UNIVERSIDAD DE QUINTANA ROO
DIVISIÓN DE CIENCIAS E INGENIERÍA

Diseño de un sistema de automatización y adquisición de datos vía web basado en un sistema embebido

TESIS
Para obtener el grado de
Ingeniero en Redes

PRESENTA
Emmanuel Cerón Lagos

DIRECTOR DE TESIS
Dr. Víctor Manuel Sánchez Huerta

ASESORES
Dr. Freddy Ignacio Chan Puc
Dr. Homero Toral Cruz
MTI. Vladimir Veniamin Cabañas Victoria
MC. Emmanuel Torres Montalvo

Chetumal Quintana Roo, México, diciembre de 2013

UNIVERSIDAD DE QUINTANA ROO
DIVISIÓN DE CIENCIAS E INGENIERÍA

Trabajo de Tesis elaborado bajo supervisión del Comité de asesoría

y aprobado como requisito parcial para obtener el grado de:

INGENIERO EN REDES

Comité de Trabajo de Tesis

Director: _____

Dr. Víctor Manuel Sánchez Huerta

Asesor: _____

Dr. Freddy I. Chan Puc

Asesor: _____

Dr. Homero Toral Cruz

Chetumal, Quintana Roo, México, Diciembre de 2013.

UNIVERSIDAD DE QUINTANA ROO

División de Ciencias e Ingeniería

Agradecimientos

Ser maestro es una profesión muy desgastante desde un principio he pensado que se necesita mucha paciencia para enseñar. Es por ello que quiero agradecer a esas personas que contribuyeron en la elaboración de este proyecto, además a todos los profesores que conforman la carrera de Ingeniería en Redes, de todos he aprendido muchas cosas que serán útiles para mi vida profesional.

Quiero agradecer a mi director de tesis el Dr. Víctor Manuel Sánchez Huerta quien me apoyo en la elaboración de este trabajo. Agradezco el tiempo y la dedicación que me fue brindado por su persona, principalmente la buena orientación que me dio para la creación de este proyecto. Agradezco a mis asesores, el Dr. Freddy Chan Puc y el Dr. Homero Toral Cruz, por el interés en este trabajo de tesis y por su tiempo invertido.

Finalmente agradezco a mis padres, que han estado siempre a mi lado y me han brindado su apoyo en cada momento.

.

UNIVERSIDAD DE QUINTANA ROO

División de Ciencias e Ingeniería

Dedicatoria

Es gratificante saber que cada esfuerzo tiene su recompensa, es hermoso mirar el pasado y darte cuenta que con el paso de los años lo único que ha cambiado en tu persona son los conocimientos adquiridos, pero lo más hermoso es darte cuenta que a pesar de los obstáculos lo has logrado. Quiero dedicar este proyecto a mis abuelos q.e.p.d. los cuales me inspiraron y guiaron por un buen camino.

Llegar hasta este nivel a punto de convertirme en un profesionista es un sentimiento incomparable. Me es grato dedicarles este proyecto en el que he trabajado con tanto esmero a mis padres quienes me han apoyado en cada momento para seguir adelante.

UNIVERSIDAD DE QUINTANA ROO

División de Ciencias e Ingeniería

Resumen

En este trabajo de tesis se presenta el desarrollo de un sistema de adquisición de datos que incorpora un servidor web vía Ethernet, basado en la plataforma de desarrollo de sistemas embebidos abierta Arduino. El sistema de adquisición de datos puede monitorear hasta 16 variables analógicas y se configura vía remota para determinar el tiempo de adquisición y registro de los datos. Las características del sistema de adquisición de datos le permiten ser utilizado de forma aislada requiriendo adicionalmente de una fuente de alimentación portátil. Además, como contribución adicional de este trabajo, se ha configurado un esquema de alarma y protección que se activa cuando algunas de las variables monitoreadas por el sistema de adquisición de datos, excede un nivel determinado por el usuario durante cierto intervalo de tiempo. Esta función podría ser utilizada por ejemplo para la protección de colectores solares o helióstatos ante la presencia de vientos con velocidades importantes que potencialmente puedan dañar la estructura de estos sistemas.

UNIVERSIDAD DE QUINTANA ROO

División de Ciencias e Ingeniería

ÍNDICE

ÍNDICE DE FIGURAS

CAPÍTULO 1

CAPÍTULO 1

INTRODUCCIÓN

Los sistemas embebidos basados en microcontroladores han tenido un desarrollo importante en las pasadas dos décadas, tanto a nivel de hardware como de software. Sin embargo, a pesar de esa innovación y crecimiento, las técnicas de diseño han evolucionado poco, debido entre otras causas al hecho de que cada fabricante de microcontroladores emplea su propia plataforma de desarrollo y con ello limitando su uso masivo lo que incrementa sus costos.

En los últimos años ha surgido una plataforma “abierta” de sistemas embebidos, tanto en software como en hardware, la cual está posicionándose de manera importante como una plataforma de desarrollo común y de bajo costo para el desarrollo de sistemas embebidos. Esta plataforma “abierta” de sistema embebido es comercialmente distribuida por la compañía italiana Arduino. La compañía Arduino ensambla esta plataforma “abierta” integrada básicamente por una etapa de comunicación basado en un microcontrolador Atmega16U2 y un sistema mínimo basado en microcontroladores de la compañía Atmel. La compañía Arduino comercializa diferentes modelos de tarjetas, cada una con especificaciones particulares para que el diseñador seleccione la más adecuada a su proyecto.

El término de “abierto” para la plataforma Arduino, significa que tanto la arquitectura de hardware así como el software de programación de estos sistemas embebidos está disponible gratuitamente lo que permite que el diseñador pueda adaptar y configurar esta clase de sistemas embebidos acorde a sus necesidades y en su caso inclusive ser modificado a las necesidades del diseñador. Además, al ser una plataforma “abierta”, los usuarios contribuyen con el desarrollo de nuevas funciones de tal forma que

en un nuevo diseño el tiempo de desarrollo puede ser reducido al contar ya con algunas soluciones desarrolladas. Asimismo, esta plataforma “abierta” puede ser vinculada con otros lenguajes de programación como Matlab, Labview, Python, Java, por mencionar algunos de ellos, lo que facilita su integración en sistemas embebidos ya implementados.
Basado en estas ventajas de la plataforma de microcontrolador embebido Arduino, en el presente trabajo se le ha seleccionado como la plataforma de diseño de un sistema de adquisición de datos controlado vía un servidor web.

Este trabajo está organizado de la siguiente manera: en el capítulo 2 se presenta una introducción a la tecnología de los microcontroladores, haciendo énfasis en los microcontroladores Atmel y en la plataforma Arduino Mega sobre la cual se ha implementado el sistema de adquisición de datos desarrollado en este trabajo.
En el capítulo 3 se presenta una descripción del ambiente de programación de la plataforma Arduino, así como programación y el diseño físico del sistema de adquisición de datos y el servidor web configurado en la plataforma Arduino Mega. En el capítulo 4 se muestran los resultados experimentales de la implementación física del sistema de adquisición de datos desarrollado en este trabajo. Finalmente en el capítulo 5 se presentan las conclusiones.

JUSTIFICACIÓN

Una de las necesidades primordiales en procesos industriales o en trabajos de investigación experimental es el monitoreo y registro de datos de las variables físicas que intervienen en ellos. Esta tarea la realiza un sistema de adquisición de datos. Sin embargo, el costo económico de adquirir este sistema o de su desarrollo con plataformas de microcontroladores propietarias es alto actualmente. De esta forma, el presente trabajo de tesis presenta el diseño e implementación de un sistema de adquisición de datos, controlador vía web basado en una plataforma de desarrollo “abierta” de bajo costo. Además, como contribución adicional de este trabajo, se ha configurado un esquema de alarma y protección que se activa cuando algunas de las variables monitoreadas por el sistema de adquisición de datos, excede un nivel determinado por el usuario durante cierto intervalo de tiempo.

OBJETIVO GENERAL

Implementar un sistema de adquisición de datos configurable mediante un servidor web en un sistema embebido de arquitectura “abierta” y en el que además se tenga la capacidad de implementarse tareas de alarma y salidas digitales.

OBJETIVOS PARTICULARES

- Descripción de la plataforma Arduino y su ambiente de programación.
- Diseño y configuración del sistema de adquisición de datos.
- Diseño del sistema de alarma y salida digital para la detección de magnitudes superiores a un valor determinado.
- Implementación del sistema de adquisición de datos y servidor web en la plataforma Arduino Mega.

CAPÍTULO 2

CAPÍTULO 2

CONCEPTOS BÁSICOS

En este capítulo se presenta los principales conceptos que son utilizados en el proyecto. El primero es la tecnología de los microcontroladores, haciendo énfasis en la marca Atmel y en la plataforma Arduino Mega que es la parte esencial del trabajo en la cual se ha implementado el sistema de adquisición de datos.

2.1 Microcontroladores

El microcontrolador es un circuito integrado programable que incluye en su interior las tres unidades funcionales de una computadora: unidad central de procesamiento (CPU, Central Proccesing Unit por sus siglas en inglés) que se encarga de realizar operaciones aritméticas y lógicas, memoria y periféricos de entrada y salida también conocidos como puertos. Ver Figura. 2.1

Figura 2.1 Microcontroladores

La capacidad de procesamiento y sus respectivos encapsulados pueden variar dependiendo la aplicación, y pueden encontrarse en casi cualquier dispositivo electrónico como automóviles, lavadoras, hornos de microondas, teléfonos, etc.

Al ser configurado y programado solo tiene una tarea en específico, por eso el tamaño de la unidad central de procesamiento, cantidad de memoria y periféricos incluidos dependerá del programa que en este reside.

2.2 Diferencia entre Microcontroladores y Microprocesadores.

Existen varias diferencias entre los microcontroladores y microprocesadores, una de las más importantes es la funcionalidad. Es decir, para que un microprocesador sea funcional necesita conectar su unidad central de proceso o CPU, por medio de sus terminales a una memoria externa y a dispositivos de entrada y salida. Como se observa a continuación en la Figura 2.2.

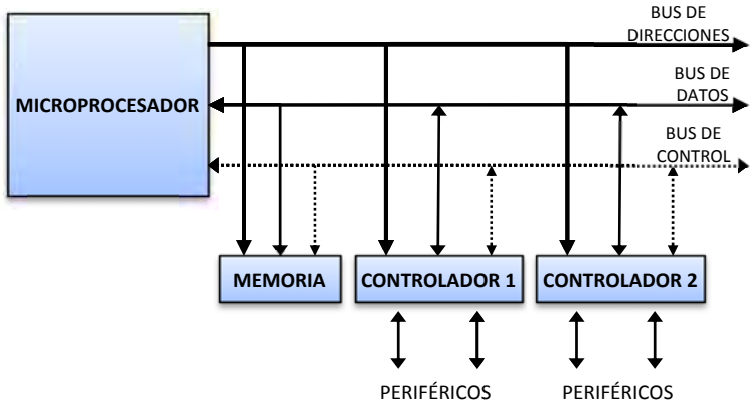

Figura 2.2 Diagrama del Microprocesador

El microcontrolador tiene la ventaja de tener tanto las capacidades de procesamiento, almacenamiento de datos e instrucciones (memoria) y la comunicación con su entorno (los periféricos de entrada y salida) dentro de un solo circuito integrado y puede llegar a considerarse una computadora a menor escala. Ver Figura 2.3

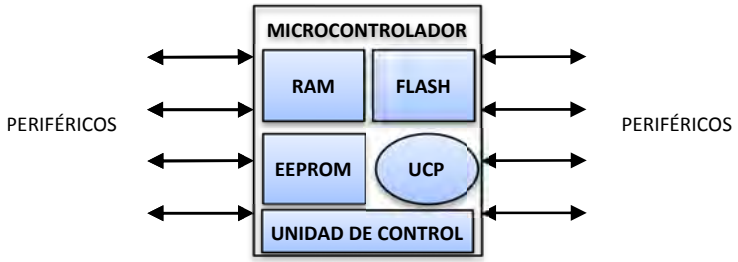

Figura 2.3 Diagrama del microcontrolador

Una desventaja del microcontrolador es que al tener todos sus componentes en un solo encapsulado se le considera un sistema cerrado, es decir que no puede llegar a ser modificado y puede tener limitaciones dependiendo a la aplicación a la que sea destinado.

2.3 Diferencias entre AVR y PIC.

Existen muchas empresas que son fabricantes de microcontroladores entre algunas podemos nombrar: Intel, Motorola, Texas Instruments, Microchip, Cypress, Atmel y otras. Sin embargo, dos fabricantes de microcontraladores destacan: la familia AVR y la familia de PIC, cuya popularidad es alta entre los diseñadores de sistemas embebidos que necesitan un alto rendimiento y un bajo costo, y son elegidos principalmente por su nivel de integración, su arquitectura, la disponibilidad de recursos o su lenguaje de programación.

La empresa Microchip es fabricante de microcontroladores PIC, que durante mucho tiempo fue líder del mercado a nivel mundial, ya que se caracterizaban por tener una unidad central de procesamiento de arquitectura RISC y memoria FLASH para el almacenamiento del Firmware.

La familia de AVR son microcontroladores de la empresa ATMEL, y al igual que los microcontroladores PIC cuentan con un CPU RISC y memoria FLASH [1]. Además de que en ambos casos cuentan con periféricos como Puertos Digitales, Convertidores Analógico-Digital (ADC por sus siglas en inglés), módulos PWM (Pulse Width Modulation), por mencionar los periféricos más frecuentemente usados. Al observar la descripción general entre PIC y AVR no se nota ninguna diferencia, pero es el ámbito de desarrollo de un sistema embebido, en donde se encuentran grandes diferencias entre ambos microcontroladores, como por ejemplo: el lenguaje de programación, Ambiente Integrado de Desarrollo (IDE, Integrate Development Environment por sus siglas en ingles), las interfaces para la programación, el reloj interno, el voltaje de alimentación, la potencia, el costo y otros más [2].

2.3.1 Lenguaje de programación y Ambiente Integrado de Desarrollo (IDE).

Los microcontroladores al ser sistemas digitales programables, requieren una serie de instrucciones o comandos para realizar un trabajo en específico, estos programas son necesarios para que el sistema embebido tenga un buen control. El lenguaje de programación de un microcontrolador, es el lenguaje de ensamblador, un lenguaje de bajo nivel. Pero para cada empresa el lenguaje es distinto.

Para AVR podemos encontrar compiladores de lenguaje C, C++, Basic, y cada uno de ellos tienen la ventaja de no tener que aprender ensamblador, además de que cualquier ambiente de desarrollo cuenta con un mejor diseño de los programas. La principal ventaja es que la descarga de cualquiera de estos es gratuita. Hoy en día existen unas plataformas ya prediseñadas basadas en

microcontroladores AVR, como es el Arduino, que reduce el tiempo de programación de un microcontrolador.

Para los PIC, además de usar lenguaje ensamblador, se pueden encontrar en el mercado diferentes tipos de ambientes de desarrollo, sin embargo cabe mencionar que no son gratuitos ya que es necesario pagar por cada uno de los entornos que se desea desarrollar la aplicación y al igual que AVR, en PIC existen hoy en día placas precargadas con un microcontrolador PIC llamadas PICAXE, que necesita un puerto serial y conocimiento en el lenguaje BASIC.

2.3.2 Interfaces de programación

Las interfaces de programación son el hardware externo que se necesita para programar o descargar el código generado en el ambiente de desarrollo sobre la memoria FLASH del microcontrolador, esta herramienta o programador físico se necesita conectar a una computadora por un puerto. En la familia AVR, el microcontrolador provee de un puerto ISP para su programación en la memoria, dicho puerto serial está formado por 3 pines del microcontrolador, además se destacan por ser simples y sencillos. Ver Figura 2.4

Figura 2.4 Interfaz de programación para AVR

En la familia de los PIC al igual que AVR, se utilizan puertos seriales para conectar a la computadora la interfaz de programación. La Figura 2.5 es un ejemplo de una interfaz de programación para PIC´s.

Figura 2.5 Interfaz de programación para PIC´s

2.3.3 Consumo de energía y otras características.

Algunas otras características entre estas familias, que pueden llegar a ser muy importantes es el consumo de energía. Dado que los microcontroladores AVR consumen en menor cantidad que un PIC tanto voltaje como en corriente.

Las familias de microcontroladores AVR tienen una característica de tener en el circuito interno un reloj. El circuito del reloj provee la señal de sincronización, frecuencia o velocidad a la que el microcontrolador ejecutará sus instrucciones. Cabe mencionar que los microcontroladores AVR aparte de tener el oscilador interno se le puede anexar uno externo. Todo esto da como resultado una respuesta más rápida por parte de los microcontroladores AVR, ya que puede leer o ejecutar más instrucciones o líneas de código por ciclo de reloj [3].

En lo referente al costo de cada uno de estas familias de microcontroladores, la diferencia es a favor de la familia AVR, ya que mientras que un PIC tiene un aproximado de 90 pesos con características similares, un AVR tiene el precio de 48 pesos. El bajo costo de los microcontroladores AVR los hace ideales para desarrollar aplicaciones de bajo costo.

2.4 Microcontroladores AVR

Esta nueva familia de microcontroladores se basa en una arquitectura RISC que ha incorporado un módulo de memoria flash destinada al programa, así como memoria EEPROM para programas y datos. Además de que esta arquitectura es completamente compatible con el lenguaje de programación C.

Los microcontroladores AVR de Atmel es un grupo muy extenso y todos comparten el mismo núcleo AVR, pero varían en el número de periféricos y cantidad de memoria (RAM y ROM), de tal forma que son clasificados según el tamaño de datos que maneja en el repertorio de instrucciones. Los microcontroladores AVR van desde los 4, 8, 16 y hasta 32 bits.

Los microcontroladores de 8 bits son relativamente rápidos ya que permiten la ejecución de un método llamado "pipeline" que consta de dos etapas: cargar y ejecutar. El método pipeline permite una o más instrucciones en un solo ciclo de reloj. Entre las herramientas que poseen los microcontroladores AVR está la comunicación o manejo de interfaces de comunicación serial como SPI, I2C, UART, USB, 1WIRE, 2WIRE. Además de estar integrados por convertidores análogo digitales, oscilador RC interno y memoria no volátil (EEPROM) interna, entre otros [4].

La familia de microcontroladores AVR se clasifican en tres grandes categorías dependiendo del mercado:

Gama baja: TINY AVR, son de propósito general con una memoria flash limitada de hasta 2 Kbytes y 128 bytes de memoria SRAM y EEPROM.

Gama Media: LCD AVR, son microcontroladores de propósito general con 8Kbytes de memoria flash y 512 bytes de memoria SRAM y EEPROM.

Gama alta o mejorada: mega AVR, con memoria flash desde 256 kbytes y 4 kbytes de memoria EEPROM y SRAM, respectivamente. El tipo de encapsulado de los microcontroladores AVR tiene presentaciones de los 28 hasta los 100 pines del tipo DIP, TQFP y MLP. Además de tener un rango de 1.8 a 5.5 .

A continuación se muestra la Tabla *2.1* con los principales tipos de microcontroladores AVR ATmega en el mercado:

Tabla *2.1* Tipos de microcontroladores AVR

PRODUCTO	FLASH (Kb)	EEPROM(Bytes)	RAM(Bytes)	I/O
		MEGA AVR		
ATmega48	4	256	512	23
ATmega88	8	512	1K	23
ATmega162	16	512	1K	35
ATmega32	32	1K	2K	32
ATmega64	64	2K	4K	53
ATmega128	128	4K	4K	53
ATmega256	256	4K	8K	53
		LCD AVR		
ATmega169	16	512	1K	53
ATmega329	32	1K	2K	53
		TINY AVR		
ATmega TINY 198	64	256	256	60

2.5 Sistema embebido Arduino

El sistema embebido Arduino es una placa de origen Italiano que está constituido en dos partes: la placa Arduino (hardware) que se caracteriza por un utilizar un microcontrolador AVR y que dispone de un grupo específico de entradas y salidas en donde se pueden construir fácilmente circuitos, y el ambiente de desarrollo Arduino IDE, que corresponde a la parte del software que facilitan el desarrollo de aplicaciones. Los sistemas Arduino son fáciles de programar ya que se basan en el lenguaje de programación C [5]. La placa del Arduino se conecta hacia la computadora mediante un puerto USB, que aparte de ser un puerto muy conocido en cualquier equipo de cómputo, no requiere la adquisición de un equipo adicional para su programación. En la Figura 2.6 se observan algunos diseños actuales de la plataforma Arduino.

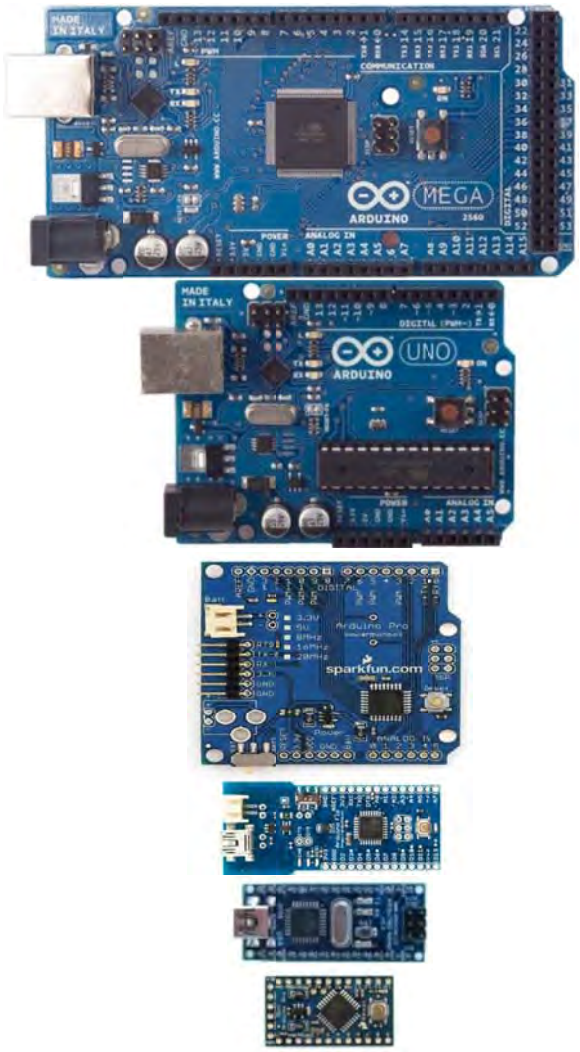

Figura 2.6 Ejemplos de placas Arduino

El microcontrolador en la placa ARDUINO de Atmel se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Y una vez cargado el programa en la memoria del microcontrolador, ya no es necesario conectar la placa a una computadora.

Existen otras placas con otros microprocesadores como son: Parallax, Basic Stamp, Phidgets, MIT's Handyboard, Picaxe y algunas otras con una funcionalidad parecida. Sin embargo, la plataforma ARDUINO ha tenido un impacto mayor

debido principalmente a su manejo como plataforma abierta de desarrollo. Las principales ventajas de los sistemas Arduino son:

- Simplifican el proceso de estar trabajando directamente con un microcontrolador.
- El Arduino contiene un programador incorporado, a diferencia de la plataforma PIC, en donde es necesario la compra del programador por separado con un costo aproximado de $ 720 pesos, mientras que el Arduino con la versión más costosa es de aproximadamente $ 400 pesos.
- Todo el software y librerías son gratis y de fácil aprendizaje, ya que se basan en un lenguaje de programación C, más conocido y sencillo que el ensamblador.
- Es multi-plataforma, es decir que la placa Arduino funciona en cualquier sistema operativo (Windows, Mac, Linux) sin ningún costo adicional por el IDE.

Existen varias versiones de la plataforma ARDUINO, dependiendo las necesidades que requiera cada proyecto y todas usan microcontroladores Atmel ATmega168 y en algunos casos ATmega8.

2.6 Sistema embebido Arduino Mega 2560.

El Arduino Mega 2560 es una plataforma de hardware y software libre basada en el microcontrolador AVR ATmeg1280 de 8 bits, el cual cuenta con 256 KB de memoria flash, 8 KB de SRAM y 4 KB de EEPROM [6]. Además, el ATmeg1280 tiene 54 entradas digitales para entrada o salida, de estas 14 son

para el uso de modulación por anchos de pulso (por sus siglas en ingles PWM) , 16 entradas analógicas, 4 entradas para el uso de puertos UARTs, un cristal oscilador de 16 MHz y un grupo de 6 pines dedicados para la programación del Bootloader del microcontrolador llamados ICSP(In Circuit Serial Programming). El Bootloader es la parte lógica de un Arduino que hace la programación de un proyecto más sencilla, sin la necesidad de algún programador externo. Ver Figura 2.7

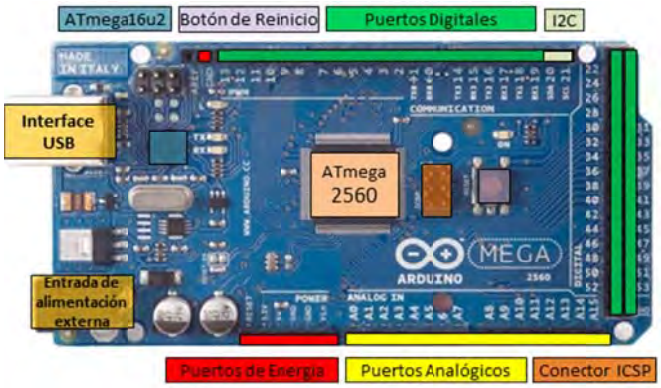

Figura 2.7 Arduino Mega 2560

Para la comunicación con una computadora la plataforma Arduino mega cuenta con un microcontrolador ATMega16U2 programado para convertir de USB a serial, que permite mayores velocidades de transmisión por su puerto USB y no requiere drivers para Linux o MAC. El Arduino Mega 2560 también soporta las tecnologías I2C y SPI para la comunicación entre microcontroladores y otras placas Arduino o tarjetas de función específica.

2.7 Placa Arduino Ethernet Shield

La Arduino Ethernet Shield es una tarjeta-accesorio que permite que un Arduino se conecte a Internet. Esta Shield está basada en el módulo Wiznet W5100, que proporciona un enlace de red (IP) tanto por TCP como por UDP [7]. La tarjeta Arduino Ethernet Shield soporta la conexión de hasta cuatro usuarios de forma simultánea. Para generar programas que se conecten a Internet usando esta placa se necesita utilizar la librería de Ethernet, que tiene incluida el entorno de desarrollo.

La placa Arduino Ethernet Shield tiene incluido un conector hembra RJ45 para la conexión a Internet y una ranura para tarjeta de memoria micro-SD, en la cual se pueden almacenar archivos para posteriormente consultarlos. En la Figura 2.8 se muestra la tarjeta Ethernet shield.

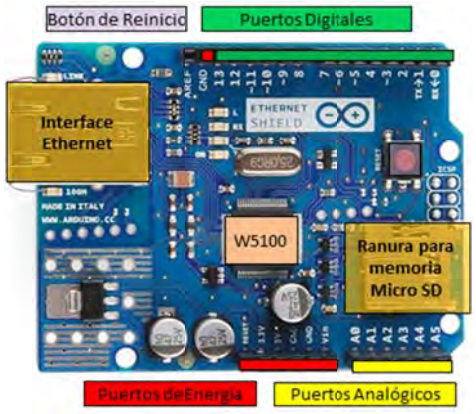

Figura 2.8 Arduino Ethernet Shield

La Arduino Ethernet Shield se comunica tanto con el módulo W5100 como con la tarjeta microSD utilizando el bus SPI (a través del conector ICSP).

Físicamente esta conexión se realiza en los pines 50, 51 y 52 de la tarjeta Mega 2560. El acceso a Ethernet o a la tarjeta microSD se realiza de forma multiplexada, de esta forma el pin 10 se usa para seleccionar el módulo W5100 y el pin 4 para la tarjeta microSD. Estos pines son reservados, por lo que no pueden ser usados para entrada o salidas genéricas.

2.7.1 Modulo Wiznet W5100

El módulo Wiznet W5100 se ha diseñado para facilitar la ejecución de la conexión a Internet sin necesidad de un sistema operativo. El W5100 es compatible con el estándar IEEE 802.3 10BASE-T y 802.3u 100BASE-TX.

El W5100 incluye un stack de TCP/IP y un Ethernet MAC y PHY Ethernet. El stack TCP/IP soporta los protocolos de comunicación más utilizados. Tiene un buffer interno de 16Kbytes que está incluido para la transmisión de datos.

Características:

- Protocolos TCP/IP admitidos: TCP, UDP, ICMP, IPv4, ARP, IGMP, PPPoE y Ethernet.
- Admite Negociación Automática (Full-duplex y Half-duplex)
- Admite conexión ADSL (con soporte de protocolo PPPoE con Modo de Autentificación PAP/CHAP)
- Admite 4 sockets independientes simultáneamente
- Admite Interfaz Periférica Serial (MODO SPI)

Para facilitar la integración, el W5100 admite tres interfaces diferentes de acceso a memoria, por bus directo o indirecto y por SPI. La integración del módulo Wiznet con la placa Arduino Mega se muestra en la Figura 2.9.

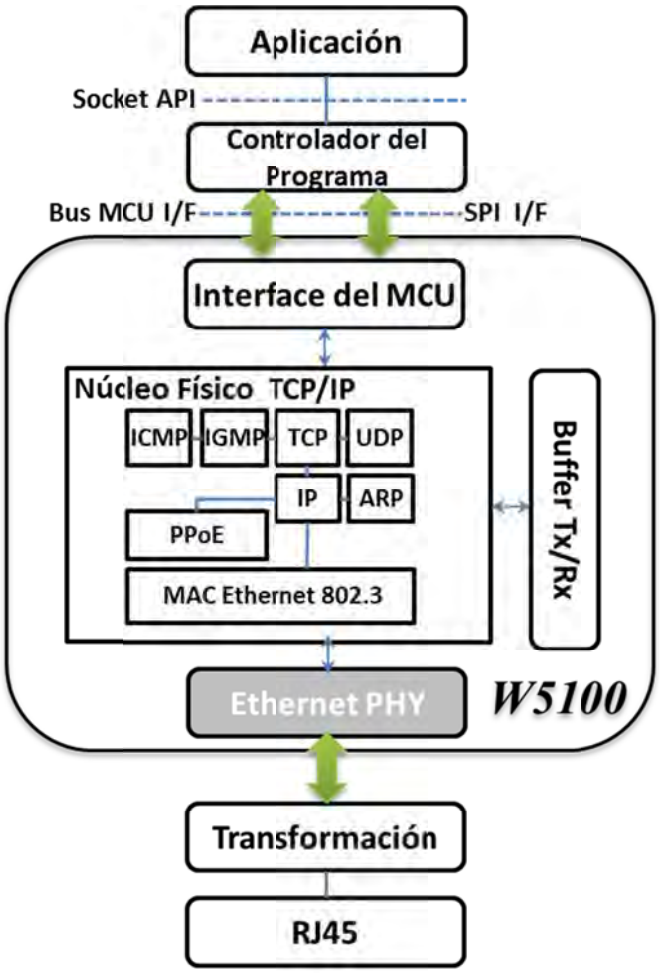

Figura 2.9 Diagrama Wiznet W5100

2.8 Sensores

Son dispositivos que están diseñados para realizar mediciones de magnitudes del medio ambiente tanto físicas o químicas y transformarlas en variables eléctricas, por ejemplo temperatura, fuerza, humedad, velocidad del viento, distancia, aceleración, entre otras. A continuación se mencionan los sensores utilizados en este trabajo.

2.8.1 Sensor de temperatura

Son dispositivos que transforman los cambios de temperatura en señales eléctricas, suelen estar formados por un elemento sensor y envueltos por un material semiconductor, ver Figura 2.10. Esta aleación tiene como objetivo que los cambios ambientales se transmitan rápidamente del elemento sensor hacia un dispositivo final. En este trabajo de tesis se empleó el sensor de temperatura LM35H.

Figura 2.10 Sensor de temperatura.

Características [8]:

- Rango entre -55°C a +150°C.
- Ideal para aplicaciones remotas.
- Bajo costo.
- Opera entre 4 y 30 volts de alimentación.
- Mínimo auto-calentamiento.
- La salida de voltaje es lineal y equivale a 10mV/°C por lo tanto:
    - +1500mV = 150°C
    - +250mV = 25°C
    - -550mV = -55°C

2.8.2 Sensor de viento.

Es un Instrumento utilizado para medir la velocidad del viento. El anemómetro mide la velocidad instantánea del viento, pero las ráfagas desvirtúan la medida, de manera que la medida más acertada es el valor medio de medidas que se tomen en intervalos de 5 a 10 minutos. El anemómetro de rotación está dotado de cazoletas (Robinson) o hélices unidas a un eje central cuyo giro es proporcional a la velocidad del viento, como se muestra en Figura 2.11.

Figura 2.11 Anemómetro.

Estos anemómetros generan una señal senoidal cuya frecuencia está relacionada directamente con su velocidad de giro y son utilizados para aplicaciones meteorológicas o en el estudio para determinar si el recurso eólico en un sitio es redituable para una inversión en este tipo de generación eléctrica.

CAPÍTULO 3

CAPÍTULO 3

DISEÑO LÓGICO E IMPLEMENTACIÓN DE LOS CIRCUITOS DEL SISTEMA DE AUTOMATIZACIÓN Y ADQUISICIÓN DE DATOS.

En este capítulo se aborda la descripción del ambiente en el que se genera la configuración lógica en la placa Arduino. La programación del microcontrolador es muy necesaria para que los valores de las variables provenientes de los sensores puedan ser monitoreados y almacenados de manera correcta. La programación del servidor web está basada en el diseño de diagramas de flujo que facilita el análisis y comprensión de cada proceso que se realiza en sistema embebido.

En esta sección se presenta el desarrollo, diseño y descripción del funcionamiento de los circuitos para un sistema de adquisición y monitoreo de datos, también para un esquema de alarma y prevención automática de condiciones ambientales que se activa en el momento en el que se excede un límite predeterminado.

3.1 Ambiente de Programación de la placa Arduino Mega 2560

La programación de placa Arduino Mega2560 se realiza con el software gratuito Arduino y que se descarga desde la página web de Arduino. El lenguaje de programación es un pseudo código en C basado en el lenguaje de programación Java. La figura 3.1 muestra la interfaz de usuario del ambiente de programación Arduino.

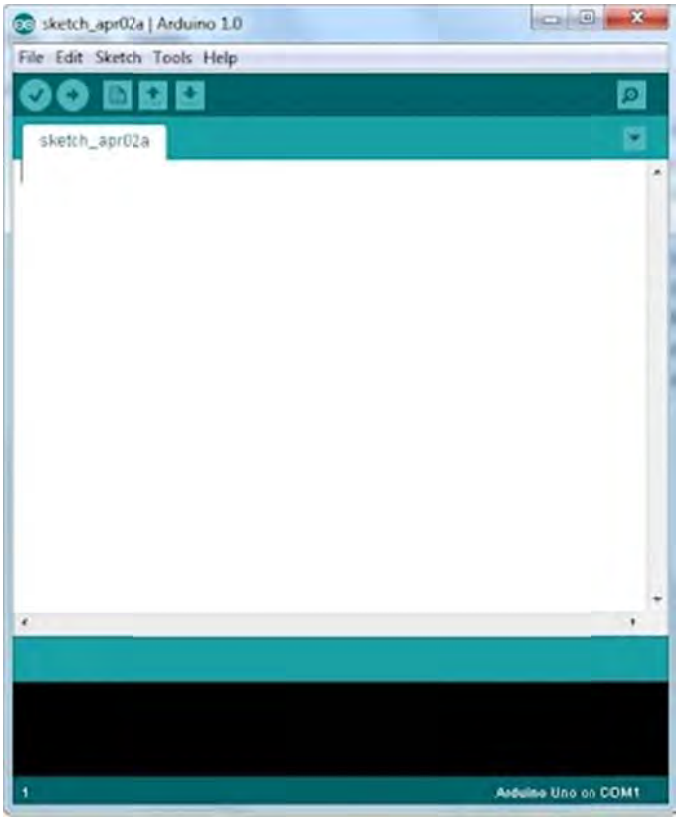

Figura 12 Ambiente de programación de Arduino

La estructura básica del lenguaje Arduino es muy simple y se compone básicamente de dos bloques fundamentales que contienen declaraciones de

variables e instrucciones para que el programa (denominado sketch en el ambiente de Arduino) sea grabado en el microcontrolador. Esta estructura tiene la siguiente distribución:

```
void setup()
{
        instrucciones;
}
void loop()
{
        instrucciones;
}
```

El bloque de la función setup() es la parte que se encarga de realizar la configuración de las variables del programa, en donde se encuentran la declaración de variables, solo se ejecuta una vez e inicia o configura los periféricos para que sean entradas o salidas. En tanto la función loop() contiene el código que se ejecutará continuamente, esta función es la parte central o núcleo de todos los programas de Arduino y la que realiza la mayor parte del trabajo.

3.2 Servidor Web

Un Servidor web es un dispositivo remoto que brinda un servicio de almacenamiento de información en forma de páginas web en la red a uno o varios clientes. Para tener acceso al dispositivo con el servidor web se necesita conocer la dirección IP estática que tiene establecida el servidor y el número de puerto.

El servidor web está en un estado abierto y pasivo, es decir que en cualquier momento un cliente puede tener acceso como muestra la Figura 3.2.

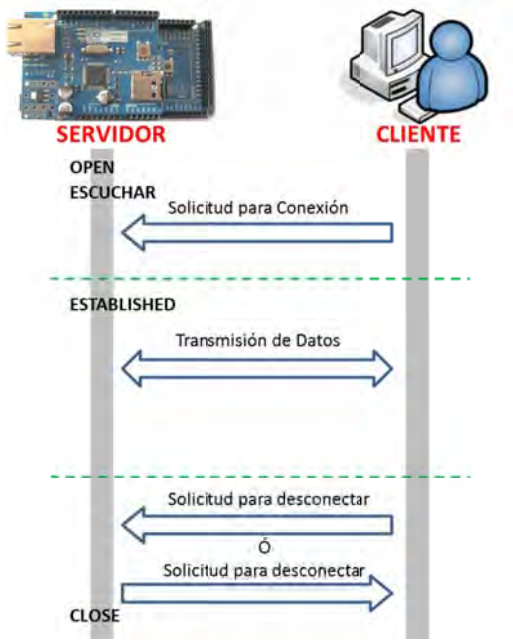

Figura 13 Cliente-Servidor

El servidor web siempre está en estado de espera de algún usuario y para establecer un vínculo o conexión, el cliente debe crear una solicitud y esperar la respuesta por parte del dispositivo. Si la conexión es exitosa el usuario podrá realizar consultas o crear nuevos archivos en el servidor. Para finalizar el enlace de cliente-servidor, el servidor web cuenta con un tiempo límite por conexión, además de la que puede tomar por decisión el usuario final. En la figura 3.3 se muestra el diagrama de flujo del servidor web empleado en este trabajo.

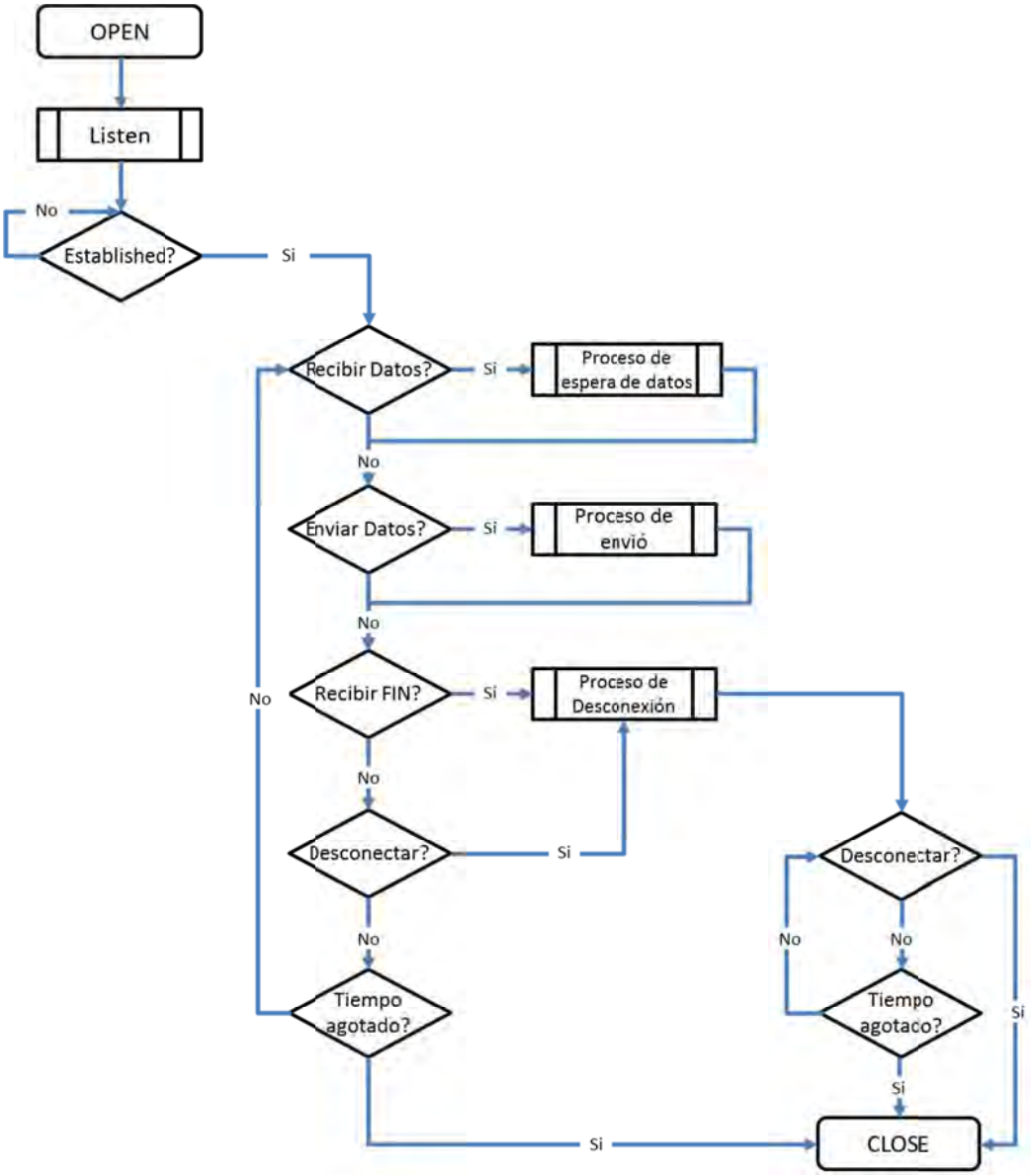

Figura 14 Diagrama del servidor web

3.3 Diagrama de flujo del programa principal del servidor web.

En primera instancia se presenta el diseño del diagrama de flujo del programa principal del servidor web y posteriormente se describirán los diferentes subprocesos que son llevados a cabo. El programa principal inicia con la detección

de algún problema físico en la memoria micro SD, en caso de no existir errores pasa a la siguiente etapa. En el bloque de setup() se declaran las configuraciones de los periféricos del microcontrolador que son utilizados por el sistema, así como también se declaran e inicializan las variables del programa.

Los periféricos del microcontrolador utilizados son:

- El puerto digital para la alarma (pin 24).
- El puerto digital para el botón de reinicio de la alarma (pin 26).
- El puerto digital para el circuito del sensor de viento (pin 2), además de configurar la función del interruptor adjunto (attach-Interrupt) en el modo flanco de bajada (FALLING Edge).
- El puerto digital para la comunicación SPI entre la placa Arduino Mega y la Ethernet Shield (pin 53).

La configuración de las direcciones estáticas del servidor web se realiza con la librería Ethernet de Arduino y se declaran fuera del módulo, pero hasta este proceso dan inicio. Las direcciones son las siguientes:

- Dirección física del Servidor web (MAC)
- Dirección IP.
- Mascara de red.
- Puerta de salida.
- El puerto de salida.

El módulo de Loop() es el encargado de dar inicio al servidor web y en el caso de presentarse algún cliente que requiera su servicio, el servidor web presentar la lista de archivos generados hasta el momento y valores actuales de los sensores o en caso de que se desee adquirir un nuevo archivo, darle acceso a los botones previamente configurados.

Los valores actuales de los sensores, el sistema de adquisición de datos y la alarma se describirán en las siguientes secciones.

La Figura 3.4 muestra el diagrama de flujo del programa principal.

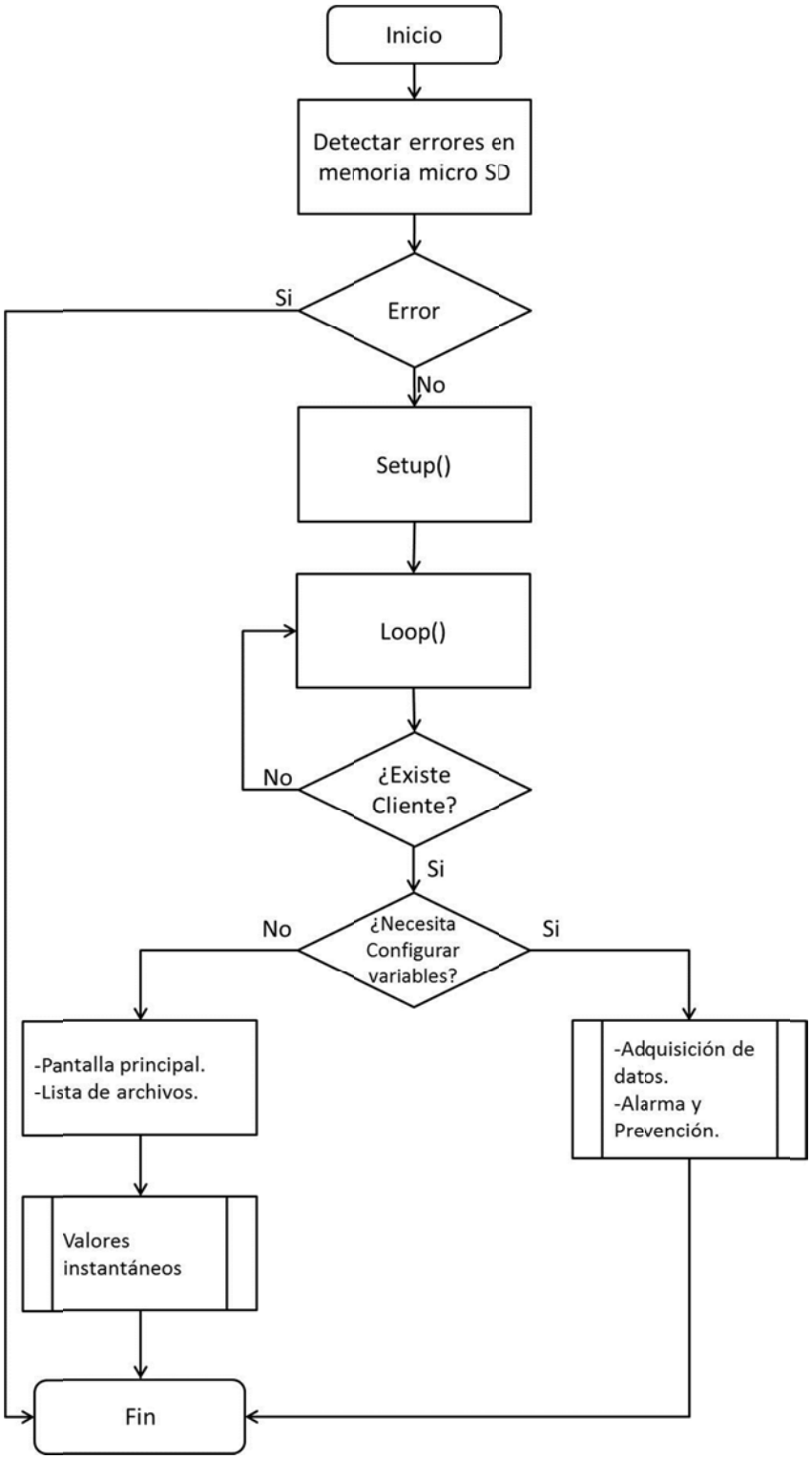

Figura 15 Diagrama general del servidor web.

3.3.1 Diagrama del subproceso para desplegar los valores instantáneos.

Este subproceso es un servicio que brinda el servidor web sobre la pantalla de inicio, informando al cliente los valores actuales de las variables ambientales que se están midiendo, a modo de ejemplo a esta variable se le denominará V_a. Esta variable es un arreglo de dos elementos $(p; t)$, uno de las cuales inicia en cero, pero aumenta en cada adquisición. El valor registrado de los sensores en el momento es almacenado de manera temporal y cuando el arreglo se llena, se realiza un promedio de los datos registrados y se imprime en la pantalla principal. La Figura 3.5 muestra el diagrama de flujo de este programa.

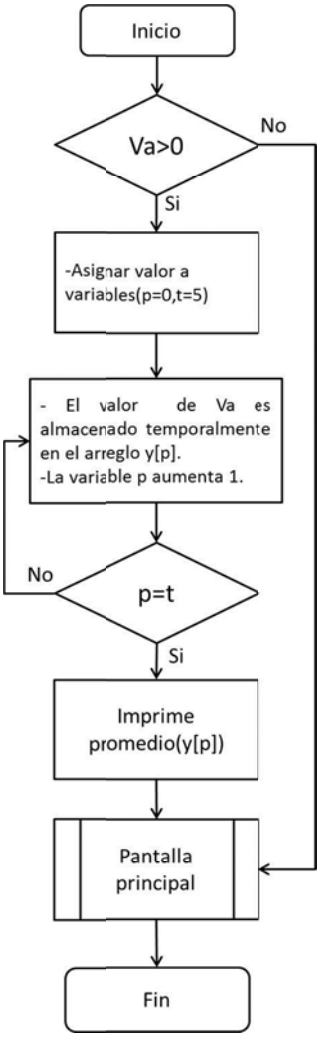

Figura 16 Diagrama de valores instantáneos.

3.3.2 Diagrama de subproceso para la adquisición de datos.

Este subproceso se encarga del registro y almacenamiento de los valores de las variables ambientales medidas para posteriormente poder ser analizados o descargados vía web desde cualquier dispositivo por ejemplo equipos de cómputo o dispositivos móviles. El diagrama de flujo de este subproceso se muestra en las Figuras 3.6.1 y 3.6.2.

Este programa está compuesto por cuatro variables, dónde dos de ellas son inicializadas en cero $(i; n)$ y se van incrementando conforme se almacena el valor que se genera de manera instantánea de los sensores.

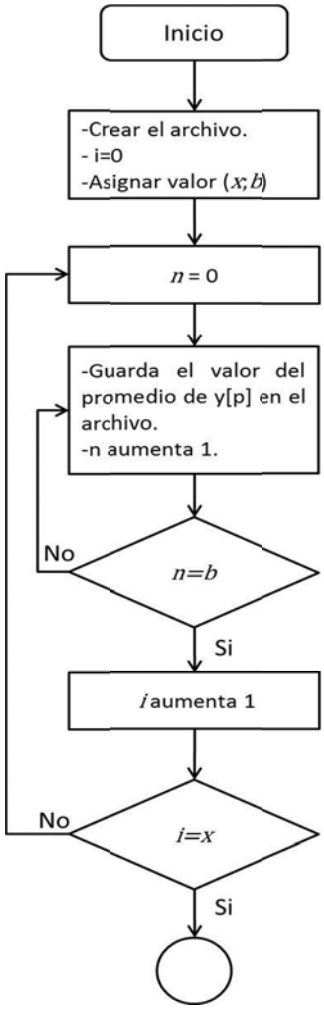

Figura 17 Diagrama de adquisición de datos parte 1.

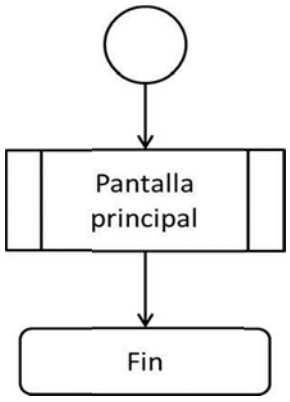

Figura 18 Diagrama de adquisición de datos parte 2.

Las variables en esta subrutina están pre-configuradas para un tiempo en específico en un grupo estático de botones sobre la página principal del servidor web. El intervalo de tiempo para la adquisición de datos es configurable y está determinado por la siguiente ecuación:

$$x(t) = \frac{t}{ab}$$

Para modificar el tiempo de ejecución (x) de este proceso, solo es necesario sustituir en la función:

- El tiempo en segundos que estará trabajando (t)
- El intervalo en el que se realizara cada adquisición (a)
- El tiempo total para un ciclo (b).

3.3.3 Diagrama de subproceso para el sistema de prevención y alarma.

Este subproceso se encarga de supervisar que la variable que se desea controlar no exceda un valor predeterminado; por ello este subproceso trabaja de manera independiente a la ejecución del programa del servidor web, ya que no requiere la presencia de algún cliente y se ejecuta desde que el sistema embebido enciende. Básicamente la alarma se activa cuando la variable medida de interés excede un límite pre-establecido durante un determinado intervalo de tiempo. En este trabajo se ha seleccionado la velocidad del viento como la variable sobre la que opera esta alarma. La velocidad del viento es de nuestro interés debido a que por nuestra ubicación física en el Caribe estamos expuestos a tormentas y huracanes en los que el viento alcanza velocidades importantes que pueden dañar estructuras. El servicio de alarma tiene como finalidad indicar si la velocidad del viento ha excedido un cierto límite pre-establecido durante cierto tiempo al mismo tiempo que activa una salida digital cómo para activar algún mecanismo de protección.

La alarma se ha configurado para discriminar entre rachas de viento de corta duración y vientos sostenidos durante un determinado intervalo de tiempo. De esta forma se evita una falsa alarma por una racha de viento.

El sistema de alarma inicia si el circuito de medición de velocidad de viento se encuentra conectado a la placa Arduino Mega. Si el circuito está instalado correctamente se establece la velocidad límite (V) con la que inicia la alarma sobre un periodo de tiempo total (z), además se declaran en cero dos variables ($x; p$) que incrementan de valor en cada prueba o muestra obtenida del sensor de viento.

Las muestras evaluadas son el resultado del promedio obtenido de un arreglo de (p) valores de un tiempo parcial (t). En el caso de que la velocidad de viento medida sea igual o mayor a la velocidad límite (V) durante el periodo (z), el sistema de alarma y prevención activará un LED, que indica que la variable ha excedido el límite pre-establecido. El diagrama del sistema de prevención y alarma

automático se presenta en la Figura 3.7. En el anexo A de esta tesis se presentan los programas desarrollados en este trabajo de tesis.

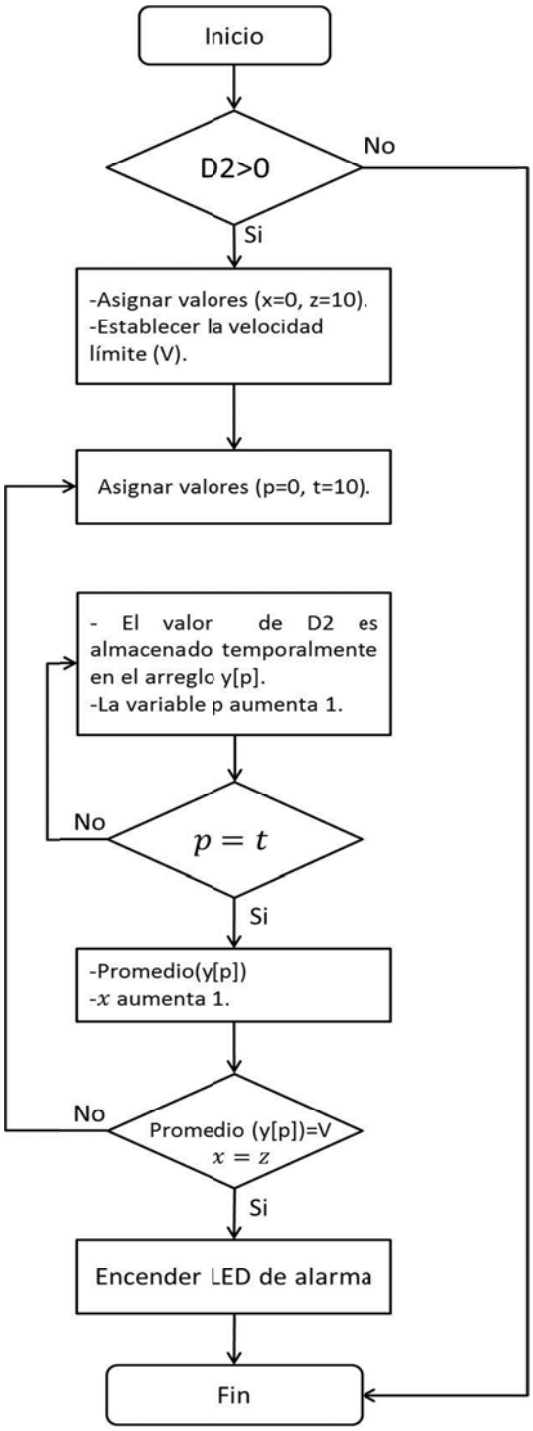

Figura 19 Diagrama des sistema de prevención y alarma.

3.4 Configuración física del sistema embebido Arduino Mega 2560.

Para la configuración física de la placa Arduino Mega2560 es necesario insertar la tarjeta Arduino Ethernet Shield sobre el Arduino Mega2560 como muestra en la Figura 3.8. La tarjeta Arduino Mega debe estar conectada a una batería de 9 volts o a un adaptador de CA-CD (Corriente Alterna a Corriente Directa) con un voltaje entre los 7 y 12 volts. La placa Arduino con una etapa de regulación de voltaje que provee un voltaje de salida por cada puerto digital de 5 volts y 40 mA, además de una salida regulada de 5 volts. Para la conexión con la red Ethernet es necesario conectar la entrada RJ45 del Arduino Ethernet Shield a un modem o roseta con un cable de conexión directa CAT5 ó CAT6.

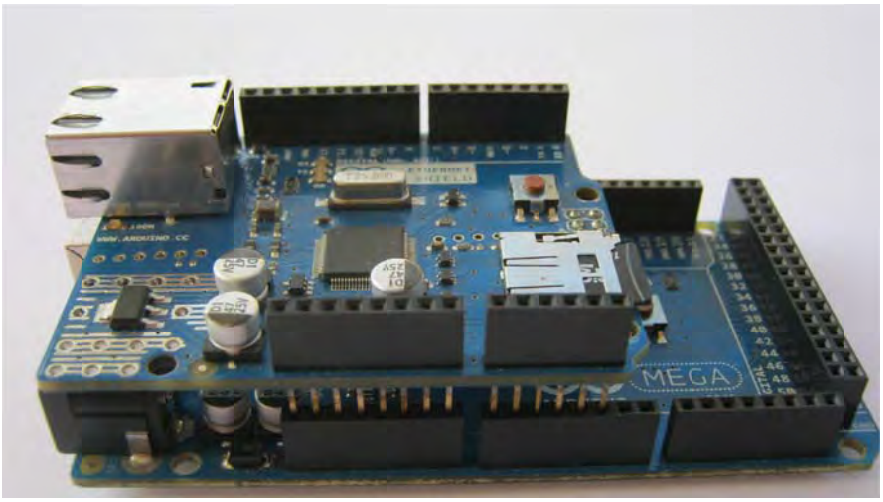

Figura 20 Arduino Mega con Ethernet Shield

3.5 ADC y circuito acondicionador del sensor de temperatura.

Para la lectura y almacenamiento de datos de un sensor analógico la placa Arduino Mega2560 cuenta con 16 canales de lectura multiplexados a un convertidor de datos analógico a digitales (ADC) de 10-bits con un voltaje de referencia de 5 volts. Los 10 bits del ADC dividen este voltaje de referencia en 1024 valores, por lo que cada combinación equivale a 0.0049 volts (4.9 mV), como se muestra la Figura 3.9. La frecuencia de muestreo del ADC es de 125 kHz y depende de la velocidad del reloj de la tarjeta Arduino que opera a 16 MHz y cuya frecuencia puede ser dividido por un pre-escalador Para conocer el número de muestras por milisegundos se divide la frecuencia del ADC (125kHz) entre el número de ciclos en los que se realiza cada muestra (13) y el límite físico de la placa Arduino Mega es de 96 muestras por milisegundo, quiere decir que la lectura de datos se realizan en un tiempo aproximado de 0.0001 segundos (100 microsegundos) entre cada adquisición [9].

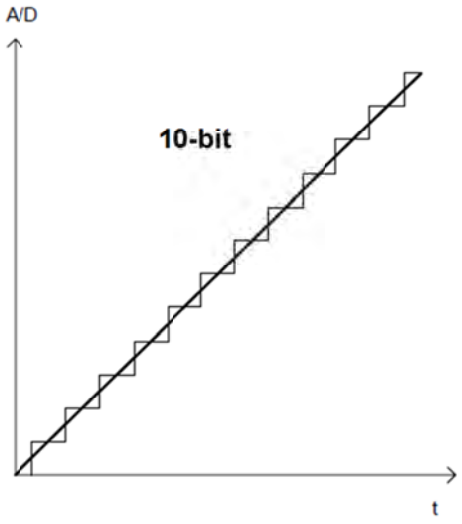

Figura 21 Gráfica del ADC de 10-bits.

El circuito consta de un sensor LM35 que proporciona una salida de voltaje lineal proporcional a la temperatura, es un sensor integrado de precisión, cuya tensión de salida es linealmente proporcional a temperatura en ºC (grados centígrados). El LM35 puede operar en rango amplio de de temperaturas que abarca desde los -55 ºC bajo cero a 150 ºC.

La etapa de acondicionamiento de la señal del sensor LM35 consiste únicamente de un filtro pasa-bajas con la finalidad de cancelar ruido de baja frecuencia. La Figura 3.10 muestra el circuito acondicionador en donde se ha empleado un capacitor de 1µF y una resistencia de 10Ω para el filtro pasa-bajas. La señal del sensor ha sido conectada a la entrada analógica A0.

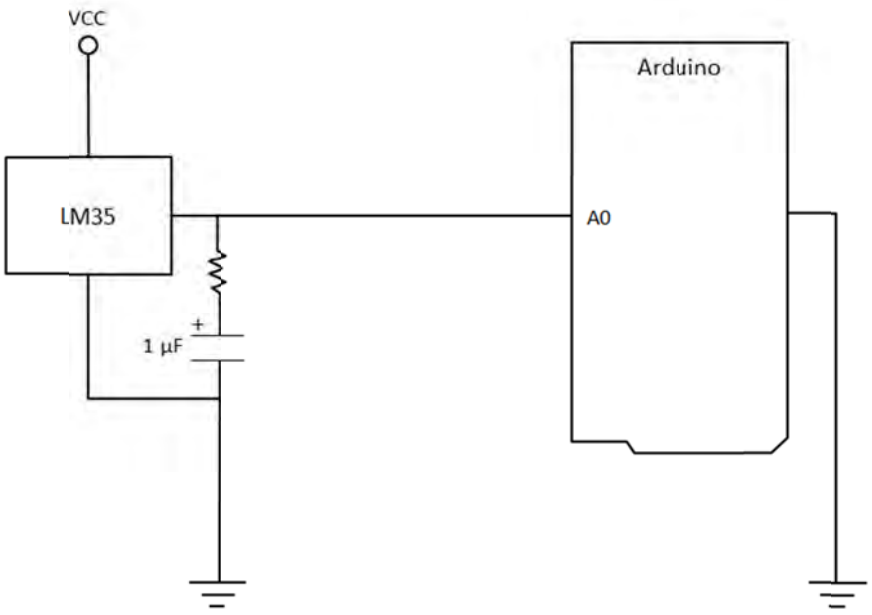

Figura 22 Circuito del sensor de temperatura.

3.6 Circuito de medición de velocidad de viento.

Para la medición de velocidad de viento se requiere de un circuito acondicionador de señal para la conexión del anemómetro con la tarjeta Arduino Mega. El anemómetro genera una señal de salida en forma de onda senoidal de Corriente Alterna (CA) con una amplitud variable, y una frecuencia que es proporcional a la velocidad del viento. Ver Figura 3.11

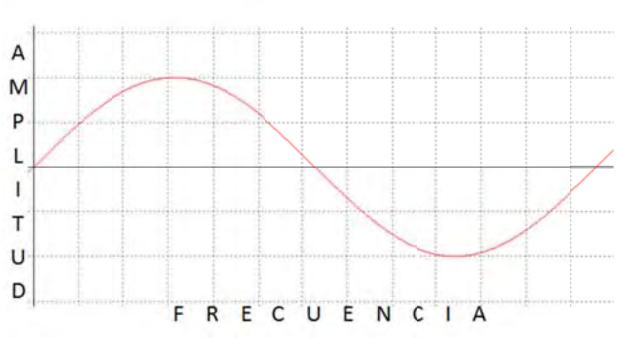

Figura 23 Grafica de Corriente Alterna.

El rango de frecuencia de la señal generada por el anemómetro es de 2 Hz a 100 Hz. La velocidad máxima que llega a medir el anemómetro es de 96 m/s que equivale a 345.6 km/h (Esta velocidad de viento excede el límite inferior de un huracán categoría 5). La amplitud de la señal tiene un mínimo de 80 mV pico a pico para una velocidad de viento de 3 km/h y un máximo de 12 Volts pico a pico para la velocidad de viento máxima que puede medirse con el anemómetro.

A continuación se presenta en la Figura 3.12 el diagrama completo del circuito con el anemómetro y la placa Arduino.

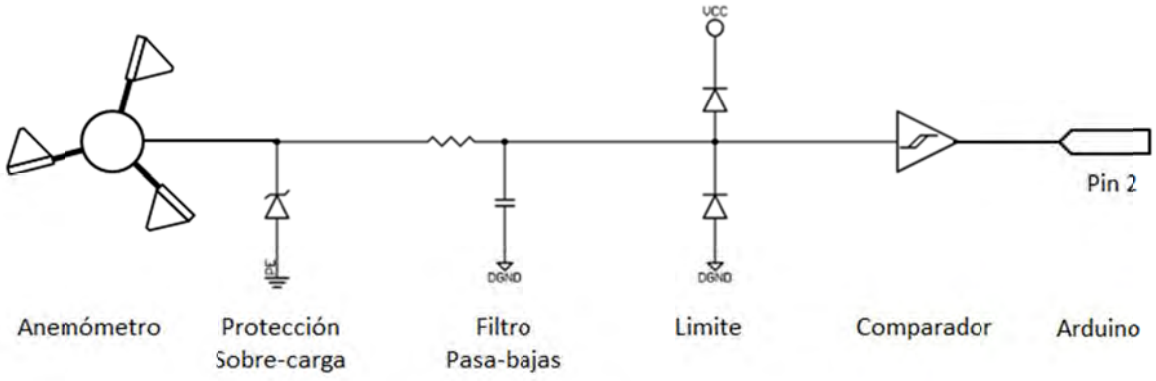

Figura 24 Circuito completo para el sensor de viento.

- Protección de Sobre-carga.

Esta protección consta de un diodo supresor de picos de voltaje que potencialmente puedan dañar al microcontrolador.

- Filtro Pasa-bajas.

El filtro es preciso para eliminar ruido de alta frecuencia que pueda presentarse en la señal del sensor: Este filtro cuenta con una resistencia de 10kΩ conectada a la salida y un capacitor de 1µF.

- Limitador de voltaje

El anemómetro produce una señal con una amplitud de Corriente Alterna (CA) y puede llegar a exceder el voltaje que soporta la placa Arduino, los diodos limitadores son utilizados para la protección de entradas en polaridad invertida y sobrecargas de voltaje. El límite está conectado directamente en serie con el filtro pasa-bajas y está constituido por un par de diodos conectados en serie.

- Comparador

Es utilizado para realizar la conversión de la onda senoidal proveniente del circuito, a una señal lógica digital con forma cuadrada para la lectura en la tarjeta Arduino Mega.

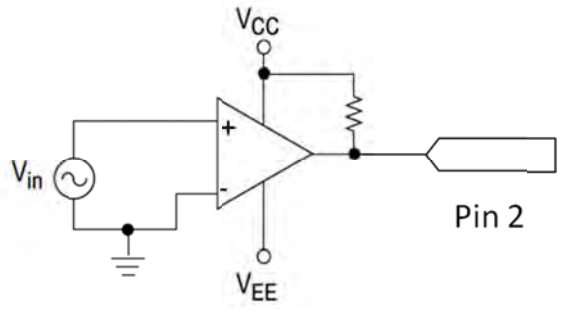

Figura 25 Circuito lógico del comparador.

El comparador utilizado es el LM339 y está conectado como muestra la Figura 3.13. La señal que produce el comparador está conectada a una resistencia 220Ω con el voltaje de entrada (Vcc) y se conecta directamente a la entrada digital 2 de la placa Arduino.

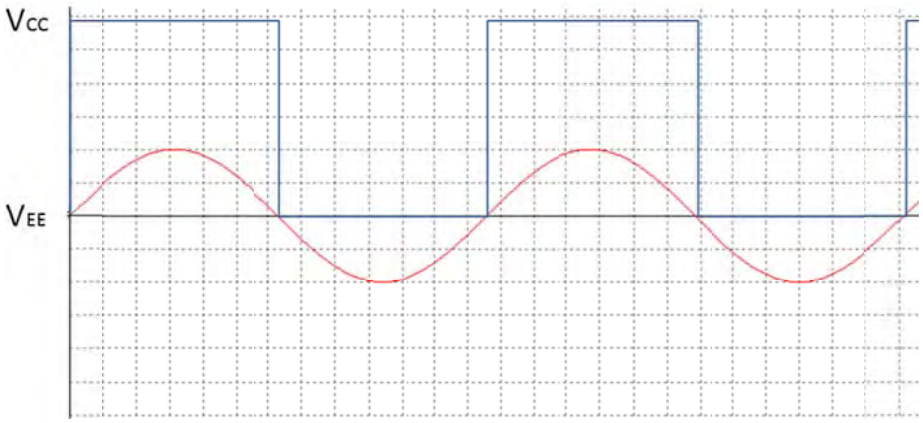

Figura 26 Detección de cruce por cero.

El LM339 crea una onda cuadrada a través de la detección de cruce de la onda senoidal por un voltaje de cero como se muestra en la Figura 3.14. La señal generada tiene como amplitud fija un valor mínimo de cero (VEE) y un máximo de 5 volts (Vcc). La frecuencia de la onda carece de un valor constante, ya que la velocidad del viento es variable. El circuito del comparador es alimentado por 5 volts (Vcc) que son provenientes de la placa Arduino y una salida de tierra, el diagrama se presenta en la Figura 3.15:

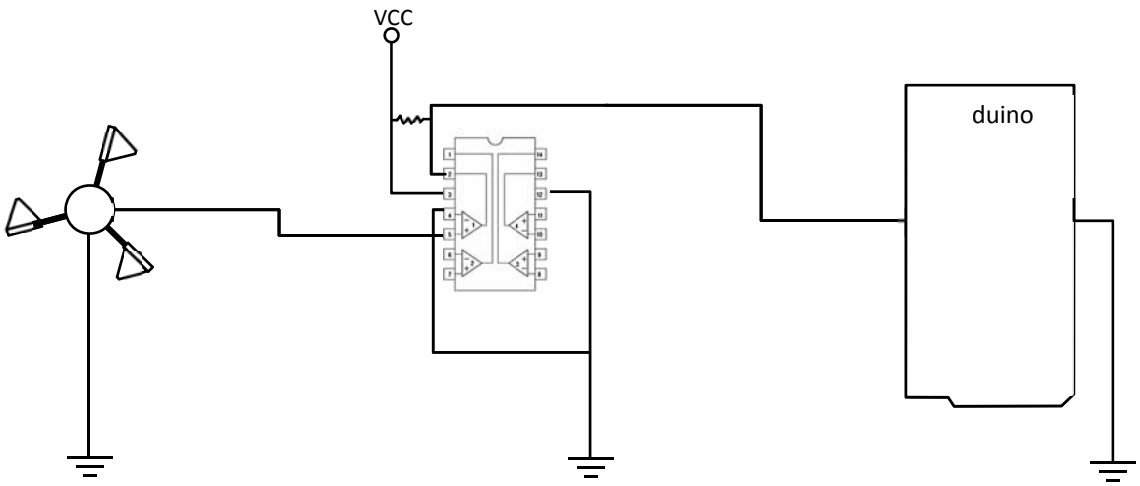

Figura 27 Diagrama físico del sensor de viento con el comparador.

Para la lectura de los datos que genera el circuito de medición de velocidad de viento, la tarjeta Arduino tiene integrado en 6 puertos digitales la función Interrupción-Adjunta (Attach-Interrupt), la cual es utilizada para realizar una acción específica automáticamente o para monitorear una señal proveniente del algún dispositivo externo al microcontrolador [10]. La Figura 3.16 muestra la señal digital que activa la interrupción.

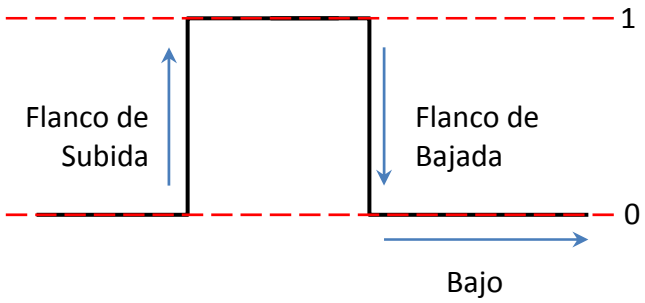

Figura 28 Interrupción adjunta.

La acción o función que es llamada por la interrupción está conformada por cuatro casos específicos:

1. Flanco de subida (RISING), se realiza en el momento en el que la transición de la onda pasa del valor mínimo de cero a 5 volts, el valor máximo de amplitud.
2. Flanco de Bajada (FALLING), se genera cuando la transición de la onda pasa de un valor máximo al mínimo.
3. Flanco Bajo (LOW), es cuando la onda cuadrada se encuentra en un valor mínimo cero.
4. Cambio de flanco (CHANGE), este modo es activado en el momento en el que la onda realice cualquier tipo de transición.

Para la parte de automatización del circuito, es necesario un indicador que sea capaz de informar que la velocidad instantánea registrada en el momento por el microcontrolador puede llegar a ser de riesgo, y así realizar una acción de emergencia. Para la señalización de la activación de la alarma se ha empleado un LED el cual está conectado a la salida digital número 24, como muestra la Figura 3.17. La resistencia de 200 ohms tiene como finalidad limitar la corriente suministrada por la terminal de salida digital del microcontrolador.

Figura 29 Circuito del LED.

Para reiniciar la alarma ante una eventual activación se ha diseñado un circuito de reinicialización como se muestra en la Figura 3.18.

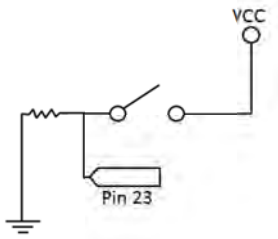

Figura 30 Circuito del interruptor.

El circuito consta de un interruptor (pushbotton), conectado en un extremo a una resistencia de 220Ω en serie y a tierra, y en el otro extremo al voltaje de entrada (Vcc), el circuito se conecta al puerto digital número 23.

CAPÍTULO 4

CAPÍTULO 4

RESULTADOS EXPERIMENTALES

En este capítulo se describe la etapa de la instalación física y resultados obtenidos con el sistema desarrollado. La fase de pruebas detalla el funcionamiento de cada uno de los circuitos de medición de temperatura y velocidad del viento utilizados. De igual forma se muestra la operación del sistema de alarma y prevención. Los resultados demuestran un adecuado desempeño de la adquisición de datos de las variables medidas al interactuar con el servidor web implementado en la tarjeta Arduino Mega.

4.1 Instalación física del Servidor Web.

La Figura 4.1 muestra el esquema general del sistema de adquisición de datos con servicio Web y alarma desarrollado en este trabajo.

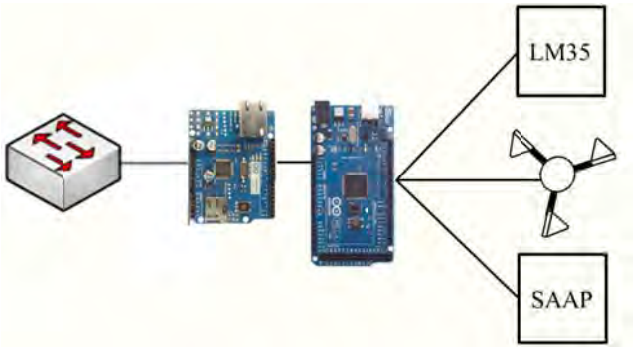

Figura 31 Esquema de Servidor web.

El servidor web y los sensores están ubicados en las instalaciones de la Universidad de Quintana Roo tal como muestra la Figura 4.2. D esta forma, el servicio web está disponible en la intranet de la Universidad.

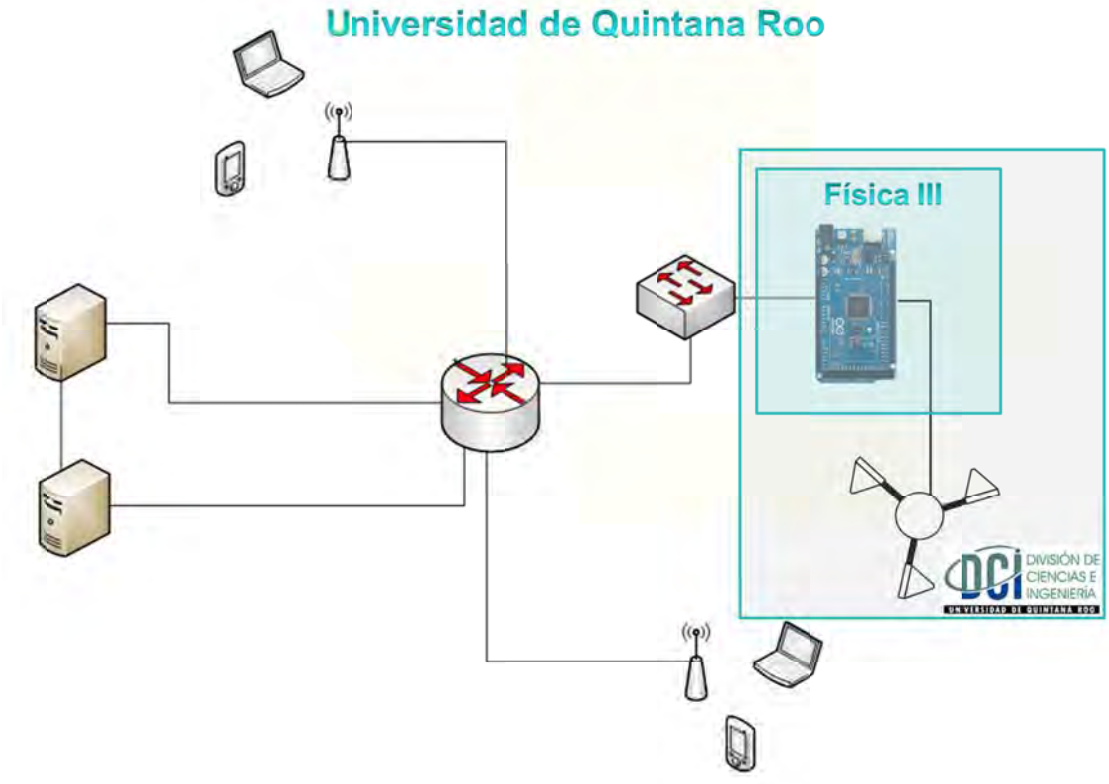

Figura 32 Esquema de la instalación en la Universidad

El sistema desarrollado se ubicó físicamente en el laboratorio de Física III. El sistema de adquisición de datos está conectado a la red mediante un cable UTP de conexión directa categoría 6.

Las conexiones del dispositivo hacia los sensores del sistema de adquisición de datos y la alarma se describirán en siguientes secciones.

4.2 Instalación física de los sensores con la placa Arduino.

Los sensores están conectados a la tarjeta Arduino Mega con hilos de cable UTP de par trenzado. Para el sensor de temperatura se utilizaron tres hilos de distinto color, como muestra la Figura 4.3. El hilo de color verde proporciona al sensor una conexión de tierra desde la placa Arduino Mega, el cable verde con blanco proporciona al sensor el voltaje de alimentación de 5 V, para el funcionamiento de este circuito y el hilo de color azul es el encargado de enviar al sistema de adquisición de datos la señal de voltaje correspondiente a la temperatura medida por el LM35.

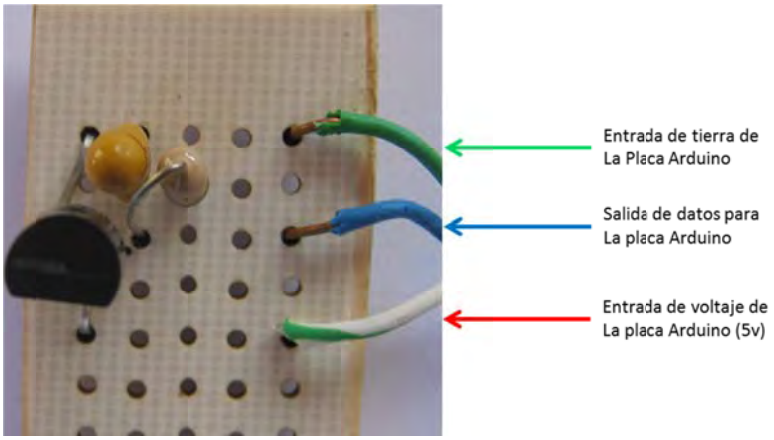

Figura 33 Sensor de Temperatura.

El Anemómetro está conectado al circuito del sensor de viento con un par de hilos de cable UTP, estos hilos proveen de un extremo positivo los datos recolectados (flecha morada) y el otro hilo una conexión de tierra (flecha verde). Para que el circuito funcione, la placa Arduino subministra de 5 volts (flecha roja) y una salida de tierra (flecha verde).

Los valores arrojados del circuito son enviados al dispositivo en un solo hilo (flecha azul). En total para el sensor de viento requirió de cinco hilos de cable UTP, el color de los hilos puede variar pero el esquema de instalación es como se muestra en la Figura 4.4.

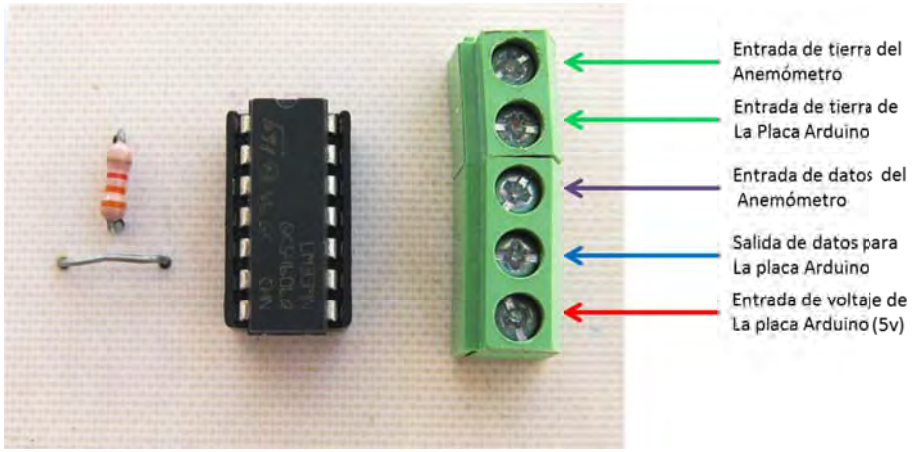

Figura 34 Sensor de viento.

La instalación del sistema de alarma y prevención del viento consta de dos circuitos. El primer circuito es utilizado para reiniciar de forma manual dicha función y el cual se muestra en la Figura 4.5. Este circuito requiere de tres hilos de cable UTP par trenzado uno de alimentación (flecha roja) hilo verde con blanco, el segundo para tierra (flecha verde) hilo verde y el tercer hilo azul con blanco (flecha azul) que tiene como tarea informar al servidor web si el botón ha sido activado.

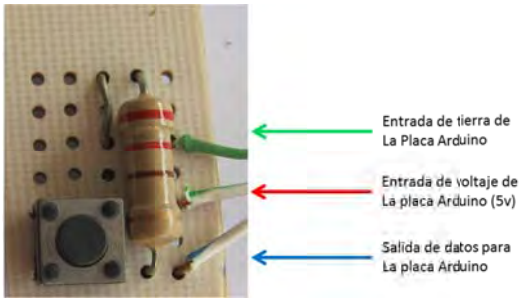

Figura 35 Circuito de re-inicialización de la alarma.

El segundo circuito está encargado de señalizar al usuario que la velocidad del viento ha excedido el límite pre-establecido durante cierto tiempo y que potencialmente puede ocasionar algún daño estructural. El LED es conectado mediante dos hilos de cable par trenzado, el primero es para la tierra y el segundo es el encargado de energizar al diodo, como se muestra en la figura 4.6. Los dos hilos están conectados directamente al servidor web.

Figura 36 Conexión del LED activado por la alarma.

En la Figura 4.7 se observa el cableado desde los sensores hacia la placa Arduino. En total se necesitó de siete hilos de cable.

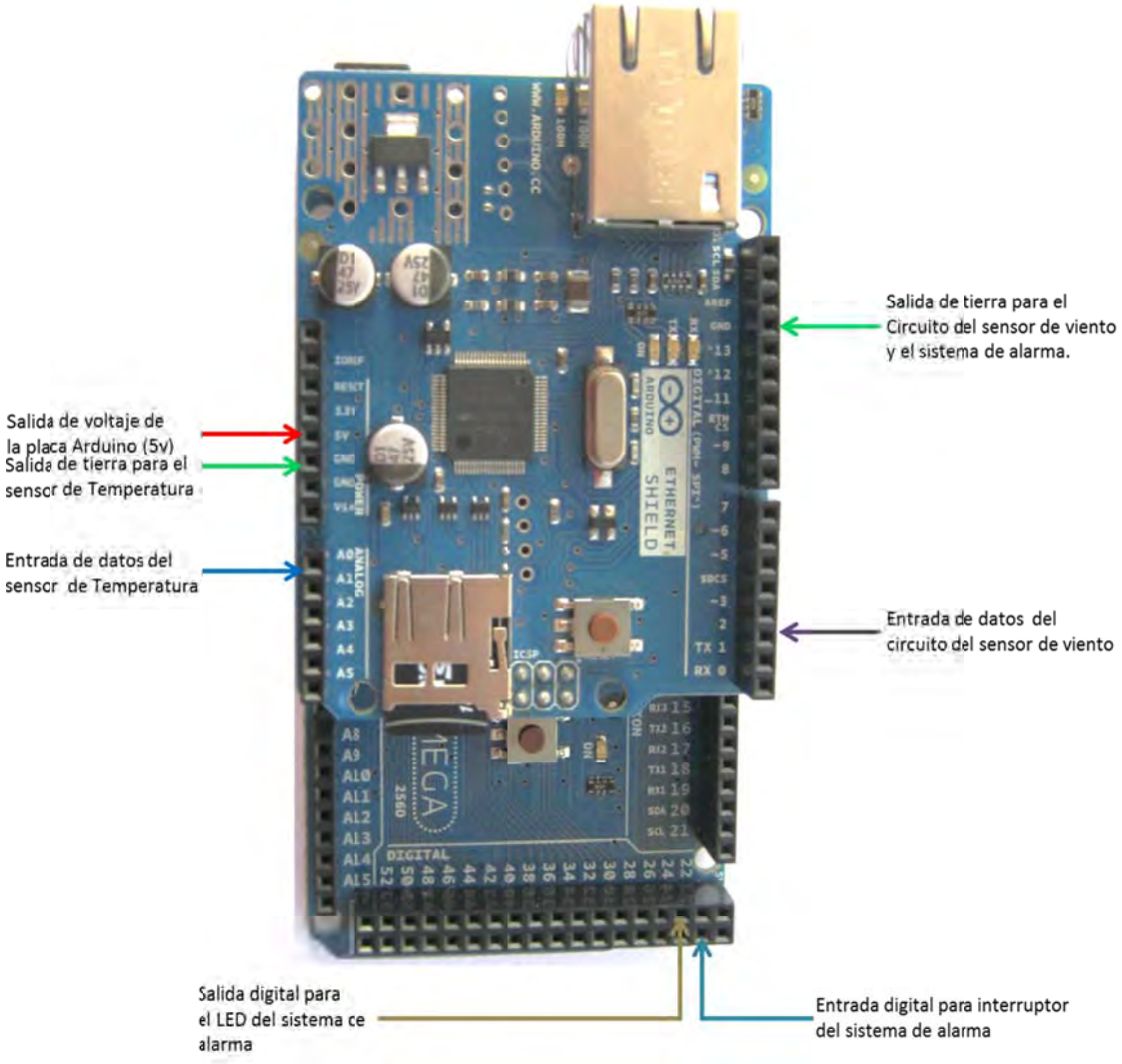

Figura 37 Cableado Arduino Mega.

4.3 Pruebas de funcionamiento del Servidor Web con los sensores.

En esta sección se muestran los resultados obtenidos del sistema de adquisición de datos con servicio web implementado en la tarjeta Arduino Mega 2560.

Para tener acceso a la página del servidor web, se necesita escribir en la barra de direcciones del buscador la dirección IP 192.168.0.253, con lo cual se desplegará en la pantalla de manera desglosada la información generada por el sistema de adquisición de datos, tal como muestra la Figura 4.8. En esta página se pueden observar los valores actuales de temperatura y velocidad del viento actual así como los archivos .CSV creados con sesiones anteriores en el servidor web y que se encuentran almacenados en la tarjeta microSD.

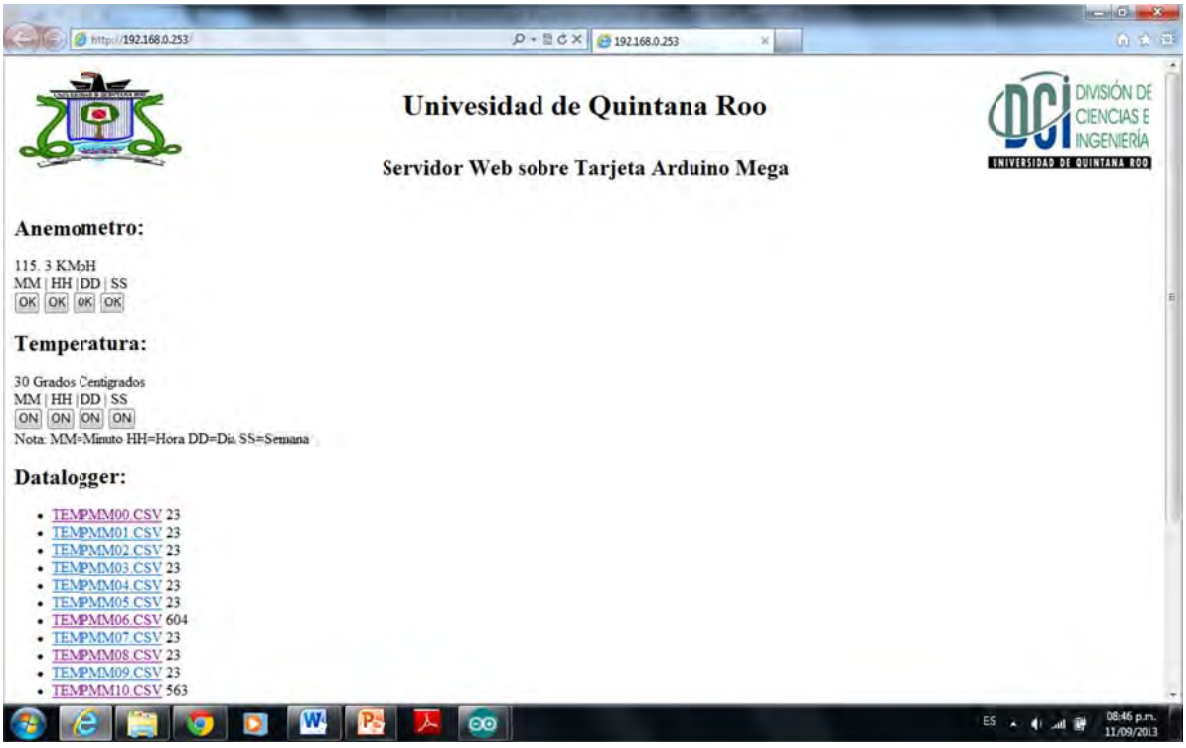

Figura 38 Pantalla principal del servidor web.

Para crear un nuevo archivo, el servidor web cuenta con cuatro botones para cada sensor previamente configurados para un tiempo en específico de adquisición. Cuando el nuevo archivo es creado, se agrega a la lista con un nuevo nombre referente hacia el tipo de variable monitoreada.

A continuación en la Figura 4.9, se observa una gráfica generada con los valores obtenidos del sensor durante un periodo de tiempo de una hora. En la gráfica se muestra los valores de la velocidad del viento registrados durante un lapso de tiempo.

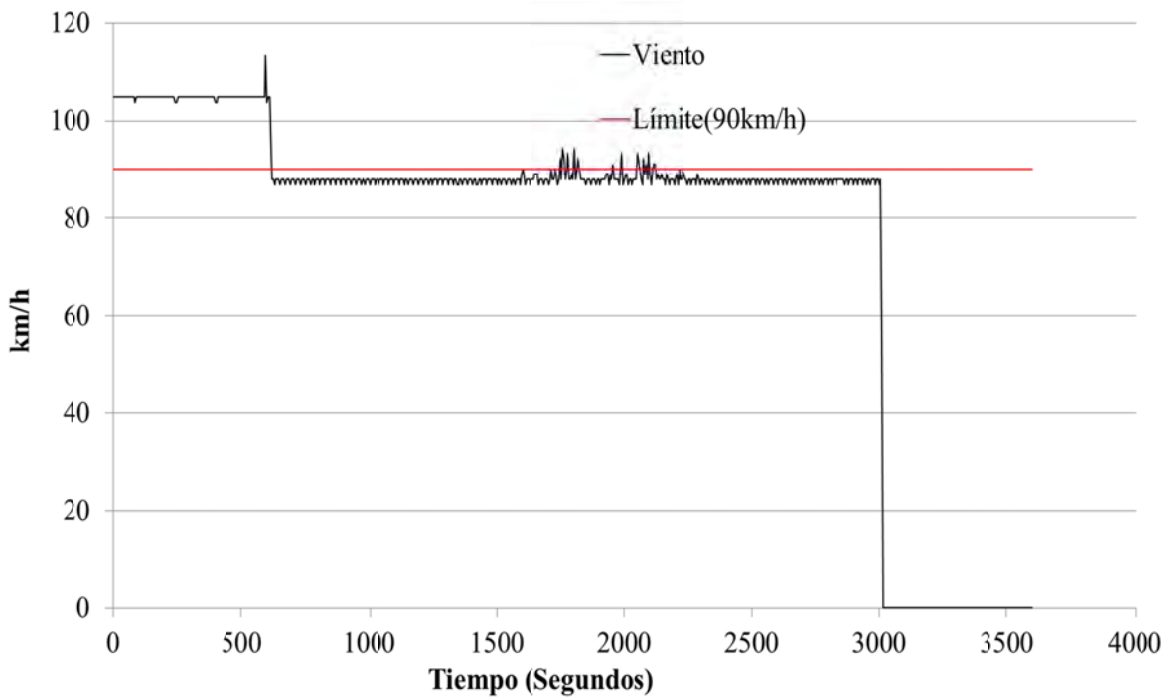

Figura 39 Gráfica del sensor de viento.

Los valores registrados fueron obtenidos cada cinco segundos y la velocidad del viento fue generada por un ventilador de 12 volts para simular rachas de aire de forma continua.

La medición de temperatura fue realizada en el interior del aula de Física III. La Figura 4.10 muestra la gráfica con los valores de temperatura registrados por el sistema de adquisición de datos. Como se aprecia en la gráfica, la temperatura tiene un comportamiento variable lo cual es debido a que el laboratorio de Física III cuenta con un sistema de aire acondicionado, por lo que la temperatura oscila entre los límites de temperatura del aire acondicionado. La medición de temperatura se realizó en un periodo de tiempo de una hora con una frecuencia de muestreo de un segundo.

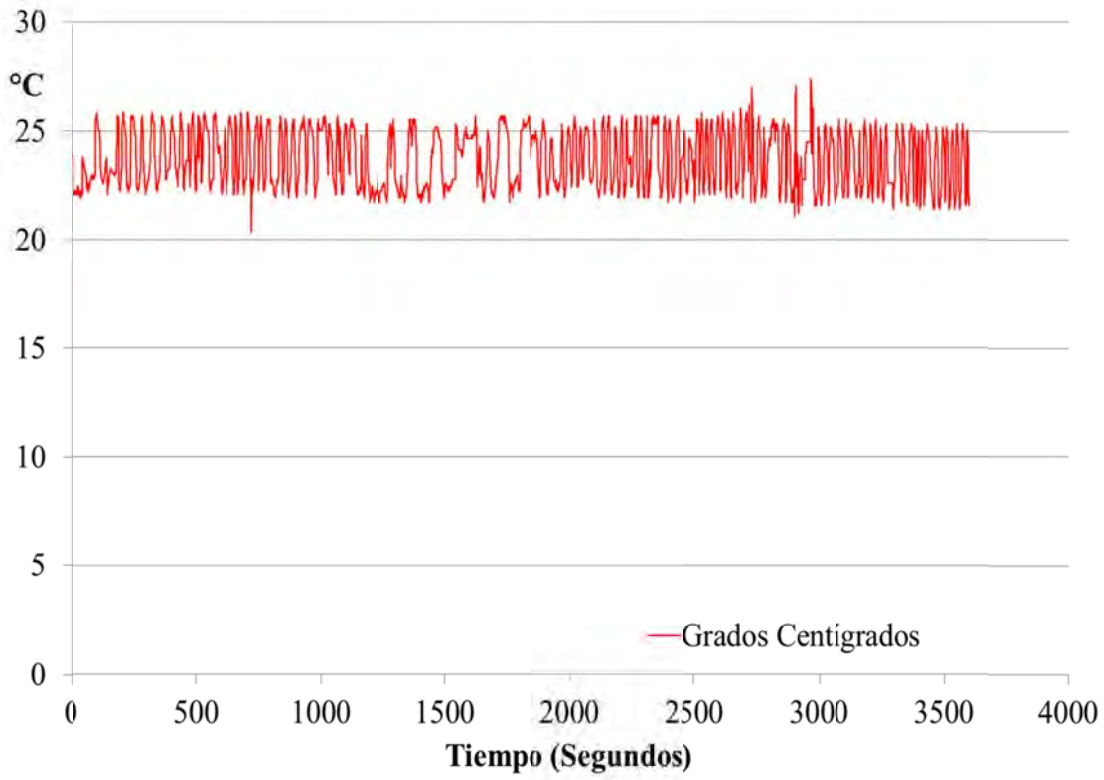

Figura 40 Gráfica del sensor de temperatura.

4.4 Pruebas de funcionamiento del Sistema Automático de Alarma y Prevención.

El sistema de alarma está integrado al servidor web pero no requiere de un cliente para realizar su tarea, es decir que la alarma funciona de manera automática desde que enciende la tarjeta Arduino con los sensores y se le conecta una fuente de alimentación.

En la Figura 4.11 se aprecia que además de la pantalla principal del servidor web existe una ventana a un costado con el logotipo de Arduino, esta ventana es una herramienta con la que cuenta la Interfaz de Desarrollo, en la que se visualiza de manera instantánea lo que ocurre con el servidor web y la alarma, es decir cada proceso que ocurre en el puerto físico USB (puerto serial para el sistema operativo) al que se encuentra conectado.

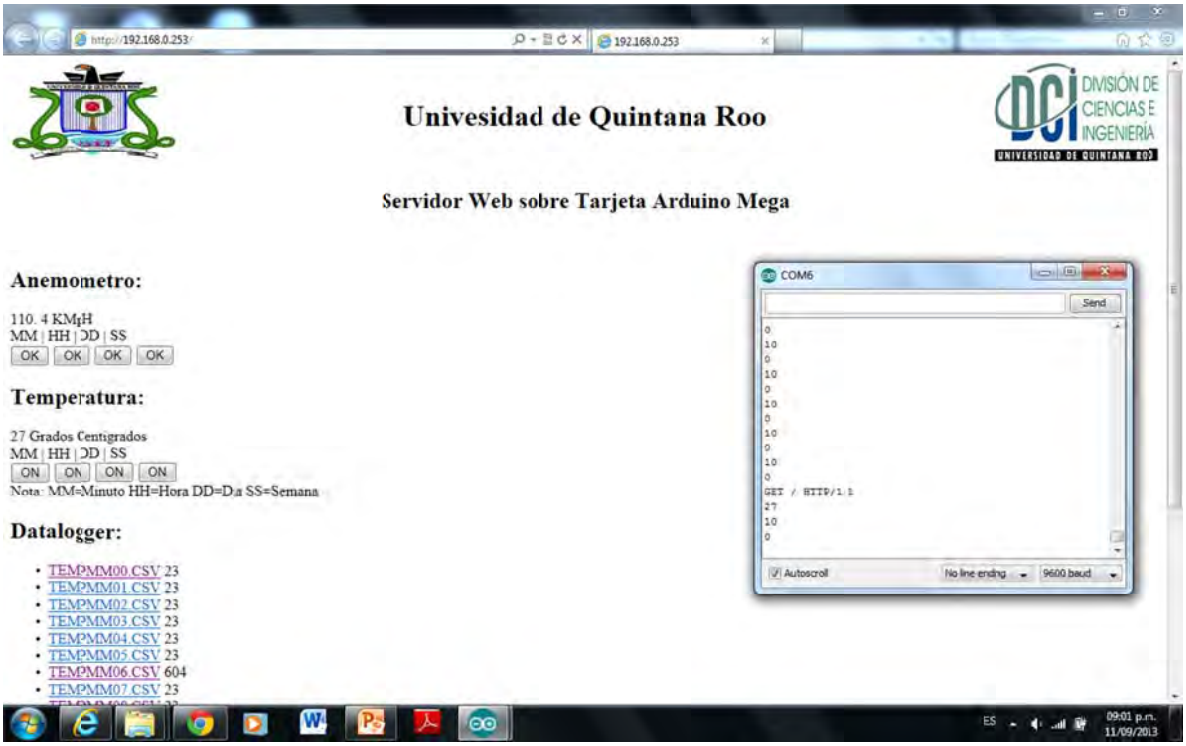

Figura 41 Servidor web y Puerto serial.

En la Figura 4.11 se observa cuando un nuevo cliente tiene acceso al servidor web, en el monitor del puerto serial aparece la leyenda GET / HTTP/1.1. El periodo de tiempo para la activación de la alarma fue de 10 s, por esta razón se aprecia la presencia del número 10 mucho antes de que se iniciara una conexión, este valor indicaba que el sistema de alarma ya estaba detectando altas velocidades en el viento, es decir que el LED ya estaba encendido, dado que las ráfagas fueron constantes en un intervalo de tiempo y que habían pasado del límite previamente establecido.

El sistema automático de alarma y prevención se reinicia de dos formas, en la Figura 4.12 se realiza el proceso por medio del interruptor por lo que ahora se observa en el monitor del puerto serial, que el número 10 es remplazado por un nuevo valor 1.

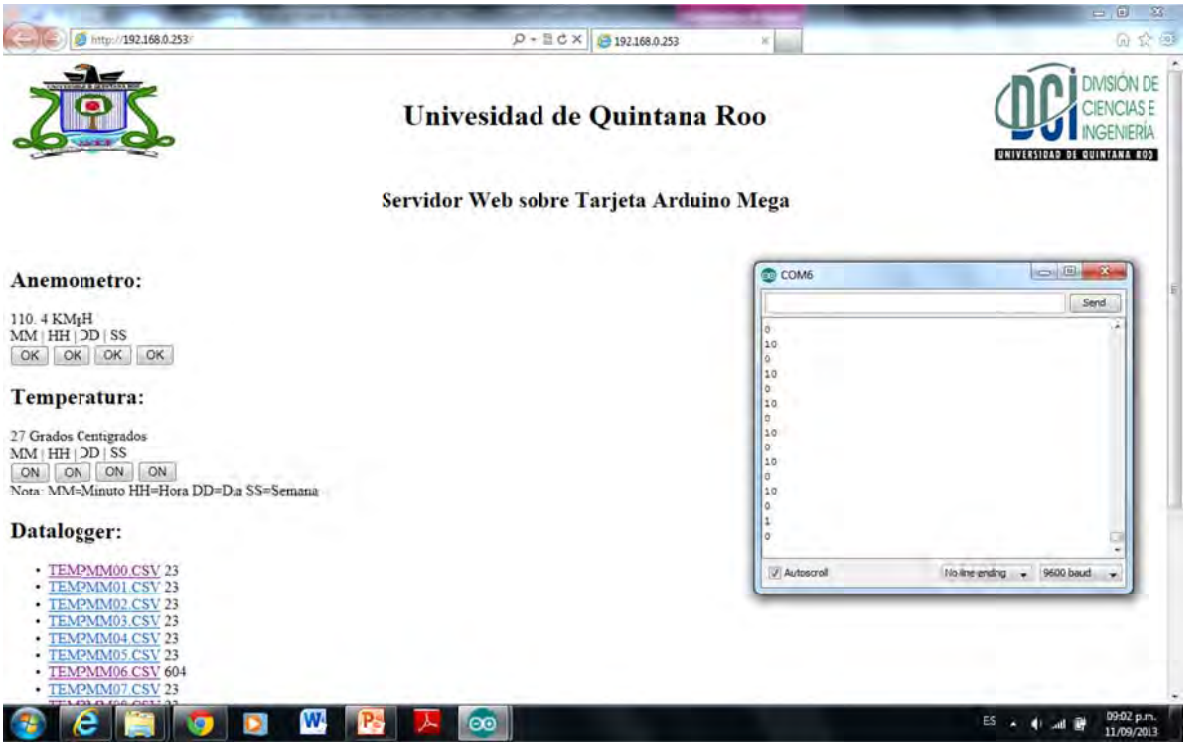

Figura 42 Servidor web y Puerto serial 2.

Si la velocidad del viento no sobrepasa el límite en el momento de que se reinicia la alarma, entonces el valor inicial seria 0 y aumentaría una unidad hasta que una ráfaga de viento sea registrada de forma constante.

En la gráfica de la Figura 4.13 se observa el comportamiento del sistema de alarma funcionando de manera automática. La línea de color violeta representa el estado de la alarma. De esta forma, se observa como la alarma es activada al inicio de la gráfica debido a que la velocidad de viento ha superado el límite pre-establecido durante un cierto lapso de tiempo. Por otra parte, también se muestra en esta misma gráfica que aunque la velocidad de viento exceda el límite pre-establecido pero por una corta duración, entonces la alarma no se activa con lo cual se discrimina el hecho de la velocidad del viento se haya incrementado por una racha de corta duración.

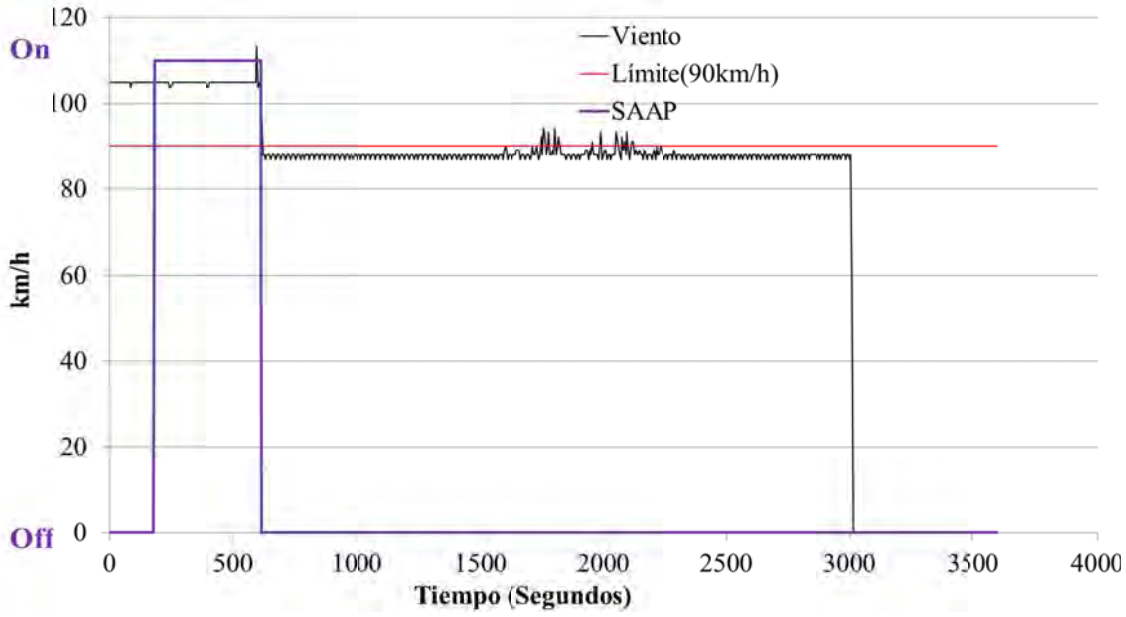

Figura 43 Gráfica del sistema de alarma.

CAPÍTULO 5

CAPÍTULO 5

CONCLUSIONES

Las conclusiones se obtuvieron a través de la experiencia adquirida con el desarrollo del proyecto, y de las pruebas de cada uno de los dispositivos. Con base en los resultados experimentales obtenidos en el desarrollo de este trabajo se demostró la factibilidad de desarrollar un sistema de adquisición y registro de datos con servicio web en un sistema embebido de plataforma abierta. Además, se presentó el diseño e implementación de un sistema de alarma que opera de manera conjunta con el sistema de adquisición de datos lo que aporta una característica adicional al sistema desarrollado en este trabajo.

De igual forma, se presentó una investigación del funcionamiento y operación de los microcontroladores Atmel en búsqueda de las capacidades y ventajas con las que este microcontrolador cuenta con respecto a otros existentes en el mercado. Dentro de las familias expuestas, desde un punto de vista educacional de aprendizaje, es más conveniente el uso de microcontroladores AVR, por los periféricos que incluye a un costo bajo.

La plataforma Arduino es una herramienta diseñada para crear prototipos de manera práctica y sencilla en un corto periodo de tiempo, que no exige un conocimiento a detalle del funcionamiento de microcontroladores; así como la manera simple de conectar sensores y su bajo consumo de energía.

En este mismo sentido, el software de la plataforma Arduino es un lenguaje de programación estructurado de alto nivel, que brinda muchas ventajas con respecto a cualquiera de los otros lenguajes por su fácil comprensión y similitud con los más conocidos, además de que la aplicación cuenta con una licencia software libre, es decir que no tiene ningún costo. Por esta misma razón, existe una contribución importante de una comunidad de desarrolladores en sistemas embebidos con funciones ya programadas y que permiten una reducción en la curvas de aprendizaje del lenguaje de programación en la plataforma Arduino.

La aportación de este trabajo consiste en la integración del sistema de adquisición y registro de datos con la funcionalidad de un servidor web y la implementación de una alarma que puede ejecutar una tarea de automatización.

REFERENCIAS BIBLIOGRÁFICAS

[1] *AVR Microcontrollers for High-Performance and Power-Efficient 8-bit Processing;* **Atmel Corporation.** San José California, USA 2013.

[2] *Batalla de microcontroladores ¿AVR o PIC?* **Cortez, Omar Otoniel Flores.** El Salvador : s.n., 2009.

[3] Atmel Reconfirms 8-bit MCU Leadership, Expands MCU Portfolio with Low-power 8-bit tinyAVR MCUs; **Atmel Corporation**. San José California, USA 2013.

[4] **Atmel.** http://www.atmel.com/. [En línea] [Citado el: 11 de Enero de 2013.] http://www.atmel.com/products/microcontrollers/avr/default.aspx.

[5]. **Arduino.** http://arduino.cc/en/. [En línea] [Citado el: 11 de Enero de 2013.] http://arduino.cc/en/Main/arduinoBoardMega2560.

[6] Hoja de Datos del ATmega640/1280/1281/2560/2561; **Atmel Corporation**. San José California, USA 2013.

[7] Hoja de Datos W5100 (WIZnet Hardwired TCP/IP Embedded Ethernet Controller); **Wiznet Technology**. Korea, 2013.

[8] LM35 Precision Centigrade Temperature Sensors; **Texas Instruments**. Texas, USA, 2000.

[9] **Barret, Steven F.** *Arduino microcontroller processing for everyone part II.* s.l. : Morgan & Claypool, 2010.

[10] **Arduino.** http://arduino.cc/es/Reference/HomePage. [En línea] [Citado el: 11 de Enero de 2013.] http://arduino.cc/es/Reference/Libraries.

[11] **Sá, Denilson Figueiredo de.** http://my.opera.com/CrazyTerabyte. [En línea] 8 de Septiembre de 2011. [Citado el: 11 de Enero de 2013.] http://my.opera.com/CrazyTerabyte/blog/2011/09/08/comparison-between-microchip-pic-and-atmel-avr-microcontrollers.

ANEXO A

```
#include <SdFat.h>
#include <SdFatUtil.h>
#include <Ethernet.h>
#include <SPI.h>


String readString;
// How big our line buffer should be. 100 is plenty!
#define BUFSIZ 100

#define SENSOR_COUNT     3 // number of analog pins to log
#define ECHO_TO_SERIAL   1 // echo data to serial port

///var Anemómetro
#define uint  unsigned int
#define ulong unsigned long
#define PIN_ANEMOMETER  2     // Digital 2
// How often we want to calculate wind speed or direction
#define MSECS_CALC_WIND_SPEED 5000
volatile int numRevsAnemometer = 0; // Incremented in the interrupt
ulong nextCalcSpeed;             // When we next calc the wind speed
ulong time;                      // Millis() at each start of loop().


int limite = 100;   //Limite de Velocidad para encender el LED.
```

```
int cont = 0;       //contador que almacena el numero de veces se excede el limite de velocidad.
int lmcont = 10;    //limite de conteo de velocidad excedida en tiempo real sobre el servidor Web.
int lcont = 10;    //limite de conteo de velocidad excedida en tiempo real sobre anemometro.

/************ ETHERNET STUFF ************/
byte mac[] = {   0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = {   192, 168, 0, 253 };
byte gateway[] = {   192, 168, 0, 1 };
byte subnet[] = {   255, 255, 255, 0 };
EthernetServer server(80);

/************ SDCARD STUFF ************/
Sd2Card card;
SdVolume volume;
SdFile root;
SdFile file;

// store error strings in flash to save RAM
#define error(s) error_P(PSTR(s))

void error_P(const char* str) {
  PgmPrint("error: ");
  SerialPrintln_P(str);
  if (card.errorCode()) {
    PgmPrint("SD error: ");
    Serial.print(card.errorCode(), HEX);
    Serial.print(',');
    Serial.println(card.errorData(), HEX);
```

```
  }
  while(1);
}

/*
 * Write CR LF to a file
 */
void writeCRLF(SdFile &f)
{
  f.write((uint8_t *)"\r\n", 2);
}
/*
 * Write an unsigned number to a file
 */
void writeNumber(SdFile &f, uint32_t n)
{
  uint8_t buf[10];
  uint8_t i = 0;
  do {
    i++;
    buf[sizeof(buf) - i] = n%10 + '0';
    n /= 10;
  }
  while (n);
  f.write(&buf[sizeof(buf) - i], i);
}
/*
 * Write a string to a file
 */
void writeString(SdFile &f, char *str)
```

```
{
  uint8_t n;
  for (n = 0; str[n]; n++);
  f.write((uint8_t *)str, n);
}

void setup() {
  Serial.begin(9600);
  pinMode(23, OUTPUT);   //configuracion de pin digital para encendido de limite de
velocidad
  pinMode(24, OUTPUT);   //configuracion de pin digital para encendido de datalogger
  pinMode(26, INPUT);    //configuracion de pin digital para lectura de pushBoton de reset.
  ////inicio de pin 2\\\\
  pinMode(PIN_ANEMOMETER, INPUT);
  digitalWrite(PIN_ANEMOMETER, HIGH);
  attachInterrupt(0, countAnemometer, FALLING);
  nextCalcSpeed = millis() + MSECS_CALC_WIND_SPEED;
  PgmPrint("Free RAM: ");
  Serial.println(FreeRam());
  // initialize the SD card at SPI_HALF_SPEED to avoid bus errors with
  // breadboards.  use SPI_FULL_SPEED for better performance.
  pinMode(53, OUTPUT);                  // set the SS pin as an output (necessary!)
  digitalWrite(53, HIGH);               // but turn off the W5100 chip!

  if (!card.init(SPI_FULL_SPEED, 4)) error("card.init failed!");

  // initialize a FAT volume
  if (!volume.init(&card)) error("vol.init failed!");

  PgmPrint("Volume is FAT");
```

```
  Serial.println(volume.fatType(),DEC);
  Serial.println();

  if (!root.openRoot(&volume)) error("openRoot failed");

  // list file in root with date and size
  PgmPrintln("Files found in root:");
  root.ls(LS_DATE | LS_SIZE);
  Serial.println();


  PgmPrintln("Files found in all dirs:");
  root.ls(LS_R);

  Serial.println();
  PgmPrintln("Done");

  // Inicio del servidor web!
  Ethernet.begin(mac, ip, gateway, subnet);
  server.begin();

}

void ListFiles(EthernetClient client, uint8_t flags) {


  dir_t p;

  root.rewind();
  client.println("<ul>");
```

```
while (root.readDir(p) > 0) {
  // done if past last used entry
  if (p.name[0] == DIR_NAME_FREE) break;

  // skip deleted entry and entries for . and  ..
  if (p.name[0] == DIR_NAME_DELETED || p.name[0] == '.') continue;

  // only list subdirectories and files
  if (!DIR_IS_FILE_OR_SUBDIR(&p)) continue;

  // print any indent spaces
  client.print("<li><a href=\"");
  for (uint8_t i = 0; i < 11; i++) {
    if (p.name[i] == ' ') continue;
    if (i == 8) {
      client.print('.');
    }
    client.print((char)p.name[i]);
  }
  client.print("\">");

  // print file name with possible blank fill
  for (uint8_t i = 0; i < 11; i++) {
    if (p.name[i] == ' ') continue;
    if (i == 8) {
      client.print('.');
    }
    client.print((char)p.name[i]);
  }
```

```
  client.print("</a>");

  if (DIR_IS_SUBDIR(&p)) {
    client.print('/');
  }

  // print modify date/time if requested
  if (flags & LS_DATE) {
    root.printFatDate(p.lastWriteDate);
    client.print(' ');
    root.printFatTime(p.lastWriteTime);
  }
  // print size if requested
  if (!DIR_IS_SUBDIR(&p) && (flags & LS_SIZE)) {
    client.print(' ');
    client.print(p.fileSize);
  }
  client.println("</li>");
 }
 client.println("</ul>");
}
void loop()
{
 loopa();
 loopb();
}
void (*resetFunc)(void)=0x0;      //funcion que reincia el codigo
void loopa(){
 char clientline[BUFSIZ];
 int index = 0;
```

```
EthernetClient client = server.available();
if (client) {

 // an http request ends with a blank line
 boolean current_line_is_blank = true;

 // reset the input buffer
 index = 0;

 while (client.connected()) {
  if (client.available()) {
    char c = client.read();

    //read char by char HTTP request
    if (readString.length() < 100) {

      //store characters to string
      readString += c;
      //Serial.print(c);
    }
    int pin = 0; // analog pin
    int tempc = 0; // temperature variables
    int samples[5]; // variables to make a better precision
    int i;
    int x, iSpeed; //variable de velocidad del viento


    if (c != '\n' && c != '\r') {
      clientline[index] = c;
      index++;
```

```
        if (index >= BUFSIZ)
          index = BUFSIZ -1;


        continue;
      }

      // got a \n or \r new line, which means the string is done
      clientline[index] = 0;

      // Print it out for debugging
      Serial.println(clientline);


      if (strstr(clientline, "GET / ") != 0) {
        // send a standard http response header
        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println();
        client.print("<p><img src='http://redturismo.uqroo.mx/Uqroo%20logo.jpg'
width='180' height='108' align='left' /></p>");
        client.println("<p><img src='http://anca.uqroo.mx/Images/Picture3.jpg' width='180'
height='108' align='right'/></p>");
        client.println("<br />");
        client.println(F("<h1 align='center'>Univesidad de Quintana Roo</h1>"));
        client.println("<br />");
        client.println(F("<h2 align='center'>Servidor Web sobre Tarjeta Arduino
Mega</h2>"));
        client.println("<br />");
```

```
        client.println(F("<h2>Anemometro:</h2>"));
        if (i<10){
          delay(1000);
          long speed = 23872;
          speed *= numRevsAnemometer;
          speed /= MSECS_CALC_WIND_SPEED;
          iSpeed = speed;
          x = (iSpeed / 10) *5;
          client.print(x);
          client.println(".");
          x = iSpeed % 10 ;
          client.print(x);
          client.println(" KMpH");
          numRevsAnemometer = 0;        // Reset counter
          i++;
        }
        client.println("<br />");
        client.println("MM | HH | DD | SS");
        client.println("<br />");
        client.print("<input type=\"button\" value=\"OK\" onmousedown=\"location.href
('/OK1~1.HTM');\"/>");
        client.print("    ");
        client.print("<input type=\"button\" value=\"OK\" onmousedown=\"location.href
('/OK2~1.HTM');\"/>");
        client.print("    ");
        client.print("<input type=\"button\" value=\"OK\" onmousedown=\"location.href
('/OK3~1.HTM');\"/>");
        client.print("    ");
        client.print("<input type=\"button\" value=\"OK\" onmousedown=\"location.href
('/OK4~1.HTM');\"/>");
```

```
        client.println(F("<h2>Temperatura:</h2>"));
        if(analogRead(0)>0){
          for(i = 0;i<=4;i++){
            samples[i] = ( 5.0 * analogRead(pin) * 100.0) / 1024.0;
            tempc = tempc + samples[i];
            delay(100);
          }
          tempc = tempc/5.0; // better precision
          Serial.println(tempc);
          client.print(tempc);
          client.println(" Grados Centigrados");
          tempc = 0;
          delay(100); // delay before loop
        }
        client.println("<br />");
        client.println("MM | HH | DD | SS");
        client.println("<br />");
        client.print("<input type=\"button\" value=\"ON\" onmousedown=\"location.href
('ON1~1.HTM');\"/>");
        client.print("   ");
        client.print("<input type=\"button\" value=\"ON\" onmousedown=\"location.href
('ON2~1.HTM');\"/>");
        client.print("   ");
        client.print("<input type=\"button\" value=\"ON\" onmousedown=\"location.href
('ON3~1.HTM');\"/>");
        client.print("   ");
        client.print("<input type=\"button\" value=\"ON\" onmousedown=\"location.href
('ON4~1.HTM');\"/>");
        client.println("<br />");
```

```
  client.println("Nota:");
  client.println("MM=Minuto");
  client.println("HH=Hora");
  client.println("DD=Dia");
  client.println("SS=Semana");
  // print all the files, use a helper to keep it clean
  client.println(F("<h2>Datalogger:</h2>"));
  ListFiles(client, LS_SIZE);
 }
 else if (strstr(clientline, "GET /") != 0) {
  // this time no space after the /, so a sub-file!
  char *filename;

  filename = clientline + 5; // look after the "GET /" (5 chars)
  // a little trick, look for the " HTTP/1.1" string and
  // turn the first character of the substring into a 0 to clear it out.
  (strstr(clientline, " HTTP"))[0] = 0;

  // print the file we want
  Serial.println(filename);

  if (! file.open(&root, filename, O_READ)) {
   client.println("HTTP/1.1 404 Not Found");
   client.println("Content-Type: text/html");
   client.println();
   client.println("<h2>File Not Found!</h2>");
   break;
  }

  Serial.println("Opened!");
```

```
    client.println("HTTP/1.1 200 OK");
    client.println("Content-Type: text/plain");
    client.println();

    int16_t c;
    while ((c = file.read()) > 0) {
      // uncomment the serial to debug (slow!)
      //Serial.print((char)c);
      client.print((char)c);
    }
    file.close();
  }
  else {
    // everything else is a 404
    client.println("HTTP/1.1 404 Not Found");
    client.println("Content-Type: text/html");
    client.println();
    client.println("<h2>File Not Found!</h2>");
  }
  break;
 }
}
// give the web browser time to receive the data
delay(1);
client.stop();
int value=0;//cantidad de segundos por cada ciclo en temperatura
int value2=0;//cantidad de segundos por cada ciclo en anemometro
int n=0; //numero de veces en que se repetira el ciclo
if (( readString.indexOf("ON1") > 0) ){
```

```
  value=60;
  n=1;
  }//para 60 segundos
  if (( readString.indexOf("ON2") > 0) ){
  value=3600;
  n=1;
  }//para 1 hora
  if (( readString.indexOf("ON3") > 0) ){
  value=3600;
  n=24;
  }//para 1 dia
  if (( readString.indexOf("ON4") > 0) ){
  value=3600;
  n=168;
  }//para 1 semana
  if(value>0&&n>0){
    Serial.println("anemometro");
    char name[] = "TEMPMM00.CSV";
    for (uint8_t i = 0; i < 100; i++) {
      name[6] = i/10 + '0';
      name[7] = i%10 + '0';
      if (file.open(root, name, O_CREAT | O_EXCL | O_WRITE)) break;
    }
    if (!file.isOpen()) error ("file.create");
    Serial.print("Logging to: ");
    Serial.println(name);
    file.writeError = 0;
    file.print("SEG");
#if ECHO_TO_SERIAL
    Serial.print("SEG");
```

```
#endif //ECHO_TO_SERIAL
#if SENSOR_COUNT > 10
#error SENSOR_COUNT too large
#endif //SENSOR_COUNT
    for (uint8_t i = 0; i < SENSOR_COUNT; i++) {
      file.print(",sens");
      file.print(i, DEC);
#if ECHO_TO_SERIAL
      Serial.print(",sens");
      Serial.print(i, DEC);
#endif //ECHO_TO_SERIAL
    }
    file.println();
#if ECHO_TO_SERIAL
    Serial.println();
#endif  //ECHO_TO_SERIAL
    if (file.writeError || !file.sync()) {
      error("write header");
    }
    int x,z;
    z=n;
    x=value;
    Serial.println(x);
    Serial.println(z);
    for(int a=0;a!=n;a++){
      for( int y=0; x!=y; y++ ){
        digitalWrite(24,HIGH);
        delay(1000);
        writeNumber(file, y);
        writeString(file, " , ");
```

```
        writeNumber(file, ( 5.0 * analogRead(0) * 100.0) / 1024.0);
        writeCRLF(file);
        digitalWrite(24,LOW);
        Serial.println(y);
      }
      Serial.print("a=");
      Serial.println(a);
    }
    file.close();
    resetFunc();
  }
  if (( readString.indexOf("OK1") > 0) ) {
     value2=12;
     n=1;
  }//1 minuto
  if (( readString.indexOf("OK2") > 0) ) {
     value2=720;
     n=1;
  }//1 hora
  if (( readString.indexOf("OK3") > 0) ) {
     value2=720;
     n=24;
  }//1 dia
  if (( readString.indexOf("OK4") > 0) ) {
     value2=720;
     n=168;
  }//1 semana
 if(value2>0 && n>0){
   Serial.println("anemometro");
int x, iSpeed;
```

```
 // create a new file
 char name[] = "VIENTO00.CSV";
 for (uint8_t i = 0; i < 100; i++) {
  name[6] = i/10 + '0';
  name[7] = i%10 + '0';
  if (file.open(root, name, O_CREAT | O_EXCL | O_WRITE)) break;
 }
 if (!file.isOpen()) error ("file.create");
 Serial.print("Logging to: ");
 Serial.println(name);
 // write header
 file.writeError = 0;
 file.print("SEG");
#if ECHO_TO_SERIAL
 Serial.print("SEG");
#endif //ECHO_TO_SERIAL
 file.print(",M/H");
#if ECHO_TO_SERIAL
 Serial.print("ANEMOMETER");
#endif //ECHO_TO_SERIAL
 file.println();
#if ECHO_TO_SERIAL
 Serial.println();
#endif  //ECHO_TO_SERIAL
 if (file.writeError || !file.sync()) {
  error("write header");
 }
 int y,z;
 y=n;
 z=value2;
```

```
    Serial.println(y);
    Serial.println(z);
 for(int a=0;a!=y;a++){
    for( int i=0; i!=z; i++ ){
      digitalWrite(24,HIGH);
      delay(5000);
      writeNumber(file, (i * 5));
      writeString(file, " , ");
      long speed = 23872;
      speed *= numRevsAnemometer;
      speed /= MSECS_CALC_WIND_SPEED;
      iSpeed = speed;         // Need this for formatting below
      x = iSpeed * 10 ;
      writeNumber(file, x);
      writeCRLF(file);
      digitalWrite(24,LOW);
      Serial.println(i);
      numRevsAnemometer = 0;        // Reset counter
    }
      Serial.print("a=");
      Serial.println(a);
  }
 file.close();
 resetFunc();
}
   readString="";
 }
   else {
   int x, iSpeed, i;
   int val=0;
```

```
val = digitalRead(26);
if ( i <10 && val == 0 ){
  delay(5000);
  long speed = 23872;
  speed *= numRevsAnemometer;
  speed /= MSECS_CALC_WIND_SPEED;
  iSpeed = speed;         // Need this for formatting below
  x = iSpeed / 10 ;
  if (x>limite)
    cont++;
  else if(x<limite){
    if (cont == 0){
      cont = 0;
    }
    else if(cont > 0){
      cont--;
    }
  }
  if(cont>=lcont){
    if (cont >= lcont){
      cont = lcont;
    }
    digitalWrite(23,HIGH);
  }
  else if (cont!=lcont)digitalWrite(23,LOW);
  Serial.println(cont);
  Serial.println(val);
  numRevsAnemometer = 0;        // Reset counter
  i++;
```

```
    }
    else if(val>0)cont=0;

  }
}

void countAnemometer() {
  numRevsAnemometer++;
}

void loopb(){
  //////////Anemometro\\\\\\\\
  time = millis();

  if (time >= nextCalcSpeed) {
    loopa();
    nextCalcSpeed = time + MSECS_CALC_WIND_SPEED;
  }
}
```