Universidad Nacional Autónoma de México

Facultad de Ingeniería

Diseño e implementación de un datalogger para aplicaciones de agrometeorología

T E S I S

que para obtener el título de:
Ingeniero Eléctrico Electrónico

presenta:
anuar lezama barquet

director de tesis:
m.i. lauro santiago cruz

México,D.F. 2010

We shall not cease from exploration
And the end of all our exploring
Will be to arrive where we started
And know the place for the first time.

— T.S. Eliot/*Little Gidding.*

Para mi mamá y mi papá, Layla Barquet y Emiliano Lezama,
que me han querido y apoyado incondicionalmente.

Agradecimientos

A la Universidad Nacional Autónoma de México por contribuir a mi formación profesional y haber sido el sitio donde conocí a mis mejores profesores y a mis mejores amigos.

Al Instituto de Ingeniería y a mi asesor, el M.I. Lauro Santiago Cruz, por permitirme desarrollar esta tesis.

A mis sinodales, por haber revisado este trabajo y hecho valiosas observaciones para mejorarlo.

A mis profesores por compartir sus conocimientos y su valiosa experiencia conmigo y con todos mis compañeros. En especial quiero agradecer al Ing. Jóse Salvador Zamora, al M.I. Lauro Santiago, al M.I. Larry H. Escobar, a la Ing. Gloria Mata, al Dr. José Cohen, al Ing. Roberto Macias, al físico Sergio Arzamendi, a la Ing. Beatriz Eslava y al Ing. Juan Manuel Rojas, porque fueron los profesores que considero más me aportaron durante la carrera. Gracias.

A mis hermanos Estefanía y Mijail.

Un especial agradecimiento a Ana Angélica por su gran apoyo, amistad y cariño.

A todos mis amigos.

A mis compañeros de laboratorio: Daniel, Humberto, Juan, Mario, David, Martín, Belit, Aleidy, Israel, Jorge, William, Kayu, Cuauhtémoc. Gracias por sus consejos, apoyo y amistad.

Al Sr. Spock.

Gracias a todos

Anuar Lezama Barquet

Contenido

Índice de Figuras

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Índice de Tablas

Capítulo 2

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Prefacio

En el presente trabajo se describe el diseño y la implementación de un sistema de instrumentación que tiene como **objetivo** medir variables ambientales de utilidad en la agricultura.

Este sistema contempla las distintas etapas de instrumentación que van desde la lectura de los sensores hasta el despliegue de los datos adquiridos, ésto último en un formato adecuado para un posterior análisis y procesamiento. El sistema fue diseñado para ser utilizado, de ser posible, por productores agrícolas nacionales como una alternativa más económica a los sistemas de instrumentación comerciales. Con el uso de este sistema, ellos serán capaces de adquirir información ambiental de sus áreas de cultivo que, sin duda, traerá como consecuencia un mejor uso de éstas y un aumento de la productividad de las mismas.

En el desarrollo de este proyecto se buscó obtener un sistema que cumpliera cabalmente con sus funciones, es decir buscar que el sistema a pesar de la sencillez de su función tuviera una completa afinación de detalles.

Este proyecto se inició con la idea de desarrollar un sistema de adquisición de datos para un pluviómetro, posteriormente a través de una investigación de posibles aplicaciones para estos datos, se encontró su gran utilidad en la agricultura. Dado esto y con el propósito de darle una aplicación al sistema en un área específica, se decidió crear un sistema de adquisición de datos que no sólo contemplara la precipitación como una de las variables a adquirir sino que también incluyera la temperatura ambiente. Una vez contemplado lo anterior, se buscaron características particulares que dichos sistemas debieran poseer para poder ser usado en esta aplicación. Tomando en cuenta estas características, se comenzó el desarrollo del sistema, se elaboraron prototipos y se les sometió a pruebas preliminares de funcionamiento, que sirvieron para hacer correcciones hasta llegar a lo que se espera pronto sea un sistema definitivo.

Del desarrollo de este proyecto se puede decir que se creó un sistema para la adquisición de datos de temperatura y precipitación pluvial ampliamente configurable, de fácil utilización, de amplia autonomía funcional y versatilidad en la salida de sus datos. También se tiene un sistema base al que fácilmente

podrían agregarse funcionalidades, con miras a crear un sistema de instrumentación agrícola aún más complejo.

Este trabajo de tesis está dividido en siete capítulos y comprende la investigación previa realizada para el desarrollo del sistema, antecedentes básicos necesarios para su diseño y el diseño e implementación de los mismos. Las características particulares de cada capítulo son:

En el capítulo 1 se verán los antecedentes de la instrumentación agrícola, la importancia de ésta y las variables de interés más importantes para ella, también se estudiarán las características de los sistemas de adquisición de datos y la diferencia entre estos y los *dataloggers* y se plantearemos de manera más concreta el problema que originó esta tesis y la solución propuesta para resolverlo.

En el capítulo 2 se dará una breve introducción de los antecedentes necesarios para entender el diseño y la implementación del sistema. Aun así se considera que en este capítulo no se pueden dar todos los antecedentes necesarios para poder realizar el diseño, por lo que se alienta al lector a revisar tanto los manuales y artículos usados como referencias, en caso de que los conceptos mencionados le sean demasiado ajenos.

El desarrollo del proyecto se explica en los siguientes tres capítulos. En el capítulo 3 se tratará todo la relacionado con el hardware del sistema, sin hacer mención de la interacción de los componentes. Esta parte se tratará en el capítulo 4, donde se estudiará la secuencia de pasos e interacción de los componentes de hardware para realizar las funciones del sistema. Finalmente, en el capítulo 5 se expondrá el programa que fue desarrollado para adquirir y manejar los datos registrados por el *datalogger*.

En el capítulo 6 se describirán las pruebas realizadas al *datalogger* y los resultados obtenidos del sistema ya operando en conjunto. Por último, en el capítulo 7, se darán las conclusiones de este proyecto y los aspectos que se podrían continuar desarrollando y mejorando en el futuro.

Al final de este trabajo se incluyen las referencias usadas para el desarrollo del proyecto, un glosario con los términos empleados, un apéndice que contiene la portada de las especificaciones de los componentes usados y un breve extracto del programa del microcontrolador.

Capítulo 1

Introducción

En este capítulo se expondrán los motivos que originaron el presente trabajo de tesis y se describirá de manera general la solución planteada a dicho problema. Para realizar lo anterior se comenzará con una breve introducción a la instrumentación agrícola, las estaciones meteorológicas y los distintos tipos de adquisidores de datos.

1.1. Antecedentes

Una necesidad vital de los países es generar un suministro estable y confiable de alimentos para su población; en particular cada año la población en México se incrementa en 1.1 millones de personas[1]. Con el propósito de satisfacer la creciente demanda de alimentos de esta población, la producción agrícola nacional debe ser mejorada e incrementada.

El sector agropecuario es uno de los más importantes en México, éste representa el 2.8% de las exportaciones[2] y en 2006 el 7.8% del PIB procedía de actividades agropecuarias[3]. Sin embargo, el atraso en la incorporación de nuevas tecnologías de siembra, cultivo y recolección han ocasionado un relativo estancamiento. México no presenta niveles uniformes de desarrollo agropecuario, las zonas productivas no dan abasto a la demanda interna del país, aunque favorecen las exportaciones.

Desde siempre un ingrediente esencial en la agricultura ha sido el clima; granizo, inundaciones y periodos de sequía, entre otros, tienen efectos adversos en la calidad y cantidad de las cosechas. El monitoreo y análisis constantes

[1] **INEGI**, Conteos de Población y Vivienda, 1995 y 2005.
[2] **Secretaría de Economía**, México exporta, volumen 6 No. 2 Febrero de 2007.
[3] **INEGI**, Variación anual del Producto Interno Bruto (PIB) por gran división de las actividad económica.

de las condiciones climáticas pueden ser utilizados por los productores para favorecer sus actividades y tomar acciones preventivas que minimicen el daño en su producción.

El crecimiento de los cultivos está principalmente determinado por las condiciones del suelo y del clima. El éxito o fracaso de los cultivos está íntimamente relacionado con las condiciones ambientales predominantes.

El clima tiene importancia en casi cada fase de la actividad agrícola, desde la preparación de la tierra hasta la cosecha y su almacenamiento. Incluso después de la producción, el clima continúa afectando el futuro de los productores, ya que un reporte de buenas o malas condiciones climáticas tiende a afectar el precio de los productos.

Un sólido conocimiento de los factores climáticos y el entendimiento de la interacción entre el clima y los procesos biológicos en las plantas, es esencial para un acercamiento científico basado en patrones de cultivo planeados, con miras en el mejoramiento de prácticas sobre el manejo del clima y la tierra. La ciencia que se encarga de adquirir y procesar este conocimiento enfocado en las prácticas agrícolas es la agrometeorología.

1.2. Agrometeorología

La agrometeorología, una abreviación de las palabras agricultura y meteorología, tiene como propósito poner la ciencia de la meteorología al servicio de la agricultura con diferentes propósitos, como lo son: ayudar con el uso moderado de la tierra, acelerar la producción de alimentos, evitar el irreversible abuso de los recursos del suelo, entre otros, y se basa en el estudio de las variables meteorológicas, climáticas e hidrológicas que son significativas para el proceso de la producción agrícola.

El campo de estudio de la agrometeorología se extiende desde la capa superficial del suelo hasta la zona donde las raíces de las plantas penetran. En la atmosfera en la capa de aire cerca de la superficie donde las cultivos crecen hasta niveles superiores de la atmosfera por donde semillas, esporas, polen e insectos se transportan.

El rol de la agrometeorología es de dos tipos: estratégico y táctico. Su rol estratégico se basa en asegurar un uso a largo plazo de los recursos naturales para el desarrollo de toda la diversidad de los cultivos, como en el proceso

de selección de los cultivos más convenientes para unas condiciones climáticas anticipadas. Su rol táctico está más enfocado en un periodo a corto plazo y en apoyar la toma decisiones en sitio que directamente influyen en el crecimiento y desarrollo de los cultivos, por ejemplo, en ayudar al agricultor en hacer ajustes en las operaciones agrícolas diarias que tendrán a minimizar las perdidas resultantes por condiciones adversas del tiempo. Empleada de manera correcta, la información agrometeorológica puede ayudar a los agricultores a llevar a cabo una agricultura más sustentable, de alta calidad y redituable con menores riesgos, costos, desechos contaminantes y menos invasiva.

La disponibilidad de una adecuada base de datos meteorológica y agrometeorológica es el mayor prerrequisito para estudiar y manejar los procesos agrícola y de producción forestal. Datos históricos y observaciones durante la presente estación de crecimiento, juegan un rol crítico en incrementar la certidumbre de los modelos de cultivo. Estos modelos tienen un gran potencial para responder interrogantes a problemas relacionados con la investigación, el manejo y los planes de acción para el mejoramiento de los cultivos y ayudan a proveer una mejor dirección para la resolución de problemas reales relacionados con la sustentabilidad agrícola, seguridad alimentaria, uso de recursos naturales y protección al ambiente.

Muchas operaciones agrícolas, servicios y estudios de investigación requieren información sobre el clima de manera diaria, semanal o con una base de diez días. Esta información puede ser generada a través de una red eficiente de estaciones agrometeorológicas, las cuales en este momento son muy pobres en la mayoría de los países. En aquellas partes donde hay una mayor disponibilidad de estaciones meteorológicas, éstas usualmente no siguen un patrón de las zonas agrícolas. Las estaciones instaladas y mantenidas por los departamentos meteorológicos en varios países están usualmente localizadas cerca de ciudades y aeropuertos, donde las observaciones registradas no son representativas para el panorama agrícola.

Sistemas de instrumentación son usados en aplicaciones agrícolas con el fin de obtener información de utilidad sobre las condiciones climáticas, estos sistemas deben proveer mediciones confiables que puedan asistir tanto en el proceso global de administración de los cultivos como en las operaciones agrícolas.

Los sistemas de instrumentación pueden ser utilizados en una variedad de aplicaciones[4], incluidas:

[4] **Campbell Scientific,** *Agriculture: Measurement Instrumentation for Agriculture*, 2007.

- Estimación de la transpiración de los cultivos
- Medición de la humedad en suelos
- Calendarización de irrigación
- Monitoreo de la temperatura
- Manejo integral de plagas
- Aplicación de pesticidas y fertilizantes
- Predicción de nevadas

Las principales variables medidas por estos sistemas son: velocidad y dirección del viento, radiación solar, humedad relativa, precipitación pluvial y temperatura del aire y del suelo. Equipos utilizados en la actualidad como sistemas de medición para realizar el registro de estas variables son los *dataloggers*.

1.3. *Dataloggers*

Un *datalogger* (también *data logger* o *data recorder*) es un dispositivo electrónico que adquiere información proveniente de variables físicas o químicas. Esto a través de un sensor integrado o por medio de sensores o de instrumentos externos. Cada vez más frecuentemente, pero no enteramente, están basados en procesadores digitales (o computadoras). Generalmente son de tamaños pequeños, alimentados con baterías, portátiles y equipados con un microprocesador, memoria interna para el almacenamiento de la información y sensores. Algunos *dataloggers* interactúan con computadoras personales que utilizan algún tipo de software para configurarlos y ponerlos en funcionamiento, además de visualizar y analizar la información recolectada por éste. Otros tienen un dispositivo de interfaz directa con el usuario (teclados, pantallas de cristal líquido, etc.) y pueden ser utilizados como dispositivos *stand-alone*.

Los tipos de *dataloggers* varían desde los de propósito general, utilizados para un amplio rango de aplicaciones, hasta aquellos diseñados para una aplicación en especifico. Los *dataloggers* electrónicos han remplazado en muchas aplicaciones a los registradores mecánicos.

Uno de los beneficios primarios de los *dataloggers* es su capacidad para colectar información las 24 horas del día. Una vez activados, los *dataloggers* son instalados y dejados sin supervisión para medir y guardar datos durante un gran período de tiempo.

1.3.1. Diferencia entre *datalogging* y adqisición de datos

Los términos *datalogging* y adquisición de datos son frecuentemente intercambiados. Sin embargo, en un contexto histórico son completamente distintos. Un *datalogger* es un sistema de adquisición de datos, pero un sistema de adquisición de datos no es necesariamente un *datalogger*.

Los *dataloggers* típicamente tienen menores tasas de muestreo. Una tasa de muestreo de 1 [Hz] puede ser considerada muy rápida para un *datalogger*, sin embargo ésta es muy lenta para un sistema de adquisición.

Los *dataloggers* son implícitamente dispositivos *stand–alone*, mientras que los sistemas de adquisición de datos permanecen ligados a una computadora para adquirir datos. Este aspecto de *stand–alone* de los *dataloggers* implica poseer una memoria integrada, que es usada para almacenar la información obtenida. Algunas veces esta memoria debe tener el tamaño suficiente para almacenar información sin supervisión por días, meses o incluso años.

Los *dataloggers* varían desde dispositivos de un solo canal a complejos instrumentos con múltiples canales. Típicamente, la simpleza del dispositivo implica una menor flexibilidad de uso.

Dado su extenso tiempo de grabado de datos, una característica típica de los *dataloggers* es poseer un mecanismo para el estampado de la información con su respectiva hora y fecha de adquisición. Para ello, dichos *dataloggers* emplean generalmente relojes en tiempo real integrados o receptores de posicionamiento global (GPS).

Por la naturaleza de su operación, independiente de supervisión y remota, muchas aplicaciones requieren que los *datalogges* operen con corriente directa, proveniente de baterías o de paneles solares. Esto también obliga a que el diseño de los *dataloggers* asegure un consumo energético eficiente. La autonomía de un *datalogger* depende básicamente de la energía disponible para su operación así como de la cantidad de memoria disponible para guardar datos.

Adicionalmente, los *dataloggers* deben de ser dispositivos extremadamente confiables, dado que deben de operar por largos periodos sin detenerse, con poca o carente supervisión humana, y pueden estar instalados en un ambiente hostil o en locaciones remotas. Los *dataloggers* son diseñados para ser inmunes a los problemas que pueden afectar a una computadora de propósito

general, realizando la misma función, cuyos fallos pueden tener como origen los programas en ejecución o la inestabilidad del sistema operativo.

Si bien ciertos modelos de *dataloggers*, debido a su aplicación, no tienen la necesidad de conectarse con otros dispositivos, la mayoría son sistemas que debido al volumen de información que adquieren requieren de algún medio de conexión para el vaciado de ésta y para la configuración del *datalogger*. La mayoría de los *dataloggers* poseen cuando menos un protocolo para realizar conexiones alámbricas y pueden integrar protocolos de conexión inalámbrico, dependiendo del alcance de esta conexión un *datalogger* podría trasmitir la información adquirida en tiempo real y sólo utilizar su memoria interna para respaldar la información en caso de una falla de conexión.

1.4. Situación de la agrometeorología en México

En México, el organismo encargado de adquirir, analizar y proporcionar la información meteorológica que se origina de manera nacional y local, es el Servicio Meteorológico Nacional (SMN). Este depende de la Comisión Nacional del Agua (CNA), la cual forma parte de la Secretaría del Medio Ambiente y Recursos Naturales (SEMARNAT).

Las objetivos principales del SMN se concentran en la vigilancia continua de las condiciones atmosféricas para identificar los fenómenos meteorológicos que pueden afectar las distintas actividades económicas y originar la pérdida de vidas humanas.

La infraestructura con la que cuenta el SMN para realizar estos objetivos son los siguientes:

- Una red de estaciones meteorológicas integrada por 72 observatorios.
- Una red de 15 estaciones de radio sondeo, cuya función es la observación de las capas altas de la atmósfera.
- Una red de 12 radares meteorológicos distribuidos en el Territorio Nacional. Los cuales permiten detectar la evolución de sistemas nubosos.

Adicionalmente en el Territorio Nacional opera una red de 171 estaciones meteorológicas automáticas (EMAs) las cuales están administradas por distintas dependencias. Estas estaciones meteorológicas automáticas están conformadas por un grupo de sensores que registran y trasmiten su información

de forma automática de diversas áreas que se consideran estratégicas. Las variables ambientales medidas por estas estaciones son: velocidad del viento, dirección del viento, presión atmosférica, temperatura ambiente, humedad relativa, radiación solar y precipitación. El área representativa de las estaciones es de 5 [km] de radio aproximadamente en terreno plano.

Si bien el SMN ha hecho una excelente labor realizando notificaciones a las autoridades de protección civil, que dan origen a acciones preventivas en casos de fenómenos naturales que puedan poner en peligro a la población, la red de estaciones con la que cuenta no tiene la capacidad ni el propósito de servir como una red de estaciones en aplicaciones agrometeorológicas, debido a que:

- La red existente no cuenta con la cubertura necesaria.
- La ubicación de las estaciones no permite tener datos significativos de las áreas agrícolas del país.
- Los sensores de las estaciones no son suficientes para los fines agrícolas, ya que, por ejemplo, no cuentan con sensores para medir la temperatura del suelo.

1.5. Planteamiento del problema y propuesta de solución

México tienen la necesidad de construir una red de estaciones agrometeorológicas que mediante el análisis de su información colectada contribuyan al mejoramiento y desarrollo de la actividad agrícola, ganadera y forestal del país. Es por ello, que en este tema de tesis se propone el desarrollo de un sistema de instrumentación que tomará la forma de una estación meteorológica agrícola simple.

Este sistema de instrumentación constará de dos partes:

- Un *datalogger* y
- Un programa de cómputo

El *datalogger* será el encargado de tomar las mediciones de las variables agrícolas en sitio, mientras que el programa de cómputo servirá como la interfaz entre éste y una computadora, donde se colectarán y manejarán los datos para un análisis en alguna otra herramienta de software. Las variables meteorológicas elegidas para la adquisición serán temperatura ambiente y precipitación pluvial, dado que son las más importantes para los fines que se persiguen y darán una mayor base de conocimiento para la integración de otros sensores.

Las características del *datalogger* le deben permitir realizar su función de acuerdo a las condiciones que el sitio de adquisición le impone y que son originados por los largos períodos de operación sin supervisión humana. Estas son una operación con base en baterías, gran estabilidad del sistema, bajo consumo de energía y memoria suficiente para almacenar un gran número de eventos. Adicionalmente, la operación del *datalogger* deberá ser sencilla y tener la facilidad de poder ser puesto en operación sin una conexión con una PC.

Dado que el programa será una interfaz entre el *datalogger* y el usuario, éste será el encargado de recibir la información colectada por el *datalogger* y el medio para la configuración de éste. Adicionalmente, dado que el programa no pretende dentro de sus funciones integrar el análisis de los datos adquiridos, éste deberá de exportar la información recibida para ser manejada por algún programa de análisis.

Por último, dado que se pretende sólo realizar la base de lo que puede constituir un proyecto con mayores alcances, se buscará en este proyecto dejar bases sólidas para su mejoramiento e integración tanto de un mayor número de características como de sensores.

1.6. Visión general del sistema

Aquí profundizaremos en la propuesta de solución planteada, además que se señalarán las generalidades necesarias para el diseño y desarrollo del sistema de instrumentación.

1.6.1. El *datalogger*

Partiendo de las función del *datalogger* dentro del sistema de instrumentación, se han considerado tres partes básicas que compondrán al diseño.

- La adquisición de datos
- El guardado de datos
- La comunicación con una PC

La adquisición de datos comprende la caracterización de los sensores, el proceso de acondicionamiento de sus señales y los instrumentos que se usarán para representar estas señales de manera digital.

De manera básica hay dos tipos de adquisición de señales, que dependen de la naturaleza de los sensores y requieren un tratamiento y medio de adquisición distinto. Si el sensor tiene como salida una señal continua en el tiempo, la adquisición de esta señal deberá realizarse a través de un convertidor analógico digital, ejemplos de estos tipos de sensores son los *resistance thermal detectors* (RTD's), los termistores, los termopares, los sensores de humedad relativa, etc. La teoría necesaria para realizar la adquisición y acondicionamiento de esta señal comprende temas como: muestreo de señales, efecto *aliasing*, filtros *antialiasing*, amplificación de señales, operación de un convertidor analógico digital, etc.

Existen otros tipos de sensores que tiene una salida con base en pulsos o señales que pueden ser acondicionadas de esta manera, estos tipo de sensores básicamente denotan algún tipo de frecuencia de un evento y su adquisición se realiza a través del conteo de pulsos, ejemplos de este tipo de sensores son medidores de velocidad del viento (anemómetros), pluviómetros de volcado, etc. La teoría necesaria para desarrollar el acondicionamiento de estas señales abarca, comparadores de tensión, filtros, efecto rebote, etc.

El *datalogger* es un instrumento que debe de adquirir estas dos clases de señales, es por ello que ambos procesos de adquisición deberán de ser implementados en su desarrollo.

La parte de guardado de los datos colectados por el *datalogger* comprende tanto la selección del tipo de memoria a utilizar como el tamaño que tendrá ésta. Lo anterior impacta de manera directa en la autonomía del *datalogger*. Muy poca capacidad de memoria y el *datalogger* no podrá ser usado para adquirir datos por un tiempo prolongado. Demasiada capacidad de ésta hará que el dispositivo consuma más energía y sólo pueda operar por un corto periodo de tiempo. Por ello es necesario realizar una estimación de las necesidades de

almacenamiento que el *datalogger* va a requerir en su aplicación, para asegurar una implementación óptima de la capacidad de almacenamiento. De igual forma el tipo de memoria que se utilizará y que es más conveniente para la aplicación. Un conocimiento de distintos tipos de memoria y su funcionamiento es necesario para el desarrollo de esta parte.

Finalmente, la parte de transmisión de datos a una PC comprende el desarrollo de los protocolos de comunicación entre el *datalogger* y una PC. Lo anterior requiere de un conocimiento de estos protocolos además de la forma de implementarlos.

1.6.2. El programa en la PC

Las funciones del programa de la PC son los siguientes:

- Recibir los datos
- Guardar los datos recibidos
- Exportar los datos

El desarrollo del programa debe comenzar por encontrar las herramientas adecuadas para su programación. El único prerrequisito indispensable de esta herramienta es que debe permitir un rápido desarrollo de las funciones y características que el programa requiere.

Una vez escogidas las herramientas de desarrollo es necesario saber cómo implementar las funciones arriba citadas.

En el siguiente capítulo se describirán las generalidades de este proyecto y se tratará de manera breve los temas de teoría que se consideraron necesarios para entender el desarrollo del proyecto.

Capítulo 2

Generalidades

En este capítulo se expondrán los conceptos teóricos que fueron necesarios para realizar este proyecto, si bien no se abarcaran todos y los que se mencionen no se harán con la suficiente profundidad que estos ameritan, servirán como una guía para una mejor comprensión de los capítulos de diseño.

Los temas abarcados se enfocan en el diseño del hardware, aunque si bien al final se hace mención a ciertas generalidades acerca del lenguaje C#, el cual fue utilizado en el diseño del programa de cómputo.

2.1. Adquisición de señales analógicas

Los temas que se han seleccionado intentan abarcar todo el proceso de adquisición de señales analógicas. Estos temas abarcan desde la teoría de muestreo, el diseño del acondicionamiento y el funcionamiento del convertidor analógico digital, este último permite la representación de las señales analógicas de manera binaria.

2.1.1. Muestreo de señales

En esta sección analizaremos a grandes rasgos el proceso de digitalización de una señal analógica y las consideraciones necesarias para realizar este proceso de manera correcta.

La digitalización es el proceso de convertir una señal analógica a un formato digital. En un formato digital la información puede ser guardada, comprimida, procesada y analizada de forma más sencilla, práctica y rápida que en su formato analógico, de ahí que se prefiera el manejo de la información en un formato digital.

La digitalización de las señales analógicas se hace a través de una conversión analógica–digital en el que la variable analógica continua es cambiada tratando de preservar su contenido esencial. Aun así la señal digitalizada sólo es una aproximación de la señal que representa.

La digitalización ocurre en dos partes, primeramente la señal es discretizada, es decir se toman muestras de la señal en intervalos regulares de tiempo, la frecuencia de estos intervalos se denomina frecuencia de muestreo. Cada lectura es llamada una muestra de la señal analógica y en esta fase se considera que ésta tiene una precisión infinita. La siguiente parte es la etapa de cuantización, en el cual la señal discretizada se establece en niveles de amplitud fijos (niveles de cuantización). En general, estas etapas pueden ocurrir al mismo tiempo aunque son conceptualmente distintas.

La frecuencia de muestreo (f_s) tiene una importancia significativa en asegurar que la señal digitalizada pueda ser posteriormente reconstruida. El teorema de muestreo de Nyquist-Shannon establece que, para que la reconstrucción de una señal sea posible a través de sus muestras, esta señal debe de ser muestreada por lo menos al doble de la frecuencia más alta que contenga[1]. Denotando la máxima frecuencia de la señal analógica a digitalizar como f_m, el teorema de muestreo requiere que:

$$f_s \geq 2f_m \tag{2.1}$$

Todas las señales de entrada con frecuencias menores a $f_s/2$ son sujetas a ser digitalizadas. Si hay una porción de la señal de entrada con frecuencias que se encuentren por arriba de $f_s/2$, esa porción será reflejada dentro del ancho de banda de interés (entre DC y $f_s/2$) con su amplitud preservada. Este fenómeno se conoce como *aliassing* de la señal y nos hace imposible discernir la diferencia entre una señal de frecuencia baja (abajo de $f_s/2$) y una de alta frecuencia (arriba de $f_s/2$).

El fenómeno de *aliasing* de una señal en el dominio de la frecuencia se ilustra en la figura 2.1. Como se puede apreciar en la parte izquierda de la figura, cinco segmentos de bandas de frecuencia son identificados. El segmento $N = 0$ se extiende desde DC hasta la mitad de la frecuencia de muestreo. Dentro del ancho de banda de esta región, el sistema de muestreo registrará de manera fiel el contenido de la señal analógica de entrada. En los segmentos en donde $N > 0$, el contenido de frecuencias de la señal analógica será registrado por el sistema de muestreo en el ancho de banda del segmento $N = 0$. Matemáticamente, las señales de alta frecuencia a la entrada del convertidor A/D (f_{ent}) serán reflejadas de acuerdo a la siguiente expresión:

[1] **Kester W.**, *Analog-Digital Conversion*, pág. 2.27.

$$f_{aliasing} = \mid f_{ent} - N f_s \mid \tag{2.2}$$

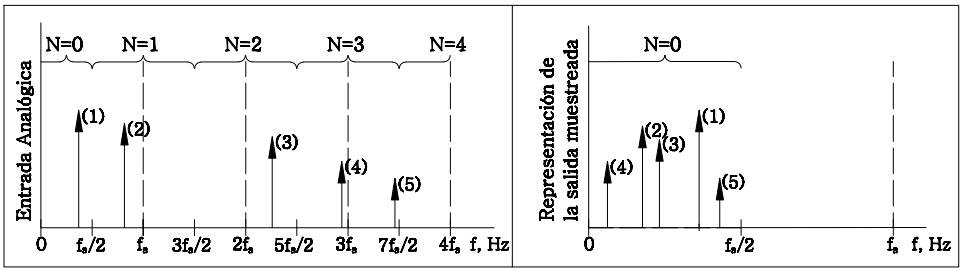


Fig. 2.1. Efecto *aliasing* en el dominio de la frecuencia.

El fenómeno de *aliasing* puede ser eliminado, o significativamente reducido, usando un filtro analógico paso bajas antes de la entrada del convertidor A/D. Este concepto se ilustra en la figura 2.2. En la gráfica, el filtro paso bajas atenúa la segunda porción de la señal de entrada a una frecuencia (2). Consecuentemente esta señal no será reflejada en la salida final muestreada.

Existen dos regiones ilustrados en la figura 2.2. La región a la izquierda que está entre DC y $f_s/2$ se encuentra dentro de la región de la banda de paso del filtro. La segunda región, la cual está sombreada, ilustra la banda de transición del filtro. Dado que esta región de transición es mayor a $f_s/2$, las señales dentro de esta banda de frecuencias serán reflejadas a la salida del sistema de muestreo. Los efectos de este error pueden ser minimizados al mover la esquina de frecuencias más abajo que $f_s/2$ o incrementando el orden del filtro. En ambos casos la mínima ganancia del filtro en la banda de atenuación a la frecuencia de $f_s/2$ debe ser menor a la razón señal a ruido (SNR) del sistema de muestreo.

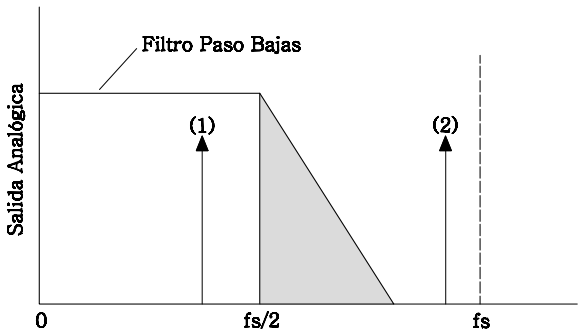


Fig. 2.2. Uso de un filtro *antialiasing*.

Esta razón señal a ruido puede ser calculada para un convertidor A/D mediante la ecuación 2.3[2].

$$SNR = (1.763 + 6.02B)[dB] \tag{2.3}$$

En donde B es el número de bits del convertidor analógico digital.

2.1.2. Diseño de filtros analógicos

Los métodos de aproximación de un filtro ideal más comúnmente utilizados son el Butterworth, Bessel y el Chebyshev. De éstos el más empleado en el diseño de filtros *anti–aliasing* es la aproximación de Butterworth. Esto se debe a que sus características de respuesta plana en la región de paso permiten que la señal no presente una distorsión al ser digitalizada, distorsión que posteriormente derive en un error de medición o tenga que ser corregido posteriormente una vez digitalizada la señal. Por esta razón nos enfocaremos únicamente a la teoría de diseño de este tipo de filtro.

El filtro Butterworth

En la figura 2.3 se muestra una respuesta de magnitud de un filtro Butterworth pasobajas. Su respuesta exhibe una transición que decrece de manera monótona con todos los ceros de transición en $\omega = \infty$, lo que lo hace un filtro totalmente de polos.

En el diseño del filtro Butterworth se deben especificar cuatro parámetros. Estos parámetros se muestran en la figura 2.4 y se listan enseguida:

A_p = atenuación en dB en la banda de paso
A_s = atenuación en la banda de rechazo
f_p = frecuencia a la cual ocurre A_p
f_s = frecuencia a la cual ocurre A_s

La magnitud de la función de transferencia al cuadrado para un filtro Butterworth de orden N-ésimo está dada por[3],

$$| T(j\omega) |^2 = \frac{1}{1 + \omega^{2N}} \tag{2.4}$$

Por ejemplo, la función de transferencia de un filtro Butterworth de segundo orden está determinada por

[2] **Baker B.**, *Nota de diseño analógico ADN010: Predict the Repeatability of Your ADC to the BITB*, Microchip.
[3] **Savant Jr., et al**, *Diseño Electrónico, Circuitos y Sistemas*, pág. 669.

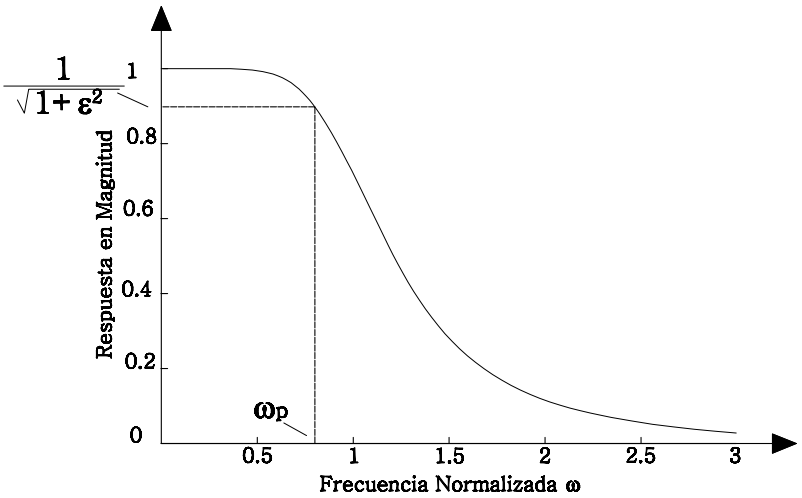


Fig. 2.3. Respuesta en magnitud de un filtro Butterworth.

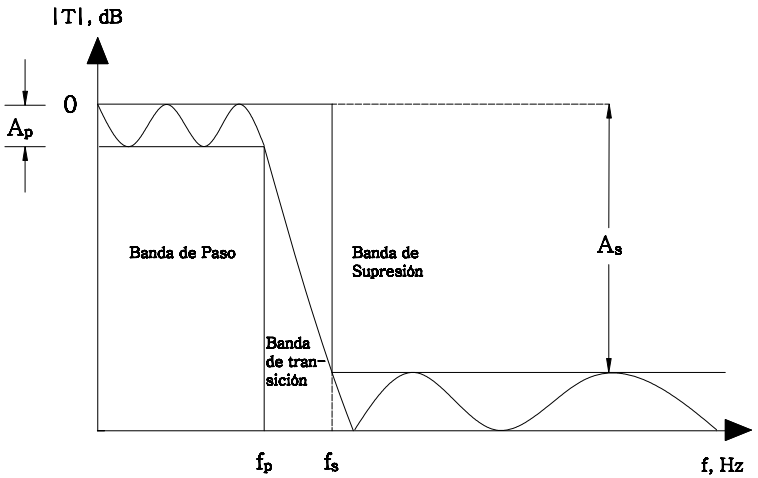


Fig. 2.4. Parámetros de diseño de un filtro.

$$H(s) = \frac{1}{s^2 + \sqrt{2}s + 1} \tag{2.5}$$

Si $s = j\omega$ se sustituye en la ecuación 2.5, la magnitud de la función compleja resultante es la dada por la ecuación 2.4 con $n = 2$.

La ecuación 2.4 es la expresión general para la magnitud al cuadrado de la función de transferencia del filtro Butterworth. Cuando el orden del filtro, N, aumenta, la pendiente de la región de transición se vuelve más inclinada.

De hecho, la pendiente es de $20N$ dB/década donde N es el orden del filtro. Esta observación permite encontrar el orden del filtro requerido cuando se especifica la pendiente.

Suponiendo que las especificaciones de un filtro indican la magnitud en dos puntos, esto es, la magnitud en $\mid H_1 \mid$ a la frecuencia de ω_1 y la magnitud de $\mid H_2 \mid$ a la frecuencia de ω_2. Se encuentra el orden requerido por el filtro de acuerdo con la ecuación 2.4. Sustituyendo los dos valores en esta ecuación, se obtiene

$$\left(\frac{\omega_2}{\omega_1}\right)^N = \sqrt{\frac{\mid H_2 \mid^{-2} -1}{\mid H_1 \mid^{-2} -1}} \tag{2.6}$$

De la ecuación 2.6, despejando el orden del filtro N obtenemos que

$$N = \frac{\log\left(\sqrt{\frac{|H_2|^{-2}-1}{|H_1|^{-2}-1}}\right)}{\log \omega_2/\omega_1} \tag{2.7}$$

De la ecuación 2.7 denotamos $\mid H_1 \mid$ como A_p, ω_1 como ω_p, $\mid H_2 \mid$ como A_s y ω_2 como ω_s. Puesto que A_p es el número de dB debajo de 0 dB, el valor de $\mid H_1 \mid$ es $10^{-A_p/20}$. Otras cantidades se transforman de manera similar. Así,

$$\mid H_1 \mid^{-2} = 10^{+0.1A_p} \tag{2.8}$$

Por lo tanto, el orden del filtro Butterworth (N_B) está dado por el resultado de la ecuación 2.9, en caso de que el resultado sea fraccional se toma el siguiente número entero.

$$N_B = \frac{\log {}^{\epsilon_2}\!/_{\epsilon_1}}{\log {}^{f_s}\!/_{f_p}} \tag{2.9}$$

donde

$$\epsilon_1 = \sqrt{10^{A_p/10} - 1} \tag{2.10}$$

y

$$\epsilon_2 = \sqrt{10^{A_s/10} - 1} \tag{2.11}$$

Los polos de un filtro Butterworth de orden N se determina con la construcción gráfica mostrada en la figura 2.5[4]. Observe que los polos quedan en un circulo de radio $\omega_p(1/\epsilon)^{1/N}$ separados por ángulos iguales de π/N, con el primer polo en un ángulo $\pi/2N$ a partir del eje imaginario.

[4] **Sedra A. y Smith C.**, *Circuitos Microelectrónicos*, pág. 1093.

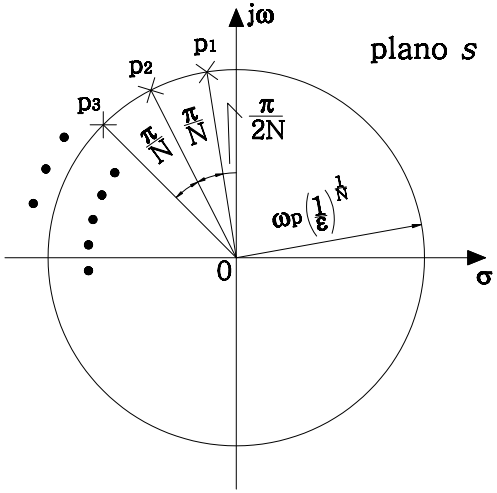


Fig. 2.5. Construcción gráfica para determinar los polos de un filtro Butterwoth.

Como todos los polos están a una distancia radial igual del origen, tienen la misma frecuencia $\omega_0 = \omega_p(1/\epsilon)^{1/N}$. Una vez que se encuentran los N polos p_1,p_2,...,p_N, la función de transferencia se escribe

$$H(s) = \frac{K\omega_0^N}{(s-p_1)(s-p_2)...(s-p_N)} \tag{2.12}$$

donde K es una constante igual a la ganancia DC requerida por el filtro.

En resumen, para encontrar la función de transferencia Butterworth que cumpla con las especificaciones de transmisión requeridas se realiza el siguiente procedimiento:

1. Se determina ϵ_1 y ϵ_2 con las ecuaciones 2.10 y 2.11
2. Se determinar el orden requerido por el filtro como el valor entero más alto del resultado de la ecuación 2.9
3. Se usa la figura 2.5 para determinar los polos del filtro Butterworth
4. Con los polos se determina el polinomio correspondiente a la función $T(s)$

2.1.3. Implementación de filtros analógicos activos

Los filtros activos usan una combinación de un amplificador, una o tres resistencias y uno o dos capacitores para implementar filtros de uno o dos polos.

Filtros de polo único

La respuesta de un filtro de un polo único activo es similar a la de un filtro de un solo polo pasivo, con la ventaja de que el filtro activo, proporciona aislamiento entre etapas. Un ejemplo de su implementación y respuesta se pueden observar en la figura 2.6.

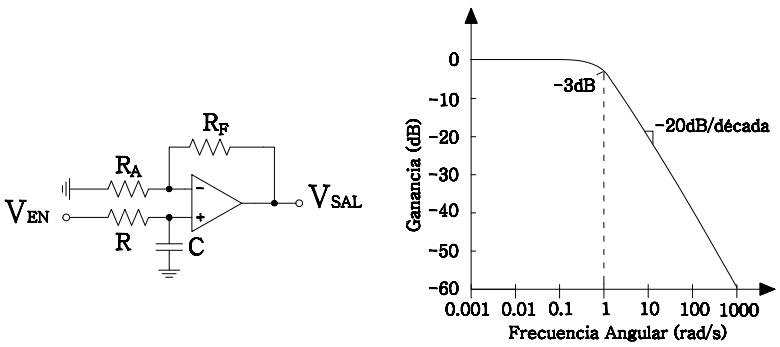


Fig. 2.6. Respuesta en frecuencia de un filtro de polo único.

La función de transferencia del filtro mostrado en la figura 2.6 está dada por la ecuación 2.13.

$$H(s) = \frac{1 + R_F/R_A}{1 + sRC} \tag{2.13}$$

Filtros de polo doble

Topología fuente de tensión controlada por tensión

Este tipo de configuración es mejor conocida como la configuración Sallen-Key. En la figura 2.7 se muestra la implementación de esta topología con una ganancia unitaria en DC. En la implementación mostrada en la figura los polos del circuito están determinados por los valores de R_1, R_2, C_1 y C_2 mediante la siguiente expresión:

$$H(s) = \frac{1}{1 + C_2(R_1 + R_2)s + C_1C_2R_1R_2s^2} \tag{2.14}$$

Topología realimentación múltiple

La topología de un filtro de segundo orden de realimentación múltiple se muestra en la figura 2.8. La ganancia en DC del filtro está dada por la relación

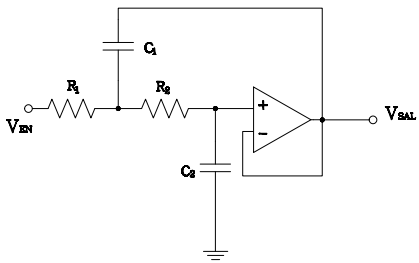


Fig. 2.7. Topología Sallen-Key para un filtro de segundo orden.

entre R_1 y R_2. Sus polos están determinados por los valores de R_1, R_3, a través de la siguiente función de trasferencia:

$$H(s) = \frac{-1/R_1R_3C_5C_6}{s^2C_2C_1 + sC_1(1/R_1 + 1/R_2 + 1/R_3) + 1/(R_2R_3C_2C_1)} \tag{2.15}$$

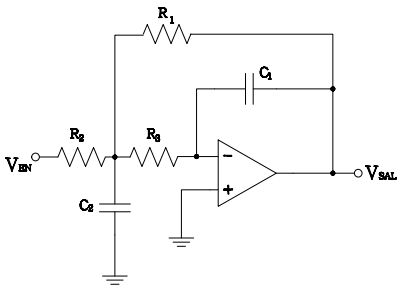


Fig. 2.8. Topología de Realimentación Múltiple para un filtro de segundo orden.

2.1.4. Amplificadores operacionales

Dejando de lado al transistor, el amplificador operacional es el bloque básico de construcción de muchas aplicaciones analógicas. Funciones fundamentales como la de amplificación, aislamiento de carga, inversión, suma y resta de señales pueden ser implementadas con el uso de un amplificador operacional.

Debido a la restricción de no disponer de una fuente de alimentación negativa o la cantidad de componentes necesarios para implementar esta fuente,

no es posible el uso de amplificadores de alimentación bipolar en aplicaciones embebidas.

Cuando se trabaja con amplificaciones de alimentación unipolar cierto tipo de consideraciones de diseño deben ser hechas.

Circuito buffer o seguidor

El circuito *buffer* es útil para resolver problemas con el acople de impedancias y aislar circuitos de potencia de circuitos sensibles. Este circuito puede ser implementado usando un amplificador de fuente unipolar como se muestra en la figura 2.9. En este circuito, como en todos los circuitos con amplificadores operacionales, un capacitor C de *by pass* debe ser usado para evitar que el amplificador pueda llegar a oscilar[5] y eliminar el posible ruido proveniente de otros circuitos. La ganancia de este circuito es unitaria.

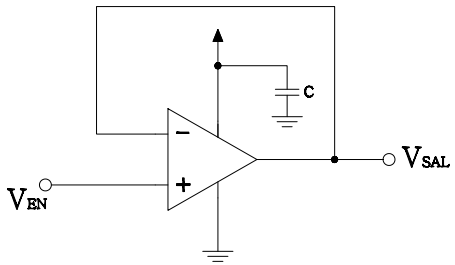


Fig. 2.9. Circuito *buffer*.

Este circuito posee un comportamiento lineal sobre el ancho de banda del amplificador. La única restricción que tiene la señal de entrada del circuito, es que ésta no debe violar los límites de modo común y el rango de variación de salida del amplificador (*output swing*). En caso de querer alimentar cargas de baja impedancia es necesario que las especificaciones del amplificador seleccionado estipulen que éste puede entregar la corriente necesaria a la salida.

Circuitos amplificadores

Si se desea darle ganancia a una señal analógica, dos tipos fundamentales de circuitos amplificadores pueden ser utilizados. El circuito de la figura 2.10 produce una señal amplificada sin inversión a la salida.

[5] **Baker B.**, *AN682: Using single supply Operational Amplifiers in Embedded Systems,* pág. 1.

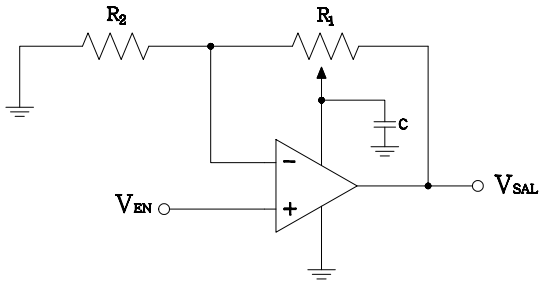


Fig. 2.10. Circuito amplificador no inversor.

La ganancia que este circuito aplica a la señal de entrada está dado por la ecuación 2.16.

$$V_{\mathrm{SAL}} = \left(1 + \frac{R_1}{R_2}\right) V_{\mathrm{EN}} \tag{2.16}$$

Valores típicos de las resistencias usadas en circuitos con fuente unipolar están por arriba de los 2 [kΩ] para R_1[6]. El valor de las resistencias se escoge tomando en consideración el valor de la ganancia deseado para el circuito. Para fijar la ganancia, adicionalmente, deben ser tomados en cuenta el nivel de ruido del amplificador y el *offset* de la señal de entrada.

De nueva cuenta, este circuito tiene restricciones en cuanto al margen de las señales de entrada y de salida. La entrada de la terminal no inversora está restringida al índice de rechazo de modo común del amplificador. El margen de salida del amplificador limita la amplitud de la señal de salida, por lo que si se observan cortes de la señal, se debe disminuir la ganancia del amplificador.

La configuración inversora de un amplificador operacional alimentado con una fuente unipolar se puede lograr a través del circuito mostrado en la figura 2.11. Como se puede apreciar en este circuito, los circuitos inversores que utilizan amplificadores de alimentación unipolar requieren de un divisor de tensión que eleve la señal de salida para mantenerla entre el nivel de alimentación negativa (generalmente tierra) y el nivel positivo.

La salida de este circuito está dado por la ecuación 2.17.

$$V_{\mathrm{SAL}} = -\left(\frac{R_1}{R_2}\right) V_{\mathrm{EN}} + \left(1 + \frac{R_1}{R_2}\right) V_{\mathrm{SESGO}} \tag{2.17}$$

[6] Ibid, pág. 2.

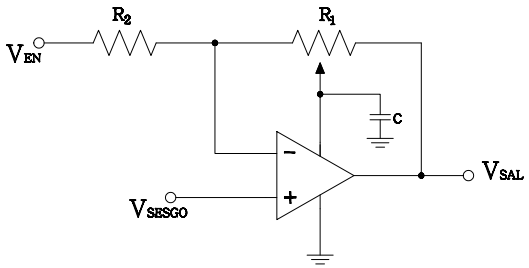


Fig. 2.11. Circuito amplificador inversor.

2.1.5. El convertidor analógico/digital

Un convertidor analógico digital (convertidor A/D) es un dispositivo que convierte una señal continua en números digitales discretos proporcionales a la magnitud de la señal.

Existen diversas formas de implementar un convertidor A/D. Cada una de ellas tiene ventajas y desventajas, la elección de uno u otro tipo de convertidor depende de la aplicación en donde se vaya a emplear. En seguida se describirá el convertidor A/D por aproximaciones sucesivas, el cual fue utilizado en este proyecto.

Si bien hay muchas variaciones para su implementación, la arquitectura básica de un convertidor por aproximaciones sucesivas es sencilla y se muestra en la figura 2.12[7].

Un convertidor A/D de aproximaciones sucesivas utiliza un comparador para rechazar intervalos de tensiones hasta, eventualmente, establecerse en un valor. Estas comparaciones se realizan entre la tensión de entrada y la tensión de salida de su convertidor digital analógico (CDA), él cual es alimentado con el actual valor aproximado. A cada paso de este proceso, el valor binario de cada aproximación es almacenado en un registro de aproximaciones sucesivas (RAS). El convertidor A/D utiliza una tensión de referencia para realizar las comparaciones. El valor de tensión de este referencia (V_{REF}) es el máximo valor de tensión que puede ser convertido sin saturar la salida del convertidor A/D.

El proceso de conversión se realiza de la siguiente manera:

La tensión analógico de entrada (V_{EN}) es mantenido en un circuito de muestreo y retención (*sample/hold*). Para implementar el algoritmo de búsqueda, el registro de N-bits es primeramente establecido a su escala media

[7] **Maxim**, *Application Note 1080: Understanding SAR ADCs.*

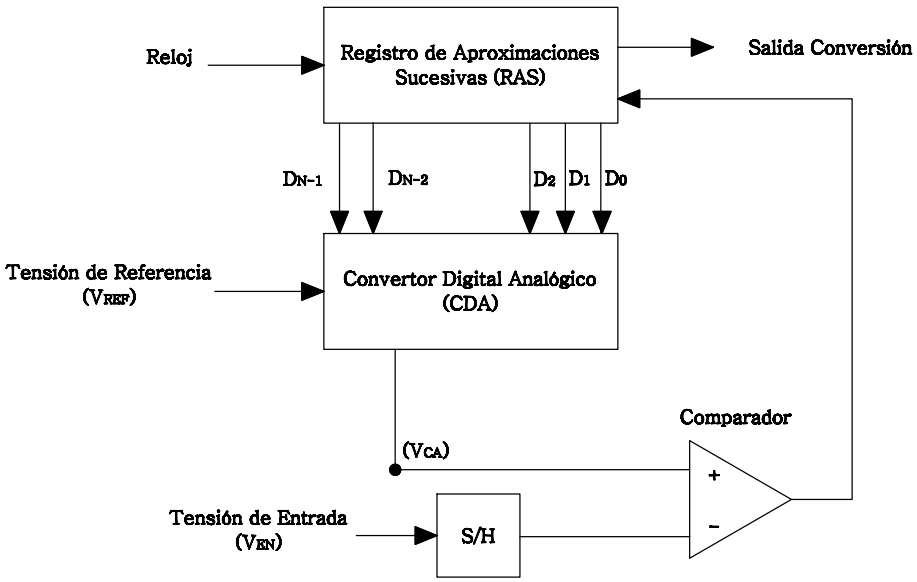


Fig. 2.12. Diagrama de un convertidor de aproximaciones sucesivas.

(esto es, 100...00), donde el bit más significativo se establece en 1. Esto fuerza la salida del CDA (V_{CDA}) volverse $V_{REF}/2$, posteriormente se realiza una comparación entre la salida del CDA y la tensión de entrada. En caso de que la tensión de salida del convertidor sea mayor o igual a V_{EN}, se determina que el bit más significativo debe de ser uno, pero si $V_{EN}<V_{CDA}$, se debe memorizar un cero en el registro de salida. En el siguiente pulso de reloj se efectúa una segunda comparación del V_{CAD} correspondiente a la palabra 110...00, si la comparación anterior había sido positiva ó 010...00 en caso contrario, la salida del comparador determina que valor debe de memorizarse en el bit de peso $V_{REF}/4$, creándose de esta forma la palabra de salida digital una vez realizadas las "n" comparaciones sucesivas; donde "n" es el número de bits de resolución del convertidor A/D.

Dado que cada comparación se realiza en un pulso de reloj, la frecuencia de muestreo del convertidor A/D es la frecuencia del reloj del convertidor dividida entre el número de bits de resolución de éste.

2.2. Adquisición de pulsos digitales

La adquisición de un pulso digital se puede realizar a través de cualquier puerto de entrada de un circuito digital. Dado que la velocidad de recepción de este tipo de circuitos suele ser muy rápida, en caso de que la señal que se recibe contenga ruido, el sistema digital podría interpretar que más de un pulso ha sido enviado, esto es especialmente importante al recibir pulsos provenientes de fuentes mecánicas debido al efecto rebote que generan.

2.2.1. Rebote mecánico

El efecto del rebote mecánico presenta un repentino prendido y apagado de un interruptor en un corto tiempo, debido a las vibraciones mecánicas producidas durante el contacto como se observa en la figura 2.13. El rebote mecánico puede causar un malfuncionamiento de una entrada de un contador o de una terminal de *reset*. Usualmente, el rebote mecánico es eliminado con un filtro pasobajas, pero para asegurar una mayor inmunidad del circuito se puede utilizar una combinación de un filtro pasobajas y un *buffer* con Schmitt Trigger[8].

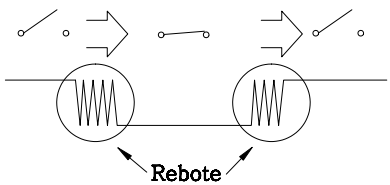


Fig. 2.13. Efecto de rebote en un interruptor mecánico.

Un *buffer* con Schmitt Trigger cambia su estado de salida cuando la tensión en su entrada sobrepasa cierto nivel (V_{u+}); la salida no vuelve a bajar cuando la entrada baja a ese nivel, sino cuando ésta llega a un nivel distinto más bajo que el primero (V_{u-}). A este efecto se le conoce como ciclo de histéresis y se ilustra en la figura 2.14. Los niveles de tensión de los *buffers* varían de acuerdo al modelo y a la tensión con la cual sea alimentado el circuito integrado. Como un ejemplo para el modelo de *buffer* 74LVC1G17 sus valores se muestran en la tabla 2.1[9].

[8] **Maxim**, *Aplication Note 287: Switch bounce and other dirty little secrets.*
[9] **NXP semiconductors**, *74LVC1G17, Single Schmitt trigger buffer.* Hoja de datos del producto.

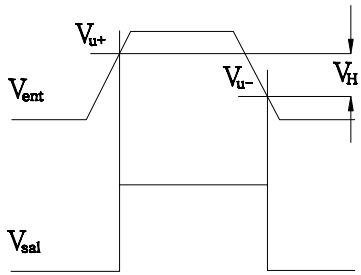


Fig. 2.14. Efecto de histéresis.

Tensión de Alimentación [V]	**Tensión de Umbral Positivo (V_{u+}) [V]**	**Tensión de Umbral Negativo (V_{u-}) [V]**	**Tensión de Histéresis (V_H) [V]**
1.8	1.0	0.6	0.4
3.0	1.5	1.0	0.5
5.5	2.5	1.8	0.7

Tabla 2.1. Valores de tensión del *buffer* 74LVC1G17.

En la figura 2.15 se muestra la propuesta de un circuito para eliminar el rebote mecánico. Como se muestra en el diagrama, el circuito se implementa con un filtro pasivo pasobajas y un *buffer* digital con Schmitt Trigger, adicionalmente, el diodo en el circuito permite la descarga del capacitor para evitar que el valor máximo a la entrada del *buffer* sea excedido.

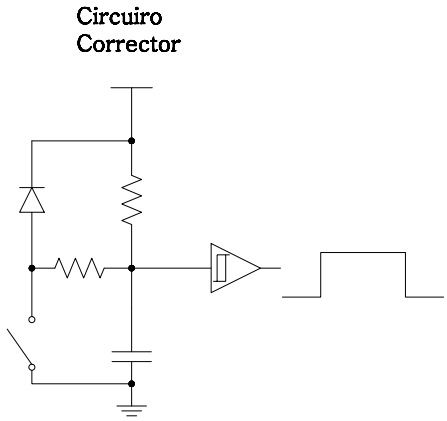


Fig. 2.15. Circuito corrector del efecto rebote.

2.3. Memorias de almacenamiento

El almacenamiento de datos se puede realizar a través de dos tipos de memoria; estos son de tipo volátil o no volátil. Las memorias volátiles abarcan diversos tipos de *Random-Access Memories* (RAM) como las *Dynamic Random-Acess Memories* (DRAM) y las *Static Random-Access Memories* (SRAM). Este tipo de memorias de almacenamiento temporal requieren constantemente estar energizadas para retener la información guardada en ellas.

Las memorias no volátiles son aquellas que retienen la información incluso si el dispositivo es desenergizado. Dentro de esta categoría se encuentran las memorias construidas con base en transistores (flash, *Electrically Erasable Programmable Read-Only Memory* (EEPROM), *Erasable Programmable Read-Only Memory* (EPROM), etc.) y otros tipos de memorias de almacenamiento magnético y óptico.

En el caso del diseño de un *datalogger*, se requiere para el almacenamiento de datos, de memoria no volátil, de tamaño pequeño, reutilizables y de bajo consumo, lo cual descarta las de guardado magnético y óptico. Se requiere también que la memoria elegida sea práctica en su uso, lo anterior descarta a la memorias EPROM debido a que su borrado es a base de luz ultravioleta y las flash dado que su modo de operación es con base en bloques de bits y no de bits individuales. Si bien las memorias EEPROM no tienen la capacidad de almacenamiento de otros tipos de memorias, incluidas las tipo flash, son lo suficientemente densas para almacenar los datos esperados por el *datalogger*. A continuación se describirá su estructura, uso y restricciones.

2.3.1. Memorias EEPROM

Las memorias EEPROM son dispositivos de almacenamiento no volátil, que como su nombre lo indica, pueden ser borrados y escritos de manera eléctrica. Son usados en computadoras y otros dispositivos electrónicos para almacenar una cantidad relativamente pequeña de información.

Las memorias EEPROM al igual que las memorias flash y EPROM son construídas con base en arreglos de *Floating Gate* MOSFETS (FGMOSFETS), gracias a su capacidad de almacenar carga por largos periodos de tiempo sin necesidad de estar alimentadas. En la figura 2.16 se muestra el corte de un transistor FGMOSFET. Como se puede apreciar el óxido de silicio rodea complemente la compuerta flotante, por lo que su carga permanece aislada. La carga en la compuerta flotante puede ser modificada al aplicar tensión eléctrica sobre la

fuente, el drenaje, el cuerpo y la compuerta de control. Esto permite que el transistor funcione como una memoria para un bit de datos.

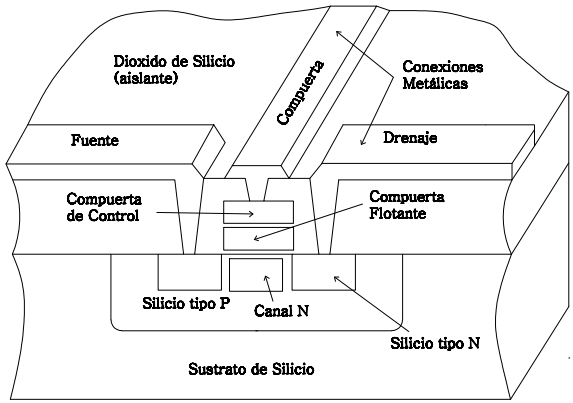


Fig. 2.16. Diagrama interno de un FGMOSFET.

Los tipos de interfaces de comunicación más comunes para este tipo de memorias son el SPI, la I^2C y la 1-WIRE. Estas interfaces requieren entre uno y cuatro terminales de control por lo que la memoria puede ser encapsulada en empaques de ocho terminales o menos.

Las memorias EEPROM típicamente operan en tres fases; fase de código de operación, fase de direccionamiento y fase de datos. La fase de código de operación es generalmente los primeros ocho bits que se envían a la memoria, seguida de ocho o 24 bits de direccionamiento, dependiendo de la cantidad de celdas de memoria que ésta posea, y finalmente, en la fase de datos, se trasmite o recibe la información a leer o escribir.

Cada memoria tiene códigos de operación distintos pero siempre poseerán códigos para la lectura y escritura de datos (en el caso del I^2C, por ejemplo, estos son inherentes al protocolo).

Los modos de falla de las memorias EEPROM en cuanto a retención de datos se deben a dos factores: número de ciclos de escritura y tiempo de retención de datos.

Durante el proceso de reescritura de las memorias, la capa de óxido que rodea a la compuerta flotante gradualmente acumula electrones. El campo eléctrico de estos electrones atrapados se suma a la de los electrones en la compuerta flotante, descendiendo el umbral de tensión entre un cero y un uno almacenado. Después de un número determinado de ciclos de reescritura, las diferencias se vuelven demasiado pequeñas para ser reconocidas y la celda

queda encasillada en un estado programado, es en ese momento que la falla por ciclos de escritura ocurre. El fabricante típicamente especifica un número de reescrituras por celda antes de que esto ocurra, normalmente este valor es de cien mil o un millón de veces.

Por otra parte, los electrones insertados en la compuerta flotante pueden fugarse a través del aislamiento, esto ocurre especialmente si hay un incremento en la temperatura, lo cual lleva a la celda a entrar en un estado de borrado. El fabricante también especifica un tiempo mínimo de retención de datos que típicamente es de diez años.

2.4. Transmisión de información a una computadora personal

Dado que las computadoras personales (PC) son elementos muy estandarizados en su arquitectura, la manera más fácil de realizar una comunicación con una PC es a través de la implementación de uno o varios de los protocolos de comunicación que ésta ya tenga presentes.

Un protocolo de comunicación es un conjunto de reglas estándar para la conexión, comunicación y transferencia de información entre una computadora y otro dispositivo. El protocolo puede ser implementado por software, hardware o una combinación de ambos. Los tipos de comunicación definidos por estos protocolos pueden ser de tipo síncrono o asíncrono.

Los protocolos que se estudiarán en esta sección son aquellos que se implementaron en el *datalogger*. Estos son el protocolo RS-232 y el protocolo USB.

2.4.1. Estándar de comunicación RS-232

El estándar RS-232, definido a comienzos de los años 60, especifica las tensiones, los tiempos y la función de las señales eléctricas; además, el protocolo de intercambio de información y los conectores mecánicos para realizar una comunicación serial asíncrona entre dos equipos que implementen el estándar. La palabra serial significa que la información es enviada bit a bit a través del *bus* de comunicación, la palabra asíncrona nos dice que la información no es enviada en espacios de tiempo predefinidos y la transferencia de datos puede empezar en cualquier momento y es tarea del receptor detectar cuando un mensaje comienza y termina.

Flujo de datos del RS-232

Con una comunicación síncrona, un reloj o una señal de disparo debe de estar presente para indicar el principio de cada transferencia. La ausencia de esta señal de reloj en una comunicación asíncrona, como la definida por el estándar RS-232, hace al canal de comunicación asíncrono más barato de operar pero con la "desventaja" de que el receptor puede empezar a recibir la información en un tiempo inadecuado. En caso de que esto suceda una resincronazación de la señal es necesaria con un costo en tiempo, adicionalmente, toda información recibida en el tiempo de resincronización se pierde. Para recuperar la información perdida es necesario que el receptor detecte los bytes de información faltantes y pida que éstos sean retransmitidos. Otra desventaja es que son requeridos bits extras en el flujo de datos para indicar el comienzo y el fin de la información útil. Estos bits extras utilizan un ancho de banda adicional.

Los bits son enviados con una frecuencia predefinida, el *buad rate*. Ambos el trasmisor y el receptor deben de ser programados para usar la misma frecuencia de transferencia de los bits. Después de recibir el primer bit, el receptor calcula en qué momento el siguiente bit de datos debe de ser recibido. Éste revisará el valor de tensión de la línea de información en ese momento.

Para poder ser enviada la información de manera serial, ésta debe de ser separada en palabras de información. En la figura 2.17 se muestra la representación de una palabra de información del estándar RS-232. La longitud de esta palabra es variable, en una PC esta longitud puede ser seleccionada entre cinco y ocho bits. Esta longitud es la longitud neta de información en cada palabra. Para una trasmisión apropiada son agregados ciertos bits con propósitos de sincronización y de detección de errores, los cuales son: el bit de inicio, el o los bits de paro y el bit de paridad. Es importante que tanto el receptor como el trasmisor utilicen la misma cantidad de bits, de otra manera el mensaje puede no ser interpretado de manera correcta o incluso no ser reconocido.

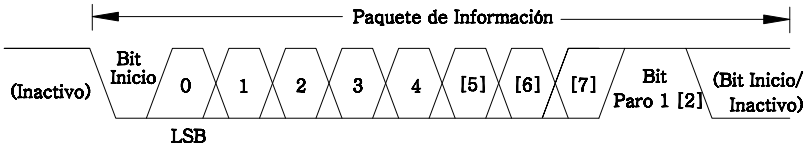


Fig. 2.17. Diagrama de tiempo de la comunicación del RS-232.

Con el estándar RS-232, la tensión de línea puede tener dos estados. El estado encendido es conocido como marca y el estado de apagado como espacio. Ningún otro estado es posible, cuando la línea no es usada, el estado es mantenido en marca.

Los tipos de bits que conforman el flujo de información son los siguientes:

El bit de inicio es un bit de atención para que el receptor se dé cuenta que un paquete de información está por ser enviado. Este bit de inicio siempre es identificado como un espacio. Debido que cuando la línea no es usada, ésta se mantiene en marca, el bit de inicio es fácilmente reconocido por el receptor.

A continuación del bit de inicio los bits de datos son enviados. Un valor de uno causa que la línea vaya a un nivel de marca, el valor de bit cero es representado mediante un nivel de espacio. El bit menos significativo siempre es enviado primero.

Para propósitos de detección de error, es posible adicionar un bit extra que indica la paridad de los bits de datos enviados. El transmisor calcula el valor de este bit dependiendo de la información que va a enviar, el receptor realiza el mismo cálculo y revisa si el valor del bit de paridad corresponde al valor calculado.

Supongamos que el receptor no se da cuenta de un bit de inicio debido al ruido de la línea de trasmisión. Así que éste comienza la recepción en el siguiente valor de espacio que encuentra en los bits de información, esto causaría que información confusa llegara al receptor. Para evitar esta situación, un mecanismo de resincronazación se encuentra presente y hace uso del enmarcamiento que los paquetes de información poseen.

Un enmarcamiento de los paquetes de información significa que todos los bits de datos y de paridad están contenidos en un marco entre un bit de inicio y un bit o bits de paro. El periodo de tiempo entre el bit de inicio y los bits de paro es una constante definida por el *baud rate* y el número de bits de datos y el bit de paridad. El bit de inicio siempre tiene un valor de espacio y el bit de paro de marca. Si el receptor detecta un valor de espacio donde un bit de paro debiera estar presente en la línea, éste reconoce que ha ocurrido un error en la sincronización. El dispositivo entonces trata de resincronizarse con los nuevos bits recibidos.

Para el proceso de resincronización, el receptor busca de la información recibida un par de bits de paro e inicio válidos. Esto funciona siempre que haya una gran variación en el patrón de las palabras de datos. Por ejemplo, si palabras de ceros son enviadas repetidamente el proceso de sincronización no es posible.

Propiedades físicas del estándar RS-232

El estándar RS-232 describe un método de comunicación capaz de intercambiar información en diferentes ambientes. Esto tiene un impacto en los niveles máximos de tensión de las señales, los conectores, etc. En la definición original del estándar fue establecida una velocidad máxima de 20 [kbps]. En la actualidad dispositivos como la UART 16550A alcanzan velocidades hasta de 1.5 [Mbps].

Tensiones

El nivel de tensión de la señal en las terminales del estándar RS-232 puede tener dos estados. Un bit alto, o marca, es identificado por una tensión negativa y para un bit bajo, o espacio, se utiliza un valor positivo. Los voltajes límites están mostrados en la tabla 2.2.

Nivel	**Capacidad del Transmisor** [V]	**Capacidad del Receptor** [V]
Nivel espacio (0)	+5 ... +15	+3 ... +25
Nivel marca (1)	–5 ... –15	–3 ... –25
Indefinido	—	–3 ... +3

Tabla 2.2. Valores de tensión del estándar RS-232.

Cables y Conectores

Según el estándar del RS-232, la distancia máxima del cable de conexión utilizado en la comunicación es aquel con una capacitancia menor o igual 2500 [pF]. Utilizando un cable estándar esta capacitancia se alcanza entre los 15 y 20 metros, pero si, por ejemplo, se utiliza cable UTP CAT-5 con una capacitancia de 52 [pF/m] esta distancia se incrementa considerablemente.

El conector originalmente definido para el estándar RS-232 es el conector DB25, el cual posee 25 terminales para realizar la conexión serial. Debido al gran tamaño de este conector, en los equipos de cómputo que aún poseen este tipo de comunicación se suele encontrar el conector DB9, el cual realiza una conexión con sólo nueve terminales y elimina las terminales del conector DB25 que realizan funciones raramente usadas. En la figura 2.18 se muestra la distribución de las terminales de un conector DB9 macho y hembra, y en la tabla 2.3 se muestra el nombre de dichas terminales y su dirección.

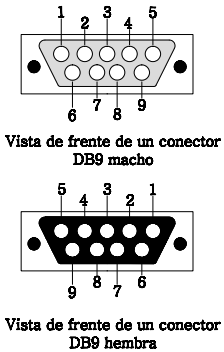


Fig. 2.18. Terminales del conector DB9.

Núm. de Terminal	Nombre	Dirección	Abreviatura
1	*Data Carrier Detect*	Ent	DCD
2	*Received Data*	Ent	RD
3	*Trasmitted Data*	Sal	TD
4	*Data Terminal Ready*	Sal	DTR
5	*Common Ground*	-	GND
6	*Data Set Ready*	Ent	DSR
7	*Request To Send*	Sal	RTS
8	*Clear To Send*	Ent	CTS
9	*Ring Indicator*	Ent	RI

Tabla 2.3. Señales del estándar RS-232 en un conector DB9.

En caso de que no se necesiten señales de control de flujo entre los dispositivos, una comunicación RS-232 se puede realizar con sólo tres terminales, la de recepción de datos (RD), la de transmisión (TD) y la referencia de las señales (GND).

2.4.2. El estándar USB

El estándar USB se ha convertido en el puerto de comunicación entre la computadora y dispositivos externos más usado en la actualidad. Dentro de las características de este estándar están el permitir a los dispositivos una conexión con la computadora sin la necesidad que ésta deba ser reinicializada y la posibilidad de alimentar a los dispositivos conectados a través del mismo *bus*.

Los tres componentes principales del estándar USB son: el anfitrión (*host*), el *hub* y los dispositivos (*devices*). Los dispositivos son el punto final del sistema como ratones, impresoras, etc. Múltiples dispositivos pueden conectarse a un solo puerto USB a través de un *hub*. El dispositivo anfitrión es el centro de la comunicación USB.

El protocolo USB define las características necesarias para la interconexión de los dispositivos con el anfitrión USB y esto incluye los conectores, los cables, la topología y las capas de comunicación.

Flujo de datos del USB

El estándar USB emplea para la transmisión de sus paquetes de información la codificación *Non Return to Zero Inverted* (NRZI). En la figura 2.19 se muestra un ejemplo de este tipo de codificación. La codificación NRZI representa un uno lógico manteniendo el nivel de la línea de datos y un cero lógico mediante el cambio en el nivel de ésta. Una cadena de ceros causa que la línea de datos cambie a cada bit trasmitido mientras que una cadena de unos causa un período sin transición de la línea.

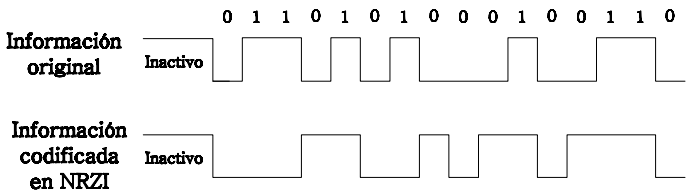


Fig. 2.19. Codificación NRZI.

Topología

La topología se basa en un modelo de conexión estrella escalonada, con el anfitrión en la punta de la pirámide y los *hubs* ejes en el centro de cada conexión con los dispositivos. El anfitrión siempre tiene un camino directo de datos hacia cada uno de los dispositivos.

El anfitrión

El anfitrión controla el estado del *bus* y de los dispositivos conectados a éste. Sólo puede existir un anfitrión en una red USB. Las siguientes son funciones del anfitrión:

- Detectar cuando un dispositivo es conectado y desconectado
- Manejar el control de flujo de datos del *bus*
- Suministrar energía a los dispositivos que así lo soliciten

Dispositivos

Los dispositivos pueden ser descritos de acuerdo a la clase a la cual pertenecen. Una clase es un grupo de dispositivos que tienen similar funcionalidad e implementación, por ejemplo: monitores, dispositivos de audio, de almacenamiento masivo, impresoras, dispositivos de interfaz humana (teclado, ratones, etc.). Esto permite el desarrollo de un controlador genérico para toda una clase de dispositivos y que el usuario no deba de instalar un controlador específico para cada uno. Si un dispositivo tiene más funcionalidades que las presentes en este controlador genérico, un controlador específico debe ser instalado para su funcionamiento.

Todos los dispositivos USB tienen un descriptor que incluye el proveedor del equipo, el identificador del producto y el número de configuraciones que el dispositivo tiene. De esta manera el anfitrión conoce cuantos descriptores de configuración debe de pedir. Los descriptores de configuración incluyen el número y el tipo de interfaces o funciones que el dispositivo posee. Después de recibir todos los descriptores de configuración, el anfitrión pide los descriptores de interfaz. El descriptor de interfaz le dice al anfitrión el tipo de características y qué clase de características el dispositivo tiene.

Un dispositivo puede pertenecer a clases distintas, por ejemplo, si se conecta una cámara web con micrófono, el dispositivo manejara una clase para el audio y otra para el video.

El protocolo de comunicación

El protocolo de comunicación está definido por las rutinas que determinan como los dispositivos interactúan en el *bus*. En el protocolo USB la comunicación entre el anfitrión y los dispositivos conectados al *bus* se realiza a través de paquetes. Una transacción es un grupo de estos paquetes que realizan una función útil y éstas son siempre iniciadas por el anfitrión.

Un ejemplo de la comunicación entre un anfitrión y un dispositivo se puede observar en la figura 2.20. Como se observa en la figura, el anfitrión utiliza un método de encuesta para obtener información de un dispositivo. La comunicación inicia con el anfitrión enviando un paquete *token*. El paquete *token*

es un tipo de paquete sólo usado por el anfitrión USB que informa a un dispositivo que el anfitrión desea realizar una acción sobre éste. Si una petición de información es enviada con el paquete *token*, el dispositivo responde con un paquete de datos o un paquete indicando que no hay información por enviar. Si no existe información, el anfitrión envía un paquete *token* al siguiente dispositivo.

El paquete *token* también puede indicar al dispositivo que espere por información. Si éste es el caso, un paquete de datos será enviado por el anfitrión, al que el dispositivo responderá con un mensaje de confirmación, ya sea que la operación haya tenido éxito (Enterado) o no (No Enterado).

El estándar USB fue diseñado para manejar transacciones a velocidades bajas, medias y altas que corresponden 1.5 [Mbps], 12 [Mbps] y 480 [Mbps] respectivamente y se les refiere como modos de señalización.

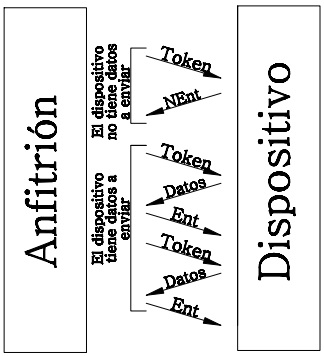


Fig. 2.20. Ejemplo de comunicación entre un anfitrión USB y un dispositivo.

Cables y conectores

Los cables consisten en dos líneas de alimentación y dos líneas para la trasmisión de información, las señales de datos son diferenciales. La máxima distancia permitida de los cables es de 5 [m]. La razón principal de esto es que el máximo retardo que la señal puede tener es alrededor de 1500 [ns]. Si los comandos de un dispositivo no son recibidos en ese tiempo, el anfitrión determina que estos fueron perdidos.

El estándar especifica varios tipos de conectores y sus receptáculos, algunos tienen la intensión de servir para los anfitriones (conector tipo A) y otros para los dispositivos (tipo B), de modo que no se pueda conectar dispositivo con dispositivo y anfitrión con anfitrión. En la figura 2.21 se puede apreciar las

terminales de cada tipo de conector y en la tabla 2.4 el nombre de cada terminal y su función.

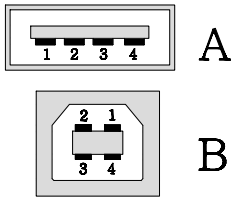


Fig. 2.21. Terminales de los conectores USB A y B.

Terminal	**Nombre**	**Descripción**
1	VCC	+5 [V] de CD
2	D-	Datos -
3	D+	Datos +
4	GND	Tierra

Tabla 2.4. Terminales de los conectores del estándar USB.

2.5. El lenguaje de programación C#

C# (pronunciado en inglés *si sharp*) es un lenguaje de programación orientado a objetos y de escritura segura. C# tiene sus raíces en la familia de lenguajes C y su escritura es parecida a los lenguajes C, C++ y Java. C# fue un lenguaje desarrollado por Microsoft como parte de su plataforma .NET y posteriormente fue estandarizado por el estándar Ecma (ECMA-334) y el estándar ISO/EIC (ISO/IEC 23270). La versión actual del lenguaje es la 3.0 y fue lanzada en conjunto con el .NET Framework 3.5 en 2007.

Dentro de las características de este lenguaje de programación están:

- Incluye programación orientada a componentes, si bien C# es un lenguaje de programación orientado a objetos.
- Contienen un colector de basura automático que libera la memoria utilizada por objetos sin uso.

- Tiene un diseño de escritura segura, lo que hace imposible de leer en variables no inicializadas, arreglos fuera de sus fronteras y hacer *cast* de variables sin revisión.
- Los apuntadores de memoria sólo pueden ser usados en bloques marcados como inseguros, un programa con código inseguro requiere de permisos apropiados para ser ejecutado.
- Incluye un tipo de variable booleano.
- No permite múltiple herencia, si bien las clases pueden ser implementadas con múltiples interfaces.
- Actualmente el lenguaje posee 77 palabras reservadas.

Un ejemplo de código en C# es:

```
using System;
class Hello
{
   static void Main()
        {
      Console.WriteLine("Hello, World");
   }
}
```

Si bien existen compiladores de C# como Mono, desarrollado por Novell, para crear aplicaciones en los distintos sistemas operativos como Linux, Solaris, Mac OS X y Windows, para la realización de este proyecto se utilizó la versión desarrollada por Microsoft, bajo su ambiente Framework .NET contenida dentro de Visual Studio 2008.

El Framework .NET es una estructura de soporte definida, mediante el cual otro proyecto de software puede ser organizado y desarrollado. Este incluye un gran número de herramientas incluyendo interfaz de usuario, conectividad con bases de datos, criptografía, desarrollo de aplicaciones web, algoritmos numéricos y comunicación en red. Su librería de clases es usada para que en combinación con código propio se puedan desarrollar aplicaciones que después puedan ser ejecutadas bajo el ambiente Windows.

Una vez presentados en este capítulo los conceptos teóricos que se consideraron relevantes para entender el desarrollo de este proyecto, el siguiente capítulo se enfoca en estudiar los componentes de hardware que componen el *datalogger*.

Capítulo 3

Desarrollo del Hardware

En este capítulo estudiaremos los elementos físicos que componen al *datalogger*, es decir su hardware, pero no se tratará nada acerca de la interacción que estos elementos realizan para desempeñar las funciones del sistema, esta parte se reservará para el siguiente capítulo.

En la figura 3.1 se observa el diagrama general de las partes que componen al *datalogger*. La figura muestra un diagrama de bloques del sistema, en el centro de éste se encuentra el microcontrolador. La función del microcontrolador es ejecutar el programa contenido en él de manera secuencial. A través de la ejecución de las instrucciones contenidas en este programa, el microcontrolador maneja la interacción tanto de sus módulos internos como los componentes externos a él. La interacción de todos estos elementos permite al *datalogger* realizar sus funciones.

En la parte superior derecha de la figura 3.1 se encuentra un bloque que representa a los sensores. La señal de éstos, una vez pasando por una etapa de acondicionamiento, es adquirida por el microcontrolador a través de dos de sus módulos; el convertidor analógico/digital (CAD) y las terminales de entrada/salida (E/S). Como ya se mencionó, el módulo usado para la adquisición de cada señal depende de la naturaleza de la misma, de esta manera, se utilizó el CAD para adquirir la señal de temperatura ambiente y los puertos de E/S para adquirir la señal de precipitación pluvial.

En la parte izquierda de la figura 3.1 se aprecian ciertos dispositivos que nos servirán para guardar la información y el manejo de las estampas de tiempo, estos son el reloj en tiempo real (RTR) y las memorias EEPROM externas. Ambos dispositivos realizan una comunicación bidireccional a través de la interfaz I^2C con el microcontrolador. Esta conexión sirve, en el caso de las memorias EEPROM, para guardar y leer información de ellas y en el caso del RTR para configurarlo y leer el valor de tiempo y hora de éste.

Adicionalmente al manejo de la fecha y hora, el RTR tiene una función adicional; a través de sus alarmas programables proporciona al microcontrolador una señal externa para que éste realice determinadas funciones programadas.

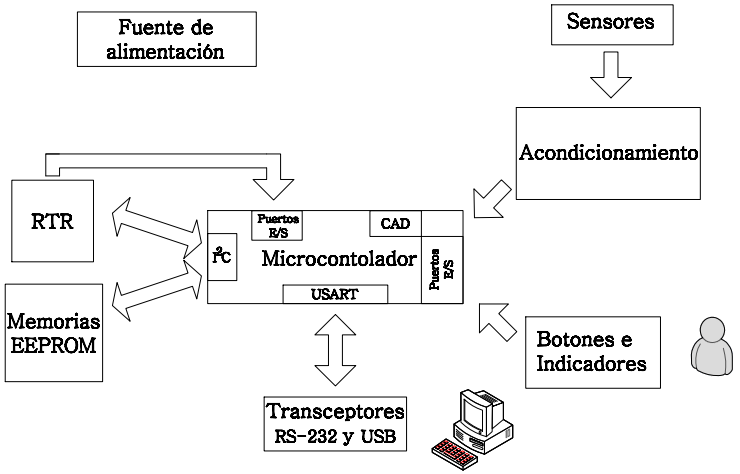


Fig. 3.1. Diagrama de los elementos de hardware que compone al *datalogger*.

Lo anterior evita que el microcontrolador sea el encargado de llevar el control de tiempo de estas funciones. Estas señales son recibidas por el microcontrolador a través de terminales de sus puertos de E/S.

De igual forma, a través de la conexión de controles e indicadores en determinadas terminales de sus puertos de E/S el microcontrolador se comunica con el usuario, lo cual permite una interacción en sitio entre éste y el *datalogger*.

La unidad serie universal síncrona/asíncrona (USART) fue utilizada para implementar los protocolos RS-232 y USB que servirán para conectar al *datalogger* con una PC. Por último se implementó una fuente de alimentación que proporcionará energía a todos los componentes del *datalogger* a partir de una batería de 1.5 [V].

3.1. El microcontrolador

El microcontrolador, al ser el dispositivo de control del *datalogger* y él que proporciona a través de sus módulos gran parte de la funcionalidad de éste, es la primera parte a escoger en la realización del proyecto. Se eligió el microcontrolador ATMega16 de marca ATMEL de la familia AVR para integrar el núcleo del *datalogger*. La elección se basó en que este microcontrolador es

un dispositivo muy barato, disponible en el mercado nacional y que además disponían de las herramientas necesarias para el desarrollo de su programa.

En esta sección se tratará de manera general las características de este microcontrolador y de los módulos e interfaces que lo componen, aunque sólo se profundizara en aquellos que fueron utilizadas para el desarrollo de este proyecto.

3.1.1. Características generales

La familia AVR integra a microcontroladores de 8 bits contenidos en un solo chip basados en una arquitectura modificada Harvard. Esta familia de microcontroladores fue una de las primeras en tener la memoria de programa almacenada en una memoria interna flash.

La arquitectura modificada Harvard es una derivación de la arquitectura Harvard original. Esta arquitectura modificada mantiene la memoria de datos y la del programa separados en diferentes regiones, pero el CPU del microcontrolador tiene la habilidad de leer datos de la memoria del programa utilizando instrucciones especiales.

Dentro de las características de la familia AVR se encuentran:

- Contienen memoria del tipo flash, EEEPROM y SRAM integradas, quitando la necesidad, para muchas aplicaciones, de utilizar memoria externa.
- Las instrucciones del programa están almacenadas en memoria no volátil flash (a pesar de ser microcontroladores de 8 bits cada instrucción ocupa una o dos palabras de 16 bits).
- Su espacio de dirección de datos consta de archivos de registro, registros de entrada/salida y la memoria SDRAM.
- Poseen 32 registros de trabajo de un solo byte.
- Tienen un diseño en base a un *pipeline* de dos niveles. Lo anterior significa que la siguiente instrucción a ejecutar es traída mientras que la actual es ejecutada por el CPU del microcontrolador.
- La mayoría de sus instrucciones se ejecutan en uno o dos ciclos de reloj. Esto los hace relativamente rápidos, en comparación con otros microcontroladores de 8 bits.
- Fueron diseñados teniendo en cuenta una ejecución eficiente, con código compilado de C y contienen varios apuntadores integrados para esta tarea.

- Soportan velocidades de reloj de 0-20 [MHz].

Dentro de las características específicas del modelo de microcontrolador usado se tienen las siguientes:

- 16 [kB] de memoria de programa flash
- 512 [Bytes] de memoria EEPROM
- 1 [kB] de memoria interna SRAM
- Cuatro puertos de entrada/salida (E/S) programables
- Cada puerto de E/S posee ocho terminales
- Dos contadores/temporizadores de 8-bits
- Un contador/temporizador de 16-bits
- Convertidor analógico-digital de ocho canales y 10-bits de resolución
- Interfaz de comunicación I^2C
- Unidad de comunicación síncrona-asíncrona USART
- Interfaz serial maestra/esclava SPI
- Comparador analógico interno

No todos estos módulos del microcontrolador son utilizados por el *data-logger*. En la figura 3.2 se muestra una vista del microcontrolador y de los módulos usados de éste.

Para que el microcontrolador pueda operar sólo requiere de ser alimentado, proporcionarle una señal de reloj y conectar su terminal de *reset* a través de una resistencia de *pull-up* a V_{CC}. Esta resistencia debe de ser no menor a 4.7 [kΩ] cuando se utiliza como fue en este caso un programador SPI. En la figura 3.3 se aprecian estos elementos; el cristal de cuarzo sirve para que el circuito oscilador interno del microcontrolador funcione y a partir de éste se genere la señal de reloj requerida por el microcontrolador. La terminal de *reset* está conectada a V_{CC} y a un interruptor para llevarla a tierra, aunque la terminal de *reset* posee internamente un filtro paso bajas, el capacitor de 22 [pF] proporciona inmunidad adicional al ruido mecánico del interruptor. Adicionalmente, a las terminales de salida del módulo SPI del microcontrolador, se colocó un conector para acoplar, en éstas, el programador que descarga el programa al microcontrolador.

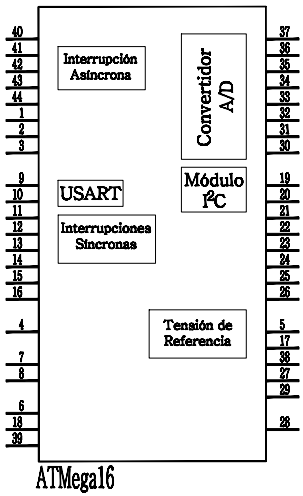


Fig. 3.2. Diagrama de los elementos del microcontrolador usados.

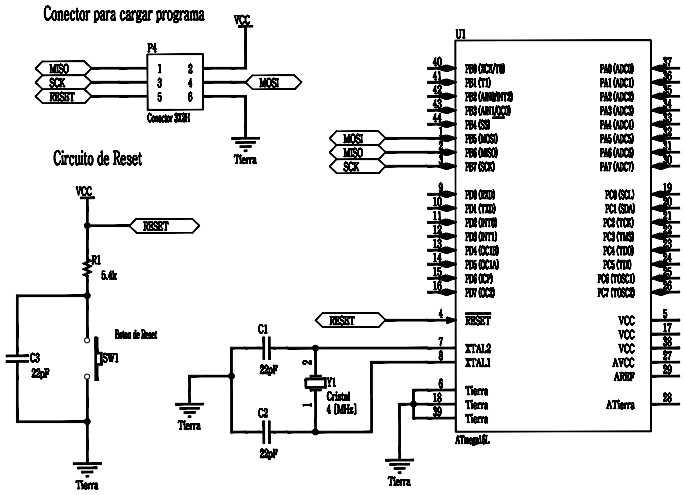


Fig. 3.3. Circuito básico del microcontrolador.

Antes de estudiar el funcionamiento de los módulos internos del microcontrolador, veremos las características generales de éste que ayudaron en la realización del *datalogger*.

3.1.2. La señal de reloj

Para la ejecución de su programa, el microcontrolador ATMega16 requiere de una señal de reloj. Esta señal puede ser provista de manera externa o

generarla al colocar un cristal de cuarzo o un circuito RC en las terminales que contiene el microcontrolador para ello. Las frecuencias de oscilación de estos elementos determinarán la frecuencia de la señal de reloj de salida del circuito oscilador.

Debido a la gran estabilidad en la oscilación de los cristales, se optó por usar uno de ellos con una frecuencia de 4 [MHz].

La frecuencia de reloj determina la velocidad a la que el CPU del microcontrolador funciona y también sirve como una señal de reloj base, para los distintos módulos del microcontrolador.

3.1.3. Modos de ahorro de energía

El microcontrolador ATMega16 incorpora distintas maneras de permitir un ahorro energético en su funcionamiento. En primer lugar, no todos los módulos que están integrados a éste se encuentran encendidos en todo momento. Estos son encendidos individualmente y pueden ser apagados en el instante en que no se requieran, para disminuir el consumo total de energía del microcontrolador.

Dependiendo de la aplicación en que el microcontrolador se use, es posible que éste no requiera estar funcionando en todo momento, con este fin el microcontrolador ATMega16 incorpora varios modos de funcionamiento de bajo consumo, que disminuyen su gasto energético al desactivar todos sus módulos o incluso su propio CPU. Estos modos de bajo consumo son los siguientes: modo inactivo (*idle mode*), modo de reducción de ruido del convertidor A/D (*ADC noise reduction mode*), modo apagado (*power-down mode*), modo de ahorro de energía (*power-save mode*), modo de espera (*standby mode*) y modo de espera extendido (*extended standby mode*).

De estos modos el único que fue usado fue el modo apagado, a continuación estudiaremos sus características.

Modo apagado

En el modo apagado el microcontrolador detiene todos las señales de reloj del sistema incluyendo la de su CPU, por lo que la ejecución de su programa se detiene. En este modo, el consumo de corriente del microcontrolador se reduce a menos de 1 [μA], lo cual representa un consumo mínimo en comparación con su estado activo, en el cual tiene un consumo promedio de 1.1 [mA]. La desventaja de este modo es que, dado que ninguna señal de reloj está activa, existen muy pocas maneras de sacar al microcontrolador de este

estado. En el diseño del *datalogger* se utilizaron las interrupciones externas del microcontrolador para ese fin.

3.1.4. Puertos de E/S e interrupciones externas

Como ya se mencionó, en las características generales del ATMega16, este microcontrolador posee 32 terminales de E/S programables, distribuidas en 4 puertos. Sin embargo, no todas estas terminales están disponibles en todo momento. Lo anterior se debe a que éstas también desempeñan la función de servir como entradas y salidas a los distintos módulos internos del microcontrolador. Una vez que cierto módulo es activado, éste toma control del puerto y lo configura de acuerdo a sus necesidades.

Las líneas de E/S se configuran a través de registros del microcontrolador. Cada terminal de un puerto puede ser configurado de manera independiente, por lo cual se puede hacer una configuración mixta de entradas y salidas de un determinado puerto. En caso de configurar una terminal como entrada, a ésta se le puede activar su resistencia interna de *pull up*. Cada terminal cuenta con diodos de protección a tierra y al voltaje de alimentación y tiene la capacidad suficiente para que se drene de ellos hasta 40 [mA]. El diagrama interno de un terminal de un puerto se puede apreciar en la figura 3.4.

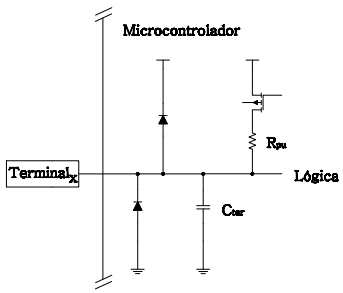


Fig. 3.4. Diagrama de un puerto E/S del microcontrolador.

Las terminales de E/S del microcontrolador, configuradas como entradas, permiten que el microcontrolador lea el valor de tensión que posea esta terminal. Si esta tensión se aplica a una terminal de entrada que pueda disparar una interrupción externa y dicha interrupción se encuentra configurada y activada, una interrupción en la ejecución del microcontrolador ocurre. Las interrupciones en el ATMega16, al igual que en cualquier microcontrolador, son señales que le indica a la ejecución de un programa que cierto evento ha ocurrido y que requiere de su atención. La ventaja del uso de interrupciones es que evitan gastar tiempo del procesador en estar revisando, constantemente, si algún tipo

de evento ha ocurrido. Las terminales con interrupción externa, de acuerdo al modo en que detectan las señales a su entrada, se dividen en dos tipos:

- Síncronas y
- Asíncronas

Una terminal con modo de detección síncrona requiere de una señal de reloj para detectar un cambio en la señal de entrada. En el caso del microcontrolador usado, esta señal de reloj es generada a través del reloj principal del sistema. Las terminales del microcontrolador ATMega16 que poseen un modo de detección síncrono son dos y se pueden configurar para lanzar una interrupción en caso de que detecten cualquiera de las siguientes señales:

- Un nivel de señal bajo en la terminal
- Cualquier cambio en el nivel de la terminal
- Un flanco de bajada en la terminal
- Un flanco de subida en la terminal

Una terminal con modo de detección asíncrona no requiere señal de reloj alguna para detectar cambios en su entrada. El microcontrolador ATMega16 sólo posee una terminal con este tipo de interrupción y ésta unicamente puede ser configurada para generar una interrupción mediante un flanco de bajada o de subida.

Todas las interrupciones externas pueden usarse para despertar al microcontrolador del modo de bajo consumo usado. Para realizar esto, en el caso de las interrupciones síncronas, sólo lo pueden realizar mediante la detección de un cambio en el nivel de la señal a la entrada, mismo que deberá de mantenerse hasta que el microcontrolador salga del estado de bajo consumo. Señales más cortas no garantizan despertar al microcontrolador. En el caso de la interrupción asíncrona la detección de un flanco de subida o bajada es suficiente para reanudar la operación normal del microcontrolador.

3.1.5. El convertidor analógico/digital

El convertidor analógico digital (A/D) del microcontrolador posee las siguientes características:

- Realiza conversiones a una resolución máxima de 10 bits.
- Su método de conversión es a través de aproximaciones sucesivas.

- Puede ser configurado para utilizar una tensión de referencia generada por el mismo microcontrolador o una generada de manera externa.

Un convertidor de aproximaciones sucesivas realiza una conversión a través de un reloj, un registro y un convertidor digital analógico, éste último requiere de una tensión de referencia. La salida del convertidor analógico digital es de 10 bits con un margen que va desde tierra (0000_{H}) hasta su tensión de referencia ($03\mathrm{FF}_{\mathrm{H}}$). En el caso del *datalogger*, para mantener un número mínimo de componentes externos, se decidió utilizar la tensión de referencia generada internamente por el microcontrolador, la cual tiene un valor de 2.56 [V]. Esta tensión está disponible en una terminal del microcontrolador con el fin de desacoplar la señal por medio de un capacitor y de esa manera obtener un mejor desempeño contra el ruido.

La palabra digital esperada a la salida del convertidor A/D está dada mediante la siguiente expresión:

$$CAD_{valor} = \frac{V_{ent} \cdot 1024}{V_{ref}} \tag{3.1}$$

donde:

- CDA_{valor} es el valor de salida del convertidor A/D
- V_{ent} es la tensión de la señal de entrada al convertidor
- V_{ref} es la tensión de referencia usada para el convertidor A/D, que en nuestro caso tiene un valor de 2.56 [V]

Con un V_{ref} de 2.56 [V] y 10 bits de resolución, la resolución de salida del convertidor A/D en terminos de volts es de 2.5 [mV].

La velocidad de conversión del convertidor A/D está dada a partir de su frecuencia de reloj. Esta frecuencia se genera a través de escalar la frecuencia del reloj principal del microcontrolador. De acuerdo a las hojas de especificaciones del microcontrolador, para realizar una conversión a 10 bits el convertidor A/D requiere de una frecuencia de reloj que se encuentre entre 50 [kHz] y 200 [kHz]. En nuestro caso esta frecuencia es de 62.5 [kHz][1]. El convertidor A/D del microcontrolador requiere de 14.5 ciclos entre conversión y conversión, por lo que la frecuencia de muestreo del convertidor A/D con esta configuración es de 4.310 [kHz].

[1] Considerando un reloj del sistema de 4 [MHz] y un preescalador de 64.

3.1.6. Unidad de trasmisión/recepción síncrona/asíncrona (USART)

La USART es un dispositivo de comunicación *full-duplex*, muy flexible en cuanto a su configuración y uso. El módulo integrado en este microcontrolador posee las siguientes características:

- Posee líneas independientes de transmisión y recepción de datos
- Operación síncrona y asíncrona
- Soporta paquetes de cinco, seis, siete, ocho o nueve bits de datos y uno o dos bits de paro
- Generación y revisión de paridad par o impar por medio de hardware

El *datalogger* solamente emplea la USART para realizar una comunicación asíncrona. Utilizada de esta manera y con una configuración adecuada, los pulsos de salida de la USART son compatibles con el estándar de comunicación RS-232, aunque requieren de cierto acondicionamiento.

Un paquete de transmisión en modo asíncrono consiste en una palabra de datos con bits de sincronización (bits de inicio y paro) y opcionalmente un bit de paridad para la detección de error. La USART acepta todas las treinta combinaciones de los siguientes características como paquetes validos:

- Un bit de inicio
- Cinco, seis, siete, ocho o nueve bits de datos
- Bit de paridad par, impar o ninguno
- Uno o dos bits de paro

Un paquete de datos comienza con un bit de inicio, seguido por el bit menos significativo del dato, posteriormente le continúan los bits de datos hasta completar el número configurado, terminando con el bit más significativo. Si ha sido configurado, el bit de paridad es insertado después de los bits de datos y antes de los bits de paro, cuando una secuencia de datos es completamente trasmitida puede inmediatamente seguir otra o la línea, puede quedar en estado inactivo (alto).

Ningún hardware externo es necesario para hacer funcionar este módulo y las líneas que se utilizan, la de transmisión (Tx) y recepción (Rx), se encuentran referenciadas a la tierra del circuito.

3.1.7. El módulo I²C

La interfaz de comunicación del I²C o *Two-Wire Interface* (TWI) es ideal para utilizarse en aplicaciones con microcontroladores, dado que con sólo dos líneas bidireccionales es posible comunicarse con hasta 128 dispositivos conectados al *bus*. Una de estas líneas es dedicada para la señal de reloj (SCL) y la otra para la señal de datos (SDA). El único hardware externo necesario para que el módulo de comunicación I²C funcione son dos resistencias de *pull-up*, una en cada línea del *bus*. Todos los dispositivos conectados al *bus* tienen direcciones independientes y el mecanismo para la conexión de los dispositivos es inherente al protocolo I²C.

En la figura 3.5 se muestra la conexión del modulo I²C a distintos dispositivos, se pueden apreciar las resistencias de *pull-up* necesarias para llevar a cabo la comunicación.

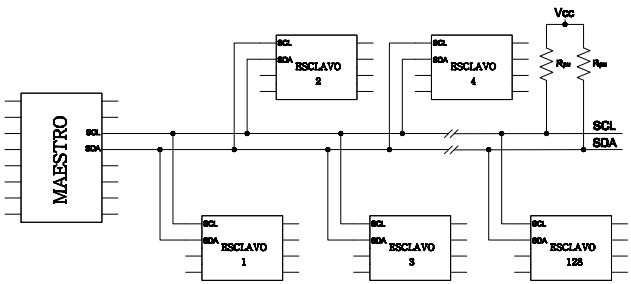


Fig. 3.5. Ejemplo de conexión del *bus* I²C.

El valor de las resistencias de *pull-up* dependen de la velocidad a la que se establece la comunicación I²C. Para una velocidad de comunicación de 100 [kHz], con la cual se configuró este módulo, un valor de resistencias de 10 [kΩ] es adecuado[2]. La máxima capacitancia permitida para *bus* I²C es de 400 [pF] valor que límita tanto su longitud como el número de dispositivos que se pueden conectar a éste.

La terminología del protocolo I²C se refiere a un dispositivo MAESTRO como aquel dispositivo que inicia la comunicación en el *bus*, éste a su vez es el encargado de generar la señal de reloj. En el *bus* puede estar conectado más de un MAESTRO, pero sólo uno puede tomar el control del *bus* a la vez. Un dispositivo ESCLAVO es el dispositivo direccionado por el dispositivo MAESTRO. El TRASMISOR es el dispositivo que está poniendo información en el *bus* y el RECEPTOR es aquel dispositivo que recibe datos del *bus*.

[2] **Microchip**, *24LC1025 CMOS Serial EEPROM.* Hoja de datos del producto.

Cada bit de datos trasmitidos en el *bus* del I²C es acompañado por un pulso en la línea del reloj. El nivel de la línea de datos debe de mantenerse estable cuando la línea de reloj esté en alto, la única excepción a esta regla es durante la generación de las señales de inicio y paro de la transmisión. Dichas operaciones las realiza un dispositivo MAESTRO, al cambiar el nivel de la línea de datos mientras la línea del reloj está en alto.

Un ejemplo de una comunicación típica de datos del protocolo I²C se puede apreciar en la figura 3.6. Una vez *iniciada la transmisión* por el MAESTRO, mediante una señal de inicio, manda un *paquete de direcciones* que consta de nueve bits, esto considerando siete bits de dirección, uno de lectura/escritura y uno de *acknowledge* (enterado)(ACK). Si el bit de lectura/escritura es un uno lógico, indica que el MAESTRO desea realizar una operación de lectura sobre el ESCLAVO direccionado, en caso contrario la operación requerida es de escritura. Cuando el dispositivo ESCLAVO reconoce que ha sido direccionado, éste debe de bajar la línea de datos en el bit nueve, si el dispositivo ESCLAVO está ocupado o por cualquier otra causa no puede atender el llamado del MAESTRO, éste debe dejar la línea datos en "alto" durante el noveno ciclo de reloj.

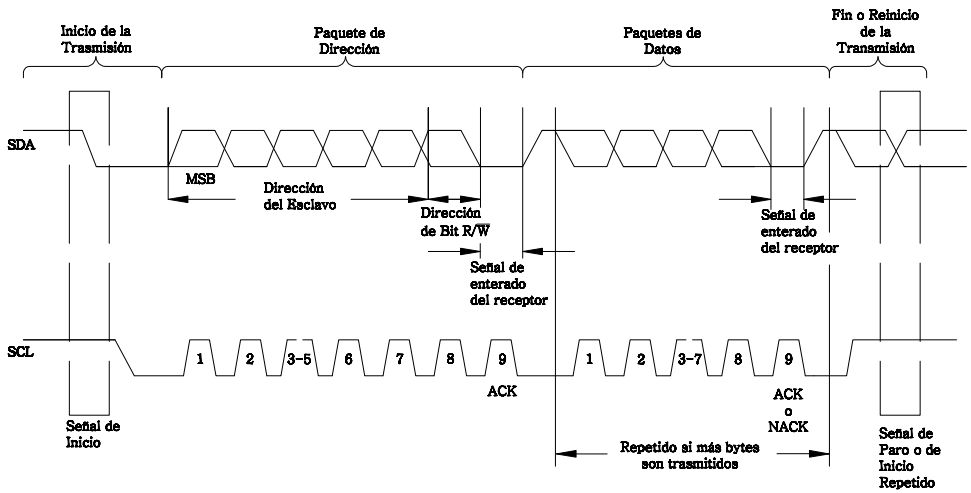


Fig. 3.6. Diagrama de tiempo del protocolo I²C.

Una vez que el paquete de dirección es reconocido por el ESCLAVO, el TRASMISOR envía los *paquetes de datos*. Todos los paquetes de datos trasmitidos por el *bus* I²C son de nueve bits, considerando ocho bits de datos y un bit de enterado. El bit más significativo del byte de datos es trasmitido primero. Durante

la transmisión de información el MAESTRO es el encargado de generar la señales de inicio y paro y la señal de reloj, mientras el RECEPTOR es el encargado de mandar las señales de enterado al final de cada paquete de información. Si el RECEPTOR mantiene la línea de datos "alta", esto se considera una señal de *no acknowledge* (no enterado) (NACK). Cuando el RECEPTOR ha recibido el último bit o por alguna razón ya no le es posible recibir más bits, éste debe informárselo al TRASMISOR con una señal de NACK al final del último byte.

Si una señal NACK es recibida por el TRASMISOR, en caso de ser éste un dispositivo ESCLAVO, dejará de trasmitir información o en caso de ser un dispositivo MAESTRO podrá enviar una señal de paro o un inicio repetido para reiniciar la transmisión. El *inicio repetido* ocurre cuando una nueva señal de inicio es hecha entre una condición de inicio y paro, y sucede cuando un MAESTRO desea realizar una nueva transmisión sin ceder el control de éste.

La *señal de paro* es trasmitida por el MAESTRO cuando el intercambio de información ha terminado y éste desea liberar el *bus*. Entre una señal de inicio y una de paro el *bus* se considera ocupado y ningún otro MAESTRO puede intentar tomar el control de éste.

3.2. Adquisición de datos

La adquisición de datos es la parte más importante de un sistema de instrumentación, es aquí donde se determina gran parte de la funcionalidad y calidad del sistema de instrumentación. Un error en el diseño de esta parte conllevará a una salida llena de errores, que hará inútil todo el sistema en su conjunto. Es por ello que, un buen diseño debe comenzar por asegurar que las variables medidas están siendo adquiridas de manera adecuada y que éstas describen de la manera más próxima la variable a medir.

3.2.1. Sensores

Los sensores son el primer componente en la cadena de instrumentación. Estos convierten las variables físicas o químicas de un fenómeno en señales eléctricas, que casi siempre serán señales analógicas. Esta conversión debe de ser lo más fiel a la señal involucrada en la medición. Sólo un conocimiento minucioso de la respuesta de un sensor garantiza el éxito de una medición. En ocasiones los sensores producen señales corruptas debido a interferencia, las condiciones de uso o por el mismo proceso de medición.

Un sensor funciona convirtiendo una cantidad física medible (m) en variable eléctrica (s). Esta señal eléctrica puede ser impedancia, una carga eléctrica, una corriente o una diferencia de potencial. La relación entre s y m es una función $s = F(m)$ y depende de:

- La ley física que determina el comportamiento del sensor
- La estructura interna del sensor
- El ambiente en el que se encuentra el elemento sensor del sensor

Mediante el uso de un estándar o unidades de medición, encontramos los valores de la señal eléctrica m ($m_1, m_2 ... m_i ...$) que son enviadas por el sensor en respuesta a la variable sensible ($s_1, s_2 ... s_i ...$). En una calibración el valor de entrada (m) debe ser una variable controlada independiente, mientras que el valor de salida (s) es la variable dependiente de la calibración. Una correlación tendrá la forma $s = f(m)$ y se determina aplicando a la curva de calibración un razonamiento físico y técnicas de ajuste de curvas. En la figura 3.7 se muestra un ejemplo de la construcción de una curva de calibración, así como ejemplos de valores medidos que permitieron trazar esta curva. La correlación obtenida a partir de la calibración de un sensor, podrá posteriormente ser utilizada para encontrar el valor de entrada desconocido con base en el valor de salida indicado por el sistema de medición.

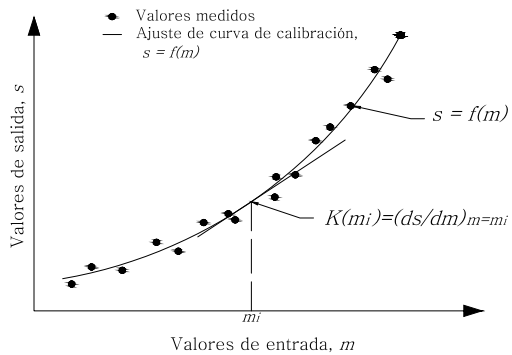


Fig. 3.7. Curva representativa de calibración.

Un parámetro importante de un sensor y que se obtiene a través de la curva de calibración es la sensibilidad. Llamaremos sensibilidad K a la razón de cambio de la señal de salida eléctrica del sensor m en relación con el cambio de la señal de entrada del sensor s, esta relación la denotaremos con una derivada:

$$K = K(m_i) = \left(\frac{ds}{dm}\right)_{s=s_1} \tag{3.2}$$

donde K es una función de m.

Puesto que las curvas de calibración pueden ser lineales o no lineales, según el sistema de medición y la variable que mide, K puede o no ser una constante en un intervalo de valores de entrada.

Como se mencionó en el capítulo introductorio, las variables medidas por el *datalogger* corresponden a las variables meteorológicas de temperatura ambiente y precipitación pluvial. Para realizar la medición de estas variables se eligieron, del universo de sensores disponibles para este fin, aquellos que se tenían disponibles y que adicionalmente estuvieran dentro de los recomendados por la Organización Meteorológica Mundial (OMM)[3] [4]. Los sensores que se utilizaron fueron una sonda de temperatura basada en un termistor, para realizar la medición de la temperatura ambiente, y un pluviómetro de balancín, para la medición de la precipitación pluvial.

La sonda de temperatura

El sensor elegido para realizar las mediciones de temperatura ambiente es la sonda de temperatura 107 fabricado por la compañía Campbell Scientific. Una fotografía de la sonda 107 utilizada se muestra en la figura 3.8.

Fig. 3.8. Fotografía de la sonda de temperatura 107 usada.

La sonda tiene como características generales las siguientes:

- Es capaz de realizar mediciones de temperatura en suelo, agua y aire

[3] **WMO**, *Guide to Meteorological Instruments and Methods of Observation*, pág. 1.2-12.
[4] ibid, pág. 1.6-10.

- Su margen de medición es de los -35 [°C] a los +50 [°C]
- Los errores de precisión de la sonda se encuentran de ±0.4 [°C], en el margen de los -24 [°C] a los 48[°C], y ±0.9 [°C], en el margen de los -38 a los 53 [°C]
- Posee una constante de tiempo entre 30 y 60 segundos para una velocidad del viento de 5 [ms^{-1}]

El sensor de la sonda de temperatura es un termistor tipo *Negative Temperature Coefficient* (NTC). Un termistor es un sensor de temperatura el cual varía su resistencia de acuerdo a su temperatura, el término NTC se refiere a que es de coeficiente negativo, por lo cual su resistencia disminuye conforme su temperatura aumenta.

La exactitud general de la sonda está determinada por una combinación de las especificaciones de intercambiabilidad, la precisión del puente de resistencias y el error de la ecuación Steinhart-Hart. El mayor error es debido a las especificaciones de intercambiabilidad del termistor. En el margen de 0 [°C] a 50 [°C] el error de intercambiabilidad es mayoritariamente un *offset* que puede ser corregido con una calibración de un solo punto[5].

La estructura interna de esta sonda se observa en la figura 3.9. Como se puede observar la resistencia R_1 junto con el termistor forman un divisor de tensión con la resistencia R_2. El divisor está formado por resistencias de una tolerancia de 0.1%, con un coeficiente de temperatura de 10 [ppm/°C].

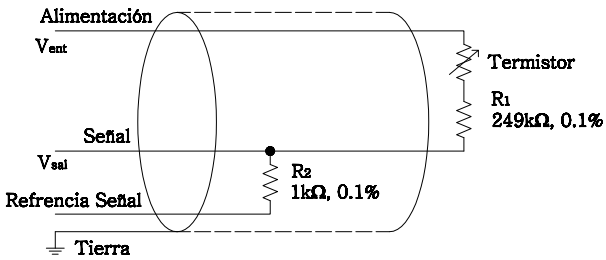


Fig. 3.9. Estructura de la sonda de temperatura 107.

La ecuación que determina el valor de tensión a la salida de la sonda está dado por:

$$V_{sal} = \frac{1000 V_{ent}}{R_{termistor} + 250000} [\text{ V}] \tag{3.3}$$

[5] **Capbell Scientific, Inc.**, *User Manual Model 107 Temperature Probe*, pág. 2.

Para determinar la relación de resistencia del termistor ($R_{termistor}$) con su temperatura es necesario utilizar la ecuación de Steinhart–Hart, cuya expresión se muestra en la ecuación 3.4.

$$\frac{1}{T} = A + B\log R + C(\log R)^3 \tag{3.4}$$

donde:

- T es la temperatura en Kelvins
- R es la resistencia a la temperatura T en Ohms y
- A, B, C son los coeficientes de la ecuación de Seinhart-Hart que varían de a cuerdo al modelo del termistor y el margen de temperatura de interés

Para este modelo de termistor los coeficientes A, B y C tienen los siguientes valores[6]:

$$\begin{aligned} A &= 8.271111\times10^{-4} \\ B &= 2.088020\times10^{-4} \\ C &= 8.059200\times10^{-8} \end{aligned}$$

Para que por medio de las ecuaciones 3.3 y 3.4 se puede determinar la tensión de salida a determinada temperatura de la sonda, es necesario solamente fijar una tensión de alimentación V_{ent}. Esta tensión de alimentación debe de ser una tensión constante y estable en todo momento, para asegurar obtener una medición libre de errores. En nuestro caso está tensión es de 2.56 [V] y es generada internamente por el microcontrolador como un voltaje de referencia para el convertidor A/D.

Finalmente, con fines de caracterizar la sonda de temperatura, se puede utilizar la ecuación inversa de Steinhart–Hart, que se muestra en la ecuación 3.5, para obtener la resistencia del termistor dada una temperatura de entrada[7].

$$R = \exp\left(\sqrt[3]{x - \frac{y}{2}} - \sqrt[3]{x + \frac{y}{2}}\right) \tag{3.5}$$

donde:

[6] ibid, pág. 9.

[7] **J. S. Steinhart and S. R. Hart**, *Calibration curves for thermistors*, Deep-Sea Res., 15:497, 1968.

$$y = \frac{A - \frac{1}{T}}{C}$$

$$x = \sqrt{\left(\frac{B}{3C}\right)^3 + \frac{T^2}{4}}$$

El pluviómetro

El pluviómetro es un tipo de instrumento usado por los meteorólogos e hidrólogos para acumular y medir la cantidad de líquido precipitado en un determinado período de tiempo. La mayor parte de los pluviómetros tienen como unidad de medida los milímetros. Una precipitación de 5 [mm] indica que si toda el agua se acumulada en un terreno plano y sin escurrir de un metro cuadrado, la altura de la capa de agua sería de 5 [mm]. Por lo que los milímetros son equivalentes a litros por metro cuadrado. Para este proyecto se utilizó un pluviómetro de balancín, dado que su salida es bastante adecuada para la toma de mediciones por medio de un *datalogger*.

Los pluviómetros de balancín o de volcado consisten en un gran cilindro de cobre u otro material sujeto al suelo, en la parte de arriba de dicho cilindro se coloca un embudo que colecta y dirige la precipitación. La precipitación cae en unas pequeños cubos que se encuentran balanceados a un columpio o balancín. Estos cubos están calibrados para que después de que cierta cantidad de lluvia vuelquen, esta volcadura cierra momentáneamente un interruptor que sirve para poder detectar si un evento ocurrió. Una fotografía del pluviómetro utilizado se muestra en la figura 3.10.

Fig. 3.10. Fotografía del pluviómetro usado.

Para detectar el evento se introduce una resitencia conectada en un extremo a una fuente de tensión y en el otro extremo a una terminal del interruptor, en el momento en que ocurra un volcado, el circuito se cierra y la tensión de entrada se muestra en la terminal de salida del interruptor. Realizando el

procedimiento anterior, el pluviómetro de balancín se convierte en un sensor que convierte una señal mecánica a una señal eléctrica, la cual puede ser registrada.

La desventaja de este tipo de pluviómetros es que suelen no ser tan precisos como los pluviómetros estándar, dado que la precipitación puede terminar antes de que el cubo tenga la cantidad necesaria para volcar, de esta manera cuando otro evento de precipitación comience puede sólo requerir de unas cuantas gotas para volcar el balancín y se registre un evento. También tienden a subestimar la cantidad de precipitación, particularmente en eventos con granizo y nieve.

3.2.2. Caracterización de la señal de los sensores

Por caracterización de la señal de los sensores nos referimos a determinar la salida esperada de los sensores, una vez que éstos son instalados para medir las variables de interés. Esta información nos será de importancia para diseñar el acondicionamiento requerido por las señales antes de ser digitalizada.

La sonda de temperatura

En el caso de la sonda de temperatura es necesario especificar el margen de mediciones en que la sonda estará sometida para de ahí determinar su salida.

Para nuestra aplicación decidimos aprovechar todo el potencial de la sonda de temperatura y utilizar el margen de mediciones que le es posible realizar, es decir de los -35 [°C] a los 50 [°C]. Utilizando la ecuación 3.5 obtenemos los valores de resistencia esperados para la sonda en el margen de temperatura mencionado, dichos valores se graficaron y se muestran en la figura 3.11.

Como se puede observar en la gráfica, la ecuación 3.5 describe un comportamiento no lineal sobre el margen de temperaturas medidos. Para el valor mínimo de temperatura (-35 [°C]) tenemos una resistencia del termistor de 2.826 [MΩ] y para el valor máximo (50 [°C]) de 33.124 [kΩ]. Ahora bien, mediante la ecuación 3.3 se determina la tensión de salida esperada por la sonda de temperatura en el margen de medición. Esta respuesta se puede apreciar en la gráfica de la figura 3.12. Se determinó que los valores de salida de la sonda van del margen de los 0.833 [mV] a los 9.0420 [mV] para los valores de temperatura mínimo y máximo respectivamente.

La variación de la señal de temperatura está dada por la constante de tiempo de la sonda. Este valor es el tiempo necesario para que la sonda de temperatura refleje a la salida el 63% de una nueva temperatura. En el caso

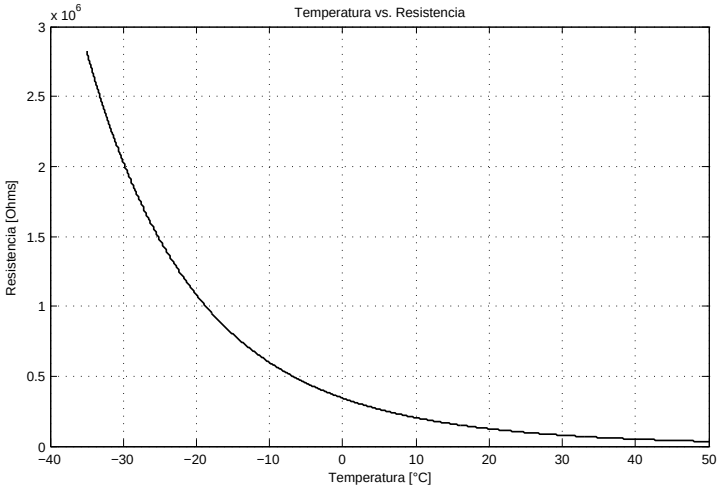


Fig. 3.11. Gráfica de la resistencia del termistor respecto a su temperatura para la sonda 107.

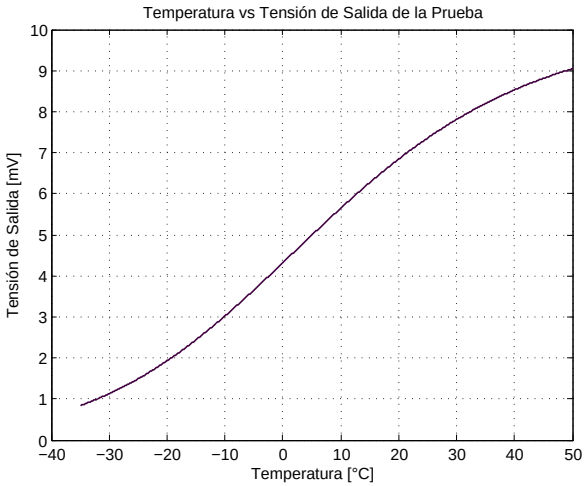


Fig. 3.12. Tensión de salida contra temperatura de la sonda 107.

de la sonda utilizada, este valor se encuentra entre los 30 y 60 segundos. Para obtener la frecuencia en la que esperamos varíe la señal de temperatura obtenemos el inverso de la constante de tiempo de la sonda, por lo tanto, la frecuencia de la señal de temperatura se encuentra entre los 0.033 [Hz] a los 0.017 [Hz]. Se considerará el valor más bajo de 0.017 [Hz].

En resumen, a la salida de la sonda de temperatura se espera una señal de tensión que se encuentra entre los 0.833 [mV] a los 9.0420 [mV] con una frecuencia de 0.017 [Hz].

El pluviómetro de balancín

El tipo de salida esperado para el pluviómetro depende totalmente de la entrada suministrada al interruptor del pluviómetro. Dado que se espera alimentar al circuito con una tensión de 3.3 [V], ésta será la tensión de salida del interruptor. En cuanto a la frecuencia, ésta no es constante y depende totalmente de la medición, de hecho la medición que se va a realizar del pluviómetro se considera un conteo de pulsos.

Una vez caracterizadas las señales se hablará del sistema de adquisición de las mismas.

3.2.3. Acondicionamiento de la señales de los sensores

El acondicionamiento de las señales de los sensores es el proceso por el cual se transforma la señal de estos para poderla hacer compatible con la entrada de los sistemas de adquisición. Este acondicionamiento es necesario para obtener una medición confiable de las señales de entrada.

La sonda de temperatura

Las características de la salida de la sonda de temperatura, como ya se vio, son las siguientes: una señal analógica que varía de los 0.833 [mV] a los 9.0420 [mV] con una frecuencia de 0.017 [Hz].

Según la configuración realizada para el convertidor A/D del microcontrolador, a su entrada éste acepta señales desde 0 [V] hasta los 2.56 [V] con una resolución de 2.5 $[\frac{\text{mV}}{\text{Bit}}]$ y muestrea las señales a 4.310 [kHz].

Si bien se puede apreciar que la frecuencia de muestreo del convertidor A/D es adecuada para la adquisición de la señal de temperatura, la tensión de salida de la misma no lo es, debido a que al ser digitalizada por el convertidor A/D se tendría una resolución pésima, en términos de $\frac{^{\circ}\text{C}}{\text{Bit}}$.

Para hacer frente a lo anterior se requiere de amplificar la señal de salida de la sonda de temperatura. Con el fin de tratar de obtener una resolución máxima, la amplificación debe de ser igual a la relación $\frac{2.56[\text{ V}]}{9.20268[\text{ mV}]}$ la cual

arroja una amplificación de 283.12 que es redondea a 283. Con esta amplificación, se obtiene una señal de la sonda de temperatura que varía de los 0.2357 [V] a los 2.559 [V].

Empleando el método anterior, existe una pérdida de resolución partiendo del hecho de que aunque es posible digitalizar valores por abajo de 0.2757 [V], éstos nunca estaran presentes a la entrada del convertidor A/D. Una mejor aproximación consideraría en restar el *offset* de la señal de temperatura antes de amplificarla, para asegurar que el margen de entrada de tensiones que representan una temperatura varíe desde 0 [V] hasta 2.56[V]. En este proyecto no se consideró realizar esta adecuación dado que, como se verá más adelante, no ofrece mejoras significativas en la resolución de salida del *datalogger* y representa el uso de un mayor número de componentes.

La ganancia calculada se implementó mediante el circuito amplificador no inversor mostrado en la figura 2.10, donde la ganancia de éste está dada a partir de la ecuación 2.16. Proponiendo un valor de resistencia para R_1 de 220 [kΩ] y una ganancia (G) de 283, el valor de R_2 se calcula mediante la ecuación 3.6.

$$R_2 = \frac{R_1}{G-1} = \frac{220[\ \mathrm{k\Omega}]}{283-1} = 780.14[\ \Omega] \tag{3.6}$$

Dado que 780.14 [Ω] no es un valor comercial de resistencia, se selecciona una resistencia de 680 [Ω] más un potenciómetro de precisión de 220 [Ω], lo cual nos dará un margen de ganancias desde los 324.52 a los 244.44. El uso de un potenciómetro permitirá hacer un ajuste fino de la ganancia de la señal de temperatura.

Una vez que se ha establecido la ganancia y el procedimiento para implementarse, es necesario conocer cuál será la resolución de salida del sistema después de esta amplificación. Con una ganancia de la señal de 283, el intervalo de valores binarios de salida posibles del convertidor A/D serían de $5E_H$ (0.2357 [V]) a $3FF_H$ (2.559 [V]). Utilizando este intervalo, obtendremos la resolución de salida en términos de temperatura una vez hecha la digitalización, para ello seguiremos el siguiente procedimiento.

- Primeramente, del intervalo de valores esperados por el convertidor A/D, obtenemos los valores de tensión representado por éstos mediante la ecuación 3.1.

- Los valores de tensión obtenidos se dividen entre la amplificación dada a la señal (en nuestro caso 283).

- Se le aplica la ecuación 3.3, que relaciona la salida de la sonda de temperatura con respecto a la resitencia del termistor, para calcular el valor de resitencia de ésta en los valores discretos de salida del convertidor A/D.
- Mediante la ecuación de Steinhart–Hart (ecuación 3.4), se obtienen los valores de temperatura de los valores de resistencia calculados en el punto anterior.
- Finalmente, se obtiene la diferencia entre los valores continuos de temperatura. Esta diferencia es la resolución del sistema de instrumentación y su comportamiento se observa en la figura 3.13.

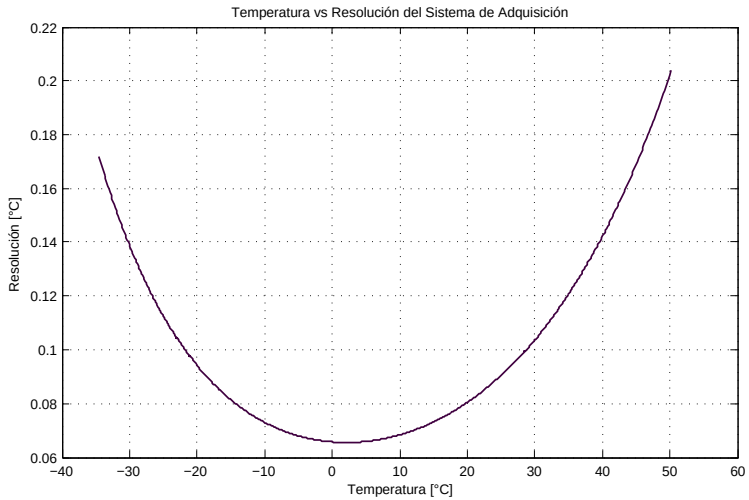


Fig. 3.13. Gráfica de la resolución de salida esperada.

Como se puede observar en la gráfica, la resolución esperada por el sistema no es lineal debido a que la respuesta de la sonda de temperatura usada no lo es. La resolución promedio en el intervalo es de 0.0914 [°C], con una resolución máxima de 0.2038 en los 50 [°C] y una mínima de 0.0655 [°C] en el intervalo de los 1.9578 [°C] a los 2.2856 [°C].

En caso de haber implementado un acondicionamiento para eliminar el *offset* de la sonda de temperatura, el procedimiento que se hubiera seguido para obtener la resolución esperada por el *datalogger* sería el siguiente.

- Partiendo de la tensión de salida esperada por la sonda para el rango de temperatura medible por ésta (0.833 [mV] y 9.0420 [mV] para -35 [°C] y 50 [°C] respectivamente), eliminar mediante un circuito restador el *offset* de 0.833 [mV].

- Del nuevo margen de salida obtenido 0 [mV] para -35 [°C] y 8.209 [mV] para 50 [°C], calcular la ganancia más adecuada para cubrir el intervalo. Esta ganancia está dada por la relación $\frac{2.56[\text{ V}]}{8.209[\text{ mV}]}$.

- La ganacia resultante de 311.85 puede ser redondeada a 311 y con ésta obtener el margen de tensión esperado a la entrada del convertidor A/D, una vez que la señal fuera amplificada. Este margen de tensión sería de los 0 [V] a los 2.55 [V].

- Una vez obtenido este margen se puede utilizar el procedimiento anteriormente descrito para obtener la resolución esperada por el *datalogger*, a este procedimiento sólo se le debe hacer la adecuación para que, una vez recuperada la señal de la sonda, sumarle el *offset* eliminado.

La gráfica que muestra la resolución que se hubiera esperado obtener por medio del *datalogger*, si el *offset* de la señal de temperatura hubierá sido eliminado, se puede ver en la figura 3.14.

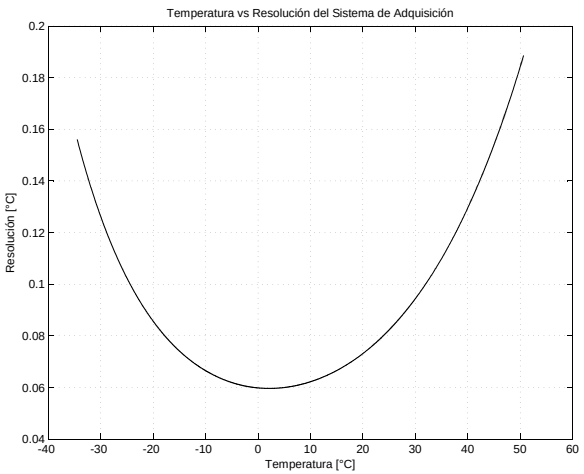


Fig. 3.14. Gráfica de la resolución de salida esperada sin *offset*.

Eliminando el *offset* de la señal de temperatura el valor de resolución promedio es de 0.834 [°C], la resolución máxima es de 0.1886 [°C] a los 50 [°C] y una mínima de 0.0596 [°C] en el intervalo entre 1.8366 [°C] y los 2.3774 [°C]. Aunque se aprecia que se puede obtener una mejora en resolución del *datalogger*, en caso de que el *offset* sea eliminado, debido a que este acondicionamiento requiere de un mayor número de componentes y que la ganancia de resolución no es significativa, se decidió no implementarla.

Una vez especificado el método para adecuar la amplitud de la salida de la sonda, adicionalmente se requiere eliminar el ruido de la señal para disminuir el efecto *aliasing* una vez que esta señal sea digitalizada. Para realizar lo anterior se propusó el diseño de un filtro paso bajas con aproximación de Butterworth como filtro *antialiasing*.

Los parámetros del filtro se consideran de la siguiente forma:

$$\begin{aligned} A_p &= 1 \text{ [dB]} \\ f_s &= 2.05 \text{ [kHz]} \\ A_s &= 62 \text{ [dB]} \\ f_p &= 10 \text{ [Hz]} \end{aligned}$$

La frecuencia de supresión del filtro es igual a la mitad de la frecuencia de muestreo del convertidor A/D (4.10 [kHz]). Para determinar la atenuación requerida en la banda de rechazo del filtro hacemos uso de la razón señal a ruido del convertidor A/D que se calcula a través de la ecuación 2.3, haciendo B igual a 10, el número de bits del convertidor A/D usado.

$$SNR = (1.763 + 6.02(10))[dB] = 62[dB]$$

Finalmente se propone que la atenuación en la banda de paso sea igual a 1 [dB] y dado que la señal de temperatura es basicamente DC se propone una frecuencia de corte del filtro sea de 10 [Hz].

Siguiendo el proceso para el diseño de un filtro Butterworth descrito en el capítulo anterior, primeramente calculamos el valor de ϵ_1 con la ecuación 2.10.

$$\epsilon_1 = \sqrt{10^{A_{máx}/10} - 1} = \sqrt{10^{0.1(1[\text{ dB}])} - 1} = 0.5088$$

Con la ecuación 2.11 calculamos ϵ_2 para la amplitud de la banda de supresión.

$$\epsilon_2 = \sqrt{10^{A_{mín}/10} - 1} = \sqrt{10^{0.1(62[\text{ dB}])} - 1} = 1258.92$$

Por medio de la ecuación 2.9 obtenemos el orden del filtro Butterworth.

$$n_B = \frac{\log {}^{\epsilon_2}/_{\epsilon_1}}{\log {}^{f_s}/_{f_p}} = \frac{\log {}^{1258.92}/_{0.5088}}{\log {}^{2050}/_{10}} = 1.47$$

Empleando el valor entero más próximo hacia arriba usaremos $n_B = 2$. Una vez conocido el orden del filtro, utilizamos la figura 2.5 para determinar los modos naturales del filtro. Encontramos que estos son:

$$\begin{aligned} p_1 &= \omega_0(-\cos 45^\circ + j \sin 45^\circ) = \omega_0(-0.7071 + j0.7071) \\ p_2 &= \omega_0(-\cos 45^\circ - j \sin 45^\circ) = \omega_0(-0.7071 - j0.7071) \end{aligned}$$

Finalmente combinando ambos polos en la expresión del filtro (ecuación 2.12) y haciendo la ganancia (K) unitaria obtenemos:

$$H(s) = \frac{\omega_0^N}{s^2 + 1.4242\omega_0 + \omega_0^2}$$

donde:

$$\omega_0 = \omega_p \left(\frac{1}{\epsilon_1}\right)^{1/N} = 2\pi \times 10 \text{ Hz} \times \left(\frac{1}{0.5088}\right)^{0.5} = 88.086$$

Con lo que finalmente, la función de transferencia del filtro queda como:

$$H(s) = \frac{7759.143}{s^2 + 125.452s + 7759.143} \tag{3.7}$$

Una vez obtenida la función de transferencia del filtro, se procede con su implementación a través de una topología Sallen-Key que se mostró en el capítulo anterior.

Partiendo de la función de transferencia del circuito (ecuación 2.15), se reescribe de la siguiente forma:

$$H(s) = \frac{1}{1 + RC(m+1)s + mnR^2C^2s^2} \tag{3.8}$$

donde:

$R_1 = mR$
$R_2 = R$
$C_1 = nC$
$C_2 = C$

De la ecuación 3.7 se normaliza para dejarla darle la forma de la ecuación 3.8

$$H(s) = \frac{1}{1.2888 \times 10^{-4}s^2 + 0.0167s + 1} \tag{3.9}$$

Igualando las ecuaciones obtenemos que el término $RC(m+1) = 0.01617$, donde proponiendo $R = 4.7[\ \text{k}\Omega]$ y $C = 2.2[\ \mu\text{F}]$ obtenemos m:

$$m = \frac{0.01617}{(4.7[\ \text{k}\Omega])(2.2[\ \mu\text{F}])} - 1 = 0.5638$$

a partir del valor de m obtenemos n, partiendo de igualar el término $mnR^2C^2 = 1.2888 \times 10^{-4}$

$$n = \frac{1.2888 \times 10^{-4}}{mR^2C^2} = \frac{1.2888 \times 10^{-4}}{0.5638 \times (4700[\ \Omega])^2 \times (2.2[\ \mu\text{F}])^2} = 2.1381$$

Por último a partir del valor de n y m calculamos C_1 y R_1:

$$C_1 = nC = (2.1381)(2.2[\ \mu\text{F}]) = 4.703[\ \mu\text{F}]$$
$$R_1 = mR = (0.5638)(4.7[\ \text{k}\Omega]) = 2.649[\ \text{k}\Omega]$$

Con lo que obtenemos todos los valores de los componentes del filtro Sallen-Key. Ajustando estos valores a componentes comerciales finalmente obtenemos que:

$C_1 = 4.7\ [\mu\text{F}]$
$R_1 = 2.7\ [\text{k}\Omega]$
$C_2 = 2.2\ [\mu\text{F}]$
$R_2 = 4.7\ [\text{k}\Omega]$

El circuito completo, implementado con un amplificador operacional uA741 y componentes con valores comerciales, se puede apreciar en la figura 3.15.

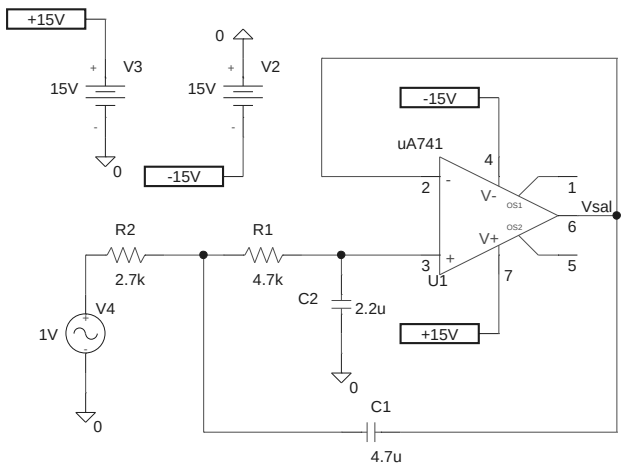


Fig. 3.15. Implementación del filtro Butterworth.

Aunque el amplificador uA741 no se utilizó en la implementación del filtro, dado que es el único amplificador disponible en el simulador usado y con el fin de dar una idea de la respuesta del filtro, este circuito se utilizó para hacer una simulación con barrido de A/C, dando como resultado la gáfica de la figura 3.16. Esta gráfica muestra la variación de la amplitud de la señal

de salida del filtro a diferentes frecuencias. Podemos apreciar en la simulación que a la frecuencia de corte de 10 [Hz] el filtro atenua la señal de entrada 1.068 [dB] ($20 \log \frac{0.884[V]}{1[V]}$), esta atenuación tiene un error de 6.8% con respecto a la atenuación definida, pero, a pesar de ello, se considera que el filtro es adecuado para ser implementado.

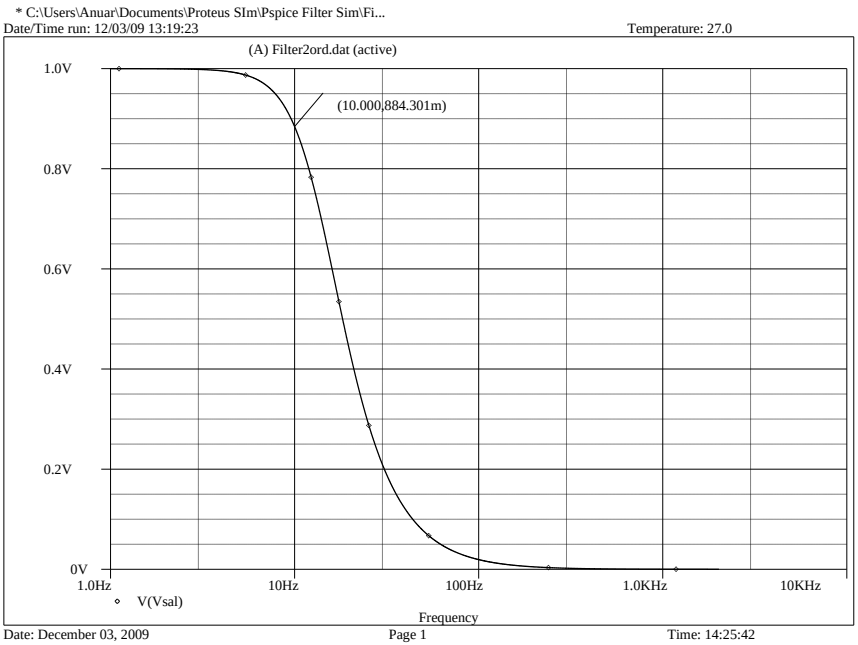


Fig. 3.16. Simulación del filtro Butterworth.

Como ya se había mencionado, la alimentación de la sonda de temperatura se realizó a través de una tensión de referencia. En nuestro caso se empleó la tensión de referencia interna del microcontrolador. Dado que esta salida no está diseñada para alimentar cargas, se empleó un amplificador operacional en configuración *buffer* para darle "potencia" y de esa manera poder alimentar la sonda de temperatura. La corriente necesaria a la salida del amplificador se encuentra dado por la carga que representa la sonda de temperatura, dicha carga está dada por el arreglo resistivo de la sonda de temperatura. En el peor de los casos la resistencia total de la sonda es de un poco menos de 1 [kΩ], siendo alimentado con una fuente de tensión de 2.56 [V], se obtiene que la corriente que circula por ella es de aproximadamente 2.56 [mA].

La selección del amplificador operacional debe contemplar caractéristicas para implementar tanto el filto, el circuito *buffer* y el circuito amplificador, para ello tomaremos los siguientes parámetros de selección:

- El tipo de alimentación
- El *gain bandwidth product* (GBWP)
- El *slew rate*
- La máxima corriente de salida

Dado que no se cuenta con una alimentación negativa en el circuito, es necesario considerar un amplificador con alimentación unipolar, lo cual se convierte en el primer parámetro de selección. Adicionalmente ya hemos calculado que la corriente mínima de salida que debe proporcionar el amplificador es de 2.56 [mA]. El GBWP necesario para un filtro con ganancia unitaria se encuentra dado por la ecuación 3.10

$$\text{GBWP}_{mín} = 100 f_c \tag{3.10}$$

donde f_c es la frecuencia de corte del filtro en nuestro caso 13.7 [Hz], por lo que se requiere de un amplificador cuya GBWP es mayor o igual a:

$$\text{GBWP}_{mín} = 100 \times 13.7[\text{ Hz}] = 1.37[\text{ kHz}]$$

El *slew rate* que el amplificador debe poseer está dado por la ecuación 3.11

$$Slew\ Rate \geq 2\pi V_{salPP} f_c \tag{3.11}$$

donde f_c es la frecuencia de corte del amplificador y V_{salPP} es la variación pico a pico esperada a la salida del amplificador abajo de f_c. En nuestro caso esta variación es mínima, dado que la razón de cambio de la señal de temperatura es muy bajo, pero utilizaremos 0.1 [V] para realizar el cálculo. Tomando esta consideración el *slew rate* requerido es igual a:

$$Slew\ Rate \geq 2\pi \times 0.1[V] \times 13.7[\text{ Hz}] = 8.6[\text{ V}/s]$$

Como podemos ver, dado que la frecuencia de corte es muy pequeña y la potencia de salida necesaria también es pequeña, no se requiere un amplificador operación con características muy específicas y podría haberse usado para su implementación un amplificador operacional de propósito general. Sin embargo, dada que la señal proveniente de la sonda de temperatura está muy cercana al nivel de tierra, nivel con el que se pretende polarizar el amplificador operacional, fue necesario el uso de un amplificador operacional con salidas y

entradas *rail to rail*. Esta característica permite que tanto las entradas como las salidas de este amplificador puedan tener niveles muy cercanos a las tensiones de polarización del amplificador operacional. Dado que se requieren de tres amplificadores operacionales para realizar la etapa de acondicionamiento de la sonda de temperatura, se utilizaron el circuito integrado MAX4168 que incorpora dos amplificadores operacionales y el circuito integrado MAX4166 que incorpora uno. Adicionalmente a su alimentación unipolar y salidas y entradas *rail to rail*, los amplificadores operacionales en estos circuitos integrados tienen las siguientes características:

$$\begin{aligned} SlewRate &= 2\ [\mathrm{V}/\mu\mathrm{s}] \\ \mathrm{GBWP} &= 5\ [\mathrm{MHz}] \\ I_{sal\ máx} &= 80\ [\mathrm{mA}] \end{aligned}$$

A simple vista se puede apreciar que las caractéristicas de los amplificadores superan los requerimientos de los circuitos por lo que se concluyó que el amplificador es adecuado.

Con el fin de asegurar un bajo consumo de energía del *datalogger* y evitar el ruido proveniente de los circuitos digitales, se realizaron consideraciones para la alimentación del amplificador operacional:

En primer lugar se realizó un circuito de tierra independiente y se unió con la tierra digital en un solo punto, además la alimentación se filtró mediante un filtro LC paso bajas pasivo. Este circuito se muestra en la figura 3.17.

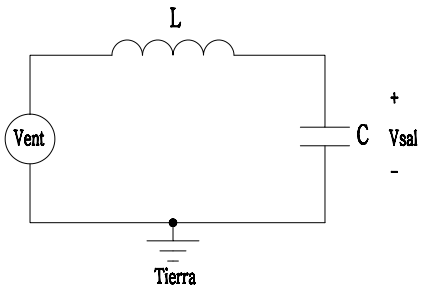


Fig. 3.17. Filtro pasivo LC.

Este filtro tiene una frecuencia de corte dada por la ecuación ecuación 3.12.

$$f_0 = \frac{1}{2\pi \times \sqrt{LC}} \tag{3.12}$$

Sustituyendo los valores de $L = 10[\ \mu\mathrm{F}]$ y $C = 100[\ \mathrm{nF}]$, con los que el fabricante del microcontrolador sugiere implementar este filtro, encontramos que la frecuencia de corte es de:

$$f_0 = \frac{1}{2\pi \times \sqrt{10[\ \mu\text{F}] \times 100[\ \text{nF}]}} = 159.15[\ \text{kHz}]$$

Dado que es un filtro que está diseñado para eliminar el ruido proveniente de los circuitos digitales con una frecuencia, en nuestro caso, de 4 [MHz] se determina, que si se considera una atenuación de 40 [dB/decada] para este filtro, éste es capaz de atenuar una señal de 4 [MHz] 56 [dB] lo cual se considera adecuado para la aplicación.

Adicionalmente, dado que durante el modo de adquisición del *datalogger* es necesario que el circuito de acondicionamiento del la sonda de temperatura opere en todo momento y con el fin de disminuir el consumo que éste representa, se decidió hacer uso de la terminal de encendido y apagado que los amplificadores operacionales usados incorporan. Haciendo uso de esta terminal se puede disminuir el consumo de energía de cada amplificador de 1.2 [mA] a 38 [μA].

Los circuitos involucrados en el acondicionamiento de la señal proveniente de la sonda de temperatura se muestran en la figura 3.18. Como se observa, la conexión de la sonda de temperatura se realiza a través de un conector de donde se alimenta (Vprueba) y se obtienen la señal de temperatura (Señal_Temp). La alimentación se realiza a través del circuito *buffer* implementado con el amplificador operacional MAX4166 y que tiene por entrada la tensión de referencia del microcontrolador (UC_Vref). Posteriormente, la señal de temperatura se amplifica mediante el circuito amplificador no inversor y se filtra mediante el filtro Butterworth de segundo orden, ambos circuitos son implementados mediante el MAX4168. Después de este proceso la señal se encuentra lista para ser adquirida, por lo que se conecta a través del puerto uC_ADC a la terminal del convertidor A/D del microcontrolador. Las terminales de control de encendido y apagado del los amplificadores operacionales son unidas y llevadas a una terminal del microcontrolador TH_SD.

El pluviómetro

La señal proveniente del pluviómetro debe ser acondicionada, para evitar posibles problemas con el ruido de la señal que pudieran ocasionar que el microcontrolador detecte pulsos en falso o que, una vez que éstos se generen, sean detectados en más de una ocasión. Otro efecto que se desea evitar es que una tensión que provenga de la línea de señal del pluviómetro sea mayor a la que el sistema pueda manejar, por lo que está es también otra consideración de diseño. Si bien el efecto de rebote mecánico no es un problema grave para

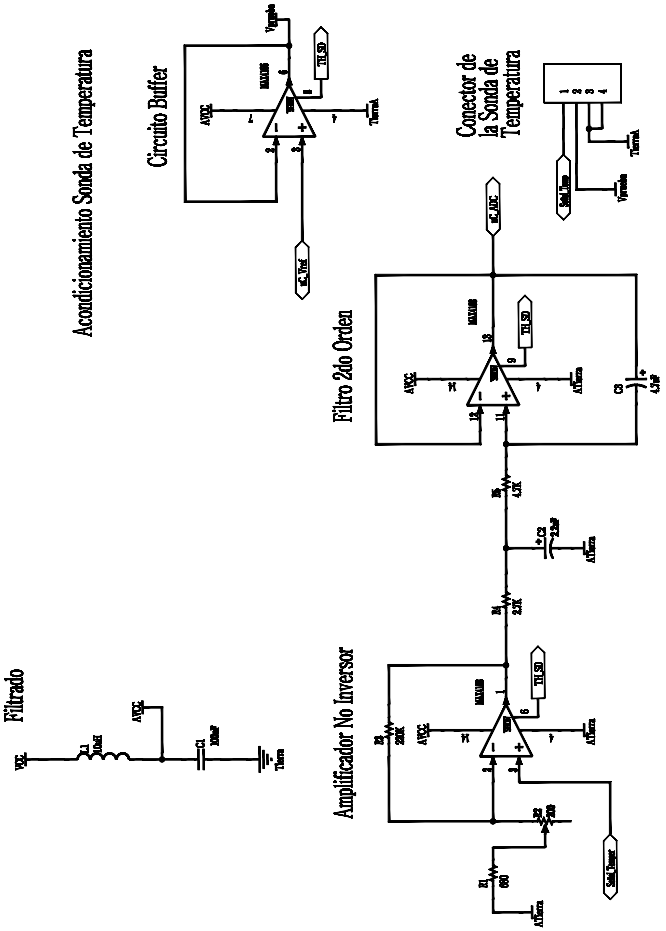


Fig. 3.18. Circuito de acondicionamiento de la señal de temperatura.

este tipo de pluviómetro, dado que su interruptor es de mercurio, se deben de considerar otras fuentes de ruido, como el ambiental.

El circuito que se diseñó para disminuir tanto el problema de ruido en la señal como de sobretensiones de la línea, es el mostrado en la figura 3.19.

En el circuito de la figura 3.19, un filtro pasivo RC paso bajas es implementado con la resistencia R_2 y el capacitor C_1, la frecuencia de corte del filtro RC pasivo se encuentra dado por la ecuación 3.13.

$$f_0 = \frac{1}{2\pi RC} \tag{3.13}$$

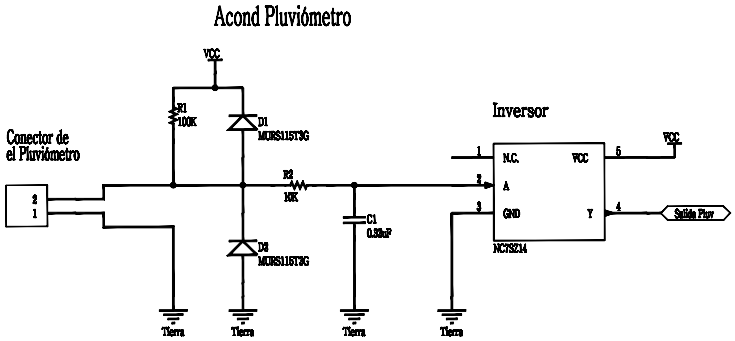


Fig. 3.19. Circuito de acondicionamiento de la señal del pluviómetro

Sustituyendo el valor de los componentes usados obtenemos que la frecuencia de corte, para el filtro implementado es igual a:

$$f_0 = \frac{1}{2\pi \times 10[\text{ k}\Omega] \times 0.33[\ \mu\text{F}]} = 48.23[\text{ Hz}]$$

La compuerta inversora con Schmitt Trigger NC7SZ14 posee, a una temperatura de operación de 25 [°C] y una tensión de 3.0 [V], con las siguientes características: una tensión de umbral positivo de 1.75 [V], una tensión de umbral negativo de 1 [V] y una tensión de histéresis de 0.75 [V].

Combinados el filtro RC y la compuerta inversora NC7SZ14 ayudan a evitar el ruido presente en la línea del pluviómetro. Dado que esta combinanción de filtro RC y de compuerta con Schmitt Trigger es normalmente implementado para evitar el efecto rebote del interruptor, se considera que el circuito implementado es más que adecuado para evitar disparos en falso causados por ruido ambiental.

Los diodos en el circuito de la figura 3.19 sirven de protección por sobretensiones, en caso de que la tensión de linea supere V_{cc} más 0.58 [V] (la tensión de encendido del diodo) el diodo D_1 entra en conducción y dirige la corriente a la fuente. Si por el contrario, la tensión de línea baja a -0.58 [V], el diodo D_2 entra en conducción dirigiendo la corriente también a la fuente. Los diodos elegidos MURS115T3G tienen una capacidad de conducción de 1 [A] y se encuentran en un empaque en montaje superficial por lo que fueron elegidos para desempeñar esta función.

La resistencia R_1 es una resistencia de *pull-up* que proporciona un nivel de tensión a la entrada al interruptor. A la salida de la compuerta inversora

(Salida_Pluv) se obtiene una señal continua de tierra y pulsos en alto en cada cierre del interruptor.

3.3. El reloj en tiempo real

Un reloj en tiempo real es un tipo de dispositivo integrado que mantiene un registro de la hora y fecha actual. Los RTR's están presentes en cualquier dispositivo que necesiten mantener la fecha y hora de manera confiable, como lo son PC's, servidores, sistemas embebidos, etc.

Si bien mantener la hora y la fecha puede ser realizado sin la necesidad de un RTR, el uso de uno nos permite:

- Un menor consumo de energía
- Liberar al microcontrolador de la tarea de manter la hora y fecha
- Obtener mayor precisión que con otros métodos

Para realizar su función los RTR requieren de una señal de reloj. Normalmente esta fuente proviene de un cristal oscilador, si bien hay algunos que pueden utilizar la frecuencia de la linea de alimentación. La frecuencia a la que oscila este cristal es de 32.768kHz, la cual es la misma frecuencia que utilizan los relojes de cuarzo, esto es debido a que esta frecuencia es exactamente 2^{15} ciclos por segundo, la cual es una razón conveniente para un contador binario.

El modelo de RTR usado en este *datalogger* fue el circuito integrado DS1337 de la empresa Maxim Dallas. Éste, a demás de poseer las funciones básicas de un RTR, incorpora dos alarmas programables con salidas independientes. Esta función nos permitirá que sea el RTR el encargado de llevar el período de muestreo de la señal de temperatura. Para su funcionamiento el DS1337 sólo requiere de ser alimentado y conectado al *bus* I^2C. Dado que sus salidas son del tipo colector abierto, se requieren de resistencias de *pull–up* para que éstas operen de manera adecuada.

En la figura 3.20 se muestra un diagrama de la conexión del DS1337. En éste se pueden apreciar tanto las resistencias de *pull–up* en las terminales del *bus* I^2C (SCL y SDA), como en las terminales de las alarmas (INTB e INTA). Estas resistencias con un valor de 12 [kΩ] permiten al tener una alimentación

de 3.3 [V] un flujo de corriente de 275[μA], lo cual es muy adecuado para ser manejado por los puertos de E/S del microcontrolador y no desperdician tanta energía al llevar la señal de salida a tierra. El DS1337 incorpora terminales para conectar el cristal de 32.768 [kHz].

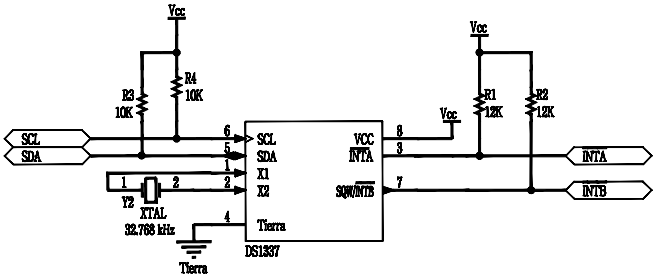


Fig. 3.20. Diagrama de conexión del RTR DS1337.

3.4. Las memorias EEPROM

Las memorias EEPROM usadas fueron las 24LC1025 de la compañía Microchip. Estas memorias tienen una capacidad de 1024 [kBits], lo cual representa 128 [kBytes]. La comunicación con ellas se realiza a través del *bus* I^2C y es posible conectar a éste hasta cuatro circuitos integrados de estas memorias, número que fue colocado en el *datalogger*. Las cuatro memorias usadas nos dan una capacidad de 512 [kBytes], memoria que el *datalogger* puede utilizar para el almacenamiento de los datos adquiridos.

En la figura 3.21 se muestra el diagrama de conexión de las memorias. Para funcionar, éstas sólo requieren ser alimentadas y conectadas al *bus* I^2C. Los circuitos integrados de memoria poseen dos terminales (E_0 y E_1) que deben ser conectadas a tierra o a V_{CC} para establecer su dirección dentro del *bus* I^2C . La terminal E_2 no es configurable y debe de ser conectada a V_{CC} para que la memoria pueda funcionar de manera correcta. Adicionalmente, las memorias cuentan con una terminal para habilitar o deshabilitar su operación de escritura (Modo), esta terminal puede ser controlada por medio del microcontrolador o interruptor mecánico, pero en esta aplicación dicha función no fue utilizada y la terminal fue conectada directamente a tierra para que la escritura de las memorias siempre se encontrara habilitada.

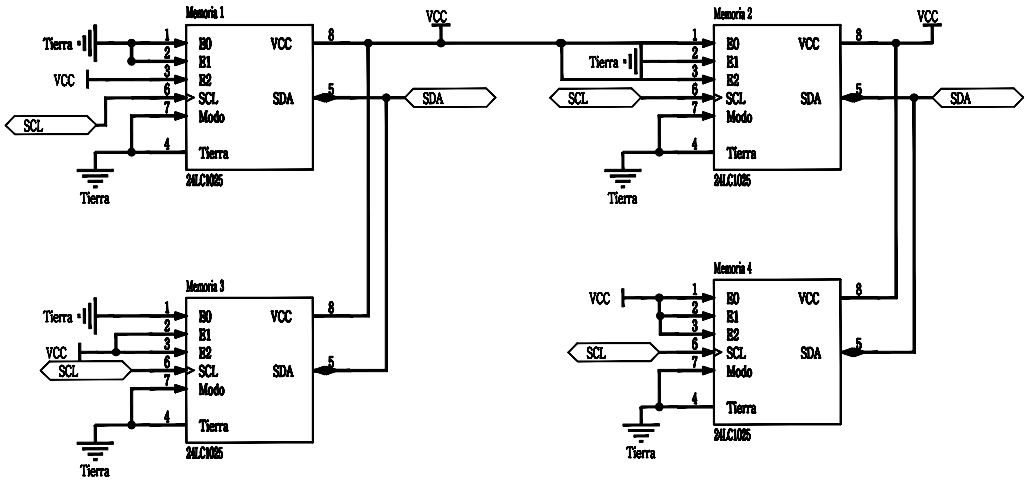


Fig. 3.21. Diagrama de conexión de las memorias EEPROM externas.

3.5. El transceptor RS-232

Debido a que las señales provenientes de la USART no cumplen con los requerimientos de tensión y polaridad del estándar RS-232, es necesario emplear un transeptor que haga dicha conversión.

Este transeptor funciona a través de dobladores de tensión con base en capacitores, por lo que para su funcionamiento éstos necesitan estar presentes. En el caso del proyecto en cuestión, se decidió utilizar el transceptor MAX3233 cuyos capacitores de carga se encuentran dentro del empaque, lo cual nos ayudó a ahorrar espacio en el PCB.

Adicionalmente a esto, este dispositivo tiene una terminal para su encendido y apagado, función que fue utilizada con miras en disminuir el consumo de energía del *datalogger*.

En la figura 3.22 se muestra el diagrama de conexión usado. Para funcionar el MAX3233 debe de ser alimentado y sus terminal de entrada deben ser conectadas a las terminales de la USART del microcontrolador (Tx y Rx). A la salida del MAX3233 se obtienen las señales listas para que, por medio de un conector DB9, éstas sean recibidas por la PC.

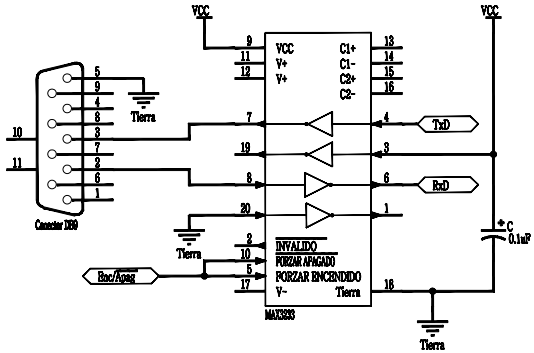


Fig. 3.22. Diagrama de conexión del MAX3233.

3.6. El transceptor USB

Debido a la complejidad en el manejo de tiempos del estándar USB es muy difícil que este estándar sea implementado mediante los puertos de un microcontrolador de propósito general. Para poder implementa este tipo de comunicación es necesario de un microcontrolador que incorpore un módulo de comunicación USB o utilizar un transceptor externo para convertir las señales de un módulo de comunicación, presente en el microcontrolador, a señales del estándar USB. En el mercado existen diversos dispositivos que realizan esta conversión de señales, entre ellos están: transceptores de UART a USB y transceptores de I^2C a USB. A través de este tipo de dispositivos se puede convertir una conexión inherente al microcontrolador en una señal USB. El proceso de enumeración y el manejo del protocolo son hechos automáticamente por la lógica del dispositivo, por lo que no es necesaria su implementación.

Debido a que el microcontrolador usado para el diseño del *datalogger* no incorpora un módulo de comunicación USB, se deció, para implementar el estándar USB, usar un transceptor USART–USB. Mediante el uso de este transceptor y un *driver* en la PC, se puede hacer una emulación de un puerto RS-232 en la computadora empleando una conexión USB. De esta manera, el dispositivo conectado a través de esta forma, aparece para el manejador de dispositivos de Windowns como un puerto COM en vez de un dispositivo USB.

Desde el punto de vista del *datalogger*, se utilizan las mismas instrucciones para hacer la comunicación USB que las que se utilizan para hacer la comunicación RS-232.

El transceptor utilizado en este proyecto fue el FT232RL de la compañia FTDI. Para funcionar, este dispositivo sólo requiere de ser conectado a las terminales de la USART del microcontrolador, unos cuantos componentes pasivos para filtrar las señales y de ser conectado al *bus* USB mediante un conector tipo B. No es necesario proporcionarle energía al circuito de manera interna, dado que éste se alimenta de la energía proporcionada por *host* USB. Los capacitores C_1 y C_2 y la ferrita sirven para suprimir el ruido proveniente de la señal de alimentación. El capacitor C_3 desacopla la salida del regular interno de 3.3 [V] que posee el circuito integrado.

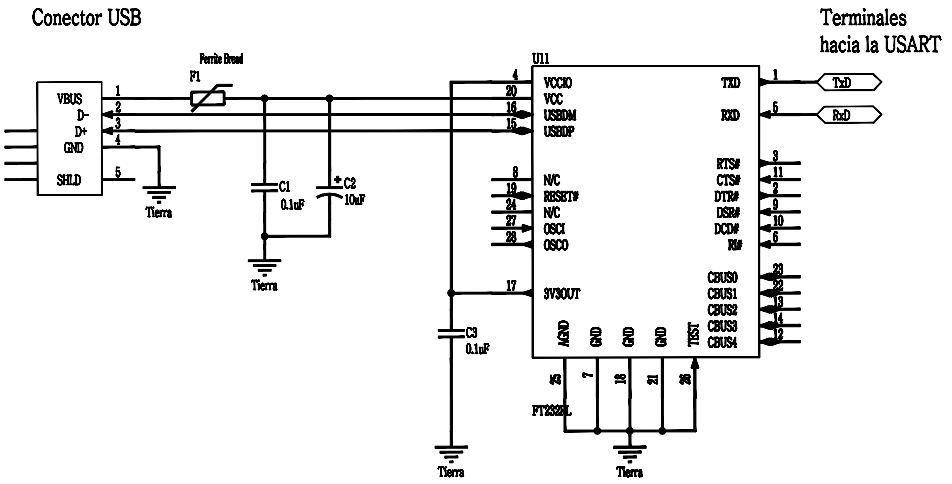


Fig. 3.23. Diagrama del transceptor USB.

3.7. La fuente de alimentación

La fuente de alimentación del *datalogger* se realizó con base en el MAX1675, el cual es un convertidor DC/DC de subida. Este circuito puede elevar una tensión de entrada desde 0.7 [V] hasta una tensión de salida (V_{sal}) configurable, desde los 2 [V] hasta los 5 [V] con una corriente máxima de 500 [mA].

En nuestro caso, dado que, se desea que el *datalogger* sea alimentado a través de una batería de 1.5 [V], este convertidor es ideal para nuestra aplicación.

En la figura 3.24 se aprecia la conexión del MAX1675 para una salida de 3.3 [V], donde V_{CC} es la tensión de salida del convertido y V_{bat} es la tensión proveniente de la batería. Para su funcionamiento, el MAX1675 requiere de muy pocos componentes, siendo el inductor el más importante. Para un funcionamiento eficiente, se debe asegurar que el inductor usado posea una baja resistencia en DC y que adicionalmente soporte la corriente de salida que se desea.

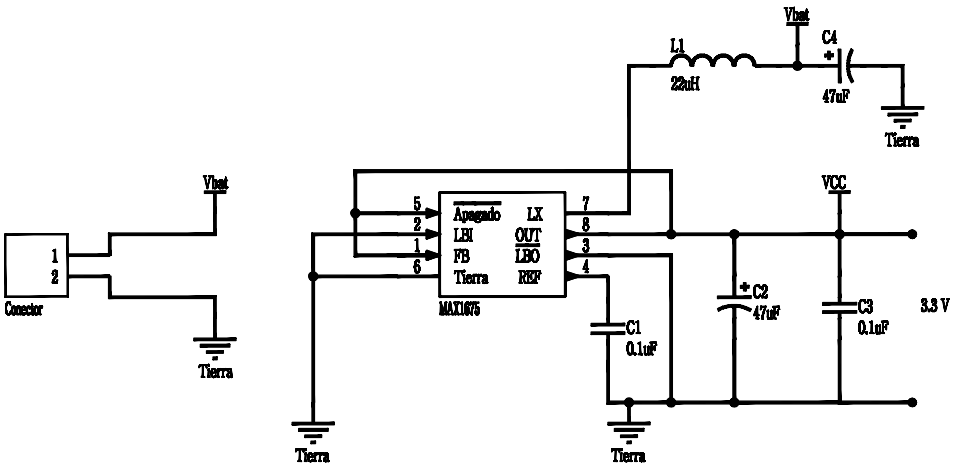


Fig. 3.24. Diagrama de la fuente de alimentación del *datalogger*.

3.8. Botones e indicadores

Con la finalidad de que el *datalogger* pueda tener una interacción con el usuario en sitio, se implementaron controles a través de botones de presión y de posición. De igual forma, para indicar acciones realizadas por el usuario, en el *datalogger* se colocó un LED que, haciendo usos de secuencias de parpadeo, puede indicarle rápidamente al usuario la operación que fue realizada por el *datalogger*, por ejemplo, el incio o paro de la adquisición.

El diagrama de conexión de los botones y el LED se puede observar en la figura 3.25. Tanto los botones como el LED están conectados directamente a terminales de E/S del microcontrolador de acuerdo a la tabla 3.1. La única excepción es el botón principal, que se conectó mediante una compuerta OR

a una de las terminales con interrupción external del microcontrolador. Esta compuerta se implementó de manera discreta con diodos. Por medio de esta compuerta OR, la señal del botón principal comparte la terminal de interrupción con una de las alarmas del reloj en tiempo real (INTB), esta interrupción también fue conectada a la terminal PD5 del microcontrolador con el fin de poder detectar que dispositivo ocasionó la interrupción.

Los botones están conectados a tierra y emplean la resistencia de *pull–up* interna de los puertos para evitar drenar corriente en exceso cuando éstos sean cerrados.

Botón	**Terminal del uC**
Botón de posición 1	PB0
Botón de posición 2	PB1
Botón de posición 3	PB3
Botón principal	NA
Botón secundario	PB4
LED	PD6

Tabla 3.1. Distribución de los botones y el LED en las terminales del uC.

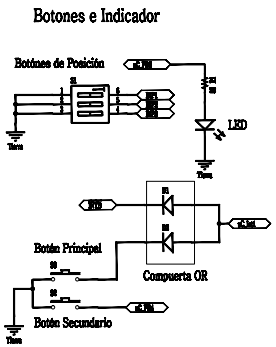


Fig. 3.25. Diagrama de conexión de los botones y el LED del *datalogger*.

3.9. Integración del hardware

Una vez que todos los componentes del *datalogger* fueron probados en tarjetas prototipo, la integración del hardware se realizó mediante la creación de

circuitos impresos por partes de los componentes usados. Las tarjetas impresas creadas fueron las siguientes:

- Una tarjeta principal que intergra el microcontrolados, el RTR y los botones e indicadores.
- Una tarjeta para agrupar las memorias EEPROM.
- Una tarjeta para realizar el acondicionamiento de la sonda de temperatura.
- Una tarjeta para el acondicionamiento del pluviómetro.
- Una tarjeta para la fuente de alimentacíón.
- Una tarjeta para el transceptor RS-232.
- Una tarjeta para la transceptor USB.

Todas las tarjetas se conectaron a la tarjeta principal del microcontrolador a tráves de cables planos, para integrar completamente el hardware del proyecto. La tarjeta principal es la que alimenta a los distintos módulos. Debido a que la tarjeta desarrollada encargada del acondicionamiento de la sonda de temperatura tuvo que ser adecuada, éste se muestra implementado en una tarjeta de prototipo. El circuito del *datalogger* completo se muestra en la fotografía de la figura 3.26.

Por último, en estos momentos se encuentra en desarrollo una tarjeta de circuito impreso que integra todos los elementos juntos. El diagrama de dicha tarjeta se muestra en las figuras 3.27 y 3.28. En esta tarjeta, la mayoría de los elemenetos usados, a excepción de algunos capacitores, son de montaje superficial y estos poseen una mayor integración. Por citar un ejemplo, el modelo de reloj en tiempo real usado en esta tarjeta incorpora el cristal dentro del empaque. Se espera que una vez que esta tarjeta se encuentre desarrollada, ésta se convierta en la tarjeta definitiva del *datalogger*.

Una vez que tenemos identificados todos los componentes de hardware del *datalogger*, en el siguiente capítulo estudiaremos la interacción que éstos llevan a cabo para realizar las funciones del *datalogger*.

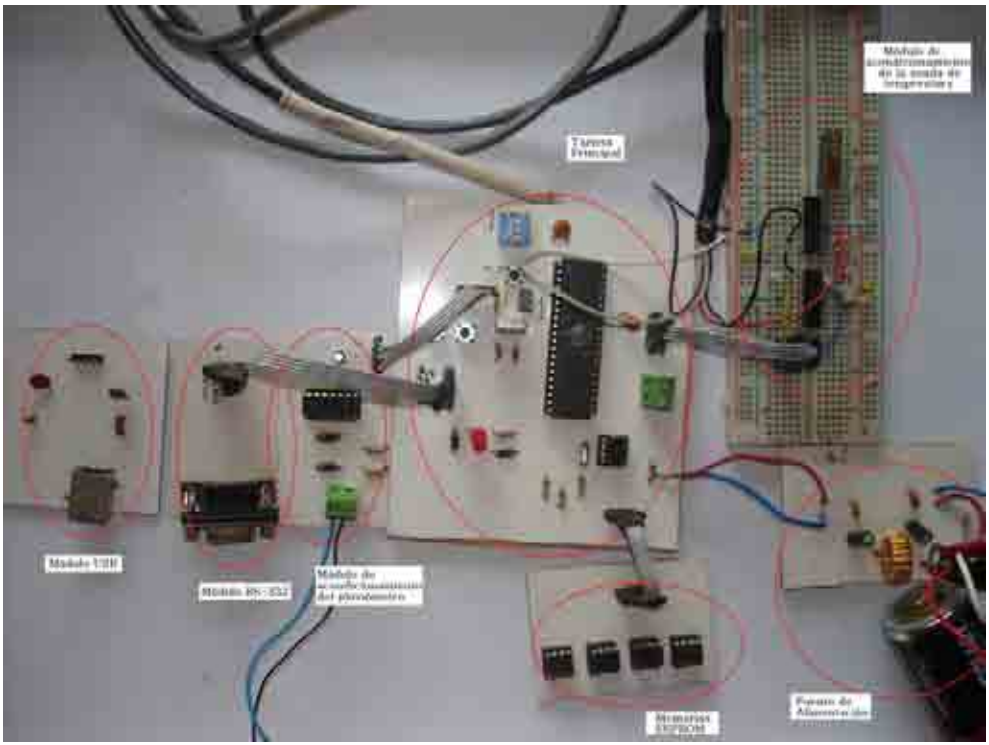


Fig. 3.26. Módulos de los circuitos del *datalogger* interconectados.

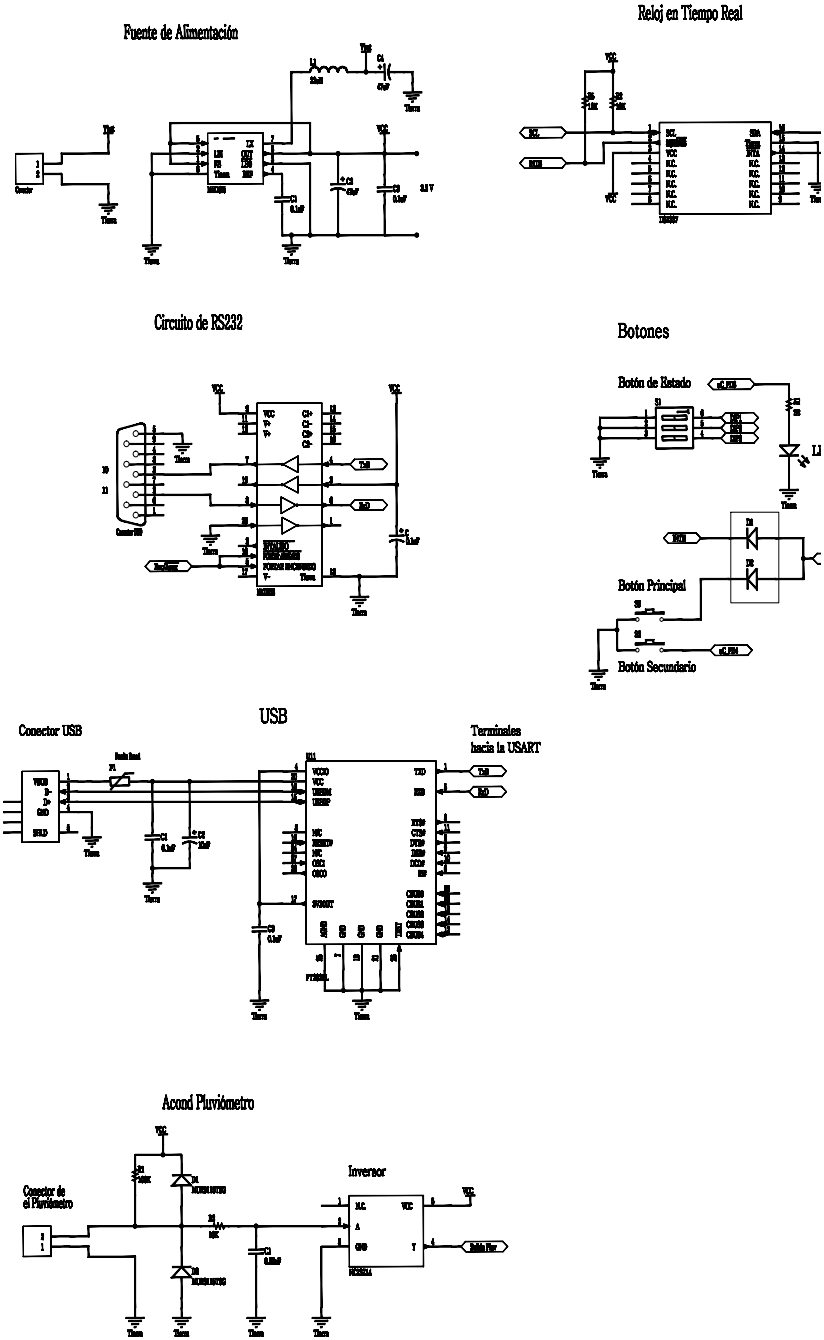


Fig. 3.27. Diagrama del *datalogger* parte 1.

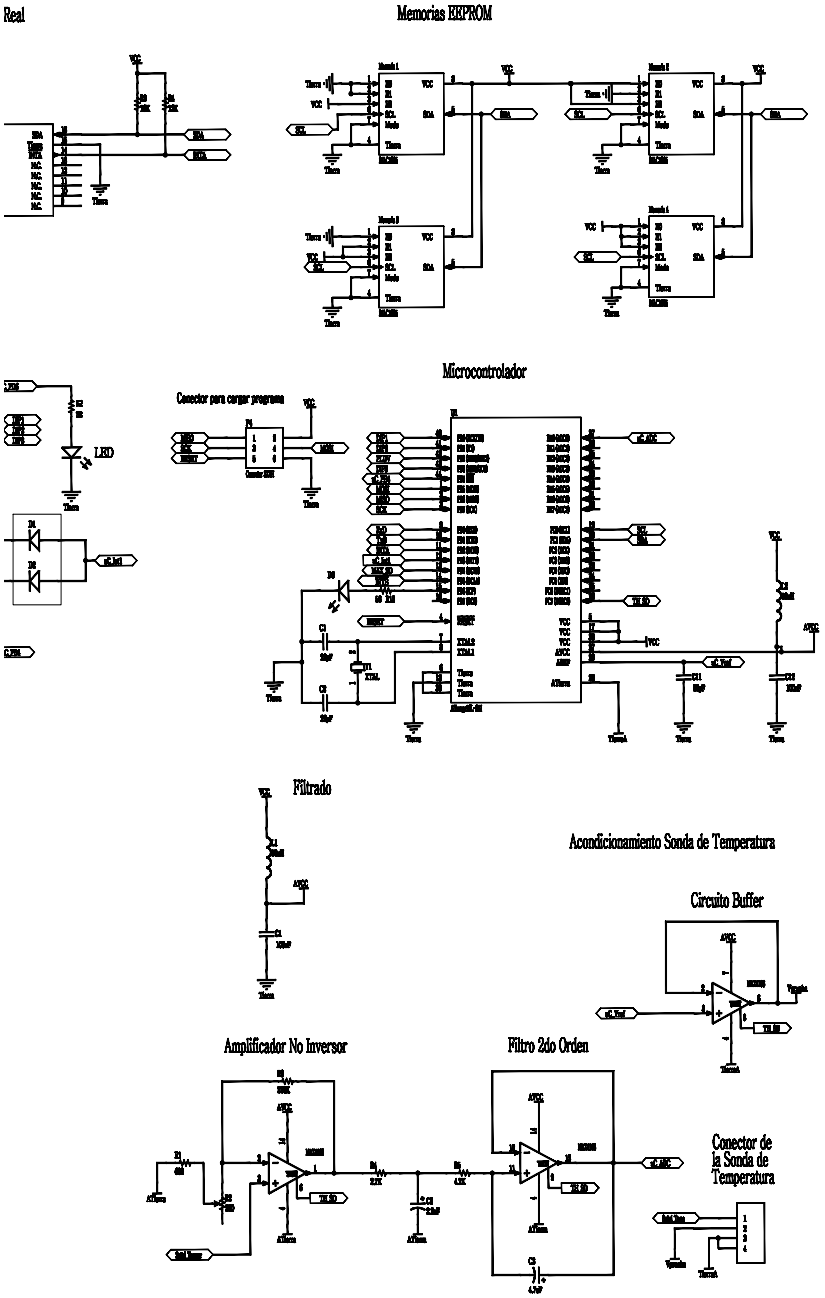


Fig. 3.28. Diagrama del *datalogger* parte 2.

Capítulo 4

Desarrollo del Programa del Microcontrolador

En este capítulo nos centraremos en la descripción de la interacción que tienen los componentes de hardware del *datalogger*. Toda esta interacción, como ya se mencionó, se realiza a través del programa del microcontrolador que, a partir de este momento, denominaremos *firmware*. El desarrollo del *firmware* del microcontrolador fue realizado a través del compilador AVR GCC, este compilador, de distribución libre, compila código programado en C y lo transforma en un código objeto que puede ser programado en el microcontrolador. La programación que se realizó fue hecha a partir de programar funciones simples, que posteriormente pudieran ser integradas en funciones más complejas hasta componer el programa principal, que realiza las funciones del sistema.

El diseño de esta parte del proyecto se dividió en las tres funciones principales que el *datalogger* debe realizar, y que son: adquisición de datos, guardado de éstos en memoria y la trasmisión de los mismos a una computadora personal (PC).

La adquisición de datos corresponde a la recepción, muestreo y tratamiento de la información proveniente de los sensores previo a ser guardados. La etapa de guardado corresponde al uso, manejo y organización de las memorias para permitir el almacenamiento de la información adquirida. Por último, la etapa de transmisión de datos corresponde a la interacción que se lleva a cabo entre el *datalogger* y la PC, para el envío de los datos guardados en memoria. Adicionalmente a estas etapas básicas, se deben considerar funciones adicionales que lleva a cabo el *datalogger* y que complementan su función principal, como son su calibración, configuración, etc.

En las siguientes secciones se estudiará de manera general la estructura del *firmware*, para posteriormente particularizar en cada una de las funciones que se implementaron en el *datalogger*.

4.1. Estructura del *firmware*

La estructura del *firmware* se desarrolló teniendo como segunda prioridad, sólo después de la realización de las funciones principales del *datalogger* anteriormente mencionadas, el garantizar un bajo consumo de energía del circuito, con el fin de lograr el mayor tiempo posible de autonomía en la adquisición de datos. Para realizar lo anterior es necesario considerar que el *datalogger* hará mediciones de variables cuya velocidad de cambio es muy lenta respecto a su velocidad de procesamiento. Lo anterior significa que el *datalogger*, de su tiempo total de funcionamiento, sólo estará trabajando de manera activa en pequeños intervalos mientras que, la mayor parte del tiempo, se encontrará en espera. La manera en que el *datalogger* permanece en espera es un componente esencial en el manejo eficiente de la cantidad de energía disponible por éste.

Los elementos de los que consta el *datalogger* y su consumo energético alimentados con 3.3 [V] se enlistan en la tabla 4.1.

Componente	**Consumo en operación** [mA]	**Consumo en espera** [μA]
Microcontrolador	5	<1
Circuito acond. sonda temp.	6.16	114
Circuito del pluv.	0.061	NA
Memorias EEPROM	20	20
Reloj en Tiempo Real	5	<1
Transceptor RS-232	5	<1
Transceptor USB	5	<1
LED	5	NA

Tabla 4.1. Consumo energético de los componentes del *datalogger*.

Con la información presentada en la tabla 4.1 nos damos cuenta que un buen diseño del *firmware* intentará que los componentes se encuentren en bajo consumo la mayor parte del tiempo. Tomando en cuenta esto, se buscó que los elementos que componen al hardware del *datalogger* incorporan medios para ser encendidos y apagados por medio del microcontrolador.

El microcontrolador es un elemento que se pone en estado de bajo consumo a través de su *firmware* pero, para sacarlo de este estado, se requiere que un dispositivo externo que así se lo indique. En el capítulo de generalidades vimos

que la manera en que se implementó la salida del microcontrolador de su estado de bajo consumo fue a través de sus tres interrupciones externas.

En el diagrama de flujo mostrado en la figura 4.1 se presenta la estructura general del *firmware*. La ejecución del microcontrolador comienza en el bloque de inicio una vez que el microcontrolador es energizado o cuando el botón de *reset* de éste es presionado. Antes de proseguir con la inicialización del sistema, el programa lee los botones de posición, con los cuales el usuario le indica al sistema que desea reiniciar parámetros o formatear la memoria de datos. Si alguna de estas funciones es seleccionada el *datalogger* las realiza y a continuación se mantiene en un ciclo infinito, con lo cual detiene su ejecución hasta que éste sea reinicializado. Lo anterior asegura que el *datalogger* no sea operado dejando estas opciones activadas.

Si ninguna de estas opciones fueron seleccionadas, el microcontrolador realiza un proceso de configuración del sistema. Esta configuración permite que el *datalogger* inicie sus componentes con valores y estados conocidos y cargue en memoria los parámetros de operación que quedaron guardados durante su última configuración. En caso de que el *datalogger* sea puesto a adquirir sin ser previamente configurado por medio de una computadora, realizará el proceso de adquisición siguiendo estos parámetros.

Una vez configurado el microcontrolador, éste activa sus interrupciones y entra en estado de bajo consumo. En este estado, el microcontrolador queda en espera de interrupciones externas que lo despierten y reanuden su operación.

Como ya se había mencionado, las terminales con interrupción externa del microcontrolador son tres y están divididas en dos con interrupción síncronas y una con interrupción asíncrona.

Para saber a qué señales conectar a estas terminales con interrupción, debemos regresarnos a las funciones que el *datalogger* realizará. En primer lugar, el *datalogger* deberá adquirir dos tipos de variables. Cada variable deberá poseer una señal que indique en qué momento ésta deba ser adquirida. Estas señales pueden ser conectadas a una interrupción externa del microcontrolador para que, una vez dicha señal esté presente, el microcontrolador reanude su operación y adquiera la variable. En segundo lugar, el *datalogger* requiere de una interacción con el usuario, interacción que no se puede dar si el microcontrolador se encuentra en bajo consumo, por lo que, con el fin de que el microcontrolador pueda ser despertado por el usuario, se conectó un botón a la última terminal con interrupción externa disponible.

Una función adicional que se le dio a la última terminal de interrupción ya mencionada, es la de servir como una señal para iniciar la adquisición de las variables meteorológicas de manera programada por tiempo, a través de

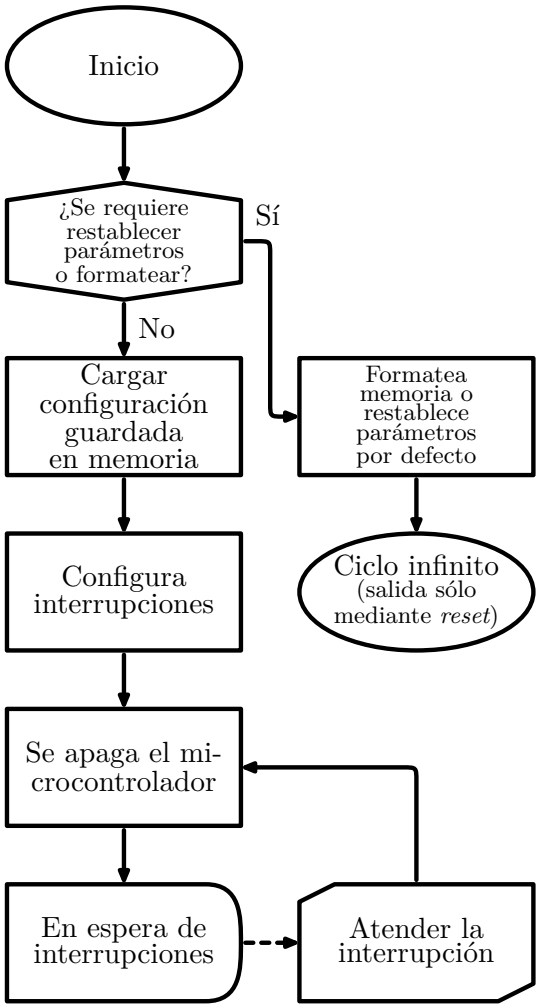


Fig. 4.1. Diagrama de operación del *datalogger*.

una señal de alarma del RTR. De esta manera el usuario puede indicarle al *datalogger* cuando comenzar con la adquisición de las variables. Dado que la señal del botón actívado por el usuario y la salida de la alarma del RTR son dos señales diferentes, se requiere realizar la conexión entre éstas y la terminal de la interrupción del microcontrolador a través de una compuerta OR. Adicionalmente a esto, se debe contar con un mecanismo para identificar la señal que ocasionó la interrupción. Esto se realizó de manera muy sencilla, como se mencionó en el capítulo anterior, al conectar la alarma del RTR a

otra terminal de entrada. De esta manera, en cuanto la interrupción se active, se lee el valor de la otra terminal de entrada y dependiendo de su valor se determina el elemento que ocasionó la interrupción.

Una vez explicado las interrupciones y su origen, en el diagrama de la figura 4.2 podemos apreciar las interrupciones que atenderá el microcontrolador durante su funcionamiento. La interrupción causada por el inicio programado y por el botón principal es básicamente la misma en el circuito pero, para fines del *firmware*, se manejarán como interrupciones distintas.

El inicio programado debe de ser activado mediante la PC y sólo es posible que éste suceda antes de que el *datalogger* comience a adquirir. En cuanto a la interrupción del botón principal, ésta depende del usuario y puede suceder en cualquier momento. Una vez que el usuario presione el botón principal, le indica al microcontrolador que salga de su estado de bajo consumo y espere algún tipo de acción sobre él. Esta acción puede ser: iniciar o parar la adquisición o, iniciar la comunicación con la PC. A través de una conexión con una PC es posible hacer las acciones que se muestran en el diagrama y que son iniciar la adquisición, configurar el *datalogger* y vaciar los datos adquiridos por éste.

Si el sistema se encuentra en un estado de adquisición dos interrupciones más son posibles. Estas son las interrupciones para adquirir las variables medidas por el *datalogger*.

Todas las interrupciones finalizan después de que el microcontrolador ha ejecutado la función programa dentro de la interrupción. En el caso de la interrupción por medio de un botón, ésta posee un tiempo de espera para que el usuario realice una acción, si después de dos minutos no se ha realizado acción alguna o se ha establecido una conexión con la computadora, ésta interrupción también termina. Una vez que las interrupciones han terminado, el microcontrolador vuelve a entrar en estado de bajo consumo esperando a que ocurra otra interrupción que lo haga despertar.

Una vez conocido el proceso general que sigue el *datalogger* en su ejecución, en las siguientes secciones ahondaremos en determinadas funciones del *datalogger*.

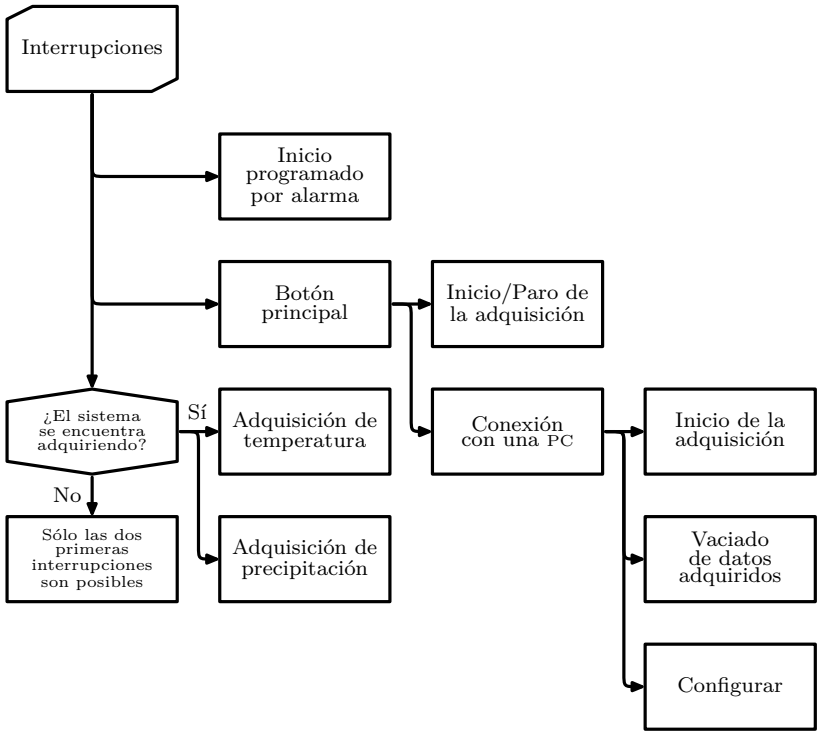


Fig. 4.2. Diagrama de interrupciones del *datalogger*.

4.2. Adquisición de datos

Antes de entrar de lleno en la adquisición de datos debemos conocer el funcionamiento del reloj en tiempo real, ya que éste será el encargado de entregarnos una referencia de tiempo de las señales adquiridas.

4.2.1. Uso del reloj en tiempo real

Como ya se estudió en la capítulo anterior, el RTR DS1337 tiene como función principal llevar el manejo de la hora y fecha del sistema, adicionalmente este dispositivo cuenta con alarmas programables que activan interrupciones en el microcontrolador. Esto convierte al reloj en tiempo real no sólo en una fuente de de información sobre la hora y la fecha sino también en el controlador de tiempos del *datalogger*.

La forma de comunicación entre el RTR y el microcontrolador es a través del *bus* I^2C. La transmisión de información entre ellos se realiza para que el microcontrolador escriba o lea sobre los registros internos del RTR. Estos registros tienen dos propósitos principales: almacenar la fecha y hora, controlar el funcionamiento de las alarmas. La localización de estos registros y la forma de mandar información y recibirla se puede ver de manera detallada en la hojas de especificaciones del DS1337. Para nuestros fines, sólo es necesario saber que la información concerniente a la hora y fecha almacenados por el DS1337 tiene el formato mostrado en la tabla 4.2 y es en éste mismo formato como el microcontrolador recibe la información de la hora y fecha del RTR.

Resistro	**Rango**
Segundos	0-60
Minutos	0-60
Horas	0-12+AM/PM o 0-24
Día de la Semana	1-7
Día del Mes	1-31
Mes	1-12
Año	0-99

Tabla 4.2. Fomato de la fecha y hora del RTR (DS1337).

Todos los registros mostrados tiene un byte de longitud y están en código BCD. Adicionalmente a los registros de fecha y hora existen otros que controlan la configuración y activación de las alarmas. Cada alarma posee un número determinado de registros donde se puede especificar la hora y la fecha en la que se desea que la alarme se active. Las alarmas del RTR se pueden configurar de la siguiente manera:

Como podemos apreciar en la tabla, la única diferencia entre la alarma 1 y la alarma número 2 del RTR es que la primera posee una resolución de segundos. Ambas alarmas pueden programarse para accionarse hasta un mes después de ser establecidas. Una vez que se han programado ya activado sus alarmas, el RTR comparará los registros de tiempo y hora con el de éstas cada segundo, en caso de que exista una coincidencia, el RTR elevará la tensión de la terminal de salida de la alarma correspondiente.

El nivel de salida de esta terminal se mantendrá en alto hasta que la bandera correspondiente a esta alarma sea limpiada por el microcontrolador. Esta es la razón por la cual se conectaron las salidas de las alarmas del RTR a las terminales de interrupciones síncronas del microcontrolador. Debido a que

Alarma 1
Cada segundo
Cada minuto
Cuando los segundos coincidan
Cuando los segundos y minutos coincidan
Cuando los segundos, minutos y la hora coincidan
Cuando los segundos, minutos, hora y día de la semana coincidan
Cuando los segundos, minutos, hora y día del mes coincidan
Alarma 2
Cada minuto, (segundo 0 de cada minuto)
Cuando los minutos coincidan
Cuando los minutos y la hora coincidan
Cuando los minutos, hora y día de la semana coincidan
Cuando los minutos, hora y día del mes coincidan

Tabla 4.3. Opciones de los menús.

la señal de interrupción se debe de mantener hasta que el microcontrolador salga de su estado de bajo consumo y es el microcontrolador el que controla cuando baja esta señal, la señal de alarma del RTR siempre reanudará la operación del microcontrolador.

4.2.2. Recepción de las señales del pluviómetro

Como ya sabemos, el pluviómetro es un instrumento mecánico que trasmite un pulso de señal en el momento en que ha ocurrido el volcado del balancín. Estos pulsos son recibidos a través de la interrupción asíncrona del microcontrolador, el flanco de subida de este pulso activa la interrupción que indica al programa que un evento ha ocurrido. Recibida la señal, el *datalogger* registra inmediatamente este evento en memoria valiéndose de la fecha y hora procedentes del RTR.

Cada evento del pluviómetro, registrado por el *datalogger*, representa cierto volumen de agua precipitado. Para poder convertir un determinado número de eventos en el volumen de precipitación colectada en cierto período de tiempo,

es necesario saber cuál es la cantidad de lluvia con la que el balancín del pluviómetro vuelca. Esta cantidad de lluvia está dada mediante una calibración que el fabricante realiza al equipo antes de venderlo y es un valor constante para cada modelo de pluviómetro. Debido a que este valor no cambia entre mediciones, no es necesario guardarlo junto con los registros, una localidad especial en la memoria EEPROM del microcontrolador está reservada para este fin y su valor se establece durante el proceso de calibración de los sensores.

4.2.3. Recepción de las señales de temperatura

La señal de la sonda de temperatura es recibida a través de una terminal del convertidor analógico digital (A/D), esta señal, dada su naturaleza, puede ser adquirida y registrada en cualquier momento por el microcontrolador. Pero para realizar un buen proceso de adquisición una frecuencia de muestreo y registro deben ser establecidas. Adicionalmente a la frecuencia de muestreo, el microcontrolador sólo obtendrá el valor de su convertidor analógico digital, valor que, para ser transformado en un valor de temperatura ambiente, requiere de un determinado procesamiento.

Recuperación del valor de temperatura

El proceso de conversión del valor del convertidor A/D a un valor de temperatura es el siguiente:

En primer lugar la palabra digital de salida del convertidor A/D se debe adecuar para que represente el valor de tensión de la terminal de entrada del convertidor A/D mediante la ecuación 3.1 que define la salida del convertidor A/D. Una vez obtenido este valor, se debe obtener la tensión a la salida de la sonda de temperatura dividiendo el valor de entrada del convertidor A/D entre la ganancia del amplificador usado en el acondicionamiento. Posteriormente, con este valor de tensión se debe obtener la resistencia del termistor y finalmente mediante la ecuación de Steinhart-Hart obtener la temperatura.

Dado que el proceso descrito es difícilmente implementable en un microcontrolador de 8 bits, por la necesidad de utilizar complejas operaciones matemáticas y datos con parte decimal, se decidió no realizar este procesamiento mediante el microcontrolador del *datalogger* y dejárselo realizar a la computadora, una vez que ésta reciba los datos adquiridos.

Para que la computadora pueda transformar el valor del convertidor A/D en un valor de temperatura, ésta requiere de ciertos valores procedentes de las características tanto de la sonda, como del acondicionamiento que recibió su señal de salida. De estos los que pueden ser considerados como constantes son:

el valor de tensión de referencia del convertidor A/D, dado que, en teoría, este valor no cambia y los parámetros de la ecuación Steinhart-Hart, debido a que el *datalogger* fue desarrollado para operar sólo con la sonda de temperatura 107 y los parámetros de esta ecuación no cambian para diferentes sondas del mismo modelo.

De los parámetros que se consideran variables son: la ganancia dada a la señal de salida de la sonda, debido a que, como ya se mencionó, ésta puede ser ajustada y el *offset* de la señal de la temperatura la cual puede variar debido a factores como el envejecimiento de la sonda. Un proceso para la obtención de dichos parámetros debe de ser considerado dentro de la calibración de los sensores.

Todas las variables de calibración de los sensores deben de estar presentes una vez que éste trasmita la información colectada a la PC. El *datalogger* obtendrá estas variables por parte del usuario antes de adquirir los datos y las almacenará en localidades de su memoria EEPROM hasta el momento en que los datos sean colectados por la PC.

Frecuencia de muestreo

Parte del proceso de adquisición de datos conlleva a considerar la velocidad de adquisición de éstos. Como se vio en la sección de adquisición de señales analógicas, en el capítulo de generalidades, la única restricción para fijar esta velocidad es el criterio de Nyquits.

En el caso del pluviómetro la velocidad de adquisición de los datos está definida por la naturaleza del sensor, sólo obtendremos una medición si el balancín ha volcado. En el caso del termistor la frecuencia de muestreo debe de ser definida por nosotros, para ello es necesario conocer la frecuencia de variación de la temperatura ambiente.

Tomando como ejemplo las estaciones meteorológicas automatizadas (EMAs) del Servicio Meteorológico Nacional (SMN) que toman mediciones de la temperatura ambiente cada minuto y las envían cada diez minutos[1] y de otro tipo de estaciones automatizadas que registran la información adquirida cada hora[2], se decidió que, para el caso del *datalogger*, la tasa de adquisición fuera en múltiplos de un minuto hasta un máximo de treinta y el registro de los datos fuera configurable por el usuario.

[1] http://smn.cna.gob.mx/productos/emas/

[2] **Alcinova S.**, *Automatization of Meteorological Measurements in Republic of Macedonia for the Needs of Environmental Sustainability.*

Para obtener la tasa de adquisición de la señal de temperatura se toma como base la tasa de registro que el usuario quiera y se divide entre treinta, el número máximo de registros que se pueden realizar. Lo anterior nos da un número fraccionario, así que se toma el valor entero más alto y se obtiene la tasa de adquisición de la temperatura. Por ejemplo, si se desea registrar cada 30 minutos, se adquirirá un valor de temperatura cada minuto pero si se desea registrar cada 45 minutos se adquirirá un valor de temperatura cada dos minutos.

La elección de un máximo de treinta muestras se realizó tomando en cuenta que se puede hacer una medición cada minuto en una tasa de registro de 30 minutos, tasa de registro comúnmente usada, y que la suma de 30 registros a 10 bits no sobrepasa el valor máximo almacenable en un entero.

La base de tiempo de un minuto para la medición y registro de la temperatura se obtuvo configurando la alarma dos del RTR para activarse cada minuto. De esta manera el proceso de adquisición y registro de la señal de temperatura se realiza de acuerdo al diagrama de flujo de la figura 4.3

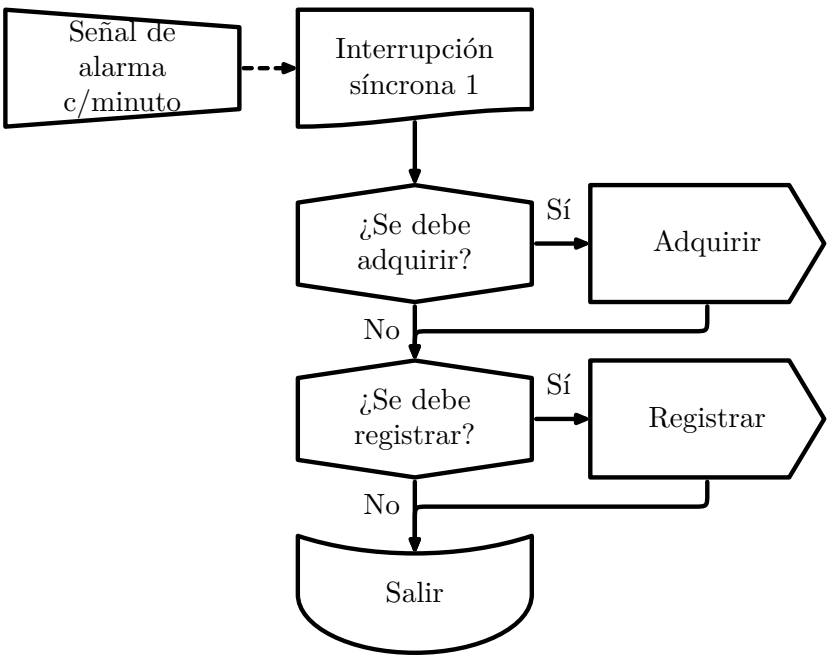


Fig. 4.3. Diagrama del proceso de adquisición y registro de la temperatura ambiente.

Como se muestra en el diagrama, todo el proceso comienza al recibir la señal de alarma del RTR. Una vez que esto sucede, el microcontrolador determina si es momento de adquirir o de registrar la temperatura, si es así, la acción se realiza y se sale de la interrupción. La adquisición se refiere al proceso de leer un valor de temperatura mientras que el registro es guardar el valor promedio de mediciones hechas durante el periodo de registro en las memorias EEPROM.

El proceso de registro de la temperatura ambiente se realiza de acuerdo a la secuencia que se indica en el diagrama de la figura 4.4. Este proceso obtiene una muestra de temperatura ambiente realizando cinco mediciones consecutivas del convertidor A/D y obteniendo el promedio de las mismas, previo a descartar el valor más alto y más bajo de éstas. El promedio obtenido es guardado para ser considerado al momento de registrar esta variable en las memorias, una vez que el período de registro establecido por el usuario haya sido alcanzado.

4.3. Guardado de los datos

4.3.1. Manejo de las memorias de registro de datos

Las memorias que se utilizaron para el registro de datos fueron las memorias EEPROM 25LC1024. Estas memorias se comunican con el microcontrolador a una velocidad de 100 [Hz] a través del *bus* I^2C. A este *bus* se encuentran conectadas cuatro memorias con una capacidad de almacenamiento total de 4.096 [MBits] o 512 [kBytes]. Cada memoria tiene una capacidad de 1024 [kBits] divididos en dos bloques de 512 [kBits]. Cada bloque a su vez se encuentra dividido en 500 páginas de 128 [Bytes] cada una.

Dentro del *bus* cada memoria es identificada a través de una dirección, que se forma basándose en la estructura mostrada en la figura 4.5

El bit R/W selecciona el tipo de operación que se desea realizar con las memorias, lectura o escritura. Los parámetros A son establecidos por hardware, a través de las conexiones que se realizaron de las termínales E_0 y E_1 del circuito integrado de memoria. El bit de bloque B_0 puede ser un uno o cero binario, de acuerdo al bloque dentro del circuito integrado que se quiera seleccionar. Finalmente, el código de control es un valor fijo que comparten todos los circuitos integrados de memoria de esta familia.

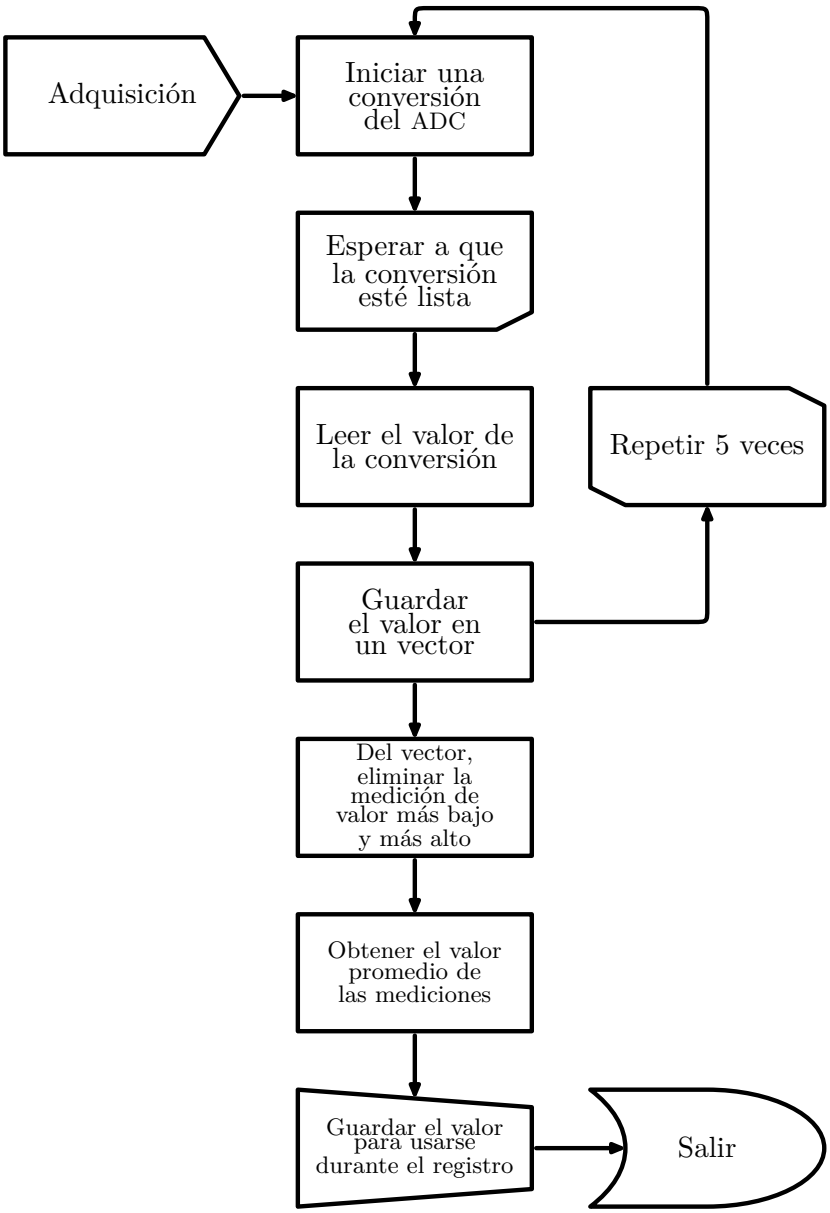


Fig. 4.4. Proceso de la adquisición de la temperatura ambiente.

Solamente dos funciones son posibles de realizar con las memorias, guardar datos en ellas y leer los datos almacenados en ellas. La escritura o lectura se realiza en dos pasos; en el primer paso se especifica la dirección de memoria

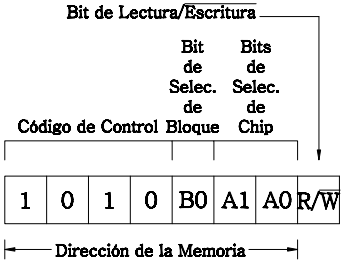


Fig. 4.5. Estructura del byte de dirección de las memorias 24LC1025.

que se quiere leer o escribir, y en el segundo paso se escribe o se lee esa localidad de memoria apuntada.

La dirección de memoria se especifica a través de dos bytes, según la estructura mostrada en la figura 4.6. Esta dirección es la de un byte en memoria y puede varias desde el 0000_{H} hasta el $\mathrm{F9FF}_{\mathrm{H}}$. Esta dirección sólo especifica una dirección dentro del bloque seleccionado en el byte de dirección.

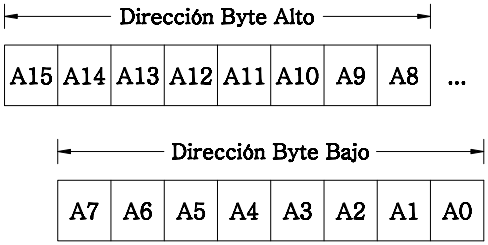


Fig. 4.6. Estructura de los bytes de dirección.

El proceso de escritura se realiza a través de la secuencia en tiempo mostrada en la figura 4.7. Primeramente se inicializa la comunicación $\mathrm{I}^2\mathrm{C}$, mediante una señal de inicio. Una vez iniciada la comunicación, a través del byte de control, se selecciona la memoria y el bloque que se desea escribir. Después se envían los dos bytes de la localidad de memoria donde la operación de escritura se desea realizar, y finalmente se envían los bytes a escribir en dicha localidad. Se pueden continuar enviando más bytes y estos serán guardados en las localidades continuas hasta alcanzar el final de una página, momento en el cual el apuntador interno de la memoria se reiniciará a la dirección de inicio de la página, con lo cual se podrían sobreescribir valores previamente enviados.

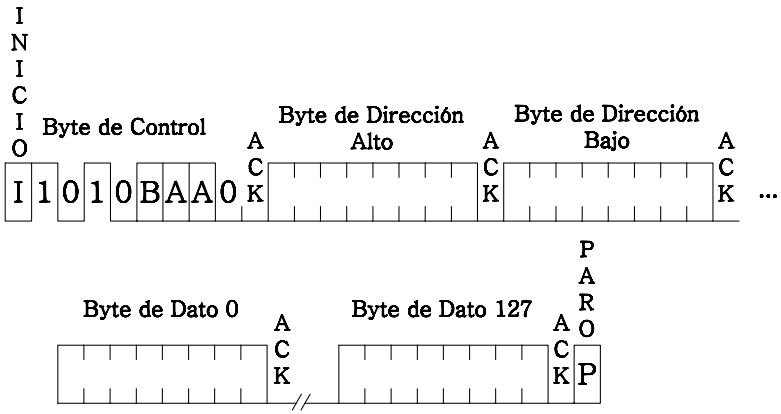


Fig. 4.7. Proceso de escritura de múltiples bytes en memoria.

El proceso de lectura es similar al de escritura pero éste se realiza en dos pasos: primeramente se establece el apuntador de memoria con una operación de escritura y después se reinicializa la comunicación y se inicia una operación de lectura. Mediante este procedimiento la memoria envía el byte de información cuya dirección fue especificada en la operación de escritura previa. Si el microcontrolador sólo desea ese byte, éste puede enviar una señal de no enterado (NACK) y parar la comunicación. Si por el contrario, el microcontrolador desea leer los bytes siguientes a dicha dirección, éste puede responder al byte recibido con una señal de enterado (ACK), entonces la memoria enviará el byte siguiente a la dirección especificada. Si el microcontrolador continúa respondiendo con señales ACK a cada byte recibido, la memoria continuará enviando el contenido de los bytes en las direcciones continuas hasta que una señal NACK es enviada. Se puede hacer lectura de bytes continuos que se encuentren entre el inicio y la mitad del bloque de memoria y entre la mitad del bloque de memoria y el final. En el caso en el que se sobrepasen estas fronteras el apuntador de lectura se regresará al principio del subbloque de memoria. Este proceso de lectura de múltiples bytes se puede ver en la figura 4.8.

4.3.2. Manejo del apuntador de memoria

El manejo de la memoria de datos se realizó mediante la implementación de dos apuntadores de dirección, uno que apunta a la localidad donde los datos empezaron a guardarse y otra que se va incrementando conforme los datos se guardan, esta idea es mostrada en la figura 4.9.

El uso de estos apuntadores permite mantener los datos útiles entre los límites de estas direcciones. Existen dos ventajas en realizar esto:

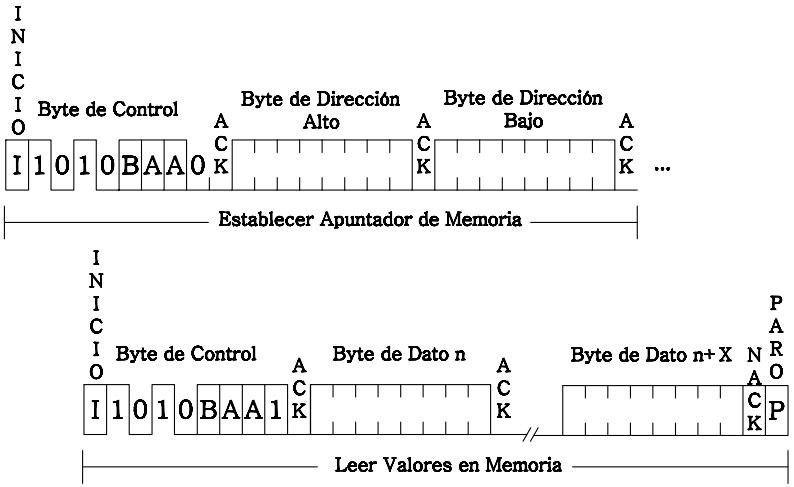


Fig. 4.8. Proceso de lectura de múltiples bytes en memoria.

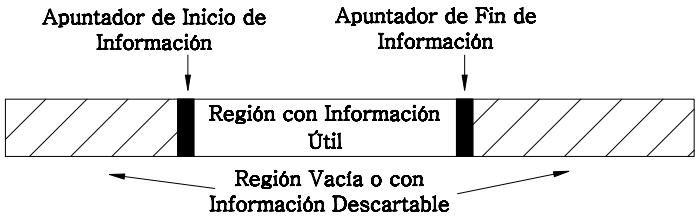


Fig. 4.9. Estructura de la memoria de datos.

- Se evitan borrar físicamente los datos cuando ya no sean útiles y
- Se permite un gasto uniforme de las memorias disponibles

La primera ventaja radica en que si se determina que ciertas localidades ya no tienen información útil, simplemente se dejan fuera de la región entre los apuntadores de inicio y fin del datos. Lo anterior logra que el sistema, en vez de borrar la información, la coloque fuera de su región de datos útiles, acción que tiene el mismo efecto que borrarlos pero que evita gastar de más las celdas de memoria.

La segunda ventaja se basa en que el apuntador de inicio puede ser movido y no siempre debe de permanecer en la primera localidad de la primera memoria. Esto hace posible que los datos puedan ser escritos de manera uniforme sobre todas las memorias, después de múltiples secciones de medición y no sólo en las primeras localidades. Sin implementar esta función, a la larga se

ocasionaría un fallo por ciclos de escritura en las primeras memorias, mientras que la última podría nunca haber sido utilizada.

Para realizar la implementación anterior se deben considerar dos apuntadores: uno para la localidad de inicio de los datos y otra para el final de los mismos. El apuntador más difícil de implementar es el de fin de datos, ya que éste tiene que ser incrementado de manera tentativa cada que se grabe una nueva medición en las memorias. El apuntador de inicio de datos no presenta tal dificultad, dado que sólo se escribe al inicio de una sesión de adquisición. Se debe de tener en cuenta que esta implementación requiere que el microcontrolador maneje la memoria como un anillo, para que, una vez que ha escrito en la dirección final de la última memoria, éste pase de manera automática a la primera.

Dado que el diseño del apuntador de fin de datos determina la manera de implementar ambos apuntadores de dirección, se comenzará con la descripción de éste.

Apuntador de fin de datos

El apuntador de fin de datos es una variable, dentro del programa del microcontrolador, que apunta a la siguiente localidad disponible para el guardado de la información adquirida. Esta variable, junto con la dirección donde se inició el registro de las mediciones, nos ayuda a delimitar la información que se ha sido guardado en la presente sesión de medición. El microcontrolador mantendrá esta variable en memoria volátil SRAM para utilizarla al momento de registrar las mediciones. El problema de esta variable se presenta cuando la alimentación del microcontrolador falla o éste es reinicializado. Cuando esto sucede el valor de esta variable se pierde y por ende perdemos el apuntador de la siguiente dirección de memoria disponible para el registro de las variables.

El problema anterior se puede solucionar guardando una copia de este apuntador en alguna localidad en memoria no volátil, en nuestro caso memoria EEPROM, y que ésta sea actualizado con cada medición. Existe un problema a la solución anterior, las memorias EEPROM sólo pueden ser escritas un determinado número de veces (aproximadamente 1 millón de veces) antes de que estas dejen de retener su valor guardado. Si utilizamos un solo registro y éste se actualiza en cada medición, la vida útil del *datalogger* estaría limitada a 1 millón de mediciones antes de que éste fallara al no poder guardar este apuntador. La solución a este problema se logra mediante la distribución de este apuntador en varias localidades de memoria, de esta manera por cada

localidad extra para guardar este apuntador se obtienen un millón más de mediciones posibles. Si bien esta solución es bastante cercana a lo que queremos, es posible mejorarla, según se describe a continuación.

Como ya se mencionó, las memorias EEPROM utilizadas están organizadas internamente a través de páginas de memoria. Estas páginas de memoria tienen una longitud de 128 bytes, comenzando en la dirección cero y están espaciadas cada 128 localidades. El manejo de la memoria a través de páginas nos da la oportunidad de tener una mejor organización y nos permiten realizar una mejor implementación del apuntador de fin de datos. Esta mejoría se logra si en vez de incrementar el apuntador de memoria en cada registro, se incrementa cada que una nueva página es utilizada. Con el método anterior se logra que el número de veces que se debe de actualizar el apuntador en memoria no volátil sea considerablemente disminuido.

Adicionalmente a tener un apuntador de página, una mejor solución conlleva que su registro no sólo se realice en una localidad de memoria sino en múltiples localidades con el fin de repartir el desgaste de las celdas.

Para realizar lo anterior se utilizó la primera página de la primera memoria para crear una tabla de apuntadores de página. Esta tabla permite organizar las distintas localidades donde el apuntador de fin de datos será almacenado. La estructura de ésta se puede ver en la figura 4.10.

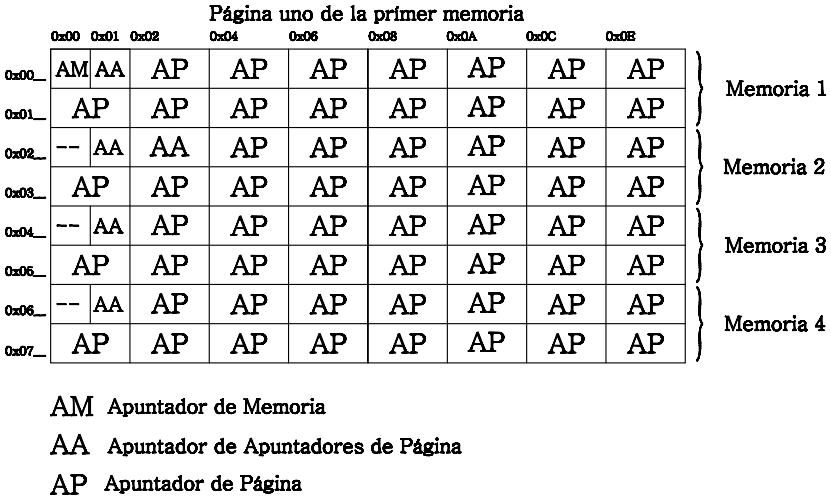


Fig. 4.10. Estructura de la tabla del apuntador de fin de datos.

La tabla está integrada para que el microcontrolador, una vez que la lea, pueda armar una dirección de una localidad en cualquiera de las cuatro memorias

disponibles. Para realizar esto, la tabla mantiene un registro tanto del valor de la memoria usada como de la página dentro de esta memoria. Cada memoria posee un conjunto de quince celdas para almacenar los apuntadores de sus páginas, pero sólo una celda está activa a la vez. El contenido de los apuntadores AM y AA son direcciones que, al ser leídas e interpretadas, originan direcciones sobre otras celdas de la tabla hasta originar la dirección del byte de inicio de la siguiente página disponible para el guardado de datos.

El apuntador de memoria (AM) indica cuál es la memoria que se está utilizando para guardar datos y por lo tanto que apuntador se encuentra en uso. El apuntador de apuntador de página (AA) especifica, de la lista de los quince apuntadores, que esa memoria dispone cuál es el que se encuentra en uso, y finalmente, el apuntador de página (AP), especifica dentro de esa memoria que página se está utilizando. Cada apuntador tiene una longitud de dos bytes, dado que el número de páginas por memoria requiere de dos bytes para ser almacenada: el apuntador de memoria y el apuntador de apuntadores de página tienen una longitud de un solo byte.

El proceso para determinar la página de memoria en uso se realiza de la siguiente manera. Se lee el contenido del primer byte de la primera memoria donde se localiza el apuntador de memoria. Con ese valor se localiza el apuntador de apuntador de página de esa memoria, al leer su contenido, se localiza el apuntador de página en uso. Una vez leído el contenido de ese apuntador se puede localizar la página en uso en determinada memoria. Para obtener la dirección del byte correspondiente al inicio de la página, sólo basta multiplicar el valor del apuntador de página por el tamaño de página (128) y se encuentra la dirección de la siguiente localidad disponible.

Una vez que una página se ha terminado de escribir, el programa escribirá en la siguiente página disponible, pero antes deberá de reportar el cambio en la tabla de apuntadores. Para realizar esto, el microcontrolador debe escribir la nueva página que se usará en el apuntador de página. En caso de que éste haya alcanzado el máximo valor del AP, que corresponde al máximo número de páginas por memoria, se deberá de escribir en otra memoria. Para hacer lo anterior, tanto el apuntador de memoria como el apuntador de apuntadores de esa memoria deberán de ser incrementados.

Con el proceso anterior se logra que el apuntador de memoria se reescriba cada que se ha llenado una memoria, el apuntador de apuntadores de página cada que se llena la memoria que representan y los apuntadores de página, aunque se reescriben con cada nueva página usada, sólo se vuelvan a utilizar cada que la memoria que representan se haya llenado quince veces.

De esta manera, cuando el microcontrolador deje de ser alimentado o sea reinicializado y su apuntador en SDRAM se pierde, éste podrá leer la tabla de apuntadores de la memoria EEPROM y a través de ésta fijar su apuntador en SDRAM.

Apuntador de inicio de datos

Adicionalmente al apuntador de fin de datos, para enmarcar la información registrada por el *datalogger* se requiere de un apuntador que guarda la dirección de la primera página donde se comenzó el registro de los datos. Este valor es establecido al inicio de una sesión de adquisición y sólo se sobrescribe si se requiere que cierta cantidad de información ya no se encuentre disponible. Para realizar lo anterior, bastará con escribir, en este apuntador, la dirección de la última página donde se guardaron datos en esa sección, es decir el contenido del apuntador de fin de datos. De esta manera, el sistema considerará que los datos anteriores son localidades disponibles.

Este apuntador, dado que no requiere una escritura de manera recurrente, se fijó en una única posición de memoria no volátil dentro de la memoria EEPROM del microcontrolador.

4.3.3. Escritura de datos en página

El uso de la memoria por páginas nos permite, además de un mejor uso de los apuntadores de dirección, organizar de una manera más ordenada la memoria. Lo anterior se logra al especificar una página por tipo de variable de medición e identificar cada una de ellas a través de una cabecera. Esto nos da una ventaja adicional, nos permite agrupar la información importante para un grupo de mediciones al inicio de la página y no tener que repetir esta información al guardar cada medición. De esta manera, toda página con información, ya sea de precipitación pluvial o de temperatura ambiente, está compuesta de dos partes:

- Una cabecera y
- La información de las mediciones

Esta estructura se puede apreciar en la figura 4.11

La información guardada en la cabecera y la composición de los datos de las mediciones varían de acuerdo a la variable medida.

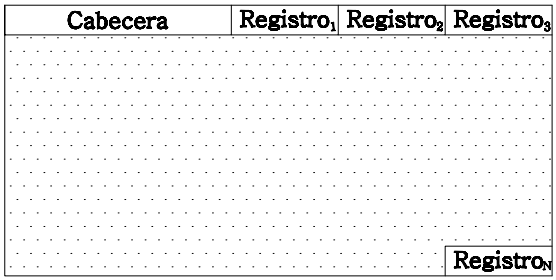


Fig. 4.11. Estructura de una página de datos en memoria EEPROM.

Registro de datos del pluviómetro

La estructura de la cabecera de una página de precipitación pluvial se muestra en la figura 4.12, el identificador que se utiliza es el valor hexadecimal 10_{H}, los dos valores siguientes FF_{H} sólo sirven para mantener la región de datos con las localidades exactas para mantener un número entero de registros.

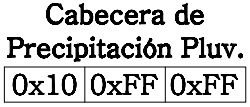


Fig. 4.12. Estructura de la cabecera de precipitación pluvial.

La estructura de los datos de precipitación pluvial se muestran en la figura 4.13, como se puede apreciar, sólo se almacena la fecha y hora en que el evento ocurrió. Dado que el valor de volcado del pluviómetro es guardado durante el momento de la calibración, ningún otro dato se necesita registrar.

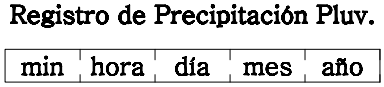


Fig. 4.13. Estructura de un registro de precipitación pluvial.

Registro de datos de la temperatura ambiente

La estructura de la cabecera de una página que guarda datos de temperatura ambiente se muestra en la figura 4.14, como se puede ver en la figura, mayor información es guardada en esta cabecera. Esto se debe a que en todo momento sabemos con qué frecuencia se registran los datos de temperatura, por lo tanto, no es necesario guardar la estampa de tiempo de cada medición

hecha. Solamente cuando se registra el primer valor de temperatura en la página, se escribe en su cabecera el valor del periodo de registro y la fecha y hora proveniente del RTR. Las mediciones consecutivas podrán ser estampadas con la información de la cabecera una vez que se reciban los datos por la PC. Una página de temperatura es identificada mediante el valor hexadecimal de 25_{H} en su primer byte.

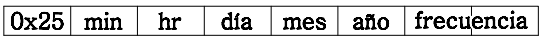


Fig. 4.14. Estructura de la cabecera de temperatura ambiente.

En la figura 4.15 se muestra la estructura de la parte de datos de una página que almacena temperatura ambiente. La información que se almacena para cada medición, es el valor promedio de las mediciones adquiridas mediante el convertidor A/D en cada período de registro. Como ya se mencionó, con este valor y las variables de calibración la PC será capaz de convertir estos valores en mediciones de temperatura al momento de recibirlos.

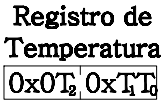


Fig. 4.15. Estructura de un registro de temperatura ambiente.

Al registrar los datos de la manera descrita, se obtiene que por cada página usada se puedan almacenar hasta 60 registros de temperatura ambiente por página y 25 registros de precipitación pluvial. Dado que ninguna variable posee memoria reservada, si el *datalogger* se configura para adquirir tanto precipitación pluvial como temperatura ambiente, ambas variables compiten por espacio en memoria. Si sólo se utilizara el *datalogger* para adquirir precipitación pluvial, la cantidad de registros que podría guardar serían 99,975. Si sólo se guardaran datos de temperatura ambiente se podrían almacenar 239,940 registros. En caso de que cada registro se realizara cada treinta minutos se puede almacenar con el *datalogger* hasta trece años y medio de mediciones. Como se puede apreciar, la mayor limitante en cuanto a la autonomía del *datalogger* es la capacidad de su fuente de energía.

4.3.4. Modos de manejo de la memoria

Los modos de guardado de datos son dos y están pensados para el caso en que las memoria agoten su capacidad durante una sesión de adquisición.

Con la primera opción, el *datalogger* para la adquisición de las variables en el momento en que la memoria se llene. El otro modo permite que la adquisición continúe, pero los nuevos registros serán guardados en el lugar de los registros más antiguos. Estas opciones son configuradas sólo mediante una conexión con la PC.

4.4. Envío de datos

Para realizar la comunicación entre el microcontrolador y la PC se hizo uso de los transceptores USB y RS-232, estos reciben señales de la USART del microcontrolador y los acondicionan para poder ser recibidos por la computadora mediante los puertos correspondientes a cada estándar.

La comunicación con la USART se realiza a una velocidad de 19.2 [kbps]. Esta velocidad tienen un error en una comunicación asíncrona de 0.2%, error que está por abajo del máximo aceptable para este tipo de velocidad y estructura del marco de datos, el cual es del 2%. Con esta velocidad se espera poder trasmitir toda la información almacenada en las memorias en aproximadamente cinco minutos considerando que 19.2 [kbps] representa una velocidad de 1920 bytes por segundo y que la capacidad máxima de las memorias es de 512 mil bytes.

El envió de datos se basa en implementar los protocolos de comunicación y enviar las páginas de información una por una cuando el programa de la computadora las pida. Previo a la petición de envió de datos, la PC solicita las direcciones de inicio y fin de los datos colectados y a partir de estas direcciones calcula cuántas páginas deben ser solicitadas al *datalogger* y la cantidad de memoria libre y en uso.

La información que es trasmitida es aquella que se encuentra entre el apuntador de inicio de datos y el de fin de datos, excepto si se seleccionó el modo de grabación de reescritura de datos y se están guardando los registros sobre registros antiguos. Cuando esto sucede el microcontrolador determina que el apuntador de inicio de información ya no es válido, por lo que envía toda la información guardada en memoria, comenzando con la página siguiente del apuntador de fin de memoria hasta la dirección de la página apuntada por éste.

Una vez que el usuario, mediante el botón principal, ha despertado al microcontrolador, el microcontrolador espera dos minutos para recibir un comando desde la PC por medio de la USART, si un comando no es recibido dentro de ese período de tiempo el microcontrolador entre en estado de bajo consumo. Si el microcontrolador recibió un comando válido, este permanecerá encendido hasta que un comando de término de la comunicación haya sido recibido. Comandos adicionales están disponibles para realizar la configuración del *datalogger*, calibrar sus sensores y habilitar el modo de prueba de éste.

4.5. Funciones adicionales

A parte de las funciones ya mencionadas en los apartados anteriores, existen otras que hacen que el *datalogger* pueda interactuar con el usuario para que éste lo pueda poner en funcionamiento, configurarlo, etc. Estas funciones están divididas en dos: las que se pueden realizar en el circuito y aquellas que requieren que el sistema esté conectado a la computadora.

Primeramente se mencionarán aquellas realizables con el circuito y posteriormente se indicarán las que requieren una conexión con la PC. Las que se pueden realizar con el sistema son las siguientes:

- Inicio y paro de la adquisición del *datalogger*
- Formateo de la memoria de registros
- Establecer parámetros de operación por defecto
- Encender o apagar la opción de indicación de cada adquisición

Cada una de estas funciones requiere de controles para poder ser indicadas, ya sean botones de acción momentánea o de posición. Estos controles se muestran en la figura 4.16.

4.5.1. Inicio y paro de la operación del *datalogger*

Como su nombre lo dice, esta función inicia o detiene la adquisición de datos del *datalogger*. Estas acciones se realizan mediante los dos botones de acción momentánea. El botón principal sirve para solicitar la opción y el secundario para ejecutarla. En el mecanismo de inicio y paro de la adquisición

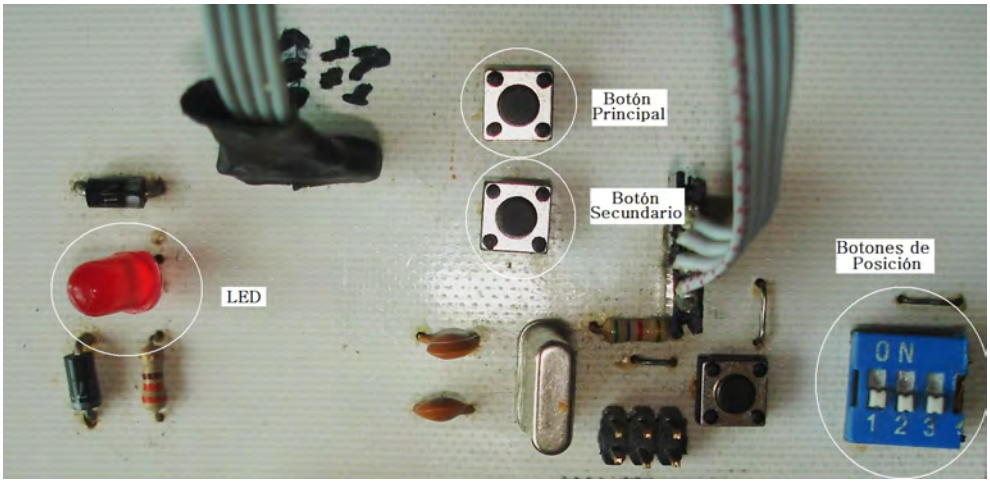


Fig. 4.16. Controles presentes en el *datalogger*.

de datos es el mismo y sólo realiza una acción o la otra, dependiendo del estado de operación en que se encuentre el sistema; si el *datalogger* se encuentra adquiriendo la operación detiene la adquisición y, si ésta se encuentra detenida, la inicializa. El sistema responde a través del parpadeo del LED con una diferente secuencia de acuerdo a la función que realizó.

El inicio del sistema se realiza con los últimos parámetros configurados y en caso de que haya sido restablecida su configuración con los parámetros por defecto.

A parte del inicio por botones se puede iniciar la adquisición del *datalogger* programado por fecha y hora. Esto requiere de una conexión con la PC, pero en caso de que se quiera cancelar este modo de inicio, se puede realizar esto al presionar el botón principal y mantener presionado el botón secundario por cinco segundos.

4.5.2. Formateo de la memoria de registros

Mediante el interruptor de posición número uno, se puede borrar todos los registros guardados en la memoria. Esta operación sólo puede ser realizada si el *datalogger* es puesto en funcionamiento con este interruptor puesto en su estado lógico "alto", el *datalogger* indica la ejecución de esta función manteniendo el LED encendido y ninguna operación más puede realizarse. Para volver a la operación normal tiene que ser apagado y reencendido el *datalogger*.

La función de formateo reescribe la tabla de apuntadores de memoria, que se creo para manejar el apuntador de fin de datos, lo cual hace que este

apuntador de fin de datos se establezca en la segunda página de la primera memoria. Adicionalmente, el apuntador de inicio de datos es igualmente establecido en este valor. Estas dos acciones ocasionan que el microcontrolador determine que ningún dato ha sido guardado en memoria. El proceso de formateado realmente nunca borra las memorias, simplemente le indican al *datalogger* que las localidades se encuentran disponibles para ser escritas.

4.5.3. Establecer los parámetros de operación por defecto

El establecimiento de los parámetros de operación por defecto se realiza colocando el interruptor de posición número dos en su estado lógico "alto" antes de que el *datalogger* entre en funcionamiento. Mediante esta opción se establecen los valores de configuración por defecto que se enlistan más adelante.

4.5.4. Encender o apagar la opción de indicación de cada registro

La opción de indicación del registro de datos le permite al usuario decidir si desea ver un parpadeo del LED cada que un valor es registrado o no. Esta opción puede ser encendida o apagada durante la adquisición de datos mediante el interruptor de posición número tres y permite indicarle al usuario que se están realizando el registro de datos. Por fines de eficiencia en el uso de energía es preferible dejarla apagada.

Otras funciones del *datalogger* están disponibles pero sólo pueden ser accedidas a través de una conexión con la computadora, éstas son:

- Calibración de los sensores
- Modo de prueba de los sensores
- Configuración del *datalogger*

4.5.5. Calibración de los sensores

La calibración de los sensores es necesaria para obtener mediciones precisas de estos por parte del *datalogger*. En el caso del pluviómetro la calibración consiste en saber cuál es el valor con el que éste vuelca. Este valor varía de acuerdo al modelo del pluviómetro y es proporcionado por el fabricante.

En el caso del termistor, la parte de acondicionamiento es tan específica para este tipo de sonda de temperatura que sólo se puede asegurar el uso del

termistor para dicho modelo de sonda. De acuerdo al fabricante, la calibración que es necesario hacer a la sonda es sólo del desplazamiento del valor medido al real (*offset*). Aún así, en la etapa de acondicionamiento, el valor de amplificación que sufre la señal proveniente de la sonda de temperatura puede variar. Por lo tanto, para realizar la calibración de la sonda de temperatura, se requerirán de dos valores: el *offset* y la ganancia del amplificador.

Una vez que la información sobre la calibración es recibida por el *datalogger*, éste almacena dicha información en memoria hasta que, previo al envío de los datos almacenados, le sea requerida por la PC. Con este fin, el *datalogger* tiene localidades reservadas dentro de la memoria EEPROM del microcontrolador para el almacenamiento de estas variables. Las variables de calibración son guardadas en un tipo de datos flotante por el *datalogger* pero nunca son realmente utilizadas por éste. La posición en memoria que estas variables tienen se puede ver en la tabla 4.4.

Parámetro	**Tipo de Dato**	**Dirección**
Dirección de inicio de datos	Entero	0x00
Número de memoria de inicio de datos	Chart	0x02
Tipo de adquisición	Chart	0x04
Tipo de guardado	Chart	0x05
Bandera de fin de memoria alcanzado	Chart	0x06
Período de registro temperatura	Chart	0x10
Identificador del *datalogger*	Arreglo chart	0x16
Capacidad del pluviómetro	Float simple	0x25
Ganancia del amplificador	Float simple	0x29
Offset de la sonda de temperatura	Float simple	0x33
Bandera de calibración	Chart	0x38

Tabla 4.4. Estructura de los datos almacenados en la memoria EEPROM del microcontrolador.

4.5.6. Modo de prueba

El modo de prueba le permitirá al usuario verificar que los sensores se encuentren funcionando y que estén obteniendo mediciones válidas antes de

que el *datalogger* sea puesto a adquirir. Lo anterior adicionalmente ayuda al usuario a detectar posibles problemas con los sensores.

Este modo de prueba se realiza, para la sonda de temperatura, leyendo el valor del convertidor A/D cada vez que se recibe un comando de la computadora y enviando el resultado de dicha conversión a ésta. En cuanto al pluviómetro, un comando es enviado a la PC cada que el balancín vuelca indicándole a ésta que un evento de precipitación pluvial ha ocurrido.

4.5.7. Configuración del *datalogger*

A través de la computadora es el único medio por donde el usuario puede configurar los parámetros de operación del sistema. Estos parámetros son los siguientes:

- Establecer la hora del reloj en tiempo real (RTR)
- Tipo de variables a adquirir: sólo temperatura, sólo precipitación pluvial o ambas
- Período de registro de la temperatura
- Modo de operación de la memoria: parar adquisición al llenarse o reescribir registros antiguos
- Modo de inicio: inmediato, al presionar el botón principal o programado por tiempo

Aún así, como ya se mencionó, para el inicio en sitio se establecieron parámetros predefinidos que son:

- Fecha y hora del RTR: 01 de Enero del 2009 a las 0:00:00 hrs
- Tipo de variables a adquirir: ambas
- Período de registro de la temperatura: cada cinco minutos
- Modo de operación de la memoria: parar adquisición cuando ésta se llene
- Modo de inicio: al presionar el botón principal

Todos estos parámetros son cargados una vez que el *datalogger* es energizado por primera vez, por lo que se almacena en la memoria EEPROM del microcontrolador, ver tabla 4.4.

4.6. Integración del *firmware*

En la figura 4.17 se muestra la estructura que se siguió para la programación del *firmware*. Partiendo desde la parte de arriba del diagrama, se desarrollaron funciones para el manejo de los módulos del microcontrolador: el convertidor A/D, la memoria EEPROM interna del microcontrolador y los módulos de comunicación I^2C y USART. A partir de las funciones que hacen el manejo de estos módulos se desarrollaron funciones más complejas, tanto para el manejo de los componentes externos al microcontrolador como para el desarrollo de las funciones de organización de memoria, registro de datos, etc. Una vez que las funciones más generales fueron creadas, se integraron al archivo principal. Es en este archivo en donde se definen los vectores de interrupción, comandos de comunicación y funciones adicionales, éstas últimas dada su sencillez no requirieron desarrollarse a partir de otras.

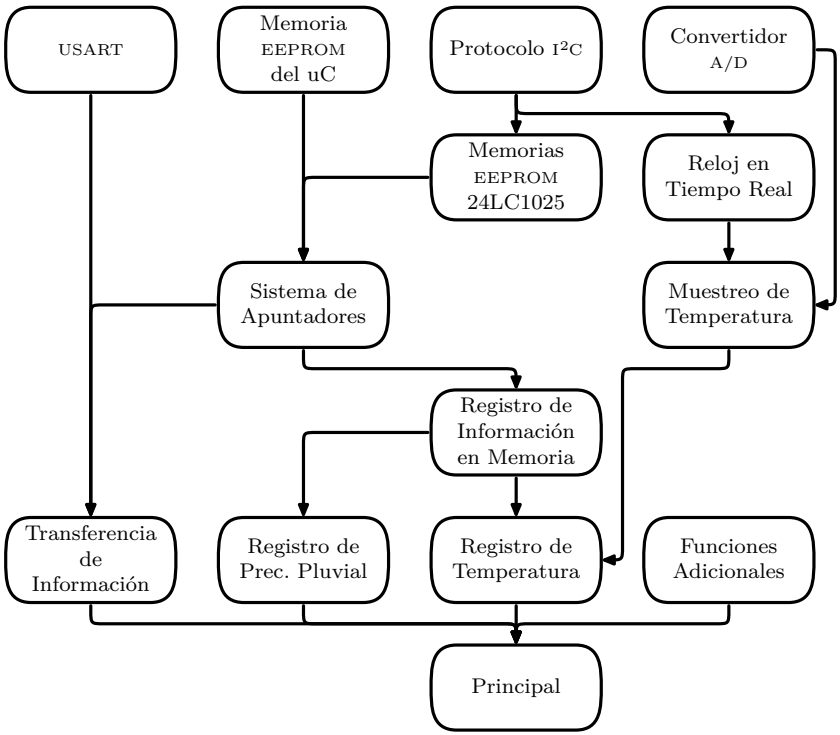


Fig. 4.17. Estructura del *firmware*.

Una vez hecho lo anterior, se compiló el programa y se programó en el microcontrolador para que a partir de la ejecución de éste el microcontrolador pusiera en funcionamiento al *datalogger*.

En el siguiente capítulo estudiaremos el programa de la computadora que al ejecutarse en una PC realizará la recepción de los datos del *datalogger* y los manejará para poder ser presentados al usuario.

Capítulo 5

Desarrollo del Software

En este capítulo se presentará el desarrollo y la descripción del programa de computadora que se implementó para recibir las mediciones realizadas por el *datalogger*. En el desarrollo de este capítulo se mostrarán las pantallas de la parte gráfica del programa y se describirá el funcionamiento que los controles realizan. Debido a cuestiones de espacio es imposible mostrar cómo se implementó cada una de las características del software. Por lo anterior, sólo nos enfocaremos en las partes que tienen que ver con la importación y manejo de la información, mientras que las otras características del programa sólo se describirán de manera breve.

5.1. Estructura general del programa

Las funciones principales del programa son las siguientes:

- Recibir los datos provenientes de los sensores
- Permitir el guardado de los datos adquiridos
- Exportación los datos a otras aplicaciones
- Permitir configurar la operación del *datalogger*

El programa fue desarrollado en el lenguaje C# sobre el Frameworks 3.5 de Microsoft, incluidos dentro de Visual Studio Professional 2008. Bajo esta plataforma, la creación de programas con interfaz gráfica se realiza mediante dos pasos: primeramente se crean los elementos gráficos del programa y posteriormente mediante código en C# se programa las acciones que estos controles generarán al ser operados por el usuario.

El programa tiene como modelo los programas con interfaz gráfica mayoritariamente desarrollados bajo esta plataforma. En dicho modelo, los programas consisten en una ventana principal y una serie de menús que ejecutan comandos sobre la información presentada en dicha pantalla o que abren nuevas ventanas para realizar funciones adicionales.

En el desarrollo del programa se decidió que la información meteorológica y su manejo debían estar incluidas en la ventana principal y las funciones de entrada y salida de la información debían ser contempladas como opciones de menús o ser implementadas en funciones adicionales.

En la figura 5.1 se muestra un diagrama en donde se divide el tipo de operaciones tanto de entrada, manejo y salida de la información meteorológica.

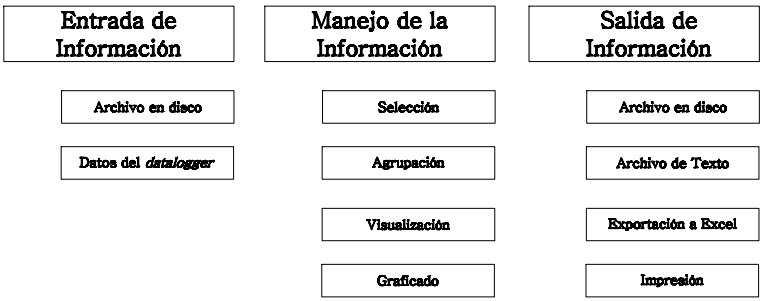


Fig. 5.1. Operaciones sobre la información meteorológica

La entrada de información tiene dos origenes:

- Un archivo en disco
- Datos provenientes de *datalogger*

El archivo en disco es un archivo binario que fue creado anteriormente por el programa para guardar mediciones obtenidas de un *datalogger*. Con lo anterior, se considera que toda información manejada por el programa tiene como único origen mediciones realizadas en algún momento por un *datalogger*. El programa realizado no puede abrir datos que no hayan sido generados de esta manera y no permite edición por parte del usuario de los datos una vez abiertos.

Una vez que la información es ingresada al programa, el manejo que se le puede dar a ésta es el siguiente:

- Selección

- Agrupación
- Visualización
- Graficado

Este manejo tiene como función que el usuario mediante la visualización, la agrupación y el graficado, pueda seleccionar la información que le es de utilidad para exportarla.

Finalmente, ya sea que el usuario desee exportar la información seleccionada o simplemente almacenarla para otra ocasión, éste tiene las siguientes opciones dentro del programa para la salida de dicha información:

- Archivo binario
- Archivo de texto
- Exportación a Microsoft Excel (MS Excel)
- Impresión

El archivo binario permite que el programa pueda abrir la información que se está manejando en otra ocasión. El archivo de texto permite crear un archivo que contiene las mediciones realizadas en un formato ASCII, el cual puede ser abierto por cualquier procesador de textos o servir para ser importado por otros programas. La exportación a MS Excel permite directamente escribir los datos que se están manejando en una hoja de MS Excel y finalmente, la opción de impresión, permite obtener un reporte en papel de la información.

Una vez tomadas las consideraciones sobre el uso de la información dentro el programa, la estructura general de éste se muestra en el diagrama de la figura 5.2. La ventana principal, la cual es la primera que se presenta una vez ejecutado el programa, contiene un conjunto de tablas en donde se muestran los datos que se están manejando en ese momento, los controles encargados del manejo de estos datos y un conjunto de menús para el acceso a funciones adicionales.

Los menús contienen diferentes opciones o herramientas que el usuario puede emplear. Las opciones de exportación y graficado de la información realizan las funciones de manejo de información no presentes en los controles de la pantalla principal. Las opciones para el manejo de los archivos permiten realizar acciones como abrir, crear y cerrar un archivo. Adicionalmente a la información meteorológica, los archivos poseen información concerniente a los datos del *datalogger* que los adquirió, esta información puede ser modificada

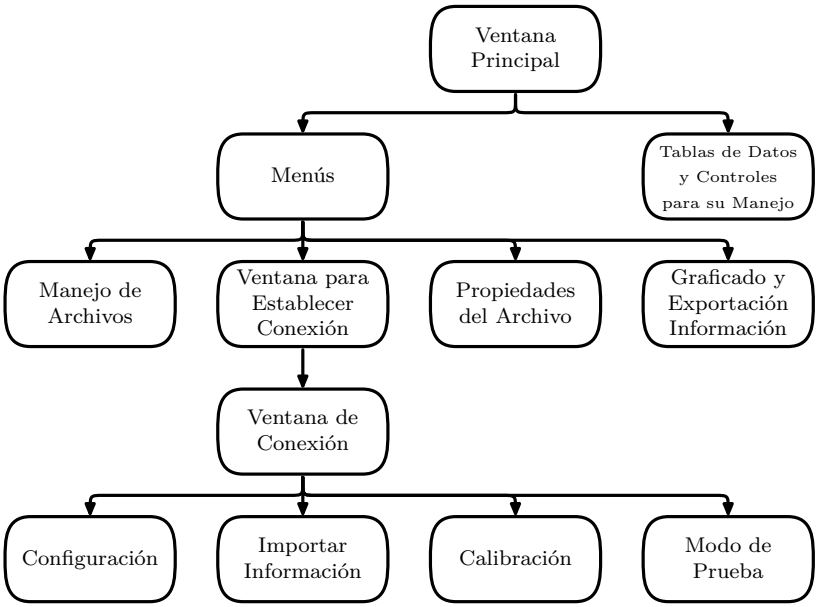


Fig. 5.2. Estructura del programa.

mediante la opción de propiedades del archivo. Finalmente, se presenta la opción para realizar la conexión con un *datalogger*, mediante esta conexión, aparte de realizar la importación de información del *datalogger*, se pueden acceder a todas las funciones de configuración, calibración y el modo de prueba que éste posee.

5.2. El manejo de información

El programa realiza el manejo de la información mediante dos elementos:

- Clases y
- Archivos

Las clases fueron usadas para integrar la información dentro de la ejecución del programa y los archivos para almacenar la información una vez que el programa deja de ejecutarse.

Una clase es un molde el cual agrupa no sólo tipos de datos internos sino también incluyen funciones que interactúan sobre esta información.

Para este programa se creó una clase por cada variable medible, esta clase contiene de manera básica dos valores, la fecha en que se tomó la medición y el valor de esa medición. Un diagrama de esta estructura se muestra en la figura 5.3. Para el caso del valor de la medición se utilizó un valor flotante de doble precisión, en el caso de la fecha y hora, el Framework maneja un tipo de dato especial llamado *DateTime*. Este tipo de dato es bastante conveniente ya que posee funciones y propiedades de gran utilidad como son: sumar un valor de tiempo, convertir el valor de fecha y hora en una cadena de caracteres, etc.

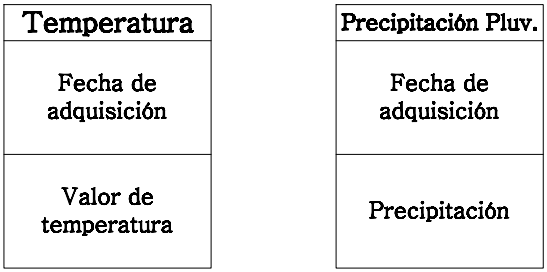


Fig. 5.3. Estructura de las clases.

Las clases sirven para declarar objetos, en nuestro caso, cada medición recibida por el programa es convertido en un objeto o instancia de algunas de las clases anteriores[1]. Estos objetos funcionan como tipos de datos y se pueden agrupar en arreglos u otro tipo de estructuras de datos. En el uso de este programa se utilizaron listas para agrupar cada uno de los objetos declarados. Las listas es un tipo de estructura de datos parecido a un arreglo unidemensional, pero cuya capacidad no se encuentra fija en su declaración y pueden ir creciendo conforme se le agregan elementos.

Mediante estas listas se crearon funciones que tienen como entrada estas listas de objetos y que permiten realizar las funciones de manejo de datos dentro del programa. Estas funciones abarcan desde el ordenar los valores contenidos en dichas listas hasta guardar el contenido de éstas en un archivo.

Los archivos es el modo de manejo de información usado cuando se desea que los datos presentes en el programa estén disponibles para un futuro. El archivo creado por el programa no sólo contiene información de las mediciones hechas por el *datalogger* sino también información del *datalogger* que realizó estas mediciones. La idea detrás de esto es que el usuario pueda guardar en

[1] El término objeto e instancia de clase son similares en la programación orientada a objetos.

un archivo las mediciones realizadas por un determinado *datalogger* y después abrirlo para guardar en éste la información colectada por este mismo *datalogger* en un momento posterior. Con lo anterior se logra que en un archivo se almacene toda las mediciones hechas por un *datalogger* a través del tiempo. La idea de usar un archivo para cada *datalogger* permite que el programa se convierta en una herramienta no sólo para extraer la información colectada por un *datalogger* sino también, para el manejo de esta información.

5.3. La pantalla principal

En la figura 5.4 se puede ver la pantalla principal del programa. Como las pantallas principales típicas de los programas desarrollados con interfaz gráfica, éste posee en la parte superior una barra de menús marcada con el número ❶. Estos menús agrupan todos los comandos disponibles por el usuario para acceder a las funciones del programa. Las opciones presentadas en la barras de menús se pueden ver en las tablas 5.1.a y 5.1.b.

Archivo	
Opción	**Función**
Nuevo	Crea un nuevo archivo
Abrir	Abre un archivo
Guardar	Guarda el archivo
Guardar como	Guarda el archivo y pide se especifique el lugar
Imprimir	Imprime las mediciones abriendo un cuadro de diálogo
Configuración página	Configura la página a imprimir
Vista previa de impresión	Muestra la vista previa de las hojas impresas
Salir	Sale del programa
Herramientas	

Tabla 5.1.a. Funciones de los menús de la pantalla principal.

Opción	**Función**
Graficar	Gráfica los datos seleccionados
Opciones	Modifica las opciones del archivo
Conectar	Abre la ventana de conexión
Exportar	
Opción	**Función**
Microsoft Excel	Exporta los datos a MS Excel
Archivo de Texto	Exporta los datos a un archivo de texto
Ayuda	
Opción	**Función**
Contenido	No implementado
Índice	No implementado
Buscar	No implementado
Acerca de	Despliega los créditos del programa

Tabla 5.1.b. Funciones de los menús de la pantalla principal.

Adicionalmente a los menús, la barra de herramientas, indicada con el número ❷, contiene accesos rápidos a las funciones de mayor uso en el programa y se identifican a través de íconos.

En la parte central de la ventana, marcado con el número ❸, se encuentra un control con pestañas. Este control permite al usuario seleccionar con qué tipo de variable desea trabajar, ya sea temperatura ambiente o precipitación pluvial. La región marcada con el número ❹ es muestra los valores de las mediciones adquiridas en forma tabular. Las mediciones mostradas en estas tablas se encuentran ordenadas de manera cronológica y sirven para que el usuario pueda visualizar los valores que están manejando. Los valores se muestran en esta región serán los que se graficarán, exportarán e imprimirán cuando estas funciones sean ejecutadas.

En los cuadros de texto marcados con el número ❺ se muestra información importante del *datalogger*, como son: la ubicación en la que éste se encuentra instalado y su identificador. El identificador es la variable que permite al programa discriminar entre los datos adquiridos por un *datalogger* u otro. De esta manera, cuando el usuario desea descargar datos de un *datalogger* sobre el archivo de mediciones de otro, el programa manda una señal de aviso al usuario para que éste considere la acción. Estos valores pueden ser cambiados en el menú Herramientas →Opciones.

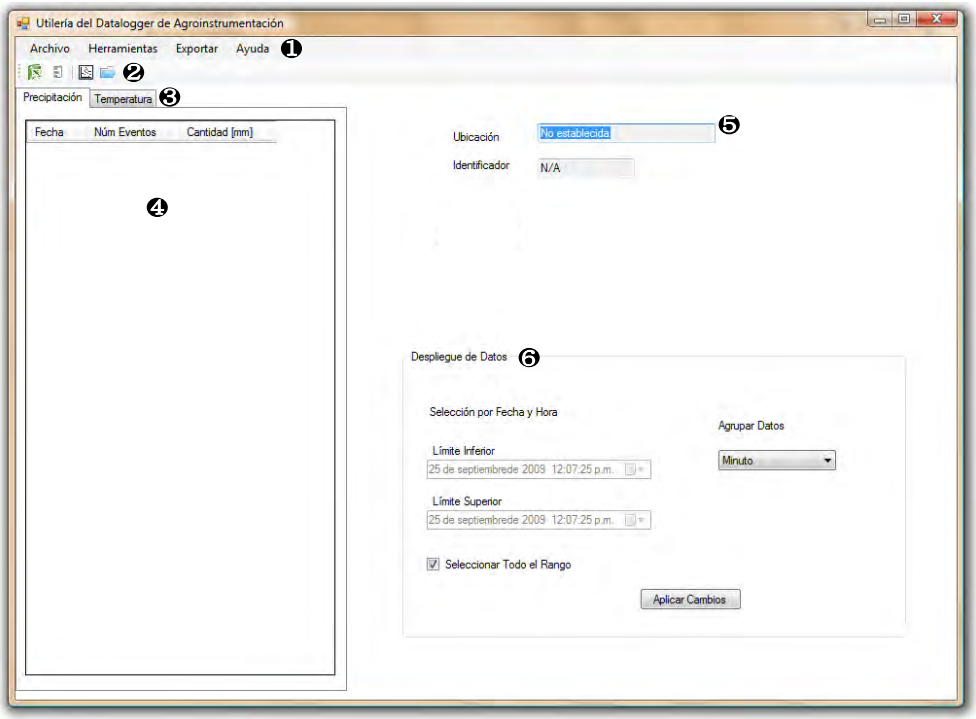


Fig. 5.4. Pantalla principal del programa.

Por último, los controles indicados con el número ❻, el usuario puede seleccionar o agrupar los datos de su interés. Dado que las mediciones se encuentran ordenadas de manera cronológica, el usuario puede seleccionarlas en un rango de fechas y posteriormente agruparlas en minuto, hora, día, mes o año. De esta manera, por ejemplo, de un conjunto de mediciones, el usuario puede seleccionar las mediciones realizadas en determinado mes y agruparlas por día. Esta agrupación y selección es reflejada de manera automática en le región tabular de esta ventana.

Una vez seleccionada y agrupada la información es posible hacer sobre ella las siguientes acciones:

- Imprimirla
- Graficarla o
- Exportarla en un archivo de texto o a MS Excel

En las siguientes subsecciones describiremos cada una de estas funciones de manera breve.

5.3.1. Graficado

Una gráfica arroja importante información a primera vista del comportamiento de la variable con respecto al tiempo. Para realizar las gráficas dentro del programa se utilizó un conjunto de librerías llamadas ZedGraph, las cuales permiten realizar distintos tipos de gráficas dentro de una aplicación.

Las gráficas generadas mediante este conjunto de librerías dan diferentes funciones adicionales al usuario, como son: imprimirlas, guardarlas como imagen, copiarlas, acercarlas, etc. Un ejemplo de gráfica generada mediante esta librería y las opciones disponibles en el menú se muestran en la figura 5.5.

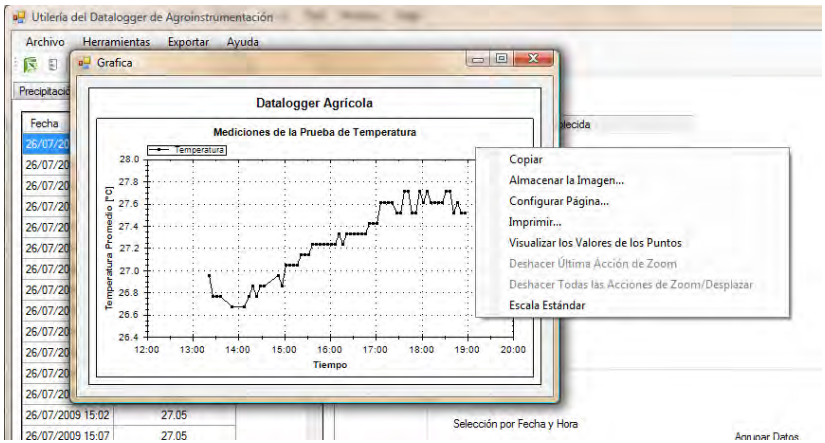


Fig. 5.5. Muestra y opciones de una gráfica.

5.3.2. Impresión

La impresión se realiza de manera muy sencilla, dado que el ambiente posee librerías destinadas para ello. El programa hace uso de estas librerías y crea arreglos de cadenas de texto, las cuales componen las líneas que se desean sean impresas. También, mediante el uso de estas librerías, se implementaron ventanas para la configuración de página, selección de la impresora y vista preliminar de impresión. En la figura 5.6 se muestra la pantalla previa a la impresión de los datos.

5.3.3. Exportación de la información

La opción de exportación de la información se realizó con la intención de que cualquier programa de índole estadístico pudiera importar la información procedente del programa. Para ello se decidió implementar dos formas de

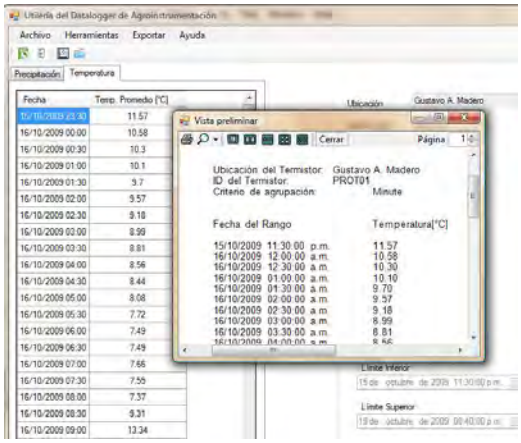


Fig. 5.6. Vista preliminar de impresión.

exportación, la primera, más sencilla de realizar, consiste en crear archivos de texto con los renglones de información separados con saltos de línea y las columnas separadas con comas. Una imagen que muestra este tipo de exportación se puede ver en la figura 5.7. El segundo método exporta los datos al programa MS Excel. Para realizar la exportación a MS Excel, en vez de utilizar un archivo, se utilizó un método para el control de MS Excel desde el programa. Mediante el método se realiza la exportación de la información al escribir directamente en las celdas de una hoja de MS Excel el valor que éstas deben de contener. Una captura de pantalla de este tipo de exportación se puede ver en la figura 5.8.

El método del archivo de texto permite realizar la importación de la información a distintos tipos de programas. En caso de que algún programa no pueda leer los archivos, siempre se tiene la opción de exportar la información a MS Excel y posteriormente a través de éste crear un archivo para exportar los datos a la aplicación de interés.

5.4. Comunicación con el *datalogger*

Una vez concluido con el manejo de la información que realiza el programa, en esta sección nos enfocaremos en la comunicación entre la PC y el *datalogger* a través de este programa. Si bien el programa, mediante su interfaz, oculta

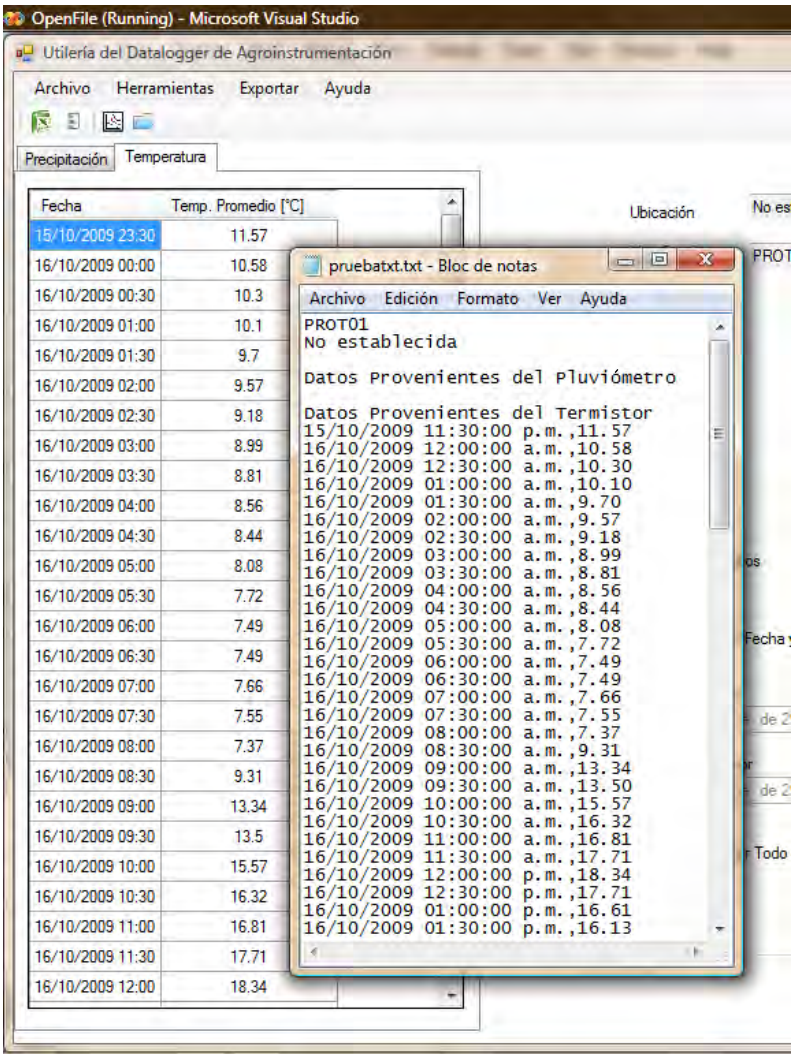


Fig. 5.7. Exportación a un archivo de texto.

al usuario como se entabla dicha comunicación, con fines de mostrar el diseño realizado, se profundizara en ésta.

La comunicación entre el *datalogger* y la PC se realiza mediante una serie de comandos que la computadora manda a través de su puerto serie RS-232 o USB y son recibidos por la USART del microcontrolador. Una vez recibido determinado comando, el *datalogger* realiza la acción que este comando le indica. La comunicación siempre es iniciada por la PC a través de alguno de los comandos mostrados en las tablas 5.2.a y 5.2.b. Todos los comandos son de un

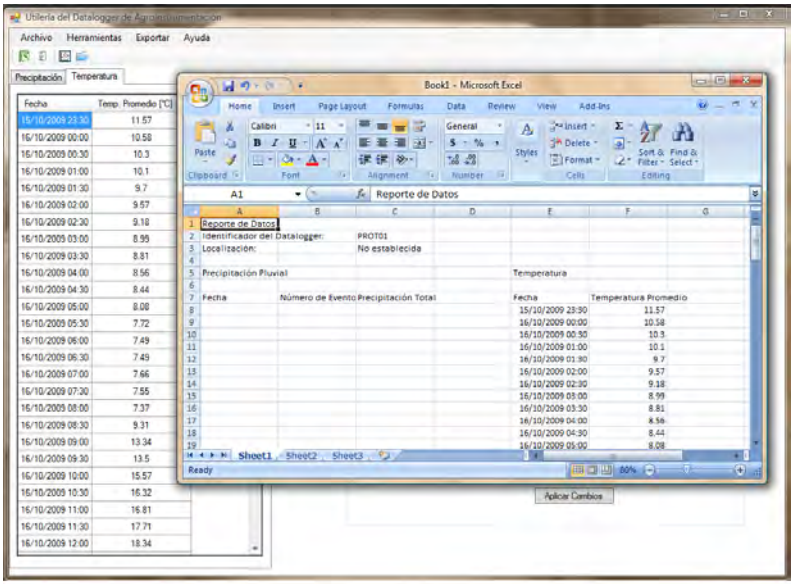


Fig. 5.8. Exportación a MS Excel.

byte de longitud y están definidos por un valor en hexadecimal, aunque todos tienen representación ASCII; si en la tabla se indica determinado comando posee una dirección de entrada (Ent), quiere decir que una vez enviado el comando, el *datalogger* responderá con el número de bytes indicados, si por el contario, la tabla indica determinado comando con una dirección de salida (Sal), una vez enviado el comando se deberán enviar a continuación de éste los bytes siguientes de su argumento. Algunos comandos no tienen respuesta o argumento alguno y se indica mediante un guión.

Comando	**Acción**	**Dirección**	**Longitud [Bytes]**	**Formato del argumento o la respuesta**
40_H	Cancelar comunicación	—	—	—
74_H	Configuración de modo de inicio	Sal	1	ASCII a,b,c
69_H	Tipo de adquisición	Sal	1	ASCII a,b,c
$6D_H$	Envía una página de memoria	Ent	128	—

Tabla 5.2.a. Comandos del *datalogger*.

Comando	**Acción**	**Dirección**	**Longitud [Bytes]**	**Formato del argumento o la respuesta**
34_H	Valor de los apuntadores de memoria	Ent	6	dir. inicio, chip de inicio, dir. fin, chip de fin
$6C_H$	Periodo de adquisición	Sal	2	—
70_H	Escribir valor del RTR	Sal	8	S,Min,Hr,DíaS,DíaM, Mes,Año,Conf
72_H	Leer valor del RTR	Ent	7	S,Min,Hr,DíaS,DíaM, Mes,Año
65_H	Modo de manejo memoria	Sal	1	ASCII a,b
27_H	Identificador del *datalogger*	Ent	Variable	longitud, ASCII
$3F_H$	Señal de conexión	Ent	1	06_H
57_H	Escribir apuntador de inicio	Sal	3	dir inicio, byte de inicio
75_H	Habilita/Deshabilita modo prueba	—	—	—
54_H	Valor de temperatura (modo prueba)	Ent	2	—
53_H	Escribir valores calibración	Sal	12	CPluv, GPrueba, OffPrueba
$5A_H$	Leer valores calibración	Ent	13	CPluv, GPrueba, OffPrueba, Bandera

Tabla 5.2.b. Comandos del *datalogger*.

La utilidad de cada comando se verá en las secciones siguientes, donde se hablará de cada una de las funciones del programa que requieren de comunicación con el *datalogger*.

5.5. Conexión con el *datalogger*

Para que una conexión entre la PC y la computadora pueda ser realizada, el *datalogger* debe de encontrarse en operación, para realizar esto, sólo basta con presionar el botón principal del *datalogger* una vez. La conexión con la PC se realiza una vez que el *datalogger* ha recibido alguno de los comandos mostrados en la tabla 5.2. La comunicación termina una vez que un comando 40_{H} es recibido por el *datalogger* (cancela comunicación). Antes que este comando sea recibido el *datalogger* permanecerá encendido todo el tiempo para poder estar atento a los comandos enviados por la PC.

La comunicación PC–*datalogger* se realiza a través de la ventana mostrada en la figura 5.9 que se despliega mediante la opción del menú Herramientas →Conectar de la ventana principal. En esta ventana el cuadro combo mostrado en ❶ se despliega una lista de todos los puertos COM disponibles en el equipo en el que se esté ejecutando el programa.

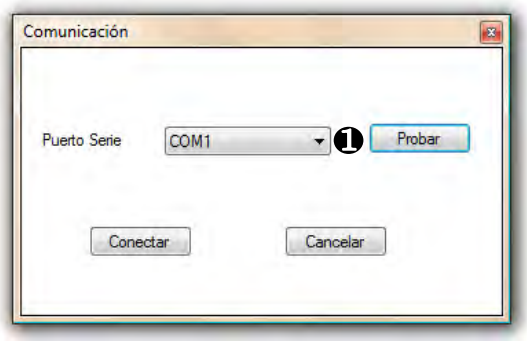


Fig. 5.9. Ventana para establecer una conexión con el *datalogger*.

Una vez que un puerto es seleccionado, éste puede ser probado para saber si un *datalogger* se encuentra conectado a él mediante el botón "Probar" presente en la ventana. Esta prueba se realiza enviando un comando $3\mathrm{F}_{\mathrm{H}}$ (señal de conexión). Si el *datalogger* responde a este comando se determina que un *datalogger* se encuentra conectado. Inmediatamente después del comando $3\mathrm{F}_{\mathrm{H}}$, se envía el comando 27_{H} para requisitar el identificador del *datalogger*. Una vez recibido éste, se muestra mediante un cuadro de diálogo un mensaje que indica que el *datalogger* con el identificador recibido se encuentra conectado

al puerto seleccionado. Si el usuario determina que ese es el *datalogger* al que se desea conectar, una vez cerrado el cuadro de diálogo, éste puede presionar el botón "Conectar" para que la ventana de conexión sea desplegada.

La ventana de conexión con el *datalogger* se muestra en la figura 5.10. Es a través de esta ventana que todas las funciones que el *datalogger* tiene pueden ser seleccionadas. En la parte superior marcada con ❶ se muestra el identificador del *datalogger* al cual se está conectado. A través de las pestañas marcadas con ❷ se tiene acceso a todas las funciones que se pueden realizar con el *datalogger* y las cuales son:

- Configuración
- Importar Información
- Modo Prueba y
- Calibración de Sensores

En el área marcada con ❸ se muestra el contenido de cada pestaña donde se localizan los controles e indicadores para cada función. El cuadro de texto indicado con el número ❹ muestra un registro de todas las acciones realizadas hasta el momento con el *datalogger*. Finalmente, el botón salir, marcado con el número ❺, permite cerrar esta ventana; una vez que la ventana se cierra, un comando 40_{H} es enviado al *datalogger*, por lo que la comunicación con éste cesa y el *datalogger* entra en estado de bajo consumo.

En las siguientes secciones estudiaremos cada una de las funciones que se pueden realizar en esta ventana.

5.5.1. Configuración

La pestaña de configuración se muestra en figura 5.11. En esta pestaña se establece el valor del reloj en tiempo real (RTR) del *datalogger* y el modo en el que el *datalogger* operará una vez que sea iniciada la adquisición.

El valor con el que se programará el reloj en tiempo real se selecciona mediante el control marcado con el número ❶, este control muestra inicialmente la hora y fecha de la computadora, en caso de que este valor sea incorrecto o se desea establecer una fecha y hora distintos se puede programa con otra fecha y hora. Para realizar la anterior, basta con marcar el cuadro de selección indicado con el número ❷, una vez marcado el selector de fecha y hora ❶

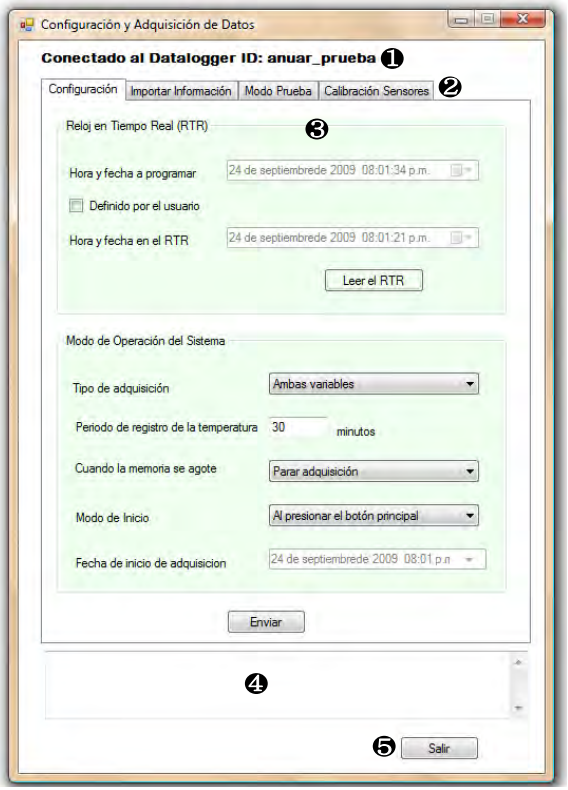


Fig. 5.10. Ventana de conexión con el *datalogger*.

quedará habilitado y se podrá indicar el valor de fecha y hora deseado. Si antes de realizar un cambio o después de escribir sobre el reloj se desea verificar cuál es el valor del RTR, se pude hacer al presionar el botón "Leer el RTR", lee el valor del RTR y muestra el valor recibido en el indicador de fecha y hora marcado con el número ❸.

El procedimiento para la lectura del RTR se realiza enviando el comando 72_{H} al *datalogger*, una vez que el valor de la hora y fecha es recibido, el programa debe realizar una decodificación de las cadenas recibidas que se encuentran en código BCD. El algoritmo que se utilizó para hacer lo anterior es el siguiente:

```
BCD = (byte)((BCD » 4) * 10 + (BCD & (byte)0x0F));
```

El algoritmo toma la parte alta del byte que contienen las decenas y lo multiplica por diez, después toma el valor de las unidades y lo suma a las decenas obteniendo de esa forma el valor en binario del valor BCD de entrada.

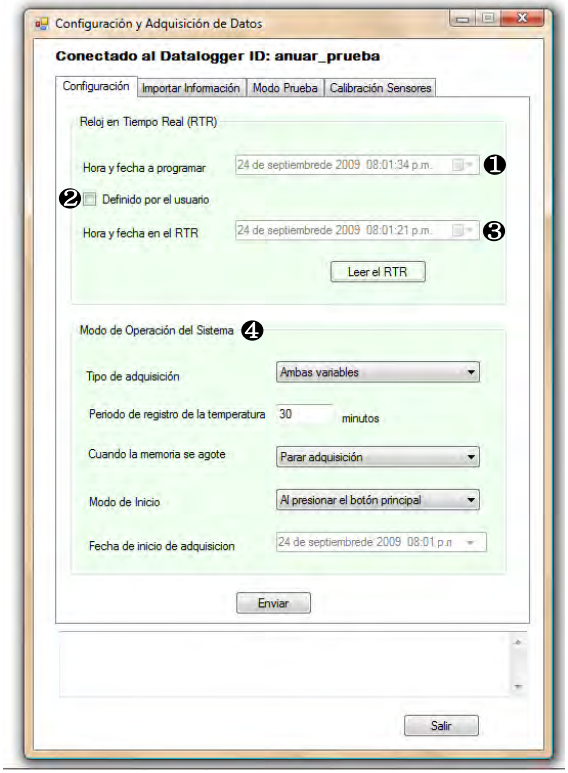


Fig. 5.11. Pestaña de configuración.

Una vez transformado en binario se puede crear un objeto *DateTime* y con éste establecer el valor del control ❸.

Los controles siguientes, marcados con el número ❹, le indican al *datalogger* la manera en que debe de operar. Cada una de las opciones indicadas es transformada en el parámetro de alguno de los comandos mostrados según se muestra en la tabla 5.3.

El tipo de adquisición indica qué variables se desea que el *datalogger* adquiera una vez iniciada su adquisición. El período de registro de la temperatura indica cuál es el período de registro deseado para esta variable, dados los dos bytes disponibles para su valor, éste puede establecerse desde uno hasta 65535 minutos (18.2 días aproximadamente). El parámetro "cuando la memoria se agote", indica cuál va ser la operación del *datalogger* en caso de que éste llene todo el espacio en memoria que posee. Finalmente, el modo de inicio especifica el evento que ocasionará que el *datalogger* inicie su adquisición. En caso de que el usuario elija inicio programado por alarma, éste deberá especificar

el día, la hora y el minuto en que desea que la adquisición comience, dichos valores se envían después de especificar este modo.

Parámetro	Comando	Tipo de dato	Argumento
Tipo de adquisición	69_H	ASCII	a ⟶ sólo temperatura
			b ⟶ sólo precipitación pluvial
			c ⟶ ambas variables
Período de registro temperatura	$6C_H$	númerico	2 bytes
Cuando la memoria se agote	65_H	ASCII	a ⟶ detener adquisición
			b ⟶ sobreescribir registros
Modo de inicio	74_H	ASCII	a ⟶ inicio inmediato
			b ⟶ al presionar botón principal
			c ⟶ inicio programado por alarma + 3 bytes de fecha de inicio (ver abajo)
Fecha de inicio	—	numérico BCD	minuto, hora, día del mes

Tabla 5.3. Parámetros de configuración.

Cuando el botón "Enviar Parámetros" es presionado, el programa convierte los valores seleccionados por el usuario en el formato correcto para enviarlos al *datalogger*. Cada parámetro es enviado mediante su comando y los argumentos de éste. Este mismo botón envía el nuevo valor del fecha y hora al RTR del *datalogger*. Debido a que este valor debe ser programado en código BCD, una transformación de números binarios a este código debe ser realizada. Para realizar la conversión anterior se utilizó la siguiente operación:

```
Numm[i] = (byte)((((byte)(Numm[i] / 10)) « 4) + Numm[i] % 10);
```

Esta operación obtiene el valor de las decenas del elemento `Numm` actual y lo recorre a la posición de las decenas del código BCD. Después, mediante la operación módulo, se obtiene el valor de las unidades y se la suma a las decenas para obtener de esa forma la representación del código BCD del número de entrada.

5.5.2. Importación de la información

La importación de la información es el proceso por el cual el programa recibe los datos guardados en el *datalogger* y los procesa para ser manejados por el usuario. En la figura 5.12 se muestra la pestaña donde el usuario podrá acceder a esta función sobre el *datalogger*.

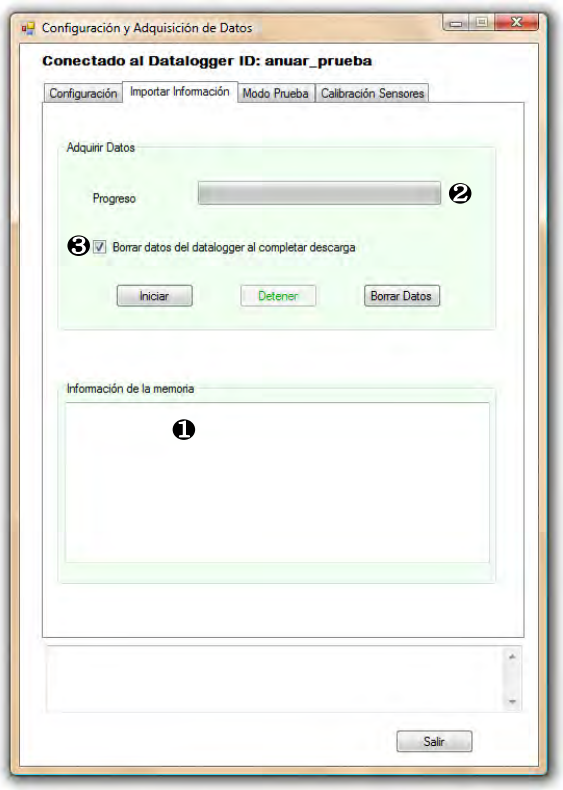


Fig. 5.12. Pestaña de importación de información.

La importación de los datos por parte del usuario se realiza de manera sencilla. Una vez que la ventana de conexión es abierta, el programa solicita las direcciones de inicio y fin de datos mediante el comando 34_{H}. La información de las direcciones es interpretada para determinar la cantidad de espacio usado y disponible en las memorias del *datalogger*. Esta información es mostrada en el cuadro marcado con el número ❶ para que el usuario también se encuentre al pendiente de ésta. Si la memoria no se encuentra vacía, en el momento que el usuario presione el botón “Iniciar” la transferencia de información comenzará.

El progreso de esta transferencia se irá mostrando mediante el incremento de la barra de progreso marcado con el número ❷.

El procediendo de recepción de datos comienza cuando el programa envía el comando $5A_H$ (leer valores de calibración) al *datalogger*, éste responde enviando los parámetros de calibración y la bandera de calibración. La bandera de calibración sirve para que el programa verifique si el usuario calibró los sensores antes de comenzar a adquirir datos, si así fue, el programa determina que los valores de calibración son válidos y los utiliza, en caso contrario, el programa utilizará los valores de calibración por defecto, los cuales se muestran en la tabla 5.4.

Parámetro	**Valor**
Capacidad del pluviómetro	0.5 [mm]
Ganancia del amplificador	282
Offset de la señal	0 [°C]

Tabla 5.4. Valores por defecto de calibración.

Si los parámetros de calibración son válidos el programa adecuará éstos en números flotantes a partir de los cuatro bytes recibidos por parámetro.

Una vez que el programa tenga los parámetros de calibración que usará para adecuar los valores de las mediciones, éste envía el comando $6D_H$ (envía una página de memoria) al *datalogger*. Recibido éste comando, el *datalogger* entra en un estado de envío de información. El proceso de envío de información se realiza de acuerdo al diagrama de flujo de la figura 5.13. Como se muestra en el diagrama, recibida una página de información, el *datalogger* la procesa. Cuando se ha terminado de procesar una página de información el programa envía otro comando $6D_H$ para recibir la siguiente página a procesar. El proceso anterior continúa hasta que el programa determina que todas las páginas con información han sido recibidas.

Procesamiento de las páginas

Una vez que una página de información ha sido recibida por el programa, éste debe de interpretar la información para adecuar los valores de las mediciones hechas por el *datalogger*.

En el capítulo sobre el programa del microcontrolador se estudió la estructura de cada página de información creada por el *datalogger*. El *datalogger* crea dos tipos de páginas de datos, cada tipo guarda información de una variable adquirida y estas tienen una longitud de 128 bytes. El proceso de recuperación

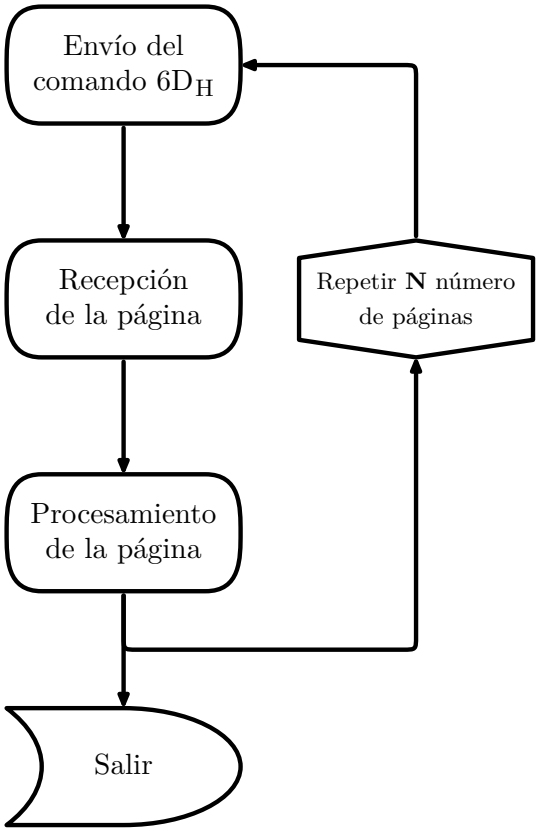


Fig. 5.13. Proceso de envío de información.

de información inicia al leer el valor de la primer byte de la página, a través de este valor el programa determina si la información siguiente pertenece a mediciones de temperatura ambiente o mediciones de precipitación pluvial.

En la figura 5.14 se muestra una secuencia de datos proveniente de una página que contiene información de mediciones hechas de un pluviómetro. El primer byte de cabecera nos indica con el valor de 10_H que la información contenida proviene del pluviómetro. Los siguientes dos bytes no contienen información alguna, como ya se mencionó sólo sirven para completar los 128 bytes de una página, por lo que son descartados. Una vez que se identificó que la información es de precipitación pluvial, el proceso de recuperación comienza. Se empieza tomando los primeros cinco bytes de información y se

crea con estos valores un objeto *DateTime*, que representa la fecha en el que el *datalogger* recibió un evento del pluviómetro de balancín.

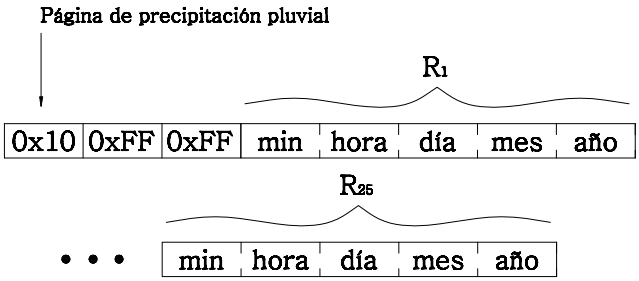


Fig. 5.14. Información de precipitación pluvial.

Para completar la información de una medición, se requiere la cantidad de precipitación que produjo el volcado del balancín. Esta información la obtenemos de los valores de calibración recibidos por el *datalogger*. Con el objeto *DateTime* y el valor de la precipitación se crea una instancia de la clase *pluviómetro*. Esta instancia es guardada en una lista que contiene todas las mediciones realizadas en esta sección de recuperación de datos. El proceso de recuperación de la información continua con los siguientes cinco bytes y prosigue hasta que se alcance el final de las mediciones. El final de las mediciones ocurre cuando el total de bytes de la página es agotado o cuando un registro comienza con el valor FF_H, si esto último sucede, se considera que dicha página no alcanzó a llenarse cuando el *datalogger* estaba adquiriendo.

Para el caso de la recepción de las señales de la sonda de temperatura, se requiere de un proceso un poco más complejo. En la figura 5.15 se muestra la información de una secuencia de datos procedentes de una página con mediciones de temperatura ambiente.

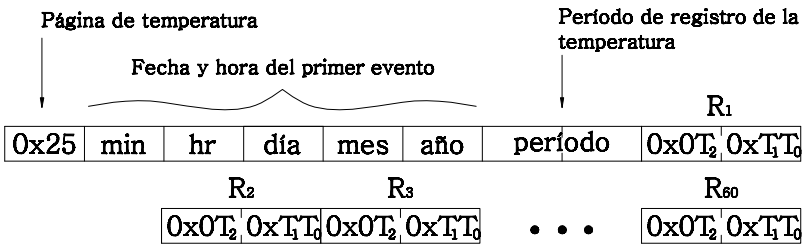


Fig. 5.15. Información de temperatura ambiente.

Las páginas que contienen información de temperatura ambiente son identificadas con el valor 25_H en el primer byte de su cabecera. Una vez que se

identifica que una página contiene información de temperatura ambiente, se lee de su cabecera el período en que se registraron las mediciones de temperatura. Adicionalmente, con la información de la cabecera, se crea un objeto *DateTime*, que representa el valor de la fecha y hora en el que la primera medición de temperatura fue registrada. Este objeto servirá para estampar las siguientes mediciones tomadas al irse incrementando con el período de registro. Después de tomar estos datos, la información de la cabecera se elimina y se hace un barrido de la región de datos en intervalos de dos bytes. Cada dos bytes representa una medición de la tensión de salida de la sonda de temperatura realizada por el convertidor A/D del microcontrolador. Para transformar este valor en temperatura ambiente se utilizan los siguientes parámetros:

- El valor de la referencia de tensión del microcontrolador
- La amplificación a la que fue sujeta la tensión de salida de la sonda de temperatura
- Los parámetros de la ecuación Steinhart-Hart del termistor de la sonda
- El *offset* que la sonda de temperatura posee con respecto al patrón de temperatura

Como se vio en capítulo 4, tanto el valor de la tensión de referencia del microcontrolador como los parámetros de la ecuación Steinhart-Hart se consideran constantes, por lo que estos son guardados como tal por el programa. Los otros valores, el *offset* de la sonda de temperatura y la amplificación de la señal de temperatura, son recibidos cuando se requisitan los parámetros de calibración del *datalagger*.

Con todos los valores necesarios ya disponibles, se realiza el proceso de transformar el valor del convertidor A/D en un valor de temperatura ambiente, el proceso que se sigue para ello es el siguiente:

- Obtener el valor de tensión representado por el valor del convertidor A/D
- Dividir este valor entre la ganancia del amplificado operacional
- Obtener el valor de resistencia del termistor mediante la ecuación de la sonda de temperatura
- Con el valor de resistencia obtener la temperatura del termistor mediante la ecuación de Steinhart-Hart
- Al valor de temperatura obtenido restar el *offset* de temperatura de la señal

El proceso descrito se presenta en el diagrama de flujo de la figura 5.16.

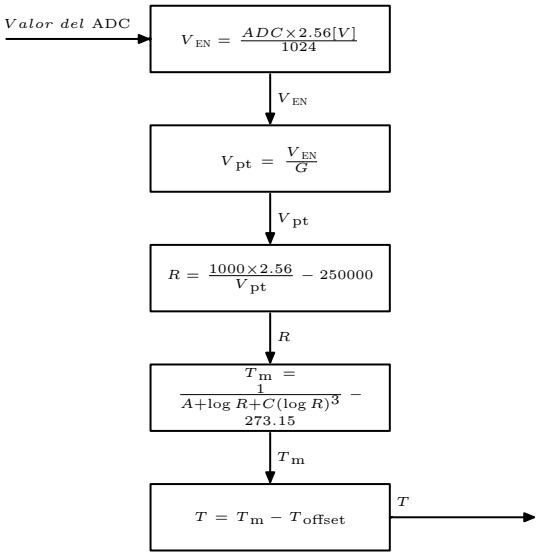


Fig. 5.16. Proceso de conversión de la temperatura ambiente.

Una vez que el valor de temperatura se ha obtenido, sólo es necesario poner su estampa de tiempo, para ello se toma el valor del objeto *DateTime* base y se le suma el número de minutos que pasaron hasta tomar esta medición. El número de minutos agregados (N_{min}) se toma con base en el número de medición (N_{med}), mediante la expresión 5.1.

$$N_{min} = (N_{med} - 1) \times P_{med} \tag{5.1}$$

donde P_{med} es el período de registro, valor que es tomado de la cabecera de la página.

De esta manera se tiene el valor de la temperatura y el momento en que se registró. Con la información anterior se crea la instancia de la clase *Temperatura* y se guarda en una lista que contiene las mediciones recibidas en esta sección. El proceso de lectura de la página continúa hasta que se alcanza el final de las mediciones. El fin de las mediciones sucede, al igual que en el caso de la precipitación pluvial, cuando todos los bytes de la página se agotan o cuando un registro comienza con el valor FF_H.

Borrado de datos

Como ya se había mencionado en el capítulo 4, el *datalogger* no borra los datos una vez que el usuario ya no los desea. En su lugar, para descartar la

información ya no deseada, el *datalogger* modifica el apuntador de inicio de datos para dejar los registros con información no deseada fuera de la región de registros válidos.

En el caso del programa desarrollado, esta operación de modificación del apuntador de inicio se realiza por medio de la casilla de verificación marcada con el número ❸ en la ventana de la figura 5.12 y con el botón "Borrar Datos", presente en esta misma ventana. Cuando la casilla es marcada durante la recepción de datos del *datalogger*, el programa, una vez que ha recibido todas las páginas con información de éste, envía el comando 57_{H} (escribir apuntador de inicio) al *datalogger* para fijar la nueva dirección del apuntador de inicio. Como argumento para este comando, el programa utiliza el valor del apuntador de fin de datos que recibió cuando la ventana de conexión fue abierta. De esta manera una vez establecido el apuntador de inicio de datos con dicho valor el *datalogger* aparenta que ha borrado su memoria.

El botón "Borrar Datos" realiza la operación de escritura del apuntador de inicio de datos de manera inmediata, esto sin la necesidad de haber descargado previamente los datos almacenados en el *datalogger*.

5.5.3. Modo prueba

La figura 5.17 muestra la pantalla para controlar el modo de prueba del *datalogger*.

El modo de prueba comienza cuando el programa envía el comando 75_{H} al *datalogger*, este comando es enviado cuando el botón "Entrar modo de prueba" que se encuentra en la ventana es presionado. Una vez que recibe el *datalogger* este comando espera que sucedan dos cosas, recibir un comando 54_{H} de la computadora o detectar una señal del volcado de su balancín.

Si un comando 54_{H} es recibido el *datalogger* responde enviando el valor de una medición de temperatura a la PC, con esta medición el programa realiza una conversión de ésta a temperatura y lo muestra, tanto en la gráfica de temperatura, marcada con el número ❶ en la figura ya mencionada, como escribiendo el valor de la temperatura en el cuadro de texto marcado con el número ❷. El período de actualización del valor de la temperatura en el modo prueba es controlado por el programa de la PC a través de un contador que se encuentra establecido de manera fija en 10 segundos. El botón de "Pausar" detiene el contador el cual se puede reanudar al presionar de nueva cuenta este mismo botón que habrá cambiado su etiqueta a "Reanudar".

Si el balancín es volcado por el usuario para probarlo, el *datalogger* envía la secuencia de bits $\mathrm{FF_H}$–01_{H} a la PC. Este valor le indica al programa que un

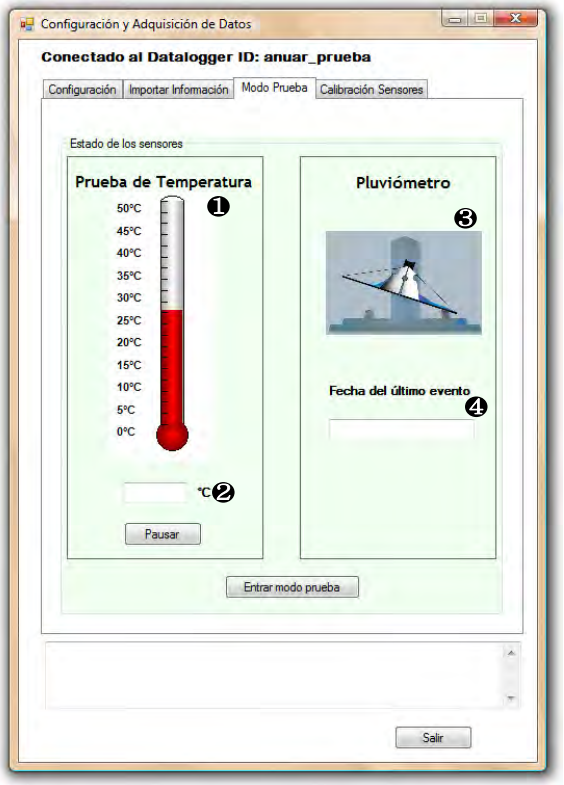


Fig. 5.17. Pestaña de modo de prueba.

volcado ha ocurrido y se lo comunica al usuario cambiando la figura indicada con el número ❸ y escribiendo la fecha y hora en el que el evento ocurrió en el cuadro de texto indicado con el número ❹.

El modo de prueba termina cuando el usuario presiona de nuevo el botón "Entrar modo de prueba" (el cual habrá cambiado su etiqueta por "Salir modo de prueba") que se encuentra en la ventana, cambie de pestaña o salga de la ventana de comunicación. Esta salida se realiza mediante el envío de nueva cuenta del comando 75_{H} al *datalogger*.

5.5.4. Calibración de sensores

En la figura 5.18 se muestra la ventana en donde se realiza la calibración de los transductores.

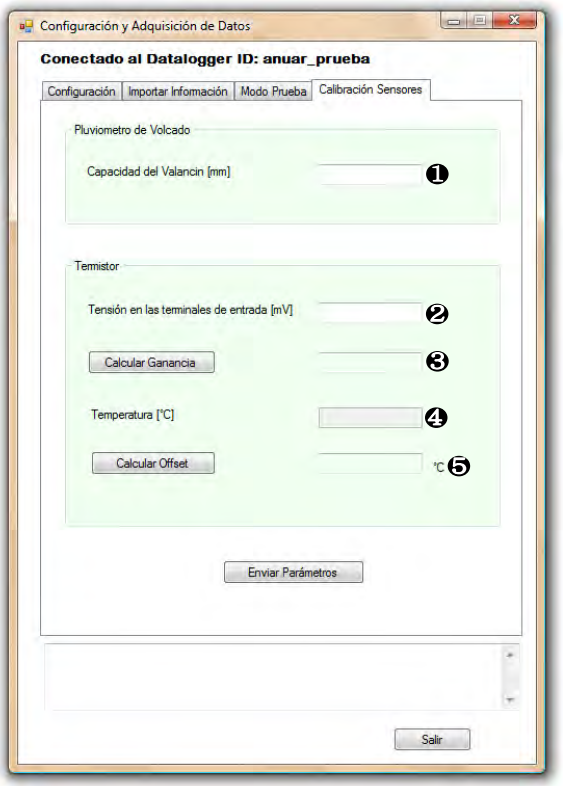


Fig. 5.18. Pantalla de calibración.

La calibración del pluviómetro se realiza proporcionándole al programa en el cuadro de texto indicado con el número ❶ el valor de volcado del balancín conectado al *datalogger*.

La calibración de la sonda de temperatura es un proceso que se realiza comparando mediciones realizadas por el *datalogger* y por el usuario de dos variables: la tensión de salida de la sonda de temperatura y la temperatura ambiente. Primeramente, se requiere que con un multímetro se tome la tensión a la salida de la sonda de temperatura. Una vez tomada la medición, ésta debe ser introducida en el cuadro de texto marcado con el número ❷ y el botón "Calcular Ganancia" debe ser presionado. En ese momento, el programa solicita el valor de la conversión A/D hecha por el *datalogger*. Este valor es transformado por el programa en un valor de tensión que representa la señal de la sonda de temperatura multiplicada por la ganancia del amplificador. Con este valor y el valor entregado por el usuario, el programa calcula la ganancia a la que está sujeta la sonda de temperatura y lo escribe en el

cuadro marcado con el número ❸. Una vez hecho esto, con un termómetro patrón cuyo error debe ser menor al de la sonda, el usuario debe introducir la temperatura ambiente en grados centígrados en el cuadro de texto marcado con el número ❹. En ese momento, el programa solicita de nueva cuenta un valor del convertidor A/D y mediante la ganancia calculada en el paso anterior obtiene un valor de temperatura. El *offset* de la sonda se calcula con la diferencia entre este valor y el valor proporcionado por el usuario y se escribe en el cuadro de texto marcado con el número ❺.

El proceso de calibración de ambas variables finaliza enviando el comando 53_{H} al *datalogger*, seguido de los parámetros de calibración introducidos y calculados para los transductores, para hacer esto, se separan los valores de tipo flotante en cuatro bytes, que son enviados para ser guardados en la memoria del *datalogger* y que se solicitarán al momento de recibir los datos adquiridos.

5.6. Integración del programa

Una vez que todo el programa fue desarrollado, la integración consistió en crear un instalador para que el usuario pueda instalar el programa en su computadora. Con la ayuda de Visual Studio el proceso es muy sencillo y se hace directamente en el ambiente mediante un asistente. A la salida del proceso se obtiene un archivo con extensión .msi (instalador de Windows) que el usuario podrá usarlo para instalar el programa en su computadora. Una vez instalado el programa, éste se encuentra listo para ser utilizado.

Capítulo 6

Pruebas

En este capítulo se presentarán las pruebas que se le realizaron al *datalogger* con el fin de verificar su funcionamiento. Dado que se considera que el *datalogger* es un sistema de instrumentación, las pruebas realizadas se enfocaron en determinar en qué grado las mediciones realizadas por éste representan a las variables meteorológicas medidas. Adicionalmente, se realizó una prueba para determinar el consumo que el *datalogger* tiene para estimar cuanto será su tiempo de autonomía una vez operando en sitio.

6.1. Precipitación pluvial

En el caso de las pruebas de precipitación pluvial, se partió del hecho de que el pluviómetro de balancín instalado en el *datalogger* se encontraba debidamente calibrado de fábrica. Considerando lo anterior, se determinó que las pruebas sólo debían corroborar que el *datalogger* registrara los eventos de volcado del balancín de manera correcta. Así, el accionar del balancín se efectuó de manera manual para simular un evento de volcado. Con lo anterior, se evitó el uso de instrumentos adicionales para realizar las pruebas del *datalogger* relacionadas con la medición de la precipitación pluvial.

6.1.1. Prueba de eventos

Objetivo

Determinar que los eventos generados por el pluviómetro de balancín sean debidamente registrados por el *datalogger* y no existan registros en falso.

Desarrollo

Primeramente, para realizar esta prueba, se estableció, por medio de la opción de calibración del programa, el valor de volcado del pluviómetro especificado por el fabricante, el cual es de 0.5 mm. Posteriormente, en un intervalo de quince minutos se volcó el balancín de acuerdo a los momentos marcados en la tabla 6.1. Finalmente, una vez terminado el último evento, se extrajeron las mediciones registradas por el *datalogger* mediante el programa desarrollado.

Hora	**Número de eventos**
13:00	5
13:04	1
13:08	15
13:11	1
13:15	2

Tabla 6.1. Eventos de volcado.

Resultados

En la figura 6.1 se muestra una captura de pantalla de las mediciones obtenidas por el programa. Se puede verificar que el número y el momento de los eventos registrados por el *datalogger* coinciden plenamente con el mostrado en latabla 6.1.

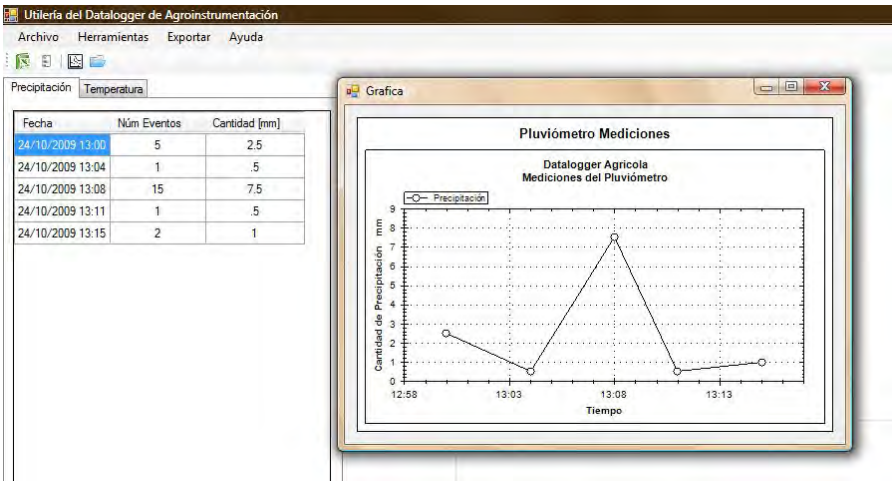


Fig. 6.1. Resultados de la prueba de eventos.

Conclusiones

Dado que los valores de las tablas coinciden, se determinó que el *datalogger* no presenta registros repetidos o mediciones incorrectas. Se concluye que, siempre que el *datalogger* haya sido calibrado con el valor de volcado del balancín, no existirán problemas en la medición de esta variable.

6.2. Pruebas de temperatura ambiente

El procedimiento general para efectuar las pruebas de la temperatura ambiente consistió en comparar las mediciones realizadas por el *datalogger* con mediciones realizadas por otro equipo de la variable a medir. El objetivo principal de todas las pruebas fue que, al efectuar una comparación entre las mediciones del *datalogger* y las del otro equipo, se encontraran discrepancias entre ambas mediciones y posteriormente intentar corregir dichas discrepancias con el propósito de tener mediciones más próximas entre las variables adquiridas.

Dado que no se contaba con un sistema que asemejara el modo de adquisición del *datalogger*, se decidió construir un equipo propio para la realización de las pruebas efectuadas sobre esta variable.

El modo en que este equipo se diseñó para que realizara las mediciones de temperatura ambiente es similar al que fue utilizado en el diseño del *datalogger*. Durante su operación, el equipo para realizar las pruebas, toma una medición de la temperatura ambiente cada minuto y las reserva. En el momento de que una tasa de registro es alcanzado, el sistema promedia las mediciones realizadas hasta ese momento y las registra en memoria no volátil.

El equipo de pruebas fue desarrollado con los siguientes elementos de hardware:

- Un microcontrolador ATMega16
- Un LM35DZ como sensor de temperatura
- Un convertidor A/D de 12 bits MCP3304 y
- Un circuito integrado DS4305 para generar la tensión de referencia

El diagrama de dicho circuito se muestra en la figura 6.2. Como se puede aprecias el convertidor A/D se conecta a través de SPI al microcontrolador. Al primer canal del convertidor A/D (CH0), se conectó el sensor de temperatura LM35DZ y en la terminal para la tensión de referencia de éste se conectó el circuito DS4305. El microcontrolador fue conectado únicamente con los elementos básicos para su funcionamiento: una resistencia de *pull-up*, en su terminal de *reset*, y un cristal, en las terminales de su oscilador interno. Las terminales de su módulo USART están indicadas ya fue a través de éstas como se realizó el vaciado de los datos adquiridos por el microcontrolador.

La implementación de este circuito en una tarjeta prototipo se puede ver en la figura 6.3 y el sensor LM35DZ acondicionado para realizar las puebas se puede ver en la figura 6.4.

Los elementos de hardware fueron seleccionados para intentar eliminar las fuentes de error que se considera están presentes en el *datalogger* durante la adquisición de la temperatura ambiente y son:

- Errores debido a la baja resolución del convertidor A/D
- Errores debido a la no linealidad de la sonda de temperatura
- Errores debido al *offset* que la sonda de temperatura presenta

El sensor de temperatura LM35DZ permite mediciones en un intervalo de los 0 [°C] a los 100 [°C], se encuentra calibrado para entregar una salida lineal de 10.0 [mV/°C] y posee una precisión de 0.4 [°C] cuando éste indica 25 [°C][1]. Características adicionales sobre la precisión del LM35DZ se muestran en la tabla 6.2.

Si bien ni el sensor LM35DZ ni el equipo realizado para las pruebas se pueden considerar un instrumentos patrones para la medición de la temperatura ambiente, dado que no han seguido una calibración que asegure el error presente en la medición de éstos. Se pretendió mediante las pruebas realizadas encontrar que la tendencia de las mediciones del *datalogger* concordaran, en cierta medida, con el del equipo de pruebas.

El equipo de pruebas diseñado presenta ciertos inconvenientes en su diseño, por ejemplo, para mantener su diseño simple, en lugar de utilizar un reloj en tiempo real (RTR) para determinar los períodos de muestreo y registro, estos se obtuvieron por medio de los temporizadores del microcontrolador. Lo anterior se espera pueda derivar en posibles retrasos o adelantos de las mediciones realizadas por el equipo. Adicionalmente, por la misma falta de

[1] **National Semiconductor**. LM35 Precision Centigrate Temperature Sensor datasheet.

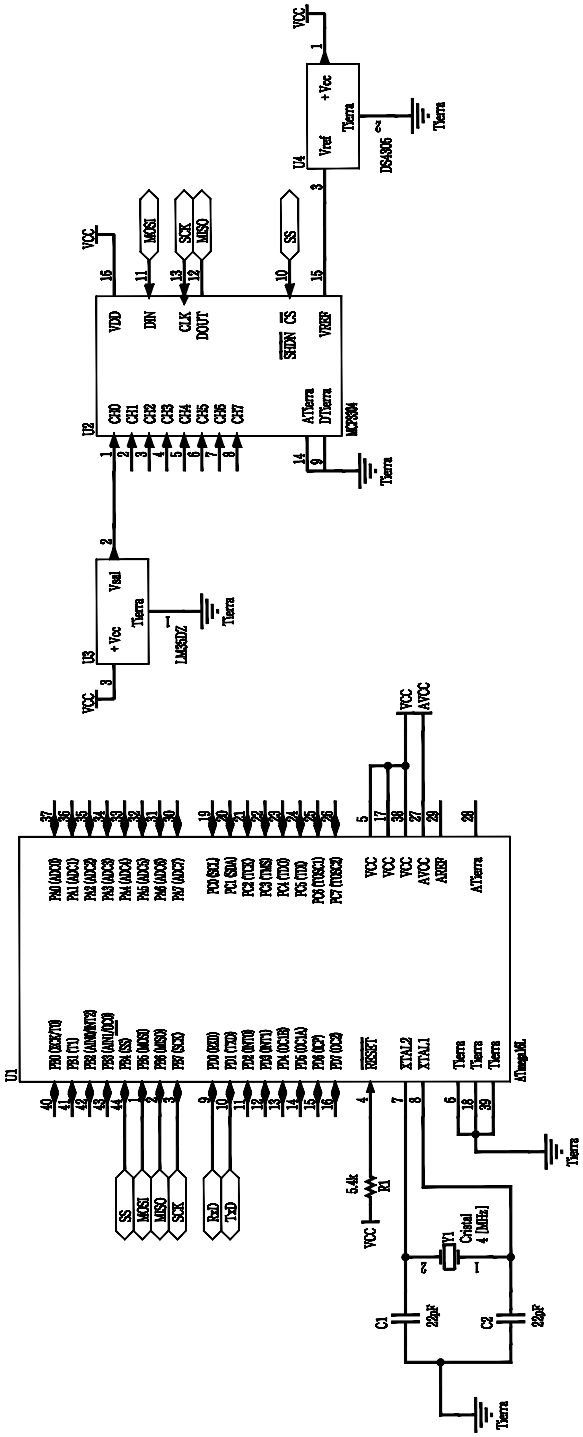


Fig. 6.2. Diagrama del equipo de pruebas.

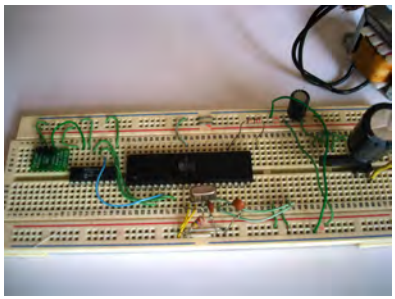


Fig. 6.3. Implementación del equipo de pruebas.

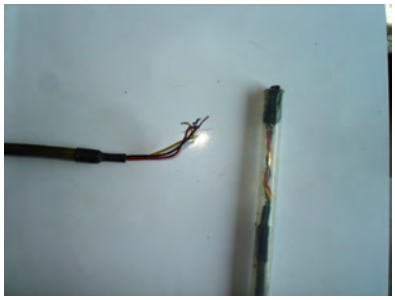


Fig. 6.4. Sensor LM35DZ.

un RTR, los datos adquiridos no se encontrarán estampados con la fecha y hora de su registro, debido a lo anterior, se tuvo que recurrir a una sincronización precisa entre el inicio de la adquisición de ambos equipos.

A continuación expondremos los procedimientos y resultados de las tres pruebas realizadas al *datalogger* ayudados por el equipo de pruebas descrito.

6.2.1. Primera prueba de 24 hrs

Objetivo

El objetivo de esta prueba es evaluar el desempeño del *datalogger* para adquirir la temperatura ambiente por un periodo de 24 horas.

Desarrollo

Tanto el *datalogger* como el equipo de pruebas se programaron para adquirir los valores de la temperatura ambiente a una tasa de registro de 30 minutos durante un periodo de 24 hrs. Previo al inicio de la prueba, las variables de calibración de la sonda de temperatura fueron establecidas, utilizando para

Temperatura[°C]	**Precisión**[°C]
25	0.4
0	0.8
100	0.8

Tabla 6.2. Características del LM35DZ.

ello el valor de temperatura mostrado por el sensor LM35DZ cuando éste indicaba una temperatura ambiente de aproximadamente 20 [°C]. Las variables de calibración quedaron de la siguiente manera:

- La amplificación obtenida fue de 293.15 y
- El *offset* ente el LM35DZ y la prueba de temperatura fue de -0.83 [°C]

Una vez ajustadas las variables de calibración, ambos sensores fueron colocados a la intemperie protegiéndolos con un cobertizo de cartón para evitar corrientes de aire e insolación.

La prueba comenzó a las 23 hrs. al iniciar la adquisición del *datalogger* por medio de su botón principal y energizar el equipo testigo para que éste, de igual forma, comenzara la adquisición.

La prueba finalizó a las 23:17 hrs. del día siguiente momento en cual, el equipo de pruebas, indicó que había realizado su último registro. El momento de término de la prueba arrojó, como primer resultado, que existía un desfase en el tiempo de la medición de los datos adquiridos, esto debido a que el equipo de pruebas no terminó exactamente a las 23 hrs.

Resultados

Durante el procedimiento de recepción de datos y de conexión del *datalogger* con la computadora no presentó ningún problema y los datos, una vez importados del *datalogger* fueron exportados a MS Excel. Este programa fue el que se utilizó para el análisis de los datos colectados tanto por el *datalogger* como por el equipo de pruebas.

En las tablas 6.3.a y 6.3.b se muestran las mediciones hechas por el *datalogger*, el equipo de pruebas y la diferencia entre estas mediciones de ambos. En la figura 6.5 se muestra la gráfica de los valores mostrados en la tabla.

Hora	**Temperatura del *datalogger*[°C]**	**Temperatura del equipo de pruebas [°C]**	**Diferencia [°C]**
23:30	18.18	20.45	–2.27
00:00	17.01	19.36	–2.35
00:30	16.94	19.05	–2.11
01:00	16.43	18.83	–2.40
01:30	16.08	18.37	–2.27
02:00	15.95	18.22	–2.27
02:30	15.42	17.77	–2.35
03:00	15.22	17.55	–2.33
03:30	14.89	17.33	–2.44
04:00	14.68	17.03	–2.35
04:30	14.38	16.89	–2.51
05:00	14.00	16.45	–2.45
05:30	13.75	16.02	–2.27
06:00	13.54	15.73	–2.18
06:30	13.72	15.73	–2.00
07:00	13.77	15.95	–2.18
07:30	13.72	15.80	–2.08
08:00	15.09	15.59	–0.50
08:30	21.06	17.92	3.14
09:00	20.38	22.30	–1.92
09:30	25.39	22.46	2.93
10:00	28.63	24.49	4.14
10:30	29.39	25.19	4.20
11:00	30.15	25.63	4.52
11:30	31.67	26.44	5.23
12:00	29.67	26.99	2.68
12:30	25.89	26.44	–0.54
13:00	25.39	25.45	–0.06

Tabla 6.3.a. Mediciones de la prueba 1.

Hora	**Temperatura del *datalogger*[°C]**	**Temperatura del equipo de pruebas [°C]**	**Diferencia [°C]**
13:30	25.31	25.01	0.31
14:00	25.14	25.01	0.13
14:30	25.34	25.10	0.24
15:00	25.36	25.19	0.18
15:30	25.09	25.45	–0.36
16:00	24.89	25.28	–0.39
16:30	23.82	24.92	–1.10
17:00	23.49	24.40	–0.91
17:30	23.67	24.40	–0.73
18:00	23.24	24.49	–1.25
18:30	24.91	24.14	0.77
19:00	23.01	24.06	–1.05
19:30	22.66	24.06	–1.40
20:00	21.98	23.63	–1.65
20:30	21.57	23.29	–1.72
21:00	21.37	22.79	–1.42
21:30	21.65	22.71	–1.06
22:00	18.00	22.54	–4.54
22:30	19.04	18.90	0.14
23:00	18.81	19.98	–1.17

Tabla 6.3.b. Mediciones de la prueba 1.

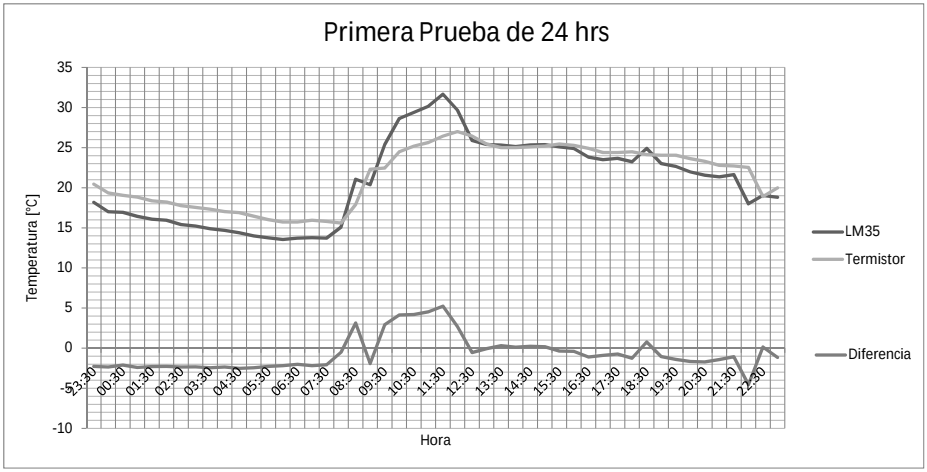


Fig. 6.5. Resultados de la primera prueba de 24hrs.

A primera vista las mediciones aparentan tener una tendencia similar. A las 23 hrs. las mediciones de temperatura tienen un comportamiento descendiente hasta las 7 de la mañana, momento en que presentan una ligera recuperación, pero vuelven a caer hasta llegar a las 8 de la mañana. En este momento la temperatura presenta un elevado ascenso que dura hasta alcanzar el medio día momento en que se alcanza su máximo valor del periodo. A partir de ese momento, la temperatura vuelve a caer de manera lenta hasta las 22:30 hrs. En la medición de este período la temperatura presenta una abrupta caída, la cual pudo ser debida a constantes ráfagas de viento observadas durante esas horas. Finalmente, la última medición arroja una temperatura muy próxima a la presentada en la primera medición del día anterior.

La diferencia máxima de temperatura entre ambos sensores se observa a las 11:30 hrs. con 5.23 [°C] y la mínima a las 14:00 hrs. con 0.13 [°C]. Estas diferencias representan un error del 19.8% y 0.51% respectivamente.

Conclusiones

Basados en la gráfica de la figura 6.5, podemos realizar las conclusiones concernientes a esta primera prueba. En primer lugar, se observa que al final de las mediciones existe un desfasamiento en las señales. Este desfasamiento fue originado por el retraso en la adquisición del equipo de pruebas.

Si bien tanto las mediciones del equipo de pruebas como las del *datalogger* siguen tendencias similares, entre ellas se presentan diferencias de varios grados centígrados. Cabe mencionar que, aunque esta diferencia no es constante, es similar entre determinados intervalos de temperatura.

Con lo anterior se concluye que aunque las variables de calibración de la sonda de temperatura fueron establecidas con respecto a las del LM35DZ antes de realizar las mediciones, aún existe un *offset* entre las mediciones adquiridas por éste y el equipo de pruebas. Aparentemente, y tomando en cuenta lo que especifica el fabricante, aunque la sonda de temperatura presenta un *offset* que puede ser eliminado con una calibración de un punto, el sistema en su conjunto presenta un *offset* que no sigue un comportamiento constante a través del rango de temperaturas a medir.

En la siguiente prueba realizada sobre el *datalogger* se intentará barrer todo el rango de medición de éste para detectar la tendencia que el *offset* presente sigue y una vez determinado intentar eliminarlo.

6.2.2. Prueba para determinar el *offset*

Objetivo

Determinar la diferencia entre las medidas tomadas por el *datalogger* y la salida del LM35DZ en el rango de temperaturas entre los +40 [°C] y los 0[°C].

Desarrollo

Ambos sensores de temperatura se sumergieron en agua a 40[°C] y se fue tomando el valor de temperatura de salida de ambos mientras el agua se enfriaba. Las mediciones del *datalogger* fueron obtenidas con un *offset* de calibración de 0.0 [°C] y mediante el modo prueba del programa de éste. Las mediciones de temperatura del LM35DZ se obtuvieron mediante la medición de la tensión a la salida de éste por medio de un multímetro. Cuando la temperatura del agua llegó a temperatura ambiente, ésta se continuó enfriando mediante la incorporación de hielo. La temperatura mínima alcanzada fue de 11.7 [°C].

Una vez obtenidas las mediciones de ambos sensores, se calculó la diferencia de temperatura entre éstos.

Resultados

En las tablas 6.4.a y 6.4.b se muestran los resultados de las mediciones de temperatura de ambos sensores y su diferencia. En la figura 6.6 se muestra la gráfica de la diferencia de temperatura entre ambos sensores con respecto a la temperatura de salida del *datalogger*. Adicionalmente, en esta gráfica, se muestra la línea de tendencia de la curva resultante y la ecuación de dicha línea.

Temperatura del LM35[°C]	**Temperatura del *datalogger* [°C]**	**Diferencia [°C]**
41.2	39.53	1.67
40.8	39.27	1.53
40.5	38.51	1.99
40.5	37.78	1.52
39.3	37.31	1.09
37.7	36.61	1.09
37.3	36.38	0.92
36.0	35.71	0.29
35.8	35.49	0.31
35.4	35.06	0.34
35.1	35.06	0.04
34.3	34.42	–0.12
33.6	33.80	–0.2
32.3	33.80	–0.5
31.8	32.41	–0.61
29.5	30.19	–0.69
28.5	29.67	–1.17
27.3	28.65	–1.35
28.0	22.33	–1.33
27.2	28.65	–1.45
26.5	28.16	–1.66
26.0	27.84	–1.84
25.5	27.05	–1.55

Tabla 6.4.a. Mediciones de calibración.

Temperatura del LM35[°C]	**Temperatura del *datalogger* [°C]**	**Diferencia [°C]**
25	26.74	–1.74
24.5	26.13	–1.63
23.8	25.53	–1.73
23	24.67	–1.67
22	23.83	–1.83
20.5	21.97	–1.47
20.3	21.85	–1.55
19.8	21.22	–1.42
18.6	20.12	–1.52
17.8	19.41	–1.61
17.6	19.30	–1.70
16.8	18.50	–1.70
15.8	17.4	–1.60
14.8	16.34	–1.54
14.2	15.93	–1.73
14.1	15.23	–1.12
13.5	15.13	–1.63
11.6	12.12	–0.53

Tabla 6.4.b. Mediciones de calibración.

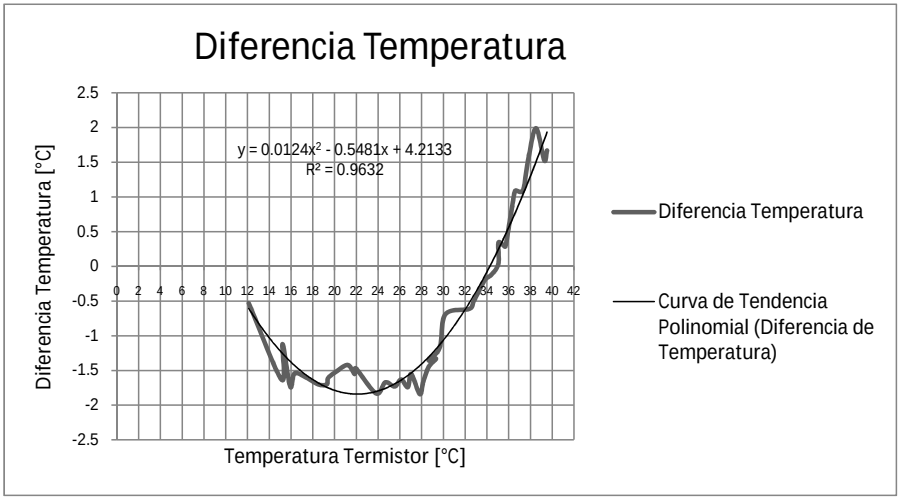


Fig. 6.6. Gráfica resultante de la prueba de *offset*.

Conclusiones

Como se puede observar en la gráfica, se determinó que el *offset* de la sonda no es constante con respecto a las mediciones hechas con el LM35DZ. Adicionalmente, se observa que las mediciones salen con muchos saltos. Durante la realización de la prueba se observó que estos saltos se deben a variaciones de la salida del *datalogger*. Dado que la resolución del convertidor A/D nos da para un bit de cambio una gran variación en términos de grados centígrados, estas variaciones del convertidor afectaron mucho las mediciones realizadas.

Se consideró que para poder realizar mediciones más próximas entre el *datalogger* y el equipo de pruebas, el *offset* existente entre ambos sensores debía ser corregido. Lo anterior se realizó incorporando la ecuación de la curva de tendencia del *offset* mostrada en la gráfica a los datos obtenidos por el *datalogger*. En la siguiente prueba de 24 horas se buscará haber eliminado el *offset* que se presentó en la primera prueba de 24 hrs.

6.2.3. Segunda prueba de 24 horas

Objetivo

Determinar si la corrección del *offset* de las señales fue suficiente para lograr una mejor aproximación entre las mediciones hechas por el *datalogger* y el equipo de pruebas.

Desarrollo

Al igual que la primera prueba de 24 horas, se dejaron ambos equipos adquiriendo la temperatura ambiente en un intervalo de 24 horas. Se protegieron los sensores en un cobertizo de cartón. Antes de realizar la prueba, se calculó de manera más precisa los tiempos del equipo de pruebas, por lo que el desfasamiento de las señales de ambos equipos debe de ser minimizado.

Resultados

En las tablas 6.5.a, 6.5.b y 6.5.c se muestran las mediciones obtenidas tanto del *datalogger* como del equipo de pruebas. Los resultados de la tabla se graficaron y se muestran en la figura 6.7. En esta última figura se puede apreciar que la tendencia de ambas gráficas es igual, sin embargo, los valores siguen presentando una diferencia de temperatura entre ambos sensores. Al igual que en la prueba número uno, se aprecia una gran discrepancia a altas temperaturas siendo registrado por el equipo testigo hasta 37.8 [°C].

La diferencia máxima entre los sensores para esta prueba fue de 9.93 [°C] a las 11:40 hrs. y la mínima fue de 0.00 [°C] a las 05:40 hrs. Estas diferencias representan un error de 35.55% y de 0% respectivamente.

Hora	**Temperatura del *datalogger*[°C]**	**Temperatura del equipo de pruebas [°C]**	**Diferencia [°C]**
20:10	16.48	16.32	–0.16
20:40	15.55	18.33	–2.78
21:10	15.87	18.67	–2.79
21:40	15.55	17.77	–2.22
22:10	15.10	17.34	–2.24
22:40	15.50	18.21	–2.71
23:10	15.55	18.21	–2.66
23:40	15.32	17.66	–2.34
00:10	12.95	15.09	–2.14
00:40	12.47	14.22	–1.75
01:10	12.47	13.89	–1.42
01:40	12.11	12.86	–0.74

Tabla 6.5.a. Mediciones de la prueba 2.

Hora	**Temperatura del *datalogger*[°C]**	**Temperatura del equipo de pruebas [°C]**	**Diferencia [°C]**
02:10	12.17	12.86	–0.69
02:40	11.97	12.56	–0.59
03:10	11.56	12.06	–0.50
03:40	11.41	11.71	–0.30
04:10	11.03	11.24	–0.21
04:40	11.28	11.31	–0.03
05:10	11.74	11.92	–0.18
05:40	11.71	11.71	0.00
06:10	12.07	12.56	–0.49
06:40	11.69	12.49	–0.80
07:10	12.29	13.64	–1.35
07:40	12.27	14.22	–1.95
08:10	17.77	14.47	3.30
08:40	16.25	15.64	0.62
09:10	21.43	19.14	2.29
09:40	19.26	19.38	–0.12
10:10	23.86	20.39	3.47
10:40	32.26	24.00	8.26
11:10	36.53	27.93	8.60
11:40	37.87	27.93	9.93
12:10	31.25	26.77	4.48
12:40	24.16	22.61	1.55
13:10	23.43	20.91	2.52
13:40	23.68	21.05	2.63
14:10	23.33	21.32	2.01
14:40	22.37	21.05	1.32
15:10	22.54	21.05	1.49
15:40	22.80	21.18	1.56

Tabla 6.5.b. Mediciones de la prueba 2.

Hora	**Temperatura del *datalogger*[°C]**	**Temperatura del equipo de pruebas [°C]**	**Diferencia [°C]**
16:10	22.80	21.60	1.20
16:40	22.21	21.46	0.75
17:10	21.68	21.32	0.36
17:40	21.00	21.18	–0.18
18:10	20.35	20.91	–0.56
18:40	19.54	20.26	–0.72
19:10	18.78	19.88	–1.10
19:40	18.28	19.63	–1.35

Tabla 6.5.c. Mediciones de la prueba 2.

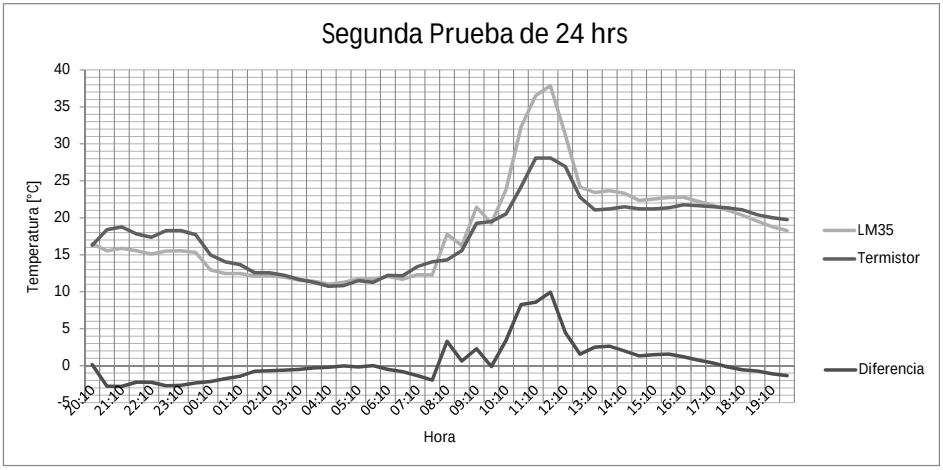


Fig. 6.7. Resultados de la segunda prueba de 24hrs.

Conclusiones

Dado que la diferencia entre las mediciones se mantuvo, aunque se incorporó la ecuación del *offset* que existe entre el LM35DZ y la sonda de temperatura, se puede concluir que no se puede lograr que las mediciones de ambos sensores tengan diferencias pequeñas en el rango de medición simplemente mediante el ajuste del *offset*.

Como conclusión general de la adquisición de temperatura se determina que, en general la adquisición de la sonda de temperatura funciona de manera

adecuada dado que las mediciones mantienen una tendencia correcta, pero no se puede determinar que éstas tengan valores correctos hasta que no se realice una adecuada calibración del *datalogger*.

6.3. Prueba del consumo energético del *datalogger*

Esta prueba tiene como objetivo estimar el tiempo de autonomía energética del *datalogger*. Para ello primeramente, se realizó una prueba para determinar el consumo de energía que presenta el *datalogger*. Si bien el *datalogger* presenta diversos estados de operación y por lo tanto diversos valores de consumo de energía, para realizar esta prueba sólo se contempló el modo de adquisición ya que, es en este estado, en donde el *datalogger* se encontrará operando la mayor parte del tiempo.

En el modo de adquisición el *datalogger* se encuentra en espera de alguna interrupción externa que le indique la adquisición de alguna de las variables meteorológicas. En este estado el microcontrolador se encuentra en su estado de bajo consumo y el consumo del *datalogger* es básicamente debido al consumo del circuito de acondicionamiento del pluviómetro y de las memorias EEPROM.

Objetivo

Determinar el consumo de energía del *datalogger* cuando éste está operando en su modo de adquisición.

Desarrollo

Una vez que el *datalogger* se encuentraba en modo de adquisición, mediante el uso de un multímetro se midió la corriente a la salida de la batería tipo C con la cual se alimentaba al *datalogger*.

Resultados

Los resultados arrojan que la corriente de salida de la batería, cuando el *datalogger* se encuentra en modo de adquisición, es de 2.21 [mA].

Análisis

Considerando la capacidad energética del tipo de batería con el que se alimentará al *datalogger* se puede estimar el tiempo de servicio que éste tendrá. Asi, considerando una batería alcalina de tamaño C con una capacidad de 8300 [mAh][2] y un consumo de 2.21 [mA] por parte del *datalogger*, se estima que éste podrá operar hasta 3755.66 horas continuas, valor aproximadamente igual a 156 días.

Conclusiones

Como conclusión se determina que si el *datalogger* operará todo el tiempo en modo de adquisición, éste tendría una autonomía en su operación de hasta 156 días. Dado que la operación del *datalogger* no es tal y que distintos modos de operación están presentes durante el uso normal del *datalogger*, se considera que éste no permanecerá en operación tanto tiempo. Si bien la mejor manera de saber cuál es la autonomía del *datalogger* es a través de mantenerlo en operación constante hasta que éste agote su batería, los cálculos aquí presentes nos permiten estimar que al menos una operación de tres meses con una batería tamaño C está garantizada.

Una vez descritas las pruebas realizadas al *datalogger* en el siguiente capítulo se expondrán las conclusiones que se obtuvieron de este proyecto.

[2] **Energizer**, *Product Datasheet Energizer E93.*

Capítulo 7

Resultados y Conclusiones

En este capítulo se darán a conocer los resultados obtenidos y las conclusiones generales del proyecto desarrollado. Al final de este capítulo también se darán recomendaciones para continuar con el desarrollo del mismo.

7.1. Resultados

Los resultados de este proyecto se pueden dividir, de acuerdo a las partes de las que consta el desarrollo del *datalogger*, en tres: **hardware**, ***firmware*** y **software**.

En primer lugar, en la parte de **hardware** se consiguió realizar el diseño de una tarjeta para la adquisición de datos de dos variables meteorológicas: temperatura ambiente y precipitación pluvial. Para lograr lo anterior se diseñaron los distintos circuitos de acondicionamiento para los sensores y el circuito digital para el registro de estas variables. Adicionalmente, dados los requerimientos de la aplicación, se puso especial énfasis en la implementación de técnicas para la disminución del consumo de los componentes usados. En cuanto a los medios de interacción entre el usuario y el *datalogger*, en la tarjeta se implementaron controles simples para su uso en sitio y, en la comunicación entre éste y la PC, se implementaron diferentes protocolos de comunicación, dando con ello una mayor versatilidad de uso al usuario. En cuanto al consumo de energía, aunque no se pudieron realizar pruebas que confirmen cuál será la duración de la batería, se estimó que ésta fácilmente podrá alcanzar para que el *datalogger* opere mínimamente tres meses.

En la parte del ***firmware*** se logró crear un programa en el microcontrolador que manejara tanto los módulos de éste como los distintos dispositivos externos que componen el *datalogger*. Mediante este manejo se logró adquirir, guardar y trasmitir a una PC las mediciones de las dos variables meteorológicas que se plantearon adquirir. Adicionalmente a la adquisición de las

variables se logró que el programa desarrollado contribuyera a disminuir el consumo de energía del *datalogger*, esto también en gran parte ayudado por los componentes de hardware seleccionados.

El **programa** desarrollado en la PC se convirtió en una herramienta muy importante dado que éste permitió una distribución de tareas entre el *datalogger* y la computadora, con lo cual se logró superar las deficiencias en el procesamiento de la información del microcontrolador. El programa se desarrolló para que fungiera como un ambiente para el uso de distintos *dataloggers*, lo cual permite mantener la información colectada por cada uno de ellos de manera separada. Adicionalmente, a través de este mismo programa, se lograron implementar diversas herramientas para darle al usuario un mejor manejo de los datos colectados como el graficado y la exportación de los mismos. En cuanto a la comunicación entre el *datalogger* y la computadora se logró, a través de la implementación de una serie de comandos, establecer funcionales adicionales en el *datalogger* que no hubiera sido posible implementar con los pocos controles en sitio que cuenta éste.

Finalmente, se espera que una vez que un *datalogger* sea instalado en una región agrícola, los datos adquiridos por éste puedan servir tanto para tomar decisiones en sitio, como para realizar estudios sobre el área de cultivo. El *datalogger*, además de que contribuirán al mejoramiento de la calidad y la cantidad de las cosechas, poseerá ventajas sobre las estaciones agrometeorológicas no automatizadas, tanto en su facilidad de uso como en la utilidad de la información adquirida. Por citar un ejemplo, los datos de precipitación pluvial pueden ser considerados para realizar estudios sobre la erosividad de una región agrícola, estudio que no se podría realizar con datos obtenidos de un simple pluviómetro.

7.2. Conclusiones

Basándonos en el objetivo de este proyecto, como conclusión general se puede decir que dicho objetivo fue cumplido. Se logró obtener un sistema capaz de realizar las mediciones deseadas y que dichas mediciones fueran presentadas al usuario de manera adecuada.

De las cosas que me gustaría destacar en cuanto el desarrollo del proyectos es que si bien había encontrado equipos desarrollados que hicieran uso de

memorias EEPROM para el guardado de los datos adquiridos, éstos no contemplan, en primer lugar, lo que sucede con la información almacenada en éstas cuando el equipo era completamente desenergizado y en segundo lugar, la degradación que sufren las celdas de memoria después de múltiples ciclos de escritura. El algoritmo desarrollado para el grabado de datos contempla ambos aspectos, además de organizar de un mejor modo la memoria y por ende hacer más eficiente el uso de la cantidad de almacenamiento disponible.

El proyecto desarrollado deja una base de conocimiento que puede servir para la realización de proyectos futuros y para la incorporación de nuevas características a éste.

Este proyecto sin duda está sujeto a mejoras entre las cuales destaco:

La parte de hardware puede en cierta medida ser integrada de mejor manera, utilizar, por citar un ejemplo, un único amplificador operacional, lo cual ahorraría espacio y disminuiría el consumo energético.

En caso de querer mejorar la resolución del *datalogger*, en cuanto a la adquisición de los valores de temperatura ambiente se requeriría un convertidor de mayor resolución. Lo anterior podría ser logrado mediante el uso de un convertidor externo o incluso mediante el remplazo del microcontrolador por uno con mejores características, entre ellas un convertidor A/D de mayor resolución.

En cuanto al *firmware* es de todas las partes del proyecto la que más satisfacción me dio tanto en su desarrollo como en el resultado final, aun así todo código está sujeto a ser optimizado y nuevas características podrían serle dadas al *datalogger* a través de la modificación del *firmware*. Dentro de estas nuevas características, la que yo considero la más importante, es la implementar junto con el programa de un método para la detección de errores en la transmisión de los datos.

Las recomendaciones del software se enfocan en mejorar el código del programa, dado que es mi primera experiencia desarrollando un programa como el aquí mostrado, sin duda deben de existir formas de hacer el desarrollo del mismo más compacto y eficiente.

Por último, en cuanto a una recomendación general para el proyecto me gustaría mencionar que al *datalogger* se le es posible incrementar el número de sensores. Este aspecto requeriría de la modificación de todos los elementos del mismo, pero dada la experiencia que se logró, no existiría mayor problema en realizarlo. De igual forma para mejorar su versatilidad, a éste se le podrían agregar medios de comunicación inalámbrica, ya sea entre el *datalogger* y la PC o incluso entre el *datalogger* y los sensores. El rumbo que tome este proyecto

sólo se encontrará limitado por las necesidades que se quiera satisfacer y la imaginación de los ingenieros que lo retomen.

Finalmente, quiero agregar que este proyecto significó una gran experiencia para mí. Contribuyó no sólo a que yo pusiera en práctica conocimientos de la carrera sino también a emplear las habilidades desarrolladas durante la misma.

Bibliografía

Libros

1. Couglin, R. (1999). *Amplificadores Operacionales y Circuitos Integrados Lineales* (5ª ed.). México: Pearson Eductation.

2. Elias Castillo, F. (1996). *Agrometeorología*. Madrid: Mundi-Prensa.

3. Kester W. (2004). *Analog–Digital Conversion*. E.U.A.: Analog Devices.

4. Ledesma, J.M. (2000). *Climatología y Meteorología Agrícola*. España: Paraninfo.

5. Predko, M. (1997). *Handbook of Microcontrollers*. (2ª ed.). E.U.A.: Ed. New Jersey.

6. Savant, C.J. (2000). *Diseño Electrónico: Circuitos y Sistemas*. España: Prentice Hall.

7. Sedra A. y Smith K. (2006). *Circuitos Microelectrónicos*. (5ª ed.). México: Mc. Graw–Hill Interamericana.

8. Sharp J. (2008). *Microsoft Visual C# 2008 Step by Step*. E.U.A.: Microsoft Press.

9. Watson K. y Nagel C. (2008). *Beginning Microsoft Visual C# 2008*. E.U.A.: Wiley Publishing Inc.

Manuales

10. *24LC1025 1024 I^2C CMOS Serial EEPROM*. (2005). E.U.A.: Microchip Tecnology Inc.

11. *74LVC1G17 Single Schmitt trigger buffer*. (2007). Holanda: NXP semiconductors.

12. *ATMEL 8–bit AVR Microcontroller ATMega16*. (2007). E.U.A.: Atmel Corporation.

13. *DS1337 I^2C Serial Real-Time Clock*. (2009). E.U.A.: Maxim Dallas Semiconductor.

14. *Energizer E93 Product Datasheet*. (2008) E.U.A.: Energizer.

15. *FT232R USB UART IC*. (2009) E.U.A.: FTDI Chip.

16. *LM35 Precision Centigrade Temperature Sensors*. (2000) E.U.A.: National Semiconductors.

17. *MAX1675 Step-Up DC-DC conevreter*. (2000) E.U.A.: Maxim Integrated Products.

18. *MAX3233 Dual RS-232 Transceiver with Internal Capacitors*. (2004) E.U.A.: Maxim Integrated Products.

19. *MAX4165 Single Supply, Rail-to-Rail I/O Amp* (2007) E.U.A.: Maxim Integrated Products.

20. *Microsoft Visual C# .NET Referencia del Lenguaje*. (2002) España: Mc. Graw Hill.

21. *Model 107 Temperature Probe*. (2008). E.U.A.: Campbell Scientific. Inc.

22. *NC7SZ14 TinyLogic UHS Inverter with Schmitt Trigger Input*. (2004) E.U.A.: Fairchild Semiconductor.

23. *WMO Guide to Meteorological Instruments and Methods of Observation* (2008) (7ª ed.). World Meteorological Organization.

Mesografía

24. Alcinova S. *Automatization of Meteorological Measurements in Republic of Macedonia for the Needs of Environmental Sustainability*. W.M.O.: Republica de Macedonia.
http://www.wmo.int/pages/prog/www/IMOP/publications/IOM-94-TECO2006/P5(4)_Monevska_Macedonia.pdf

25. Baker B. (2003). *Precision Temperature Sensing with RTD Circuits*. E.U.A.: Microchip Technology Inc.
http://ww1.microchip.com/downloads/en/AppNotes/00687b.pdf

26. Baker B. (2004). *Analog Design Note ADN010: Predict the Repeatability of your ADC to the BIT*. E.U.A.: Microchip Technology Inc
http://ww1.microchip.com/downloads/en/devicedoc/ADN010.pdf

27. Baker B. (2000). *AN682: Using Single Supply Operational Amplifiers in Embedded Systems.*. E.U.A.: Microchip Technology Inc.
http://ww1.microchip.com/downloads/en/AppNotes/00682c.pdf

28. Buitenkant P. (2000). *Overcoming Erase/Write Endurance Limitation in EEPROM's*. E.U.A.: Electronic Design, Strategy News.
http://www.edn.com/article/CA47235.html

29. JChampion (2007). *A flexible charting library for .NET*. E.U.A.: Code Project.
http://www.codeproject.com/KB/graphics/zedgraph.aspx

30. Maxim (2000). *AN 287: Switch Bounce and other Dirty Little Secrets*. E.U.A.: Maxim Integrated Products.
http://pdfserv.maxim-ic.com/en/an/AN287.pdf

31. Maxim (2001). *AN 1080: Understanding SAR ADCs*. E.U.A.: Maxim Integrated Products.
http://pdfserv.maxim-ic.com/en/an/AN1080.pdf

32. Peacock C. (2002). *USB in a Nutshell*. E.U.A.: Beyong Logic.
http://www.beyondlogic.org/usbnutshell/usb-in-a-nutshell.pdf

33. Servicio Meteorológico Nacional.
http://smn.cna.gob.mx/

34. Wilkie D. (2005). *EEPROM Endurance Tutorial*. E.U.A.: Microsoft Technology Inc.
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en025550

Glosario

A

ACK *Acknowledge.* En una comunicación I²C es usado por el receptor para indicarle al transmisor que éste puede continuar con la comunicación.

B

Bus Canal eléctrico para transferir datos entre diferentes dispositivos.

C

Clase En una programación orientada a objetos, es un tipo de dato que define la implementación de un objeto en particular.

Compilador Programa informático que traduce un programa escrito en un lenguaje de programación a otro ejecutable por un procesador.

E

EEPROM Tipo de memoria no volátil que puede ser borrada usando medios electrónicos.

Erosividad Erosión causada por la lluvia.

Estampa de tiempo Tiempo en el que un evento es registrado.

F

FGMOSFET *Floating Gate* MOSFET, MOSFET's que poseen una compuerta aislada y es usado para almacenar un bit de información en las memorias EEPROM y flash.

Firmware Programa que se encuentra embebido en el hardware de un dispositivo, por ejemplo un microcontrolador.

H

Hardware Componentes físicos de un sistema.

Host Dispositivo anfitrión que provee servicios otros dispositivos.

Hub Punto de conexión común entre dispostivos de una red.

I

I²C *Inter-Integrated Circuit* tambien llamado TWI. es un tipo de comunicación serie utilizado para conectar dispostivos con sólo dos lineas de comunicación.

Intercambiabilidad Error de. Diferencia entre el valor nominal de determinado termistor y la curva de Steinhart-Hart que caracteriza el modelo al que pertenece dicho termistor.

N

NACK *Negative Acknowledge.* En una comunicación I²C es usada por el receptor para indicarle al transmisor que no puede continuar con la comunicación.

NRZI *Non-Return-to-Zero Inverted.* Tipo de codificación que representa un 0 con un cambio de nivel y un 1 sin cambio.

NTC *Negative Temperature Coefficient.* Tipo de termistor que decrementa su resistencia conforme su temperatura se incrementa.

O

Objeto En una programación orientada a objetos, es una realización concreta de una clase que consiste en información y las operaciones asociadas con dicha información.

P

Pluviómetro Dispositivo usado para medir la cantidad de precipitación pluvial.

Pull-up Resistencia. Resistencia que se conecta entre Vcc y una entrada para mantener la entrada en un valor alto.

R

RS-232 Norma que designa una interfaz de intercambio de datos binarios serie.

S

SCL En una comunicación I^2C es la línea que lleva la señal de reloj generada por el dispositivo maestro.

SDA En una comunicación I^2C es la línea donde se trasmiten datos entre el transmisor y el receptor.

Software Colección de programas de computadora que realizan una tarea sobre un sistema computacional.

Steinhart-Hart Ecuación. Ecuación que relaciona la resistencia de un termistor con su temperatura.

T

Termistor Semiconductor que varía el valor de su resistencia eléctrica en función de su temperatura.

TWI *Two-wire interface,* ver I^2C.

U

USART *Universal Synchronous/Asynchronous Receiver/Trasmitter.* Dispositivo que permite establecer una comunicación síncrona o asíncrona serie.

USB *Universal Serial Bus.* Estándar de comunicación serie que es usado para conectar dispositivos incluyendo teléfonos, computadoras, etc.

Apéndice 1

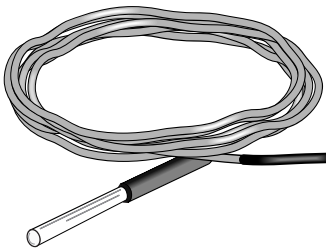
Portada de las hojas de especificaciones de los componentes usados

- Sonda de temperatura 107
- Micocontrolador ATMega16
- Memoria EEPROM 24LC1025
- Reloj en tiempo real DS1337
- Transceptor RS-232 MAX3233
- Transceptor RS-232 a USB FT232
- Compuerta inversora con Schmit Trigger NC7SZ14
- Amplificadores operacionales MAX4166 y MAX4168
- Convertidor de DC-DC de subida MAX1676

INSTRUCTION MANUAL

Model 107 Temperature Probe

Revision: 10/08



Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 16K Bytes of In-System Self-programmable Flash program memory
  - 512 Bytes EEPROM
  - 1K Byte Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels in TQFP Package Only
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7 - 5.5V for ATmega16L
  - 4.5 - 5.5V for ATmega16
- Speed Grades
  - 0 - 8 MHz for ATmega16L
  - 0 - 16 MHz for ATmega16
- Power Consumption @ 1 MHz, 3V, and 25·C for ATmega16L
  - Active: 1.1 mA
  - Idle Mode: 0.35 mA
  - Power-down Mode: < 1 µA

8-bit Microcontroller with 16K Bytes In-System Programmable Flash

ATmega16
ATmega16L

Rev. 2466S–AVR–05/09

24AA1025/24LC1025/24FC1025

1024K I²C™ CMOS Serial EEPROM

Device Selection Table:

Part Number	Vcc Range	Max Clock Frequency	Temp Ranges
24AA1025	1.8-5.5V	400 kHz†	I
24LC1025	2.5-5.5V	400 kHz	I
24FC1025	2.5-5.5V	1 MHz	I

†100 kHz for VCC < 2.5V.

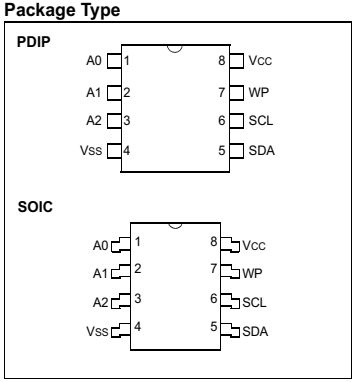
Features:

- Low-power CMOS technology:
  - Maximum write current 5 mA at 5.5V
  - Maximum read current 500 µA at 5.5V
  - Standby current 100 nA, typical at 5.5V
- 2-wire serial interface bus, I²C™ compatible
- Cascadable for up to four devices
- Self-timed erase/write cycle
- 128-byte Page Write mode available
- 5 ms max. write cycle time
- Hardware write-protect for entire array
- Output slope control to eliminate ground bounce
- Schmitt Trigger inputs for noise suppression
- 1,000,000 erase/write cycles
- Electrostatic discharge protection > 4000V
- Data retention > 200 years
- 8-pin PDIP, SOIC packages
- Temperature ranges:
  - Industrial (I): -40°C to +85°C

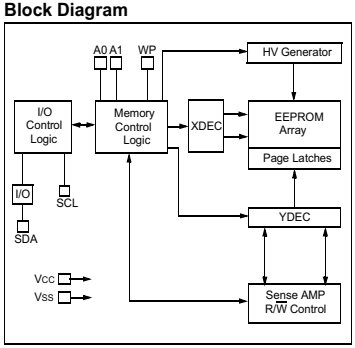
Description:

The Microchip Technology Inc. 24AA1025/24LC1025/24FC1025 (24XX1025*) is a 128K x 8 (1024K bit) Serial Electrically Erasable PROM, capable of operation across a broad voltage range (1.8V to 5.5V). It has been developed for advanced, low power applications such as personal communications or data acquisition. This device has both byte write and page write capability of up to 128 bytes of data. This device is capable of both random and sequential reads. Reads may be sequential within address boundaries 0000h to FFFFh and 10000h to 1FFFFh. Functional address lines allow up to four devices on the same data bus. This allows for up to 4 Mbits total system EEPROM memory. This device is available in the standard 8-pin plastic DIP and SOIC packages.

Package Type



Block Diagram



*24XX1025 is used in this document as a generic part number for the 24AA1025/24LC1025/24FC1025 devices.

19-4652; 7/09

DS1337
I²C Serial Real-Time Clock

www.maxim-ic.com

GENERAL DESCRIPTION

The DS1337 serial real-time clock is a low-power clock/calendar with two programmable time-of-day alarms and a programmable square-wave output. Address and data are transferred serially through an I^2C bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator.

The device is fully accessible through the serial interface while V_{CC} is between 1.8V and 5.5V. I^2C operation is not guaranteed below 1.8V. Timekeeping operation is maintained with V_{CC} as low as 1.3V.

APPLICATIONS

Handhelds (GPS, POS Terminal, MP3 Player)

Consumer Electronics (Set-Top Box, VCR/Digital Recording)

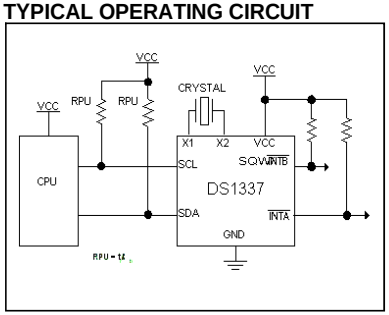
Office Equipment (Fax/Printer, Copier)

Medical (Glucometer, Medicine Dispenser)

Telecommunications (Router, Switch, Server)

Other (Utility Meter, Vending Machine, Thermostat, Modem)

TYPICAL OPERATING CIRCUIT



FEATURES

- **Real-Time Clock (RTC) Counts Seconds, Minutes, Hours, Day, Date, Month, and Year with Leap-Year Compensation Valid Up to 2100**
- **Available in a Surface-Mount Package with an Integrated Crystal (DS1337C)**
- **I^2C Serial Interface**
- **Two Time-of-Day Alarms**
- **Oscillator Stop Flag**
- **Programmable Square-Wave Output Defaults to 32kHz on Power-Up**
- **Available in 8-Pin DIP, SO, or µSOP**
- **-40°C to +85°C Operating Temperature Range**

ORDERING INFORMATION

PART	TEMP RANGE	PIN-PACKAGE	TOP MARK†
DS1337+	-40°C to +85°C	8 DIP (300 mils)	DS1337
DS1337S+	-40°C to +85°C	8 SO (150 mils)	DS1337
DS1337U+	-40°C to +85°C	8 µSOP	1337
DS1337C#	-40°C to +85°C	16 SO (300 mils)	DS1337C

+ Denotes a lead(Pb)-free/RoHS-compliant device.
 # Denotes a RoHS-compliant device that may include lead that is exempt under the RoHS requirements. The lead finish is JESD97 category e3, and is compatible with both lead-based and lead-free soldering processes.
† A "+" anywhere on the top mark denotes a lead-free device. A "#" denotes a RoHS-compliant device.

Pin Configurations appear at end of data sheet.

***Note:** Some revisions of this device may incorporate deviations from published specifications known as errata. Multiple revisions of any device may be simultaneously available through various sales channels. For information about device errata, go to: www.maxim-ic.com/errata.*

19-1473; Rev 2; 8/04

MAXIM

±15kV ESD-Protected, 1µA, 250kbps, 3.3V/5V, Dual RS-232 Transceivers with Internal Capacitors

General Description

The MAX3233E/MAX3235E are EIA/TIA-232 and V.28/V.24 communications interfaces with automatic shutdown/wake-up features, high data-rate capabilities, and enhanced electrostatic discharge (ESD) protection. All transmitter outputs and receiver inputs are protected to ±15kV using IEC 1000-4-2 Air-Gap Discharge, to ±8kV using IEC 1000-4-2 Contact Discharge, and to ±15kV using the Human Body Model. The MAX3233E operates from a +3.3V supply; the MAX3235E operates from +5.0V.

All devices achieve a 1µA supply current using Maxim's revolutionary AutoShutdown Plus™ feature. These devices automatically enter a low-power shutdown mode when the following two conditions occur: either the RS-232 cable is disconnected or the transmitters of the connected peripherals are inactive, and the UART driving the transmitter inputs is inactive for more than 30 seconds. They turn on again when they sense a valid transition at any transmitter or receiver input. AutoShutdown Plus saves power without changes to the existing BIOS or operating system.

The MAX3233E/MAX3235E have internal dual charge pumps requiring no external capacitors. Both transceivers have a proprietary low-dropout transmitter output stage that enables true RS-232 performance from a +3.0V to +3.6V supply for the MAX3233E or a +4.5V to +5.5V supply for the MAX3235E. These devices are guaranteed to operate up to 250kbps. Both are available in space-saving 20-pin wide SO or plastic DIP packages.

Applications

Subnotebook and Palmtop Computers
Cellular Phones
Battery-Powered Equipment
Handheld Equipment
Peripherals
Embedded Systems

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX3233ECWP	0°C to +70°C	20 SO
MAX3233ECPP	0°C to +70°C	20 Plastic DIP
MAX3233EEWP	-40°C to +85°C	20 SO
MAX3233EEPP	-40°C to +85°C	20 Plastic DIP

Ordering Information continued at end of data sheet.

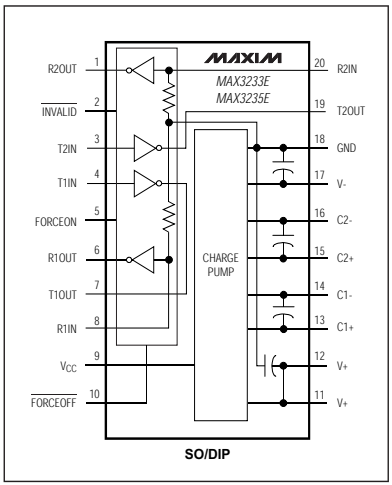
AutoShutdown Plus is a trademark of Maxim Integrated Products, Inc.

†Covered by U.S. Patent numbers 4,636,930; 4,679,134; 4,777,577; 4,797,899; 4,809,152; 4,897,774; 4,999,761; 5,649,210; and other patents pending.

Features

- **ESD Protection for RS-232 I/O Pins**
  ±15kV—Human Body Model
  ±8kV—IEC 1000-4-2, Contact Discharge
  ±15kV—IEC 1000-4-2, Air-Gap Discharge
- **Latchup Free**
- **1µA Supply Current**
- **AutoShutdown Plus—1997 EDN Magazine Innovation of the Year**
- **Single-Supply Operation**
  +3.0V to +3.6V (MAX3233E)
  +4.5V to +5.5V (MAX3235E)
- **250kbps Guaranteed Data Rate**
- **6V/µs Guaranteed Slew Rate**
- **Meets EIA/TIA-232 Specifications Down to 3.0V (MAX3233E)**
- **Internal Charge-Pump Capacitors**

Pin Configuration/ Functional Diagram

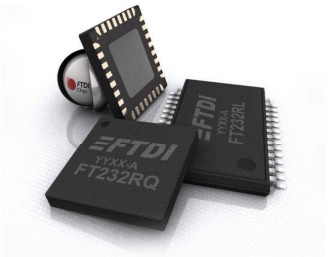


Typical Operating Circuit appears at end of data sheet.

Document No.: FT_000053
FT232R USB UART IC Datasheet Version 2.04
Clearance No.: FTDI# 38

Future Technology Devices International Ltd.

FT232R USB UART IC



The FT232R is a USB to serial UART interface with the following advanced features:

- Single chip USB to asynchronous serial data transfer interface.
- Entire USB protocol handled on the chip. No USB specific firmware programming required.
- Fully integrated 1024 bit EEPROM storing device descriptors and CBUS I/O configuration.
- Fully integrated USB termination resistors.
- Fully integrated clock generation with no external crystal required plus optional clock output selection enabling a glue-less interface to external MCU or FPGA.
- Data transfer rates from 300 baud to 3 Mbaud (RS422, RS485, RS232) at TTL levels.
- 128 byte receive buffer and 256 byte transmit buffer utilising buffer smoothing technology to allow for high data throughput.
- FTDI's royalty-free Virtual Com Port (VCP) and Direct (D2XX) drivers eliminate the requirement for USB driver development in most cases.
- Unique USB FTDIChip-ID™ feature.
- Configurable CBUS I/O pins.
- Transmit and receive LED drive signals.
- UART interface support for 7 or 8 data bits, 1 or 2 stop bits and odd / even / mark / space / no parity
- FIFO receive and transmit buffers for high data throughput.
- Synchronous and asynchronous bit bang interface options with RD# and WR# strobes.
- Device supplied pre-programmed with unique USB serial number.
- Supports bus powered, self powered and high-power bus powered USB configurations.
- Integrated +3.3V level converter for USB I/O.
- Integrated level converter on UART and CBUS for interfacing to between +1.8V and +5V logic.
- True 5V/3.3V/2.8V/1.8V CMOS drive output and TTL input.
- Configurable I/O pin output drive strength.
- Integrated power-on-reset circuit.
- Fully integrated AVCC supply filtering - no external filtering required.
- UART signal inversion option.
- +3.3V (using external oscillator) to +5.25V (internal oscillator) Single Supply Operation.
- Low operating and USB suspend current.
- Low USB bandwidth consumption.
- UHCI/OHCI/EHCI host controller compatible.
- USB 2.0 Full Speed compatible.
- -40°C to 85°C extended operating temperature range.
- Available in compact Pb-free 28 Pin SSOP and QFN-32 packages (both RoHS compliant).

FAIRCHILD SEMICONDUCTOR™

October 1996
Revised August 2004

NC7SZ14
TinyLogic® UHS Inverter with Schmitt Trigger Input

General Description

The NC7SZ14 is a single Inverter with Schmitt Trigger input from Fairchild's Ultra High Speed Series of TinyLogic®. The device is fabricated with advanced CMOS technology to achieve ultra high speed with high output drive while maintaining low static power dissipation over a very broad V_{CC} operating range. The device is specified to operate over the 1.65V to 5.5V V_{CC} range. The input and output are high impedance when V_{CC} is 0V. Inputs tolerate voltages up to 6V independent of V_{CC} operating voltage.

Features

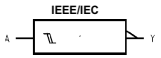
- Space saving SOT23 or SC70 5-lead package
- Ultra small MicroPak™ leadless package
- Ultra High Speed; t_{PD} 3.7 ns Typ into 50 pF at 5V V_{CC}
- High Output Drive; ±24 mA at 3V V_{CC}
- Broad V_{CC} Operating Range; 1.65V to 5.5V
- Matches the performance of LCX when operated at 3.3V V_{CC}
- Power down high impedance inputs/output
- Overvoltage Tolerant inputs facilitate 5V to 3V translation
- Patented noise/EMI reduction circuitry implemented

Ordering Code:

Order Number	Package Number	Product Code Top Mark	Package Description	Supplied As
NC7SZ14M5X	MA05B	7Z14	5-Lead SOT23, JEDEC MO-178, 1.6mm	3k Units on Tape and Reel
NC7SZ14P5X	MAA05A	Z14	5-Lead SC70, EIAJ SC-88a, 1.25mm Wide	3k Units on Tape and Reel
NC7SZ14L6X	MAC06A	B6	6-Lead MicroPak, 1.0mm Wide	5k Units on Tape and Reel

Logic Symbol

IEEE/IEC



Pin Descriptions

Pin Names	Description
A	Input
Y	Output
NC	No Connect

Function Table

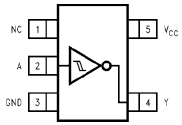
$Y = \overline{A}$

Input	Output
A	Y
L	H
H	L

H = HIGH Logic Level
L = LOW Logic Level

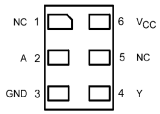
Connection Diagrams

Pin Assignments for SOT23 and SC70



(Top View)

Pad Assignments for MicroPak



(Top Thru View)

TinyLogic® is a registered trademark of Fairchild Semiconductor Corporation.
MicroPak™ is a trademark of Fairchild Semiconductor Corporation.

 DS012176

19-1224; Rev 3; 1/07

MAXIM

High-Output-Drive, Precision, Low-Power, Single-Supply, Rail-to-Rail I/O Op Amps with Shutdown

General Description

The MAX4165–MAX4169 family of operational amplifiers combines excellent DC accuracy with high output current drive, single-supply operation, and rail-to-rail inputs and outputs. These devices operate from a single +2.7V to +6.5V supply, or from dual ±1.35V to ±3.25V supplies. They typically draw 1.2mA supply current, and are guaranteed to deliver 80mA output current.

The MAX4166/MAX4168 have a shutdown mode that reduces supply current to 38µA per amplifier and places the outputs into a high-impedance state. The MAX4165–MAX4169's precision performance combined with high output current, wide input/output dynamic range, single-supply operation, and low power consumption makes them ideal for portable audio applications and other low-voltage, battery-powered systems. The MAX4165 is available in the space-saving 5-pin SOT23 package and the MAX4166 is available in a tiny 2mm x 2mm x 0.8mm µDFN package.

Selector Guide

PART	AMPS PER PACKAGE	SHUTDOWN MODE
MAX4165	Single	—
MAX4166	Single	Yes
MAX4167	Dual	—
MAX4168	Dual	Yes
MAX4169	Quad	—

Applications

Portable/Battery-Powered Audio Applications
Portable Headphone Speaker Drivers
Laptop/Notebook Computers
Sound Ports/Cards
Set-Top Boxes
Cell Phones
Hands-Free Car Phones (kits)
Signal Conditioning
Digital-to-Analog Converter Buffers
Transformer/Line Drivers
Motor Drivers

Typical Operating Circuit appears at end of data sheet.

Features

- **80mA (min) Output Drive Capability**
- **Rail-to-Rail Input Common-Mode Voltage Range**
- **Rail-to-Rail Output Voltage Swing**
- **1.2mA Supply Current per Amplifier**
- **+2.7V to +6.5V Single-Supply Operation**
- **5MHz Gain-Bandwidth Product**
- **250µV Offset Voltage**
- **120dB Voltage Gain (R_L = 100kΩ)**
- **88dB Power-Supply Rejection Ratio**
- **No Phase Reversal for Overdriven Inputs**
- **Unity-Gain Stable for Capacitive Loads to 250pF**
- **Low-Power Shutdown Mode: Reduces Supply Current to 38µA Places Outputs in High-Impedance State**
- **Available in 5-Pin SOT23 Package (MAX4165) or 2mm x 2mm x 0.8mm µDFN (MAX4166)**

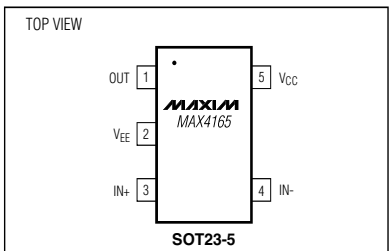
Ordering Information

PART	TEMP RANGE	PIN-PACKAGE	TOP MARK
MAX4165EUK-T	-40°C to +85°C	5 SOT23-5	AABY
MAX4166EPA	-40°C to +85°C	8 Plastic DIP	—
MAX4166ESA	-40°C to +85°C	8 SO	—
MAX4166EUA	-40°C to +85°C	8 µMAX	—
MAX4166ELA+T	-40°C to +85°C	8 µDFN-8	AAG

+Denotes lead-free package.

Ordering Information continued on last page.

Pin Configurations



Pin Configurations continued at end of data sheet.

19-1360; Rev 3; 3/00

EVALUATION KIT MANUAL FOLLOWS DATA SHEET

MAXIM

High-Efficiency, Low-Supply-Current, Compact, Step-Up DC-DC Converters

General Description

The MAX1674/MAX1675/MAX1676 compact, high-efficiency, step-up DC-DC converters fit in small µMAX packages. They feature a built-in synchronous rectifier, which improves efficiency and reduces size and cost by eliminating the need for an external Schottky diode. Quiescent supply current is only 16µA.

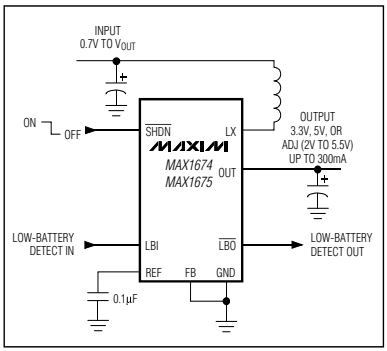
The input voltage ranges from 0.7V to V_{OUT}, where V_{OUT} can be set from 2V to 5.5V. Start-up is guaranteed from 1.1V inputs. The MAX1674/MAX1675/MAX1676 have a preset, pin-selectable output for 5V or 3.3V. The outputs can also be adjusted to other voltages using two external resistors.

All three devices have a 0.3Ω N-channel MOSFET power switch. The MAX1674 has a 1A current limit. The MAX1675 has a 0.5A current limit, which permits the use of a smaller inductor. The MAX1676 comes in a 10-pin µMAX package and features an adjustable current limit and circuitry to reduce inductor ringing.

Applications

Pagers
Wireless Phones
Medical Devices
Hand-Held Computers
PDAs
RF Tags
1 to 3-Cell Hand-Held Devices

Typical Operating Circuit



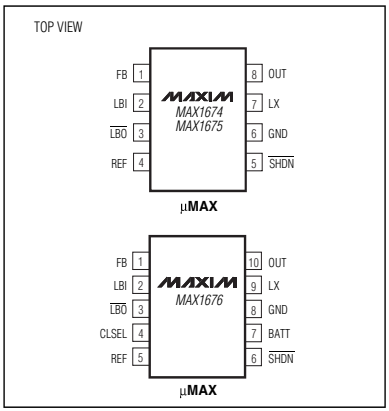
Features

- **94% Efficient at 200mA Output Current**
- **16µA Quiescent Supply Current**
- **Internal Synchronous Rectifier (no external diode)**
- **0.1µA Logic-Controlled Shutdown**
- **LBI/LBO Low-Battery Detector**
- **Selectable Current Limit for Reduced Ripple**
- **Low-Noise, Anti-Ringing Feature (MAX1676)**
- **8-Pin and 10-Pin µMAX Packages**
- **Preassembled Evaluation Kit (MAX1676EVKIT)**

Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE
MAX1674EUA	-40°C to +85°C	8 µMAX
MAX1675EUA	-40°C to +85°C	8 µMAX
MAX1676EUB	-40°C to +85°C	10 µMAX

Pin Configurations



Apéndice 2

Extracto del código del microcontrolador

- Main
- Inicio de la adquisición
- Interrupción de temperatura
- Registro de temperatura
- Interrupción del pluviómetro

```
/************MAIN****************/
 int main(void)
{

    DDRD|=(1<<DDD6); //LED's pin as an output
    PORTB |= (1<<PB0)|(1<<PB1)|(1<<PB3); //PB1, PB1 and PB3 are
the dip switch inputs

    if ((!(PINB & 0x02))|(!(PINB & 0x01))) { //check the pinB4
and pinB1, the statement is executed if PINDB4 or PINDB1 Swit-
ches are closed
        PORTD |= (1<<PD6);

        if (!(PINB & 0x02)) //PB0 int 1
        {

            format_directory (); //format the external EEPROM
memory (24LC1025)

        }
        if(!(PINB & 0x01)) //PB1 int 2
        {
            format_eeprom(); //format the internal EEPROM

        }
            while(1) //infinite loop, the programs start again
until is reset and the switches are placed off again
            asm volatile ("nop");
    }

// If no  format is needed, the execution continues

    USART_Init (MYUBRR); //configure and start USART

    conf_default(); //default time in the RTC

    GICR|=(1<<INT1); //enable external interrupt 1

    lesen_verz( &mut_byte_ad,&mut_chip_ad);//initialize the
pointer in case the user wants to download data
    //save the initial addresses for ring mode configuration
```

```
    start_chip_address = mut_chip_ad;
    start_byte_address = mut_byte_ad;

    sei(); //enable external interrupts
    set_sleep_mode(SLEEP_MODE_PWR_DOWN );// define power save
mode: Power down
    sleep_enable(); //enable power save mode
    PORTD |= (1<<PD3);  //PD3 is configured as an output
    PORTB |= (1<<PB4); //PB4 is configured as an output

    while(1)
    {
    GICR|=(1<<INT1); //enable external interrupt 1 this will be
raised when the main sw is pressed
    sleep_cpu(); //sleep the microcontroller, the program conti-
nues only by interruptions
    asm volatile("nop"); //dummy nop
    }
    return 0;

}
/************ACQUISITION START***************/
unsigned char Start_Sys (void)
/* This function enables the acquisition interrupts according
with the established configuration*/
{

    read_adqtype(&acc_type); //read from memory what kind of ac-
quisition
    //will be performed: temperature, rainfall or both
    blink(); //blink LED to indicate start of acquisition
    oper=1; //the system is operating

    read_memorybeh(&memory_mode); //stores the mode in which the
memory is going to behave

    if (acc_type=='a') //record temperature
    {

        reg_rate = eeprom_read_word(reg_rate_EEPROM); //read
from EEPROM the rate in minutes
```

```
        //get the rate of measurement
        //this is defined as the minimum value of samples that
can be made
        //in the register interval without bypassing 30 measure-
ments
        while(1)
        {
             if(30*rate_min >= reg_rate)
             break;
             else
             rate_min++;
        }

        lesen_verz(&byte_ad_temp, &chip_ad_temp); //initialize
the recording pointers by reading the memory 24Lxx

        mut_byte_ad = byte_ad_temp; //the last available address
is in the rain gauge pointers
        mut_chip_ad = chip_ad_temp;

        config_ADC(); //configure the ADC
        set_conf (1); //enable one minute alarm
        clear_flag(); //clear the interrupt flag of the RTC
        _delay_ms(50); //this delay is to giving change to the
signal to get enough low to
                       // prevent a interrupt
        GICR |= (1<<INT0); //enable external interrupt 0

    }

    if (acc_type=='b') //record rainfall
    {

        lesen_verz(&byte_ad, &chip_ad); //initialize the recor-
ding pointers by reading the memory 24Lxx

        mut_byte_ad = byte_ad; //the last available address is
in the rain gauge pointers
        mut_chip_ad = chip_ad;
```

```
        MCUCSR|=(1<<ISC2); //interrupt on rising edge
        GIFR=(1<<INTF2); //clear the flag that could be set
        GICR|=(1<<INT2); //enable external interrupt 2 connected
to the rain gauge
    }

    if (acc_type=='c') //record both
    {
        reg_rate = eeprom_read_word(reg_rate_EEPROM); //read
from EEPROM the rate in minutes

        MCUCSR|=(1<<ISC2); //interrupt on rising edge
        GIFR=(1<<INTF2); //clear the flag that could be set
        GICR|=(1<<INT2); //enable external interrupt 2 connected
to the rain gauge

        while(1)  //get the rate of measurement
        {
             if(30*rate_min >= reg_rate)
             break;
             else
             rate_min++;
        }

        lesen_verz(&byte_ad, &chip_ad); //initialize the recor-
ding pointers by reading the memory 24Lxx

        byte_ad_temp = byte_ad;
        chip_ad_temp = chip_ad;
        mutual_pointer_inc (& byte_ad_temp, & chip_ad_temp); //
the temperature pointer will have the next page

        mut_byte_ad = byte_ad_temp; //the last available address
is in the temperature pointers
        mut_chip_ad = chip_ad_temp;

        config_ADC();
        set_conf (1); //enable 1 minute alarm
        clear_flag();
        _delay_ms(50);
        GICR |= (1<<INT0); //enable external interrupt 0
```

```
    }

}

/************THERMISTOR ACQUISITION INTERRUPT***************/
ISR(INT0_vect) //External Interrupt 0 code: Thermistor alarm A
from RTC

{
    clear_flag(); //clear the signal interrupt from the RTC

    add_count++; //increment the counters
    reg_count++;

    if(add_count==rate_min) //if it's time to acquire data from
the thermistor
    {

        data_mean[index]=ADC_mean(); // in the vector of tempe-
rature measures add one more
        add_count=0; // the add_count counter is reset
        index++; //the index is increment by one, this will help
us in other
                 //part of the code (regist()) to generate the
mean from all the measurements

    }

    if(reg_count==reg_rate) //if it's time to register a value
in memory execute the code
    {

        regist(); // register the temperature value in memory
        reg_count=0; //reset the counter for a new loop
        index = 0; //reset the index

    }

}
```

```
/************TEMPERATURE REGISTER****************/
void regist() //temperature register in EEPROM
{
    unsigned char i;
    unsigned int mean=0;
    unsigned char eql; //the comparison is true calculate the
mean of all the measurements made
    for (i=0;i<index;i++)
    {
        mean = mean+data_mean[i];

    }
    mean = mean/index;

    st = 0; // the system had recorded dates for first time

    if (!(byte_ad_temp % 128)) //if the data is multiple from
128 we need to write in another page but before we need to pre-
pare it
    {

        schreibt_verz();
        lesen_verz(&byte_ad_temp, &chip_ad_temp);
        reinigen_Speicher (&byte_ad_temp,&chip_ad_temp);
        header_temp (reg_rate, &byte_ad_temp,&chip_ad_temp);
        mutual_pointer_inc(&mut_byte_ad,&mut_chip_ad);
        eql = Compare_Addresses (&mut_chip_ad,&mut_byte_ad,
                    &start_chip_address,&start_byte_address);
        if(eql==0x01)
            if (memory_mode == 'a')
            {
                Stop_Sys(); //if the mode was set to stop when
the memory is full
                return; //exit the function no other acquisition
has to be made
            }
            if (memory_mode == 'b')
            {
                set_memoryendreached(); //the memory ended and
now the registers are been overwritten
```

```
            }
    }

        //write the temperature on memory

        write_temp (&byte_ad_temp,&chip_ad_temp,mean);

        if (!(PINB & 0x04))
        {
            blink_2//blink the LED to acknowledge that a measu-
rement has been recorded
        }
}

/************RAINGAUGE ACQUISITION INTERRUPT***************/
ISR(INT2_vect) //rain gauge interrupt
{
    unsigned char eql;

    if (test_mode) //if test_mode was set
    {
        USART_Trasmit(0xFF);
        USART_Trasmit(0x01); //transmit a token to the computer
    }
    else
    {
        asm volatile ("nop"); //dummy nop

        st = 0; //the system has started

        _delay_ms(5); //delay for avoiding a possible rebound

        if (!(byte_ad % 128)) { // if the current page is full
record the change and jump to another one

            schreibt_verz();
            lesen_verz(&byte_ad, &chip_ad);
            reinigen_Speicher (&byte_ad,&chip_ad);
            header_otemp (&byte_ad, &chip_ad);
            mutual_pointer_inc(&mut_byte_ad,&mut_chip_ad);
```

```
            eql = Compare_Addresses (&mut_chip_ad,&mut_byte_ad,
                        &start_chip_address,&start_byte_address);

            if(eql==0x01)
            {
                if (memory_mode == 'a')
                {
                    Stop_Sys(); //if the mode was set to stop
when the memory is full
                    return; //exit the function no other acqui-
sition needs to be made
                }
            if (memory_mode == 'b')
                {
                    set_memoryendreached(); //the memory ended
and now the registers are been overwritten
                }
            }
        }

        write_pluv_otemp (&byte_ad,&chip_ad); //write the vale
in memory

        if (!(PINB & 0x04)) //blink if the switch is set
        {
            blink_2();//blink the LED to acknowledge that a mea-
surement has been recorded
        }
    }
}
```