Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Interfaz de adquisición de datos de una red de sensores por medio de un sistema embebido basado en microcontrolador PIC32 con comunicación USB

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

José Pablo Vernavá Amador

Carlos García Ramírez

Cartago, Noviembre 2010

INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENERÍA ELECTRÓNICA

PROYECTO DE GRADUACIÓN

TRIBUNAL EVALUADOR

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

Miembros del Tribunal

Ing. Johan Carvajal Godínez

Profesor lector

Ing. Néstor Hernández

Profesor lector

Ing. Marvin Hernández

Profesor asesor

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

Cartago. Noviembre 2010

Declaratoria de autenticidad

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

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

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

Cartago, Noviembre 2010

José Pablo Vernavá Amador

Ced. 1-1237-0248

Carlos García Ramírez

Ced. 1-1006-0263

RESUMEN

La recolección de datos de una red de sensores con el fin de medir variables climatológicas para la agricultura puede convertirse en una herramienta importante y útil en el desarrollo de planes de cultivo eficientes. Al tener una base de datos sobre el comportamiento del clima en una zona en específico, llámese microclima, es posible incrementar la productividad de esa zona y obtener un mejor aprovechamiento de los recursos disponibles tanto humanos como naturales.

Este documento presenta una descripción analítica del desarrollo de un sistema embebido basado en microcontrolador PIC32 para la adquisición de datos de una red de sensores principalmente de variables climatológicas. El sistema embebido es capaz de recolectar los datos de la red de sensores y transmitirlos por medio de un puerto USB a una computadora, en la cual el usuario pueda crear su propia base de datos y a la vez sea capaz de visualizarlos.

Entre las características de dicho sistema se encuentra su bajo costo, lo cual lo hace una solución viable para pequeños y medianos productores que no poseen una herramienta que les ayude a mejorar la producción, calidad y resistencia de una cosecha.

Palabras clave: conversión analógico-digital, microclima, microcontrolador PIC32, puerto USB, red de sensores, sistema embebido, variables climatológicas

ABSTRACT

The acquisition of data from a network of sensors to measure weather variables for agriculture can become an important and useful tool to develop efficient cropping plans. By having a database of weather patterns in a specific area, called microclimate, it is possible to increase the productivity of that area and obtain a better use of available resources both human and natural.

This document presents an analytic description of the development of an embedded system based on PIC32 microcontroller to acquire data from a sensor network composed mainly of weather type sensors. The embedded system is capable of collecting and transmitting data through a USB port to a computer, in which the user can create their own database and may be able to view the data.

Among the features of this system is its low cost, making it a viable solution for small and medium producers who do not have a tool to help them improve production, quality and resistance of a crop.

Keywords: analog-digital conversion, climatic variables, embedded system, microclimate, PIC32 microcontroller, sensor network, USB port

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1: Introducción

En este se expone el contexto en el cual se desarrollo el proyecto: el problema existe, la importancia de la solución y detalles sobre la solución empleada.

El proyecto D2ARS (“Diseño y Desarrollo de Aplicaciones de Redes de Sensores”) ha sido llevado a cabo por un consorcio de universidades de España, México, Colombia, Brasil, Cuba y Costa Rica. En nuestro país, el proyecto se encuentra bajo la coordinación del Dr. Pablo Alvarado, de la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica, y está inscrito ante la Vicerrectoría de Investigación del ITCR (proyecto 5402 1360 1701).

El objetivo general del proyecto D2ARS es el desarrollo de una red de sistemas empotrados de tiempo real que abstraiga, controle y garantice el comportamiento de un número significativo de sistemas "pequeños" o redes de sensores y actuadores. Estas redes de sensores y actuadores difieren de los sistemas tradicionales en red en que deben ser altamente adaptativos, resolver tasas relevantes de fallos de conexión, ser móviles y ser reconfigurables. Asimismo, tienen que exportar un conjunto de servicios parametrizados referidos a la coordinación de una gama amplia de tipos de sensores y actuadores. [1]

El área de aplicación de las redes de sensores es muy amplia. Ejemplos son: vigilancia del medio ambiente (bosques protegidos, actividades ilegales de tala y caza, incendios forestales), aplicaciones en salud (body sensor networks), milicia, rastreos en sistemas de inventario, rastreos en sistemas de control de tránsito, y otros [2].

1.1 Problema existente e importancia de su solución

Muchos de los productores agrícolas en nuestro país no poseen a su disposición herramientas tecnológicas que les brinden más información acerca del medio ambiente en que se encuentran sus plantaciones.

Por otro lado, los cultivos son sensibles a las condiciones climatológicas y estas pueden afectar su desarrollo y calidad, afectando directamente al productor. Por ejemplo, hay productos químicos que se deben aplicar a los cultivos dependiendo de las condiciones climatológicas para que se obtenga un mejor resultado y se optimicen recursos.

Hoy en día existen sistemas de monitoreo de variables climatológicas muy robustos, pero su costo es elevado por lo tanto en muchas ocasiones se hacen inaccesibles a los pequeños y medianos productores.

Con el desarrollo de este proyecto se brinda una opción de bajo costo para que los productores puedan obtener información del medio ambiente en que se encuentran sus cultivos.

Esto con el fin de tener información que los respalde a la hora de desarrollar planes de siembra y manejo de los cultivos y así obtener un mejor aprovechamiento de los recursos y mayor productividad por área sembrada.

Además, la visualización de los datos en tiempo real les permite a los productores tomar decisiones correctivas y preventivas en el instante para así evitar daños por:

- Plagas
- Temperaturas excesivas
- Exceso o falta de humedad en el ambiente
- Daños por ráfagas fuertes de viento, entre otras

Pero por otro lado, si las condiciones climáticas se prestan, el productor puede incrementar y diversificar sus cultivos, por lo cual la ganancia también se eleva.

1.2 Solución seleccionada

El sistema desarrollado consta de una red de sensores para medir las variables climatológicas, un sistema embebido basado en microcontrolador de arquitectura PIC32MX y una interfaz grafica de usuario.

La red de sensores se encarga de transformar las variables de instrumentación a señales eléctricas para que sean procesadas por el microcontrolador. Éste se encarga de obtener los valores de las señales y almacenarlos. Además, está en capacidad de transmitir estos datos a una computadora por medio del puerto USB cuando el usuario así lo indique.

Por medio de la interfaz grafica, el usuario controla el flujo de información entre computadora y el sistema embebido. Además, la interfaz le permite al usuario tener a su disposición permanentemente los datos obtenidos por la red de sensores, guardándolos en el disco duro en un formato xls que puede ser leído en otra computadora que no tenga instalada la interfaz.

De igual forma, la interfaz le permite al usuario visualizar gráficamente el comportamiento del clima en un lapso definido por el mismo.

En la Figura 1 se muestra un diagrama de bloques de la solución seleccionada.

Figura 1.1 Diagrama de bloques de la solución seleccionada

Por lo tanto, el sistema desarrollado brinda una herramienta al usuario para respaldar la toma de decisiones objetivas con información real y confiable del microclima.

Capítulo 2: Meta y Objetivos

2.1 Meta

Brindar al usuario una herramienta tecnológica con la cual analizar el medio ambiente en que se encuentran sus cultivos.

2.2 Objetivo General

Realizar un sistema basado en una red de sensores, un microcontrolador PIC32MX y una interfaz grafica de usuario capaz de obtener, guardar y visualizar datos climatológicos.

2.3 Objetivos Específicos

- Validar los sensores utilizados en la red con el fin de determinar y estudiar su comportamiento.

- Diseñar y acondicionar una red de sensores para que las señales se puedan acoplar a los requerimientos del sistema embebido.

- Desarrollar un programa en un lenguaje de alto nivel mediante el cual el microcontrolador PIC32 efectúe la conversión y almacenamiento periódico de los datos provenientes de la red de sensores.

- Desarrollar rutinas en un lenguaje de alto nivel para que el microcontrolador PIC32 sea capaz de transmitir los datos a una computadora por medio del puerto USB.

- Desarrollar una interfaz gráfica de usuario en una computadora que permita la recepción de los datos guardados en el microcontrolador vía puerto USB y la visualización de los mismos.

- Guardar de forma permanente los datos recibidos por la interfaz de usuario.

Capítulo 3: Marco Teórico

Toda medición exige tres funciones básicas: detección de la magnitud que se desea medir, procesamiento de las señales que contienen la información sobre dicha magnitud y presentación del resultado. Dado que actualmente las señales se procesan en último término con procesadores digitales y que las señales de salidas de los sensores, detectores o captadores son normalmente analógicas, hay que digitalizarlas. Si la información de las señales está en su amplitud, se digitalizan con convertidores analógico-digitales (CADs).

En muchas aplicaciones suele ser necesario acondicionar las señales de los sensores antes de digitalizarlas, debido a las limitaciones de la conversión analógica-digital y de los procesadores digitales. Por ejemplo, cuando hay que procesar señales alternas moduladas en amplitud, puede ser mejor desmodularlas primero y digitalizarlas después. También la utilización de filtros para evitar la aparición de frecuencias falsas (alias) o ruido, puede ser útil. La amplificación es también una de las funciones habituales en el acondicionamiento de señales. La linealización de un sensor o de un acondicionador de señal puede ser más simple mediante un circuito analógico que mediante cálculo digital [3]. La Figura 3.1 muestra la organización secuencial de todas estas funciones.

Figura 3.1 Funciones básicas en un sistema de medida[1]

3.1 Sensores

Los sensores son los encargados de obtener señales eléctricas a partir de señales no eléctricas. Las señales no eléctricas pueden ser magnitudes físicas o químicas, como por ejemplo: temperatura, intensidad de luz, humedad relativa del aire, distancia, aceleración, inclinación, presión atmosférica, nivel de pH, nivel de dióxido de carbono, etc. Todas estas variables se denominan variables de instrumentación.

Los sensores son dispositivos capaces de detectar estas variables de instrumentación y transformarlas en magnitudes eléctricas, como pueden ser resistencia, capacidad, tensión, corriente, inductancia, entre otras.

[1] Tomada de [3].

Existen una gran cantidad de sensores y formas de clasificarlos. Si se clasifican por la salida que entregan se podrán clasificar en dos grandes grupos: los analógicos y los digitales.

En los sensores analógicos en la salida se obtiene una magnitud física (tensión o corriente, etc.) que puede ir variando dentro de un rango determinado desacuerdo a la lectura que este obteniendo.

Los sensores digitales entregan a su salida un código digital, entendemos como digital que entregan o un 1 (alto) o un 0 (bajo), en otras palabras no varían en valores intermedios de un rango como los analógicos.

Existen 3 tipos de sensores digitales:

- Los que dan a su salida un tren de pulsos en que su frecuencia es proporcional al mesurando
- Los sensores que entregan un pulso en que la duración del mismo es directamente proporcional a la medición realizada.
- Los que a su salida dan un código digital.

En los sensores en los cuales su salida es dada por una duración de pulsos o por un tren de pulsos, es necesaria la utilización de un contador para poder medir la frecuencia (f_x) que ocurre en un tiempo de medida T_m.

3.1.1 Sensor de Temperatura LM35DT

Para realizar las mediciones de la temperatura ambiental se utilizo el sensor analógico LM35DT. El sensor LM35DT presenta entre sus características una salida lineal proporcional en grados centígrados (de -55 a 150 °C), una baja impedancia (0,1 W para cargas de 1mA), bajo consumo (60 μA) además de un bajo costo [4].

El sensor tiene un factor de escala lineal de 10 mV por grado centígrado como se muestra en la Figura 3.2.

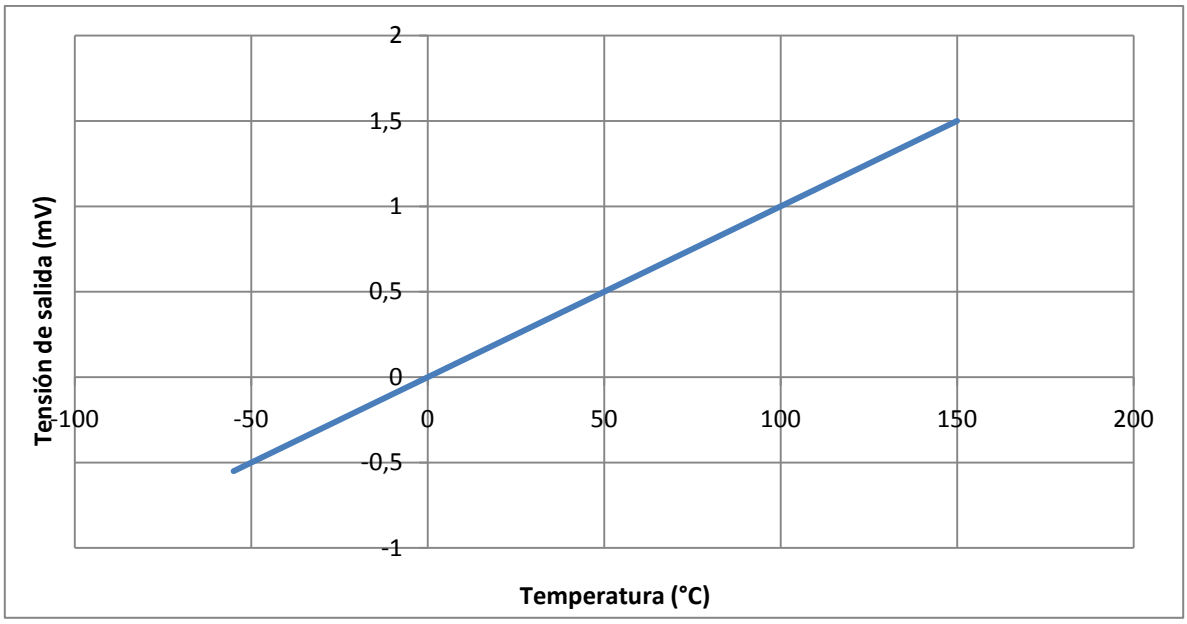

Figura 3.2 Voltaje de salida vs. Temperatura del sensor LM35DT

3.1.2 Sensor de Humedad HIH-4000-001

Para medir la humedad se utilizó el sensor analógico Honeywell HIH-4000. Este sensor necesita una tensión de 4 a 5.8 V para su funcionamiento y típicamente consume una corriente de 200 µA. Su rango de medición es de 0 a 100% de la humedad relativa.

Es un sensor de bajo costo, de un tamaño manejable y robusto que lo hace resistente a condiciones extremas de humedad, suciedad, polvo aceites y ambientes químicos.

Una de sus características es que no necesita de calibración alguna y posee una respuesta lineal como se logra observa en la Figura 3.3, tomada de su hoja de datos.

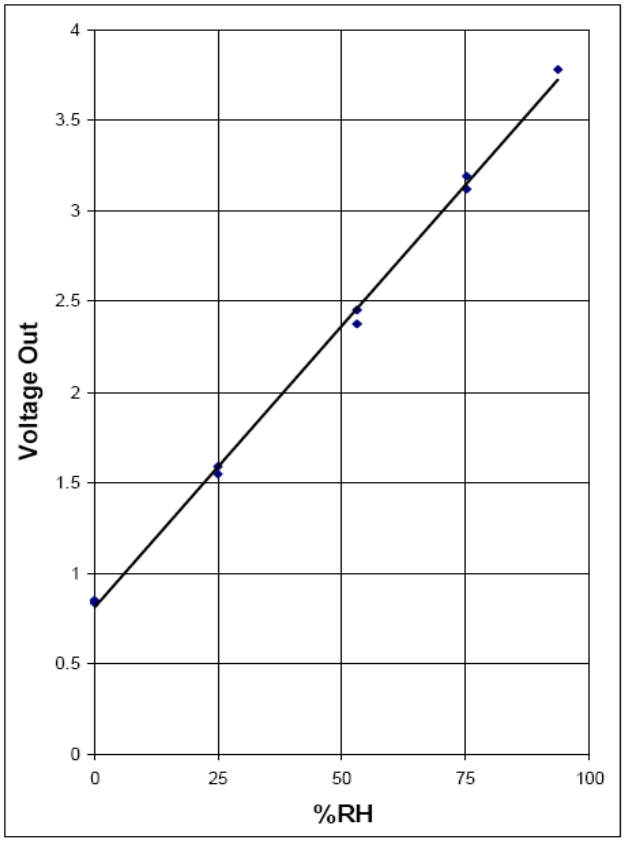

Figura 3.3 Voltaje de salida vs. Humedad relativa del sensor HIH-4000-001

Tiene una precisión de ±3.5% de 0-100% a una temperatura de 25 °C y con un voltaje de alimentación de 5V. Pero para poder compensar la medición de humedad a diferentes temperaturas se utiliza la siguiente formula de compensación:

$$\boxed{V_{HIH}(RH,T) = (0.0305 + 44x10^{-6} * T - 1.1x10^{-6} * T^2) * RH + (0.9237 - 0.0041 * T + 40x10^{-6} * T^2)}$$ Ec. [3.1]

En donde la temperatura está dada en grados centígrados.

Este sensor se ve afectado por la luz y por la estática, por lo que se recomienda en la medida de lo posible aislarlos de esta 2 variables físicas. Además se recomienda que este sensor funcione dentro de su área de operación recomendada como se muestra en la Figura 3.4.

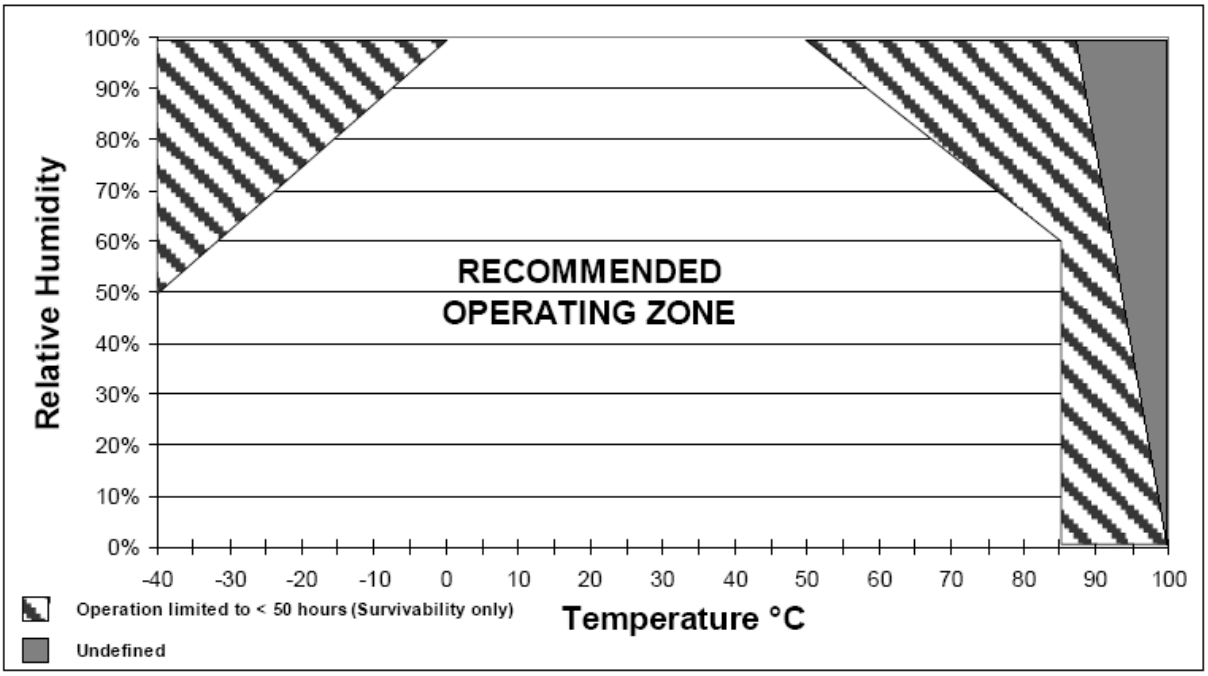

Figura 3.4 Área recomendada de trabajo del sensor Honeywell HIH-4000.

3.2 Microcontrolador PIC32MX

El microcontrolador de 32 bits PIC32MX es producido por la compañía Microchip desde finales de 2007. Estos microcontroladores tienen una velocidad de procesamiento de 1.5 DMIPS/MHz en un núcleo M4K.

Microchip tiene además a disposición placas de desarrollo (Starter Kit) que pueden comunicarse por USB o por Ethernet según sea la necesidad del usuario. Además también presenta una placa de expansión DM3200002, ver Figura 3.5.

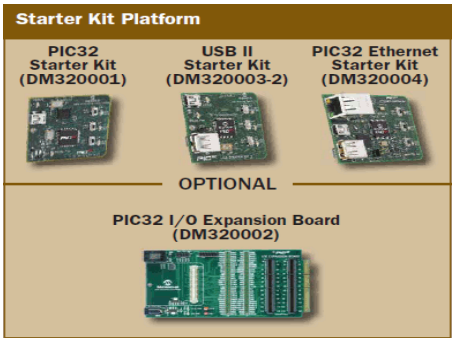

Figura 3.5 Placas de desarrollo del PIC32

3.3 Sistema Embebido[2]

Un sistema embebido es un dispositivo que posee un microprocesador o un microcontrolador para realizar una tarea en específico. Estos sistemas se pueden encontrar en dispositivos que comúnmente se manejan en la casa, oficina o hasta en el automóvil. Como ejemplos se puede citar teléfonos, sistemas de seguridad, sistemas de climatización, sistemas de video juegos.

Estos sistemas tienen como grandes ventaja que soportar diferentes sistemas operativos y ejecutan tareas en un corto periodo de tiempo. En las industrias se utiliza mucho este tipo de sistemas ya que pueden ser controlados externamente por un procesador central y son de un bajo costo.

3.4 Java[3]

En el año de 1995 Sun Microsystems ofreció al público un lenguaje de programación orientado a objetos llamado Java. Pero este lenguaje ya se venía desarrollando desde principios de los años 90 por un grupo de Sun Microsystems llamado “Green Team”, dirigidos por James Gosling.

En un principio este lenguaje se llamaba Oak, en honor a un roble que se encontraba a las afueras de la oficina de James. Luego cambió al nombre “Green” y finalmente se llamó Java al parecer tomaron el nombre de un café de una cafetería cercana a su área de trabajo. Esta hipótesis del nombre es la más fuerte ya que el icono de este lenguaje es una taza de café caliente.

Java es un lenguaje muy parecido a otros lenguajes de programación como son C y C++, solo que en JAVA eliminaron herramientas de bajo nivel, lo cual lo convierte en un leguaje más amigable como el programador.

Entre los principales objetivos a la hora de crear este lenguaje destacan permitir la ejecución de un mismo programa en diferentes sistemas operativos, el soporte de trabajar en red y la programación orientada a objetos.

3.5 MPLAB

MPLAP es un software gratuito que se utiliza para escribir lenguaje de programación y poder desarrollar proyectos en los productos de la marca Microchip Technology Inc. es una empresa fabricante de microcontroladores, memorias y semiconductores analógicos.

[2] Tomado de [17]
[3] Tomado de [18]

El programa MPLAB no corre bajo sistemas operativos como Linux, Unix o Macintosh, pero si corre como una aplicación de 32-bits bajo el sistema operativo Windows. Al MPLAP se le denomina como un software IDE (entorno de desarrollo informático, traducido del Ingles) y consta básicamente de 3 partes:

- Editor de texto
- Compilador
- Simulador

3.6 Protocolo USB[4]

Universal Serial Bus o más comúnmente conocido como USB, es un protocolo de comunicación creado en 1996 por 7 diferentes compañías de la industria de las computadoras. Fue creada con la finalidad dejar de comprar tarjetas separadas de los puertos ISA o PCI y desarrollada para lograr tener una mejor comunicación entre los periféricos y una computadora.

Existen 4 tipos diferentes de USB y se clasifican por su velocidad como se muestra en la Tabla 3.1.

Tabla 3.1 Clasificación del protocolo USB de acuerdo a su velocidad

Clasificación	Tasa de Transferencia
USB 1	1,5 Mbps
USB 1.1	12 Mbps
USB 2	480 Mbps
USB 3	4.8 Gbps

El cable USB está compuesto de 4 conductores, dos son de señal y dos son de potencia. Los de señal se demonizan D- y D+ y los de potencia se denominan V_{CC} y GND.

3.7 Estadística

3.7.1 Promedio

El promedio ($\overline{X}$) o media aritmética es igual a la suma de todos los datos o muestras dividida entre la cantidad de sumandos, tal como se muestra en su fórmula:

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n} \qquad \text{Ec. [3.2]}$$

[4] Tomado de [16]

El promedio nos muestra un número que simboliza el comportamiento de una cantidad de muestras durante un tiempo. Este número puede ser muy útil cuando los la gran mayoría de muestras o datos han tenido un comportamiento similar, pero dicho número puede ser muy fácilmente alterado por algún dato que se salga de dicho comportamiento, es decir por un valor muy alto o muy bajo de la gran mayoría de muestras o datos.

3.7.2 Varianza

La varianza o también conocido como coeficiente de variación es una medida de dispersión que posee una variable con respecto a su esperanza. Se entiende como esperanza matemática a un valor que se consideraría como correcto o esperado en un fenómeno aleatorio.

El símbolo de la varianza es el σ^2 y las unidades de la varianza van a ser el cuadrado de los valores aleatorios que se estaban tomando en consideración. Al igual que el promedio la varianza puede ser fácilmente alterada por valores atípicos, por lo cual en presencia de ellos se recomienda utilizar otras medidas de dispersión.

La formula de la varianza es la siguiente:

$$V = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n} \qquad \text{Ec. [3.3]}$$

3.7.3 Desviación Estándar

La desviación estándar es la raíz cuadrada de la varianza y es identificado por el símbolo griego de Sigma (σ).

Al igual que la varianza es una medida de dispersión para variables aleatorias y de intervalo. La desviación estándar es una medida que nos da el valor medio que aparta a los datos de la media o de la esperanza matemática. Las unidades de la desviación estándar van a ser iguales a las muestras o datos que se tomaron.

La desviación estándar se describe por la siguiente formula

$$S = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n}} \qquad \text{Ec. [3.4]}$$

Capítulo 4: Procedimiento Metodológico

4.1 Métodos y Actividades

4.1.1 Análisis de soluciones comerciales existentes y escogencia y validación de sensores

Por medio de correos electrónicos y de visitas se empezaron a buscar empresas que resolvieran este tipo de necesidad a nivel comercial, esto con el fin de establecer los costos, tipo de soluciones que se plantean además del seguimiento de mantenimiento de los equipos.

Se analizaron los sensores que existentes en los mercados nacionales e internacionales, se solicitaron cotizaciones. Los mercados investigados fueron el estadounidense, el chileno, el mexicano y el nacional. Se escogieron los sensores LM35DT y el sensor HIH-4000 por su fácil adquisición en el mercado nacional y de ser el caso de fácil obtención en el mercado estadounidense además por su precio y por la simplicidad de su uso. Los sensores fueron aprobados por el profesor asesor Ing. Marvin Hernández.

El sensor de temperatura LM35DT se validó utilizando un termómetro de resistencia de platino estándar con ayuda de la Escuela de Física del Instituto Tecnológico de Costa Rica. Se tomaron datos de temperatura y salida del sensor para obtener la ecuación de salida del sensor.

4.1.2 Análisis del protocolo USB y escogencia del kit de desarrollo.

Se procedió al análisis del protocolo de comunicación USB, por medio de investigación. Se evaluaron las ventajas sobre otros protocolos y se procedió a la escogencia del mismo

Para poder desarrollar un sistema embebido, se buscó un Kit de desarrollo de microcontroladores. Luego de investigar se escogió el kit de desarrollo de Microchip PIC32, el cual posee comunicación USB y una tarjeta desarrollo de expansión de puertos. La compra fue hecha por el ITCR y aprobada por el Ing. Marvin Hernández.

4.2 Implementación y desarrollo

4.2.1 Acondicionamiento de la señales de sensores a tarjeta de expansión de puertos

Los sensores fueron sometidos a pruebas para medir su respuesta y para determinar el rango de tensión de salida. Se comenzó a acoplar las señales de salida de los sensores a la tarjeta de expansión de puertos, además se acondicionó físicamente la tarjeta de puertos para la conexión de las señales de los sensores.

4.2.2 Conversión de datos analógicos a digitales

Por medio del convertidor analógico-digital de 10 bits del PIC32MX se procedió a convertir las señales de analógicas a digital. Se probaron varias configuraciones del módulo ADC (auto conversión, conversión manual, conversión iniciada por interrupción de un contador), y al final se decidió usar la auto conversión, ya que se ajustaba a las necesidades del sistema embebido.

4.2.3 Desarrollo de la comunicación USB de tarjeta de desarrollo a computadora

Se programó un módulo en el cual el PIC32MX funciona como un dispositivo periférico y le responde a la computadora cada vez que esta le demande un dato. Se establecieron dos tipos de descargas, una es la descarga de todos los datos guardados en memoria y la otra la descarga de el último dato que se midió.

4.2.4 Interfaz gráfica de usuario

Con la programación en lenguaje Java versión 6, se programa una interfaz grafica. Por medio de esta interfaz el usuario puede visualizar los datos guardados y los datos en vivo, además de poder graficar los datos y guardarlos en un documento de Excel.

Capítulo 5: Descripción detallada de la solución

5.1 Validación del sensor de temperatura LM35DT

Para realizar las mediciones de la temperatura ambiental se utilizó el sensor de temperatura LM35DT. El sensor LM35DT presenta entre sus características:

- Una salida lineal proporcional en grados centígrados (de -55 a 150 °C)
- Baja impedancia (0,1 W para cargas de 1 mA)
- Bajo consumo (60 μA)
- Bajo costo

El factor de escala lineal del sensor es de 10 mV/°C. Este sensor tiene un comportamiento lineal en su salida de voltaje en el área de trabajo de este proyecto, o sea entre 5 °C y 40 °C que son las temperaturas entre las cuales se mantiene la gran parte del territorio de Costa Rica.

Con el objetivo de corroborar las mediciones del sensor LM35DT y además de obtener su curva característica se solicitó la colaboración del Departamento de Física del Instituto Tecnológico de Costa Rica. Con la ayuda de profesor Juan Carlos Lobo Zamora se realizaron mediciones con los siguientes equipos calibrados:

- Baño de mantenimiento de agua de triple punto, marca Fluke modelo 7312
- Termómetro de resistencia estándar de platino (SPRT), marca Fluke modelo 5698. En el Anexo A.2 se muestra la hoja de calibración de este termómetro.

En la Figura 5.1 se muestran estos equipos.

Figura 5.1 Termómetro Fluke 5698 y baño de mantenimiento Fluke 7312

El sensor LM35DT fue encapsulado y sellado en un tubo de ensayo como se muestra en la Figura 5.2, con el objetivo de ubicarlo dentro del equipo Fluke 7311 y poder hacer las mediciones en el mismo ambiente que el termómetro Fluke 5698.

Figura 5.2 Sensor LM35DT encapsulado en un tubo de ensayo

El medio utilizado para tomar estas mediciones fue agua destilada por lo que la temperatura más baja que se logró alcanzar fue de 3.6°C.

La metodología de obtención de datos fue recomendada por el profesor Lobo. Se procedió a bajar la temperatura del baño maría hasta 3.8°C, dicho enfriamiento tomó cerca de 6 horas. La siguiente medición fue a 5°C y se fue subiendo la temperatura cada 5°C hasta llegar finalmente a los 40°C.

Una vez que el profesor indicara que el sistema estaba estable en la temperatura deseada, se realizaron 30 mediciones de voltaje cada 2 segundos entregado por el sensor. De estas 30 mediciones se procedía a hacer un promedio como se muestra en la Tabla 5.1.

Tabla 5.1 Datos de temperatura del SPRT y tensión promedio de salida del sensor LM35DT

Temperatura SPRT (°C)	Tensión Promedio Salida Sensor LM35DT (mV)
3,6266	49,91
5,0066	61,25
10,4114	113,89
15,1191	156,88
19,8323	201,43
24,9986	248,46
29,9268	294,44
34,9078	339,02
40,0643	390,65

Tomando estos datos podemos graficar la temperatura del termómetro calibrado contra la tensión promedio de salida del sensor, como se muestra en la Figura 5.3.

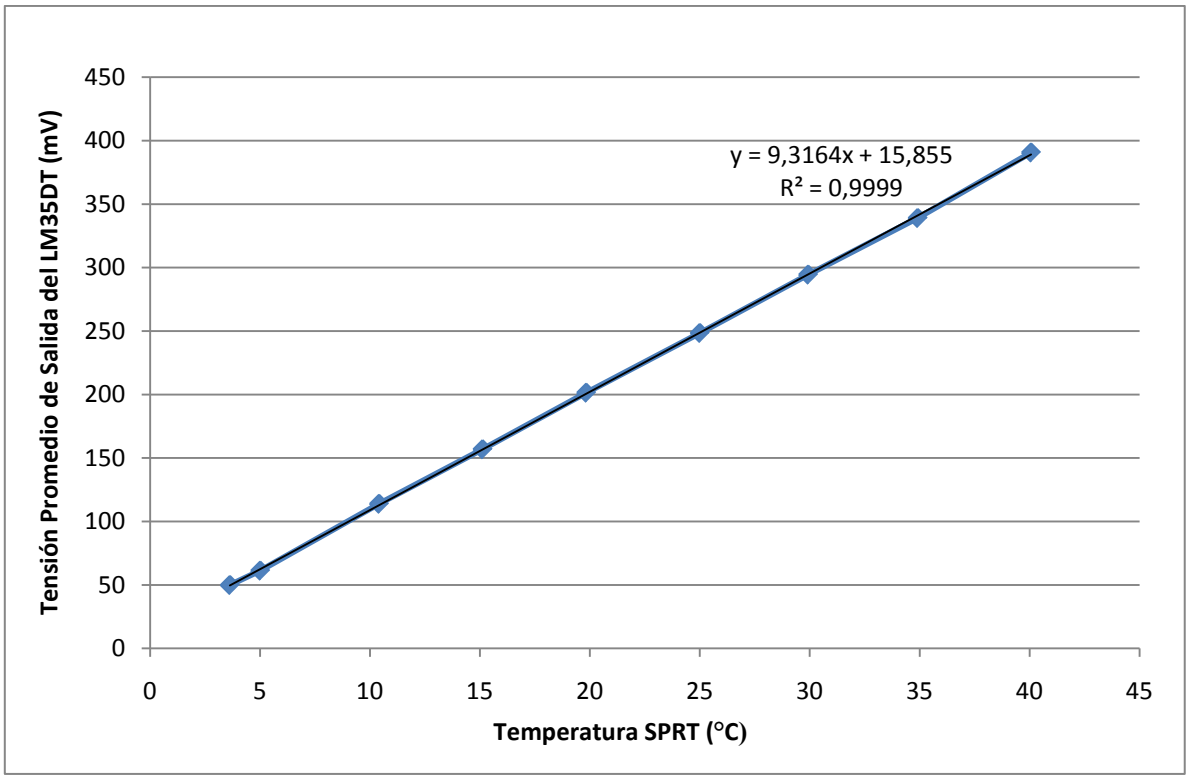

Figura 5.3 Tensión promedio de salida del sensor LM35DT vs. Temperatura del SPRT

De la Figura 5.3 podemos vemos que se obtuvo una respuesta lineal de la forma:

$$V_{LM35DT} = 9.3164\ \frac{mV}{^{\circ}C} * T + 15.855\ mV \qquad \text{Ec. [5.1]}$$

De la ecuación 5.1 se observa que el factor de escala obtenido fue de 9.3164 mV/ºC, en vez de los 10 mV/ºC que indica la hoja de datos. Además, se obtuvo un cruce por el eje "Y" diferente de cero, lo que indica un offset de 15.855 mV a cero grados Celsius.

Ahora bien, en nuestro sistema lo importante es convertir la tensión entregada por el LM35DT a un valor de temperatura, por lo tanto si graficamos los datos de la Tabla 5.1 usando la tensión de salida del sensor en el eje "X" y la temperatura en el eje "Y" se obtiene la curva de la Figura 5.4.

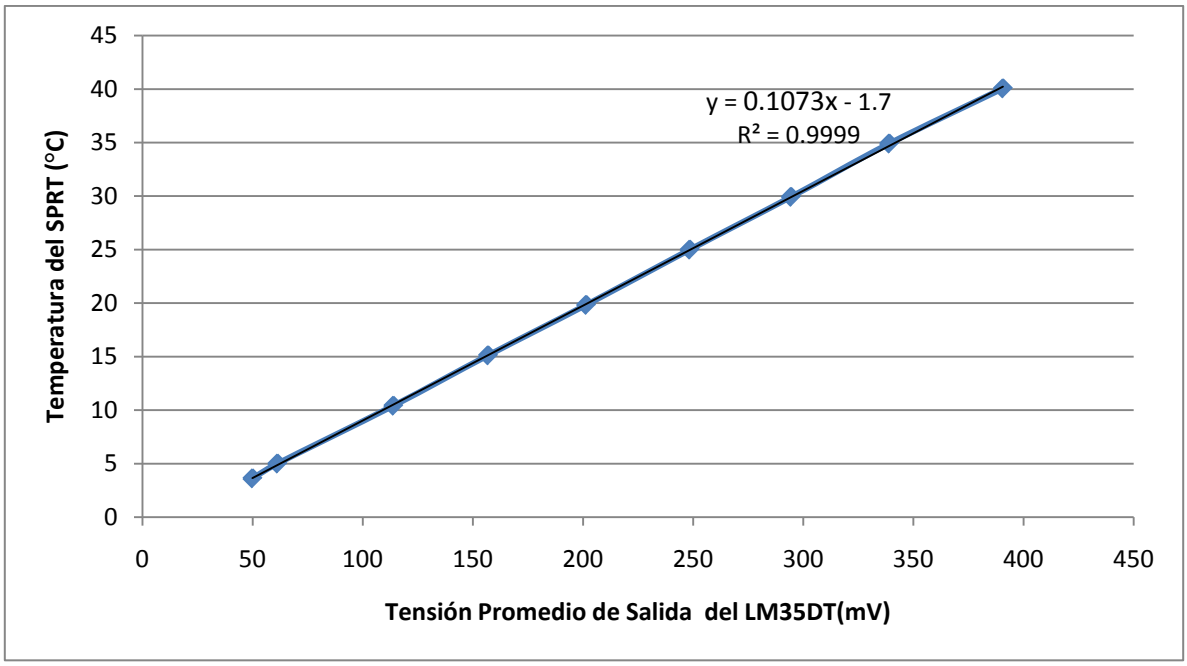

Figura 5.4 Temperatura del SPRT vs. Tensión promedio de salida del sensor LM35DT

La ecuación obtenida que describe la temperatura en función del voltaje de salida del sensor LM35DT es:

$$T(V_{LM35DT}) = 0.1073\ \frac{℃}{mV} * V_{LM35DT} - 1.7\ ℃ \qquad \text{Ec. [5.2]}$$

La ecuación 5.2 es la que se debe programar en el microcontrolador para obtener la temperatura a partir de la tensión entregada por el sensor.

5.2 Diagrama de bloques del sistema

El sistema en su totalidad se compone de tres bloques principales: red de sensores, sistema embebido basado en microcontrolador PIC32MX y computadora en la cual se presenta la interfaz grafica de usuario.

En la Figura 5.5 se muestra el diagrama de bloques de todo el sistema.

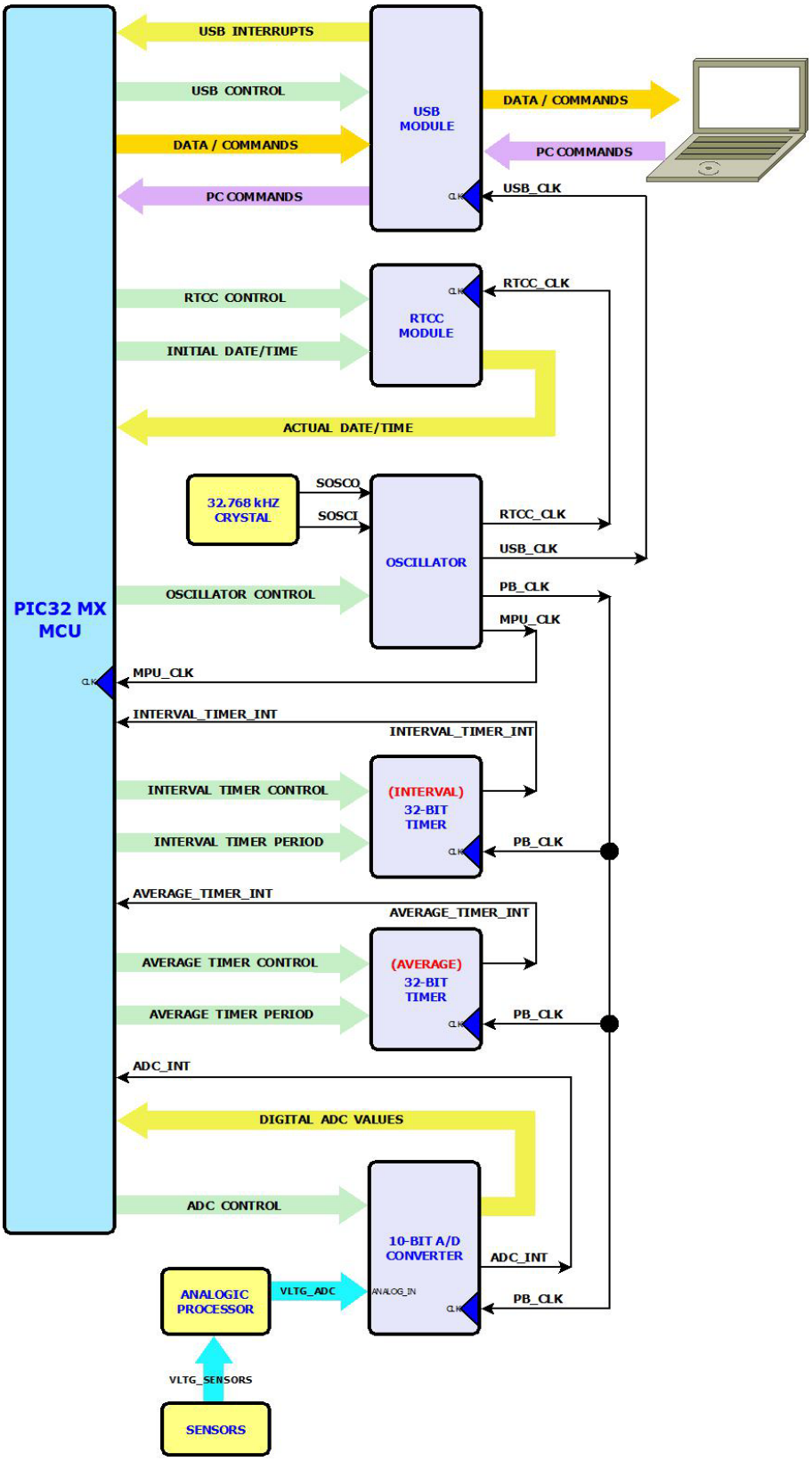

Figura 5.5 Diagrama de bloques del sistema

En la Figura 5.5, se puede decir que el bloque de sensores junto con el procesador analógico constituye la red de sensores. El PIC32MX junto con todos los dispositivos periféricos (ADC, contadores, módulo RTCC, oscilador, módulo USB, y el cristal de cuarzo de 32.768 kHz) forman el denominado sistema embebido. La computadora es donde se programa la interfaz grafica de usuario.

Ahora veamos más en detalle cada uno de los bloques.

5.2.1 Bloque PIC32MX MCU

El bloque PIC32MX Microcontroller Unit (MCU) corresponde a la unidad de procesamiento central de una computadora. Es quien se encarga de realizar las instrucciones y manejar el flujo de información entre sistema central, memoria y periféricos. El PIC32MX utilizado fue el PIC32MX460F512L, que viene en la tarjeta de desarrollo USB Starter Kit, cuya identificación de producto se desglosa así:

Tabla 5.2 Identificación del microcontrolador utilizado (PIC32MX460F512L)

Término	Significado
PIC32	Marca Microchip
MX	Arquitectura 32-bit RISC MCU Core
460	Grupo de Producto: USB
F	Memoria de programa Flash
512	Tamaño de la memoria de programa (KB)
L	100 pin

El MCU realiza operaciones bajo el control del programa con el cual es cargado. Las instrucciones son buscadas por el CPU, decodificadas y ejecutadas de forma sincronizada. Las instrucciones pueden estar tanto en la memoria de programa Flash o la memoria de datos RAM. El PIC32MX MCU está basado en una arquitectura cargar/guardar y realiza la mayoría de las operaciones en un set de registros internos. Instrucciones de cargar y guardar específicas se usan para mover los datos entre estos registros internos y el mundo exterior.

En general, PIC32MX constituye una reestructuración de la arquitectura interna de los microcontroladores de las familias anteriores en 4 diferentes bloques: el procesador, un sistema de memoria, un sistema de integración y los periféricos [6].

Como se muestra en la Figura 5.5 (Diagrama de bloques del sistema) el MCU se encarga de controlar las funciones de los periféricos, además de manejar

las interrupciones y recibir datos de los periféricos. Se encarga también del control del módulo USB para hacer posible la comunicación con la computadora.

La Figura 5.6 muestra el diagrama de bloques de un PIC32MX MCU.

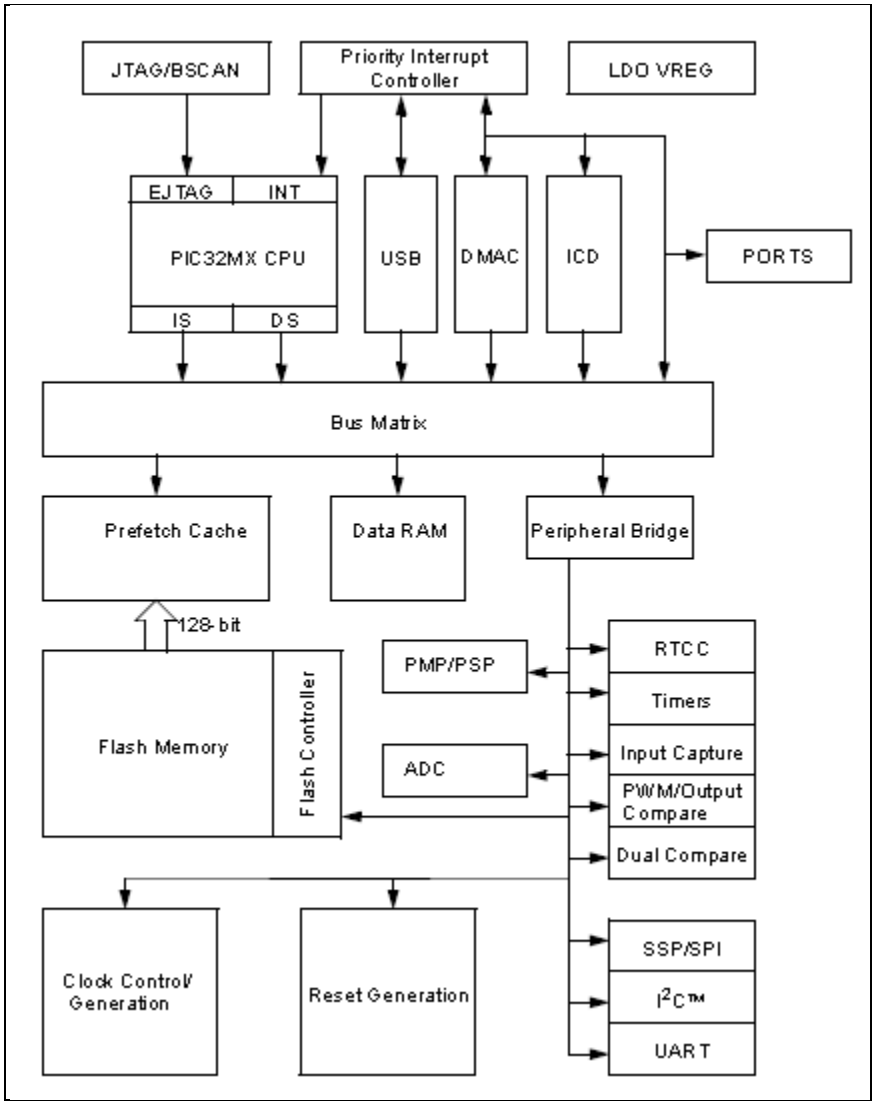

Figura 5.6 Diagrama de bloques del PIC32MX MCU[5]

[5] Tomada de [7]

5.2.2 Bloque Oscilador

El bloque Oscilador se encarga de generar las señales de reloj para: la unidad central del microprocesador y periféricos específicos, los otros periféricos, el módulo de reloj y calendario en tiempo real (RTCC), y el módulo USB.

Este bloque genera los relojes a partir de los registros de control, los cuales son escritos por el PIC32MX MCU, dependiendo de las necesidades del usuario.

En la Figura 5.7, se muestra la conexión entre los bloques.

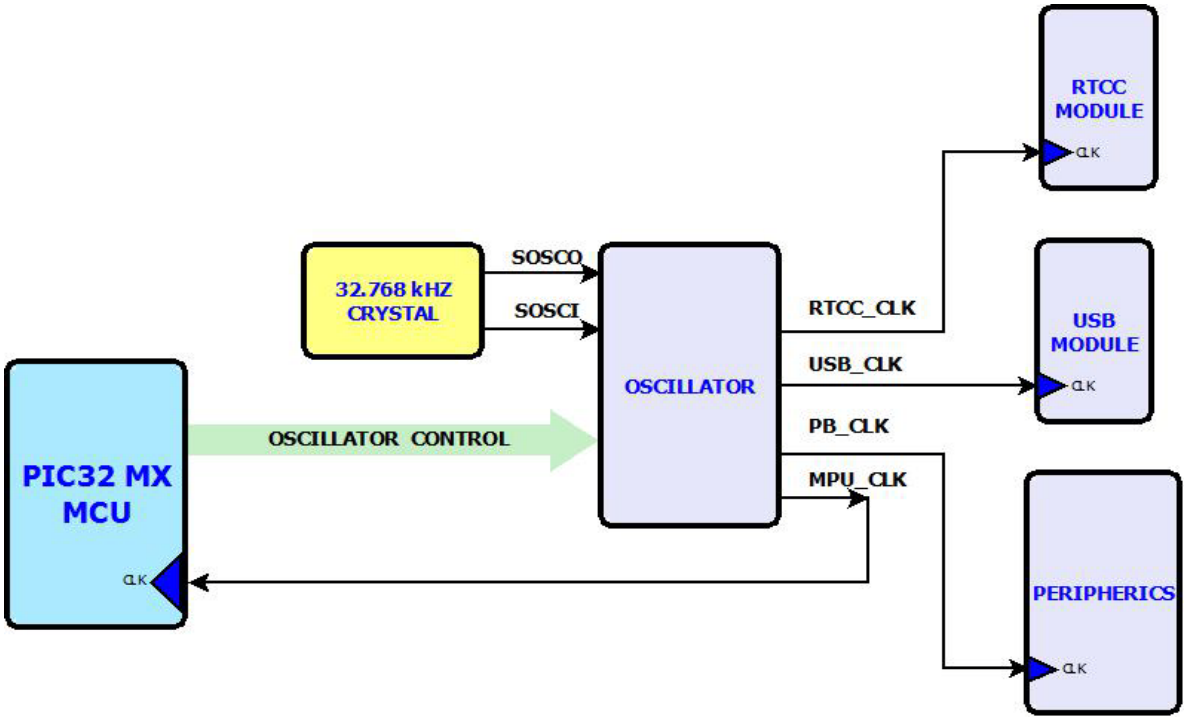

Figura 5.7 Conexión del bloque Oscilador con los otros bloques

En nuestro caso se desea trabajar con:

Tabla 5.3 Frecuencias utilizadas en el sistema PIC32MX

Parámetro	Valor
Frecuencia del MCU	80 MHz
Frecuencia de los Periféricos	80 MHz
Frecuencia del módulo USB	48 MHz
Frecuencia del RTCC	32.768 kHz

En la Figura 5.8 se muestra el diagrama de bloques del módulo oscilador. En ella vemos resaltada con color los bloques principales que se configuran para obtener las frecuencias de salida deseadas.

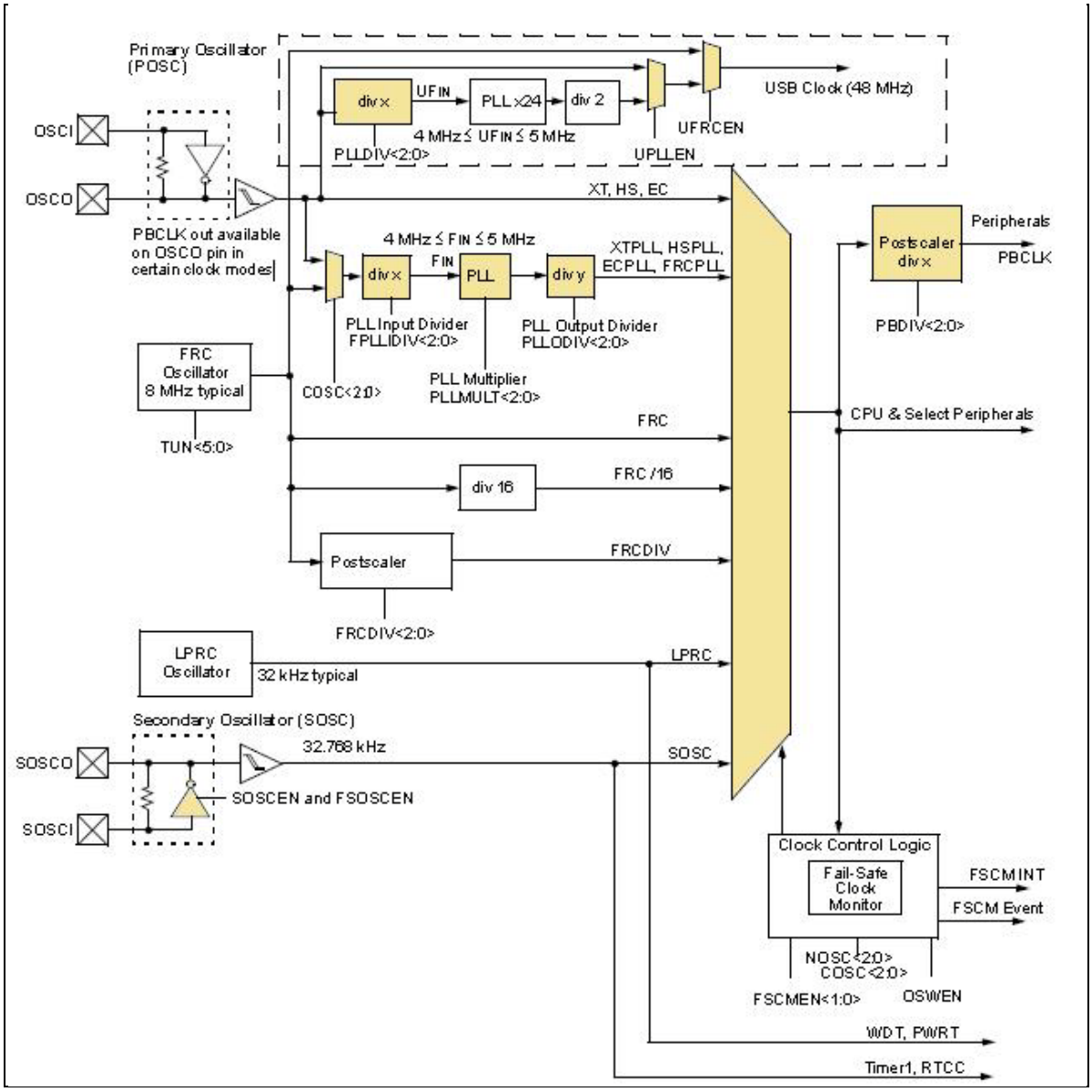

Figura 5.8 Diagrama de bloques del módulo Oscilador[6]

El reloj del MCU (SYSCLK) es principalmente usado por el MCU y periféricos específicos como el DMA (Direct Memory Access), el Controlador de Interrupciones, y la Memoria Cache de Pre-búsqueda de Instrucciones. El SYSCLK se derivó del oscilador principal (POSC). La frecuencia del oscilador

[6] Tomada de [8]

principal se pasa a un PLL (Phase Locked Loop), donde primero se divide la frecuencia del oscilador principal, luego se pasa a un multiplicador y por ultimo por otro divisor de frecuencia, para así obtener el SYSCLK deseado.

De forma que, para obtener los 80 MHz para el SYSCLK se toman los 8 MHz del Fast RC Oscillator (FRC), se dividen entre 2, para obtener 4 MHz. Estos 4 MHz se multiplican por 20, con lo cual se obtienen los 80 MHz. Ahora como no hace falta dividir esta frecuencia, se utiliza un valor de 1 para el último divisor de frecuencia y listo.

El reloj de bus de periféricos se obtiene de dividir el SYSCLK entre un pos-escalador, que puede ser de 1, 2, 4 o 8. Para la frecuencia de 80 MHz requerida, se usa un pos-escalador de 1.

La frecuencia del módulo USB se obtiene de pasar la frecuencia del oscilador principal por un PLL, donde lo único por definir es el divisor de entrada, ya que el multiplicador y el divisor de salida ya están fijos. Se toman los 8 MHz del FRC y se dividen entre 2, para obtener 4 MHz, luego este valor se multiplica por 24 para obtener 96 MHz, y con el divisor se logra reducir esta frecuencia a la mitad y así obtener los 48 MHz con los cuales trabaja el módulo USB.

Para obtener la frecuencia del módulo RTCC, se conectó un cristal de cuarzo externo de una frecuencia 32.768 kHz en los pines OSCI y OSCO del oscilador, además se habilitó al oscilador secundario, para así obtener la frecuencia requerida para el funcionamiento correcto del reloj y calendario de tiempo real.

En la Tabla 5.4 se muestra como se configuró el bloque Oscilador para obtener las frecuencias deseadas de la Tabla 5.3.

Tabla 5.4 Configuración del módulo Oscilador

Registro de Función Especial	Configuración
OSCCON (Oscillator Control Register)	• Divisor de entrada para el PLL principal = 2 • Multiplicador del PLL principal = 20 • Divisor de salida para el PLL principal =1 • Divisor del reloj del bus de periféricos = 1 • Usar el PLL del USB como fuente para el reloj del USB • Habilitar oscilador secundario (32.768 kHz)
DEVCFG1 (Boot Configuration Register 1)	• Cambio de reloj deshabilitado • Módulo FSCM (Fail Safe Clock Monitor) deshabilitado • PBCLK es SYSCLK/1 • Señal de salida del CLK0 activa en pin OSCO deshabilitado • Configuración del oscilador principal: Modo oscilador de alta velocidad (HS, high speed) • Cambio de reloj interno-externo deshabilitado • Oscilador secundario habilitado • Selección de oscilador: Primario con PLL
DEVCFG2 (Boot Configuration Register 2)	• Salida del PLL del sistema dividida por 1 • Habilitar PLL del reloj USB • Divisor de entrada para el PLL USB es 2 • Multiplicador del PLL del sistema es 20 • Entrada del PLL del sistema dividida por 2

5.2.3 Bloque RTCC

El módulo de reloj y calendario en tiempo real (RTCC, Real Time Clock and Calendar) se encarga de proveer una solución para aquellas aplicaciones que ocupen llevar un control de tiempo durante largos períodos con poca o nula intervención del MCU.

Para el funcionamiento correcto del RTCC, se necesita una frecuencia de 32.768 kHz, la cual es obtenida por medio de un oscilador de cristal de cuarzo. El bloque Oscilador provee esta frecuencia que utiliza el RTCC.

El MCU se encargó de configurar el RTCC por medio de sus registros de control, además de transmitirle los valores de la fecha y hora inicial. Una vez que fue programado con la fecha y hora inicial y que se habilita el RTCC y el oscilador secundario está corriendo y activo, entonces el RTCC lleva un registro de la hora

y fecha actual, en tiempo real. Cuando el MCU lo indique, el RTCC se encarga de pasar la fecha y hora en la que se llevo a cabo la última medición.

En la Figura 5.9 se muestra la conexión entre el bloque RTCC y los demás bloques.

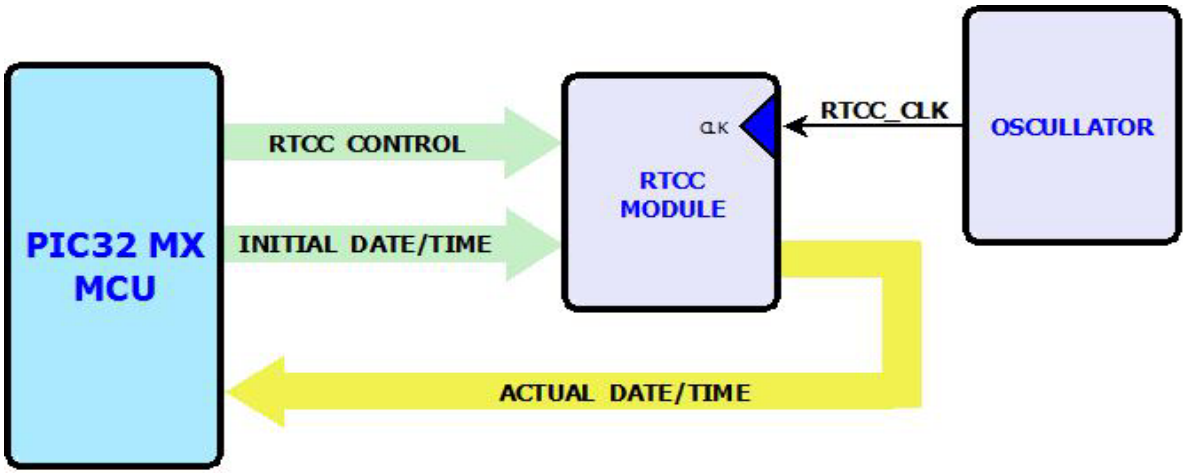

Figura 5.9 Conexión del módulo RTCC con los otros bloques

El diagrama de bloques interno del RTCC corresponde al de la Figura 5.10.

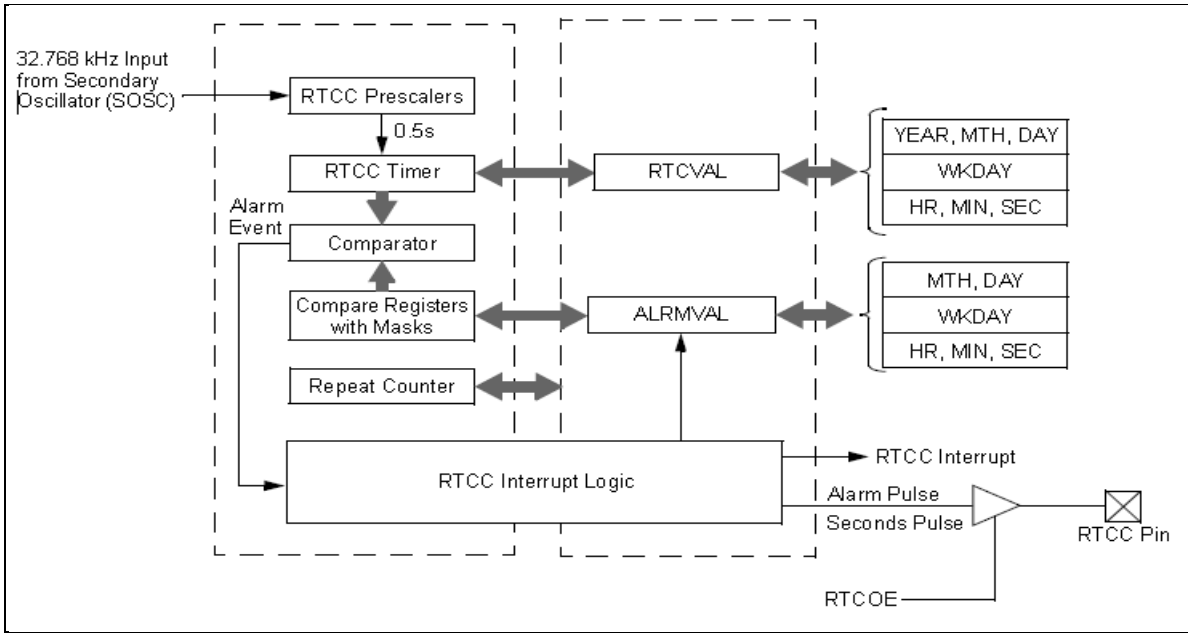

Figura 5.10 Diagrama de bloques del módulo RTCC[7]

[7] Tomada de [9]

Como se puede ver de la Figura 5.10, a partir de la frecuencia de 32.768 kHz y los valores con que se cargan su registro RTCVAL, el módulo RTCC es capaz de determinar las horas, minutos, segundos, día de la semana, año, mes y día. Es posible programar una alarma, sin embargo para nuestra aplicación no es necesaria.

La forma en que se configuró el módulo RTCC se muestra en la Tabla 5.5.

Tabla 5.5 Configuración del módulo RTCC

Registro de Función Especial	Configuración
RTCCON (RTCC Control Register)	• Módulo RTCC encendido • Módulo RTCC continua operación cuando el emulador esta en modo de depuración(DEBUG) • Módulo RTCC continua operación cuando el MCU entra a estado ocioso (IDLE)
RTCTIME (RTC Time Value Register)	• Se carga con el valor inicial de la hora
RTCDATE (RTC Date Value Register)	• Se carga con el valor inicial de la fecha

La configuración del RTCC es bastante básica, y lo que se requiere es realizar las lecturas de la fecha y hora cuando el MCU lo requiere, que es cuando se cumple el intervalo de tiempo que fue programado para guardar las mediciones.

5.2.4 Bloque Sensores

Como se mencionó anteriormente en el marco teórico, los sensores se encargan de transformar las variables físicas a medir en magnitudes eléctricas. En nuestro caso, los sensores utilizados son sensores analógicos que entregan en su patilla de salida una tensión eléctrica proporcional a la variable medida.

El bloque Sensores lo constituyen los diferentes tipos de sensores utilizados: temperatura (LM35-DT) y humedad (HIH-4000-001). Además, se simularon los sensores de luminosidad y dióxido de carbono con un potenciómetro.

Los sensores utilizados y el potenciómetro constan de tres patillas: tensión de alimentación, tierra y salida. Es importante tomar en cuenta el rango de la

tensión de alimentación que acepta cada sensor, con el fin de no dañar el sensor o perjudicar su funcionamiento correcto.

Los rangos de las tensiones de alimentación de los sensores, según las hojas de datos [4,5] utilizados son los de la Tabla 5.6.

Tabla 5.6 Tensión de alimentación de los sensores

Variable	Sensor	Mínimo (V_{DC})	Máximo (V_{DC})
Temperatura (ºC)	LM35 DT	4	30
Humedad Relativa (%)	HIH-4000-001	4	5.8

La Figura 5.11 muestra el esquema del bloque de sensores.

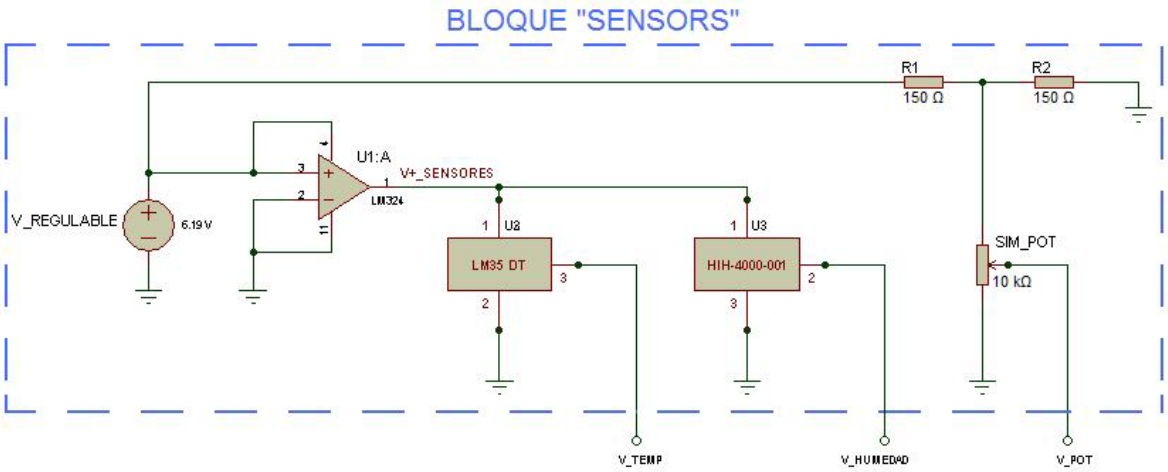

Figura 5.11 Diagrama del bloque Sensores

Como se puede ver en la Figura 5.11, se utilizó un convertidor de corriente alterna a corriente directa, con salida regulable, para energizar el circuito. La salida del convertidor se reguló a 6 V_{DC}, sin embargo al conectar el convertidor y medir su salida, este entregó una tensión de 6.19 V_{DC}. Este nivel de tensión es superior a las 5.8 V_{DC} que indica el fabricante para el sensor de humedad HIH-4000-001.

Por lo tanto, se utilizó un amplificador operacional sin realimentación en configuración de comparador para alimentar a los sensores. Dada la alta ganancia de tensión en lazo abierto de un amplificador operacional, una tensión de entrada positiva provoca una saturación positiva. Debido a que un amplificador operacional no puede suministrar más de la tensión a la que se alimenta, por el efecto de saturación, entonces al conectar los 6.19 V_{DC} a la patilla no inversora del A.O, y la patilla inversora conectada a tierra, en su salida se obtuvo una tensión de

saturación de 4.9 V_{DC}, la cual se encuentra dentro del rango de alimentación de ambos sensores.

El potenciómetro utilizado para simular los sensores de luminosidad y dióxido de carbono, se alimentó a través de un divisor de tensión formado por dos resistencias de mismo valor (150Ω). Esto porque, como indica la hoja de datos de la familia PIC32MX [6], el máximo valor permitido en un pin analógico del PIC32MX es de (V_{DD} + 0.3) V_{DC}, lo cual corresponde a aproximadamente 3.6 V_{DC}. [4] Al utilizar el divisor de tensión, se reparte la tensión del convertidor AC/DC entre las dos resistencias:

$$V_{DIV\_TENSION} = V_{REGULABLE} * \left(\frac{R_2}{R_1 + R_2}\right)$$

$$V_{DIV\_TENSION} = 6.19\ V_{DC} * \left(\frac{150\ \Omega}{150\ \Omega + 150\ \Omega}\right)$$

$$\boxed{V_{DIV\_TENSION} = 3.095\ V_{DC}}$$

Con esto se logró limitar el máximo de la tensión de salida del potenciómetro a 3.095 V_{DC}, y asegurar que no sobrepase el máximo de 3.6 V_{DC}.

A final de cuentas, la salida del bloque de sensores la constituyen las tres tensiones analógicas: la del sensor de temperatura, la del sensor de humedad, y la del potenciómetro utilizado para simular otros sensores.

5.2.5 Bloque Procesador Analógico

Antes de explicar el diseño del bloque “Procesador Analógico”, cabe recordar que el máximo valor de tensión (con respecto a tierra) que se puede aplicar a una patilla de entrada analógica de los microcontroladores de la familia PIC32MX es de (V_{DD}+0.3) V_{DC}, lo que corresponde aproximadamente a 3.6 V_{DC} [6]. En la Figura 5.12 se muestra el diagrama del bloque Procesador Analógico.

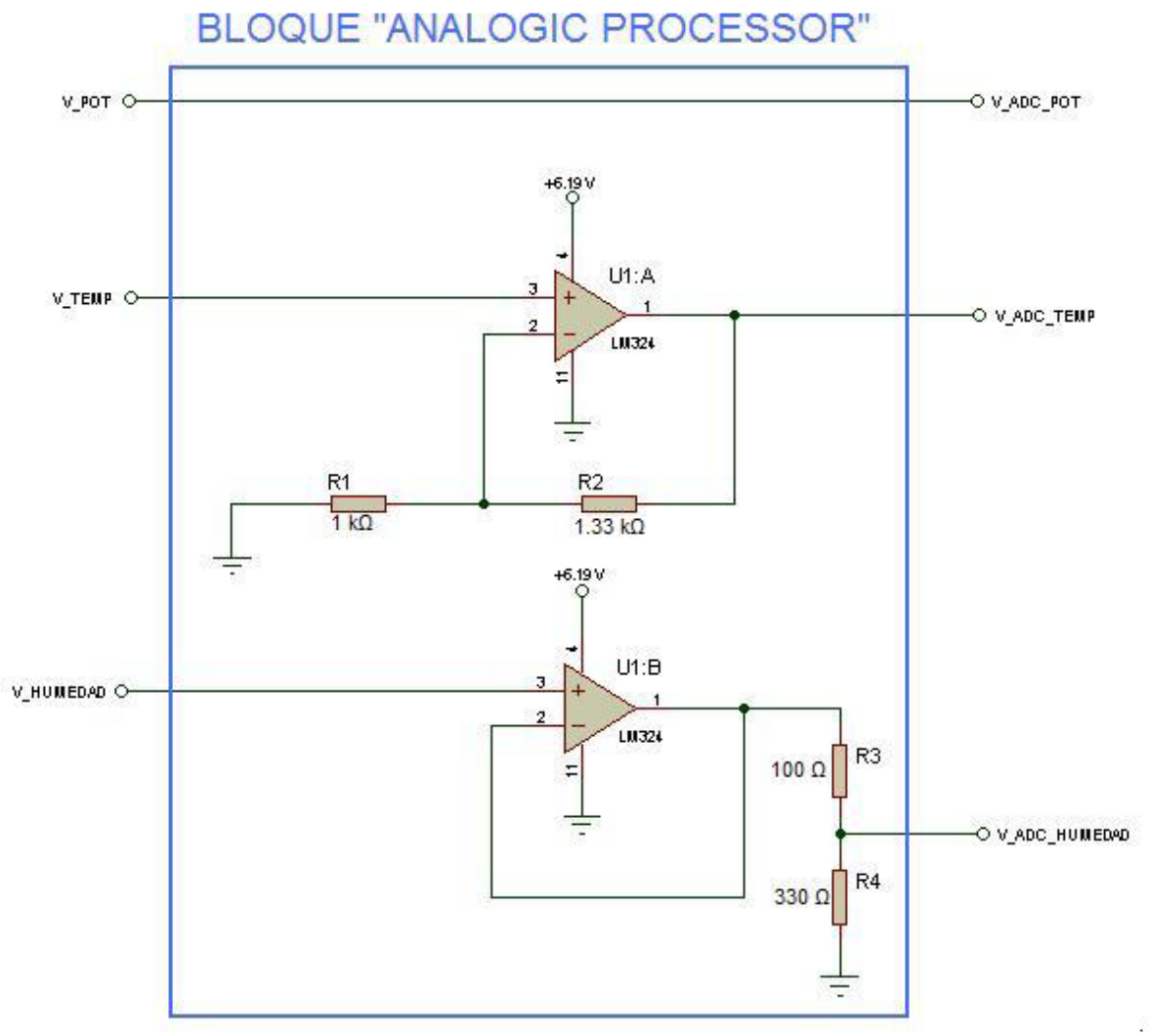

Figura 5.12 Diagrama del bloque Procesador Analógico

A continuación se explica la forma en que se acondicionó cada señal.

5.2.5.1 Acondicionamiento de la señal del potenciómetro

Debido a que el potenciómetro se alimentó con una tensión de 3.095 V_{DC}, a su salida no va entregar una tensión mayor a esta, por lo tanto esta tensión se puede conectar directamente a uno de los canales del convertidor analógico-digital.

5.2.5.2 Acondicionamiento de la señal del sensor de temperatura LM35DT

El sensor de temperatura LM35 DT, según la conexión presentada en la Figura 5.11, tiene un rango de medición de +2 ºC a +150 ºC. Según la Ecuación 5.1 (tensión de salida del sensor versus temperatura) se tiene que el rango de la tensión de salida del sensor es de:

$$V_{LM35}(T) = 9.3164\ \frac{mV}{°C} * T + 15.855\ mV$$

$$V_{LM35}(150\ °C) = 9.3164\ \frac{mV}{°C} * (150\ °C) + 15.855\ mV$$

$$\boxed{V_{LM35}(150\ °C) = 1.413\ V}$$

$$V_{LM35}(T) = 9.3164\ \frac{mV}{°C} * T + 15.855\ mV$$

$$V_{LM35}(2\ °C) = 9.3164\ \frac{mV}{°C} * (2\ °C) + 15.855\ mV$$

$$\boxed{V_{LM35}(2\ °C) = 34.49\ mV}$$

Por lo tanto, el rango de la tensión de salida del sensor LM35 DT es de [34.49 mV – 1.413 V], Ahora bien, el ADC tiene un rango de operación de entrada de [0 - 3.3 V], por lo tanto, para utilizar todo ese rango de tensión del ADC se utilizó un amplificador operacional con realimentación negativa en configuración amplificador no inversor. La ganancia del amplificador se calculó así:

$$G = \frac{Máximo\ Tensión\ ADC}{Máximo\ Tensión\ Sensor} = \frac{3.3\ V}{1.413\ V}$$

$$\boxed{G = 2.33}$$

Ahora bien, un amplificador no inversor como el de la Figura 5.12, tiene una ganancia dada por:

$$G_{NO\ INV} = 1 + \frac{R_2}{R_1} = 2.33$$

$$R_2 = (2.33 - 1)R_1 \quad \Rightarrow sea\ R_1 = 1\ k\Omega$$

$$\boxed{R_2 = 1.33\ k\Omega}$$

De ahí el valor de 1330 como R_2. Ahora bien, midiendo mediante multímetro los valores de las resistencias usadas se tiene que:

R_1 = 1005 Ω

R_2 = 1345.5 Ω

Por lo tanto, la ganancia real es de:

$$G_{NO\ INV} = 1 + \frac{R_2}{R_1} = 1 + \frac{1345.5\ \Omega}{1005\ \Omega}$$

$$\boxed{G_{NO\ INV} = 2.339}$$

A la hora de programar en software la rutina para obtener el dato de temperatura es necesario recordar sobre esta ganancia y convertir correctamente la tensión leída por el ADC.

5.2.5.3 Acondicionamiento de la señal del sensor de humedad relativa HIH-4000-001

El sensor de humedad relativa HIH-4000-001 tiene un rango de medición de 0 a 100% de humedad relativa. La ecuación 3.1 relaciona la tensión de salida del sensor con la humedad relativa del ambiente, tomando en cuenta la compensación por temperatura (en ºC):

$$V_{HIH}(RH, T) = (0.0305 + 44x10^{-6} * T - 1.1x10^{-6} * T^2) * RH + (0.9237 - 0.0041 * T + 40x10^{-6} * T^2)$$

Ahora bien, la tensión de salida va ser máxima cuando la humedad relativa sea 100% y la derivada de la tensión con respecto a la temperatura sea igual a cero.

$$\frac{\partial\left(V_{HIH}(RH,T)\right)}{\partial T} = (44x10^{-6} - 2.2x10^{-6} * T) * RH + (-0.0041 + 80x10^{-6} * T)$$

Sustituyendo RH por 100% e igualando a cero, para obtener la temperatura a la cual se obtiene la máxima tensión de salida del sensor de humedad:

$$\frac{\partial\left(V_{HIH}(100,T)\right)}{\partial T} = 0 = 0.0044 - 0.00022 * T - 0.0041 + 80x10^{-6} * T$$

$$0.00014 * T = 0.0003$$

$$\boxed{T = 2.143\ ℃}$$

Por lo tanto, la máxima tensión de salida que se puede obtener es de:

$$V_{HIH}(RH,T) = (0.0305 + 44x10^{-6} * T - 1.1x10^{-6} * T^2) * RH + (0.9237 - 0.0041 * T + 40x10^{-6} * T^2)$$

$$V_{HIH}(100,2.143) = (0.0305 + 44x10^{-6} * 2.143 - 1.1x10^{-6} * 2.143^2) * 100 + (0.9237 - 0.0041 * 2.143 + 40x10^{-6} * 2.143^2)$$

$$\boxed{V_{HIH} = 3.974\ V}$$

Este valor de tensión máxima que puede entregar el sensor de humedad es mayor a los 3.3 V_{DC} que es el valor máximo de entrada del ADC y además es mayor a los 3.6 V_{DC} que se pueden aplicar a una patilla de entrada de un PIC32MX.

Por lo tanto, se utilizó un divisor de tensión para lograr disminuir esta tensión a valores dentro del rango del ADC. Se utilizó un divisor de tensión como el que se muestra en la Figura 5.12. Los valores medidos de las resistencias de este divisor de tensión fueron:

R_3 = 101 Ω

R_4 = 327.7 Ω

De esta manera, el máximo valor de tensión a la salida del divisor de tensión es de:

$$V_{DIV\_TENSION\_HUM} = V_{HIH} * \left(\frac{R_4}{R_4 + R_3}\right)$$

$$V_{DIV\_TENSION\_HUM} = 3.974\ V * \left(\frac{327.7\ \Omega}{327.7\ \Omega + 101\ \Omega}\right)$$

$$\boxed{V_{DIV\_TENSION\_HUM} = 3.038\ V}$$

Esta tensión se puede conectar al ADC sin ningún problema.

Sin embargo, cuando se conectó la salida del sensor de humedad al divisor de tensión, el valor de dicha salida se caía, debido a un desacople de impedancias. El sensor de humedad posee una impedancia de salida alta, mientras que el divisor de tensión, al ser valores bajos de resistencia es por ende una impedancia baja (430Ω). Por esta razón se utilizo un seguidor de tensión cuya característica es que entrega a la salida el mismo valor de tensión de entrada, pero además, por la realimentación negativa máxima produce una impedancia de entrada de lazo cerrado que es mucho mayor que la impedancia de entrada en lazo abierto. También, la retroalimentación negativa máxima produce una impedancia de salida en lazo cerrado que es mucho menor que la impedancia de salida en lazo abierto. Por tanto, se obtiene un método casi perfecto para convertir una fuente de alta impedancia en una fuente de baja impedancia.

Es importante recordar que el valor que lee el ADC corresponde a la tensión de salida del divisor de tensión y no al valor entregado por el sensor de humedad, por lo tanto, en la rutina de software hay que sumar al valor leído por el ADC, la fracción de la tensión que se queda en la otra resistencia.

$$V_{HIH} = V_{R3} + V_{R4}$$

$$V_{HIH} = V_{HIH}\left(\frac{R_3}{R_3 + R_4}\right) + V_{HIH}\left(\frac{R_4}{R_3 + R_4}\right)$$

Ahora bien, la relación entre V_{R4} y V_{R3} es:

$$\frac{V_{R4}}{V_{R3}} = \frac{V_{HIH}\left(\frac{R_4}{R_3 + R_4}\right)}{V_{HIH}\left(\frac{R_3}{R_3 + R_4}\right)}$$

$$\frac{V_{R4}}{V_{R3}} = \frac{R_4}{R_3}$$

$$\boxed{V_{R3} = \left(\frac{R_3}{R_4}\right)V_{R4}}$$

Entonces,

$$V_{HIH} = V_{R3} + V_{R4}$$

$$V_{HIH} = \left(\frac{R_3}{R_4}\right) V_{R4} + V_{R4}$$

$$\boxed{V_{HIH} = V_{R4}\left(1 + \frac{R_3}{R_4}\right)} \qquad \text{[Ec. 2]}$$

Por lo tanto, de esta forma se puede obtener el valor entregado por el sensor de humedad a partir del valor leído por el ADC (proveniente del divisor de tensión), y esta ecuación es la que debe programarse en la rutina para convertir la tensión a humedad.

5.2.6 Bloque Convertidor A/D de 10 bits

En la Figura 5.13 se muestra la conexión entre el bloque ADC con los demás bloques.

La familia de microcontroladores PIC32MX posee un convertidor analógico-digital con una resolución de 10 bits. Como se puede ver en la Figura 5.13, este módulo se encarga de convertir las tensiones provenientes del bloque Procesador Analógico a datos digitales para que la unidad central del microprocesador los manipule según corresponda.

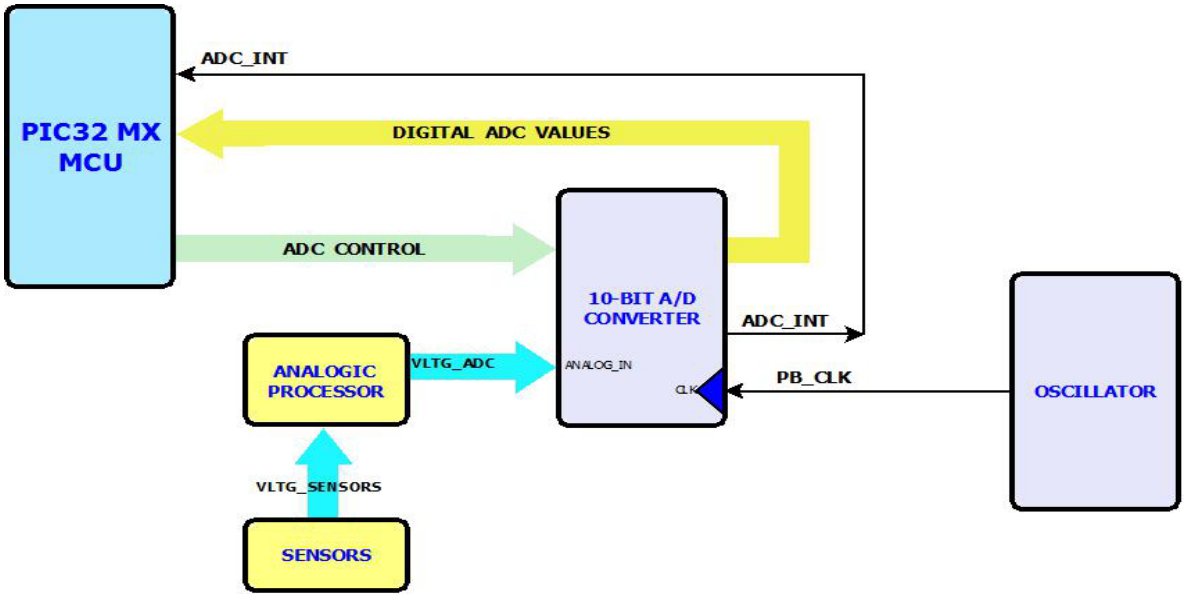

Figura 5.13 Conexión del bloque ADC con los demás bloques

De la Figura 5.13 se observa que, el procesador analógico se encarga de acondicionar las tensiones entregadas por los sensores para que estas tengan niveles adecuados para la lectura del ADC. El ADC posee hasta 16 diferentes entradas analógicas. En nuestro caso se hizo de uso de cuatro entradas (humedad, temperatura, y dos que fueron conectadas al potenciómetro que simula otros sensores).

Además, el ADC posee registros de control con los cuales se regula su funcionamiento, ya que existen diferentes formas en que este se puede configurar, dependiendo de las necesidades del usuario. El PIC32MX MCU se encarga de pasar a los registros de control del ADC los valores adecuados.

El bloque Oscilador se encarga de proveer al ADC con una señal de reloj, la cual es utilizada por los dispositivos periféricos del PIC32, con la cual el ADC se sincroniza para llevar a cabo las conversiones analógico-digital. Esta frecuencia también determina los tiempos de adquisición y conversión del ADC, y cuya suma es el tiempo total de muestreo del ADC.

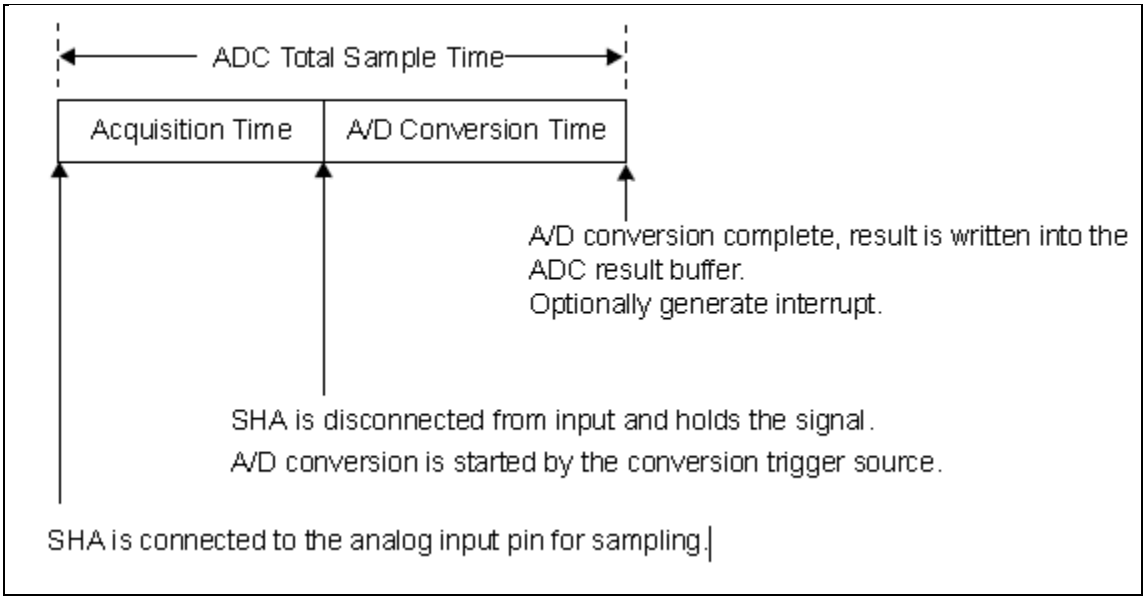

Figura 5.14 Secuencia de Muestreo/Conversión del ADC[8]

Al principio de la secuencia de muestreo/conversión del ADC, el Sample and Hold Amplifier (SHA) se conecta a una de las entradas analógicas para que empiece el muestreo. El tiempo durante el cual el SHA está conectado a la entrada y se encuentra muestreando se denomina tiempo de adquisición. En el momento que la entrada analógica es desconectada del SHA y este retiene la

[8] Tomada de [10]

señal medida y la conversión a digital es iniciada por una de las fuentes de inicio (trigger source), el tiempo de adquisición se acaba y empieza el tiempo de conversión analógico-digital. Cuando esta conversión se completa y el resultado se guarda en uno de los buffers del ADC, se termina el tiempo de conversión y con esto el tiempo total de muestreo del ADC. Como se puede ver de la Figura 5.14, se puede generar una interrupción cuando se finalizó con la secuencia de muestro y conversión. Esta interrupción se ve reflejada en el diagrama de bloques de la Figura 5.13, donde el MCU recibe la interrupción y es el momento en el cual se procede a hacer lectura de los datos digitales que entrega el ADC.

En la Figura 5.15 se observa el diagrama de bloques interno del ADC de 10 bits que posee la familia de microcontroladores PIC32MX.

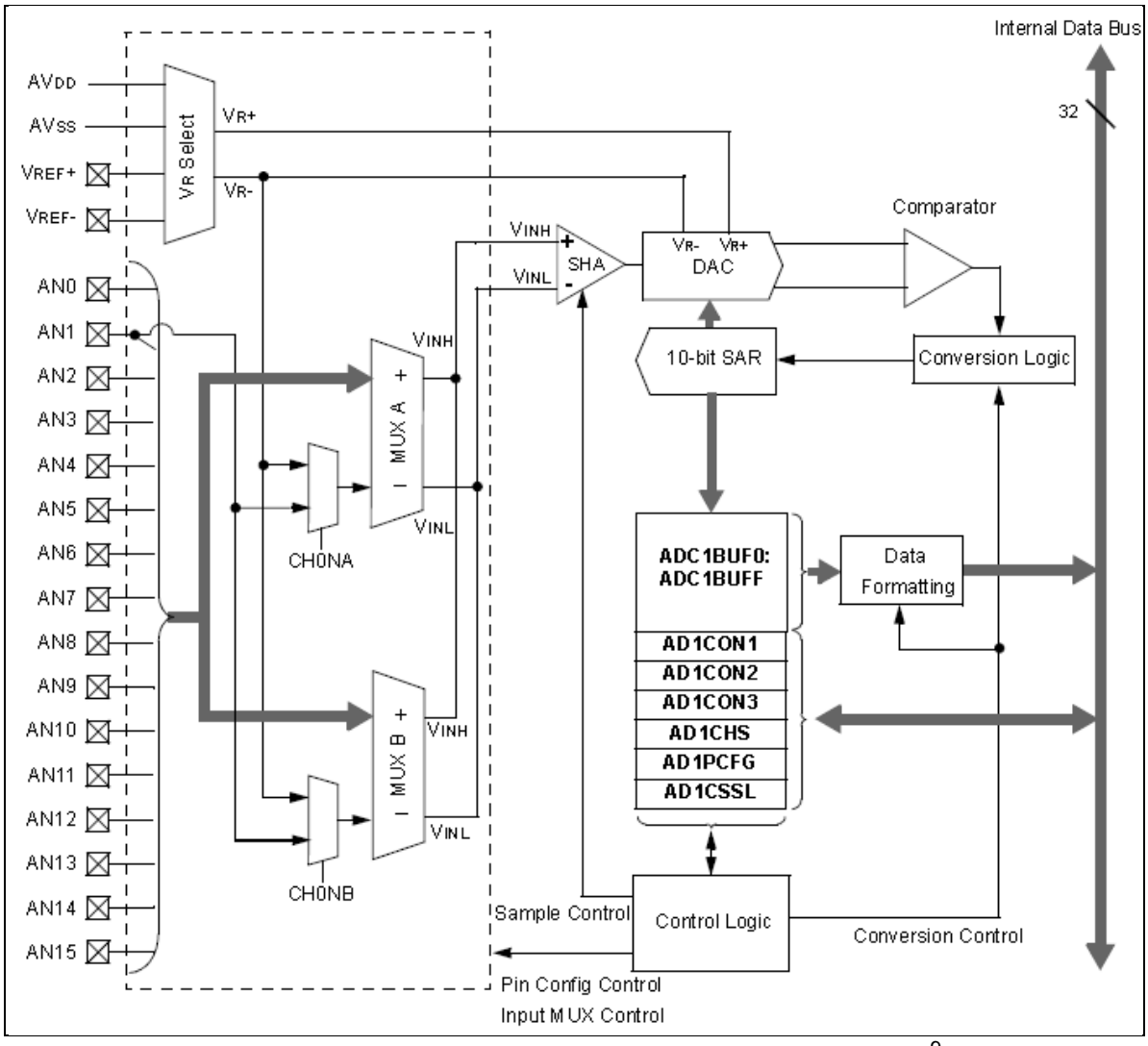

Figura 5.15 Diagrama de bloques del ADC de 10 bits[9]

[9] Tomada de [10]

Las señales de las entradas analógicas, como ya se mencionó provienen del procesador analógico que se encarga de adecuar las tensiones a valores permitidos por el ADC, se configuraron de la siguiente manera:

Tabla 5.7 Conexión de las señales al ADC de 10 bits

PIN	SEÑAL
AN0	Temperatura
AN1	Potenciómetro
AN2	
AN3	Humedad

Las tensiones de referencia para la conversión utilizadas fueron las de AV_{DD} (3.3 V) y AV_{SS} (0 V). Estas tensiones son las tensiones por defecto que trae el PIC32MX. Si uno quisiera una mejor resolución se pueden aplicar tensiones a los pines V_{REF+} y V_{REF}- ajustándolas a los valores necesarios, dependiendo de las señales que se conecten a los pines de entrada. Como en el procesador analógico se llevaron las tensiones de los sensores a niveles que se encuentran entre 0 y 3.3 V, no fue necesaria la utilización de tensiones de referencia externas.

El SHA se conecta a las entradas analógicas a través de uno de los dos multiplexores(A y B) presentes en el ADC. En nuestro caso solo se utilizo solo el MUX A. También el ADC tiene la capacidad de hacer un escaneo de una serie de entradas utilizando un solo multiplexor. Esta opción se utilizo para realizar la lectura en orden de las entradas de AN0 a AN3.

Una vez que se realizó la conversión de las entradas analógicas estas se almacenan en los buffers internos que posee el ADC, esperando que los valores sean leídos por el MCU. Los valores pueden ser leídos en 8 formatos diferentes, según lo desee el usuario.

El ADC se puede programar por medio de sus registros de control de forma que genere una interrupción al llevarse a cabo un número n de conversiones. Como se utilizaron cuatro entradas analógicas, se programó la interrupción para que se diera al término de la cuarta conversión, con lo cual ya el MCU sabe que puede leer los valores digitales de las entradas y procesarlos.

El “trigger source” utilizado para que el ADC terminara de muestrear la señal y empezara a convertirla fue un contador interno, en la configuración que se denomina “Auto Conversión”.

A forma de resumen, en la Tabla 5.8 se presenta como se configuró el ADC.

Tabla 5.8 Configuración del módulo ADC de 10 bits

Registro de Función Especial	Configuración
AD1CON1 (ADC Control Register 1)	• En principio, ADC apagado • Continuar operación cuando se entra a estado de depuración (DEBUG) • Continuar operación cuando se entra a estado ocioso (IDLE) • Formato de salida de los datos: entero de 16 bits • Contador interno termina el muestreo e inicia conversión • Muestreo empieza inmediatamente después de que la ultima conversión termino (auto-sampling) • Operación normal, los buffers van a ser sobrescritos por la secuencia de conversión siguiente
AD1CON2 (ADC Control Register 2)	• Tensiones de referencia: AV_{DD}, AV_{SS} • Deshabilitar modo de calibración de offset • Escanear entradas seleccionadas conectadas al MUX A • Generar interrupción al final de la cuarta secuencia de muestreo/conversión • Buffers configurados como una sola palabra de 16 bits • Usar siempre las entradas conectadas al MUX A
AD1CON3 (ADC Control Register 3)	• Fuente de reloj de conversión: reloj del bus de periféricos (PB_{CLK}) • Tiempo de auto-muestreo: 31 T_{AD} • Selección de reloj de conversión: 128 T_{PB}=T_{AD}
AD1CHS (ADC Input Select Register)	• Entrada negativa para canal 0 es la tensión de referencia negativa • En principio, entrada positiva para canal 0 es AN0
AD1PCFG (ADC Port Configuration)	• Configurar como entradas analógicas a AN0, AN1, AN2, AN3
AD1CSSL (ADC Input Scan Select)	• Escoger entradas analógicas AN0, AN1, AN2 y AN3 como entradas para escanear
IEC1 (Interrupt Enable Control Register 1)	• Habilitar la interrupción del ADC
IPC6 (Interrupt Priority Control Register 6)	• Interrupción del ADC con prioridad de 6 • Interrupción del ADC con sub-prioridad de 3

El modo de escaneo permite hacer un recorrido de las 4 entradas analógicas conectadas al MUX A del ADC, donde se inicia en la AN0 y se va recorriendo cada entrada hasta llegar terminar en la AN3. Al terminarse la conversión de la entrada AN3, se genera la interrupción, la cual es atendida por el MCU.

El módulo ADC puede usar un oscilador RC interno o el reloj del bus de periféricos como fuente de reloj. En nuestra configuración se usa el reloj del bus

de periféricos como fuente de reloj para la conversión. Dicho reloj se configuro en el bloque Oscilador a una frecuencia de 80 MHz, lo que equivale a un periodo del reloj de bus de periféricos (T_{PB}) de 12.5 ns.

El módulo ADC posee reloj de periodo T_{AD} el cual controla el tiempo de conversión, debido a que el convertidor A/D tiene una tasa de conversión máximo a la cual las conversiones pueden ser completadas. La conversión A/D requiere de 12 periodos de reloj (12 T_{AD}) como mínimo. Para conversiones A/D correctas, el reloj de conversión del ADC (T_{AD}) debe ser escogido para asegurar un periodo mínimo de 83.33 ns [10]. Ahora bien, el T_{AD} escogido mediante el registro de control AD1CON3 es de:

$$T_{AD} = 2 * \left(T_{PB}(ADCS + 1)\right)$$

Donde ADCS es el valor escogido para el pre-escalador del reloj del bus de periféricos, en nuestro caso ADCS=63. Por lo tanto,

$$T_{AD} = 2 * \left(12.5\ ns(63 + 1)\right)$$
$$\boxed{T_{AD} = 1.6\ \mu s}$$

Valor que es aproximadamente 19 veces mayor a los 83.33 ns requeridos para asegurar una buena conversión.

5.2.7 Bloques Contador de 32 bits

Los temporizadores o contadores son útiles para generar periódicamente eventos de interrupción basados en una escala de tiempo precisa, ya sea para aplicaciones de software o sistemas de tiempo real [11]. Los temporizadores utilizados en nuestra aplicación son dos temporizadores de 32 bits formados por la unión de dos temporizadores de 16 bits cada uno, en lo que se denomina uso como “Tipo B”, debido a sus características funcionales. El PIC32MX posee 5 contadores de 16 bits.

En la Figura 5.16 se muestra el diagrama de bloques interno de un contador de 32 bits.

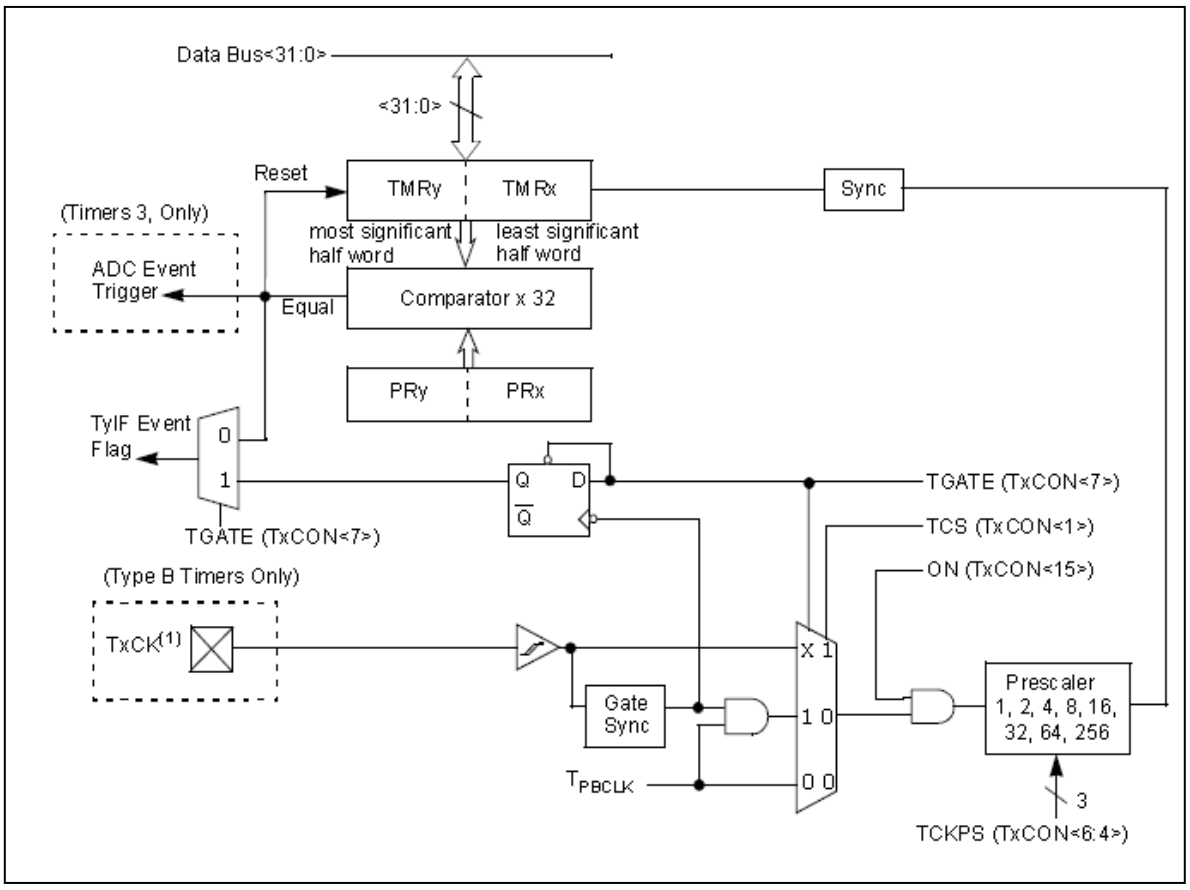

Figura 5.16 Diagrama de bloques de un contador de 32 bits[10]

De la Figura 5.16 se observa que la señal de reloj con la que trabajan los contadores proviene del reloj del bus de periféricos (T_{PBCLK}). Esta señal de reloj es ajustada por medio de un pre-escalador, el cual se determina por medio de un registro de control del temporizador. Se pueden usar pre-escaladores de 1, 2, 4, 8, 16, 32, 64, y hasta 256. Los temporizadores tipo A y tipo B que trabajan con una pre-escala de 1:1 operan a una frecuencia de reloj igual a la del bus de periféricos, e incrementa la cuenta del temporizador en cada flanco ascendente. Temporizadores que utilizan una pre-escala igual a N operan a una frecuencia de PBCLK/N, y el contador incrementa su cuenta cada N flanco ascendente. La cuenta continua asciendo hasta que la cuenta es igual al periodo determinado, momento en el cual se crea una interrupción. Luego el conteo se reinicia a cero y empieza de nuevo el ciclo.

El Temporizador utiliza una frecuencia de f_{PB}/N, donde N es el pre-escalador utilizado. Por lo tanto, el tiempo que tarda el contador en incrementar su cuenta en uno es $T_{PB}*N$ donde T_{PB} es el periodo del reloj del bus de periféricos.

[10] Tomada de [11]

De esta forma, para determinar el valor con que se debe cargar el registro de periodo:

$$\boxed{Periodo = \frac{tiempo\ deseado(en\ segundos)}{N * T_{PB}}}$$

El valor del pre-escalador utilizado corresponde a 256 (el máximo pre-escalador posible), con lo cual se puede programar el periodo de los registros para que correspondan a intervalos de tiempo mayores a los que se obtendrían si se utiliza un pre-escalador menor. La frecuencia programada del bus de periféricos es de 80 MHz, la cual la entrega el bloque Oscilador.

Debido a que el registro de periodo corresponde a un registro de 32 bits, el máximo valor que se puede cargar es de 2^{32}-1= 4294967295 (0xFFFFFFFF en hexadecimal) lo cual corresponde a un tiempo, en segundos de:

$$tiempo\max(en\ segundos) = Periodo\ max * N * T_{PB}$$
$$tiempo\max(en\ segundos) = 42944967295 * 256 * \left(\frac{1}{80\ MHz}\right)$$
$$\boxed{tiempo\max(en\ segundos) = 13743.89534}$$

Este tiempo equivale a 229.065 minutos=3.818 horas=3 horas y 49 minutos. Este el máximo que se puede programar como intervalo para guardar mediciones.

Es importante saber que, en el modo de contadores de 32 bits, el contador se forma combinando un contador tipo B número par (denominado TemporizadorX) con un contador tipo B número impar (denominado TemporizadorY), consecutivos. Por ejemplo Temporizador 2 con Temporizador 3 o Temporizador 4 con Temporizador 5. El número de parejas de contadores depende de la variante del dispositivo dentro de la familia. Para nuestro caso, se pueden formar dos contadores de 32 bits, ya que el PIC32MX utilizado posee 5 contadores de 16 bits. El comportamiento específico en el modo de 32 bits es:

- TemporizadorX es el contador maestro y TemporizadorY es el contador esclavo.
- El registro de cuenta del TemporizadorX es la media palabra menos significante del valor de la cuenta de 32 bits.
- El registro de cuenta del TemporizadorY es la media palabra más significante del valor de la cuenta de 32 bits.

- El registro de periodo del TemporizadorX es la media palabra menos significante del valor del periodo de 32 bits.
- El registro de periodo del TemporizadorX es la media palabra más significante del valor del periodo de 32 bits.
- Los bits de control del TemporizadorX configuran la operación del contador de 32 bits.
- Los bits de control del TemporizadorY no tienen efecto sobre la operación del contador de 32 bits.
- Los bits de interrupción y estado del TemporizadorX son ignorados.
- El TemporizadorY provee la habilitación de interrupción, la bandera de interrupción y los bits de control de prioridad de interrupción.

En la Tabla 5.9 se muestra un resumen de la configuración de los temporizadores. Cabe mencionar antes que los temporizadores utilizados fueron el 2 y el 3 para formar el temporizador de 32 bits denominado “Timer Interval” y los temporizadores 4 y 5 para formar el temporizador de 32 bits denominado “Timer Average”.

En la Figura 5.17 se muestra como se enlazan los dos contadores utilizados con el resto de bloques del sistema.

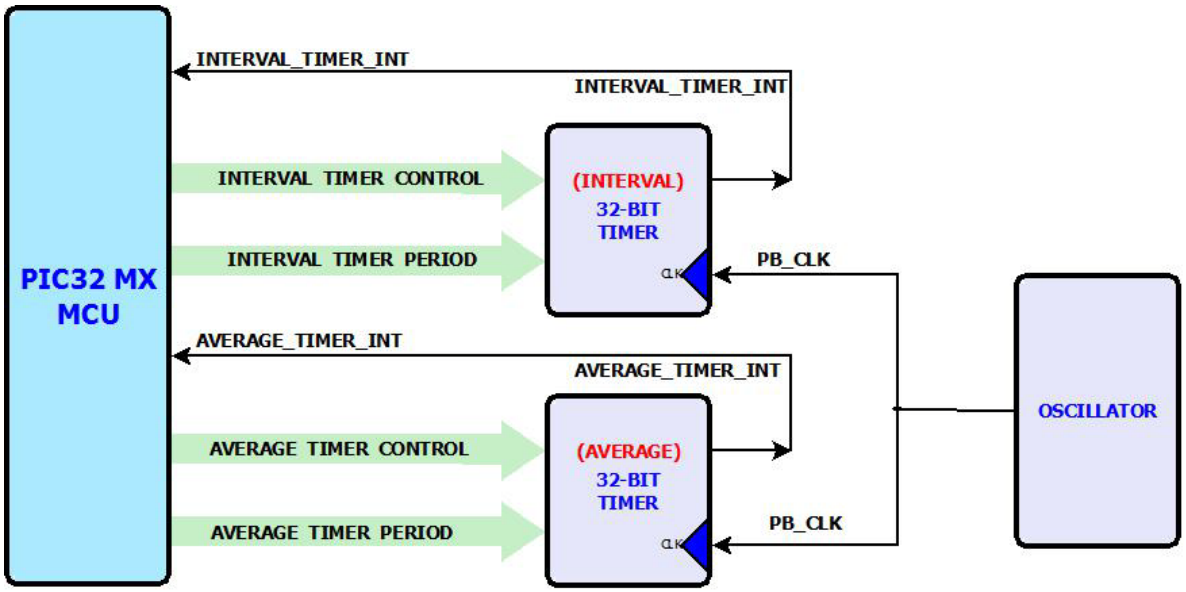

Figura 5.17 Conexión de los bloques temporizadores de 32 bits con los demás bloques

Tabla 5.9 Configuración de los contadores de 32 bits

TEMPORIZADOR	Registro de Función Especial	Configuración
INTERVAL (Temporizadores 2 y 3)	T2CON (Timer 2 Control Register)	• Encender el Temporizador 2 • Continuar operación cuando se entra a estado de depuración (DEBUG) • Continuar operación cuando se entra a estado ocioso (IDLE) • Pre-escalador de 256 • Modo de contador de 32 bits • Fuente de reloj derivada del PBCLK
	PR2 (Timer 2 Period Register)	• Se carga con el valor de la media palabra menos significante del periodo de 32 bits
	PR3 (Timer 3 Period Register)	• Se carga con el valor de la media palabra más significante del periodo de 32 bits
	IEC0 (Interrupt Enable Control Register 0)	• Habilitar la interrupción del Temporizador 3
	IPC3 (Interrupt Priority Control Register 3)	• Interrupción del Temporizador 3 con prioridad de 7
AVERAGE (Temporizadores 4 y 5)	T4CON (Timer 4 Control Register)	• Encender el Temporizador 4 • Continuar operación cuando se entra a estado de depuración (DEBUG) • Continuar operación cuando se entra a estado ocioso (IDLE) • Pre-escalador de 256 • Modo de contador de 32 bits • Fuente de reloj derivada del PBCLK
	PR4 (Timer 4 Period Register)	• Se carga con el valor de la media palabra menos significante del periodo de 32 bits
	PR5 (Timer 5 Period Register)	• Se carga con el valor de la media palabra más significante del periodo de 32 bits
	IEC0 (Interrupt Enable Control Register 0)	• Habilitar la interrupción del Temporizador 5
	IPC5 (Interrupt Priority Control Register 5)	• Interrupción del Temporizador 5 con prioridad de 2

En la Figura 5.17 vemos como el bloque Oscilador se encarga de proveer la señal de reloj a los contadores de 32 bits. Esta corresponde al reloj del bus de periféricos, la cual se configuró para 80 MHz.

El PIC32MX MCU se encarga de configurar los registros de control de cada temporizador para así controlar su funcionamiento, además que se encarga de llenar los registros que determinan el periodo con el valor correspondiente para cada contador. Cuando la cuenta de los temporizadores iguala al valor cargado en los registros de periodo, entonces se produce una interrupción. El MCU se encarga entonces de llevar a cabo las rutinas de servicio de interrupción, dependiendo de cual temporizador generó la misma.

El temporizador denominado “Interval” se encarga de contar hasta el número de minutos determinado por el usuario en los cuales quiere guardar una medición. Por ejemplo, si el usuario desea que se guarden mediciones cada 10 minutos, este temporizador es quien se encarga de determinar cuándo transcurrieron esos 10 minutos para entonces guardar en memoria el promedio de las mediciones realizadas durante esos 10 minutos.

Con el temporizador “Average”, cada cinco segundos se están convirtiendo los datos analógicos a digital con el fin de tomar varias mediciones durante el intervalo escogido por el usuario, y no solo una medición cuando se cumpla el intervalo. De esta forma cuando se guardan las mediciones, estas corresponden al promedio del total de mediciones tomadas cada cinco segundos. Además, tomando mediciones cada cinco segundos se elimina también el efecto de las pequeñas variaciones de las señales analógicas, por la distribución normal que posee naturalmente un sistema de mediciones.

En la Figura 5.18 se muestran las rutinas de servicio de interrupción para cada temporizador.

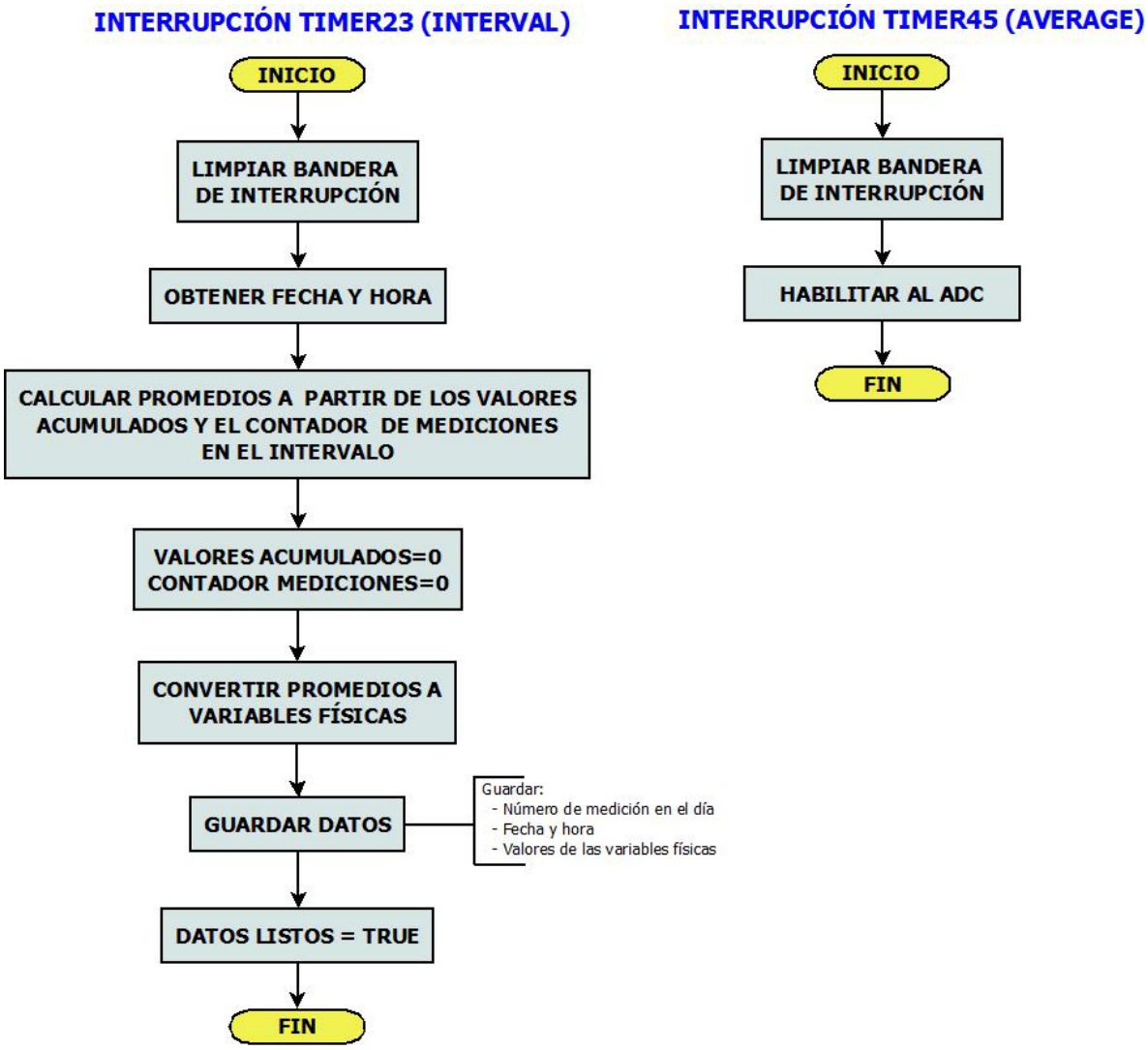

Figura 5.18 Diagrama de flujo de las rutinas de servicio de interrupción de los contadores

Como se mencionó anteriormente, con el temporizador de 32 bits, formado por los temporizadores 2 y 3, denominado “Interval” es donde se programa el intervalo de tiempo que desea el usuario que transcurra entre mediciones. Cuando se cumple este tiempo, entonces se genera la interrupción, y la cuenta se reinicia en cero. Cuando se entra a la rutina de servicio de interrupción entonces:

- Se limpia la bandera de interrupción.
- El MCU obtiene la fecha y hora de la medición, la cual es entregada por el módulo RTCC.
- Se calculan los promedios de las mediciones que se hicieron durante el intervalo.

- Se reinician los valores acumulados y el contador de mediciones a cero, esto para reiniciar un nuevo intervalo de tiempo.
- Se convierten los datos promedios (corresponden a tensiones analógicas promedio durante el intervalo de tiempo) a las diferentes variables físicas (humedad, temperatura, dióxido de carbono y luminosidad).
- Se guardan los datos en memoria: numero de medición en el día, fecha y hora, y valores de las variables físicas.
- Se actualiza una bandera de “datos listos” a verdadera.

En el caso de la rutina de servicio de interrupción del temporizador de 32 bits, formado por los temporizadores 4 y 5, denominado “Average”, este cuenta hasta 5 segundos y es entonces cuando se genera la interrupción y se reinicia la cuenta automáticamente. Lo que se hace en esta rutina es simplemente limpiar la bandera de interrupción y habilitar el ADC para que empiece a convertir los datos analógicos.

Ahora bien, una vez que ya se ha explicado el funcionamiento tanto de los contadores de 32 bits y del ADC, en la Figura 5.19 se muestra un diagrama de flujo útil para explicar cómo se enlazan las interrupciones del contador “Average” y del ADC.

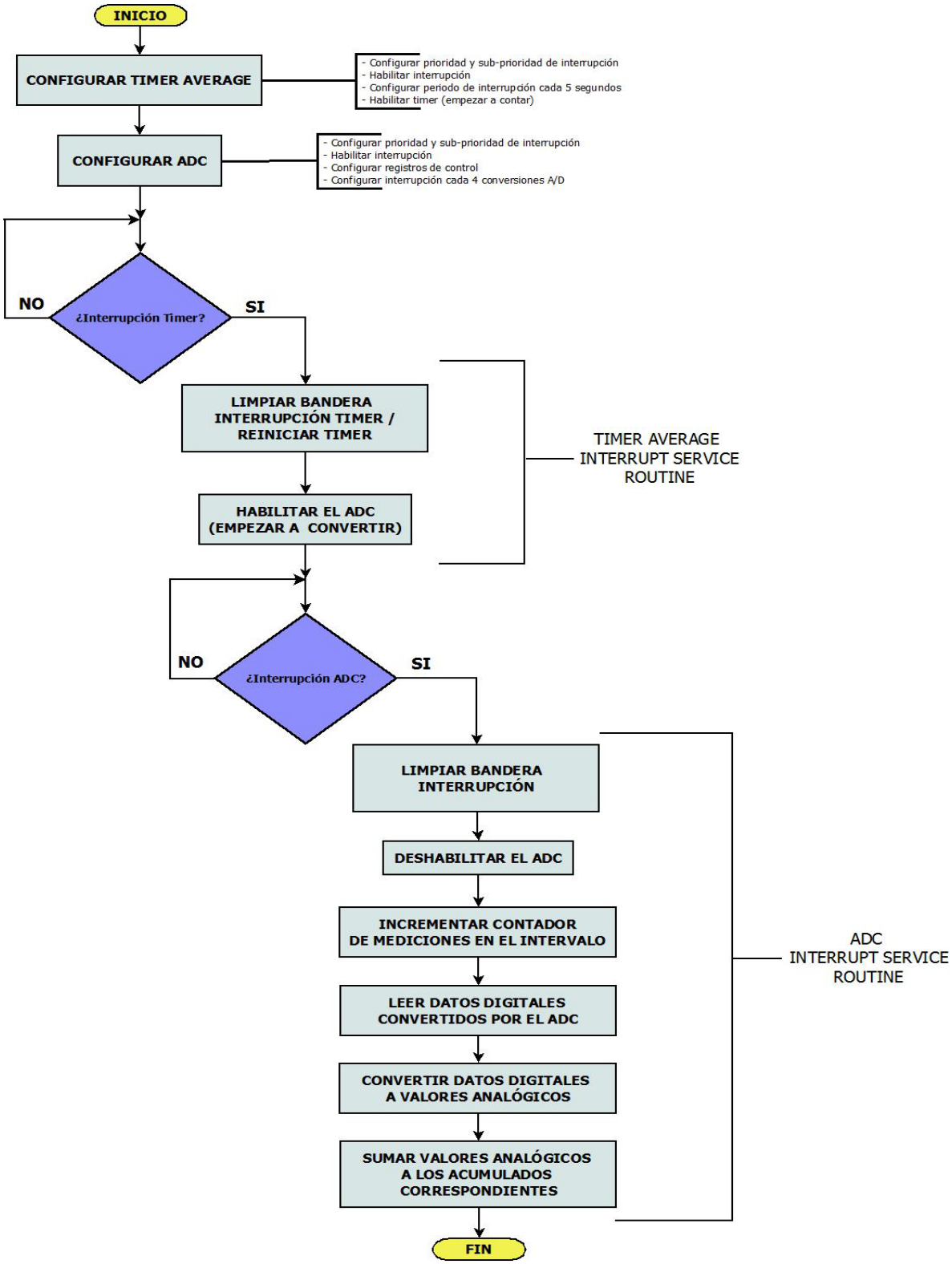

Figura 5.19 Diagrama de flujo relacionando las interrupciones del Temporizador y del ADC

Lo primero es configurar el contador de 32 bits, denominado “Timer Average” para que produzca una interrupción cada 5 segundos. Se habilita la interrupción de este contador y se habilita al mismo para que empiece su conteo.

Inmediatamente, se configura el ADC como se observa en la Tabla 5.8 para que vaya escaneando y convirtiendo los canales de AN0 a AN3, y genere una interrupción al fin de la cuarta conversión. Se habilita la interrupción, pero el ADC todavía no está habilitado.

Al cabo de 5 segundos, el “Timer Average” genera la interrupción. Paso siguiente se limpia la bandera de interrupción de este contador y se reinicia la cuenta. Por lo tanto, se convierte en un ciclo sin fin donde cada 5 segundos se da la interrupción. Es aquí donde se habilita al ADC para que empiece con la primera de las conversiones (entrada AN0).

El ADC va continuar escaneando y convirtiendo las entradas analógicas hasta que convierta la última de las entradas (AN3). Es aquí cuando se genera la interrupción del ADC.

Una vez que se genera la interrupción, el MCU atiende la misma. La bandera de interrupción se limpia, se deshabilita al ADC para que no reinicie conversiones hasta que se hayan cumplido los 5 segundos de nuevo.

Ahora bien, se están tomando mediciones cada 5 segundos, esto es para tomar varias mediciones y realizar un promedio de cada canal, para que, cuando se cumpla con el tiempo establecido en el “Timer Interval”, entonces el dato que se guarda es el promedio de todas las mediciones que se hicieron durante el intervalo de adquisición de datos.

Por lo tanto, una vez deshabilitado el ADC, se procede a incrementar un contador, el cual lleva el número total de mediciones en el intervalo. Se leen los datos digitales almacenados en los buffers internos del ADC. El MCU se encarga de pasar dichos valores digitales de nuevo a tensiones analógicas para ir sumando a unas variables que llevan la suma (por canal) de los valores acumulados hasta el momento. Esto con el fin de que, cuando se cumpla el intervalo de tiempo programado en el otro contador, entonces se divida el valor acumulado entre el número total de mediciones y así obtener el promedio de las mediciones.

5.2.8 Bloques Módulo USB y Computadora

El PIC32MX en esta aplicación está funcionando en modo dispositivo (Device Mode). Los dispositivos USB aceptan comandos y datos del anfitrión (host) y responden a peticiones por datos. Los dispositivos USB realizan entonces funciones periféricas. Las siguientes características generalmente describen un dispositivo USB [12]:

- La funcionalidad puede ser específica por clase o por vendedor.
- Consume 100 mA o menos del bus antes de la configuración.
- Puede consumir hasta 500 mA del bus después de una negociación exitosa con el anfitrión.
- Puede soportar protocolos: low-speed, full-speed o high-speed.
- Soporta transferencias tanto de control como de datos requeridas por la aplicación.
- Puede ser alimentado por sus propios medios o por medio del bus USB.

El módulo USB se encarga de realizar la comunicación entre el PIC32MX MCU y la computadora. Por medio de él, el MCU espera que la computadora le envíe comandos para que él los procese. Además, el bloque USB genera todas las interrupciones que corresponden al protocolo de comunicación por puerto USB. Si el comando enviado por la computadora corresponde a uno de los comandos programados en el MCU, entonces se entra en la rutina correspondiente. Básicamente hay dos diferentes comandos: transferir el último dato que se midió y transferir todos los datos que haya guardados en memoria. El MCU se encarga de dividir los datos en paquetes que utiliza el módulo USB para luego enviarlos por medio de él a la computadora.

La computadora a su vez está lista para la recepción de paquetes por parte del módulo USB. Una vez recibido un paquete, la computadora se encarga de procesarlo y formar el dato que fue enviado. En cada paquete se envía:

- 1 bandera por dato (4 banderas en total) para indicar el número de decimales en el dato
- Los 4 datos medidos, separados en parte entera y parte decimal
- Fecha de la medición, separada en día, mes y año
- Hora de la medición, separada en hora, minutos y segundos
- Número de la medición en el día

5.3 Diagrama de flujo del sistema

En la Figura 5.20 se muestra un diagrama de flujo de cómo se comporta el sistema.

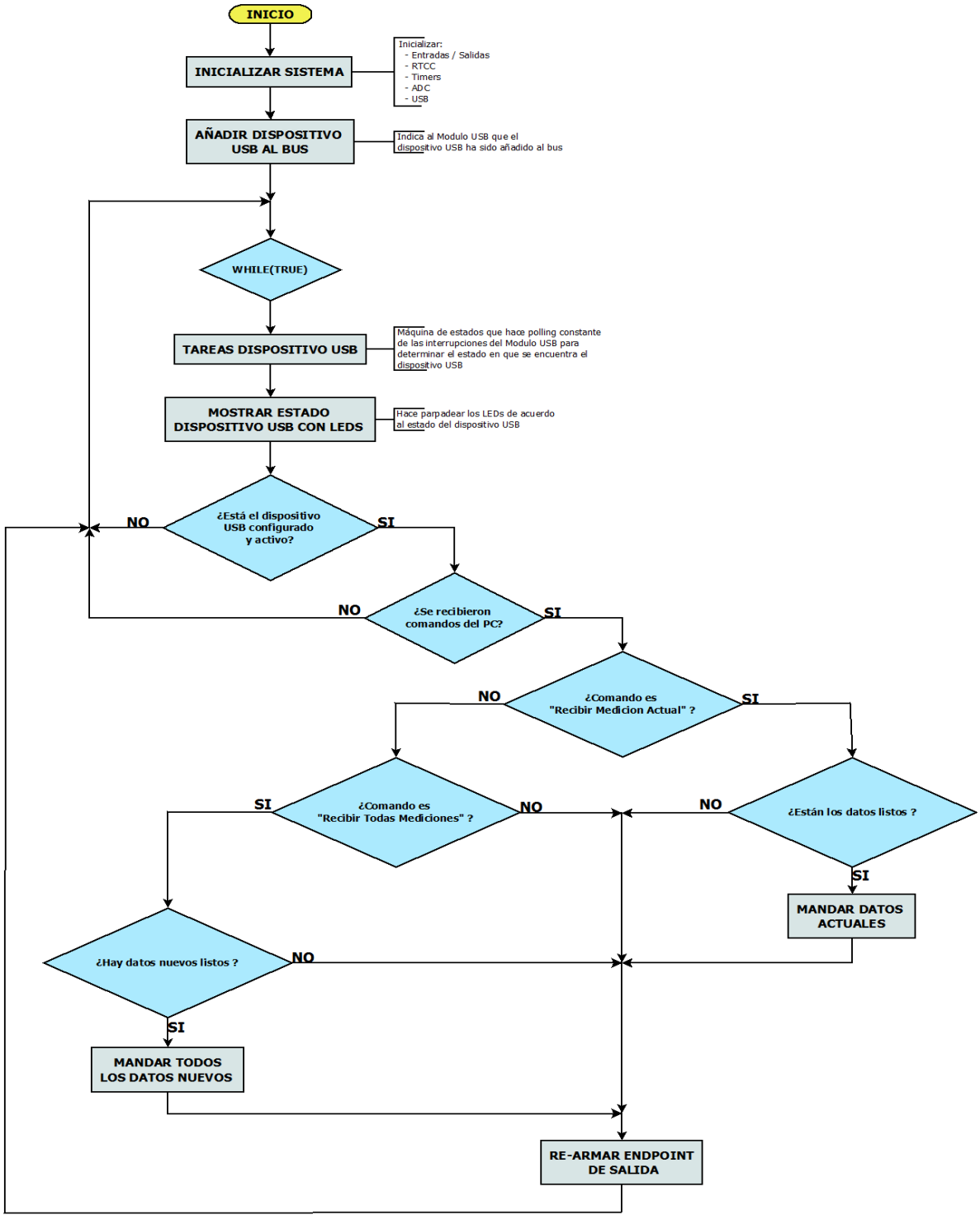

Figura 5.20 Diagrama de flujo del sistema en general

El diagrama de flujo de la Figura 5.20 se puede entender así:

- Lo primero es inicializar todo el sistema. Esto es llevar todos los módulos a un punto inicial donde se configuren de acuerdo a como se explico anteriormente. Se inicializan: las entradas y salidas del PIC32MX, el módulo RTCC, los temporizadores de 32 bits, el ADC, y el módulo USB.

- A continuación, se ejecutan instrucciones para añadir el dispositivo USB al bus. Con esto, ya la computadora está en capacidad de reconocer al dispositivo PIC32MX como dispositivo USB.

- Una vez inicializado todo el sistema, se entra a un ciclo while (true), donde periódicamente se está preguntando por el estado del módulo USB. Por lo tanto el MCU siempre va estar activo esperando que se le manden comandos por parte de la computadora y además realizando las mediciones de las entradas analógicas.

- Si el módulo USB está corriendo y activo, y llegan nuevos comandos por parte de la computadora, se pasa a revisar que comando se recibió.

- Ahora bien, si el comando es “Recibir Medición Actual”, entonces se procede a mandar la última medición guardada en memoria, que corresponde a la última medición realizada.

- Por otro lado, si se recibió el comando de “Recibir Todas Mediciones”, hay que preguntarse si hay datos nuevos listos para enviar, ya que puede ser que se dé el caso que no haya datos guardados aun, o que ya se han enviado todos los datos guardados y no hay datos nuevos.

- En caso de que si los hubiese, se prosigue a mandar todos aquellos datos que no han sido enviados anteriormente.

- Como punto final, se debe rearmar el punto de salida (endpoint) del módulo USB para estar listos para recibir otro comando nuevo. Una vez rearmado el endpoint, se devuelve al ciclo infinito while.

5.4 Interfaz gráfica de usuario en Java

5.4.1 Generalidades de la interfaz gráfica de usuario

La interfaz grafica donde se maneja casi que toda la acción corresponde a una aplicación escrita en el lenguaje de programación Java. La versión de Java utilizada es 1.6.0_22 (Versión 6 Actualización 22).

Ahora bien, la aplicación se creó NetBeans IDE versión 6.9. Con NetBeans es posible escribir el programa en Java, compilarlo, correrlo y depurarlo. Además crea un archivo .jar (Java ARchive), que permite ejecutar la aplicación escrita en java con solo hacer doble click sobre el icono del archivo jar.

La interfaz grafica que ve el usuario en su computadora es la herramienta principal para recibir, visualizar y procesar las mediciones recibidas por medio del puerto USB de su computadora. Con la interfaz es posible ver las mediciones "en vivo", donde se presentan tanto la última medición recibida junto con la medición anterior, para lograr realizar una pequeña comparación de cómo se está comportando el clima. Además, se pueden recibir todas las mediciones que hayan guardadas en la memoria del PIC32MX. Cuando se reciben todos estos datos de las mediciones, se crea un archivo Excel donde se guardan todas las mediciones recibidas, separadas en hojas por día. Además la hoja Excel es capaz de calcular una serie de estadísticas sobre las mediciones: máximo, mínimo, promedio, varianza y desviación estándar.

La interfaz de usuario también es capaz de realizar gráficos de los datos recibidos en un día, o también graficar las estadísticas calculadas en 2 o más días.

5.4.2 APIs utilizadas para la interfaz gráfica de usuario

Básicamente, para lograr todas las tareas de la interfaz gráfica de usuario, se hizo uso de tres interfaces escritas en Java.

5.4.2.1 JPicUSB[11]

JPicUSB es una clase en Java que utiliza interfaces nativas de Java (JNI, Java Native Interface) y permite a una aplicación Java hacer llamadas a una biblioteca de enlace dinámica (DLL), en este caso jpicusb.dll.

[11] Tomado de [13]

JPicUSB.dll es una biblioteca que implementa todas las funciones de la API (Application Programing Interface) USB de Microchip (mpusbapi.dll) con la diferencia de que está especialmente recompilada para permitir a la clase jPicUSB que haga llamados a sus funciones. JPicUSB proporciona un método fácil para acceder a la biblioteca mpusbapi.dll y utilizar sus funciones.

A la hora de conectar el PIC32MX como dispositivo USB, la primera vez que se conecta es necesario instalar el driver mchpusb, para que la computadora pueda reconocer al PIC32MX como un dispositivo USB.

5.4.2.2 JExcel[12]

JExcel API es una interfaz de fuente abierta que permite a sus usuarios leer, escribir y modificar dinámicamente hojas de Excel. Esta interfaz es capaz de leer datos de libros de Excel 95, 97, 2000, XP y 2003.

5.4.2.3 JChart2D[13]

JChart2D es una interfaz útil para realizar graficas tanto en tiempo real como gráficos estáticos. Esta API está diseñada para mostrar múltiples curvas al mismo tiempo si es necesario.

5.4.3 Diagrama de flujo de la interfaz gráfica de usuario

El flujo del programa se puede entender por medio de la Figura 5.21.

[12] Tomado de [14]
[13] Tomado de [15]

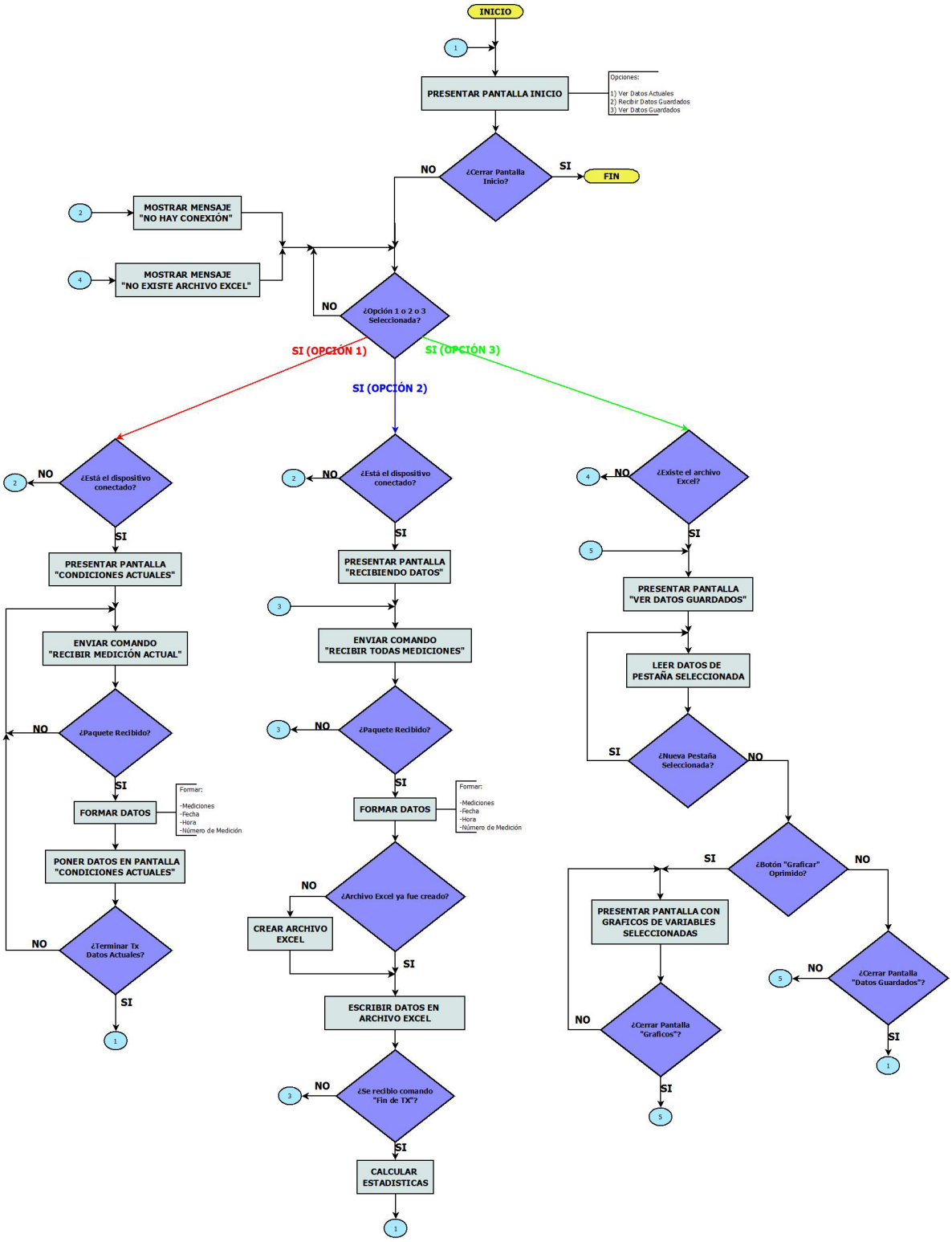

Figura 5.21 Diagrama de flujo de la interfaz gráfica de usuario

El punto inicial de la interfaz gráfica de usuario es la pantalla de inicio.

Figura 5.22 Pantalla de inicio de la interfaz grafica de usuario

La aplicación en Java comienza por presentar la pantalla de inicio de la Figura 5.22. A partir de ahí se puede seleccionar alguna de las tres opciones:

1. Ver Datos Actuales
2. Recibir Datos Guardados
3. Ver Datos Guardados

O simplemente se puede cerrar la aplicación con lo cual se termina la ejecución de la misma.

Si se selecciona la opción **"Ver Datos Actuales"**, pero el dispositivo PIC32MX aun no ha sido conectado al puerto USB, entonces se muestra un mensaje indicando esta condición, como se muestra en la Figura 5.23.

Figura 5.23 Mensaje de no conexión con el dispositivo USB

Ahora bien, si la conexión con el dispositivo USB está estable se presenta la pantalla denominada “Condiciones Actuales”, la cual luce como se muestra en la Figura 5.24.

Cuando se seleccionó la opción de **“Ver Datos Actuales”** es cuando la computadora manda hacia el sistema embebido PIC32MX el comando de “Recibir Medición Actual”. El módulo USB recibe el paquete de bytes enviado por la computadora, y el PIC32MX MCU se encarga de tomar la última medición (por medición entiéndase la fecha y hora, número de medición en el día y los cuatro datos provenientes de los sensores) guardada en memoria y convertirlo a bytes para acomodarlo en un paquete de salida hacia la computadora.

La computadora cuando recibe el paquete entonces se encarga de formar nuevamente los datos y una vez formados, estos se despliegan en la pantalla como se muestra en la Figura 5.24.

Figura 5.24 Pantalla de visualización de las condiciones climatológicas actuales

En la pantalla “Condiciones Actuales” se muestran la última medición realizada por el sistema de recolección de datos y la medición anterior a esta, con el fin de comparar el comportamiento del clima en el último intervalo de tiempo. Se muestran además la fecha y hora actual, para así conocer hace cuanto se recibió la última medición.

El usuario puede también detener la transmisión para observar detenidamente las condiciones climatológicas en un instante de tiempo determinado, y luego volver a reiniciar la transmisión cuando lo desee.

En caso que el usuario cierra la pantalla “Condiciones Actuales”, entonces se deja de enviar el comando hacia el microcontrolador, y se retorna a la pantalla de inicio.

Estando en la pantalla de inicio, si el usuario selecciona la opción **"Recibir Datos Guardados"** pero el dispositivo USB no está conectado correctamente, entonces de nuevo se muestra el mensaje de no conexión de la Figura 5.23. Cuando la conexión con el dispositivo está funcionando, los datos por recibir van a ser guardados en un archivo Excel denominado "DatosSensores.xls". En caso que este archivo aun no haya sido creado, este se crea en la carpeta "Red Sensores" (que debe ser creada anteriormente en la unidad C de la computadora), o sea en la ruta "C:\Red Sensores\". En la Figura 5.25 se muestra el mensaje que es enviado al usuario.

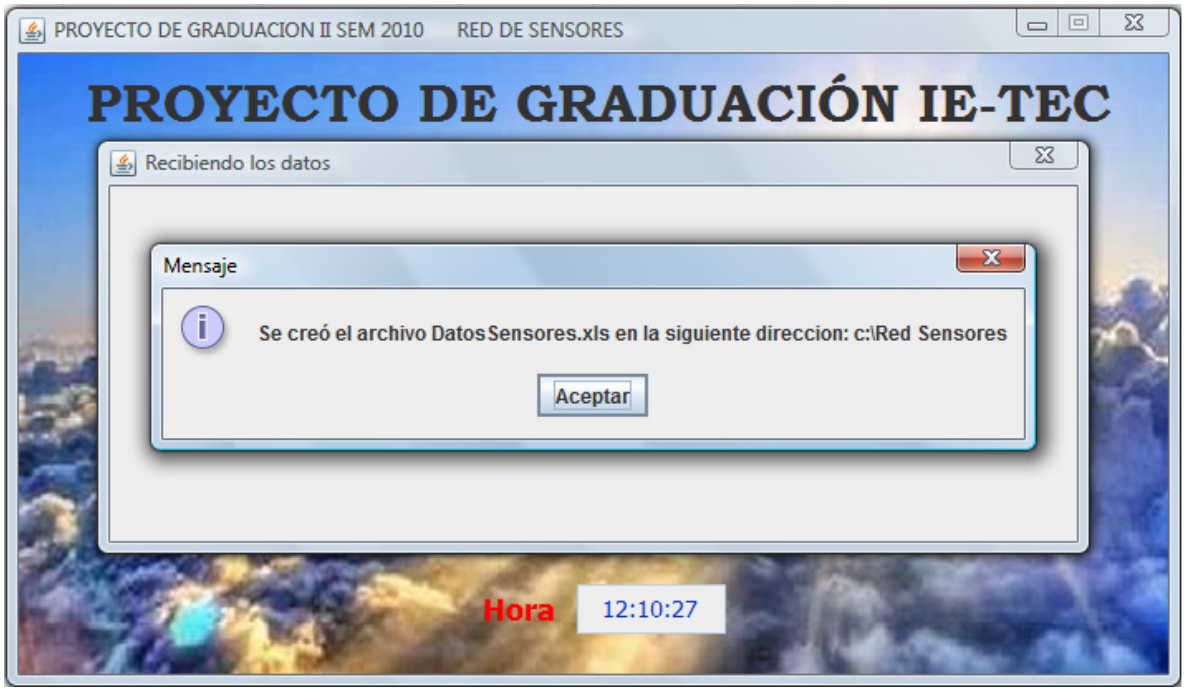

Figura 5.25 Mensaje indicando la creación del archivo Excel

Si el archivo "DatosSensores.xls" ya fue creado anteriormente, entonces, para escribir los datos que van a ser recibidos en él, es necesario que esté cerrado, de lo contrario los datos no podrán ser escritos, y se producirá un daño en el archivo. Se muestra el mensaje de la Figura 5.26 para recordar al usuario de esta condición.

Figura 5.26 Mensaje indicando que archivo Excel debe estar cerrado para la recepción de datos

Luego de que se muestren alguno de los dos mensajes anteriores, la interfaz muestra la pantalla “Recibiendo Datos”, la cual luce como la Figura 5.27.

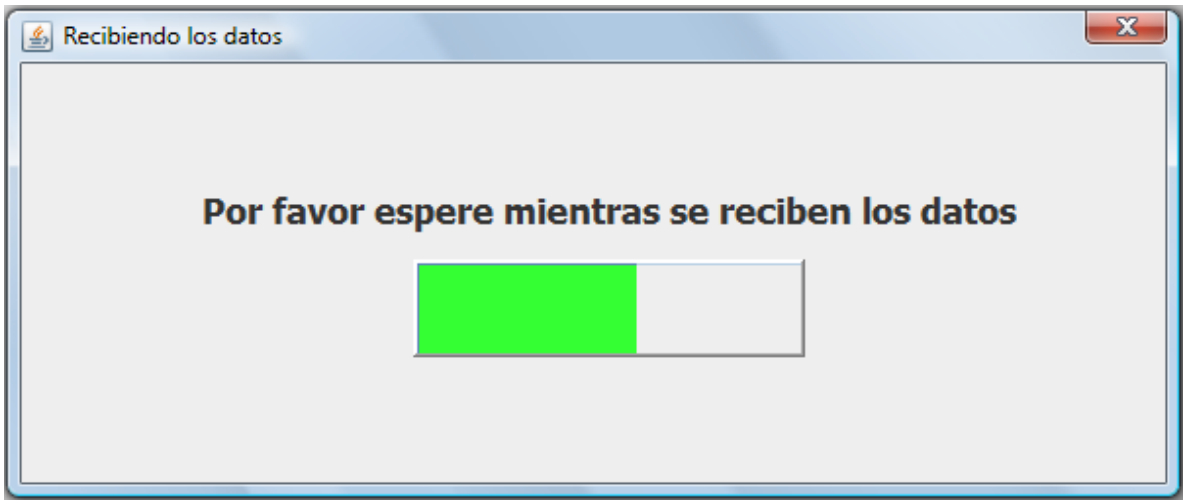

Figura 5.27 Pantalla indicando que se están recibiendo los datos provenientes del PIC32MX

Al mismo tiempo, la computadora envía el comando “Recibir Todas Mediciones” al PIC32MX. Este entonces empieza a enviar, una por una, todas las mediciones guardadas en memoria que **NO** hayan sido enviadas, ya que con un índice se controla cuales mediciones fueron ya enviadas. Una vez que la

computadora recibe un paquete con los bytes que forman los datos, esta forma todos los datos y procede a escribir los mismos en el archivo Excel.

La recepción de datos continua hasta que se hayan enviado todas las mediciones que no habían sido enviadas anteriormente, momento en el cual el PIC32MX MCU envía el comando “Fin de Transmisión”. Una vez que la computadora recibe este comando, sabe que la recepción de datos ha finalizado y se lo indica al usuario.

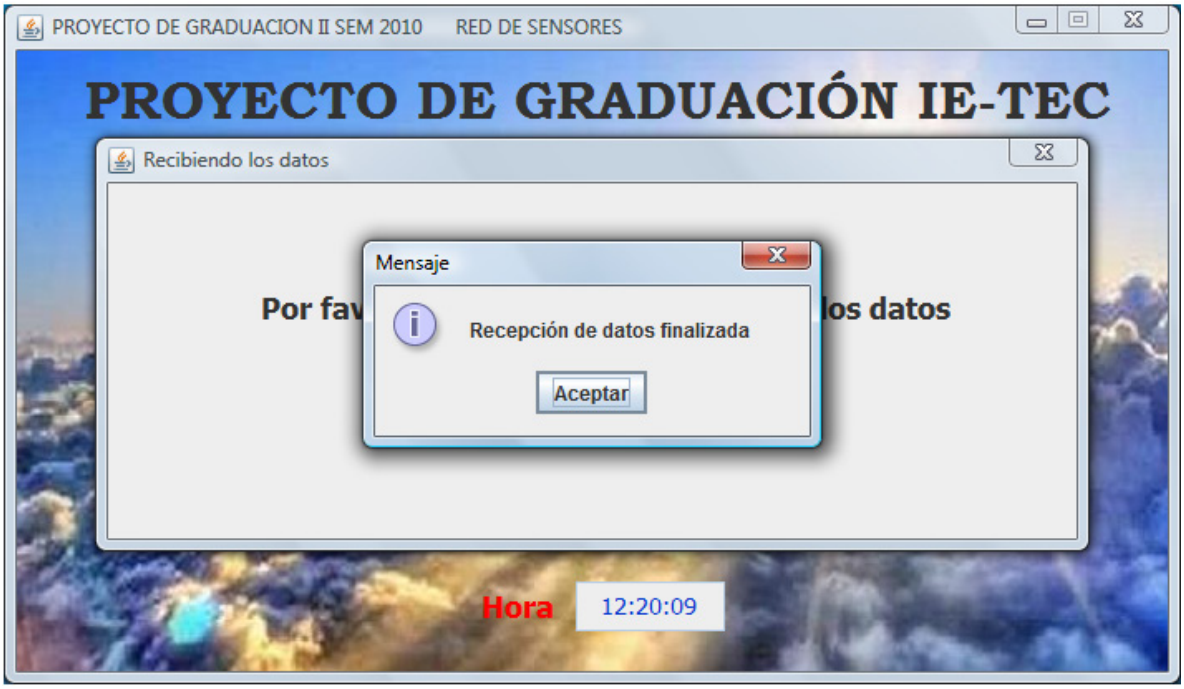

Figura 5.28 Mensaje indicando que la recepción de datos ha finalizado

Una vez que finalizó la recepción de datos, entonces se procede a calcular las estadísticas de los datos que estén guardados en la hoja de Excel. La hoja de Excel con los datos guardados y las estadísticas calculadas luce como la Figura 5.29.

# MEDICION	FECHA (dd-mm-aaaa)	HORA (hh:mm:ss)	TEMPERATURA (°C)	DIÓXIDO DE CARBONO (ppm)	LUMINOSIDAD (Lux)	HUMEDAD RELATIVA (%)
1	17-11-2010	15:14:00	25.50	2.40	2.40	76.20
2	17-11-2010	15:24:00	25.60	2.40	2.40	75.10
3	17-11-2010	15:34:00	25.30	2.40	2.40	77.30
4	17-11-2010	15:44:00	25.50	2.40	2.40	76.40
5	17-11-2010	15:54:00	25.40	2.40	2.40	77.40
6	17-11-2010	16:04:00	25.50	2.40	2.40	76.90
7	17-11-2010	16:14:00	25.20	2.40	2.40	77.40
8	17-11-2010	16:24:00	25.20	2.40	2.40	79.30
9	17-11-2010	16:34:00	25.60	2.40	2.40	76.70
10	17-11-2010	16:44:00	26.10	2.40	2.40	75.30
11	17-11-2010	16:54:00	26.30	2.40	2.40	75.10
12	17-11-2010	17:04:00	26.50	2.40	2.40	75.00
13	17-11-2010	17:14:00	26.50	2.40	2.40	75.50
14	17-11-2010	17:24:00	26.60	2.40	2.40	76.30
15	17-11-2010	17:34:00	26.50	2.40	2.40	76.40
16	17-11-2010	17:44:00	26.50	2.40	2.40	75.90

Figura 5.29 Apariencia de la hoja Excel con los datos guardados

Como se puede ver, se almacena el número de medición de cada una de las mediciones, la fecha y hora en que se dio esa medición, además de los valores de temperatura, dióxido de carbono, luminosidad y humedad que se recolectaron a esa específica hora. Ahora bien, las estadísticas de los datos guardados se calculan más abajo en la hoja, después del último dato recibido en ese día.

48	17-11-2010	23:03:59	24.50	2.41	2.41	84.40
49	17-11-2010	23:13:59	24.50	2.41	2.41	84.40
50	17-11-2010	23:23:59	24.50	2.41	2.41	84.40
51	17-11-2010	23:33:59	24.50	2.41	2.41	84.30
52	17-11-2010	23:43:59	24.50	2.41	2.41	84.30
53	17-11-2010	23:53:59	24.50	2.41	2.41	84.20

VALOR	TEMPERATURA (°C)	DIÓXIDO DE CARBONO (ppm)	LUMINOSIDAD (Lux)	HUMEDAD RELATIVA (%)
MAXIMO	26.60	2.41	2.41	85.00
PROMEDIO	25.23	2.37	2.37	79.53
MINIMO	24.40	2.02	2.02	73.30
VARIANZA	0.40	0.01	0.01	16.61
DESV. EST	0.63	0.10	0.10	4.08

17-11-2010 18-11-2010 19-11-2010

Listo 100%

Figura 5.30 Apariencia de la hoja Excel con los datos guardados y las estadísticas calculadas

Como se puede ver de la Figura 5.30, después de la última medición del día, viene un cuadro con los valores del: máximo, promedio, mínimo, varianza y desviación estándar de las mediciones realizadas en ese día. Estas estadísticas se calculan por medio de formulas programadas desde la aplicación Java, por lo tanto, si se van agregando más datos, la formula se actualiza automáticamente. Debido a que Excel calcula los valores de las formulas utilizadas cuando el archivo es abierto y pone el resultado en las casillas correspondientes, entonces cuando uno desea cerrar el archivo, Excel le indica si uno desea guardar los cambios realizados. En realidad la respuesta a esta pregunta es indiferente, ya que cuando se vuelve a abrir el archivo no se percibe cambio alguno.

Otro punto importante es que cuando se da un cambio de fecha, entonces la aplicación en Java crea una nueva hoja en la cual se almacenan las mediciones correspondientes a esa fecha. Esta situación se ve ilustrada en el recuadro rojo de la Figura 5.30.

Ahora bien, si el usuario quisiera visualizar esos datos guardados en el archivo Excel sin necesidad de abrir dicho archivo, entonces se selecciona la opción **"Ver Datos Guardados"**. Si se selecciona esta opción sin haber creado el archivo Excel anteriormente (no se ha recibido ni un solo dato por parte del PIC32MX), la interfaz se lo indica al usuario.

Figura 5.31 Mensaje indicando que el archivo Excel no ha sido creado o está dañado

El mensaje le indica que el archivo no ha sido creado o que puede ser que este dañado. El archivo se daña cuando la aplicación trata de escribir en él cuando éste está abierto, por lo tanto es importante recordar tener el archivo cerrado cuando se quiera recibir todos los datos.

Cuando el archivo esta creado y sin daños, los datos guardados en él se muestran en la pantalla "Ver Datos Guardados", como se muestra en la Figura 5.32.

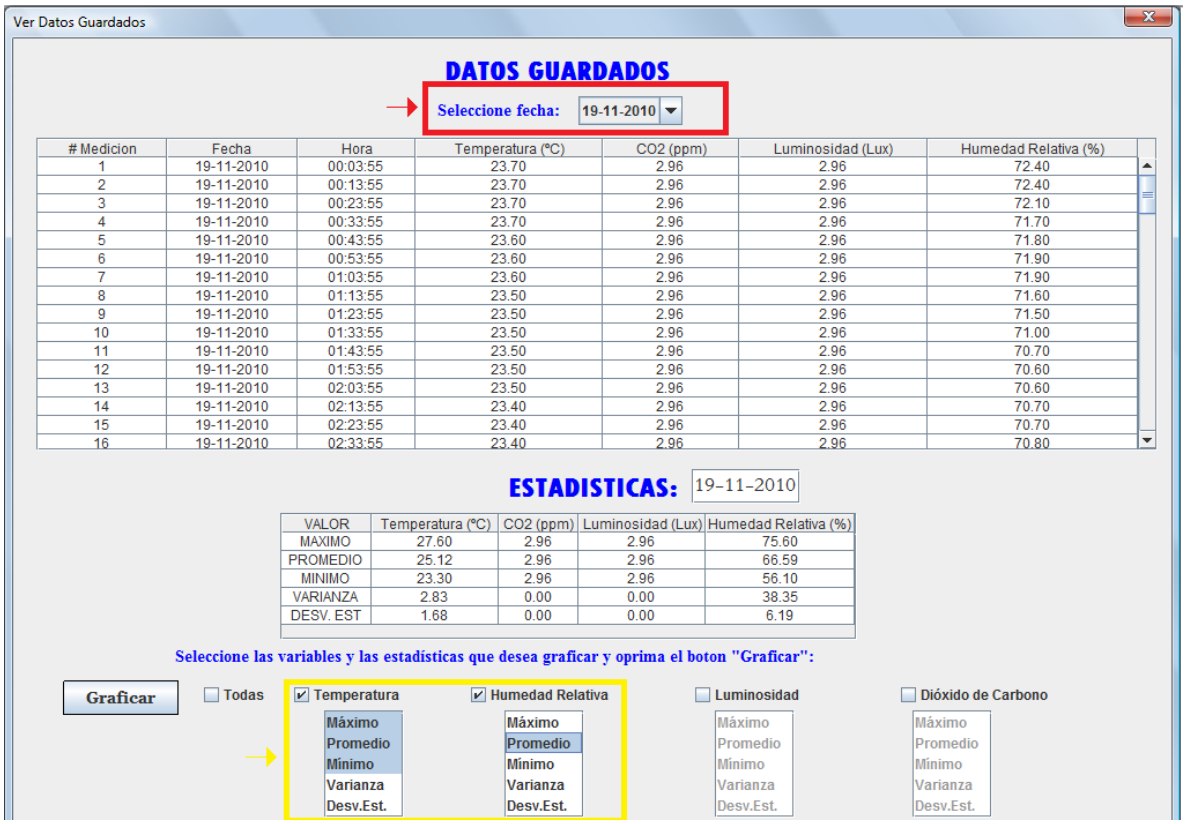

Figura 5.32 Pantalla donde se visualizan las mediciones guardadas

En esta última figura vemos como se muestran las mediciones guardadas del día seleccionado (recuadro rojo) en una tabla y las estadísticas de ese día en otra tabla separada. Además, el usuario puede ver todas las posibles graficas que puede realizar, ya sea del día en cuestión o de las estadísticas seleccionadas para las variables seleccionadas (recuadro amarillo).

Ahora bien, si se oprime el botón “Graficar” en la pantalla anterior con la selección que se indica en la Figura 5.32 se obtiene una ventana que posee un panel con todas las graficas seleccionadas, como se muestra en la Figura 5.33.

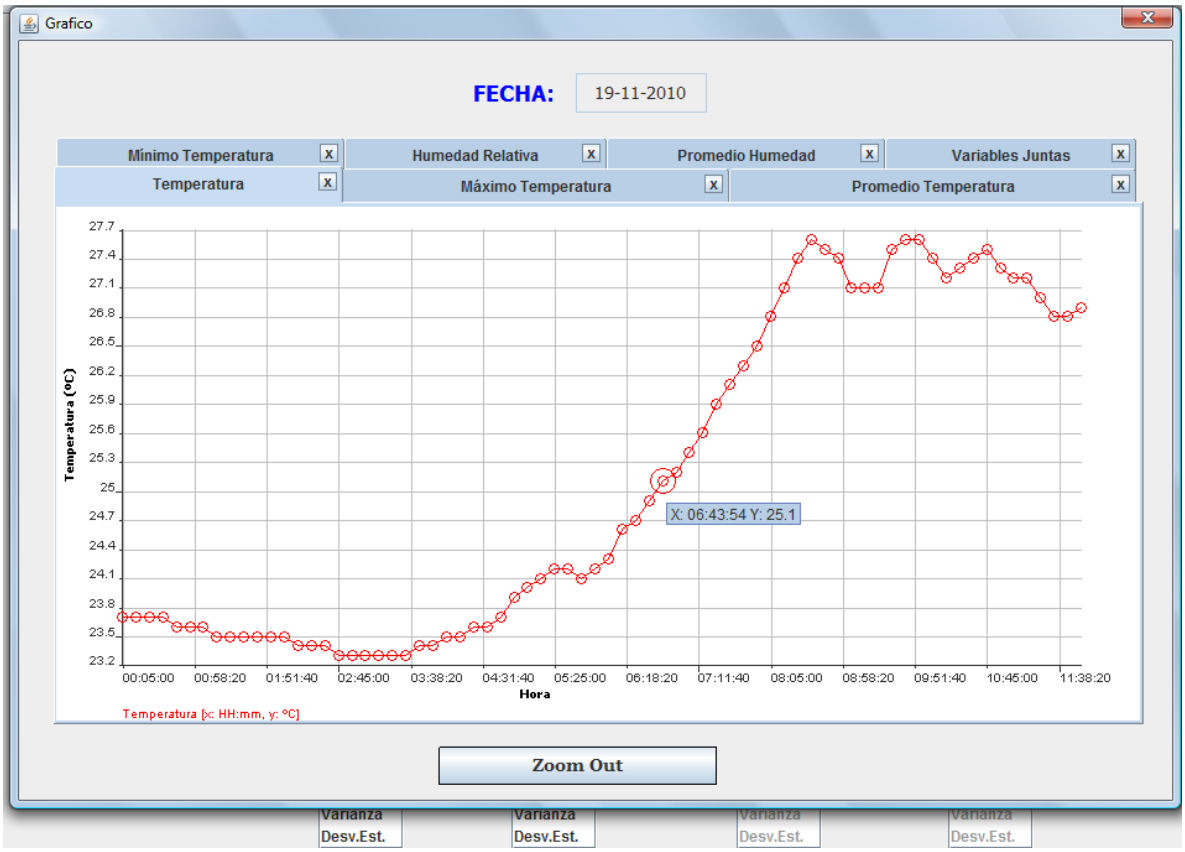

Figura 5.33 Pantalla con las graficas seleccionadas

El panel posee todas las graficas que se seleccionaron previamente, y las ordena por medio de pestañas en niveles. Cada una de las pestañas se puede cerrar individualmente. Además, al situar el ratón cerca de uno de los datos de la curva que se está visualizando, se despliega un recuadro con la información de dicho punto (valor del eje “x” y valor del “y”).

Además se puede hacer zoom sobre un área en especial, al arrastrar el ratón por el gráfico, como muestra la Figura 5.34.

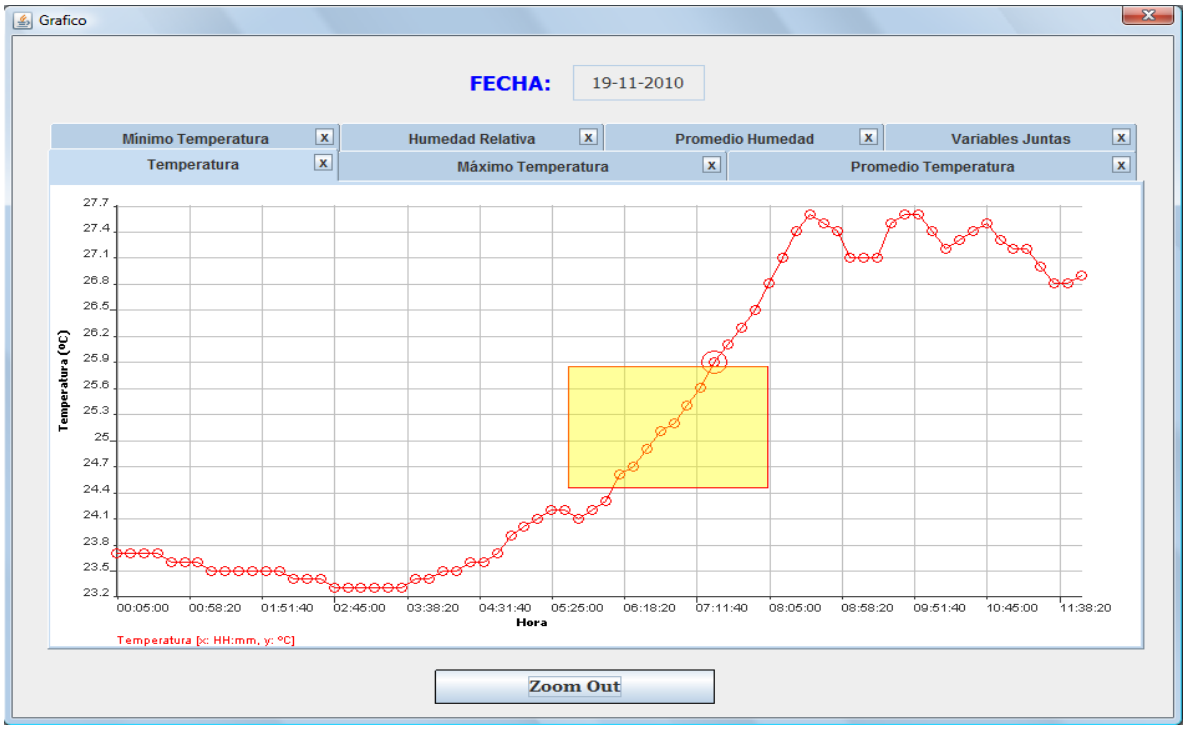

Figura 5.34 Pantalla mostrando el área a la cual se desea hacer zoom

La gráfica con zoom se muestra en la Figura 5.35.

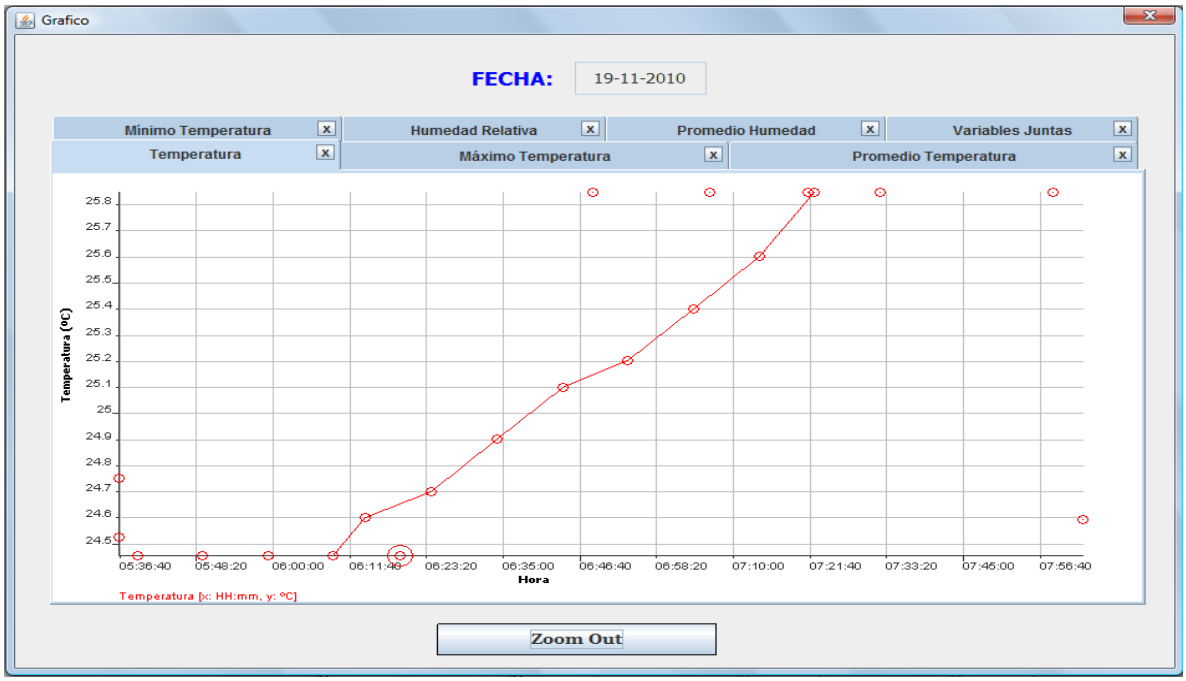

Figura 5.35 Gráfica con zoom

Al hacer zoom, tanto el eje “Y” como el eje “X” se redimensionan al recuadro seleccionado.

Si lo que se desea graficar es alguna estadística, entonces el eje “X” se dimensiona con todas las fechas que se hayan guardado en el archivo Excel, mientras el eje “Y” contiene los valores de la estadística para esas fechas, como muestra la Figura 5.36.

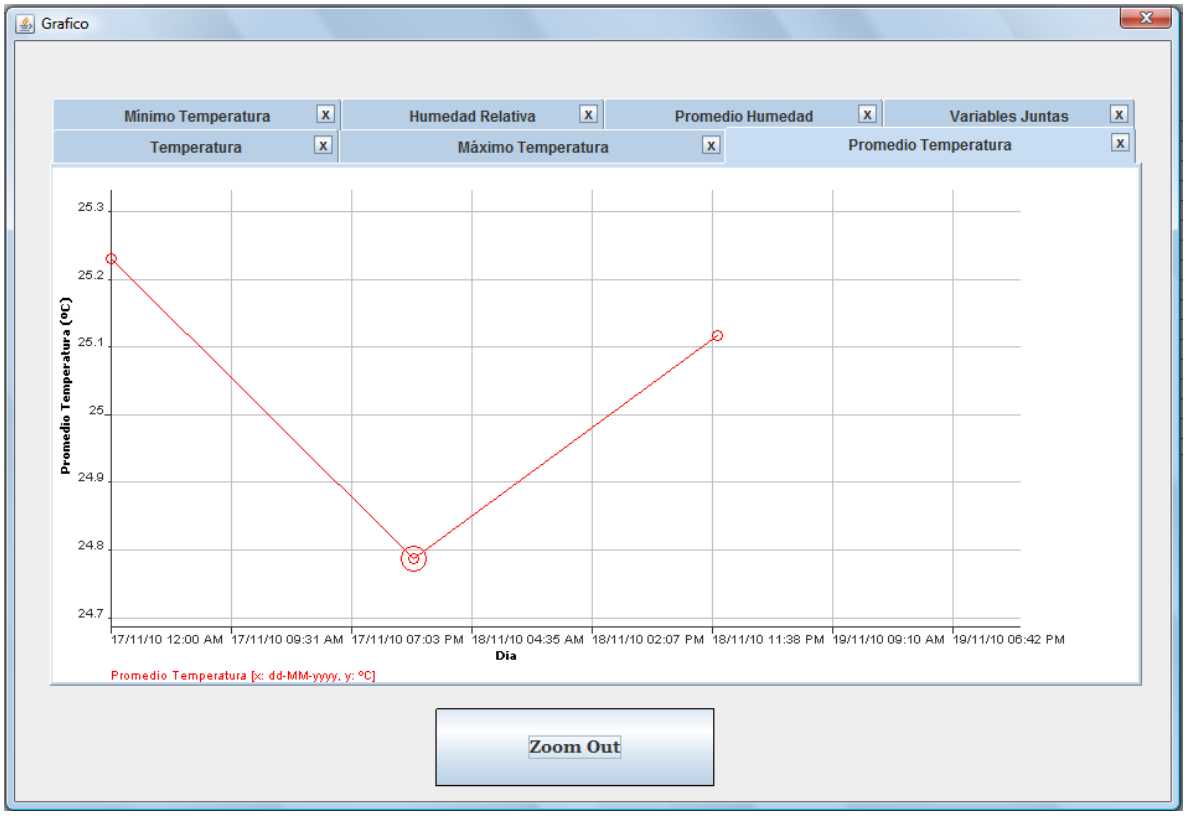

Figura 5.36 Gráfico de un valor estadístico por fecha

De nuevo, se puede hacer zoom sobre el área que se desee.

Una vez que el usuario observó las graficas y cierre la ventana, entonces se vuelve a visualizar los datos guardados, si esta pantalla se cierra se retorna a la pantalla de inicio. Se ha explicado entonces las diferentes funciones de la interfaz.

Capítulo 6: Análisis de Resultados

6.1 Resultados

6.1.1 Prueba del comportamiento del sistema embebido

Con el fin de evaluar cómo se comportó el sistema embebido, se procedió a medir la exactitud en el tiempo en el que se generan las interrupciones del contador de 32 bits donde se programó el intervalo el cual el usuario desea que se tomase una medición. El intervalo se programó para que se diera la interrupción cada minuto durante un periodo de 10 minutos.

Además, se aplicó una tensión fija de 1.649 V a las entradas analógicas AN1 y AN2, con el fin de evaluar la exactitud de la toma de datos. En la Tabla 6.1 se muestran las mediciones y el tiempo obtenidos.

Tabla 6.1 Datos de la prueba del sistema embebido

Numero de Medición	Hora Sistema Embebido	Dato AN1 (V)	Dato AN2 (V)
1	17:56:00	1.648	1.648
2	17:57:00	1.648	1.648
3	17:58:00	1.648	1.648
4	17:59:00	1.648	1.648
5	18:00:00	1.648	1.648
6	18:01:00	1.648	1.649
7	18:02:00	1.648	1.648
8	18:03:00	1.648	1.648
9	18:04:00	1.648	1.648
10	18:05:00	1.648	1.648
PROMEDIO (V)		1.648	1.6481

6.1.2 Prueba del manejo de archivos y gráficos de la interfaz gráfica de usuario

Como prueba para la interfaz, se tomó datos de todo un día, para verificar el funcionamiento del sistema y a la vez del manejo de archivos y gráficos de la interfaz grafica de usuario. El archivo DatosSensores.xls fue creado exitosamente y recopiló los datos del 20 de noviembre del 2010, obtenidos a través del sistema embebido.

En las Figura 6.1 se muestran las curvas de temperatura y humedad relativa graficadas en Microsoft Excel. En la Figura 6.2 se muestran dichas curvas graficadas a través de la interfaz grafica de usuario desarrollada.

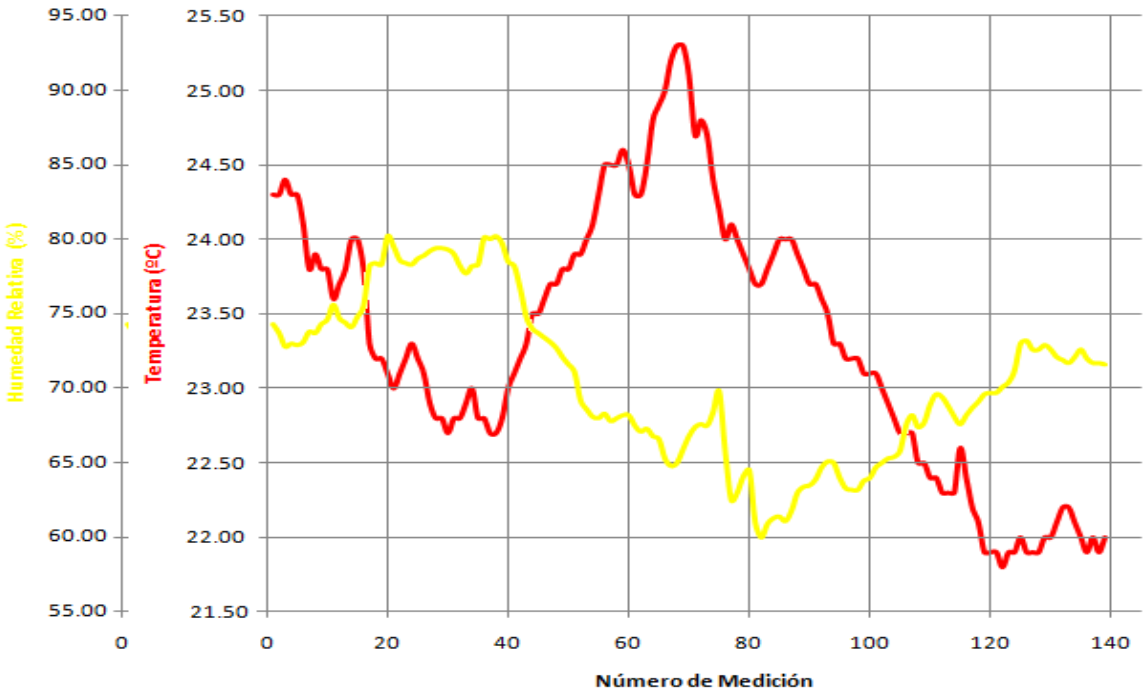

Figura 6.1 Temperatura y humedad del 20/11/10 graficadas con Microsoft Excel

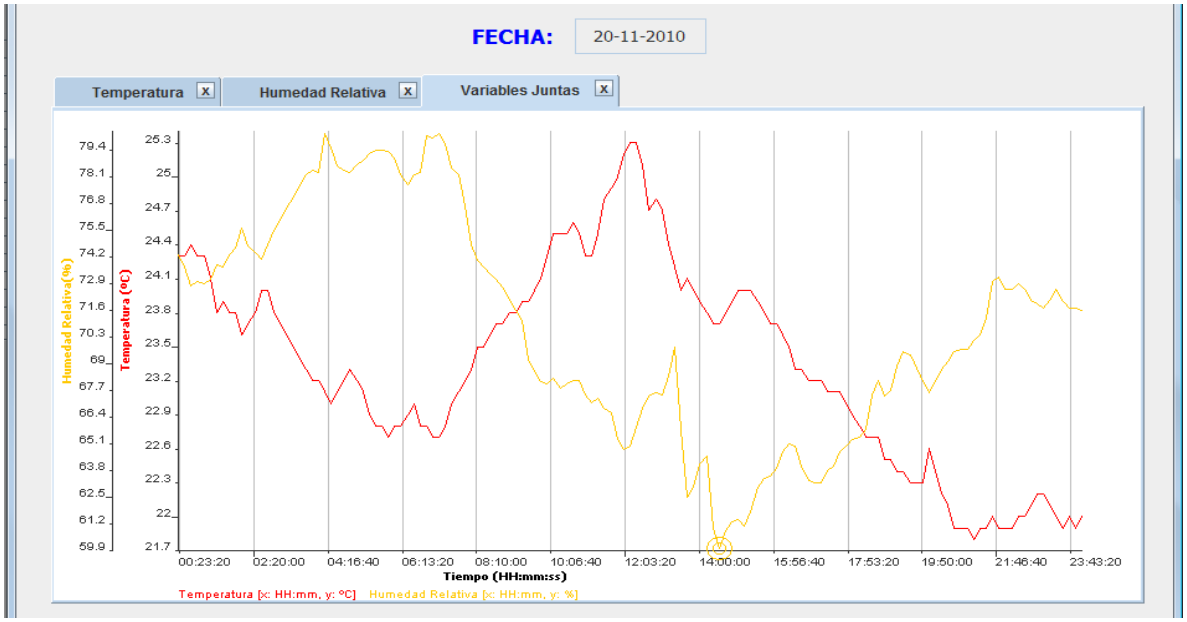

Figura 6.2 Temperatura y humedad del 20/11/10 graficadas con la interfaz gráfica de usuario

6.2 Análisis

6.2.1 Análisis de la prueba del comportamiento del sistema embebido

La tensión fija que se utilizo como referencia conectada a las entradas analógicas AN1 y AN2 del PIC32MX fue de 1.649 V. Como se puede observar en la Tabla 6.1, los valores obtenidos por el ADC no se alejan más de 1 mV de la tensión de referencia. Sin embargo, se observa que la mayoría de las conversiones tuvieron un valor de 1.648 V. Esto por el intervalo de cuantificación (Q) del ADC, que se define como

$$Q = \frac{V_{ADC}}{2^N}$$

Donde V_{ADC} es el margen de tensiones de entrada, y N es el número de bits de resolución del ADC. Para el PIC32MX, VADC=3.3 V y N=10, por lo tanto

$$Q_{PIC32} = \frac{3.3}{2^{10}} \approx 3.223\ mV$$

Esto significa que se necesita un cambio de 3.223 mV en la entrada para producir un cambio en la salida de 1 bit en la salida del ADC. Por lo tanto es sumamente difícil obtener el valor exacto de la tensión de referencia.

Para convertir el dato entregado por el ADC, el PIC32MX MCU debe multiplicar este dato por el intervalo de cuantificación. Si lo hacemos a la inversa, para obtener a que valor digital corresponde 1.649 V se tiene que:

$$ADC = V_{REF} * \frac{2^N - 1}{V_{ADC}} = 1.649\ V * \frac{2^{10} - 1}{3.3\ V} \approx 511.19$$

El ADC entregaría a su salida un valor de 511, el cual corresponde a una tensión de

$$V_{OUT\_ADC} = ADC * \frac{V_{ADC}}{2^N - 1} = 511 * \frac{3.3\ V}{2^{10} - 1} = 1.648387$$

Este dato al redondearse, corresponde a 1.648 V. Aun al redondearse, el porcentaje de error de la conversión es de apenas 0.06%.

Con lo cual se cumple con lo establecido de un uno de los objetivos específicos planteados: los datos guardados en el microcontrolador no se alejan más de un 5% de los valores sensados.

Ahora bien, en la Tabla 6.1 también se observa que las horas en que se guardan las mediciones poseen una diferencia de exactamente un minuto entre mediciones consecutivas. Esto indica que la interrupción del contador donde se programó el intervalo entre mediciones se generó con la exactitud deseada.

6.2.2 Análisis de la prueba del manejo de archivos y gráficas de la interfaz gráfica de usuario

El archivo Excel DatosSensores.xls se creó efectivamente en la carpeta deseada (C:\Red Sensores). En este archivo se encuentran almacenadas todas las mediciones del día 20 de noviembre del 2010. La similitud entre la grafica producida por Excel y la producida por la interfaz es visible, sin embargo, la grafica de la interfaz tiene la ventaja de que el rango de sus ejes se autoajusta a los valores máximos y mínimos de las curvas. En Excel es necesario redefinir el formato de los ejes para obtener una mejor visualización de las curvas.

Además, la interfaz le permite al usuario: hacer zoom sobre un área de interés en especial, retornar al zoom original, seleccionar los datos a graficar más fácilmente, graficar varias curvas con diferentes ejes verticales.

6.2.3 Análisis del comportamiento del módulo RTCC en un período extenso de funcionamiento

En la Figura 6.3 se observan los datos recolectados del día 19 de noviembre del 2010. El intervalo programado para la toma de mediciones era de 10 minutos. El módulo RTCC del PIC32MX funcionó correctamente las primeras 6 horas de mediciones, sin embargo al cumplirse la sexta hora de funcionamiento (medición numero 37) la hora obtenida fue un segundo menor a la hora que se esperaba. Este comportamiento se notó cada 6 horas, donde la hora disminuía un segundo. Esto puede deberse a que el cristal de cuarzo no está calibrado correctamente.

Tabla 6.2 Datos guardados del 19/11/2010

# MEDICION	FECHA (dd-mm-aaaa)	HORA (hh:mm:ss)	TEMPERATURA (ºC)	HUMEDAD RELATIVA (%)
1	19-11-2010	15:10:00	26.90	76.00
2	19-11-2010	15:20:00	27.10	75.90
3	19-11-2010	15:30:00	27.10	76.40
4	19-11-2010	15:40:00	27.00	76.90
5	19-11-2010	15:50:00	26.90	76.70
6	19-11-2010	16:00:00	26.90	78.70
7	19-11-2010	16:10:00	26.80	80.60
8	19-11-2010	16:20:00	26.20	85.70
9	19-11-2010	16:30:00	26.60	80.90
10	19-11-2010	16:40:00	26.80	77.30
11	19-11-2010	16:50:00	26.80	78.20
12	19-11-2010	17:00:00	26.70	78.50
13	19-11-2010	17:10:00	26.30	78.60
14	19-11-2010	17:20:00	26.30	78.30
15	19-11-2010	17:30:00	26.20	81.50
16	19-11-2010	17:40:00	26.30	81.50
17	19-11-2010	17:50:00	26.40	80.00
18	19-11-2010	18:00:00	26.40	79.00
19	19-11-2010	18:10:00	23.80	71.50
20	19-11-2010	18:20:00	21.80	65.50
21	19-11-2010	18:30:00	26.30	77.80
22	19-11-2010	18:40:00	26.30	77.80
23	19-11-2010	18:50:00	26.40	78.40
24	19-11-2010	19:00:00	26.40	78.60
25	19-11-2010	19:10:00	26.50	78.90
26	19-11-2010	19:20:00	26.30	79.10
27	19-11-2010	19:30:00	26.30	78.90
28	19-11-2010	19:40:00	26.00	78.50
29	19-11-2010	19:50:00	25.80	77.50
30	19-11-2010	20:00:00	25.80	75.70
31	19-11-2010	20:10:00	25.80	75.40
32	19-11-2010	20:20:00	25.80	77.70
33	19-11-2010	20:30:00	25.80	78.40
34	19-11-2010	20:40:00	25.80	79.40
35	19-11-2010	20:50:00	25.70	79.90
36	19-11-2010	21:00:00	25.70	80.30
37	19-11-2010	21:09:59	25.70	80.70
38	19-11-2010	21:19:59	25.60	81.50
39	19-11-2010	21:29:59	25.60	81.50

6.2.4 Análisis del costo del sistema desarrollado

El costo total aproximado del sistema desarrollado en este proyecto ronda alrededor de los $232, según la Tabla 6.3.

Tabla 6.3 Desglose del costo total del sistema

Cantidad	Descripción	Precio Unitario ($)	Fuente
1	Paquetes de Desarrollo PIC32 (DM320003-2)	55	www.microchip.com
1	Tarjeta de expansión de E/S para PIC32 (DM320002)	72	
1	Sensor LM35DT	5	www.microtroniks.com
1	Sensor HIH-4000-001	50	
	Otros	50	
	Total	**232**	

Este es un monto razonable comparado con sistemas comerciales, que aunque son más robustos y poseen más funciones, su costo los hace inaccesibles a pequeños productores. Además, al ser productos exclusivos y sofisticados, requieren un mantenimiento periódico cuyo costo también es elevado.

Capítulo 7: Conclusiones y recomendaciones

7.1 Conclusiones

- La ecuación de salida del sensor de temperatura LM35DT fue validada utilizando el termómetro calibrado de resistencia de platino estándar y bajo condiciones controladas.

- Al acondicionar las señales provenientes de los sensores se aprovecha la máxima resolución del ADC obteniendo así mediciones más precisas.

- El desarrollo de un sistema embebido es una solución viable económica y funcionalmente para la obtención de información valiosa para la agricultura.

- Los datos transmitidos desde el sistema embebido y recibidos por la computadora por medio del puerto USB coinciden en un 100%.

- El intervalo de cuantificación del ADC del PIC32MX provoca las pequeñas diferencias entre los valores sensados y los almacenados en memoria.

7.2 Recomendaciones

- Buscar o construir un lugar con ambiente controlado en el cual hacer pruebas exhaustivas con los sensores.

- Conseguir instrumentación calibrada para probar los diferentes tipos de sensores que se poseen.

- Encapsular el sensor HIH-4000-001 para que no se vea afectado por luminosidad excesiva.

- Trabajar siempre con las versiones más recientes tanto del MPLAB como del NetBeans.

- Calibrar el cristal de cuarzo para evitar errores en el módulo de reloj y calendario de tiempo real.

- Utilizar tensiones de referencia en el ADC según la naturaleza del nivel a convertir para obtener un error menor en la conversión.

Bibliografía

[1] Universidad de Colima y Universidad de Valencia.

D2ARS [en línea]. 2007

Disponible en: http://www.d2ars.org

[2] Universidad de Colima, Universidad Politécnica de Valencia, Universidad Autónoma de Occidente, Instituto Tecnológico de Costa Rica, Centro de Investigación Científica y de Educación Superior de Ensenada.

Proyecto D2ARS, Entregable D1.1: *"Especificación del Sistema"*.

D2ARS [en línea]. 2007 [visitado el 06 de agosto de 2009].

Disponible en: http://www.d2ars.org/d2ars/system/files/D2ARS_FORMATO_v1.pdf

[3] Pallàs, Ramón; Casas, Óscar y Bragós, Ramón, **Sensores y acondicionadores de señal. Problemas resueltos**, 1era Edición, México: Alfaomega Grupo Editor S.A, México, Enero 2009.

[4] National Semiconductor, **LM35 Precision Centigrade Temperature Sensors Datasheet**. November 2000.

[5] Honeywell Inc., **HIH-4000 Series Humidity Sensors Datasheet**.

[6] Microchip Technology Inc., **PIC32MX5XX/6XX/7XX Data Sheet – 64/100-Pin General Purpose and USB Flash Microcontrollers** (DS61143F), Revision F, June 2009.

[7] Microchip Technology Inc., **PIC32 Family Reference Manual – Section 2 MCU** (DS61113C), Revision C, May 2008.

[8] Microchip Technology Inc., **PIC32 Family Reference Manual – Section 6 Oscillators** (DS61112E), Revision E, July 2008.

[9] Microchip Technology Inc., **PIC32 Family Reference Manual – Section 29 Real-Time Clock and Calendar** (DS611125D), Revision D, June 2008.

[10] Microchip Technology Inc., **PIC32 Family Reference Manual – Section 17 10-Bit AD Converter** (DS61104D), Revision D, June 2008.

[11] Microchip Technology Inc., **PIC32 Family Reference Manual – Section 14 Timers** (DS61105D), Revision D, May 2008.

[12] Microchip Technology Inc., **PIC32 Family Reference Manual – Section 27 USB On-The-Go** (DS61126E), Revision E, August 2009.

[13] Oñativia, Gerónimo Isidro. **jPicUSB: Clase Java para comunicación USB con PICs usando API de Microchip.** Universidad Nacional de Tucumán. Facultad de Ciencias Exactas. Ingeniería en Computación.

Disponible en: http://www.edutecne.utn.edu.ar/microcontrol_congr/comunicaciones/JPICUS.PDF

[14] Página Web: **Java Excel API- A Java API to read, write, and modify Excel spreadsheets.**
Disponible en: http://jexcelapi.sourceforge.net/

[15] Página Web: **JChart2D**
Disponible en: http://jchart2d.sourceforge.net/index.shtml

[16] Wikipedia. **Universal Serial Bus**. Última modificación 27 noviembre 2010. Última visita: 28 noviembre 2010.
Disponible en: http://es.wikipedia.org/wiki/Universal_Serial_Bus

[17] Wikipedia. **Java (lenguaje de programación)**. Última modificación 20 noviembre 2010. Última visita: 28 noviembre 2010.
Disponible en: http://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

[18] Wikipedia. **Sistema embebido**. Última modificación 22 noviembre 2010. Última visita: 28 noviembre 2010.
Disponible en: http://es.wikipedia.org/wiki/Sistema_embebido

Apéndices

Apéndice A.1 Glosario y Abreviaturas

ADC: Analog to Digital Converter. En español es convertidor analógico digital y es un dispositivo electrónico que tiene la capacidad de convertir una señal analógica en una digital.

A.O: Amplificador Operacional. Es un dispositivo electrónico que generalmente se presenta como un circuito integrado. Desarrollado en los años 1960, es un dispositivo muy común y muy utilizado por su versatilidad.

API: Application Programming Interface. En español se traduce como "Interfaz de Programación de Aplicaciones". Se define como el conjunto de procedimiento y funciones que ofrece una biblioteca para ser utilizado por otro software.

IDE: Integrated Development Environment. Traducido al español significa Entorno integrado de desarrollo. Es un programa informático compuesto por un conjunto de herramientas de programación

DLL: Dynamic Link Library. Es una biblioteca de enlace dinámico. Son archivos con código ejecutable que corren bajo demanda de un programa por parte del sistema operativo.

JAR: Java Archive. Es un archivo ejecutable escrito en lenguaje de programación Java.

JNI: Java Native Interface. Es un módulo de software que permite que un programa escrito en lenguaje Java pueda interactuar con programas escritos en otros lenguajes de programación.

MCU: Microcontroller Unit. Se refiere a la unidad central de procesamiento de un microcontrolador.

RTCC: Real-Time Clock and Calendar. Módulo para tener conocimiento del tiempo actual de forma precisa en una aplicación o sistema.

SHA: Sample and Hold Amplifier. Es un dispositivo analógico que captura el voltaje de una señal que varia continuamente y mantiene su valor en un nivel constante durante un período mínimo determinado.

SPRT: Standard Platinum Resistance Thermometer. Termómetro que utiliza la variación de una resistencia de platino con el calor para medir la temperatura.

USB: Universal Serial Bus. Especificación para establecer comunicación serial entre dispositivos y un controlador anfitrión.

Apéndice A.2 Fotografía del prototipo del sistema

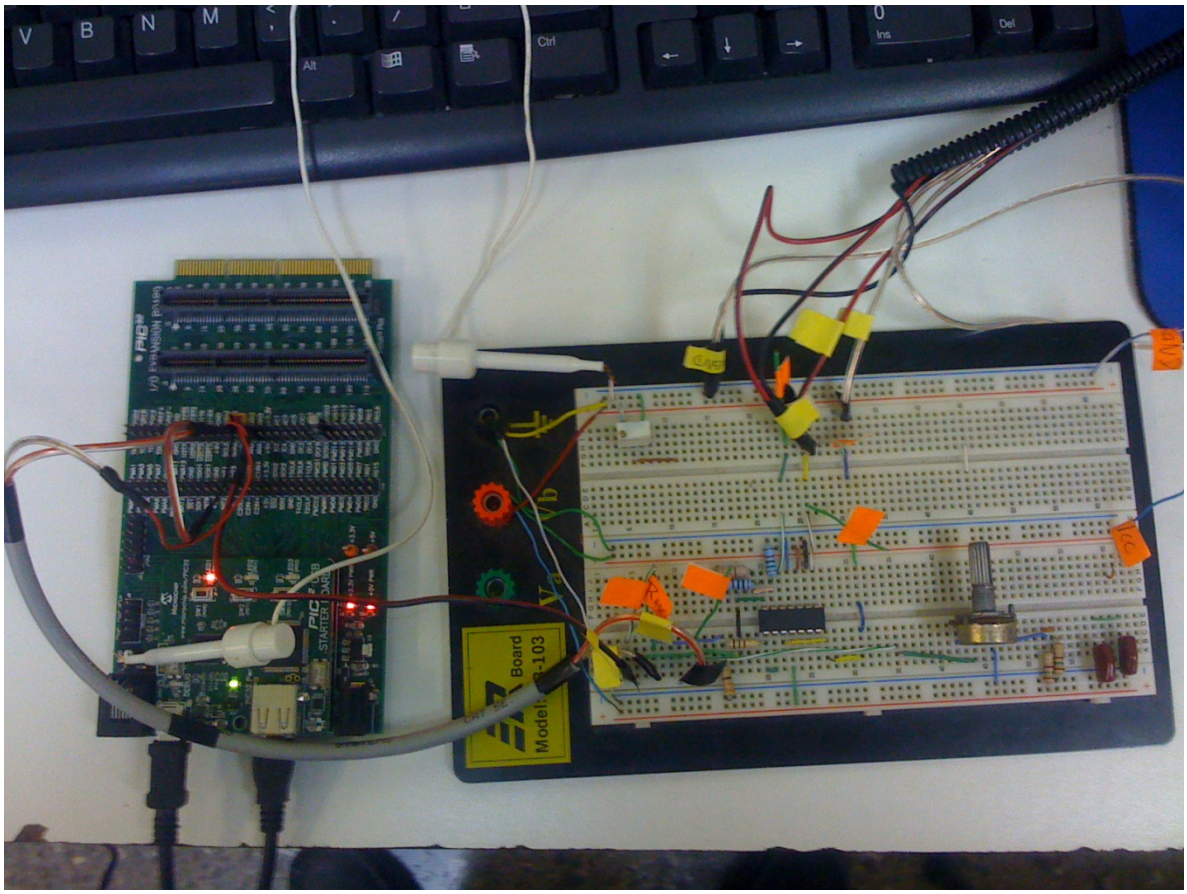

Figura A.1 Fotografía del prototipo del sistema

Apéndice A.3 Hoja de información del proyecto

Información del estudiante:

Nombre: José Pablo Vernavá Amador

Cédula: 1-1237-0248 **Carné ITCR:** 200307876

Dirección de su residencia en época lectiva: San Rafael, Escazú. Urb.Trejos Montealegre

Dirección de su residencia en época no lectiva: San Rafael, Escazú. Urb.Trejos Montealegre

Teléfono en época lectiva: 88216238 **Teléfono época no lectiva:** 88216238

Email: josepaver@hotmail.com **Fax:** 22895419

Información del estudiante:

Nombre: Carlos Marco García Ramírez

Cédula: 1-1006-0263 **Carné ITCR:** 9518915

Dirección de su residencia en época lectiva: San Juan de Tibas, De Antojitos 300m Oeste, Condominio Casa Mia Casa #15

Dirección de su residencia en época no lectiva: San Juan de Tibas, De Antojitos 300m Oeste, Condominio Casa Mia Casa #15

Teléfono en época lectiva: 83559758/22362439

Teléfono época no lectiva: 83559758/22362439

Email: siqui@hotmail.com **Fax:** 22407175

Información del proyecto:

Nombre del Proyecto: Interfaz de adquisición de datos de una red de sensores por medio de un sistema embebido basado en microcontrolador PIC32 con comunicación USB

Área del Proyecto: comunicaciones, microcontroladores, programación en lenguajes de alto nivel

Información de la empresa:

Nombre: Escuela de Ingeniería Electrónica del ITCR

Zona: Cartago

Información del encargado en la empresa:

Nombre: Ing. Marvin Hernández Cisneros

Puesto que ocupa: Profesor de la Escuela de Ingeniería Electrónica del ITCR

Departamento: Escuela de Ingeniería Electrónica

Profesión: Ingeniero **Grado académico:** Licenciatura en Ing. Electrónica

Teléfono: 25521842 **Email:** marhernandez@itcr.ac.c

Anexos

Anexo B.1 Certificado de calibración del SPRT 5698 por parte de la compañía Fluke

Certification

Certification of Standard Platinum Resistance Thermometer

Model 5698
Serial No. 95134

Hart Scientific certifies that the standard platinum resistance thermometer above was produced from high purity platinum wire with a stress-free design sensor. The standard platinum resistance thermometer was tested in Hart Laboratory. The thermometer is satisfactory as an interpolation instrument of the ITS-90 in accordance with the criteria that W(302.9146 K) ≥ 1.11807. The stability of the thermometer at the triple point of water satisfies the specification after 100 hours operation at 660 °C. The resistance at the triple point of water was 25.51527 ohms at zero-power dissipation at the end of the stability test.

Performed by: Matt Newman

Approved by: Mingjian Zhao

Date: May 29, 2007

Fluke Corporation
Hart Scientific Division
799 E Utah Valley Drive
American Fork, Utah 84003-9775

Anexo B.2 Certificado de calibración del SPRT 5698 por parte de Lacomet

LACOMET Laboratorio Costarricense de Metrología

LACOMET 08190208

CERTIFICADO DE CALIBRACIÓN
LACOMET 08190208

Fecha de Calibración:	2008-05-12
Objeto a Calibrar:	Termómetro de resistencia de platino, tipo SPTR 25[(2)], marca FLUKE, modelo 5698-25
Serie/Identificación:	95134
Número de Solicitud:	PS 0337- 08
Solicitante:	Instituto Tecnológico Costa Rica, Escuela de Física
Dirección del solicitante:	Cartago, Costa Rica
Referencia de Datos:	ASM-TE-03, Folios 58 a 63
Lugar de la Calibración:	Laboratorio de Temperatura, LACOMET

Lic. Roger Irías Campos
Departamento de Metrología Física

Adrián Solano Mena
Responsable de la Calibración

Laboratorio Costarricense de Metrología. Tels.: (506) 2283 6580, (506) 2280 5387, Fax: (506) 2283 5133
Ciudad de la Investigación, Apdo. 1736-2050, San Pedro de Montes de Oca, Costa Rica

Información de los Patrones Utilizados

Descripción	Serie/Identificación	Trazabilidad
Celda de punto fijo de H_2O, marca Isotech, modelo B50	422	NTPL 251139 Inglaterra
Celda de punto fijo de Sn, marca Isotech, modelo ITL-M17669	Sn88	NAMAS-Nº 175 / 99-03-64 Inglaterra
Celda de punto fijo de Zn, marca Isotech, modelo ITL-M17671	Zn136	NAMAS-Nº 175 / 99-03-65
Resistor de 25 Ω, marca, Tinsley, modelo 5685A	274958	NRC ES-20070030-03, Canadá

Otros Equipos

Puente de Resistencia, Marca ISOTECH, modelo, T.T.I.3	990108

Resultados de la Calibración (1)

Temperatura Nominal °C	Celda patrón de punto fijo de:	Resistencia del instrumento sujeto a calibración para una corriente de 1 mA (Ω)	Incertidumbre expandida (±) mΩ
0,010	H_2O	25,515 44	0,10
231,928	Sn	48,293 07	0,14
419,527	Zn	65,542 16	0,24

Laboratorio Costarricense de Metrología. Tels.: (506) 2283 6580, (506) 2280 5387, Fax: (506) 2283 5133
Ciudad de la Investigación, Apdo. 1736–2050, San Pedro de Montes de Oca, Costa Rica

Coeficientes ITS90 para el ámbito de 0 °C a 420 °C			
	Corriente de medición de 0 mA		Corriente de medición de1 mA
R (Ω)	25,515 31		25,515 44
a_8 = A	-9,198 037 2 x10^{-5}		-9,317 389 3 x10^{-5}
b_8 = b	-1,729 599 9 x10^{-5}		-1,868 248 8 x10^{-5}

Observaciones

- La incertidumbre expandida reportada se obtuvo multiplicando la incertidumbre estándar combinada por un factor de cobertura con el que se alcanza un nivel de confianza de al menos 95 %. La incertidumbre estándar de la medición se determinó conforme a la *"Guide to Expression of Uncertainty in Measurement, BIPM-IEC-IFCC-ISO-IUPAC-IUPAP-OIML"*, en la cual se toma en cuenta la incertidumbre de los patrones, del método de calibración, de las condiciones durante la calibración y del equipo sujeto a calibración.
- El factor de cobertura es de k = 2, para un nivel de confianza de un 95 %
- Este Certificado de Calibración sólo ampara las mediciones reportadas en el momento y en las condiciones ambientales y de uso en que se realiza la calibración.
- Los resultados emitidos en este certificado se refieren únicamente al objeto calibrado y a las magnitudes especificadas.
- [2] W(302,9146 K) = 1,118 13 lo que satisface la condición W(302,9146 K) ≥ 1,118 07 para SPTR
- Condiciones Ambientales:

  Temperatura: (20,9 ± 0,5) °C
  Humedad relativa: (52 ± 5) %

[1] **Método de calibración:** Por comparación, la lectura del patrón con la del equipo sujeto a calibración , utilizando celdas del punto fijo como medio de comparación, acorde al procedimiento MF-TE-PR-06

--- Última línea ---

Laboratorio Costarricense de Metrología. Tels.: (506) 2283 6580, (506) 2280 5387, Fax: (506) 2283 5133
Ciudad de la Investigación, Apdo. 1736–2050, San Pedro de Montes de Oca, Costa Rica