.1.4. ACONDICIONAMIENTO DE SEÑALES[21]

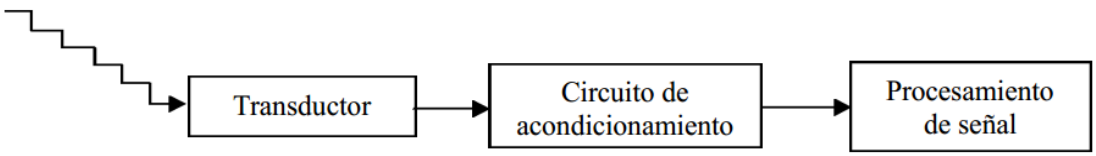


Figura 2.5: *Acondicionamiento de Señales.*

[21] INELE. Acondicionamiento de Señales. [Documento PDF]. http://www.inele.ufro.cl/apuntes/Instrumentacion_y_Control/Ivan_Velazquez/Catedra/Capitulo%203.%20Acondicionamiento%20de%20senales.pdf.

La señal de salida de un sistema de medición en general se debe procesar de una forma adecuada para la siguiente etapa de la operación. Por ejemplo: si la señal de salida es demasiado pequeña, sería necesario amplificarla; podría contener interferencias que eliminar; ser no lineal y requerir su linealización; ser análoga y requerir su digitalización; ser digital y convertirla en análoga; ser un cambio de voltaje y convertirla a un cambio en corriente de magnitud adecuada, etc., a todas estas modificaciones se les designa por lo general con el término "Acondicionamiento de Señales".

Los sistemas de instrumentación se pueden clasificar en dos clases principales: analógicos y digitales.

Los sistemas analógicos tratan en forma analógica la información de mediciones, se pueden definir como una función continua.

Los sistemas digitales manejan la información en forma digital. Una cantidad digital puede consistir en un número de pulsos discretos y discontinuos cuya relación de tiempo contiene información referente a la magnitud o naturaleza de la cantidad.

Los siguientes son algunos de los procesos que se pueden presentar en el acondicionamiento de una señal:

a) Protección para evitar el daño al siguiente elemento.

b) Acoplar la impedancia de salida de uno con la impedancia de entrada del otro, ya que los sensores presentan una salida de alta impedancia, que normalmente no puede excitar la entrada de un convertidor, cuya impedancia típica suele estar entre 1 y 10 k.

c) Convertir una señal en un tipo de señal adecuado. Sería el caso de cuando es necesario.

d) La adaptación entre los rangos de salida y entrada del sensor tiene como objetivo el aprovechar el margen dinámico del sensor, de modo que la máxima señal de entrada debe coincidir con la señal máxima que soporte el convertidor.

e) Obtención del nivel adecuado de la señal. En muchos casos es necesario amplificar una señal para que esta pueda ser leída.

f) Eliminación o reducción de ruido. La forma más común es utilizar filtros.

g) Manipulación de la señal. Por ejemplo, convertir una variable en una función lineal.

2.1.4.1. ACONDICIONAMIENTO DE LA SEÑAL DE VOLTAJE

Mediante el siguiente circuito que se muestra en la figura 2.6 se acondiciona la señal de voltaje para que pueda ingresar al conversor análogo-digital de las tarjetas Arduino UNO y Arduino Nano.

El transformador empleado en el circuito acondicionador de la señal de voltaje proporciona una tensión en el secundario de 12Vca para 220Vca a la entrada, y como en la entrada se está aplicando una tensión de 110Vca por eso la tensión en el secundario es de 6Vca, la cual al ser rectificada es de 4.6Vcd ya que el puente de diodos consume 1.4V. Pero este voltaje de 4.6Vcd a 110Vca es muy alto aún para ser ingresado al conversor A/D de las tarjetas Arduino UNO y Arduino Nano, por lo que se diseñó un divisor de voltaje con el que se obtiene 2.3Vcd a 110Vca.

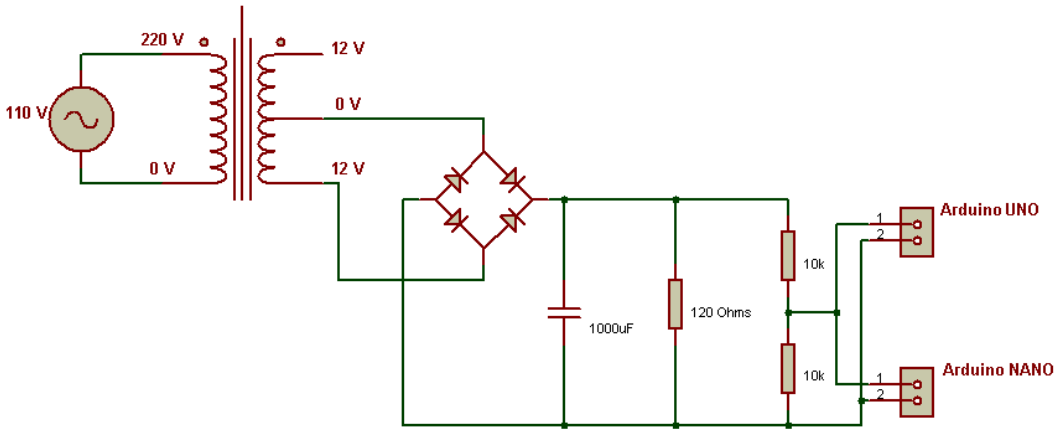


Figura 2.6: *Circuito acondicionador de la señal de voltaje.*

Con el circuito acondicionador realizado se obtiene la siguiente tabla de mediciones:

Tabla 2.1: *Tabla de mediciones del sensor de voltaje.*

Vca (V)	Vcd (V)
0	0
10	0
20	0
30	0.37
40	0.67
50	0.97
60	1.27
70	1.60
80	1.90
90	2.23
100	2.53
110	2.86
120	3.19
130	3.52
140	3.85
150	4.18
160	4.51
170	4.84

Mediante el análisis de los resultados obtenidos a través de las mediciones se pudo determinar que el rango de medición del sensor de voltaje va desde 30Vca hasta 170Vcd, lo cual es suficiente para la aplicación en desarrollo. En la figura 2.7 se indica la gráfica del voltaje de corriente continua versus el voltaje de corriente alterna.

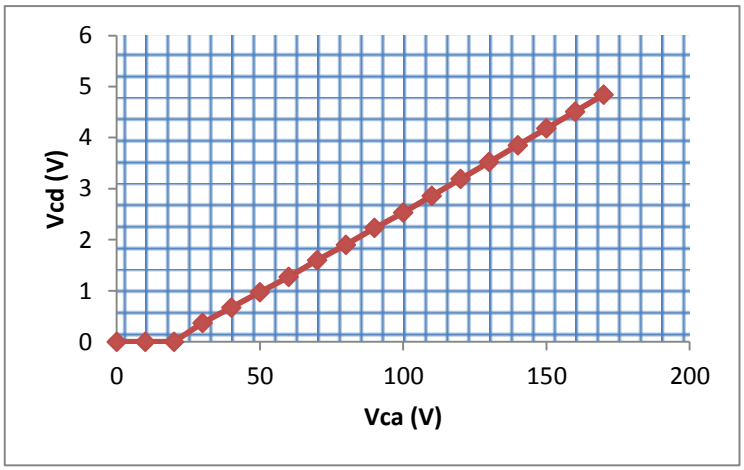


Figura 2.7: *Vcd versus Vca.*

Con las mediciones realizadas también se determinó que la sensibilidad de salida del sensor de voltaje es de 0.3Vcd/10Vca aproximadamente desde 30Vca hasta 100Vca; y, de 0.33Vcd/10Vca desde 100Vca hasta 170Vca.

2.1.4.2. ACONDICIONAMIENTO DE LA SEÑAL DE CORRIENTE

El siguiente circuito que se indica en la figura 2.8 acondiciona la señal de corriente para que pueda ingresar al conversor análogo-digital de las tarjetas Arduino UNO y Arduino Nano.

El sensado de las tres líneas de corriente se realizó a través del sensor ACS712 20A. El rango de medición del sensor de corriente va desde -20A hasta 20A. Además, este sensor requiere para su funcionamiento una alimentación de 5Vcd y a su salida entrega un voltaje proporcional a la corriente de entrada en un rango de 0 a 5Vcd (nominalmente 2.5Vcd a una corriente de 0A). Este rango de voltaje se ajusta perfectamente al rango dinámico del conversor análogo-digital de las tarjetas Arduino UNO y Arduino Nano, por lo que el circuito de acondicionamiento de la señal de corriente resulta bastante simple.

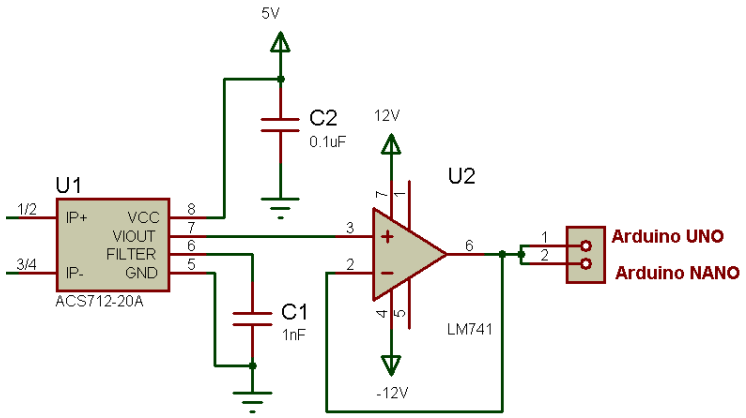


Figura 2.8: *Circuito acondicionador de la señal de corriente.*

Se utiliza un seguidor de tensión operado por un amplificador operacional, teniendo en cuenta que cumple con las siguientes características: Alta resistencia a la entrada y una muy baja resistencia a la salida, por estas razones se elimina todo lo que se refiere a los efectos de carga, los cuales producen caídas de tensión y hacen que la señales se debiliten.

En la figura 2.9 se indica la gráfica del voltaje de salida versus la corriente sensada con una alimentación de 5 Vcd.

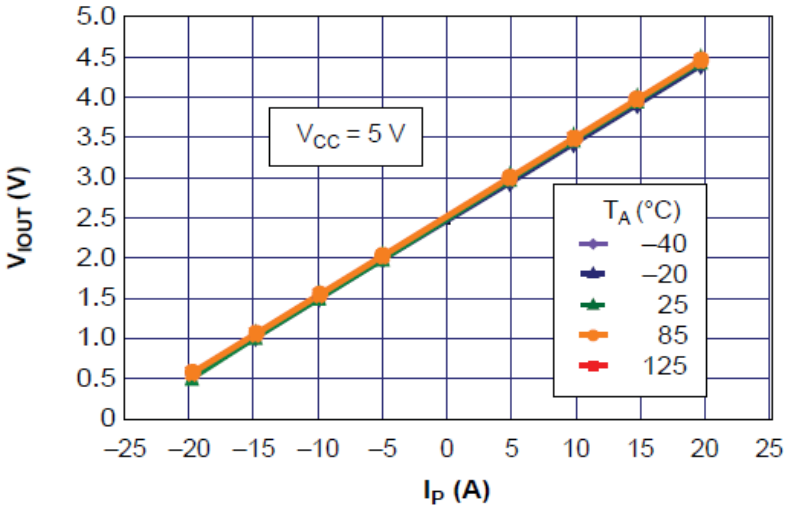


Figura 2.9: *Voltaje de salida versus corriente sensada.*

2.1.5. CONVERSIÓN ANÁLOGA DIGITAL

Las tarjetas Arduino UNO y Arduino Nano leen el valor de tensión en el pin analógico declarado. La tarjeta Arduino UNO posee 6 canales y la tarjeta Arduino Nano posee 8 canales conectados a un conversor analógico digital de 10 bits. Esto significa que convertirá tensiones entre 0 y 5 voltios a un número entero entre 0 y 1023. Esto proporciona una resolución en la lectura de: 5 voltios / 1024 unidades, es decir, 0.0049 voltios (4.9 mV) por unidad.

El conversor tarda aproximadamente 100 microsegundos en leer una entrada analógica por lo que se puede llevar una tasa de lectura máxima aproximada de 10000 lecturas por segundo.

La función que se utiliza para la lectura de las entradas analógicas es:

analogRead(*pin*)

Donde, *pin* indica el número del pin de la entrada analógica que deseamos leer (0 a 5 en la mayoría de las tarjetas, de 0 a 7 en la tarjeta Arduino Nano y de 0 a 15 en la tarjeta Arduino Mega).

La función analogReference(*tipo*) (donde, *tipo* indica el tipo de referencia que se desea usar: DEFAULT, INTERNAL y EXTERNAL) configura el voltaje de referencia usado por la entrada analógica. La función analogRead() devolverá un valor de 1023 para aquella tensión de entrada que sea igual a la tensión de referencia. Las opciones son:

a) DEFAULT: Es el valor de referencia analógico que viene por defecto que es de 5 voltios en tarjetas Arduino y 3.3 voltios en otros modelos de tarjetas Arduino que funcionan con dicho voltaje.

b) INTERNAL: Es una referencia de tensión interna de 1.1 voltios en el ATmega168 o ATmega328 y de 2.56 voltios en el ATmega8.

c) EXTERNAL: Se usará una tensión de referencia externa que tendrá que ser conectada al pin AREF.

2.1.6. MUESTREO, CUATIZACIÓN Y CODIFICACIÓN[22]

Para realizar la transformación de una señal analógica a digital se utiliza como elemento intermedio un dispositivo denominado conversor analógico-digital o ADC. Este elemento recibe las señales analógicas provenientes de los transductores y a continuación las convierte en señales digitales, codificadas en valores numéricos binarios, “ceros” y ”unos”.

2.1.6.1. MUESTREO

El método típico para obtener una representación en tiempo discreto de una señal continua, es tomar muestras de esta en intervalos periódicos de

[22] OPPENHEIM A. “Señales y Sistemas” Pág. 515.

tiempo. Una señal discreta en el tiempo se representa matemáticamente como una serie de números, en la cual el nuevo número en la secuencia se denota como X[n], y es formalmente escrito como:

$$X = \{X[n]\}, \qquad -\infty < n < \infty \qquad \text{Ec. 2.1}$$

Donde n es un entero. En el caso práctico de tomar muestras periódicas a una señal analógica el valor numérico del nuevo número en la secuencia es igual al valor de la señal analógica, $X_a(t)$, en el tiempo nT;

$$X[n] = X_a(nT), \qquad -\infty < n < \infty \qquad \text{Ec. 2.2}$$

La cantidad T se conoce como el periodo de muestreo, y su reciproco es la frecuencia de muestreo fs. El muestreo se puede entender como un proceso de apertura y cierre de un switch a una frecuencia fs, como se muestra en la figura 2.10.

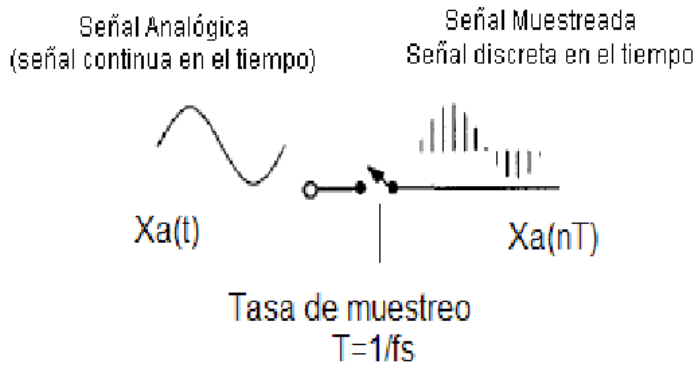


Figura 2.10: *Muestreo de una señal analógica Xa(t).*

En general, una señal discreta en el tiempo X[n], puede ser expresada en términos de una señal analógica X(t):

$$X[n] = \sum_{-\infty}^{\infty} n = X(t)\delta(t - nT) \qquad \text{Ec. 2.3}$$

Dónde:

δ(t) = 1 en t = 0,

0, para otro valor de t.

T = 1/fs

En la ecuación 2.3 se establece la relación entre la señal discreta X[n] y la señal analógica X(t).

2.1.6.1.1. CRITERIO DE NYQUIST

Cuando se pretenda muestrear una señal analógica, la frecuencia de muestreo fs, debe ser escogida de una manera muy cuidadosa para asegurar obtener una representación lo más aproximada posible a la señal original. Es claro, que entre más muestras se tomen más información se obtiene de la señal muestreada. Por otro lado, si se toman pocas muestras podría perderse una cantidad considerable de información que haría imposible reconstruir la señal original.

Las bases matemáticas del muestreo fueron dadas por Harry Nyquist en 1928. De manera simple, el criterio de Nyquist requiere que la frecuencia

de muestreo debe ser como mínimo el doble de la mayor frecuencia contenida en la señal analógica a muestrear, o de lo contrario, se presentará el fenómeno conocido como aliasing haciendo imposible la reconstrucción de la señal original. Es decir que la frecuencia de Nyquist denotada como fn es la frecuencia más alta contenida en la señal a muestrear y la frecuencia 2fn es lo que se conoce como la tasa de Nyquist.

2.1.5.2. CUANTIZACIÓN

Para esta parte del proceso los valores continuos de la señal muestreada se convierten en series de valores numéricos decimales discretos, correspondientes a los diferentes niveles o variaciones de amplitud que contiene la señal analógica original, por lo que se puede argumentar que el muestreo representa el tiempo de captura de una señal y la cuantización es el componente de la amplitud del muestreo.

2.1.5.3. CODIFICACIÓN

La codificación es la representación numérica de la cuantización utilizando sistemas numéricos ya establecidos.

2.2. DISEÑO DEL SOFTWARE

2.2.1. AMBIENTE DE LABVIEW UTILIZADO EN EL PROYECTO

Una vez revisado el diseño del hardware del registrador de variables eléctricas se procederá a explicar las funciones utilizadas del programa de LabVIEW para este proyecto.

2.2.1.1. FUNCIONES EMPLEADAS EN LABVIEW

Para enviar información por Ethernet es necesario enviar la información como un tipo de dato “string”, por lo cual en este proyecto se usó diferentes funciones para el manejo de strings que permiten enviar, recibir y procesar la información deseada. Las funciones utilizadas para el manejo de strings son las siguientes:

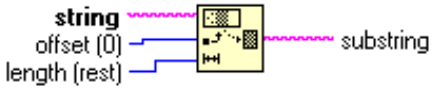


Figura 2.11: *Función String Subset.*

Devuelve una subcadena de caracteres de la cadena de caracteres de entrada que comienza en el offset y contiene el número de caracteres de la longitud.

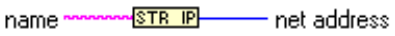


Figura 2.12: *Función String To IP.*

Convierte una cadena de caracteres en una dirección IP válida.

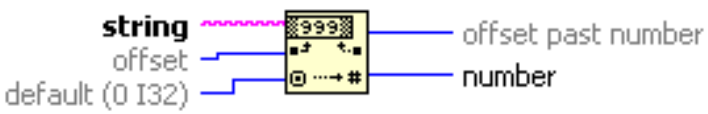


Figura 2.13: *Función Decimal String To Number.*

Convierte los caracteres numéricos contenidos en una cadena en un número decimal.

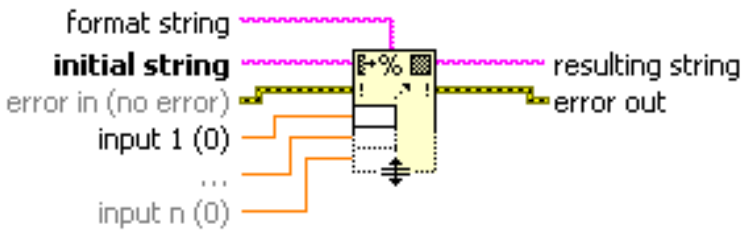


Figura 2.14: *Función Format Into String.*

Une en un solo string formatos de cadena, rutas, tipos de enumerados, marcas de tiempo, booleano o datos numéricos. Se utiliza esta función para dar formato a los datos en forma de texto para que sean escritos en un archivo.

Otras funciones utilizadas en este proyecto, y que son de fácil manejo y compresión son las siguientes:

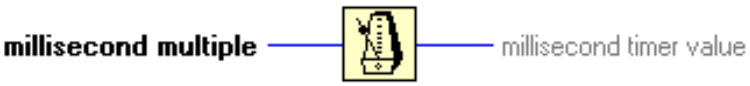


Figura 2.15: *Función Wait Until Next ms Multiple.*

Le da un retardo al lazo de la cantidad de milisegundos especificado por el usuario.

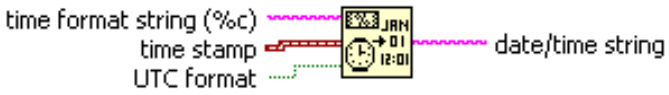


Figura 2.16: *Función Format Date/Time String.*

Permite obtener la fecha y la hora del sistema en cualquier tiempo dado mediante códigos de formato de tiempo. Por ejemplo: %d/%m/%Y %H:%M:%S, esté código permite visualizar la fecha en el formato de dd-mm-aaaa y la hora del sistema con segundos.

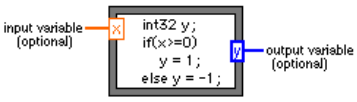


Figura 2.17: *Función Formule Node.*

Evalúa fórmulas matemáticas y expresiones similares a lenguaje C en el diagrama de bloques.

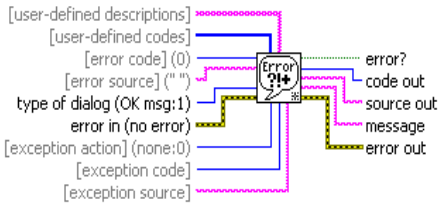


Figura 2.18: *Función General Error Handler VI.*

Indica si se ha producido un error. Si se produce un error, el VI devuelve una descripción del error y opcionalmente muestra un cuadro de diálogo.

Para la comunicación Ethernet se empleó las siguientes funciones:

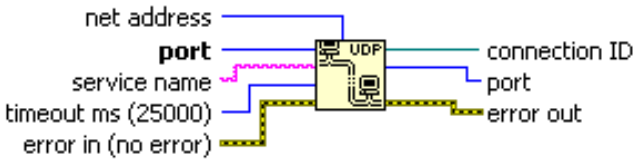


Figura 2.19: *Función UDP Open.*

Abre un puerto de comunicación Ethernet con comunicación UDP con el número de puerto especificado por el usuario.

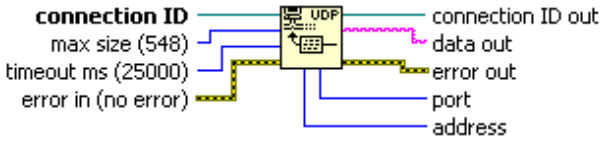


Figura 2.20: *Función UDP Read.*

Lee los datos recibidos por UDP en el puerto especificado. El tiempo de espera por los datos es ingresado por el usuario.

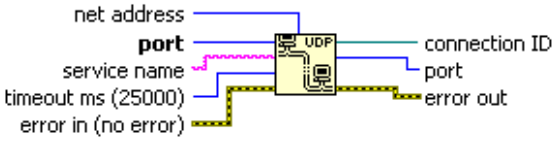


Figura 2.21: *Función UDP Write.*

Escribe información a un puerto remoto de conexión.

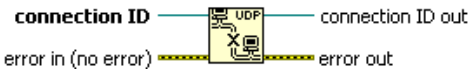


Figura 2.22: *Función UDP Close.*

Cierra la conexión UDP.

Las funciones utilizadas para la creación y escritura del archivo de almacenamiento de datos son las siguientes:

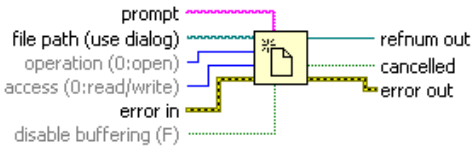


Figura 2.23: *Función Open/Create/Replace File.*

Abre un archivo existente, crea un nuevo archivo, o reemplaza un archivo existente, mediante programación o de forma interactiva mediante un cuadro de diálogo de archivo. Esta función no se ejecuta para archivos dentro de una licencia privada.

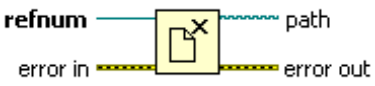


Figura 2.24: *Función Close File.*

Cierra un archivo abierto especificado por *refnum* y devuelve la ruta de acceso al archivo asociado a *refnum*.

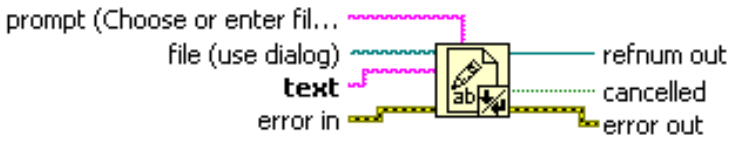


Figura 2.25: *Función Write to Text File.*

Escribe una cadena o un arreglo de cadenas de caracteres como una línea en un archivo.

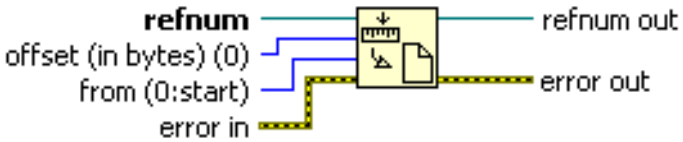


Figura 2.26: *Función Set File Position.*

Mueve la posición de la marca de un archivo actual del archivo identificado por *refnum* a la posición indicada por offset.

2.2.2. ENTORNO DE DESARROLLO PARA ARDUINO

El entorno de desarrollo para Arduino está constituido por un editor de texto para escribir el código, un área de mensajes, una consola de texto, una barra de herramientas con botones para las funciones comunes, y una serie de menús como se indica en la figura 2.27. Este software

permite la conexión con el hardware de Arduino para cargar los programas y comunicarse con ellos.

Figura 2.27: *Entorno de desarrollo para Arduino.*

Arduino utiliza para escribir la programación lo que denomina "sketch" *(*programa*),* los programas son escritos en el editor de texto. En el entorno de desarrollo de Arduino existe la posibilidad de cortar/pegar y buscar/remplazar texto. En el área de mensajes se muestra información mientras se cargan los programas y también muestra los errores. La consola muestra el texto de salida para el entorno de Arduino incluyendo los mensajes de error completos y otras informaciones. La barra de herramientas permite verificar el proceso de carga, creación, apertura y guardado de programas, y la monitorización serie:

a) Verify/Compile: Chequea el código en busca de errores.

b) Stop: Finaliza la monitorización serie y oculta otros botones.

c)  New: Crea un nuevo sketch.

d)  Open: Presenta un menú de todos los programas de su “sketchbook”. Un clic sobre uno de ellos lo abrirá en la ventana actual.

e)  Save: Salva el programa.

f)  Upload to I/O Board: Compila el código y lo vuelca en la tarjeta Arduino.

g)  Serial Monitor: Muestra los datos enviados desde la tarjeta Arduino. Para enviar datos a la tarjeta Arduino hay que escribir el texto y pulsar el botón “SEND”. Seleccionar la velocidad (baud rate) en el menú desplegable que coincida con el configurado en la función Serial.begin() dentro del sketch. Las tarjetas Arduino se resetean cuando se conectan con el monitor serie.

Se sugiere apagar el Bluetooth del computador al momento de utilizar el software de Arduino para evitar el error Launch4j.

Además se encuentran otros comandos en los cincos menús: File, Edit, Sketch, Tools, Help. Los menús son sensibles al contexto, lo que significa que estarán disponibles sólo los elementos relevantes para la tarea que se esté realizando en ese momento.

a) Edit
   - Copy for Forum: Copia el código del sketch en el portapapeles con el formato adecuado para publicarlo en un foro, incluyendo sintaxis coloreada.

- Copy as HTML: Copia el código del sketch en el portapapeles en formato HTML, adecuándolo para insertarlo en una página web.

b) Sketch
   - Verify/Compile: Verifica los errores del sketch.
   - Import Library: Añade una librería al sketch insertando la sentencia #include en el código.
   - Show Sketch Folder: Abre la carpeta de programas en el escritorio.
   - Add File: Añade un sketch al programa. El sketch aparece en una nueva pestaña en la ventana del programa.

c) Tools
   - Auto Format: Da formato al código proporcionando estética: por ejemplo realiza las tabulaciones entre la apertura y cierre de llaves, y las sentencias que tengan que ser tabuladas lo estarán.
   - Board: Se utiliza para seleccionar la tarjeta Arduino que se va a utilizar.
   - Serial Port: Este menú contiene todos los dispositivos series (reales o virtuales) de su equipo. Se refrescarán automáticamente cada vez que se abra el menú Tools.
   - Burn Bootloader: Este elemento del menú permite grabar un gestor de arranque (bootloader) dentro del microcontrolador de la la tarjeta Arduino. Aunque no es un requisito para el funcionamiento normal de las tarjetas Arduino, esto es útil si se compra un nuevo ATmega, el cual viene normalmente sin gestor de arranque. Asegúrese que ha seleccionado la placa correcta en el menú Boards antes de cargar el bootloader.

2.2.3. PROGRAMACIÓN EN ARDUINO[23]

El microcontrolador en las placas Arduino (ATmega328) se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo de Arduino (basado en Processing).

a) Wiring: Es un lenguaje de programación de código abierto para microcontroladores.
   Wiring permite escribir software multiplataforma para controlar dispositivos ligados a una amplia gama de microcontroladores, para crear todo tipo de códigos.

b) Processing: Es un lenguaje de programación de código abierto y el entorno para las personas que desean crear imágenes, animaciones e interacciones. Inicialmente se desarrolló para servir como un cuaderno de bocetos de software y enseñar los fundamentos de la programación dentro de un contexto visual.

El lenguaje de programación de las tarjetas Arduino es de código abierto basado en la flexibilidad y en el uso simple tanto del software como del hardware.

El software de Arduino consiste en un entorno de desarrollo (IDE) y las librerías centrales. El IDE está escrito en Java y basado en el entorno de Processing. Las librerías centrales están escritas en C y C++; y, compilado con avr-gcc y AVR Libc. El código fuente para Arduino está alojado en GitHub.

Los programas desarrollados con Arduino se dividen en tres partes principales: estructura, valores (variables y constantes), y funciones. El lenguaje de programación Arduino se basa en C/C++. Estos proyectos se ejecutan sin la necesidad de conectarse con un ordenador, además

[23] ARDUINO. Referencia del Lenguaje. [Documento HTML]. http://arduino.cc/es/Reference/HomePage.

pueden comunicarse con diferentes tipos de software (Flash, Processing, MaxMSP).

Tabla 2.2: *Estructura, variables y funciones utilizadas en Arduino.*

ESTRUCTURA	VARIABLES	FUNCIONES
setup() *(inicialización)*	**Constantes**	**E/S Digitales**
loop() *(bucle)*	HIGH | LOW	pinMode()
	INPUT | OUTPUT	digitalWrite()
Estructura de control	true | false	digitalRead()
If	Constante	
If…else	Numéricas	**E/S Analógicas**
for		analogRead()
switch case	**Tipos de Datos**	analogWrite() *(PWM)*
while	boolean *(booleano)*	
do…while	char *(carácter)*	**E/S Avanzadas**
break *(salida de bloque de código)*	byte	tone()
	int *(entero)*	noTone()
continue *(continuación en bloque de código)*	unsigned int *(entero sin signo)*	shiftOut()
		pulseIn()
return *(devuelve valor a programa)*	long *(entero 32b)*	
	unsigned long *(entero 32b sin signo)*	**Tiempo**
Sintaxis		millis()
;	float *(en coma flotante)*	micros()
{}		delay()
// *(comentarios en una línea)*	double *(en coma flotante de 32b)*	delayMicroseconds()
		Matemáticas
/* */ *(comentarios en múltiples líneas)*	string *(cadena de caracteres)*	min() *(mínimo)*
		max() *(máximo)*

Operadores Aritméticos = + - * / % *(resto)* **Operadores Comparativos** == != *(distinto de)* < > <= >= **Operadores Booleanos** && *(y)* || *(o)* ! *(negación)* **Operadores de composición** ++ *(incrementa)* -- *(decrementa)* += -= *= /=	array *(cadena)* void *(vacio)* **Conversión** char() byte() int() long() float()	abs() *(valor absoluto)* constrain() *(limita)* map() *(cambia valor de rango)* pow() *(eleva a un número)* sq() *(eleva al cuadrado)* sqrt() *(raíz cuadrada)* **Trigonometría** sin() *(seno)* cos() *(coseno)* tan() *(tangente)* **Números Aleatorios** randomSeed() random() **Comunicación** Serial

2.2.4. LIBRERÍAS

Las librerías proveen funcionalidad extra a nuestro sketch, por ejemplo: para trabajar con hardware o para manipular datos. Para usar una librería dentro de un sketch, se puede seleccionar desde Sketch > Import Library.

Si se desea usar una librería que no vienen junto con Arduino, es necesario instalarla. Para hacerlo, hay que descargar la librería y descomprimirla. Debe ubicarse a la librería en una carpeta propia, y normalmente, contiene dos archivos, uno con sufijo ".h" y otro con sufijo ".cpp". Al abrir la carpeta del software de Arduino, si ya existe una carpeta llamada "libraries", se debe colocar la carpeta de la librería ahí dentro (Por ejemplo, para instalar la librería Data Time, sus archivos deben entrar en una subcarpeta y su ruta de acceso queda así Arduino-1.0/libraries/DataTime). A continuación se tiene que reiniciar el IDE de Arduino para encontrar la nueva librería en el menú Sketch > Import Library.

2.2.4.1. LIBRERÍA DS1307.h

Esta librería permite cargar la hora y la fecha actualizada en el RTC DS1307, el mismo que es un reloj en tiempo real, mediante la comunicación I2C sin la necesidad de la librería Wire. Esta librería tiene las siguientes funciones:

- DS1307(SDA, SCL). Uso: DS1307 rtc(2, 3); // Inicia una instancia de la clase DS1307.

- getTime(). Uso: t = rtc.getTime(); // Permite leer la fecha y la hora actual.

- setTime(hour, min, sec). Uso: rtc.setTime(23, 59, 59); // Ajusta la hora a las 23:59:59.

- setDate(date, month, year). Uso: rtc.setDate(4, 10, 2012); // Ajusta la fecha al 4 de Octubre del 2012.

- halt(value). Uso: rtc.halt(true); // Establece al RTC al modo de ejecución.

- setSQWRate(rate). Uso: rtc.setSQWRate(SQW_RATE_1); // Estable la velocidad del SQW a 1 Hz.

- enableSQW(enable). Uso: rtc.enableSQW(true); // Activa la salida de onda cuadrada en SQW / OUT.

- getTimeStr([format]). Uso: Serial.print(rtc.getTimeStr()); // Envía la hora actual en una conexión serial.

- getDateStr([format]). Uso: Serial.print(rtc.getDateStr()); // Envía la fecha actual en una conexión serial.

2.2.4.2. LIBRERÍA SdFat.h

SdFat Arduino es una librería que soporta los sistemas de archivos FAT16 y FAT32 en tarjetas de memoria SD estándar y tarjetas de memoria de alta capacidad SDHC.

SdFat sólo admite nombres de archivos cortos.

Las clases principales utilizadas en la librería SdFat son: Sd2Card, SdVolume y SdFile.

a) Sd2Card: Soporta el acceso a las tarjetas SD estándar y SDHC. La mayoría de las aplicaciones solo necesitan llamar a la función SdVolume :: init(). Además la función de esta clase es reducir la latencia en la escritura por los que los datos pueden ser grabados con una pequeña cantidad de memoria RAM en las tarjetas Arduino.

b) SdVolume: Soporta particiones FAT16 y FAT32. La mayoría de las aplicaciones solo necesitan llamar a la función SdVolume :: init().

c) SdFile: Proporciona funciones de acceso a archivos, tales como: open(), read(), remove(), write(), close() y sync(). Esta clase permite el acceso al directorio raíz y subdirectorios.

Algunos ejemplos se encuentran en la carpeta de ejemplos de la librería SdFat, los ejemplos fueron realizados para probar e ilustrar a los usuarios sobre el uso de la librería SdFat.

2.2.4.3. LIBRERÍA Ethernet.h[24]

Junto con el Arduino Ethernet Shield, esta librería permite a la tarjeta Arduino UNO conectarse a internet. Puede funcionar tanto como servidor capaz de aceptar conexiones entrantes o como cliente permitiendo realizar conexiones de salida. La librería permite hasta cuatro conexiones simultáneas (conexiones entrantes, salientes, o una combinación de ambas).

a) Clase Ethernet: La clases Ethernet inicializan la librería Ethernet y la configuración de la red. Funciones:

[24] ARDUINO. Librería Ethernet. [Documento HTML]. http://arduino.cc/es/Reference/Ethernet.

- begin(): Inicializa la librería Ethernet y las configuraciones de la red.
- localIP(): Obtiene la dirección IP de la tarjeta Arduino Ethernet Shield, Es útil cuando la dirección es auto asignada mediante DHCP.

b) Clase del modo Servidor: La clase del modo Servidor crea servidores que permiten enviar y recibir datos desde los clientes conectados (programas funcionando en otros ordenadores o dispositivos). Funciones:
   - EthernetServer(): Crea un servidor que recibe conexiones entrantes en el puerto especificado.
   - begin(): Indica al servidor que comience a recibir conexiones entrantes.
   - available(): Obtiene un cliente que esté conectado al servidor y posea datos disponibles para lectura. Está conexión se mantiene activa cuando el objeto-cliente obtenido salga del ámbito de la función; para cerrarla se debe utilizar la función EthernetClient.stop().
   - write(): Escribe datos a todos los cliente conectados al servidor.
   - print(): Imprime datos a todos los clientes conectados al servidor.
   - println(): Imprime datos seguidos por un salto de línea a todos los clientes conectados al servidor.

c) Clase del modo Cliente: La clase del modo Cliente crea clientes que pueden conectarse con servidores y enviar datos a los mismos, o recibir datos de ellos. Funciones:

- EthernetClient():Crea un cliente que puede conectarse a una dirección IP y puerto especificados (definido en la función EthernetClient.connect()).
- If(EthernetClient): Indica si el cliente Ethernet especificado está listo.
- connected(): Si el cliente está o no conectado. Un cliente se considera conectado aunque la conexión se haya cerrado ya que quedan datos sin leer.
- connect(): Se conecta a una dirección IP y puerto especificados. El valor de retorno indica si la conexión falló o fue satisfactoria. También es compatible con las búsquedas de DNS a través de un nombre de dominio.
- write(): Escribe datos al servidor que está conectado el cliente.
- print(): Imprime datos al servidor por parte del cliente conectado.
- println(): Imprime datos al servidor seguido de un salto de línea por parte del cliente conectado.
- available(): Retorna el número de bytes disponibles para la lectura. Es decir, la cantidad de datos escritos en el cliente por el servidor al que está conectado).
- read(): Lee el siguiente Byte recibido por el servidor al que el cliente está conectado.
- flush(): Descarta los bytes escritos en el cliente, pero que aún no han sido leídos.
- stop(): Desconecta del servidor.

d) Clase del modo EthernetUDP: La clase del modo EthernetUDP habilita él envió y la recepción de mensajes UDP. Funciones:

- begin(): Inicializa la librería EthernetUDP.h y la configuración de la red.
- read(): Lee datos UDP desde el buffer especificado. Si no pasan argumentos, devolverá el siguiente carácter en el buffer. Esta función soló puede ejecutarse correctamente después de la función UDP.parsePacket().
- write(): Escribe datos UDP a una conexión remota. Esta función debe de estar entre las funciones beginPacket() y endPacket(). La función beginPacket() inicializa el paquete de datos, y no se envía hasta que la función endPacket() sea llamada.
- beginPacket(): Empieza una conexión UDP para escribir datos a una conexión remota.
- endPacket(): Es llamada después de escribir datos UDP a una conexión remota.
- parsePacket(): Verifica si hay un paquete UDP, y reporta el tamaño.
- available(): Obtiene el número de bytes disponibles para la lectura del buffer. Se trata de datos que ya llegaron. Para ejecutarse correctamente está función debe ser llamada después de la función UDP.parsePacket().available().
- remoteIP(): Obtiene la dirección IP de la conexión remota.
- remotePort(): Obtiene el puerto de la conexión remota UDP.

2.2.4.4. LIBRERÍA SPI.h

Esta librería permite la comunicación con dispositivos SPI con la tarjeta Arduino, siendo está el dispositivo maestro. Funciones:

- begin(): Inicializa el bus SPI configurando SCK, MOSI y SS a salidas, SCK como pulling, MOSI en low y SS en high. Hay que

tener en cuenta que una vez que se configura el pin, no se puede utilizar más como una E/S, a menos que se llame la función SPI.end() en el mismo pin. Los únicos pines que pueden ser configurados para ser gestionados por la interfaz SPI son los pines 4 y 10 en la tarjeta Arduino UNO.

- end(): Deshabilita el bus SPI (dejando el modo de los pines sin cambios).

- setBitOrder(): Define el orden de los bits que se desplazan tanto dentro como fuera del bus SPI, ya sea LSBFIRST (el bit menos significativo primero) o MSBFIRST (el bit más significativo primero).

- setClockDivider(): Define el divisor del reloj SPI respecto al reloj del sistema. En tarjetas basadas en AVR, los divisores disponibles son 2, 4, 8, 16, 32, 64 o 128. La configuración por defecto es SPI_CLOCK_DIV4, que configura al reloj SPI a un cuarto de la frecuencia del reloj del sistema (4 MHz para las tarjetas a 16 MHz).

- setDataMode(): Define el modo de los datos SPI: es decir, la polaridad y fase del reloj.

- transfer(): Transfiere un byte a través del bus SPI, tanto en envió como en recepción.

2.2.4.5. LIBRERÍA LiquidCrystal.h

La librería LiquidCrystal.h permite que una tarjeta Arduino pueda controlar pantallas LCD que sean compatibles con el chipset Hitachi HD44780, que se encuentra en la mayaría de las LCD basadas en texto. La librería

trabaja ya sea utilizando 4 u 8 bits (es decir, utilizando 4 u 8 líneas de datos, además de RS, Enable y opcionalmente RW). Funciones:

- LiquidCrystal lcd(RS, E, D4, D5, D6, D7): Crea una variable de tipo LiquidCrystal e inicializa la librería con el número de los pines de interface.

- lcd.begin(): Configura el número de filas y columnas de la pantalla LCD.

- lcd.setCursor(): Coloca el cursor en la columna y línea deseada.

- lcd.print(): Imprime un mensaje en la pantalla LCD.

- lcd.write(): Escribe un carácter en la pantalla LCD.

- lcd.clear(): Limpia la pantalla LCD y coloca el cursor en la esquina superior izquierda.

2.2.5. PROGRAMACIÓN DE LA TARJETA ARDUINO NANO

El programa que se diseñó en la tarjeta Arduino Nano realiza las siguientes tareas:

- Graba la hora y la fecha actualizada en un Real-Time-Clock DS1307 mediante la interfaz de comunicación I2C.

- Muestrea los datos de los sensores de corriente ACS712 20A tomando 5000 muestras cada 500 milisegundos.

- Acondiciona digitalmente la señal de los sensores de corriente ACS712 20 A en un rango de 0A a 20A.

- Acondiciona digitalmente la señal del sensor de voltaje en un rango que va desde 30V hasta 170V.

- Calcula los kWh.

- Presenta en la pantalla LCD los valores del voltaje, la corriente y los kWh. Además presenta la hora y fecha actualizada.

En el Anexo C – 1 se muestra el programa con sus instrucciones debidamente detalladas.

2.2.5.1. DIAGRAMA DE FLUJO DEL PROGRAMA DE LA TARJETA ARDUINO NANO

En la figura 2.28 y figura 2.29 se muestra el diagrama de flujo de la programación que se implementó en la tarjeta Arduino Nano.

Figura 2.28: *Diagrama de flujo de la tarjeta Arduino Nano.*

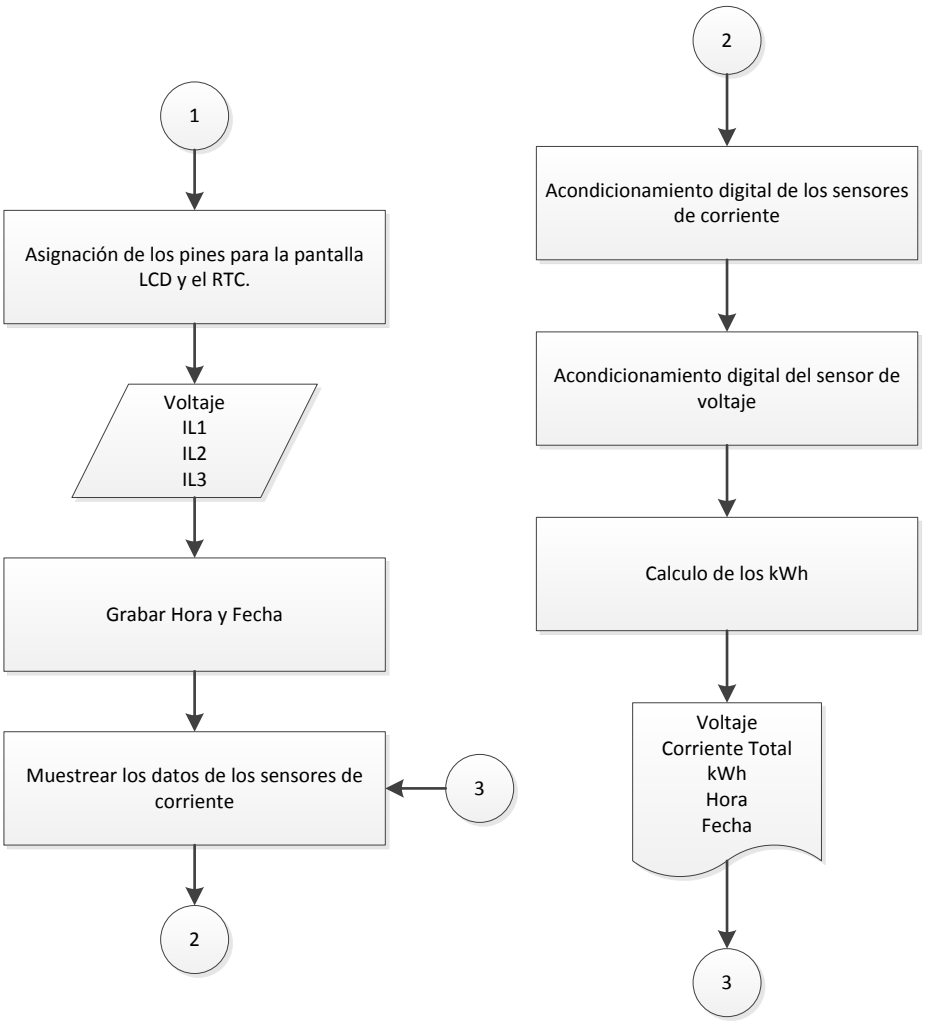


Figura 2.29: *Diagrama de flujo de la tarjeta Arduino Nano.*

2.2.6. PROGRAMACIÓN DE LA TARJETA ARDUINO UNO

Para cumplir con el objetivo planteado se diseñó tres programas para la tarjeta Arduino UNO y estos son:

a) Registrador de variables eléctricas remoto.

b) Registrador de variables eléctricas con supervisión HMI diseñada en LabVIEW.

c) Registrador de variables eléctricas remoto con monitoreo a través de un navegador web.

2.2.6.1. REGISTRADOR DE VARIABLES ELÉCTRICAS REMOTO

Este programa realiza las siguientes actividades:

- Graba la hora y la fecha actualizada en un Real-Time-Clock DS1307 mediante la interfaz de comunicación I2C.

- Crea el archivo de almacenamiento “ESPE-L00.CSV” en la tarjeta de memoria SD.

- Crea un nuevo archivo “ESPE-L00.CSV” cada vez que se reinicia la tarjeta Arduino UNO, el mismo que se enumera de 0 hasta 99 automáticamente.

- Creación de la cabecera del archivo “ESPE-L00.CSV” de almacenamiento (ID, HORA, FECHA, VOLTAJE, IL1, IL2, IL3, IT).

- Muestrea los datos de los sensores de corriente ACS712 20A tomando 5000 muestras cada 500 milisegundos.

- Acondiciona digitalmente la señal de los sensores de corriente ACS712 20 A en un rango de 0A a 20A.

- Acondiciona digitalmente la señal del sensor de voltaje en un rango que va desde 30V hasta 170V.

- Almacena la información procesada de los sensores de corriente y del sensor de voltaje con hora y fecha en una tarjeta de memoria SD cada cinco segundos en el archivo “ESPE-L00.CSV”.

En el Anexo D – 1 se muestra el programa con sus instrucciones debidamente detalladas.

2.2.6.1.1. DIAGRAMA DE FLUJO

En las figura 2.30 y 2.31 se muestra el diagrama de flujo de la programación que se implementó en la tarjeta Arduino UNO.

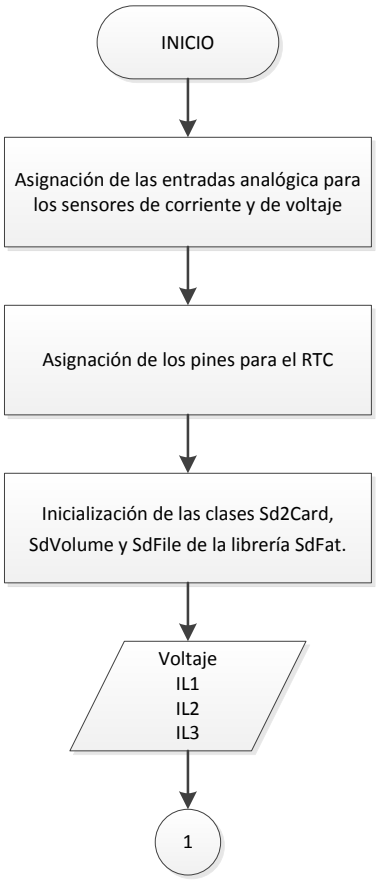


Figura 2.30: *Diagrama de flujo del Registrador de variables eléctricas remoto.*

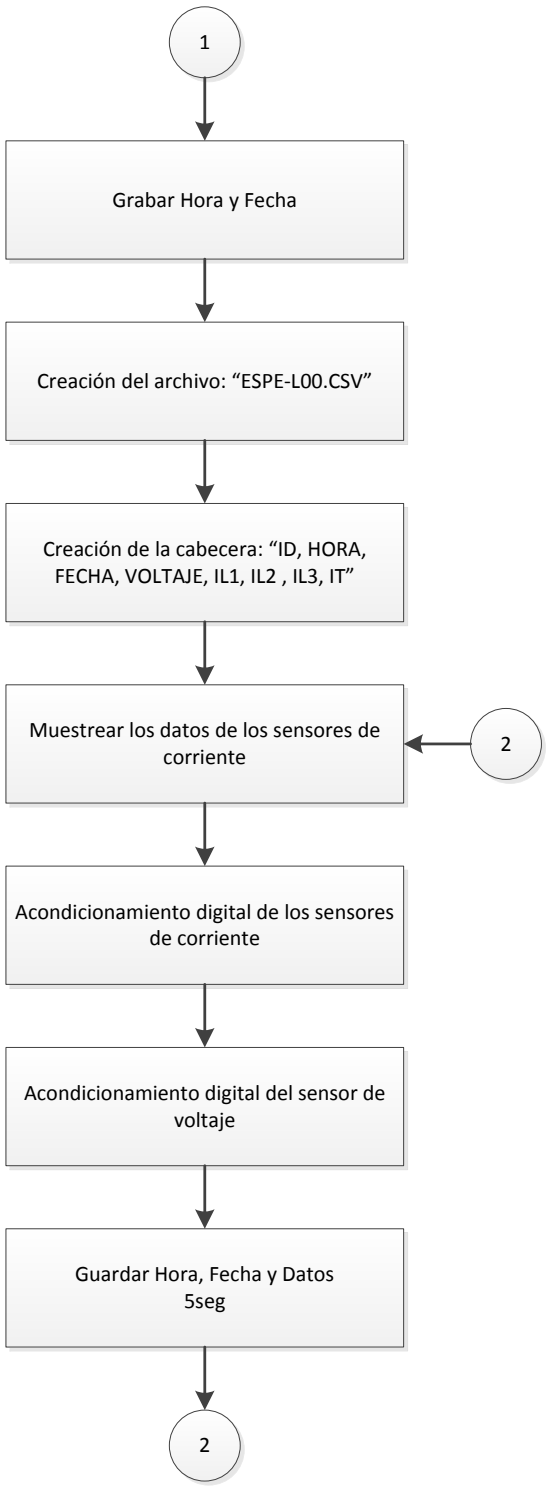


Figura 2.31: *Diagrama de flujo del Registrador de variables eléctricas remoto.*

2.2.6.2. REGISTRADOR DE VARIABLES ELÉCTRICAS CON SUPERVISIÓN HMI DISEÑADA EN LABVIEW

Para realizar esta aplicación se desarrolló dos programas: uno en el software de Arduino y otro en LabVIEW. A continuación se describen las actividades que realiza el programa diseñado en el software de Arduino:

- Envía los datos de los sensores de corriente y del sensor de voltaje a LabVIEW mediante una comunicación Ethernet UDP, por lo que se le asigna una dirección MAC, una dirección IP y un puerto local a las tarjetas Arduino UNO y Arduino Ethernet Shield, los mismo que son:
    - Dirección MAC: 0xDE.0xAD.0xBE.0xEF.0xFE.0xED
    - Dirección IP: 192.168.1.177
    - Puerto local: 62024

- Muestrea los datos de los sensores de corriente ACS712 20A tomando 5000 muestras cada 500 milisegundos.

- Lee los datos del sensor de voltaje.

- Lee el puerto local y la dirección IP de la PC con la cual se estable la comunicación Ethernet UDP.

- Convierte los datos de tipo “int” de los sensores de corriente y del sensor de voltaje a datos de tipo “char”, para poder ser enviados en una sola cadena de caracteres a LabVIEW, a través de la comunicación Ethernet UDP.

En el Anexo E – 1 se muestra el programa con sus instrucciones debidamente detalladas.

El programa diseñado en LabVIEW realiza las siguientes actividades:

- Abre la comunicación Ethernet UDP con la función de LabVIEW UDP Open. En la función UDP Open se debe asignar la dirección IP y el puerto local de la PC.
    - Dirección IP: 192.168.1.166
    - Puerto local: 49124

- Crea un archivo denominado “Datos.txt” en el escritorio de la computadora, en el cual se van a guardar los datos registrados de los sensores de corriente y del sensor de voltaje.

- Divide el string enviado por la tarjetas Arduino UNO y Arduino Ethernet Shield en cuatro substrings de cuatro caracteres mediante la función de LabVIEW String Subset, para luego convertir estos strings en valores decimales utilizando la función de LabVIEW Decimal String To Number.

- Acondiciona digitalmente la señal de los sensores de corriente ACS712 20 A en un rango de 0A a 20A.

- Acondiciona digitalmente la señal del sensor de voltaje en un rango que va desde 30V hasta 170V.

- Presenta las líneas de consumo de los sensores de corriente y del sensor de voltaje

- Guarda la información procesada de los sensores de corriente y del sensor de voltaje con hora y fecha cada dos segundos en el archivo “Datos.txt”.

En el Anexo E – 2 se muestra el programa con sus instrucciones debidamente detalladas.

2.2.6.2.1. DIAGRAMA DE FLUJO DEL PROGRAMA DISEÑADO EN EL SOFTWARE DE ARDUINO

En la figura 2.31 se muestra el diagrama de flujo de la programación que se diseñó para las tarjetas Arduino UNO y Arduino Ethernet Shield.

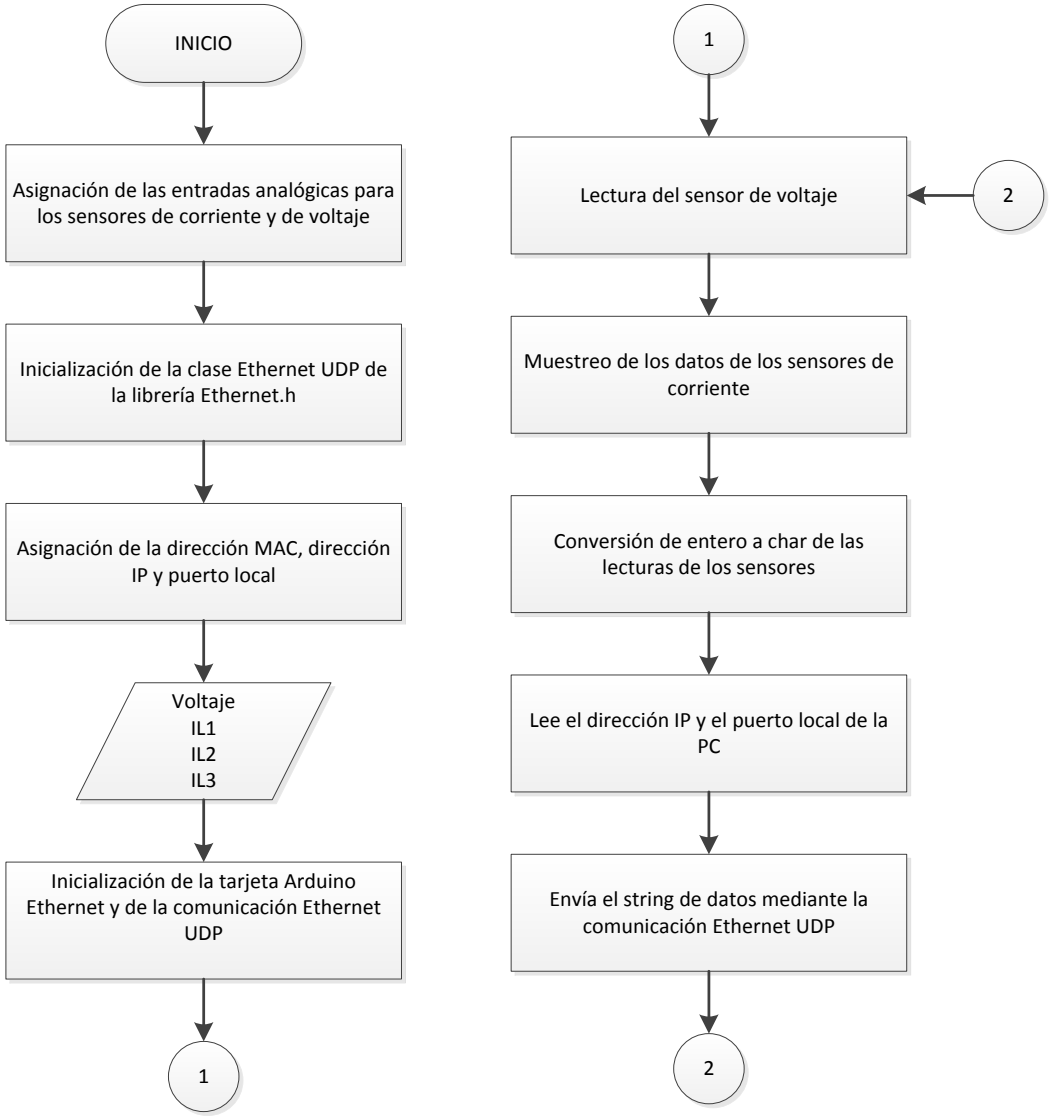


Figura 2.32: *Diagrama de flujo del registrador de variables eléctricas con supervisión HMI diseñada en LabVIEW (Software de Arduino).*

2.2.6.2.2. DIAGRAMA DE FLUJO DEL PROGRAMA DISEÑADO EN LABVIEW.

En la figura 2.32 se muestra el diagrama de flujo de la programación que se implementó en LabVIEW.

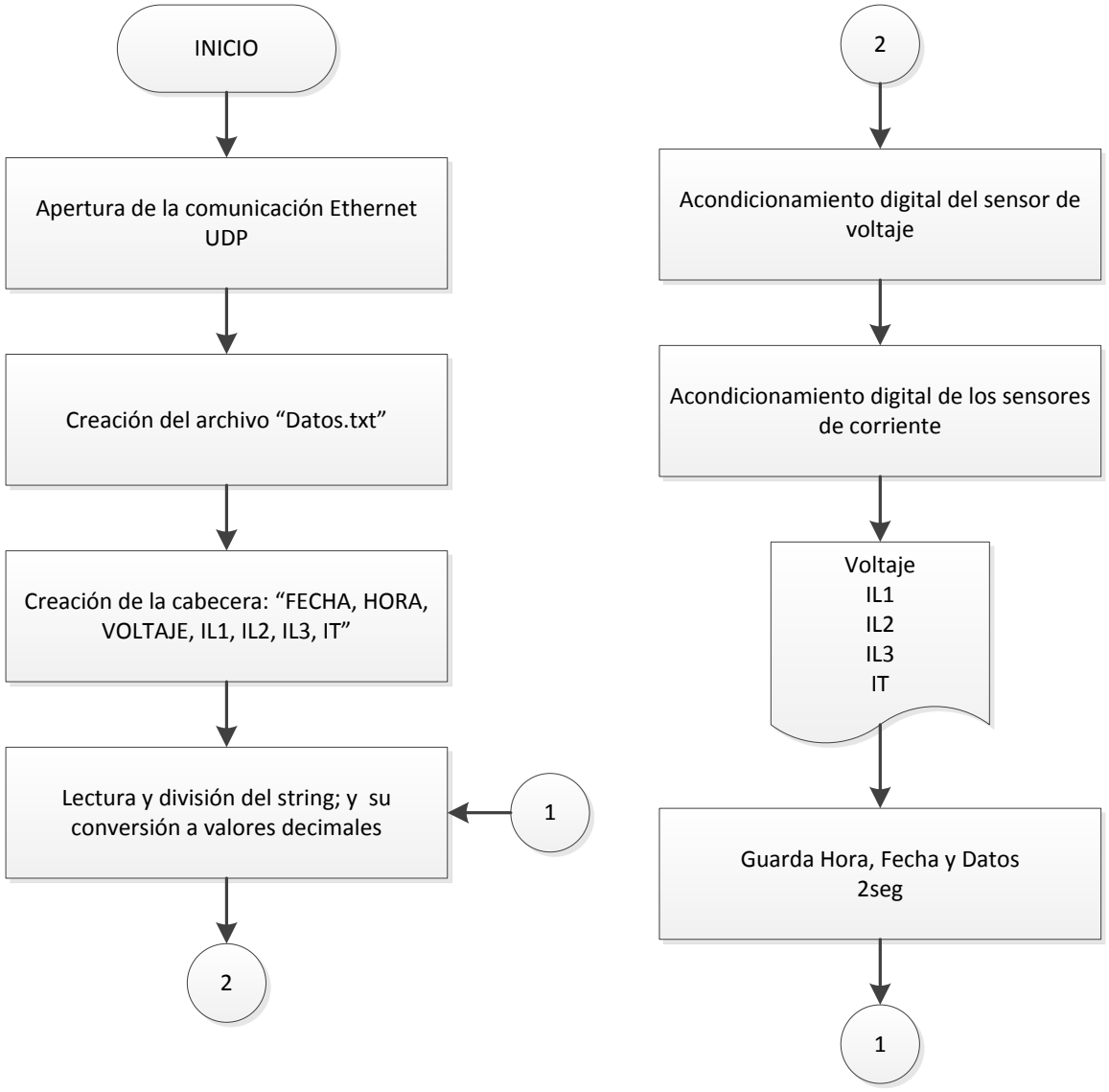


Figura 2.33: *Diagrama de flujo del registrador de variables eléctricas con supervisión HMI diseñada en LabVIEW (LabVIEW).*

2.2.6.3. REGISTRADOR DE VARIABLES ELÉCTRICAS REMOTO CON MONITOREO A TRAVÉS DE UN NAVEGADOR WEB

El programa diseñado para esta aplicación realiza las siguientes tareas:

- Como su nombre lo dice está aplicación permite monitorear en tiempo real los datos registrados en la tarjeta de memoria SD a través de cualquier navegador web, ya que la tarjeta Arduino UNO y la tarjeta Arduino Ethernet Shield están programadas para funcionar como un servidor web que puede soportar hasta cuatro conexiones simultáneas. Para que las tarjetas permitan el acceso a su información mediante un navegador web es necesario habilitar el puerto 80 (HTTP); y, además hay que asignar una dirección IP y una dirección MAC a las tarjetas Arduino UNO y Arduino Ethernet Shield, las mismas que son:
    - Dirección MAC: 0xDE.0xAD.0xBE.0xEF.0xFE.0xED
    - Dirección IP: 192.168.1.177

- Crea el archivo de almacenamiento “ESPE-L00.CSV” en la tarjeta de memoria SD.

- Crea un nuevo archivo “ESPE-L00.CSV” cada vez que se reinician las tarjetas Arduino UNO y Arduino Ethernet Shield, el mismo que se enumera de 0 hasta 99 automáticamente.

- Las tarjetas Arduino UNO y Arduino Ethernet Shield pueden ser reiniciadas desde la página web diseñada, mediante el botón de RESET.

- Creación de la cabecera del archivo “ESPE-L00.CSV” de almacenamiento (ID, VOLTAJE, IL1, IL2, IL3, IT).

- Muestrea los datos de los sensores de corriente ACS712 20A tomando 5000 muestras cada 500 milisegundos.

- Acondiciona digitalmente la señal de los sensores de corriente ACS712 20 A en un rango de 0A a 20A.

- Acondiciona digitalmente la señal del sensor de voltaje en un rango que va desde 30V hasta 170V.

- Almacena la información procesada de los sensores de corriente y del sensor de voltaje en una tarjeta de memoria SD cada cinco segundos en el archivo “ESPE-L00.CSV”.

- Actualiza los datos registrados en la tarjeta de memoria SD cada vez que se refresca la página web y también permite visualizar todos los archivos que se encuentran almacenados en la tarjeta de memoria SD.

En el Anexo F – 1 se muestra el programa con sus instrucciones debidamente detalladas.

2.2.6.3.1. DIAGRAMA DE FLUJO

En la figura 2.32 se indica el diagrama de flujo de la programación que se diseñó en el software de Arduino para las tarjetas Arduino UNO y Arduino Ethernet Shield.

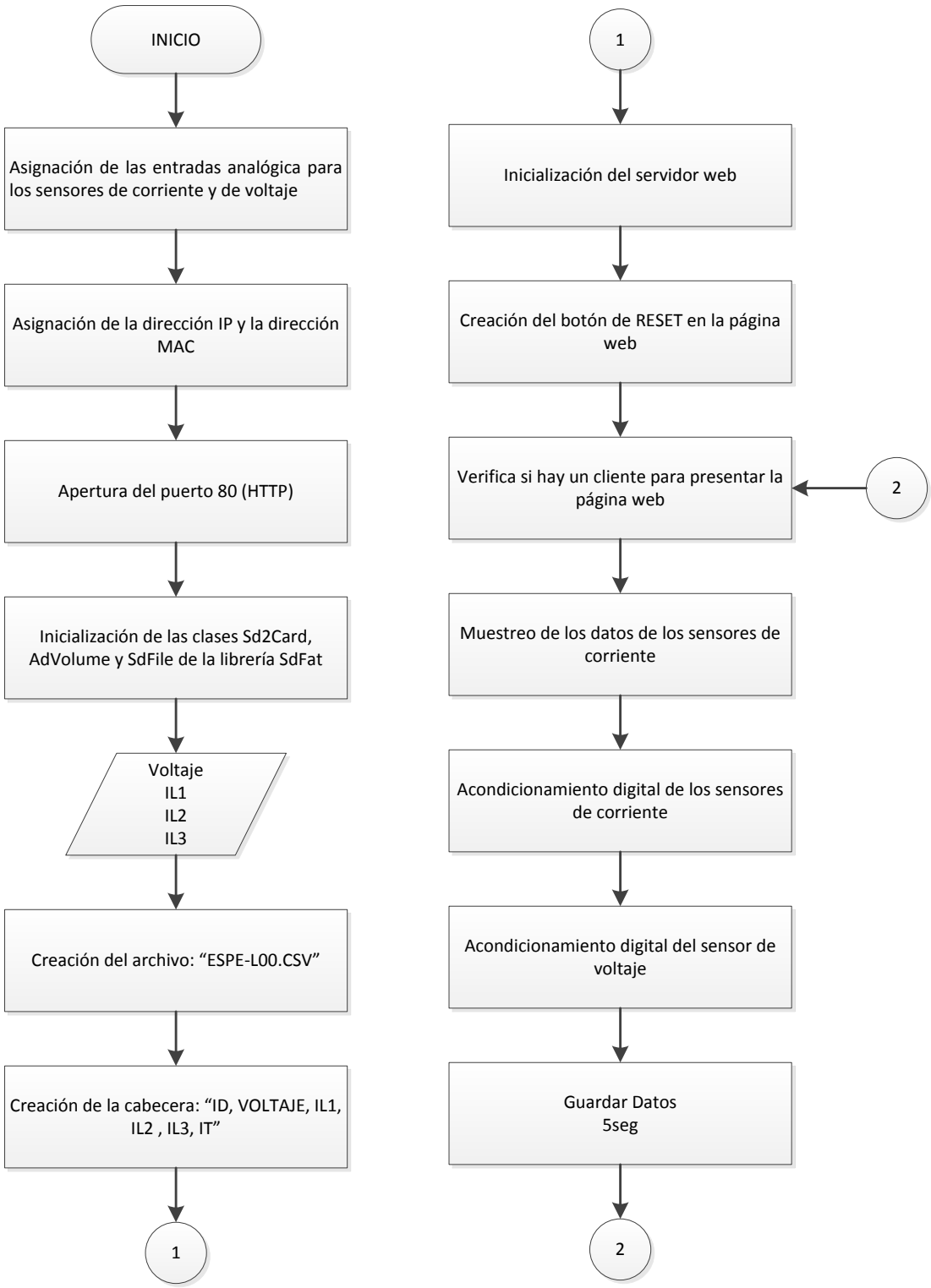


Figura 2.34: *Diagrama de flujo del registrador de variables eléctricas remoto con monitoreo a través de un navegador web*

CAPITULO 3

PRUEBAS Y RESULTADOS

3.1. GENERALIDADES

En este capítulo se realizó el análisis del comportamiento práctico del dispositivo, en un ambiente real de trabajo. Las pruebas realizadas al prototipo ayudaran a conocer los errores que se producen durante el funcionamiento para mejorar el sistema y realizar las respectivas correcciones. En tanto que los resultados de las pruebas ayudaran a la determinación de importantes conclusiones.

3.2. PRUEBAS REALIZADAS CON LOS SENSORES DE VOLTAJE Y CORRIENTE

Una vez realizado el respectivo escalamiento de los sensores en el software del registrador de variables eléctricas se efectúo las pruebas para determinar el grado de confiabilidad que brindará el prototipo.

Para el efecto de las pruebas en voltaje se utilizó el multímetro Fluke 117 y para las pruebas en corriente se utilizó el multímetro GW Instek GDM-8246. La forma en la que se llevó a cabo esta prueba fue mediante la comparación de las mediciones de los dispositivos anteriormente mencionados y el valor mostrado en el registrador de variables eléctrica. En las tablas 3.1 y 3.2 se indican las mediciones realizadas para esta prueba.

Tabla 3.1: *Comparación de las mediciones realizadas con el sensor de voltaje.*

Fluke 117 (V)	Sensor de voltaje (V)	Desviación	Error (%)
30	30,114	0,114	0,38
40	40,067	0,067	0,1675
50	50,021	0,021	0,042
60	59,974	-0,026	-0,0433
70	70,923	0,923	1,3186
80	80,877	0,877	1,0962
90	91,826	1,826	2,0289
100	100,094	0,094	0,094
110	110,023	0,023	0,0209
120	119,952	-0,048	-0,04
130	129,881	-0,119	-0,0915
140	139,81	-0,19	-0,1357
150	149,739	-0,261	-0,174
160	159,669	-0,331	-0,2069
170	169,598	-0,402	-0,2365
		ERROR	**0,4051%**

Tabla 3.2: *Comparación de las mediciones realizadas con el sensor de corriente ACS712 20A.*

GW Instek GDM-8246 (A)	Sensor ACS712 20A (A)	Desviación	Error (%)
0,525	0,53	0,0050	0,9524
1,172	1,162	-0,0100	-0,8532
1,47	1,5	0,0300	2,0408
2,12	2,13	0,0100	0,4717
2,639	2,64	0,0010	0,0379
3,088	3,1	0,0120	0,3886
3,485	3,5	0,0150	0,4304
4,12	4,15	0,0300	0,7282
4,616	4,6	-0,0160	-0,3466
5,098	5,01	-0,0880	-1,7262

5,568	5,555	-0,0130	-0,2335
6,135	6,14	0,0050	0,0815
6,675	6,67	-0,0050	-0,0749
7,245	7,25	0,0050	0,0690
		ERROR	**0,6025%**

Para el cálculo de la desviación se realizó una resta entre las dos medidas, la que marca el registrador de variables eléctricas menos el valor de los dispositivos de medida (Fluke 117 y GW Instek GDM-8246). En tanto que para calcular el error de la medida del prototipo se divide la desviación para el valor verdadero y se multiplica por 100 para obtener el error en porcentaje. Como se puede observar el error de las mediciones de voltaje y corriente que marca el registrador de variables eléctricas para el sensor de voltaje es de 0,4051%, mientras que, para el sensor de corriente ACS712 20A es de 0,6025%, por lo que se puede determinar como válidas las mediciones realizadas por el registrador de variables eléctricas.

3.3. ANÁLISIS DE LA ADQUISICIÓN DE DATOS DEL REGISTRADOR DE VARIABLES ELÉCTRICAS

En las figuras 3.1, 3.2, 3.3, 3.4, 3.5 y 3.6 se muestra el comportamiento y variación de la carga eléctrica de las variables consideradas por el registrador. La adquisición de datos se realizó en el laboratorio de circuitos electrónicos.

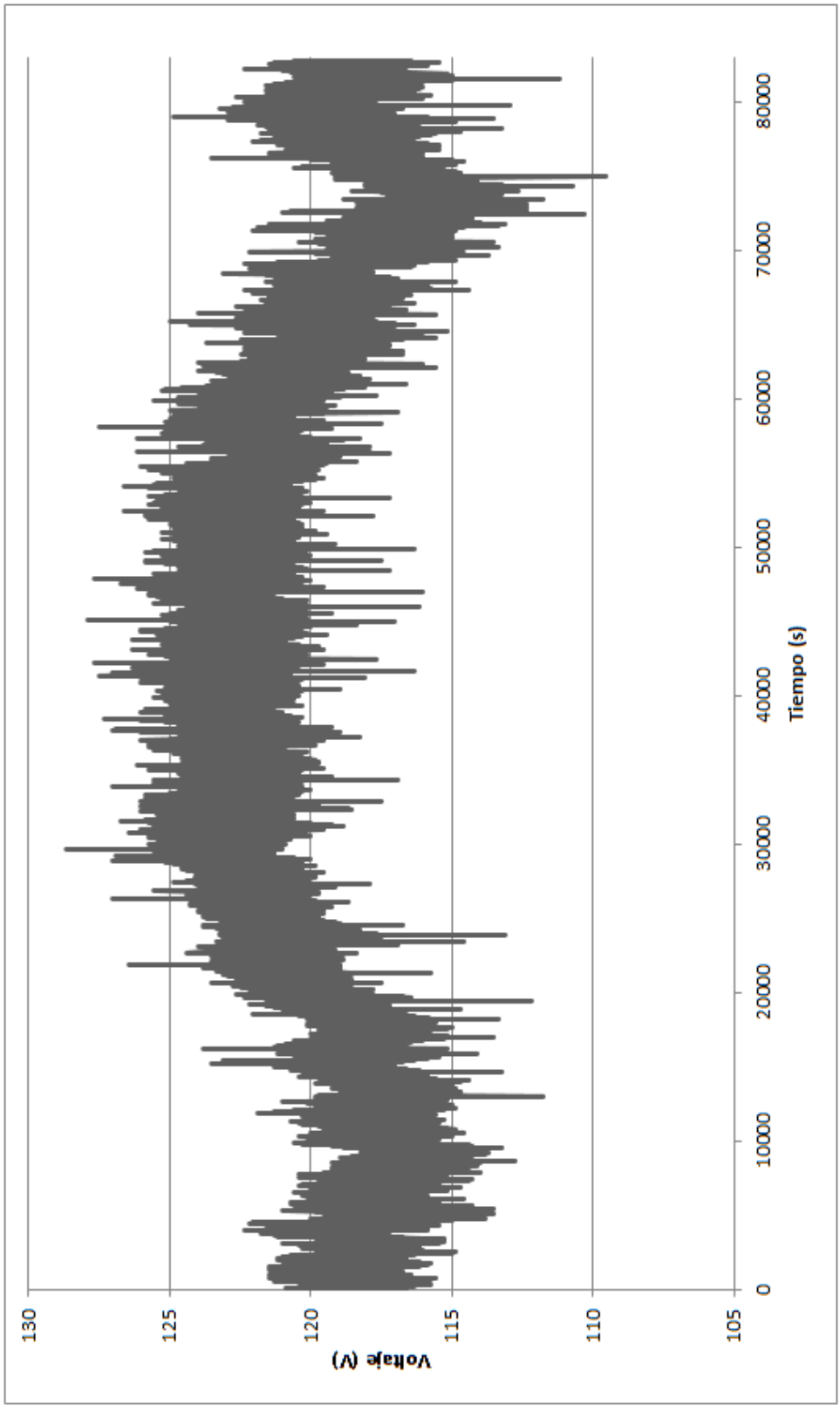


Figura 3.1: *Variación del voltaje versus tiempo.*

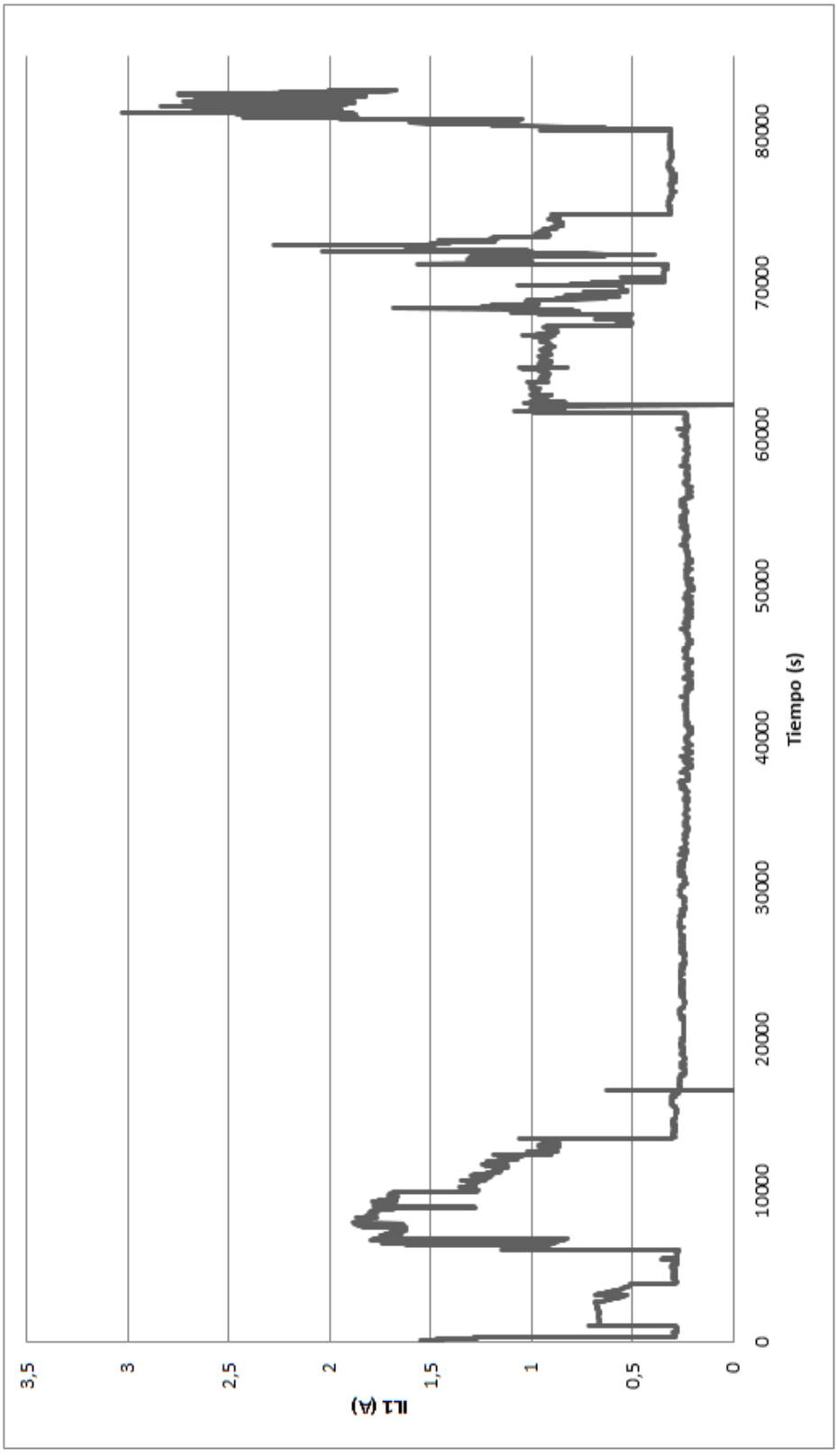


Figura 3.2: *Variación de la corriente IL1 versus tiempo.*

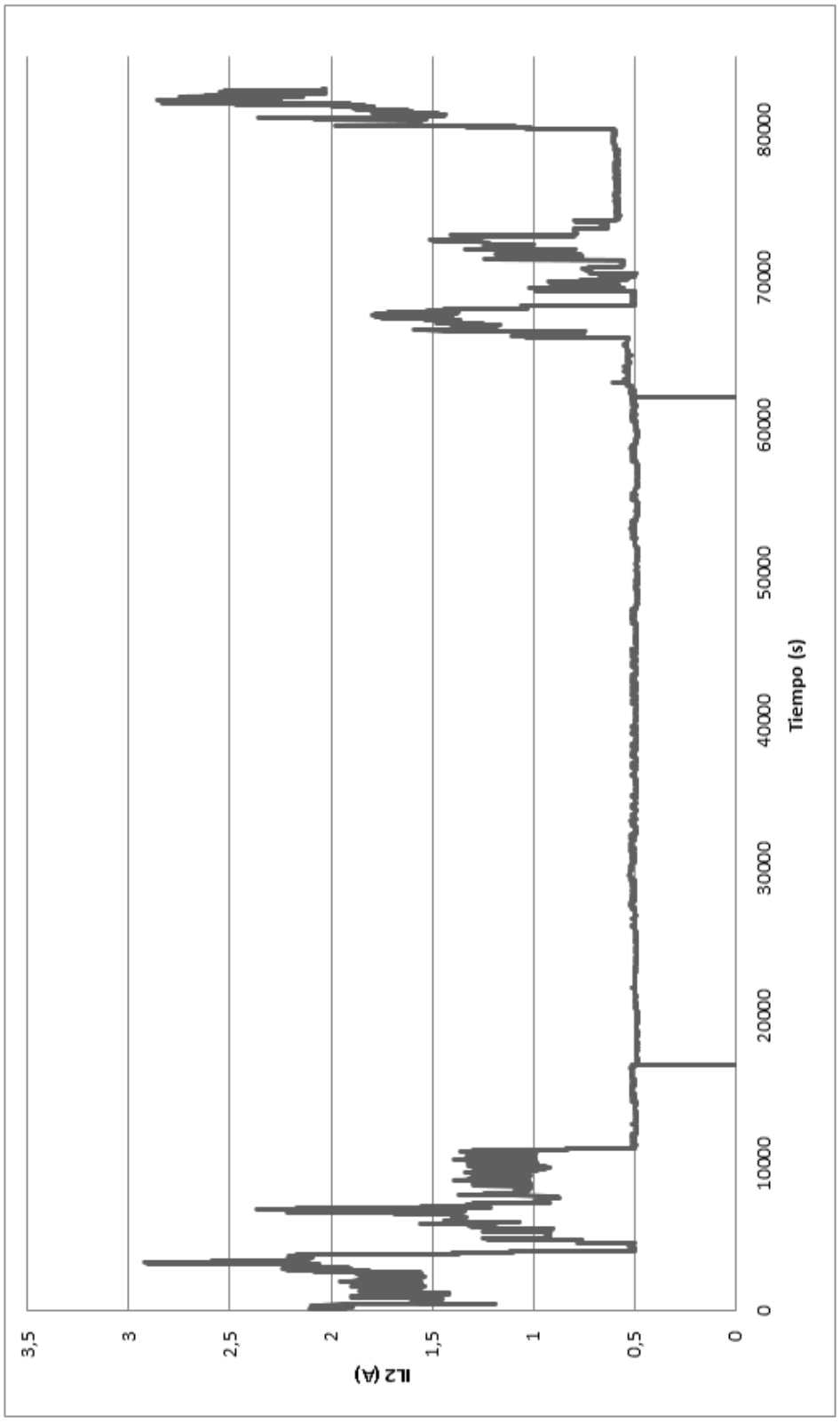


Figura 3.3: *Variación de la corriente IL2 versus tiempo.*

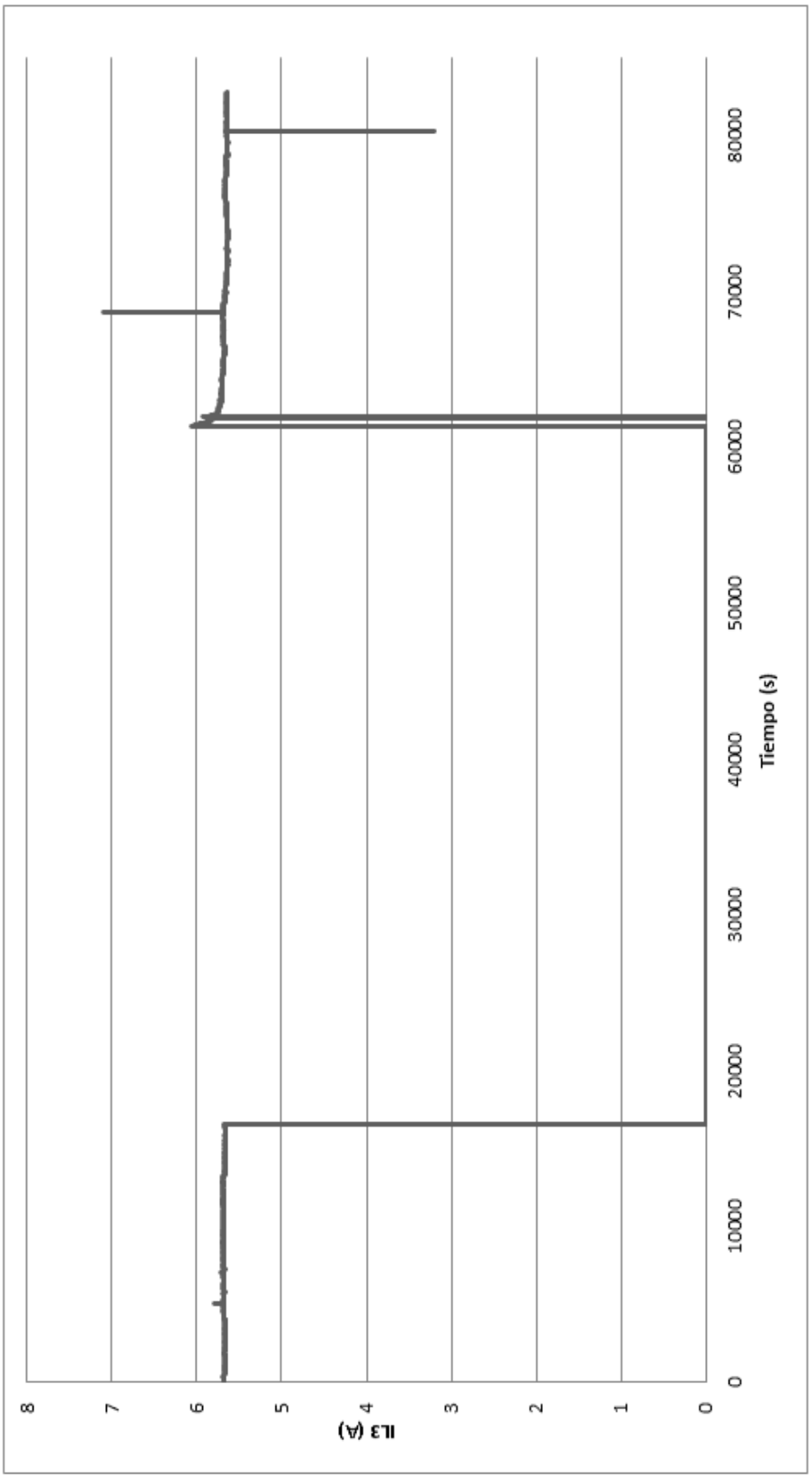


Figura 3.4: *Variación de la corriente IL3 versus tiempo.*

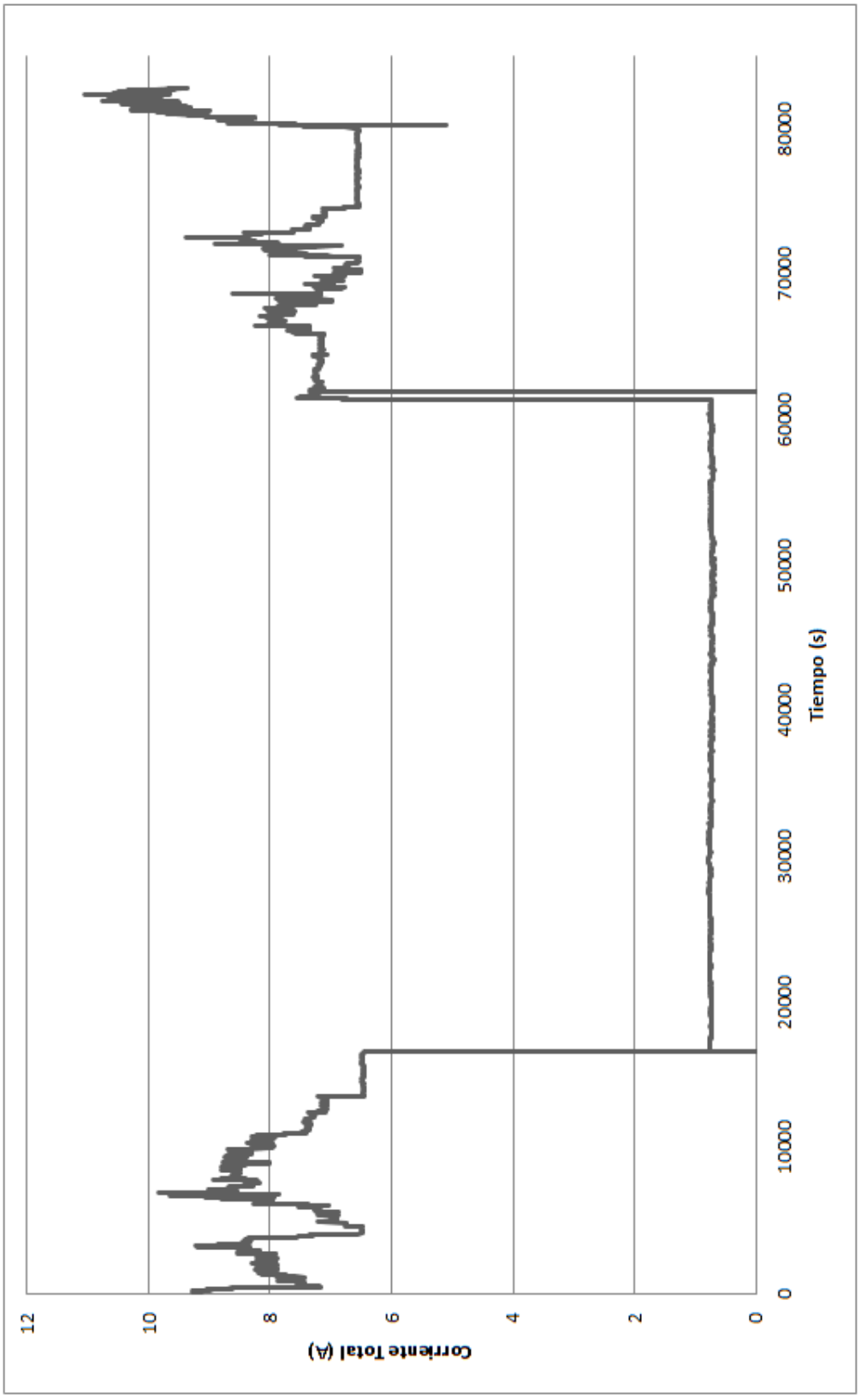


Figura 3.5: *Variación de la corriente total versus tiempo.*

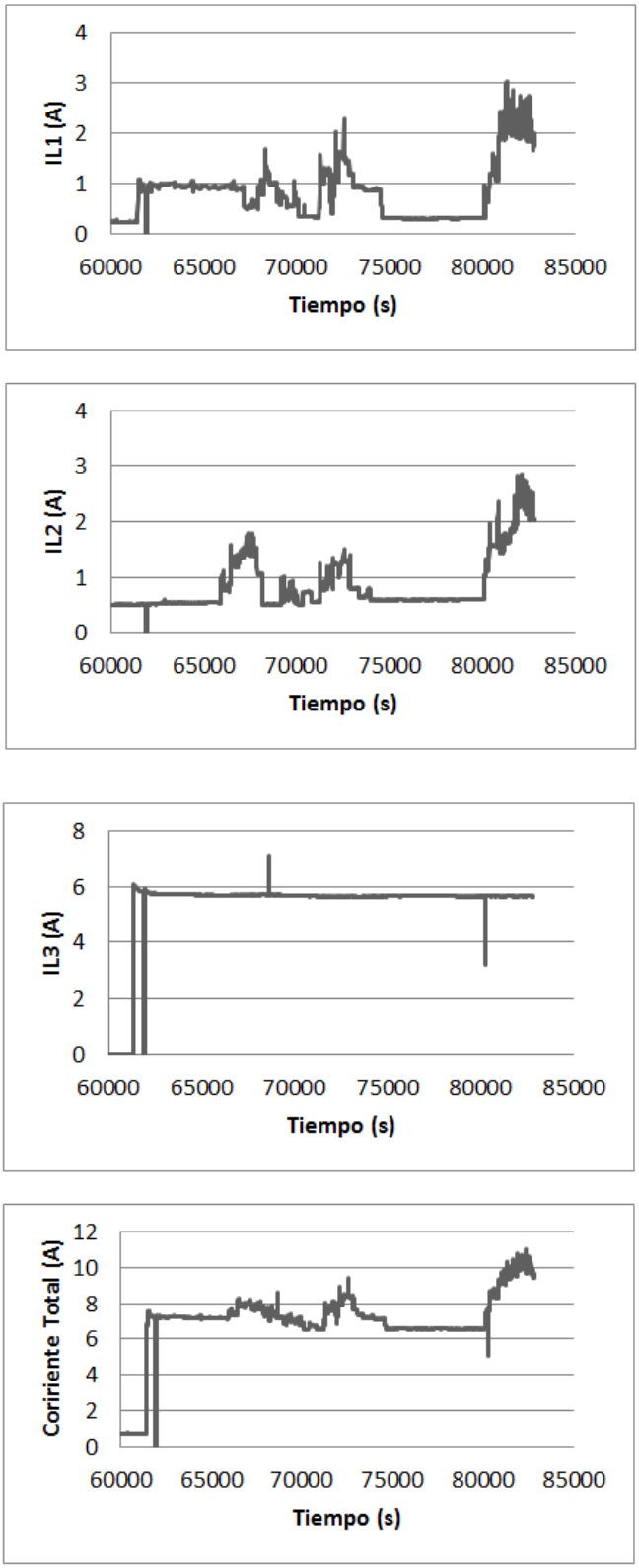


Figura 3.6: *Visualización general de las mediciones de corriente.*

La figura 3.4 corresponde al circuito de iluminación del laboratorio de electrónica en el cual se encuentran instaladas 16 lámparas fluorescentes de 40 W a 110 V, por lo que la potencia total del mismo es de 640 W y al dividir la potencia por el voltaje, se deduce que la corriente ideal es de 5.81 A, en tanto que la medición tomada por el registrador es de 5.69 A. Como se puede observar al restar la corriente ideal de la medición obtenida hay un error de 128mA, el cual se produce por el deterioro de las lámparas o la variación del voltaje de alimentación, demostrándose así el correcto funcionamiento del dispositivo desarrollado.

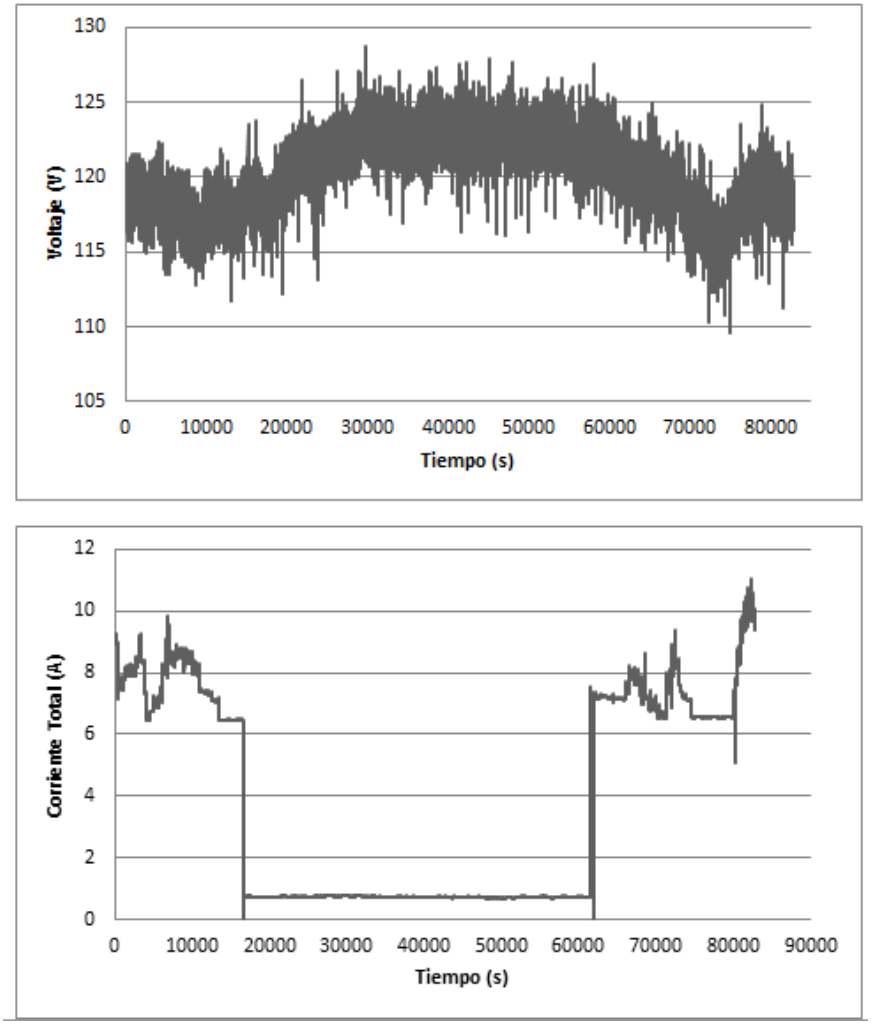


Figura 3.7: *Gráfica del voltaje y de la corriente total.*

Como se puede observar en la figura 3.7 la tensión medida varía mucho, ya que al aumentar la demanda de corriente el voltaje tiende a disminuir, lo cual valida el funcionamiento del registrador.

Las gráficas anteriores se generaron a partir del archivo de almacenamiento del registrador de variables eléctricas. La adquisición de datos se la realizó durante 23 horas por lo que se obtuvieron 13800 muestras, ya que el registrador almacena una muestra cada 6 segundos, el archivo de almacenamiento ocupó un espacio de 453 KB en la tarjeta de memoria SD por lo que se puede determinar que una tarjeta de memoria de 2 GB permite almacenar la información de 4629 días aproximadamente, lo cual es suficiente para el propósito del registrador.

3.4. PRESUPUESTO DEL REGISTRADOR DE VARIABLES ELÉCTRICAS

En la tabla 3.3 se detalla los gastos totales del registrador, los mismos que incluyen la estructura en acrílico, la programación y los componentes electrónicos.

Tabla 3.3: *Presupuesto del registrador de variables eléctricas.*

Cantidad	Descripción	Valor U.	Valor Total
1	Tarjeta Arduino UNO	35	35
1	Tarjeta Arduino Nano	30	30
1	Tarjeta Arduino Ethernet Shield	50	50
1	Pantalla LCD 20x4	20	20
3	Sensores de corriente ACS712 20A	15	45
1	Transformador de 110V a 9-0-9V 1A	7	7
1	Transformador de 220V a 12-0-12V 300mA	4	4
2	Placas PCB 20x30	4,5	9
3	LM741	0,7	2,1

2	Puentes de diodos	1	2
2	RTC DS1307	4	8
2	Osciladores de 32 kHz	1	2
2	Conectores ICD 10P	1,5	3
20	Molex 2P M&F	0,4	8
2	Molex 3P M&F	0,5	1
1	Ventilador DC 5V a 0,3 A	3,5	3,5
5	Cable Plano (m)	1,25	6,25
7	Borneras 2P	0,5	3,5
1	Tarjeta de memoria 2 GB	6	6
4	Cable Ethernet (m)	0,75	3
1	Cable USB tipo B	5	5
1	Cable USB tipo mini B	5	5
1	Estructura de acrílico	110	110
1	Varios	25	25
240	Horas de programación	12	2880
7	Diseño e impresión de placas PCB	15	105
		Total	3378,35

El precio total del registrador ya implementado y funcionando es de 3378,35 dólares. Existen en el mercado registradores de variables eléctricas producidos en serie que cumplirían con funciones similares pero a valores superiores de los 1000 dólares.

CAPITULO 4

CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

- El registrador tuvo éxito en la medición de corriente y voltaje dentro de un rango aceptable de error. Además presento una alta velocidad de actualización de la información registrada.

- Se cumplió con el objetivo de desarrollar un registrador de variables eléctricas de bajo costo y multifuncional ya que permite comunicaciones Ethernet TCP y UDP.

- Se diseñó una estructura que satisface las necesidades del registrador de variables eléctricas ya que permite realizar su conexión sin dificultades.

- El presente proyecto fue una valiosa experiencia en el diseño, implementación y prueba de un sistema que involucró el desarrollo de varios programas y la puesta a punto de diferentes componentes electrónicos.

- El sistema presenta al usuario los datos registrados en un archivo de Excel en el cual mediante sus aplicaciones gráficas se puede determinar el comportamiento de la carga eléctrica de una forma clara.

- El almacenamiento de datos del registrador de variables eléctricas permite realizar comparaciones numéricas o visuales entre datos antiguos y datos actuales.

- Los valores medidos difieren de los valores calculados porque la tolerancia de las resistencias internas de los dispositivos utilizados no son exactas.

- Las comunicaciones Ethernet que proporciona la tarjeta Arduino Ethernet Shield permite eliminar las limitaciones de distancia, ya que el computador donde se realice la supervisión puede estar a una corta o larga distancia del registrador.

- La exactitud del prototipo es directamente proporcional a la calibración realizada, por lo que requieren patrones de referencia adecuados para su correcto funcionamiento.

- Las funciones en LabVIEW que manejan la comunicación UDP facilitaron el desarrollo de este proyecto por su fácil manejo y sencillo funcionamiento.

- Las tarjetas Arduino UNO y Arduino Nano soportan perfectamente las comunicaciones SPI e I2C, lo cual facilita la conexión con diferentes periféricos.

- El ADC de las tarjetas Arduino tarda aproximadamente 100 microsegundos en leer una entrada analógica por lo que la lectura máxima por segundo aproximada es de 10000 lecturas, por eso se realizó la toma de 5000 muestras de los sensores de corriente lo que sería igual a 0.5 segundos por medida permitiendo al usuario observar la medición en un aparente tiempo real.

- El software de programación de Arduino es una herramienta muy poderosa, en especial por la facilidad que tiene para agregar librerías, las mismas que proporcionan el soporte para realizar todo tipo de aplicaciones.

- El entorno de Arduino no presenta escalabilidad en sus diferentes versiones lo cual es una desventaja al momento de programar ya que el nombre de las funciones cambia, lo que complica el uso de las librerías del software.

- La interfaz desarrollada en LabVIEW se puede visualizar los datos adquiridos en el dominio del tiempo y también presenta los datos almacenados en un archivo de Excel en tiempo real.

- Para evitar posibles efectos de carga se debe realizar el acoplamiento de etapas mediante el uso de un amplificador operacional en conexión seguidor de tensión.

- El dispositivo desarrollado puede ser utilizado para registrar el comportamiento de la carga de una residencia o de una industria, siempre y cuando sea un sistema monofásico.

- Las señales proporcionadas por los sensores deben ser transformadas a niveles estándares de señales eléctricas para que puedan ser procesadas por el ADC de las tarjetas Arduino.

- Se realizó una investigación sobre los diferentes sensores de corriente existentes en el mercado y se escogió el sensor de corriente ACS712 20A ya que presenta un tamaño compacto, bajo costo y errores de medida inferiores al 2%.

- El reloj en tiempo real (DS1307) implementado en el registrador de variables eléctricas permite visualizar y mantener en la pantalla LCD una hora y calendario real en el sistema, por lo que se puede concluir que el integrado DS1307 fue la mejor opción para implementar un RTC en el dispositivo.

- Al finalizar este proyecto se concluye que este dispositivo es de beneficio para las empresas ya que permite comparar el consumo de kWh con el de la planilla eléctrica, así como el de poder determinar las horas de mayor consumo a través de las líneas de carga, lo cual facilitaría determinar el uso de cierta maquinaria para aprovechar las diferentes tarifas eléctricas y así generar ahorro monetario para la empresa.

4.2. RECOMENDACIONES

Se recomienda:

- Apagar el Bluetooth del computador cuando se vaya a utilizar el software de programación de Arduino para evitar el error launch4j, el cual es un error de java.

- Leer detalladamente el manual de usuario antes de realizar la instalación del registrador de variables eléctricas para tener un funcionamiento adecuado del mismo.

- Es importante conectar los sensores de corriente después de las protecciones del sistema eléctrico para evitar el daño de los mismos.

- Colocar el registrador en un sitio donde no este contacto con líquidos, calor excesivo entre otros. El sitio donde se lo coloque tiene que estar al alcance del usuario para que puede visualizar y configurar al dispositivo.

- Se recomienda la difusión del uso de las tarjetas Arduino ya que cuentan con herramientas de desarrollo que facilitan la programación permitiendo desarrollar un sin número de aplicaciones.

- Leer detenidamente las hojas técnicas de cada uno de los elementos utilizados en el desarrollo del registrador de variables eléctricas.

- Soldar con cuidado para evitar daños en las placas y en los elementos electrónicos.

BIBLIOGRAFÍA

1. EDMINISTER, JOSEPH E. “Circuitos eléctricos”. Editorial McGraw Hill, Madrid, 1987.

2. NILSSON, JAMES W.; RIEDEL, SUSAN A. “Circuitos eléctricos”. Editorial Pearson Educación, S.A., Madrid, 2005.

3. ANGULO U., JOSÉ M.; ANGULO M., IGNACIO “Microcontroladores PIC Diseño práctico de aplicaciones”. Editorial McGraw Hill, Madrid, 2003.

4. BLAKE R. “Sistemas electrónicos de comunicaciones”. Editorial International Thomson S.A., México, 2004.

5. OPPENHEIM A. “Señales y Sistemas”. Editorial Prentice-Hall, México, 1998.

NETGRAFÍA

1. ARDUINO. “Arduino”. [Documento HTML]. 4 de agosto del 2012. http://www.arduino.cc/es/

2. ARDUINO. “Arduino UNO”. [Documento HTML]. 5 de agosto del 2012. http://arduino.cc/en/Main/ArduinoBoardUno

3. ARDUINO. “Arduino Nano”. [Documento HTML]. 6 de agosto del 2012. http://arduino.cc/en/Main/ArduinoBoardNano

4. ARDUINO. “Arduino Ethernet Shield”. [Documento HTML]. 7 de agosto del 2012. http://arduino.cc/en/Main/ArduinoEthernetShield

5. HISPAVILA. “DS1307”. [Documento PDF]. 10 de agosto del 2012. http://www.hispavila.com/3ds/atmega/descargas/rtc_ds1307.pdf

6. FERNÁNDEZ C., ANTONIO M. “Bus I2C”. [Documento PDF]. 11 de agosto del 2012. http://www.uco.es/~el1mofer/Docs/IntPerif/Bus%20I2C.pdf

7. SISTEMA DE DESARROLLO PARA MC, MODELO MDK-B28 “Comunicación Serial Sincrónica (SPI)”. [Documento PDF]. 15 de agosto del 2012. http://proyecto-test-hm1.googlecode.com/files/Cap_No_05.pdf

8. ESCUDERO G. “USB”. [Documento PDF]. 15 de agosto del 2012. http://www.inti.gov.ar/electronicaeinformatica/informatica/pdf /usb.pdf

9. PROFERAMON. “Dirección IP”. [Documento PDF]. 16 de agosto del 2012. http://www.proferamon.com/documents/adreces_ip.pdf

10. TELEFÓNICA. “Dirección MAC”. [Documento PDF]. 16 de agosto del 2012. http://www.telefonica.net/web2/fja/mac-ip.pdf

11. LÓPEZ Q., JUAN A. “Diferencias entre UDP y TCP”. 18 de agosto del 2012. http://dis.um.es/~lopezquesada/documentos/IES_0506/RAL_0506/doc/UT10.pdf

12. INELE. Instrumentación Virtual. [Documento PDF]. 19 de agosto del 2012. http://www.inele.ufro.cl/apuntes/Instrumentacion_y_Control/Ivan_Velazquez/Catedra/Capitulo%207.%20Instrumentacion%20Virtual.pdf.

13. INELE. Acondicionamiento de Señales. [Documento PDF]. 1 de septiembre del 2012. http://www.inele.ufro.cl/apuntes/Instrumentacion_y_Control/Ivan_Velazquez/Catedra/Capitulo%203.%20Acondicionamiento%20de%20senales.pdf.

14. ARDUINO. Referencia del Lenguaje. [Documento HTML]. 5 de septiembre del 2012. http://arduino.cc/es/Reference/HomePage.

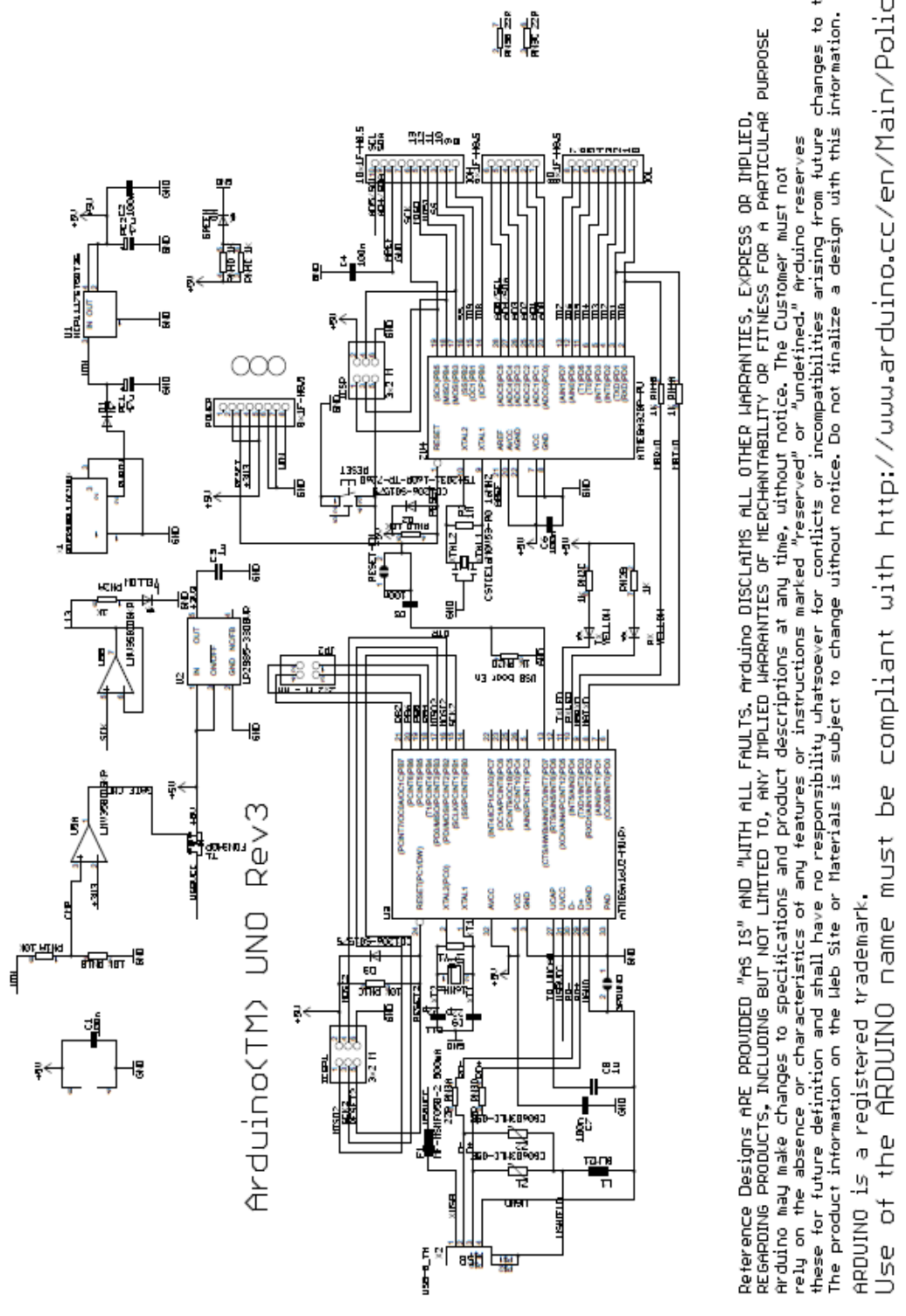
15. ARDUINO. Librería Ethernet. [Documento HTML]. 5 de septiembre del 2012. http://arduino.cc/es/Reference/Ethernet.

ANEXOS

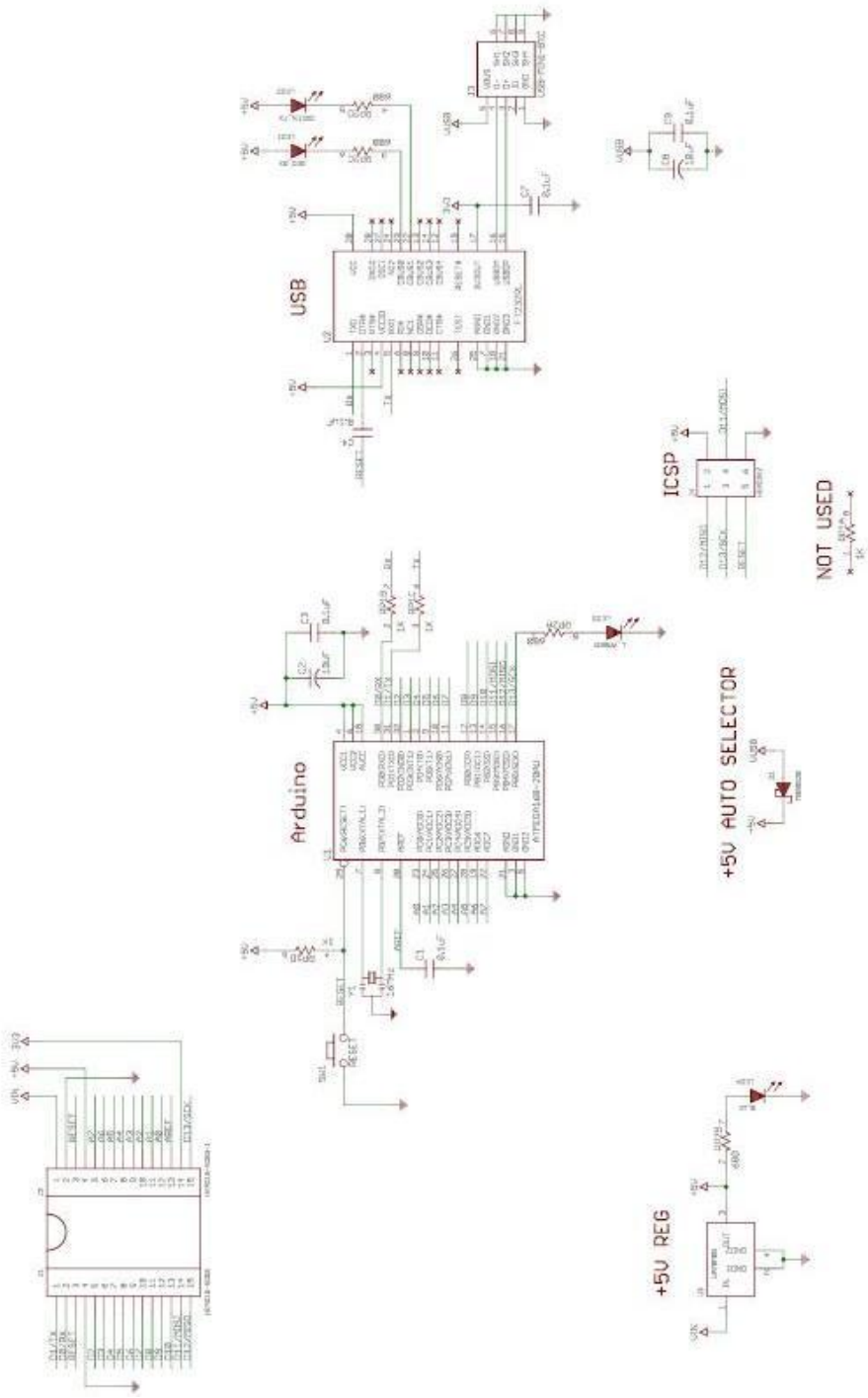
ANEXO A

CIRCUITOS ESQUEMÁTICOS DE LAS TARJETAS ARDUINO

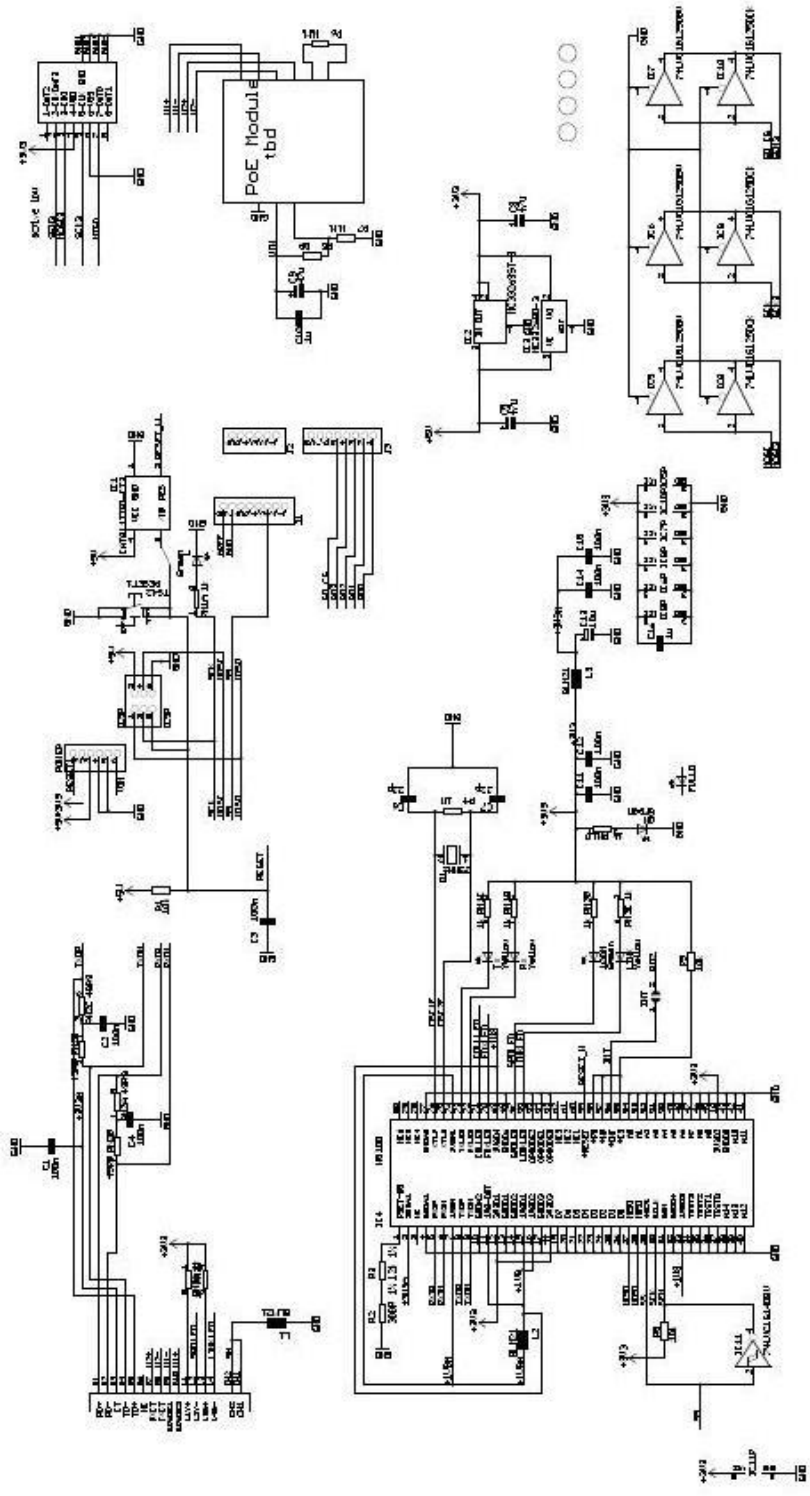
ANEXO A	Circuito esquemático de la tarjeta Arduino UNO	A – 1



ANEXO A	Circuito esquemático de la tarjeta Arduino Nano	A – 2



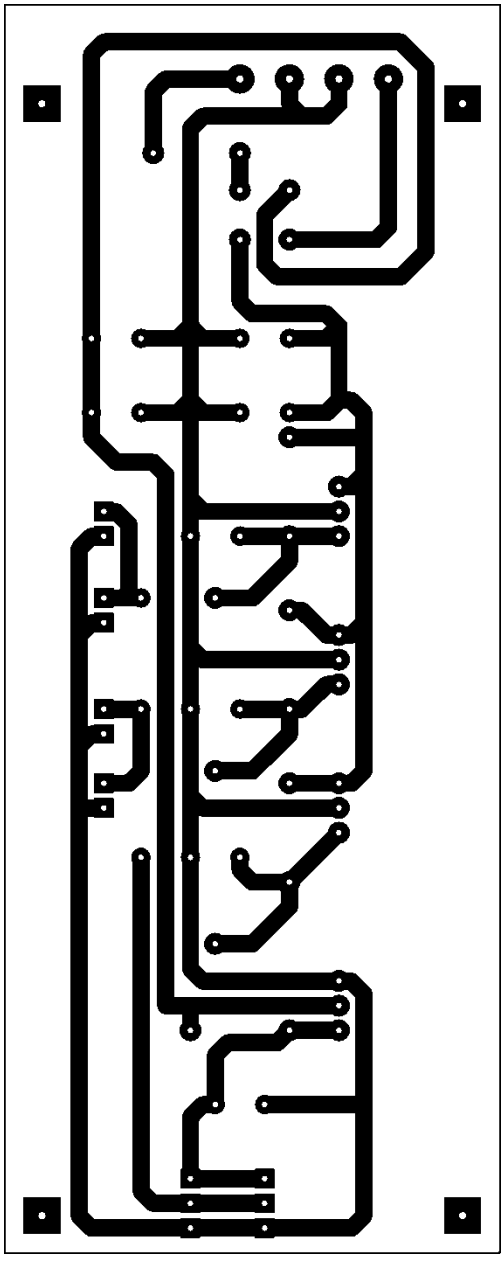
ANEXO A	Circuito esquemático de la tarjeta Arduino Ethernet Shield	A – 3



ANEXO B

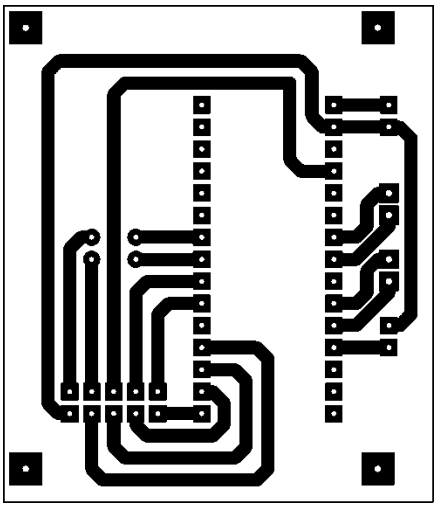
CIRCUITOS IMPRESOS

ANEXO B	Circuito impreso de la fuente de alimentación	B – 1

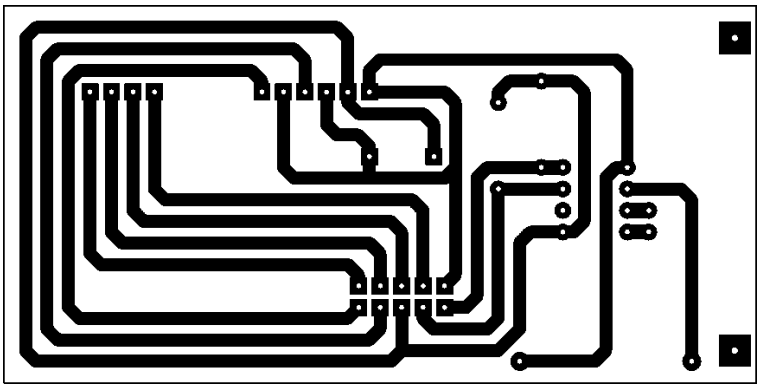


Circuito para la fuente de alimentación

ANEXO B	Circuitos impresos de la tarjeta Arduino Nano, RTC DS1307 y de la pantalla LCD	B – 2

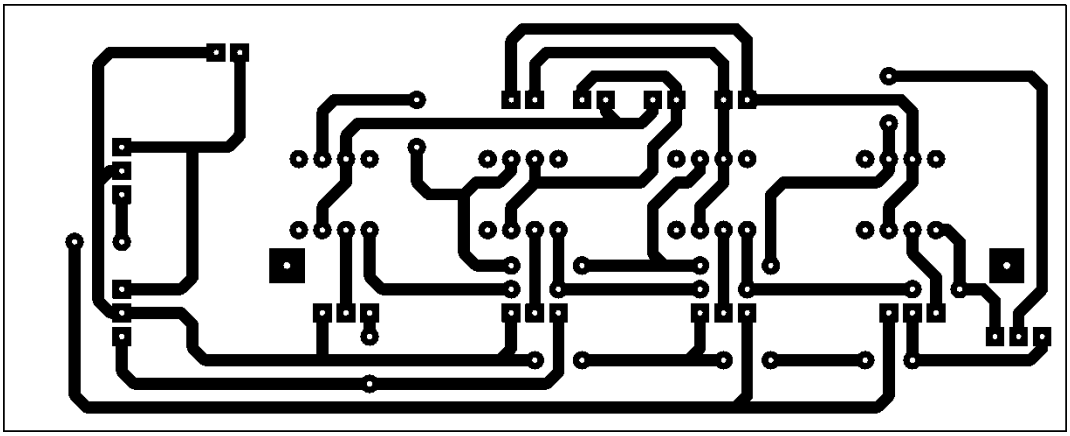


Circuito para la tarjeta Arduino Nano

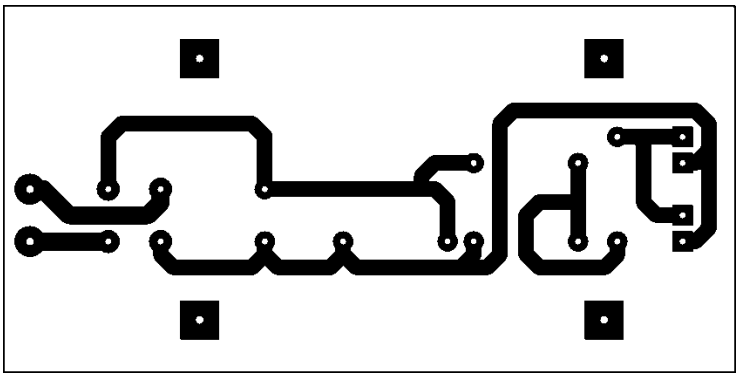


Circuito para el RTC DS1307 y pantalla LCD

ANEXO B	Circuitos impresos de los sensores de corriente y del sensor de voltaje	B – 3

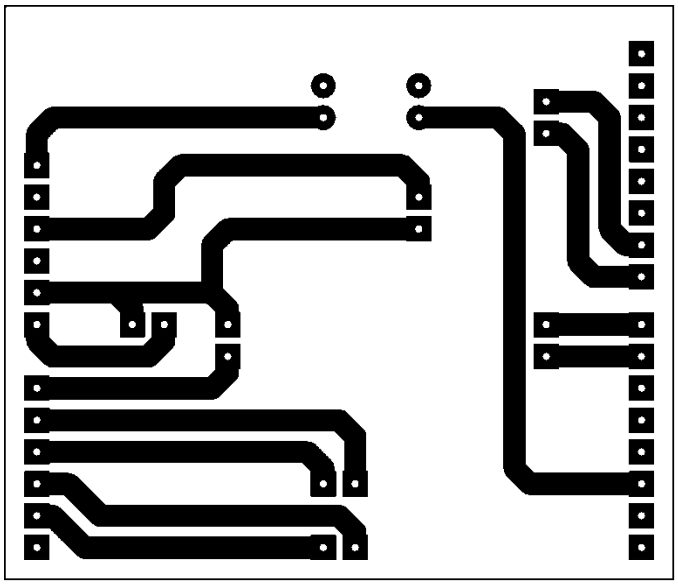


Circuito para los sensores de corriente

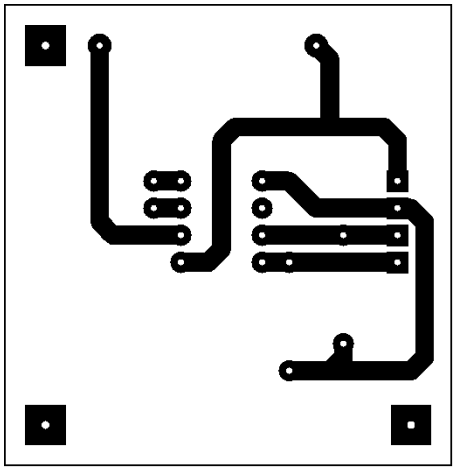


Circuito para el sensores de voltaje

ANEXO B	Circuitos impresos de la tarjeta Arduino UNO y RTC DS1307	B – 4



Circuito para la tarjeta Arduino UNO



Circuito para el RTC DS1307

ANEXO C

PROGRAMACIÓN DE LA TARJETA ARDUINO NANO

```
#include <LiquidCrystal.h>
#include <avr/wdt.h>
#include <DS1307.h>
DS1307 rtc(4, 5);
LiquidCrystal lcd(6, 7, 9, 10, 11, 12);
int voltPin = 0;
int corrPin = 1;
float kwh=0;
void setup()
{
wdt_disable();
lcd.begin(20, 4);
lcd.setCursor(5, 1);
lcd.print("Registrador");
lcd.setCursor(7, 2);
lcd.print("Arduino");
delay(3000);
rtc.halt(false);
rtc.setSQWRate(SQW_RATE_1);
rtc.enableSQW(true);
lcd.clear();
wdt_enable(WDTO_2S);
}
void loop()
{
float volt;
float voltaje;
float corr;
float corrTotal;
float watts;
float kwf;
volt = analogRead(voltPin);
corr = analogRead(corrPin);
if (volt>=75 || volt<517){
  voltaje = ((30*volt)+3300)/185;
  if (volt>=517){
    voltaje = ((5*volt)+815)/34;
  }
}
corrTotal = (45*corr)/1023;
watts = (voltaje*corrTotal)/1000;
kwf = watts/3600;
```

```
kwh = kwh + kwf;
delay(100);
lcd.setCursor(0, 0);
lcd.print("Voltaje");
lcd.setCursor(10, 0);
lcd.print(voltaje);
lcd.setCursor(0, 1);
lcd.print("Corriente");
lcd.setCursor(10, 1);
lcd.print(corrTotal);
lcd.setCursor(0, 2);
lcd.print("Kwh");
lcd.setCursor(10, 2);
lcd.print(kwh);
delay(150);
lcd.setCursor(0, 3);
lcd.print(rtc.getTimeStr());
lcd.setCursor(10, 3);
lcd.print(rtc.getDateStr());
delay (750);
wdt_reset();
}
```

ANEXO D

PROGRAMACIÓN DEL REGISTRADOR DE VARIABLES ELÉCTRICAS REMOTO

ANEXO D	Programación del registrador de variables eléctricas remoto	D – 1

```
#include <SdFat.h>
#include <SdFatUtil.h>
#define CHIP_SELECT SS_PIN
#define USE_DS1307       0
#define LOG_INTERVAL  1000
#define SENSOR_COUNT     3
#define ECHO_TO_SERIAL   1
#define WAIT_TO_START    1
#define ADC_DELAY       10
SdFat sd;
ofstream logfile;
ArduinoOutStream cout(Serial);
char buf[80];
#if SENSOR_COUNT > 6
#error SENSOR_COUNT too large
#endif
#define error(s) sd.errorHalt_P(PSTR(s))
#if USE_DS1307
#error remove this line and uncomment the next two lines.
RTC_DS1307 RTC;
void dateTime(uint16_t* date, uint16_t* time) {
    DateTime now = RTC.now();
  *date = FAT_DATE(now.year(), now.month(), now.day());
  *time = FAT_TIME(now.hour(), now.minute(), now.second());
}
ostream& operator << (ostream& os, DateTime& dt) {
  os << dt.year() << '/' << int(dt.month()) << '/' << int(dt.day()) << ',';
  os << int(dt.hour()) << ':' << setfill('0') << setw(2) << int(dt.minute());
  os << ':' << setw(2) << int(dt.second()) << setfill(' ');
  return os;
}
#endif
void setup() {
  Serial.begin(9600);
  cout << endl << pstr("FreeRam: ") << FreeRam() << endl;
#if WAIT_TO_START
  cout << pstr("Type any character to start\n");
  while (Serial.read() < 0) {}
#endif
#if USE_DS1307
```

```
  Wire.begin();
  if (!RTC.begin()) error("RTC failed");
  SdFile::dateTimeCallback(dateTime);
  DateTime now = RTC.now();
  cout  << now << endl;
#endif
  if (!sd.init(SPI_HALF_SPEED, CHIP_SELECT)) sd.initErrorHalt();
  char name[] = "LOGGER00.CSV";
  for (uint8_t i = 0; i < 100; i++) {
    name[6] = i/10 + '0';
    name[7] = i%10 + '0';
    if (sd.exists(name)) continue;
    logfile.open(name);
    break;
  }
  if (!logfile.is_open()) error("file.open");
  cout << pstr("Logging to: ") << name << endl;
  obufstream bout(buf, sizeof(buf));
  bout << pstr("millis");
#if USE_DS1307
  bout << pstr(",date,time");
#endif
  for (uint8_t i = 0; i < SENSOR_COUNT; i++) {
    bout << pstr(",sens") << int(i);
  }
  logfile << buf << endl;
#if ECHO_TO_SERIAL
  cout << buf << endl;
#endif
}
void loop() {
  uint32_t m;
  do {
    m = millis();
  } while (m % LOG_INTERVAL);
  obufstream bout(buf, sizeof(buf));
  bout << m;
#if USE_DS1307
  DateTime now = RTC.now();
  bout << ',' << now;
#endif
  for (uint8_t ia = 0; ia < SENSOR_COUNT; ia++) {
#if ADC_DELAY
    analogRead(ia);
```

```
    delay(ADC_DELAY);
#endif
    bout << ',' << analogRead(ia);
  }
  bout << endl;
  logfile << buf << flush;
  if (!logfile) error("write data failed");
#if ECHO_TO_SERIAL
  cout << buf;
#endif
  if (m == millis()) delay(1);
}
```

ANEXO E

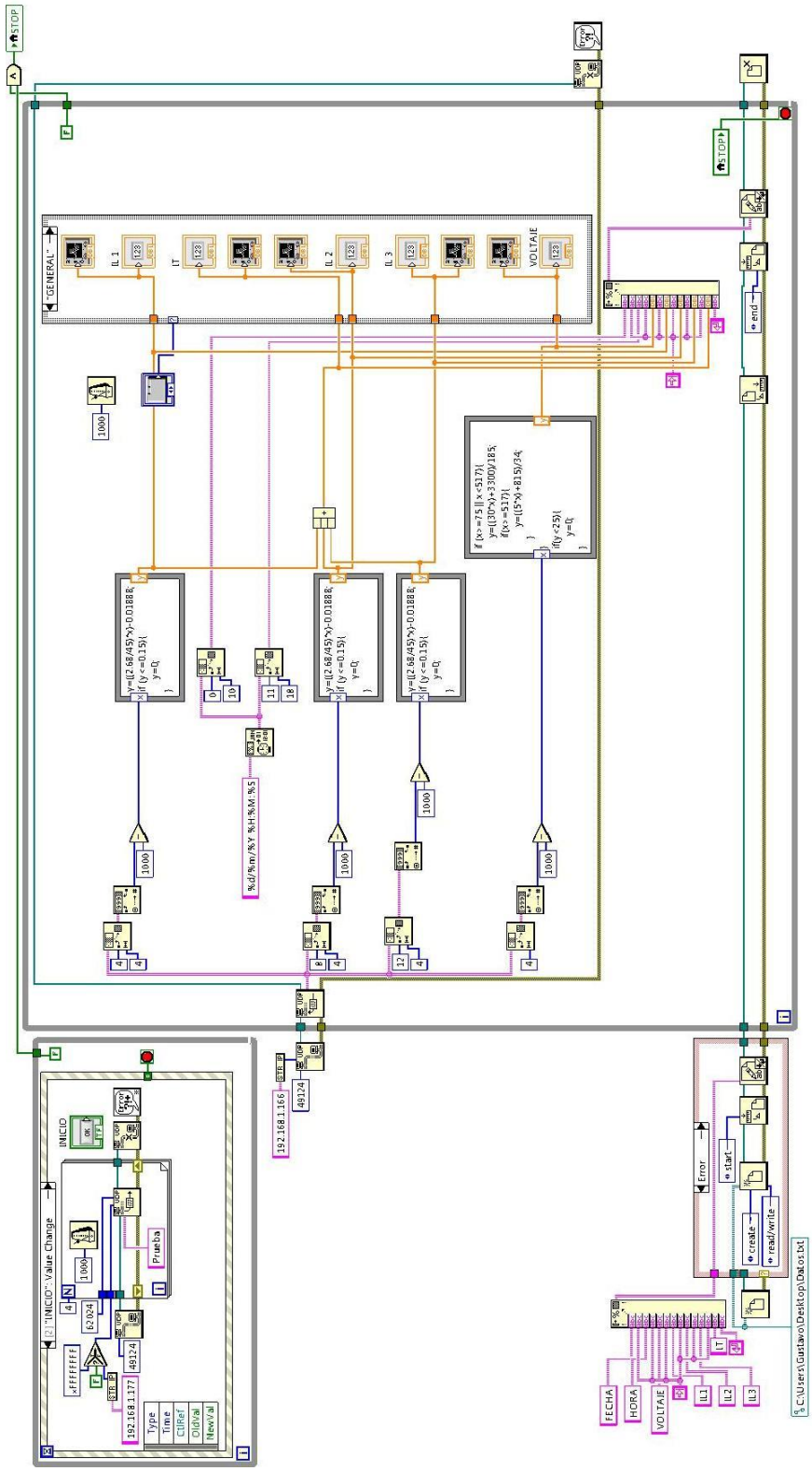
PROGRAMACIÓN DEL REGISTRADOR DE VARIABLES ELÉCTRICAS CON SUPERVISIÓN HMI DISEÑADA EN LABVIEW

ANEXO E	Programación diseñada en el software de Arduino	E – 1

```
#include <SPI.h>
#include <Ethernet.h>
#include <EthernetUdp.h>
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192, 168, 1, 177);
unsigned int localPort = 8888;
char packetBuffer[UDP_TX_PACKET_MAX_SIZE];
char  ReplyBuffer[] = "acknowledged";
EthernetUDP Udp;
void setup() {
  Ethernet.begin(mac,ip);
  Udp.begin(localPort);
  Serial.begin(9600);
}
void loop() {
  int packetSize = Udp.parsePacket();
  if(packetSize)
  {
    Serial.print("Received packet of size ");
    Serial.println(packetSize);
    Serial.print("From ");
    IPAddress remote = Udp.remoteIP();
    for (int i =0; i < 4; i++)
    {
      Serial.print(remote[i], DEC);
      if (i < 3)
      {
        Serial.print(".");
      }
    }
    Serial.print(", port ");
    Serial.println(Udp.remotePort());
    Udp.read(packetBuffer,UDP_TX_PACKET_MAX_SIZE);
    Serial.println("Contents:");
    Serial.println(packetBuffer);
    Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
    Udp.write(ReplyBuffer);
    Udp.endPacket();
  }
```

```
  delay(10);
}
```

ANEXO E	Programación diseñada en LabVIEW	E – 2



ANEXO F

PROGRAMACIÓN DEL REGISTRADOR DE VARIABLES ELÉCTRICAS REMOTO CON MONITOREO A TRAVÉS DE UN NAVEGADOR WEB

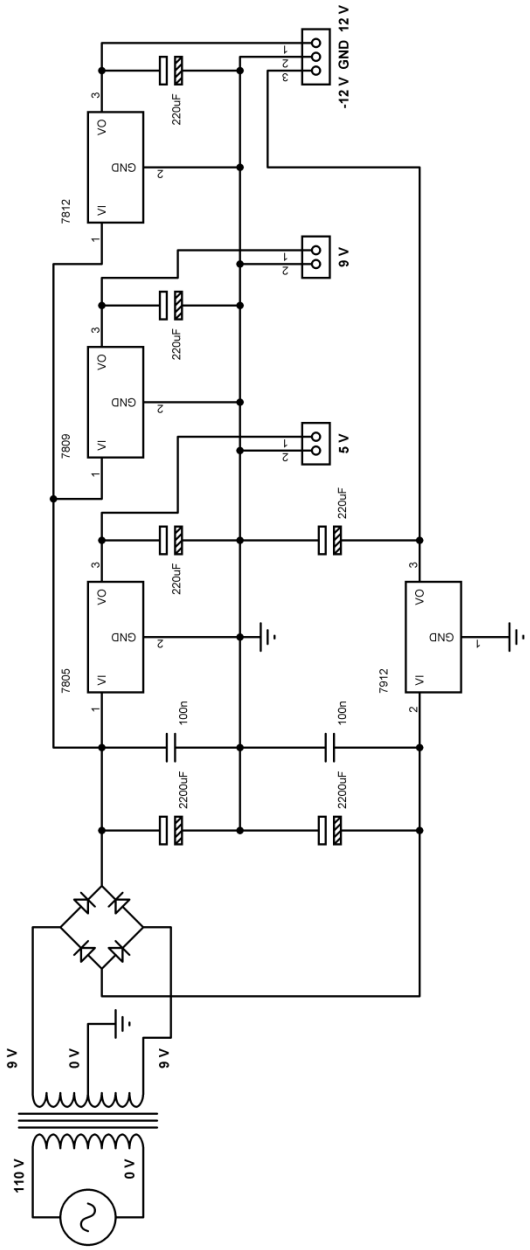
```
#include <SPI.h>
#include <Ethernet.h>
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,1, 177);
EthernetServer server(80);
void setup()
{
  Ethernet.begin(mac, ip);
  server.begin();
}
void loop()
{
  EthernetClient client = server.available();
  if (client) {
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        if (c == '\n' && currentLineIsBlank) {
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println();
          for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
            client.print("analog input ");
            client.print(analogChannel);
            client.print(" is ");
            client.print(analogRead(analogChannel));
            client.println("<br />");
          }
          break;
        }
        if (c == '\n') {
          currentLineIsBlank = true;
        }
        else if (c != '\r') {
          currentLineIsBlank = false;
        }
      }
    }
```

```
    delay(1);
    client.stop();
  }
}
```

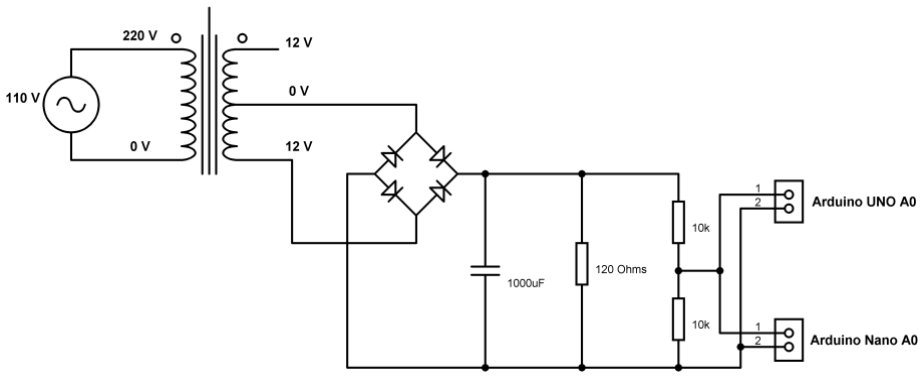
ANEXO G

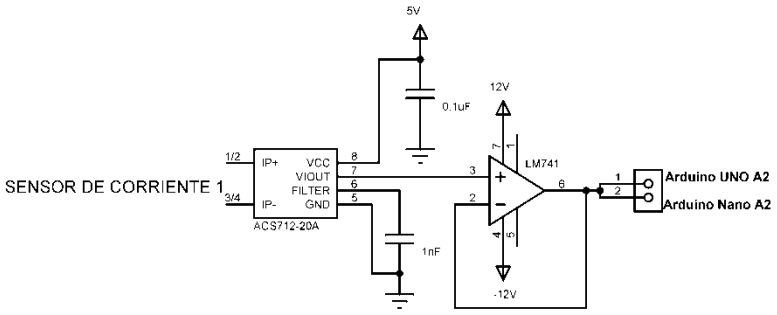
CIRCUITOS ESQUEMÁTICOS DEL REGISTRADOR DE VARIABLES ELÉCTRICAS

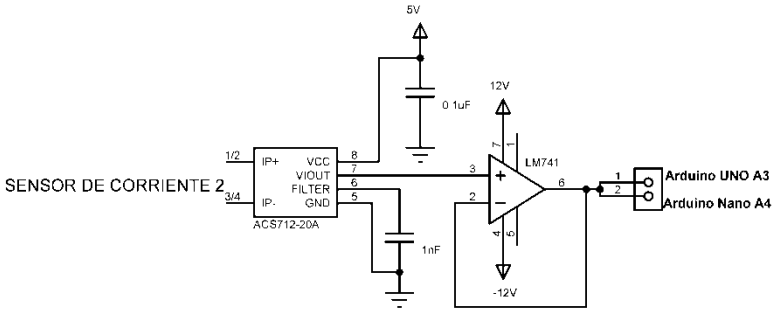
ANEXO G	Circuito esquemático de la fuente de alimentación	G – 1

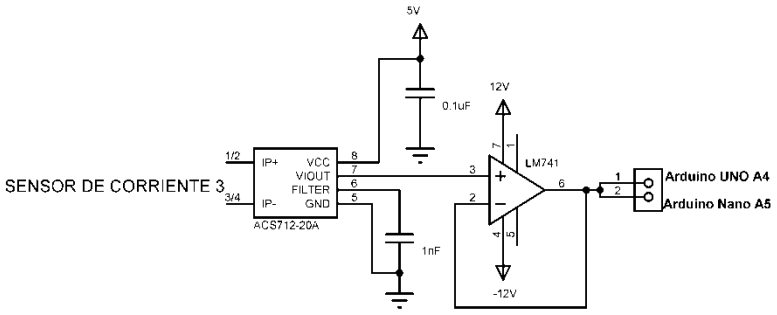


ANEXO G	Circuito esquemático del sensor de voltaje y de los sensores de corriente	G – 2

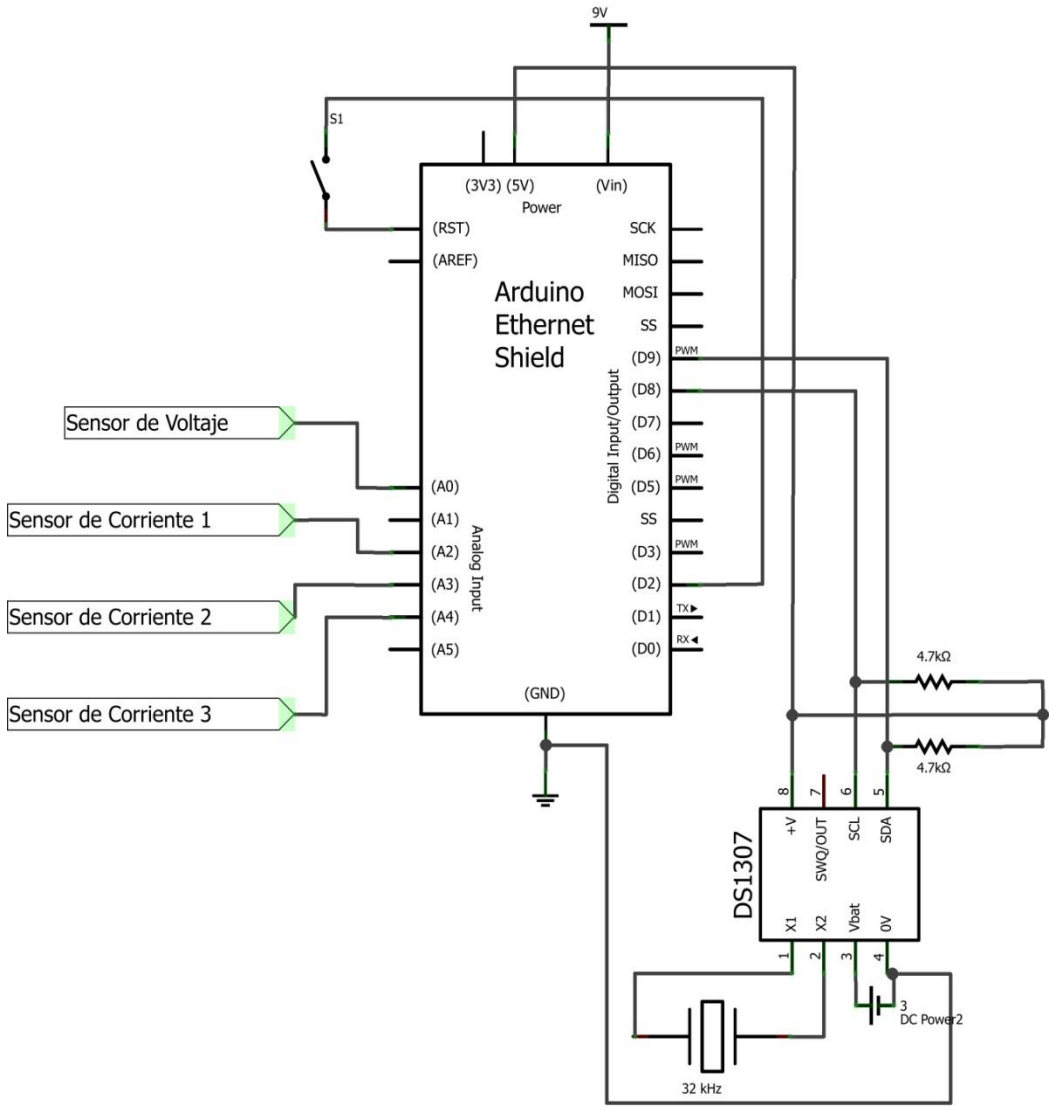




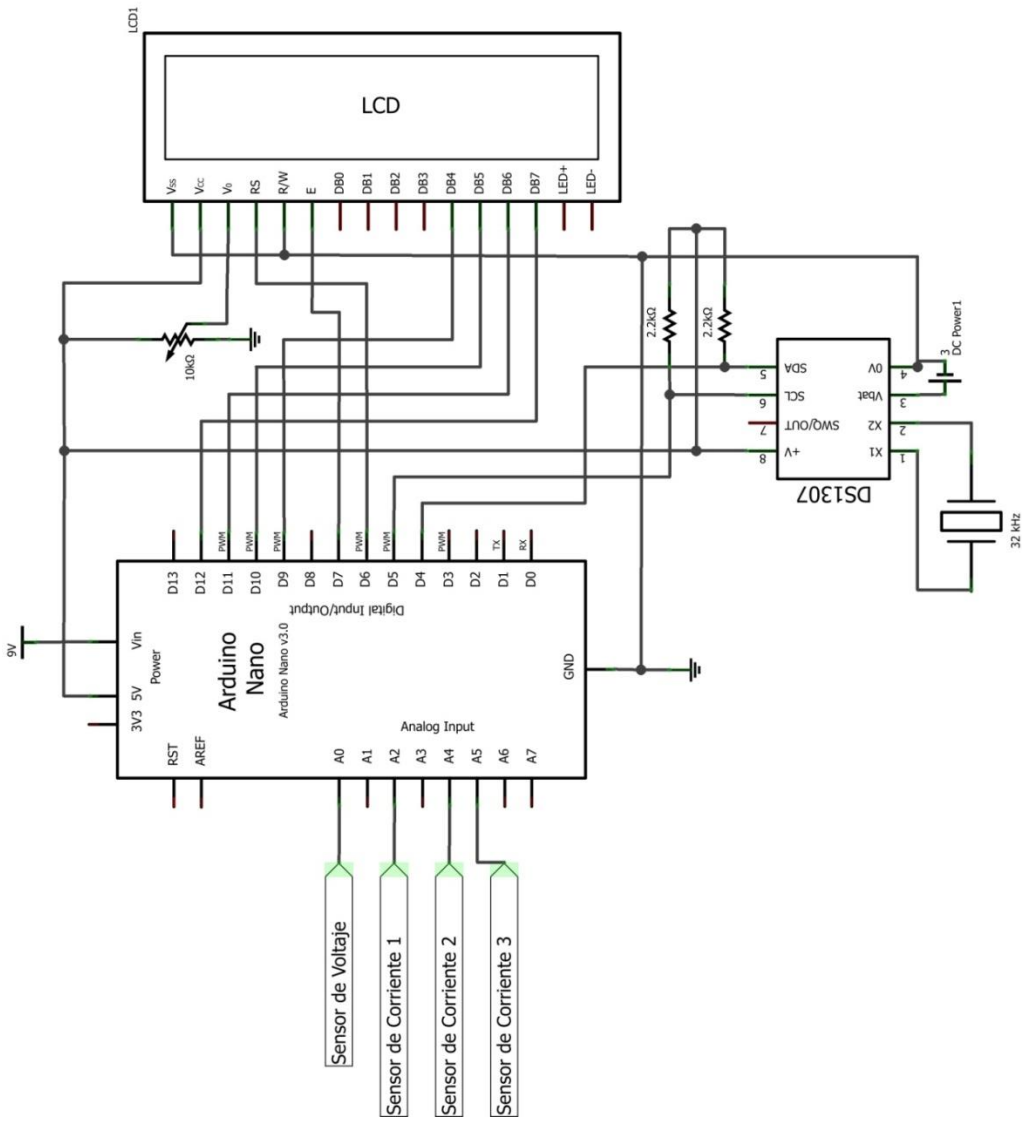




ANEXO G	Circuito esquemático de la conexión en la tarjeta Arduino Ethernet Shield	G – 3



ANEXO G	Circuito esquemático de la conexión en la tarjeta Arduino Nano	G – 4



ANEXO H

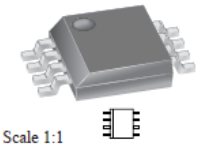
HOJAS TÉCNICAS

Features and Benefits

- Low-noise analog signal path
- Device bandwidth is set via the new FILTER pin
- 5 µs output rise time in response to step input current
- 80 kHz bandwidth
- Total output error 1.5% at T_A = 25°C
- Small footprint, low-profile SOIC8 package
- 1.2 mΩ internal conductor resistance
- 2.1 kVRMS minimum isolation voltage from pins 1-4 to pins 5-8
- 5.0 V, single supply operation
- 66 to 185 mV/A output sensitivity
- Output voltage proportional to AC or DC currents
- Factory-trimmed for accuracy
- Extremely stable output offset voltage
- Nearly zero magnetic hysteresis
- Ratiometric output from supply voltage

TÜV America
Certificate Number:
U8V 06 05 54214 010

Package: 8 Lead SOIC (suffix LC)



Approximate Scale 1:1

Description

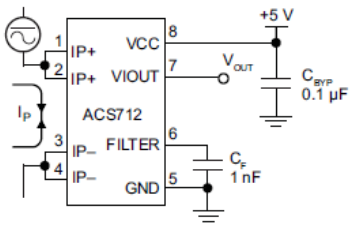
The Allegro® ACS712 provides economical and precise solutions for AC or DC current sensing in industrial, commercial, and communications systems. The device package allows for easy implementation by the customer. Typical applications include motor control, load detection and management, switch-mode power supplies, and overcurrent fault protection. The device is not intended for automotive applications.

The device consists of a precise, low-offset, linear Hall circuit with a copper conduction path located near the surface of the die. Applied current flowing through this copper conduction path generates a magnetic field which the Hall IC converts into a proportional voltage. Device accuracy is optimized through the close proximity of the magnetic signal to the Hall transducer. A precise, proportional voltage is provided by the low-offset, chopper-stabilized BiCMOS Hall IC, which is programmed for accuracy after packaging.

The output of the device has a positive slope (>$V_{IOUT(Q)}$) when an increasing current flows through the primary copper conduction path (from pins 1 and 2, to pins 3 and 4), which is the path used for current sampling. The internal resistance of this conductive path is 1.2 mΩ typical, providing low power loss. The thickness of the copper conductor allows survival of

Continued on the next page...

Typical Application



Application 1. The ACS712 outputs an analog signal, V_{OUT}, that varies linearly with the uni- or bi-directional AC or DC primary sampled current, I_P, within the range specified. C_F is recommended for noise management, with values that depend on the application.

Description (continued)

the device at up to 5× overcurrent conditions. The terminals of the conductive path are electrically isolated from the signal leads (pins 5 through 8). This allows the ACS712 to be used in applications requiring electrical isolation without the use of opto-isolators or other costly isolation techniques.

The ACS712 is provided in a small, surface mount SOIC8 p
The leadframe is plated with 100% matte tin, which is con
with standard lead (Pb) free printed circuit board assembly pr
Internally, the device is Pb-free, except for flip-chip high-temp
Pb-based solder balls, currently exempt from RoHS. The d
fully calibrated prior to shipment from the factory.

Selection Guide

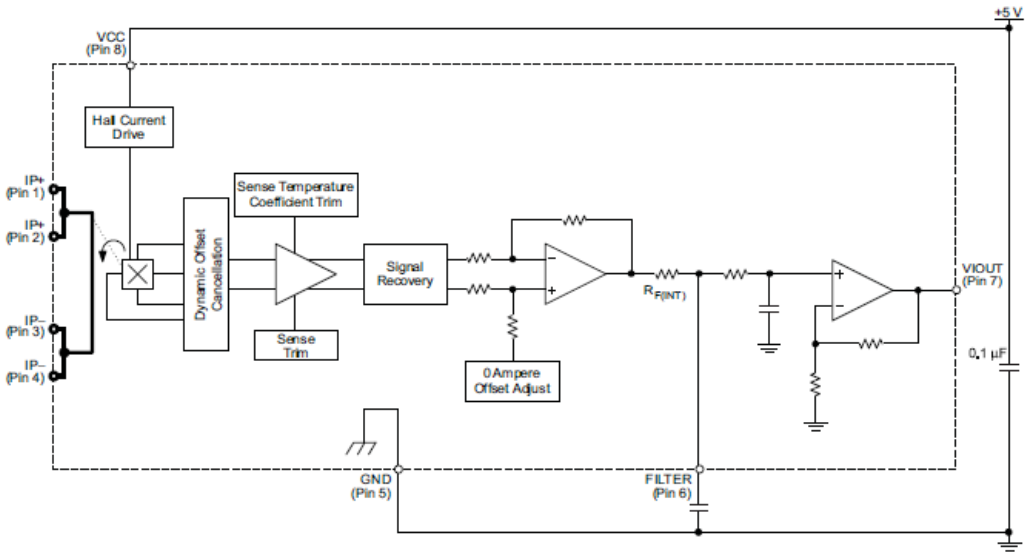
Part Number	Packing*	T_A (°C)	Optimized Range, I_P (A)	Sensitivity, Sens (Typ) (mV/A)
ACS712ELCTR-05B-T	Tape and reel, 3000 pieces/reel	–40 to 85	±5	185
ACS712ELCTR-20A-T	Tape and reel, 3000 pieces/reel	–40 to 85	±20	100
ACS712ELCTR-30A-T	Tape and reel, 3000 pieces/reel	–40 to 85	±30	66

*Contact Allegro for additional packing options.

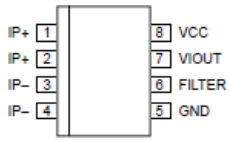
Absolute Maximum Ratings

Characteristic	Symbol	Notes	Rating	Units
Supply Voltage	V_{CC}		8	V
Reverse Supply Voltage	V_{RCC}		–0.1	V
Output Voltage	V_{IOUT}		8	V
Reverse Output Voltage	V_{RIOUT}		–0.1	V
Reinforced Isolation Voltage	V_{ISO}	Pins 1-4 and 5-8; 60 Hz, 1 minute, T_A=25°C	2100	VAC
		Maximum working voltage according to UL60950-1	184	V_{peak}
Basic Isolation Voltage	$V_{ISO(bsc)}$	Pins 1-4 and 5-8; 60 Hz, 1 minute, T_A=25°C	1500	VAC
		Maximum working voltage according to UL60950-1	354	V_{peak}
Output Current Source	$I_{IOUT(Source)}$		3	mA
Output Current Sink	$I_{IOUT(Sink)}$		10	mA
Overcurrent Transient Tolerance	I_P	1 pulse, 100 ms	100	A
Nominal Operating Ambient Temperature	T_A	Range E	–40 to 85	°C
Maximum Junction Temperature	T_J(max)		165	°C
Storage Temperature	T_{stg}		–65 to 170	°C

Functional Block Diagram



Pin-out Diagram



Terminal List Table

Number	Name	Description
1 and 2	IP+	Terminals for current being sampled; fused internally
3 and 4	IP–	Terminals for current being sampled; fused internally
5	GND	Signal ground terminal
6	FILTER	Terminal for external capacitor that sets bandwidth
7	VIOUT	Analog output signal
8	VCC	Device power supply terminal

COMMON OPERATING CHARACTERISTICS[1] over full range of T_A, C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
ELECTRICAL CHARACTERISTICS						
Supply Voltage	V_{CC}		4.5	5.0	5.5	V
Supply Current	I_{CC}	V_{CC} = 5.0 V, output open	–	10	13	mA
Output Capacitance Load	C_{LOAD}	VIOUT to GND	–	–	10	nF
Output Resistive Load	R_{LOAD}	VIOUT to GND	4.7	–	–	kΩ
Primary Conductor Resistance	$R_{PRIMARY}$	T_A = 25°C	–	1.2	–	mΩ
Rise Time	t_r	I_P = I_P(max), T_A = 25°C, C_{OUT} = open	–	5	–	µs
Frequency Bandwidth	f	–3 dB, T_A = 25°C; I_P is 10 A peak-to-peak	–	80	–	kHz
Nonlinearity	E_{LIN}	Over full range of I_P	–	1.5	–	%
Symmetry	E_{SYM}	Over full range of I_P	98	100	102	%
Zero Current Output Voltage	$V_{IOUT(Q)}$	Bidirectional; I_P = 0 A, T_A = 25°C	–	V_{CC} × 0.5	–	V
Power-On Time	t_{PO}	Output reaches 90% of steady-state level, T_J = 25°C, 20 A present on leadframe	–	35	–	µs
Magnetic Coupling[2]			–	12	–	G/A
Internal Filter Resistance[3]	$R_{F(INT)}$			1.7		kΩ

[1]Device may be operated at higher primary current levels, I_P, and ambient, T_A, and internal leadframe temperatures, T_A, provided that the Maximum Junction Temperature, T_J(max), is not exceeded.

[2]1G = 0.1 mT.

[3]$R_{F(INT)}$ forms an RC circuit via the FILTER pin.

COMMON THERMAL CHARACTERISTICS[1]

			Min.	Typ.	Max.	Units
Operating Internal Leadframe Temperature	T_A	E range	–40	–	85	°C
					Value	Units
Junction-to-Lead Thermal Resistance[2]	$R_{\theta JL}$	Mounted on the Allegro ASEK 712 evaluation board			5	°C/W
Junction-to-Ambient Thermal Resistance	$R_{\theta JA}$	Mounted on the Allegro 85-0322 evaluation board, includes the power consumed by the board			23	°C/W

[1]Additional thermal information is available on the Allegro website.

[2]The Allegro evaluation board has 1500 mm^2 of 2 oz. copper on each side, connected to pins 1 and 2, and to pins 3 and 4, with thermal vias connecting the layers. Performance values include the power consumed by the PCB. Further details on the board are available from the Frequently Asked Questions document on our website. Further information about board design and thermal performance also can be found in the Applications Information section of this datasheet.

x05B PERFORMANCE CHARACTERISTICS[1] T_A = –40°C to 85°C, C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
Optimized Accuracy Range	I_P		–5	–	5	A
Sensitivity	Sens	Over full range of I_P, T_A = 25°C	180	185	190	mV/A
Noise	$V_{NOISE(PP)}$	Peak-to-peak, T_A = 25°C, 185 mV/A programmed Sensitivity, C_F = 47 nF, C_{OUT} = open, 2 kHz bandwidth	–	21	–	mV
Zero Current Output Slope	$\Delta I_{OUT(Q)}$	T_A = –40°C to 25°C	–	–0.26	–	mV/°C
		T_A = 25°C to 150°C	–	–0.08	–	mV/°C
Sensitivity Slope	ΔSens	T_A = –40°C to 25°C	–	0.054	–	mV/A/°C
		T_A = 25°C to 150°C	–	–0.008	–	mV/A/°C
Total Output Error[2]	E_{TOT}	I_P =±5 A, T_A = 25°C	–	±1.5	–	%

[1]Device may be operated at higher primary current levels, I_P, and ambient temperatures, T_A, provided that the Maximum Junction Temperature, $T_{J(max)}$, is not exceeded.

[2]Percentage of I_P, with I_P = 5 A. Output filtered.

x20A PERFORMANCE CHARACTERISTICS[1] T_A = –40°C to 85°C, C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
Optimized Accuracy Range	I_P		–20	–	20	A
Sensitivity	Sens	Over full range of I_P, T_A = 25°C	96	100	104	mV/A
Noise	$V_{NOISE(PP)}$	Peak-to-peak, T_A = 25°C, 100 mV/A programmed Sensitivity, C_F = 47 nF, C_{OUT} = open, 2 kHz bandwidth	–	11	–	mV
Zero Current Output Slope	$\Delta I_{OUT(Q)}$	T_A = –40°C to 25°C	–	–0.34	–	mV/°C
		T_A = 25°C to 150°C	–	–0.07	–	mV/°C
Sensitivity Slope	ΔSens	T_A = –40°C to 25°C	–	0.017	–	mV/A/°C
		T_A = 25°C to 150°C	–	–0.004	–	mV/A/°C
Total Output Error[2]	E_{TOT}	I_P =±20 A, T_A = 25°C	–	±1.5	–	%

[1]Device may be operated at higher primary current levels, I_P, and ambient temperatures, T_A, provided that the Maximum Junction Temperature, T_J(max), is not exceeded.

[2]Percentage of I_P, with I_P = 20 A. Output filtered.

x30A PERFORMANCE CHARACTERISTICS[1] T_A = –40°C to 85°C, C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
Optimized Accuracy Range	I_P		–30	–	30	A
Sensitivity	Sens	Over full range of I_P, T_A = 25°C	63	66	69	mV/A
Noise	$V_{NOISE(PP)}$	Peak-to-peak, T_A = 25°C, 66 mV/A programmed Sensitivity, C_F = 47 nF, C_{OUT} = open, 2 kHz bandwidth	–	7	–	mV
Zero Current Output Slope	$\Delta I_{OUT(Q)}$	T_A = –40°C to 25°C	–	–0.35	–	mV/°C
		T_A = 25°C to 150°C	–	–0.08	–	mV/°C
Sensitivity Slope	ΔSens	T_A = –40°C to 25°C	–	0.007	–	mV/A/°C
		T_A = 25°C to 150°C	–	–0.002	–	mV/A/°C
Total Output Error[2]	E_{TOT}	I_P = ±30 A, T_A = 25°C	–	±1.5	–	%

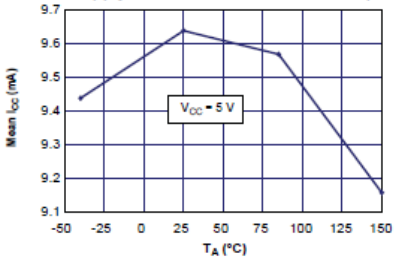
[1]Device may be operated at higher primary current levels, I_P, and ambient temperatures, T_A, provided that the Maximum Junction Temperature, T_J(max), is not exceeded.

[2]Percentage of I_P, with I_P = 30 A. Output filtered.

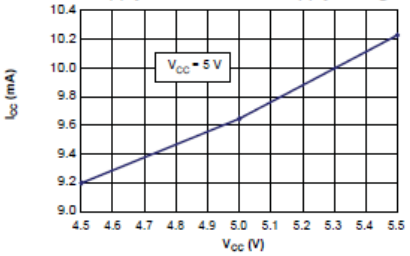
Characteristic Performance

I_P = 20 A, unless otherwise specified

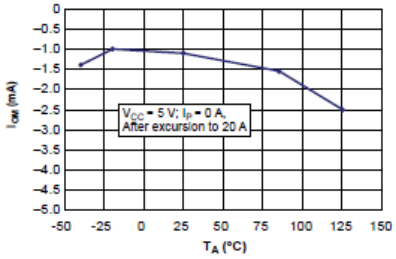
Mean Supply Current versus Ambient Temperature



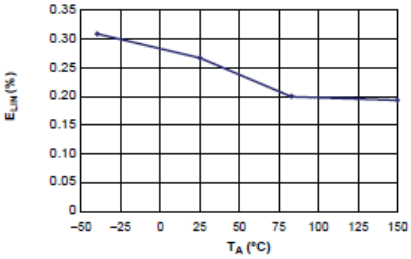
Supply Current versus Supply Voltage



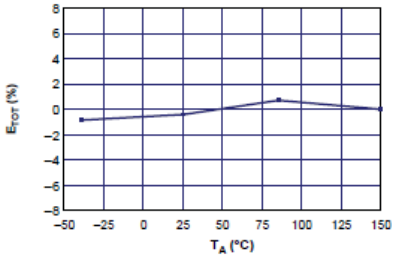
Magnetic Offset versus Ambient Temperature



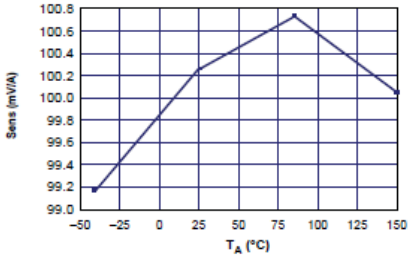
Nonlinearity versus Ambient Temperature



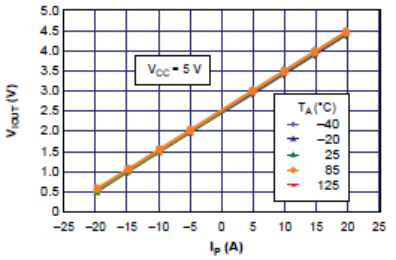
Mean Total Output Error versus Ambient Temperature



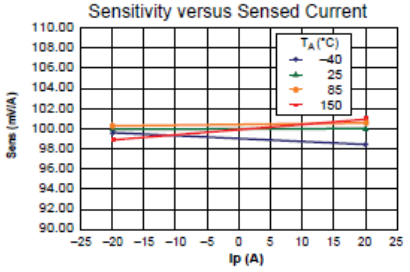
Sensitivity versus Ambient Temperature



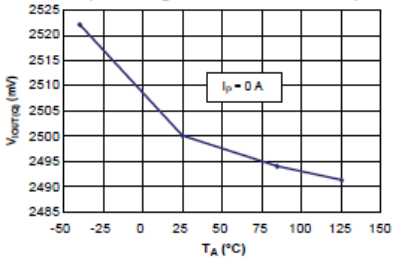
Output Voltage versus Sensed Current



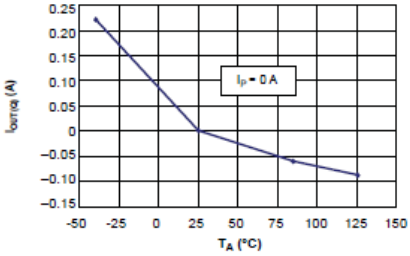
Sensitivity versus Sensed Current



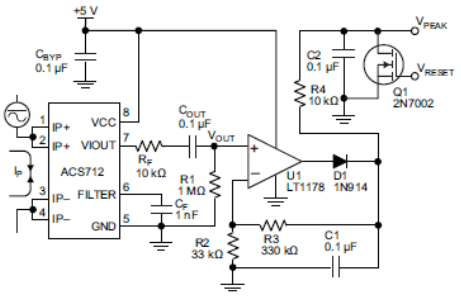
0 A Output Voltage versus Ambient Temperature



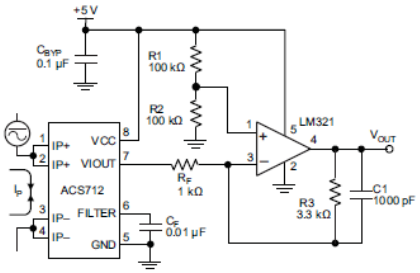
0 A Output Voltage Current versus Ambient Temperature



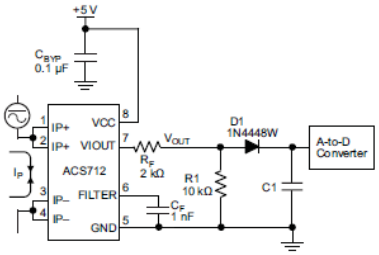
Typical Applications



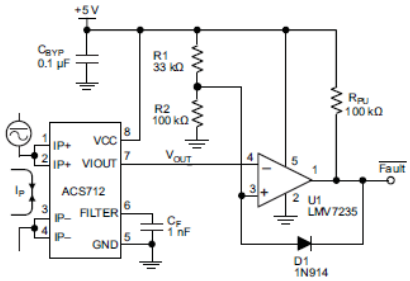
Application 2. Peak Detecting Circuit



Application 3. This configuration increases gain to 610 mV/A (tested using the ACS712ELC-05A).



Application 4. Rectified Output. 3.3 V scaling and rectification application for A-to-D converters. Replaces current transformer solutions with simpler ACS circuit. C1 is a function of the load resistance and filtering desired. R1 can be omitted if the full range is desired.



Application 5. 10 A Overcurrent Fault Latch. Fault threshold set by R1 and R2. This circuit latches an overcurrent fault and holds it until the 5 V rail is powered down.

DALLAS SEMICONDUCTOR

DS1307
64 x 8 Serial Real-Time Clock

www.maxim-ic.com

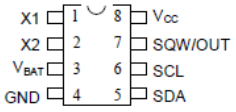
FEATURES

- Real-time clock (RTC) counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap-year compensation valid up to 2100
- 56-byte, battery-backed, nonvolatile (NV) RAM for data storage
- Two-wire serial interface
- Programmable squarewave output signal
- Automatic power-fail detect and switch circuitry
- Consumes less than 500nA in battery backup mode with oscillator running
- Optional industrial temperature range: -40°C to +85°C
- Available in 8-pin DIP or SOIC
- Underwriters Laboratory (UL) recognized

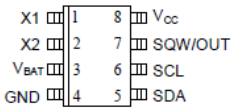
ORDERING INFORMATION

DS1307	8-Pin DIP (300-mil)
DS1307Z	8-Pin SOIC (150-mil)
DS1307N	8-Pin DIP (Industrial)
DS1307ZN	8-Pin SOIC (Industrial)

PIN ASSIGNMENT



DS1307 8-Pin DIP (300-mil)



DS1307 8-Pin SOIC (150-mil)

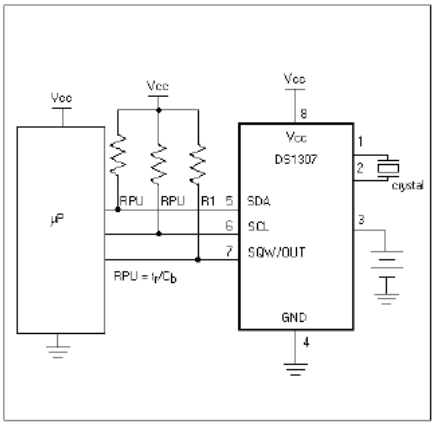
PIN DESCRIPTION

V_{CC}	- Primary Power Supply
X1, X2	- 32.768kHz Crystal Connection
V_{BAT}	- +3V Battery Input
GND	- Ground
SDA	- Serial Data
SCL	- Serial Clock
SQW/OUT	- Square Wave/Output Driver

DESCRIPTION

The DS1307 Serial Real-Time Clock is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially via a 2-wire, bi-directional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power sense circuit that detects power failures and automatically switches to the battery supply.

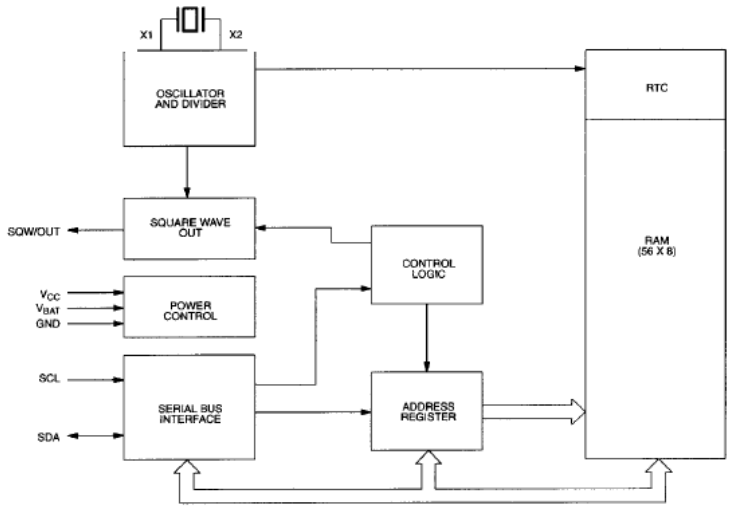
TYPICAL OPERATING CIRCUIT



OPERATION

The DS1307 operates as a slave device on the serial bus. Access is obtained by implementing a START condition and providing a device identification code followed by a register address. Subsequent registers can be accessed sequentially until a STOP condition is executed. When V_{CC} falls below 1.25 x V_{BAT} the device terminates an access in progress and resets the device address counter. Inputs to the device will not be recognized at this time to prevent erroneous data from being written to the device from an out of tolerance system. When V_{CC} falls below V_{BAT} the device switches into a low-current battery backup mode. Upon power-up, the device switches from battery to V_{CC} when V_{CC} is greater than V_{BAT} + 0.2V and recognizes inputs when V_{CC} is greater than 1.25 x V_{BAT}. The block diagram in Figure 1 shows the main elements of the serial RTC.

DS1307 BLOCK DIAGRAM Figure 1



SIGNAL DESCRIPTIONS

V_{CC}, GND – DC power is provided to the device on these pins. V_{CC} is the +5V input. When 5V is applied within normal limits, the device is fully accessible and data can be written and read. When a 3V battery is connected to the device and V_{CC} is below 1.25 x V_{BAT}, reads and writes are inhibited. However, the timekeeping function continues unaffected by the lower input voltage. As V_{CC} falls below V_{BAT} the RAM and timekeeper are switched over to the external power supply (nominal 3.0V DC) at V_{BAT}.

V_{BAT} – Battery input for any standard 3V lithium cell or other energy source. Battery voltage must be held between 2.0V and 3.5V for proper operation. The nominal write protect trip point voltage at which access to the RTC and user RAM is denied is set by the internal circuitry as 1.25 x V_{BAT} nominal. A lithium battery with 48mAhr or greater will back up the DS1307 for more than 10 years in the absence of power at 25°C. UL recognized to ensure against reverse charging current when used in conjunction with a lithium battery.

See "Conditions of Acceptability" at http://www.maxim-ic.com/TechSupport/QA/ntrl.htm.

SCL (Serial Clock Input) – SCL is used to synchronize data movement on the serial interface.

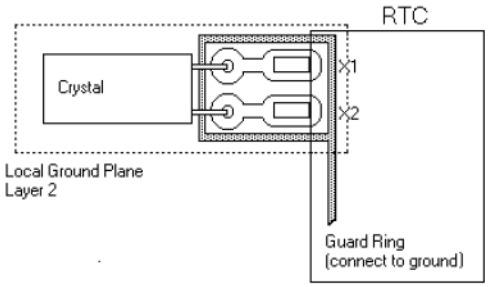
SDA (Serial Data Input/Output) – SDA is the input/output pin for the 2-wire serial interface. The SDA pin is open drain which requires an external pullup resistor.

SQW/OUT (Square Wave/Output Driver) – When enabled, the SQWE bit set to 1, the SQW/OUT pin outputs one of four square wave frequencies (1Hz, 4kHz, 8kHz, 32kHz). The SQW/OUT pin is open drain and requires an external pull-up resistor. SQW/OUT will operate with either Vcc or Vbat applied.

X1, X2 – Connections for a standard 32.768kHz quartz crystal. The internal oscillator circuitry is designed for operation with a crystal having a specified load capacitance (CL) of 12.5pF.

For more information on crystal selection and crystal layout considerations, please consult Application Note 58, "Crystal Considerations with Dallas Real-Time Clocks." The DS1307 can also be driven by an external 32.768kHz oscillator. In this configuration, the X1 pin is connected to the external oscillator signal and the X2 pin is floated.

RECOMMENDED LAYOUT FOR CRYSTAL



ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground	-0.5V to +7.0V
Storage Temperature	-55°C to +125°C
Soldering Temperature	260°C for 10 seconds DIP See JPC/JEDEC Standard J-STD-020A for Surface Mount Devices

* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

Range	Temperature	V_{CC}
Commercial	0°C to +70°C	4.5V to 5.5V V_{CC1}
Industrial	-40°C to +85°C	4.5V to 5.5V V_{CC1}

RECOMMENDED DC OPERATING CONDITIONS

(Over the operating range*)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	V_{CC}	4.5	5.0	5.5	V	
Logic 1	V_{IH}	2.2		V_{CC} + 0.3	V	
Logic 0	V_{IL}	-0.5		+0.8	V	
V_{BAT} Battery Voltage	V_{BAT}	2.0		3.5	V	

*Unless otherwise specified.

DC ELECTRICAL CHARACTERISTICS

(Over the operating range*)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Input Leakage (SCL)	I_{LI}			1	µA	
I/O Leakage (SDA & SQW/OUT)	I_{LO}			1	µA	
Logic 0 Output (I_{OL} = 5mA)	V_{OL}			0.4	V	
Active Supply Current	I_{CCA}			1.5	mA	7
Standby Current	I_{CCS}			200	µA	1
Battery Current (OSC ON); SQW/OUT OFF	I_{BAT1}		300	500	nA	2
Battery Current (OSC ON); SQW/OUT ON (32kHz)	I_{BAT2}		480	800	nA	
Power-Fail Voltage	V_{PF}	1.216 x V_{BAT}	1.25 x V_{BAT}	1.284 x V_{BAT}	V	8

*Unless otherwise specified.

AC ELECTRICAL CHARACTERISTICS

(Over the operating range*)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
SCL Clock Frequency	f_{SCL}	0		100	kHz	
Bus Free Time Between a STOP and START Condition	t_{BUF}	4.7			µs	
Hold Time (Repeated) START Condition	$t_{HD:STA}$	4.0			µs	3
LOW Period of SCL Clock	t_{LOW}	4.7			µs	
HIGH Period of SCL Clock	t_{HIGH}	4.0			µs	
Set-up Time for a Repeated START Condition	$t_{SU:STA}$	4.7			µs	
Data Hold Time	$t_{HD:DAT}$	0			µs	4,5
Data Set-up Time	$t_{SU:DAT}$	250			ns	
Rise Time of Both SDA and SCL Signals	t_R			1000	ns	
Fall Time of Both SDA and SCL Signals	t_F			300	ns	
Set-up Time for STOP Condition	$t_{SU:STO}$	4.7			µs	
Capacitive Load for each Bus Line	C_B			400	pF	6
I/O Capacitance (T_A = 25°C)	$C_{I/O}$		10		pF	
Crystal Specified Load Capacitance (T_A = 25°C)			12.5		pF	

*Unless otherwise specified.

NOTES:

1. I_{CCS} specified with V_{CC} = 5.0V and SDA, SCL = 5.0V.
2. V_{CC} = 0V, V_{BAT} = 3V.
3. After this period, the first clock pulse is generated.
4. A device must internally provide a hold time of at least 300ns for the SDA signal (referred to the V_{IHMIN} of the SCL signal) in order to bridge the undefined region of the falling edge of SCL.
5. The maximum $t_{HD:DAT}$ has only to be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal.
6. C_B – Total capacitance of one bus line in pF.
7. I_{CCA} – SCL clocking at max frequency = 100kHz.
8. V_{PF} measured at V_{BAT} = 3.0V.

ACM2004D SERIES LCD MODULE

1.0 MECHANICAL SPECS

1. Overall Module Size	98.0mm(W) x 60.0mm(H) x max 14.0mm(D) for LED backlight version 98.0mm(W) x 60.0mm(H) x max 9.5mm(D) for reflective version
2. Dot Size	0.55mm(W) x 0.55mm(H)
3. Dot Pitch	0.60mm(W) x 0.60mm(H)
4. Duty	1/16
5. Controller IC	KS0066
6. LC Fluid Options	TN, STN
7. Polarizer Options	Reflective, Transflective, Transmissive
8. Backlight Options	LED
9. Temperature Range Options	Standard(0°C ~ 50°C), Wide(-20°C ~ 70°C)

2.0 ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Min	Typ	Max	Unit
Operating temperature (Standard)	Top	0	-	50	°C
Storage temperature (Standard)	Tst	-10	-	60	°C
Operating temperature (Wide temperature)	Top	-20	-	70	°C
Storage temperature (Wide temperature)	Tst	-30	-	80	°C
Input voltage	Vin	Vss		Vdd	V
Supply voltage for logic	Vdd- Vss	2.7	-	5.5	V
Supply voltage for LCD drive	Vdd- Vo	3.0	4.6	6.5	V

3.0 ELECTRICAL CHARACTERISTICS

Item	Symbol	Condition	Min	Typ	Max	Unit
Input voltage (high)	Vih	H level	2.2	-	Vdd	V
Input voltage (low)	Vil	L level	0	-	0.6	V
Recommended LC Driving Voltage (Standard Temp)	Vdd - Vo	0° C	-	4.8	5.4	V
		25° C	4.2	4.6	-	
		50° C	3.9	4.3	-	
Recommended LC Driving Voltage (Wide Temp)	Vdd -Vo	-20° C	-	6.4	7.2	V
		0° C	-	4.8	-	
		50° C	-	4.3	-	
		70° C	3.7	4.2	-	
Power Supply Current	Idd	Vdd=5.0V, fosc=270kHz	-	0.5	1.0	mA
LED Power Supply Voltage	Vfled	R=6.8Ω	-	4.6	5.0	V
LED Power Supply Current	Ifled	R=6.8Ω	-	240	480	mA

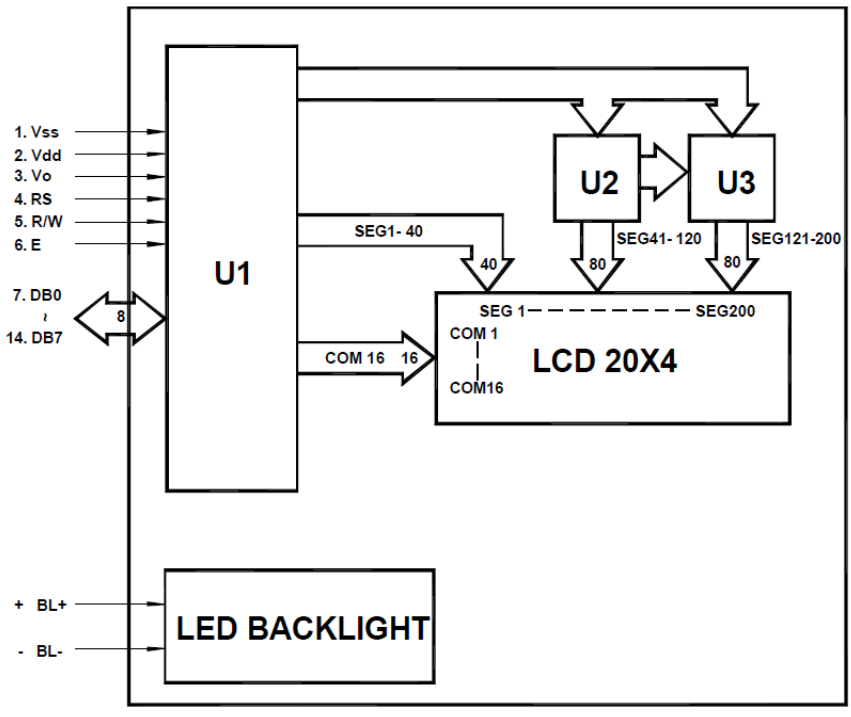
4.0 OPTICAL CHARACTERISTICS (Ta=25° C, Vdd= 5.0V±0.25V, TN LC fluid)

Item	Symbol	Condition	Min	Typ	Max	Unit
Viewing angle (horizontal)	θ	Cr ≥ 4.0	-25	-	-	deg
Viewing angle (vertical)	ϕ	Cr ≥ 4.0	-30	-	30	deg
Contrast Ratio	Cr	ϕ=0°, θ=0°	-	2	-	
Response time (rise)	Tr	ϕ=0°, θ=0°	-	120	150	ms
Response time (fall)	Tf	ϕ=0°, θ=0°	-	120	150	ms

4.1 OPTICAL CHARACTERISTICS (Ta=25° C, Vdd= 5.0V±0.25V, STN LC fluid)

Item	Symbol	Condition	Min	Typ	Max	Unit
Viewing angle (horizontal)	θ	Cr ≥ 2.0	-60	-	35	deg
Viewing angle (vertical)	ϕ	Cr ≥ 2.0	-40	-	40	deg
Contrast Ratio	Cr	ϕ=0°, θ=0°	-	6	-	
Response time (rise)	Tr	ϕ=0°, θ=0°	-	150	250	ms
Response time (fall)	Tf	ϕ=0°, θ=0°	-	150	250	ms

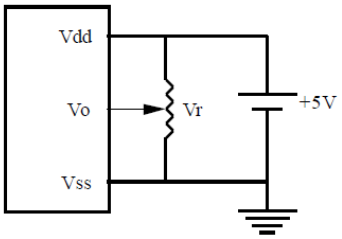
5.0 BLOCK DIAGRAM



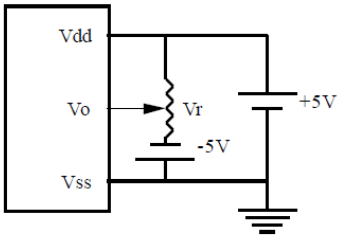
6.0 PIN ASSIGNMENT

Pin No.	Symbol	Function
1	Vss	Ground
2	Vdd	+5V
3	Vo	LCD contrast adjust
4	RS	Register select
5	R/W	Read / write
6	E	Enable
7	DB0	Data bit 0
8	DB1	Data bit 1
9	DB2	Data bit 2
10	DB3	Data bit 3
11	DB4	Data bit 4
12	DB5	Data bit 5
13	DB6	Data bit 6
14	DB7	Data bit 7
+	BL+	Power Supply for BL+
-	BL-	Power Supply for BL-

7.0 POWER SUPPLY



STANDARD TEMP RANGE



WIDE TEMP RANGE

Vr = 10KΩ ~ 20KΩ

8.0 TIMING CHARACTERISTICS

Item	Symbol	Test Condition	Min.	Typ.	Max.	Unit
Enable cycle time	t_C	Fig. a, Fig. b	500	-	-	ns
Enable pulse width	t_W	Fig. a, Fig. b	220	-	-	ns
Enable rise/fall time	t_R , t_F	Fig. a, Fig. b	-	-	25	ns
RS, R/W set up time	t_{SU}	Fig. a, Fig. b	40	-	-	ns
RS, R/W hold time	t_H	Fig. a, Fig. b	10	-	-	ns
Data delay time	t_D	Fig. b	-	-	120	ns
Data set up time	t_{DSU}	Fig. a	60	-	-	ns
Data hold time	t_{DH}	Fig. a, Fig. b	20	-	-	ns

ANEXO I

MANUAL DE USUARIO

Registrador de variables eléctricas con comunicación Ethernet

ÍNDICE

PERSPECTIVAS DEL REGISTRADOR DE VARIABLES ELÉCTRICAS

El registrador de variables eléctricas consta de varias conexiones que permiten al usuario monitorear y visualizar las mediciones realizadas durante su uso.

A continuación se muestra las conexiones a realizarse.

Parte superior:

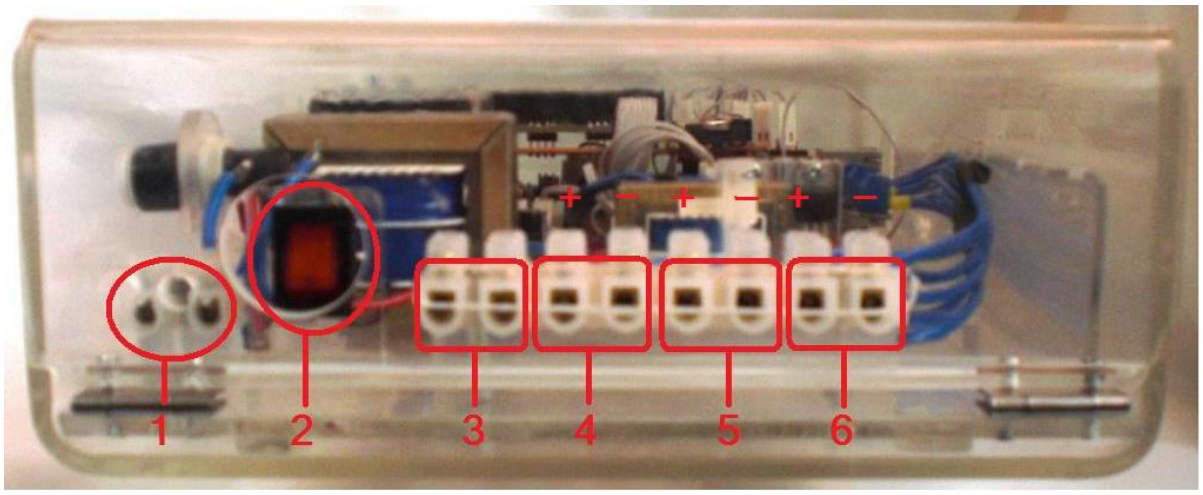


Figura 1: *Parte superior del registrador.*

1. Entrada para la alimentación 110 Vca.
2. Interruptor encendido / apagado.
3. Entrada para sensor de voltaje.
4. Entrada para el sensor de corriente IL3
5. Entrada para el sensor de corriente IL2
6. Entrada para el sensor de corriente IL1

Parte inferior:

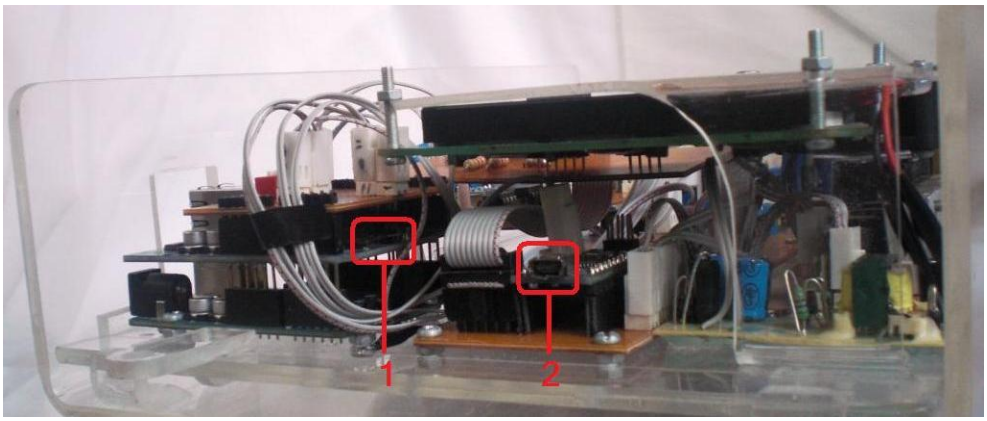


Figura 2: *Parte inferior del registrador.*

1. Entrada de la tarjeta de memoria SD.
2. Conector hembra USB tipo mini B para la comunicación PC a Arduino Nano

Parte lateral izquierda:

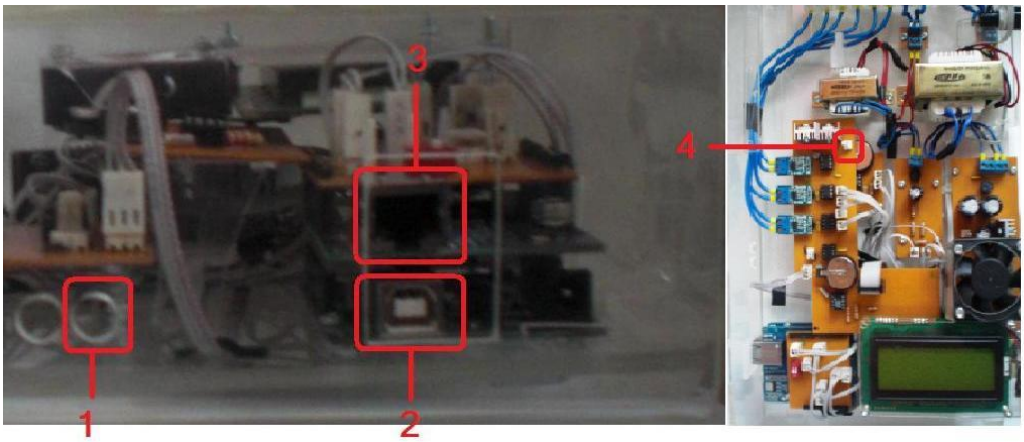


Figura 3: *Parte lateral izquierda del registrador.*

1. Entrada para el cable del adaptador de 9 Vcd
2. Conector hembra USB tipo B para la comunicación PC a Arduino UNO.

3. Conector hembra RJ-45 para la comunicación Ethernet.

4. Conector para el adaptador de 9 Vcd.

Parte lateral derecha:

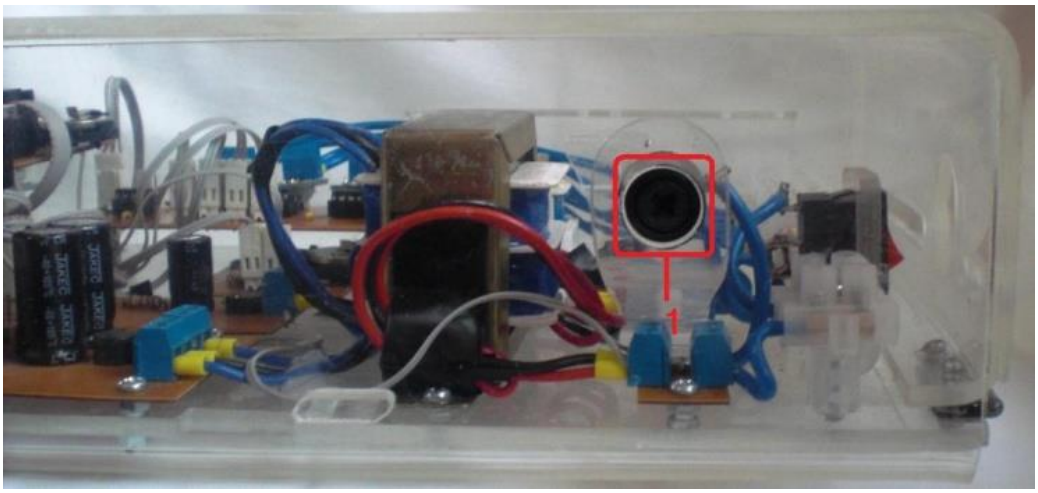


Figura 4: *Parte lateral derecha del registrador.*

1. Porta fusible.

CARACTERÍSTICAS TÉCNICAS DEL REGISTRADOR DE VARIABLES ELÉCTRICAS

Dimensiones: Largo: 34 cm.

Ancho: 21 cm.

Alto: 8.7 cm.

Voltaje de la alimentación principal: 120 Vca a 50/60 Hz.

Voltaje de la alimentación de los sensores de corriente: 9 Vcd.

Cable 12 AWG flexible para la conexión externa de los sensores de corriente y del sensor de voltaje.

CARGA DE CUALQUIER PROGRAMA PARA LAS TARJETAS ARDUINO

Este documento explica como conectar las tarjetas Arduino UNO y Arduino Nano al computador y cargar el código de cualquier aplicación.

Conexión

Para conectar la tarjeta Arduino UNO se necesita un cable estándar USB tipo B (figura 5). En el caso de la tarjeta Arduino Nano se necesita un cable estándar USB tipo mini B (figura 6).

Figura 5: *Arduino UNO y cable estándar USB tipo B.*

Figura 6: *Arduino Nano y cable estándar USB tipo mini B.*

IDE de Arduino

Para el desarrollo del prototipo se utilizó la versión 1.0 del software de Arduino.

Instalación de drivers

Al conectar cualquiera de las dos tarjetas Arduino al computador usando el respectivo cable USB, el led verde indicador de alimentación (nombrado PWR en las tarjetas) debería quedar encendido a partir de ese momento; así mismo Windows debería inicializar la instalación de los drivers siempre y cuando no se haya utilizado ese computador con una tarjeta Arduino anteriormente. En Windows Vista y Windows 7, los drivers deberían descargarse e instalarse automáticamente.

Ejecución del software de Arduino

Para ejecutar el entorno de Arduino se debe abrir la carpeta “arduino-1.0” y dar doble clic en la aplicación de Arduino.

Apertura de un programa

A continuación se desplegara la ventana que se muestra en la figura 7 en la que se debe dar clic en el menú File>Open y seleccionar el programa de la aplicación deseada desde el CD.

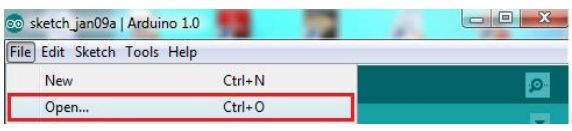


Figura 7: *Menú File.*

Selección de tarjeta

Para seleccionar el tipo de tarjeta Arduino que se está utilizando se da clic en el menú Tools>Board (figura 8).

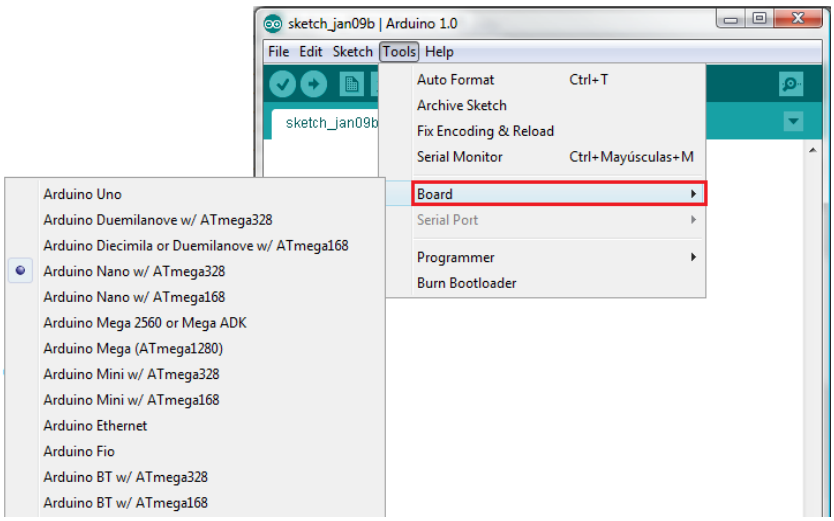


Figura 8: *Menú Tools.*

Selección del puerto serie

Para seleccionar el dispositivo serie de la tarjeta Arduino se da clic en el menú Tools>Serial Port.

Carga de un sketch a la tarjeta Arduino

El último paso es simplemente pulsar sobre el botón “Upload” en el entorno de Arduino. Si el programa se cargó exitosamente en la tarjeta Arduino aparecerá el mensaje “Done uploading” en la barra de estado.

Figura 9: *Botón Upload.*

PROGRAMA DE LA TARJETA ARDUINO NANO

Para está aplicación se debe cargar en la tarjeta Arduino el programa “Arduino_Nano.ino” que está localizado en la carpeta “Programas Arduino” del CD. Este programa nos permite visualizar en la pantalla LCD las variables eléctricas medidas, la hora y la fecha presente como se muestra en la figura 10.

Figura 10: *Pantalla LCD.*

REGISTRADOR DE VARIABLES ELÉCTRICAS REMOTO

Para correr está aplicación se debe cargar en la tarjeta Arduino UNO el programa “Registrador_SD.ino” que está localizado dentro de la carpeta “Programas Arduino” del CD. Este programa crea un archivo “.csv” denominado “ESPE-LXX.CSV” en la tarjeta de memoria SD el mismo que se enumera de 0 a 99 automáticamente cada vez que se reinicia la tarjeta Arduino UNO. En este archivo se almacena las variables eléctricas medidas con su respectiva hora y fecha cada 5 segundos aproximadamente como se muestra en la figura 11.

	A	B	C	D	E	F	G	H
1	ID	HORA	FECHA	VOLTAJE	IL1	IL2	IL3	IT
2	1	18:24:00	27.11.2012	117.94	0.67	1.55	5.68	7.89
3	2	18:24:06	27.11.2012	119.12	0.67	1.55	5.67	7.89
4	3	18:24:12	27.11.2012	120.15	0.67	1.69	5.67	8.03
5	4	18:24:18	27.11.2012	116.91	0.67	1.71	5.68	8.06
6	5	18:24:24	27.11.2012	116.47	0.67	1.74	5.68	8.09
7	6	18:24:31	27.11.2012	116.32	0.67	1.90	5.67	8.25
8	7	18:24:37	27.11.2012	119.71	0.67	1.87	5.67	8.21

Figura 11: *Datos almacenados.*

Para cargar cualquier programa en la tarjeta Arduino UNO se debe poner en apagado al interruptor de reinicio que se encuentra en el shield de la misma.

REGISTRADOR DE VARIABLES ELÉCTRICAS CON SUPERVISIÓN HMI DISEÑADA EN LABVIEW

Para está aplicación se debe cargar en la tarjeta Arduino UNO el programa “Registrador_UDP.ino” que está localizado en la carpeta “Programas Arduino” del CD. Además se debe abrir el programa “UDP.vi” localizado en la carpeta “Programa LabVIEW” del CD el mismo que permite realizar la supervisión del registrador en la PC mediante LabVIEW.

En la figura 12 se indica el panel frontal del programa desarrollado en LabVIEW.

Figura 12: *Panel frontal del UDP.vi*

1. Indica la medición del voltaje.

2. Indica la medición del sensor de corriente 1.

3. Indica la medición del sensor de corriente 2.

4. Indica la medición del sensor de corriente 3.

5. Indica la medición de la corriente total.

6. Presenta en una sola pantalla todas las mediciones de las variables eléctricas (figura 13).

7. Inicia la comunicación UDP entre la PC y la tarjeta Arduino.

8. Presenta en Excel el reporte de las variables eléctricas con su hora y fecha respectiva cada 2 segundos aproximadamente (figura 14).

9. Cierra la comunicación UDP entre la PC y la tarjeta Arduino.

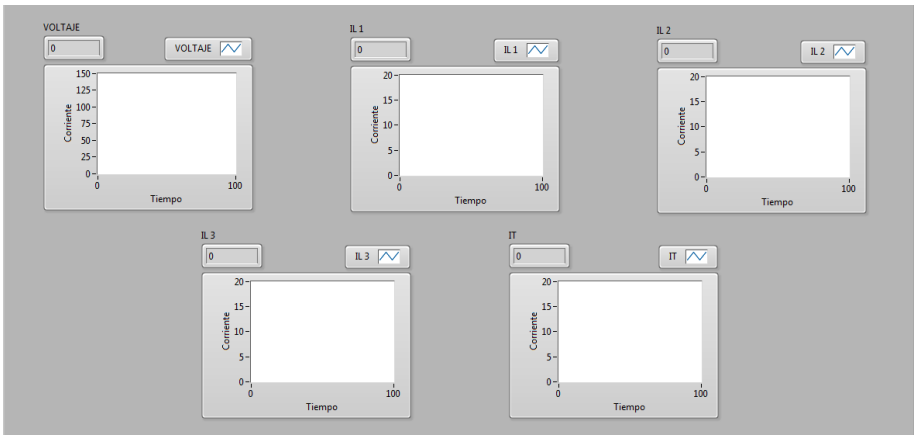


Figura 13: *Supervisión de las variables eléctricas*

	A	B	C	D	E	F	G
1	FECHA	HORA	VOLTAJE	IL1	IL2	IL3	IT
2	29/11/2012	12:14:49	120,294118	0,755342	0,457564	5,579342	6,792249
3	29/11/2012	12:14:50	123,235294	0,695787	0,457564	5,579342	6,732693
4	29/11/2012	12:14:52	120,147059	0,755342	0,457564	5,579342	6,792249
5	29/11/2012	12:14:56	121,911765	0,755342	0,457564	5,579342	6,792249
6	29/11/2012	12:14:58	123,823529	0,755342	0,457564	5,579342	6,792249
7	29/11/2012	12:15:00	122,794118	0,695787	0,457564	5,579342	6,732693
8	29/11/2012	12:15:02	120,441176	0,755342	0,457564	5,579342	6,792249
9	29/11/2012	12:15:04	121,911765	0,755342	0,457564	5,579342	6,792249
10	29/11/2012	12:15:05	123,529412	0,695787	0,51712	5,579342	6,792249

Figura 14: *Reporte en Excel.*

Nota: Para un correcto funcionamiento de la aplicación de debe de asignar los siguientes datos al computador:

- IP address: 192.168.1.166
- Subnet mask: 255.255.255.0
- Default gateway: 192.168.1.1

Además debe de estar conectado el cable Ethernet entre el computador y la tarjeta Arduino Ethernet Shield.

REGISTRADOR DE VARIABLES ELÉCTRICAS REMOTO CON MONITOREO A TRAVÉS DE UN NAVEGADOR WEB

Para está aplicación se debe cargar en la tarjeta Arduino UNO el programa “Registrador_TCP.ino” que está localizado en la carpeta “Programas Arduino” del CD. Además se debe asignar al computador los siguientes datos:

- IP address: 192.168.1.4
- Subnet mask: 255.255.255.0
- Default gateway: 192.168.1.1

En esta aplicación se debe conectar el cable Ethernet entre el computador y la tarjeta Arduino Ethernet Shield.

Para acceder a la página web de la tarjeta Arduino se debe digitar la dirección IP 192.168.1.177 en cualquier navegador web como se indica en la figura 15. En la página se muestra todos los archivos de almacenamiento que se encuentran en la tarjeta de memoria SD. Se

puede acceder a cualquier archivo con tan solo dar clic sobre uno de ellos.

Si el usuario (a) necesita un nuevo archivo de almacenamiento tiene que dar clic sobre el botón de RESET cerrar el navegador web y volver a abrirlo, y podrá visualizar en la página un nuevo archivo en el cual se están grabando las mediciones de las variables eléctricas cada 5 segundos (figura 16). El interruptor de reinicio que se encuentra en el shield de la tarjeta Arduino UNO debe de estar en encendido para el correcto funcionamiento del botón de RESET de la página de la tarjeta Arduino UNO.

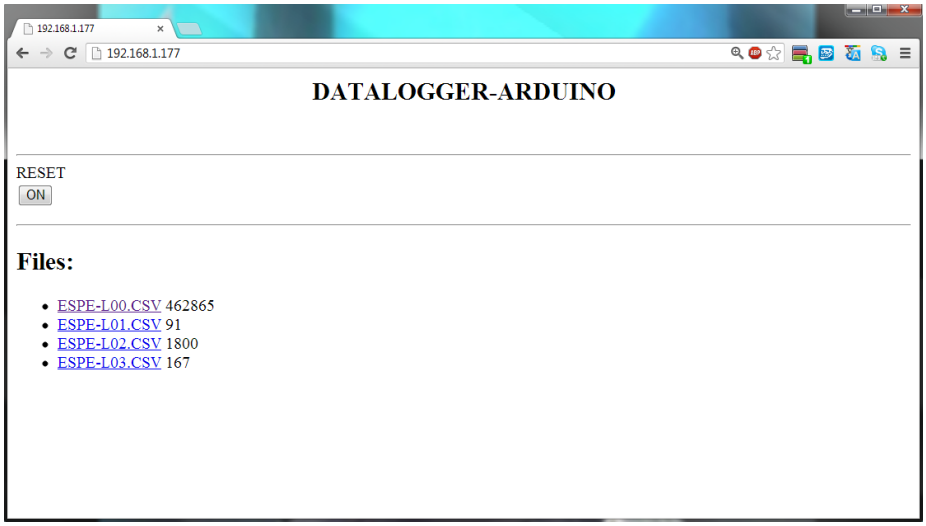


Figura 15: *Página de la tarjeta Arduino.*

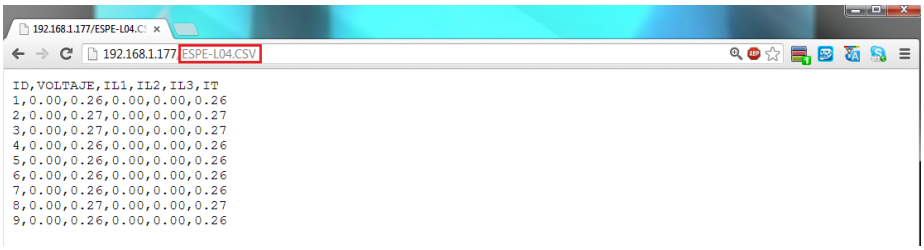


Figura 16: *Página del archivo ESPE-L04.*