UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FISICAS Y MATEMATICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

"OSCILOSCOPIO PORTÁTIL USB CON CONEXIÓN DE RED PARA MONITOREO REMOTO"

MEMORIA PARA OPTAR AL TITULO DE INGENIERO CIVIL ELECTRICISTA

NÉSTOR MANUEL PALOMINOS GONZÁLEZ

PROFESOR GUÍA:
MAURICIO BAHAMONDE BARROS

MIEMBROS DE LA COMISION:
NÉSTOR BECERRA YOMA
HÉCTOR AUGUSTO ALEGRÍA

SANTIAGO DE CHILE
OCTUBRE 2008

RESUMEN DE LA MEMORIA
PARA OPTAR AL TITULO DE
INGENIERO CIVIL ELECTRICISTA
POR: NESTOR PALOMINOS G.
FECHA: 27/10/2008

"OSCILOSCOPIO PORTÁTIL USB CON CONEXIÓN DE RED PARA MONITOREO REMOTO"

El presente trabajo de título consistió en la elaboración de un osciloscopio digital de propósito general, sencillo de implementar y de bajo costo.

El sistema está dividido en tres fases principales: la etapa de adquisición y acondicionamiento de la señal, transporte de la información por USB y transmisión de dicha información dentro de una red LAN utilizando microcontroladores PIC.

En cuanto al acondicionamiento de la señal, se diseñó en base a lo visto en la revisión bibliográfica una atenuación inicial y posterior amplificación variable mediante amplificadores operacionales de ganancia variable, controlada digitalmente mediante la multiplexión de un arreglo de resistencias que definen la ganancia. La idea fundamental fue acondicionar señales de distinta amplitud al rango dinámico del conversor análogo-digital utilizado (0 – 5 volts).

Una vez convertido el dato mediante interrupciones que garantizan la separación constante de tiempo entre muestras, se guarda dentro de la RAM interna del PIC y mediante transacciones PC-Microcontrolador PIC18F2550 se envían paquetes de datos por USB, los cuales son procesados y posteriormente graficados.

Paralelo a esto, se trasfieren los datos captados a un segundo PIC (18F452), el cual define una conexión TCP/IP mediante el controlador Ethernet ENC28J60 y envía paquetes con la información recibida dentro de una LAN.

Independiente al método de adquisición utilizado (USB o Ethernet), se diseñó un software basado en capas o módulos, el cual recibe la información, y la guarda en una cadena, y dependiendo si está o no definida la opción de trigger, localiza el punto de disparo y lo centra en pantalla, lo cual da la posibilidad de implementar también un control de desplazamiento vertical.

Palabras clave: *Osciloscopio, USB, Ethernet, Conversión analógico-digital*

Dedicatoria:

Este trabajo esta dedicado a mi madre ya mi abuela, quienes me apoyaron desde el comienzo; a mis compañeros con los que pasamos trasnoches de estudio antes de los exámenes.. al Pablo, Patiño, Lucho, Yofer, Orlando, Llopez, y un montón de viejos cracks con quienes pasamos mas de un susto en algún recupera ya olvidado.. Exceso de café y tabaco.. y de cerveza luego del examen, claro..

También a mi profesora de básica, Ana Santana, quien fue la primera piedra.. y A Sergio Pozo por enseñarme a prender luces y a jugar con motores cuando tenia 7 años. También mención aparte a Mario Pozo, su padre, quien también me enseño muchos aspectos de la ciencia y la tecnología que me ayudaron mucho en el camino

A cada persona que marco otro capitulo mas de mi vida…con quienes pude experimentar tanto el sabor del triunfo como del fracaso...que al final es lo que compone la vida y que es lo que no se enseña en clases….

Gracias... Totales =)

Agradecimientos:

Principalmente Le agradezco a Pablo Olivares por su notable aporte al trabajo.. Por enseñarme a programar microcontroladores y trucos de programación.. También por hacerme ver varios vacíos conceptuales y ayudarme a taparlos. Te debo un café compare'

También se agradece la perseverancia y la confianza que tuvieron en mi a mis profesores guía (Mauricio Bahamonde) , co-guia (Néstor Becerra) e integrante (Hector Augusto)

También mención honrosa a los foreros de todoPIC en especial a J1M a RedPic y Nocturno por sus grandes proyectos en microcontrol y su buena disposición para explicarlos y que cada vez el open source vaya ganando mas terreno.

INDICE DE CONTENIDOS

INDICE DE FIGURAS 7

CAPITULO 1: INTRODUCCION Y OBJETIVOS 10

1.1 Fundamentos y objetivos generales. 10
1.2 Objetivos específicos. 11
1.3 Hipótesis de trabajo y metodología. 11
1.4 Plan de trabajo. 12
1.5 Infraestructura disponible. 13

CAPITULO 2: 14

2.1 El Osciloscopio 14
2.1.1 Introducción 14
2.1.2 Diagrama de bloques 15
2.1.3 Etapa de entrada y acondicionamiento de la señal 17
2.1.3.1 Divisores de tensión 17
2.1.3.2 Atenuación 18
2.1.3.3 Amplificadores de ganancia programable 18
2.1.3.4 Amplificadores sumadores 20
2.1.4 Conversión analógico digital 22
2.1.5 Técnicas de muestreo 23
2.1.6 Teorema de Nyquist 24
2.1.7. Sistema de deflexión vertical y filtrado 26
2.1.8 Sistema de deflexión horizontal 26
2.1.9 Ancho de banda 26
2.1.11 Trigger 26
2.1.12 Relación Señal a ruido (SNR) 27
2.1.13 Errores de cuantizacion 27
2.1.13.1 Error de compensación 28
2.1.13.2 Error de ganancia 28
2.1.13.3 Error de apertura 29
2.1.13.4 Error de no-linealidad diferencial 30
2.1.13.5 Error de no-linealidad integral 30
2.1.13.7 Error absoluto 32
2.1.12 Parámetros de una hoja de datos para un osciloscopio digital 33

2.2 El puerto USB 34
2.2.1 Introducción 34
2.2.2 Funcionamiento y topología 34
2.2.3 Pinout USB 36
2.2.4 Potencia 37
2.2.5 Librería de funciones USB 37
2.2.6 Estructura del paquete USB 38

2.3 Conectividad de red **40**
2.3.1 Introducción **40**
2.3.2 Modelo OSI **40**
2.3.3 Protocolos **42**
2.3.3.1 Protocolos de red 42
2.3.3.1.1 El protocolo IP 42
2.3.3.2 Protocolos de transporte 43
2.3.3.2.1 El protocolo TCP 43
2.3.3.2.2 El protocolo UDP 44
2.3.3.3 Standard de enlace 45
2.3.3.3.1 Ethernet 45

2.3.4 Capa física **48**
2.3.4.1 Codificación de señales ethernet 49
2.3.4.2 Sistemas con par trenzado (10BASE-T) 49
2.3.4.3 Elementos del sistema 10BASE-T 50
2.3.4.4 Polaridad de señal y polaridad reversible 50

CAPITULO 3: IMPLEMENTACION DEL PROYECTO **51**

3.1 Características de componentes **51**
3.1.1 Conversión análogo-digital y microcontrol **51**
3.1.2 Amplificación de ganancia programable **57**
3.1.3 Controlador Ethernet **60**

3.2 ETAPA DE ENTRADA Y PROTECCIONES **62**

3.3 PRIMERA FASE: USB **73**

3.3.1 HARDWARE **73**
3.3.1.1 Conversión ADC, Microcontrol y Transmisión USB 73
3.3.1.2 PCB del proyecto. 75
3.3.2 FIRMWARE USB **77**
3.3.2.1 Estructura del código 77
3.3.2.2 Tiempo de muestreo 79
3.3.3 SOFTWARE USB **81**
3.3.3.1 Recepción de datos 82
3.3.3.2 Mecanismo de Trigger 82
3.3.3.3 Gráfico de datos 85
3.3.3.4 Sistemas de deflexión horizontal y vertical 87

3.4 SEGUNDA FASE: ETHERNET 88
3.4.1 HARDWARE 88
3.4.2 Firmware 91
3.4.3 Software 97

CAPITULO 4: DISCUSION DE RESULTADOS 99

4.1 Sobre la selección de componentes. 99
4.3 Comunicación USB 102
4.4 Comunicación Ethernet 105
4.5 Puesta en marcha 109

CAPITULO 5: CONCLUSIONES 110

BIBLIOGRAFIA 112

ANEXOS 116

***A.1 FIRMWARE USB* 116**

A.2 SOFTWARE 119
A.2.1 VENTANA PRINCIPAL 119
A.2.2 FUNCIONES USB 122
A.2.3 ADQUISICION DE DATOS 124
A.2.4 GRAFICOS 125
A.2.5 TRIGGER 127
A.2.6 ANALISIS DE FOURIER 129
A.2.7 GUARDAR IMAGEN 130

INDICE DE FIGURAS

Capítulo 2

Figura 2.1: Osciloscopio analógico y osciloscopio digital
Figura 2.2: Diagrama de bloques de un osciloscopio analógico
Figura 2.3: Señal sin trigger (izquierda) y con trigger (derecha)
Figura 2.4: Diagrama de bloques de un osciloscopio digital
Figura 2.5: Divisor de tensión básico
Figura 2.6: Amplificador inversor de ganancia programable
Figura 2.7: Amplificador no inversor de ganancia programable
Figura 2.8: Amplificador sumador
Figura 2.9: Análisis de amplificador sumador.
Figura 2.10: Conversión A/D.
Figura 2.11: Comparación de interpolaciones.
Figura 2.12: Muestreo en tiempo equivalente.
Figura 2.13: Muestreo en tiempo equivalente
Figura 2.14: Efectos del muestreo a distintas Fs
Figura 2.15: Trigger sin especificar flanco
Figura 2.16: Error de compensación
Figura 2.17: Error de ganancia
Figura 2.18: Error de apertura
Figura 2.19: Error de no linealidad diferencial
Figura 2.20: Error de no linealidad integral
Figura 2.21: Error de cuantización
Figura 2.22: Error de absoluto
Figura 2.23: Jerarquía de descriptores y endpoints
Figura 2.24: Pinout USB
Figura 2.25: Tipos de paquetes USB
Figura 2.26: Modelo OSI
Figura 2.27: Encabezado IP
Figura 2.28: Encabezado TCP
Figura 2.29: Encabezado UDP
Figura 2.30: Encabezado Ethernet y 802.3
Figura 2.31: Pinout RJ45

Capítulo 3

[3.1 Selección de componentes]

Figura 3.1: Pin-out del microcontrolador PIC18F2550
Figura 3.2: Modelo del conversor análogo-digital del PIC18F2550
Figura 3.3: Multiplexion de canales análogo-digital del PIC18F2550
Figura 3.4: Requerimientos del conversor análogo-digital del PIC18F2550
Figura 3.5: Diagrama de tiempo del conversor
Figura 3.6: Diagrama de bloques USB
Figura 3.7: Líneas de datos diferenciales USB
Figura 3.8: Amplificador de ganancia programable
Figura 3.9: Opción 2 para el diseño del amplificador de ganancia programable
Figura 3.10: Pin-out del multiplexor CMOS4051
Figura 3.11: Pin out LM324
Figura 3.12: Pin out ENC28J60
Figura 3.13: Diagrama de bloques ENC28J60
Figura 3.14 Conexiones ENC28J60

[3.2 Etapa de entrada y protecciones]

Figura 3.15: Divisor de tensión
Figura 3.16: Señal atenuada
Figura 3.17: Sumador de componente continua
Figura 3.18: Atenuación y desplazamiento
Figura 3.19: Amplificador no inversor
Figura 3.20: Amplificador no inversor con selector de ganancia manual
Figura 3.21: Diseño de amplificador de ganancia programable
Figura 3.22: Arreglo de resistencias seleccionadas mediante multiplexion
Figura 3.23: Diagrama de flujo para el acondicionamiento de una señal
Figura 3.24: Etapas del acondicionamiento
Figura 3.25: Diagrama de flujo para la amplificación de la señal acondicionada (opción 1)
Figura 3.26: Diagrama de flujo para la amplificación de la señal acondicionada (opción 2)

[3.3.1 Etapa de adquisición y transmisión de datos por USB - Hardware]

Figura 3.27: Bloques de entrada y salida al microcontrolador
Figura 3.28: Prototipo implementado (Simulación)
Figura 3.29: Diseño del PCB en Eagle
Figura 3.30: PCB del proyecto en EAGLE ya ruteado

[3.3.2 Etapa de adquisición y transmisión de datos por USB - Firmware]

Figura 3.31: Diagrama de flujo para la conversión de datos y transmisión por USB
Figura 3.32: Estructura de la interrupción e intercambio de paquetes USB

[3.3.3 Etapa de adquisición y transmisión de datos por USB - Software]

Figura 3.33: Modelo de capas para el software
Figura 3.34 Esquema de funcionamiento del trigger
Figura 3.35: Diagrama de flujo para la adquisición de datos
Figura 3.36: Diagrama de flujo para graficar los puntos
Figura 3.37: Interfaz gráfica

[3.4.1 Etapa de transmisión de datos por Ethernet - Hardware]

Figura 3.38: Obtención de 5 volts para placa PICWEB
Figura 3.39: Obtención de 3.3 volts para placa PICWEB
Figura 3.40: Bloque controlador Ethernet
Figura 3.41: Conector RJ45
Figura 3.42: Transductor MAX232 y conector DB9
Figura 3.43: Microcontrolador PIC18F452

[3.4.2 Etapa de transmisión de datos por Ethernet - Firmware]

Figura 3.44: Pin-out Puerto EXT
Figura 3.45: Ubicación del pin 1
Figura 3.46: Conexionado entre placas
Figura 3.47: Diagrama de flujo para la adquisición de datos y transmisión por Ethernet
Figura 3.48: Registro de control del ADC
Figura 3.49: Diagrama de flujo para la segunda alternativa
Figura 3.50: Handshacking entre placas
Figura 3.51: Diagrama Frame Ethernet

[3.4.2 Etapa de transmisión de datos por Ethernet - Software]

Figura 3.52: Conmutación USB-Ethernet

Capitulo 4

Figura 4.1: Pinout CMOS4066
Figura 4.2: Pinout Decodificador 74hc138
Figura 4.3: Pinout Negador 74hc04
Figura 4.4: Opción 1 para el diseño del amplificador de ganancia programable
Figura 4.5: Pin Out LM741
Figura 4.6: Lectura ADC y Transmisión USB
Figura 4.7: Lectura ADC y Transmisión USB (medición de tiempos)
Figura 4.8: Ciclo principal
Figura 4.9: Configuración de parámetros de red
Figura 4.10: Negociación DHCP
Figura 4.11: Red utilizada para pruebas
Figura 4.12: Resultados de PING
Figura 4.13: Estructura de un paquete TCP
Figura 4.14: Tren de pulsos de 4.59 kHz y 2.775 Vrms
Figura 4.15: Equivalente en el equipo implementado

CAPITULO 1: INTRODUCCION Y OBJETIVOS

1.1 FUNDAMENTOS Y OBJETIVOS GENERALES.

Esta memoria consiste en la elaboración de un sistema de adquisición y visualización de señales de propósito general, reducido en tamaño y costo, con la posibilidad de poder conectarlo a una red local para monitorear señales remotamente.

La finalidad de este trabajo surge a partir de la necesidad de contar con un equipo que sea una alternativa a osciloscopios comerciales. En especial porque éstos son instrumentos de laboratorio y se dañarían fácilmente en un ambiente industrial.

Entre las aplicaciones que se le puede dar están:

- Procesos industriales de adquisición de datos, como por ejemplo presión o temperatura, previa conexión a un transductor, para su monitoreo remoto en tiempo real, sin tener que trasladarse de un punto a otro en una planta y con la posibilidad de guardar estos datos para su posterior procesamiento en una base de datos.

- Sistema remoto de alarmas, es decir, al obtener alguna frecuencia anómala relacionada con el fallo de una máquina, se puede programar un sistema que cuando detecte dicha anomalía, avise al operador si el sistema está fallando.

- Es ideal para fines educativos, ya que es factible de elaborarlo para una sala de clases completa, tal de que cada alumno tenga uno para experimentar, debido a su costo

- También tiene la ventaja de su portabilidad como se mencionó anteriormente, ya que un operador puede trasladarlo fácilmente de un punto a otro a diferencia de un osciloscopio convencional.

1.2 OBJETIVOS ESPECÍFICOS.

Con el trabajo de esta memoria se persiguen los siguientes objetivos específicos:

1º Diseñar un dispositivo que digitalice y grafique señales externas con conversores análogo-digital.

2º El dispositivo podrá conectarse a un computador mediante el puerto USB para el monitoreo local de la señal y a una red LAN para el monitoreo remoto.

3º Se creará una etapa de adaptación del nivel de voltaje de la señal con el fin de no dañar el dispositivo.

4º Se implementará un software tal, que el operador pueda ver gráficamente la señal muestreada, con la opción de procesar esta misma y desplegar su FFT en pantalla.

5º Este dispositivo deberá cumplir la condición de portabilidad y costo reducido para su potencial producción en serie, tal que pueda competir en el mercado con osciloscopios simples y conversores de aplicabilidad similar.

6º El dispositivo deberá tener protecciones con el fin de no poner en riesgo la integridad del usuario, el equipo a analizar, el computador al cual se conectará y/o la red a la que se enviarán los datos.

7º Se elaborará un manual de usuario, y de mantenimiento técnico los cuales describirán satisfacer las necesidades de forma de uso, configuración tanto de hardware como de software y de mantenimiento técnico.

1.3 HIPÓTESIS DE TRABAJO Y METODOLOGÍA.

En una primera fase, se verán los distintos dispositivos de características similares disponibles en el mercado con el fin de establecer los márgenes económicos para diseñar un dispositivo sea competitivo.

Los componentes a utilizar en el diseño dependerán además de su capacidad para afrontar el proyecto, de su disponibilidad y precio. Como resultado de esta etapa, se obtendrá una hoja de especificaciones y pruebas sobre las que se basará el diseño del dispositivo, tanto en hardware como en software.

Luego de definir los componentes a utilizar, se realizará el diseño del hardware propiamente tal, ver qué funciones serán implementadas por software y cuáles por hardware.

1.4 PLAN DE TRABAJO.

Para la planificación del trabajo, se pretende separar en dos tareas el proyecto: la primera tarea comprende el semestre primavera 2007 y corresponde al diseño del osciloscopio sólo con funcionalidad USB (Universal Serial Bus). La segunda tarea, correspondiente al período del semestre otoño 2008, es añadirle la funcionalidad de red al equipo para poder así hacer un monitoreo remoto.

Para la primera fase se contempla una etapa de evaluación y cotización de productos existentes en el mercado. Además, será necesario analizar la amplia gama de posibilidades en cuanto a selección de componentes para el diseño del dispositivo según los requerimientos que se tengan.

Posteriormente, luego de haber hecho la investigación bibliográfica, se hará necesario montar un prototipo del equipo.

Una vez terminada la implementación del hardware base, será necesario hacer la programación tanto del PIC, como del PC. Para la programación del PIC se dispone del software CCS (fase USB) y MICROCHIP C18 (Fase Ethernet), los cuales transforman un código fuente en C a un archivo .hex y el software ICPROG y WINPIC para enviar este archivo al dispositivo

Posteriormente, se simulará el circuito en el software PROTEUS, el cual es ideal para este trabajo ya que se pueden simular PICs indicando el archivo .hex a probar, además de disponer de un puerto USB virtual.

Una vez programado el PIC, se diseñará e implementará el software en el PC, utilizando librerías correspondientes al enfoque del proyecto.

En cuanto al driver del dispositivo USB, existen open sources genéricos ya implementados para esta familia de microcontroladores de Microchip, con posibilidad de modificarlos en el camino, adaptándolo al proyecto.

También será necesario realizar una etapa de adaptación de la señal a muestrear en cuanto a la amplitud del voltaje de entrada, como medida de protección, ya que de lo contrario pondría en riesgo tanto al chip como al computador al cual se conectará.

Además se realizara el PCB (Printed Circuit Borrad) correspondiente en EAGLE (software para crearlos) y finalmente fresarlo y soldarlo.

En este punto, se evaluará si es posible usar el mismo chip o acoplarle uno especializado para esta tarea (para no disminuir el rendimiento en cuanto a muestreo) y que así se pueda trabajar en paralelo (un chip que se dedique a digitalizar los datos y otro que cumpla la función de resolver una IP y trabajar como un host de una red).

Como comunicación entre ambas fases, se puede mencionar una intercomunicación entre puertos de propósito general paralelos o seriales disponibles en esta familia de microcontroladores.

Respecto a la estructura del presente informe, este esta dividido en cinco capítulos mas una sección de anexos: el primer capitulo (actual) corresponde a la introducción al tema y los objetivos que se quieren realizar; el capitulo dos, se refiere a la investigación previa sobre las materias a utilizar; el tercer capitulo es el desarrollo del proyecto propiamente tal ; en el capitulo cuatro se pone a prueba el equipo donde se analizan resultados; y en el quinto y ultimo se sacan las conclusiones respectivas y posibles mejoras. En la sección anexos básicamente va el código desarrollado organizado en forma de módulos específicos

1.5 INFRAESTRUCTURA DISPONIBLE.

Para realizar esta memoria se cuenta con el apoyo del laboratorio de electrónica (del Departamento de Ingeniería Eléctrica de la Universidad de Chile), el cual dispone de instrumentos tales como osciloscopios, generadores de funciones, fuentes de poder y programadores de PICs.

Además, se cuenta con una fresa especializada en la elaboración de PCB's, que funciona a partir del software EAGLE disponible en los computadores del laboratorio.

Para lograr una mayor independencia se obtuvo el programador PG2C de OLIMEX, hardware que se conecta por RS232 y cuya finalidad es grabar el código en el microcontrolador

CAPITULO 2:

CARACTERISTICAS DE UN OSCILOSCOPIO Y TRANSMISION DE DATOS

En este capítulo se comentará sobre las principales clasificaciones y características de los osciloscopios, además de la transmisión de datos mediante USB y Ethernet, lo cual será la base teórica de este trabajo.

2.1 El Osciloscopio

2.1.1 Introducción

Un osciloscopio corresponde básicamente a un dispositivo de visualización gráfica que muestra señales eléctricas variables en el tiempo. El eje vertical (Y) representa el voltaje, mientras que el eje horizontal (X) representa el tiempo [1]. Básicamente existen las siguientes aplicaciones para estos dispositivos: determinar directamente el período y el voltaje de una señal, determinar indirectamente la frecuencia de una señal y localizar fallas en un circuito.

Un osciloscopio puede medir un gran número de fenómenos provisto del transductor adecuado. Esto corresponde a un elemento que convierte una magnitud física en señal eléctrica, con el cual el osciloscopio será capaz de darnos el valor de una presión, ritmo cardíaco, potencia de sonido, etc. [2].

Existen dos clasificaciones básicas de osciloscopios: analógicos y digitales. Los analógicos son aquellos que trabajan con señales continuas de voltaje y los digitales los que trabajan con señales discretas de voltaje

Un osciloscopio analógico trabaja modificando un haz de electrones sobre una pantalla directamente mediante el voltaje introducido. En cambio, un osciloscopio digital previamente debe muestrear la señal a medir usando un conversor análogo-digital (ADC) para, posteriormente, reconstruir la señal en pantalla [3]. Esto implica la aparición de un error de cuantizacion, del cual se hablará posteriormente

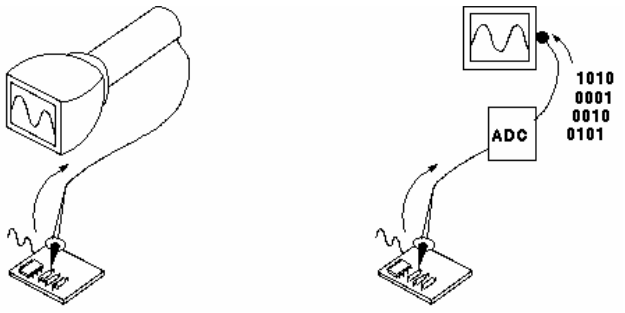


Figura 2.1: Osciloscopio analógico (izquierda) y osciloscopio digital (derecha).

Generalmente se prefieren los osciloscopios analógicos, cuando se quiere medir rápidamente señales en tiempo real. En cambio se prefieren los osciloscopios digitales, cuando se desea guardar la señal para procesarla posteriormente.

2.1.2 Diagrama de bloques

Para aclarar el método de adquisición de datos de ambas categorías se muestra sus diagramas de bloques respectivos y una breve descripción de éstos.

A continuación se muestra el diagrama de bloques de un osciloscopio analógico:

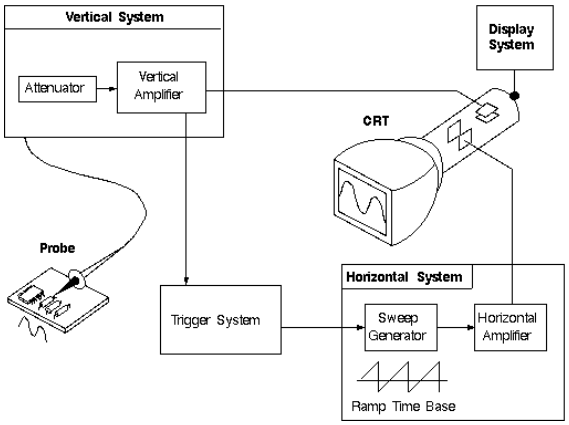


Figura 2.2: Diagrama de bloques de un osciloscopio analógico.

Cuando se conecta la punta de prueba a determinado circuito, la señal de voltaje viaja hacia el sistema de deflexión vertical. Dependiendo de la configuración la escala vertical, un atenuador reduce la señal de voltaje y un amplificador ajusta la ganancia para que la señal se vea nítidamente [4].

Posteriormente, la señal viaja directamente a las placas de deflexión vertical de un tubo de rayos catódicos, donde el haz de electrones se mueve hacia arriba o hacia abajo, dependiendo si el voltaje es positivo o negativo.

También la entrada viaja al sistema de disparo o trigger para un barrido horizontal. La función de trigger consiste en sincronizar una señal en pantalla para que se pueda ver y analizar sin que esta se mueva considerablemente en pantalla [5].

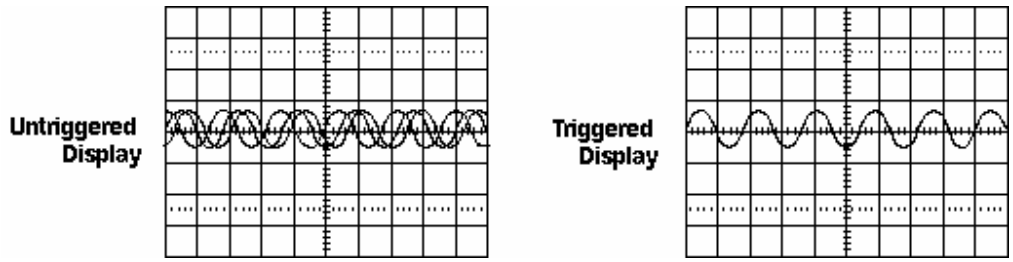


Figura 2.3: Señal sin trigger (izquierda) y con trigger (derecha).

En resumen, para utilizar un osciloscopio analógico, es necesario configurar la atenuación o amplificación en la fase de entrada (volts/div), la escala de tiempo (seg/div) y el trigger o disparo (flanco de subida o bajada).

A continuación se muestra el esquema de un osciloscopio digital:

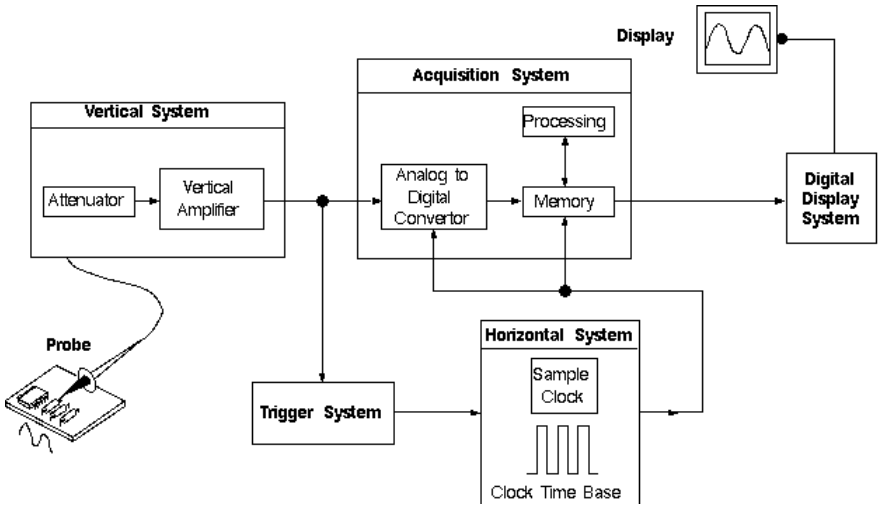


Figura 2.4: Diagrama de bloques de un osciloscopio digital.

A diferencia de los osciloscopios analógicos, los osciloscopios digitales contienen más bloques para su procesamiento. Se observa que existe una fase de entrada donde se acondiciona la señal para su muestreo, para posteriormente ser ingresada a un conversor análogo-digital [6].

En este punto, la señal continua se muestrea, mientras que un sistema de reloj digital, da el paso para adquirir el dato siguiente. Estas muestras por segundo definen la llamada tasa de muestreo.

Posteriormente, estos datos se guardan en una "ventana" de longitud determinada según la escala de tiempo que se quiera ver.

También para señales periódicas se define una etapa de trigger, la cual puede ser diseñada tanto antes como después de la conversión análoga digital.

En resumen, para tomar una medida en un osciloscopio digital hay que ajustar la escala vertical, horizontal y el trigger, como en el caso de los osciloscopios analógicos.

2.1.3 Etapa de entrada y acondicionamiento de la señal

2.1.3.1 Divisores de tensión

Las entradas de los conversores análogo digital soportan un máximo de tensión. Generalmente se usa un máximo de 5 Volts. Por lo tanto, para monitorear señales de mayor amplitud, es necesario adaptar la entrada a las condiciones exigidas por el conversor [9].

Lo normal en estos casos es recurrir a un divisor de tensión, el cual se muestra en la figura 2.5:

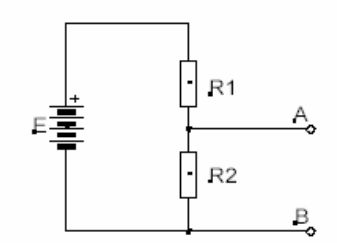


Figura 2.5*: Divisor* de tensión básico.

En la figura, se tiene una resistencia Rt definida como:

$$R_T = R_1 + R_2$$

Además, por ley de Ohm se cumple:

$$I = \frac{V_E}{R_T} \qquad I = \frac{V_1}{(R_1 + R_2)}$$

La tensión de salida Vs se puede calcular entonces como:

$$V_S = I * R_2$$

$$V_S = \frac{V_E \cdot R_2}{(R_1 + R_2)}$$

2.1.3.2 Atenuación

A los canales de un osciloscopio se conectan las puntas o sondas de medición. Estos terminales están específicamente diseñadas para trabajar con este dispositivo.

La mayoría presentan la posibilidad de atenuar la señal de entrada en un factor de 10. Por convención la atenuación se indica con una X después del factor (10X, 100X) y la amplificación con una x antes del factor (x10, x100). El utilizar o no la atenuación afecta los rangos de medida, tanto en amplitud como en frecuencia.

Cuando se está utilizando el factor de atenuación se vuelve importante la impedancia capacitiva del conjunto punta-osciloscopio.

Dado que los distintos osciloscopios poseen distintas capacidades en las puntas, éstos cuentan con un condensador variable en paralelo que permite adaptarlas a los diferentes instrumentos.

2.1.3.3 Amplificadores de ganancia programable

Un amplificador programable se define como aquel cuyo funcionamiento puede ser modificado mediante señales de control [10].

El parámetro que interesa modificar es la ganancia, la cual permite en un osciloscopio digital, acondicionar la entrada antes de entrar al conversor análogo digital.

Sumado al divisor de tensión, es posible crear un sistema retroalimentado con la finalidad de poder muestrear distintos rangos de voltaje, sin perder calidad en la señal de salida, lo cual se verá posteriormente en la sección protecciones.

Se pueden clasificar según la configuración en que esté el amplificador operacional.

Amplificador Inversor:

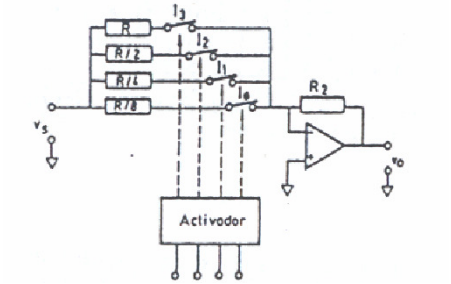


Figura 2.6: Amplificador inversor de ganancia programable.

En este caso se cumple que:

$$v_o = -\frac{R_2}{R_R} v_s$$

En esta configuración, esta el inconveniente de que hay inversión del signo, además de que, al activarse se interruptor, existirá una resistencia implícita en serie con la resistencia externa, lo cual se traduce como un error que hay que considerar.

Amplificador no Inversor:

Con esta configuración se arregla el error del cambio de signo.

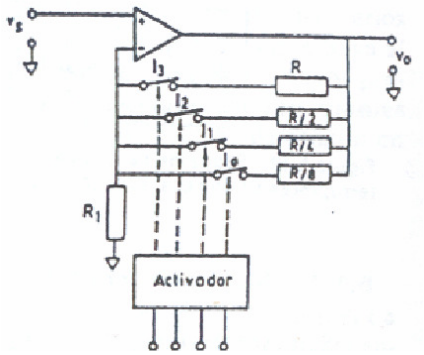


Figura 2.7: Amplificador no inversor de ganancia programable.

2.1.3.4 Amplificadores sumadores

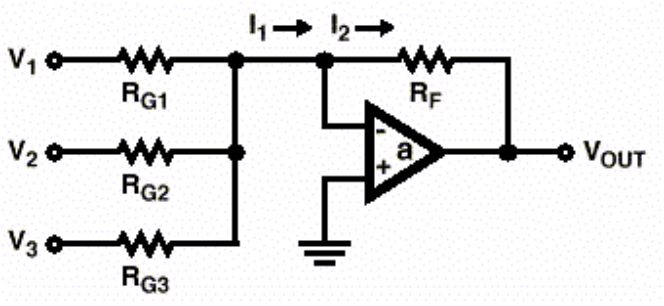


Figura 2.8: Amplificador sumador.

En el circuito de la figura 2.8, como en el amplificador inversor, la tensión V(+) está conectada a tierra, por lo que la tensión V(-) estará conectada a una tierra virtual, y como la impedancia de entrada es infinita, toda la corriente I1 circulará a través de RF, la cual se denominará I2. Lo que ocurre en este caso es que la corriente I1 es la suma algebraica de las corrientes proporcionadas por V1, V2 y V3, es decir:

$$I_1 = \frac{V_1}{R_{g1}} + \frac{V_2}{R_{g2}} + \frac{V_3}{R_{g3}}$$

Y también:

$$I_2 = -\frac{V_{out}}{R_f}$$

Como I1 = I2 se concluye que:

$$V_{out} = -\left(\frac{V_1 R_f}{R_{g1}} + \frac{V_2 R_f}{R_{g2}} + \frac{V_3 R_f}{R_{g3}} \right)$$

Que establece que la tensión de salida es la suma algebraica invertida de las tensiones de entrada multiplicadas por un factor corrector, que se puede observar que en el caso en que

$$R_F = R_{G1} = R_{G2} = R_{G3} \rightarrow V_{OUT} = -(V_1 + V_2 + V_3).$$

La ganancia global del circuito la establece RF, que, se comporta como en el amplificador inversor básico. A las ganancias de los canales individuales se les aplica independientemente los factores de escala R_{G1} R_{G2} R_{G3} ,... etc. Del mismo modo, R_{G1} R_{G2} R_{G3} son las impedancias de entrada de los respectivos canales.

Otra característica interesante de esta configuración es el hecho de que la mezcla de señales lineales, en el nodo suma, no produce interacción entre las entradas, puesto que todas las fuentes de señal alimentan el punto de tierra virtual. El circuito puede acomodar cualquier número de entradas añadiendo resistencias de entrada adicionales en el nodo suma. Aunque los circuitos precedentes se han descrito en términos de entrada y de resistencias de realimentación, las resistencias se pueden reemplazar por elementos complejos, y los axiomas de los amplificadores operacionales se mantendrán como verdaderos. Dos circuitos que demuestran esto, son dos nuevas modificaciones del amplificador inversor [11].

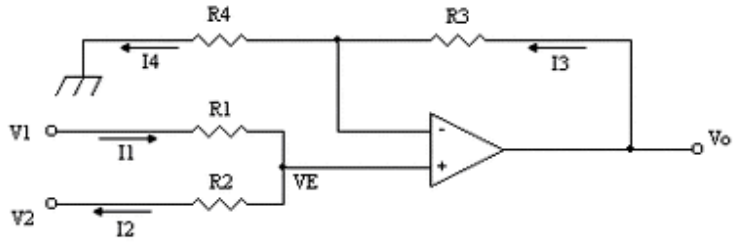


Figura 2.9: Análisis de amplificador sumador.

$$I_1 = I_2 \quad I_3 = I_4$$

$$I_4 = \frac{V_e - 0}{R_4} \quad I_3 = \frac{V_o - V_e}{R_3} \quad I_1 = \frac{V_1 - V_e}{R_1} \quad I_2 = \frac{V_e - V_2}{R_2}$$

$$\frac{V_1 - V_e}{R_1} = \frac{V_e - V_2}{R_2} \Rightarrow R_2(V_1 - V_e) = R_1(V_e - V_2)$$

$$V_1 \cdot R_1 - V_e \cdot R_2 = V_e \cdot R_1 - V_2 \cdot R_1 \Rightarrow V_1 \cdot R_2 + V_2 \cdot R_1 = V_e \cdot R_1 + V_e \cdot R_2$$

$$V_1 \cdot R_2 + V_2 \cdot R_1 = V_e(R_1 + R_2)$$

$$V_e = \frac{V_1 \cdot R_2 - V_2 \cdot R_1}{R_1 + R_2}$$

$$\frac{Vo - VE}{R3} = \frac{VE}{R4} \Rightarrow R4 \cdot (Vo - VE) = R3 \cdot VE \Rightarrow R4 \cdot Vo - R4 \cdot VE = R3 \cdot VE \Rightarrow R4 \cdot Vo = R3 \cdot VE + R4 \cdot VE \Rightarrow$$

Además:

$$\frac{Vo - VE}{R3} = \frac{VE}{R4} \Rightarrow R4 \cdot (Vo - VE) = R3 \cdot VE \Rightarrow R4 \cdot Vo - R4 \cdot VE = R3 \cdot VE \Rightarrow R4 \cdot Vo = R3 \cdot VE + R4 \cdot VE \Rightarrow$$

$$R4 \cdot Vo = VE \cdot (R3 + R4) \Rightarrow VE = \frac{R4 \cdot Vo}{R3 + R4}$$

Igualando las expresiones para Ve:

$$\frac{R4 \cdot Vo}{R3+R4} = \frac{V1 \cdot R2 + V2 \cdot R1}{(R1+R2)} \Rightarrow Vo = \frac{(V1 \cdot R2 + V2 \cdot R1)}{R1+R2} \cdot \frac{R3+R4}{R4}$$

La expresión final de Vo se puede simplificar para el supuesto de que el valor en paralelo de R1 y R2 sea igual al valor en paralelo de R3 y R4.

$$\frac{R1 \cdot R2}{R1+R2} = \frac{R3 \cdot R4}{R3+R4} \Rightarrow R3+R4 = \frac{R3 \cdot R4}{R1 \cdot R2} \cdot (R1+R2) \Rightarrow$$

$$Vo = \frac{V1 \cdot R2 + V2 \cdot R1}{R1+R2} \cdot \frac{\frac{R3 \cdot R4}{R1 \cdot R2}(R1+R2)}{R4} = V1R2 + V2R1\frac{R3}{R1 \cdot R2}$$

$$Vo = \frac{R3}{R1} \cdot V1 + V2 \cdot \frac{R3}{R2}$$

2.1.4 Conversión analógico digital

El conversor analógico-digital del sistema de adquisición de datos muestrea la señal a intervalos de tiempo determinados y convierte la señal de voltaje continua en una serie de valores digitales llamados *muestras*. En el eje horizontal del grafico, una señal de reloj determina cuando el conversor A/D toma una muestra [7]. La velocidad de este reloj se denomina *velocidad de muestreo* y se mide en muestras por segundo.

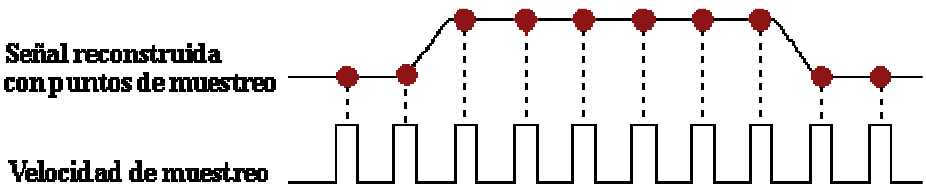


Figura 2.10: Conversión A/D.

Los valores digitales muestreados se almacenan en una memoria como puntos de señal. El número de los puntos de señal utilizados para reconstruir la señal en pantalla se denomina registro. La sección de disparo determina el comienzo y el final de los puntos de señal en el

registro. La sección de visualización recibe estos puntos del registro, una vez almacenados en la memoria, para presentar en pantalla la señal.

2.1.5 Técnicas de muestreo

Un punto fundamental en el funcionamiento de un osciloscopio digital es la forma de reunir los puntos de muestreo. Para señales de lenta variación, los osciloscopios digitales pueden perfectamente reunir más puntos de los necesarios para reconstruir posteriormente la señal en la pantalla [8]. No obstante, para señales de mayor frecuencia, el osciloscopio no puede recoger muestras suficientes y debe recurrir a una de las técnicas siguientes:

Interpolación:

Corresponde a estimar un punto intermedio de la señal basándose en el punto anterior y posterior.

Muestreo en tiempo equivalente:

Si la señal es repetitiva es posible muestrear durante unos cuantos ciclos en diferentes partes de la señal para después reconstruir la señal completa.

Muestreo en tiempo real con interpolación.

El método standard de muestreo en los osciloscopios digitales es el muestreo en tiempo real: el osciloscopio reúne los suficientes puntos como para reconstruir la señal. Para señales no repetitivas o la parte transitoria de una señal es el único método válido de muestreo.

Los osciloscopios utilizan la interpolación para poder visualizar señales que son más rápidas que su velocidad de muestreo. Existen básicamente dos tipos de interpolación:

Lineal: Simplemente conecta los puntos muestreados con líneas.

Senoidal: Conecta los puntos muestreados con curvas según un proceso matemático, de esta forma, los puntos intermedios se calculan para rellenar los espacios entre puntos reales de muestreo. Usando este proceso es posible visualizar señales con gran precisión disponiendo de relativamente pocos puntos de muestreo.

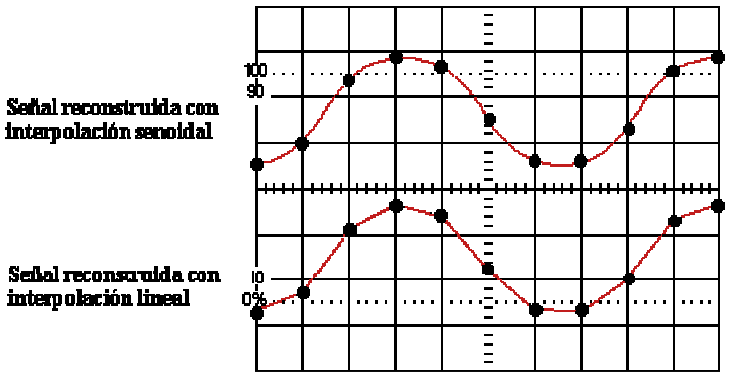


Figura 2.11: Comparación de interpolaciones.

Muestreo en tiempo equivalente.

Algunos osciloscopios digitales utilizan este tipo de muestreo. Se trata de reconstruir una señal repetitiva capturando una pequeña parte de la señal en cada ciclo.

Existen dos tipos básicos:

Muestreo secuencial: Los puntos aparecen de izquierda a derecha en secuencia para conformar la señal.

Muestreo aleatorio: Los puntos aparecen aleatoriamente para formar la señal.

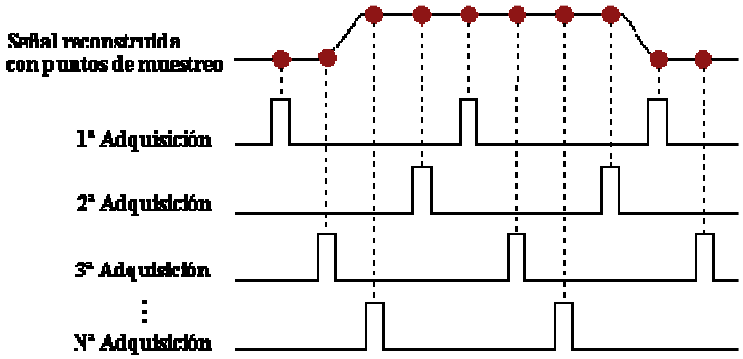


Figura 2.12: Muestreo en tiempo equivalente.

2.1.6 Teorema de Nyquist

Previamente conviene hacer la distinción entre muestreo y cuantificación. Ambos se relacionan en el proceso de digitalización de una señal, donde la etapa de cuantificación es posterior a la de muestreo, además de ser un proceso irreversible (se pierde información) [8].

Las muestras discretas de una señal son valores que no han pasado por un redondeo, sino que son valores exactos en tiempos discretos definidos.

El teorema de Nyquist demuestra que una reconstrucción exacta de una señal periódica continua en banda base a partir de sus muestras es matemáticamente posible, si la señal está limitada en banda y la tasa de muestreo es superior al doble de su ancho de banda.

La información completa de la señal analógica que cumple este criterio se define por la serie total de muestras que resultaron del proceso de muestreo.

Matemáticamente, si se tiene una señal continua x(t) donde su frecuencia más alta es Fmax=B, y esta señal se muestrea a una tasa Fs>2Fmax=2B entonces x(t) es completamente recuperable a partir de sus muestras.

Como observación, cabe mencionar que el concepto de ancho de banda no siempre coincide con el valor de la frecuencia más alta. Cuando se cumple esto, a la señal se le llama de "banda base" y no toda señal cumple este requisito. Además, si el teorema de Nyquist no se cumple, en el muestreo aparecerá un efecto de aliasing, que consiste en frecuencias en el que el muestreo coincide con otra, como se aprecia en la figura 2.13. En esta figura, los primeros tres gráficos cumplen la condición de que Fs no puede superar a la mitad de la máxima frecuencia de la señal, no axial el cuarto grafico, donde la señal reconstruida no representara a la verdadera

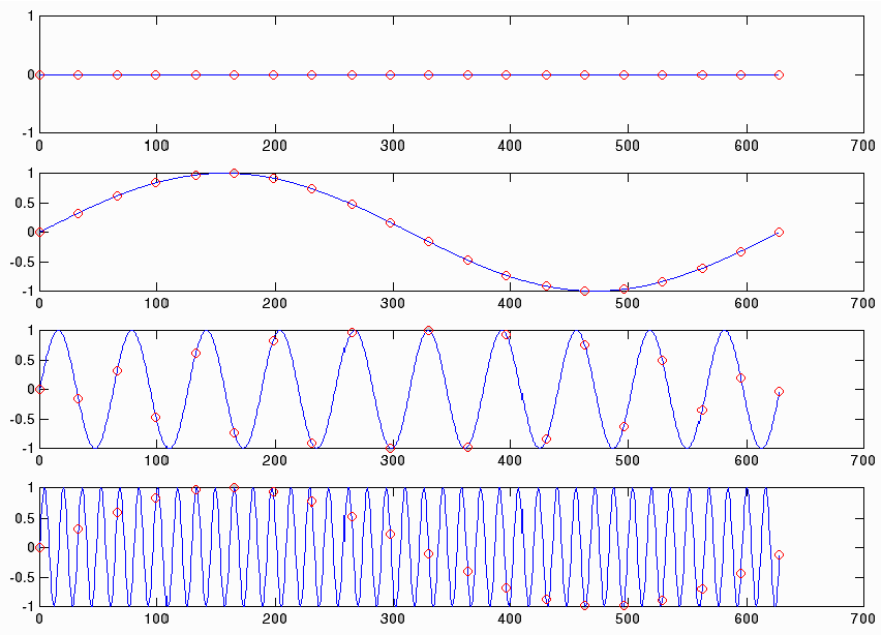


Figura 2.13: Efectos del muestreo a distintas Fs.

2.1.7. Sistema de deflexión vertical y filtrado

Este sistema de deflexión controla la escala de amplitud de la señal y cada control aparece por separado para cada uno de los canales.

La escala de amplitud se mide en Volt/división y se controla mediante un conmutador central en saltos discretos de, aproximadamente, un orden de magnitud.

Otro control permite desplazar el "cero" de la escala vertical. Por último, un selector de tres posiciones permite acondicionar la señal: en la posición DC la señal se visualiza tal cual llega a las terminales, en la posición AC un condensador filtra la componente continua de la señal y en la posición GND la señal se desconecta permitiendo ubicar el cero [3].

2.1.8 Sistema de deflexión horizontal

El sistema de deflexión controla la base de tiempo. Al igual que con el eje vertical, existe un control para la posición del cero y dos para la escala temporal.

2.1.9 Ancho de banda

El ancho de banda determina la capacidad básica de un osciloscopio para medir una señal. Conforme aumenta la frecuencia de la señal, disminuye la capacidad del osciloscopio para presentar la señal con exactitud.

Esta especificación indica el rango de frecuencia que el osciloscopio puede medir con precisión.

El ancho de banda de un osciloscopio se define como la frecuencia a la cual una señal sinusoidal se presenta atenuada un 70,7% respecto a la amplitud real de la señal. Este punto se conoce como el punto a -3 dB, término basado en una escala logarítmica.

Sin un ancho de banda adecuado, el osciloscopio no podrá resolver los cambios de alta frecuencia. La amplitud se distorsionará. Los flancos se desvanecerán. Los detalles se perderán. Sin un ancho de banda adecuado, todas las características y prestaciones de un osciloscopio no tendrán ningún valor.

2.1.10 Precisión de la ganancia

La precisión de la ganancia indica la exactitud con que el sistema vertical atenúa o amplifica una señal, habitualmente indicada como un porcentaje de error.

2.1.11 Trigger

Al momento de visualizar una señal con el fin de estudiarla se vuelve importante la sincronización entre la misma y el barrido temporal.

Si la señal es periódica es posible verla estatica en la pantalla sólo si se encuentra en sincronía con la señal que controla el barrido horizontal. De esta manera, cada vez que el haz regrese al extremo izquierdo de la pantalla, la señal a medir tendrá la misma amplitud y el recorrido del punto luminoso será el mismo que en el barrido anterior.

Para lograr esto, el osciloscopio posee un control del "Nivel de disparo" (Trigger level) con el cual se determina el valor umbral de voltaje que debe alcanzar la señal a medir para que el haz comience a barrer la pantalla. El ejemplo más simple e ilustrativo es el de una señal sinusoidal. Si se determina que el barrido comience cuando la señal pase por cero volt se vería lo mostrado en la figura 2.14:

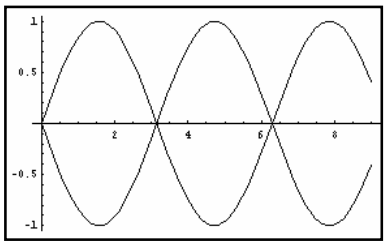


Figura 2.14: Trigger sin especificar flanco.

Se observa que, en efecto, el gráfico comienza en un cero de la señal de entrada. De esta manera, el barrido continúa hasta el final de la pantalla, en ese momento el instrumento espera a que el valor especificado se repita y reinicia el recorrido.

También se puede observar un problema, dado que una señal sinusoidal pasa por cero dos veces por período, en la pantalla se superponen dos recorridos diferentes. Para evitar esto el osciloscopio cuenta con otro control que permite seleccionar el tipo de flanco (subida o bajada) que debe tener la señal de entrada para iniciar el barrido. Así, se elimina uno de los dos gráficos anteriores.

2.1.12 Relación Señal a ruido (SNR)

La relación señal a ruido es una razón de señal deseada a indeseado de la señal (ruido) en el promedio de nivel de potencia de una transmisión.

Si hay demasiado ruido en un circuito, la relación señal a ruido es baja. Si el circuito es de buena calidad, la relación señal / ruido será alto.

2.1.13 Errores de cuantizacion

Estos errores son parte de la etapa de conversión de la señal analógica en una representación digital de la misma. A continuación presentamos los diferentes tipos de errores que encontramos en este proceso [8].

2.1.13.1 Error de compensación

El error de compensación está definido como la diferencia entre la compensación nominal y la que realmente se tiene. Para el caso de un conversor analógico-digital, el punto de compensación nominal es el valor que se tiene en la entrada para obtener una salida igual a cero según muestra la figura 2.15.

Cuando en la entrada se coloca este supuesto valor y en su salida se obtiene un valor diferente de cero es, justamente, cuando existe un error de compensación. Esto bien puede corregirse ajustando los valores de compensación, y si no es posible ajustarlos, entonces se puede realizar en la etapa de análisis de los datos obtenidos, haciendo los ajustes necesarios en estos valores.

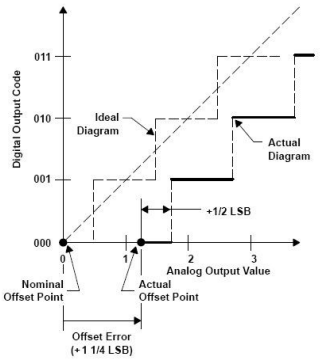


Figura 2.15: Error de compensación.

2.1.13.2 Error de ganancia

El error de ganancia se define como la diferencia entre el valor de ganancia nominal (ganancia esperada, calculada) y la ganancia real. Este valor se calcula cuando el error por compensación es nulo, caso contrario se obtendrá un error en un cálculo generado por otro error.

La figura 2.16 muestra de forma simple el significado de este error. Cuando se tiene un valor conocido a la entrada del conversor, de espera un valor de salida determinado. Cuando la diferencia entre el valor esperado y el obtenido es constante e igual sin importar cual sea el valor en su entrada, entonces se trata de un error de ganancia. Este error es lineal y constante en todo el rango posible de entrada. Este error es posible solucionarlo, aunque también se puede corregir en etapas posteriores.

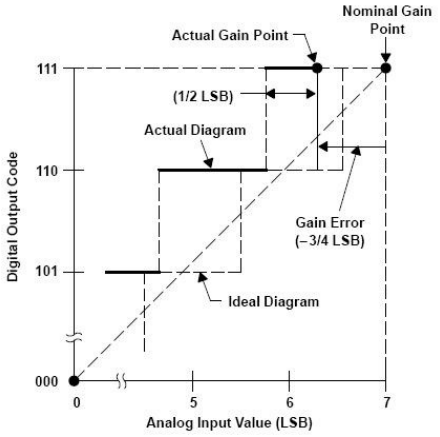


Figura 2.16: Error de ganancia.

2.1.13.3 Error de apertura

El error de apertura es un tipo de error generado en los propios componentes y no es corregible. La incertidumbre en la señal de entrada en el momento en que se muestrea y retiene dicha señal (sample&hold) está dado desde el instante en que se la muestrea hasta que se la retiene, antes de pasar al proceso de conversión. El error de apertura puede ser generado por ruidos o también por variaciones en la señal de reloj. Este error también influye en las características finales del sistema.

Este error puede ser reducido si el tiempo de muestreo y retención disminuye. Si este valor es lo más pequeño posible, entonces se tiene mayor certeza de que la conversión realizada es correcta según se muestra en la figura 2.17

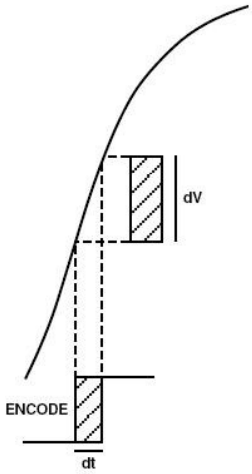


Figura 2.17: Error de apertura.

2.1.13.4 Error de no-linealidad diferencial

El error de no-linealidad diferencial es la diferencia entre la variación de tensión nominal que genera un cambio en un bit a la salida del conversor de 1 LSB (Least Significant Byte) y la variación real que debe ocurrir. Es decir, que si el cambio de tensión que debe ocurrir a la entrada para que haya un cambio en la salida de un bit es exactamente el esperado, entonces el error DNL es cero. En cambio, si para que haya un cambio en la salida, el diferencial de tensión a la entrada debe ser mayor (o menor) al que se especifica, entonces existe este tipo de error. Incluso este tipo de error puede generar que existan valores binarios de salida que nunca se lleguen a dar, debido que el rango de entrada tiene como respuesta una menor cantidad de valores debido al error citado (cuando el diferencial de tensión de entrada es mayor que el especificado para 1 LSB).

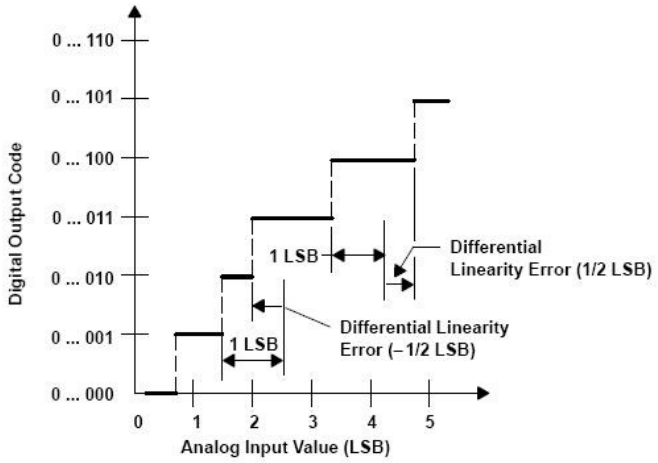


Figura 2.18: Error de no linealidad diferencial.

2.1.13.5 Error de no-linealidad integral

El error de no-linealidad integral está definido como la desviación de los valores de la función de transferencia real sobre una línea recta. Esta línea recta puede ser definida como la mejor aproximación que minimice estas desviaciones, o bien entre los puntos extremos de la función de transferencia una vez que se hayan anulado los errores de ganancia y compensación. Este segundo método es conocido como “linealidad de punto final”, y es el más usado ya que su verificación es más simple.

En un conversor analógico-digital esta desviación se mide en la transición de un paso al siguiente (es decir, un aumento en 1 LSB), y el nombre de error integral proviene de que se trata de la suma de estas desviaciones desde el nivel más bajo hacia un valor determinado. Con esta definición se puede conocer entonces el error causado por la no-linealidad integral en cada valor.

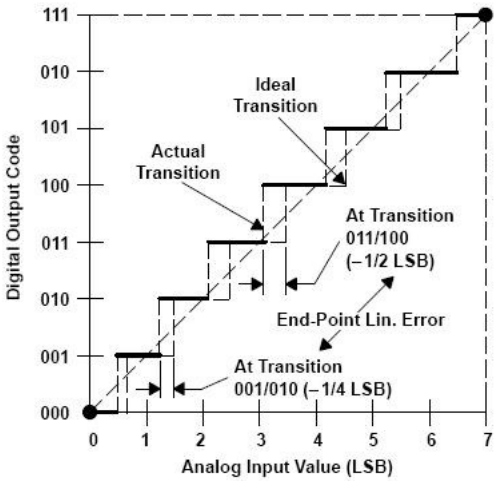


Figura 2.19: Error de no linealidad integral.

2.1.13.6 Error de cuantización

El error de cuantización está basado en la naturaleza de las señales analógicas y las digitales. Una señal analógica es continua y puede tener infinitos valores (por más que la señal esté acotada), mientras que una señal digital es discreta, tiene una cantidad finita de valores posibles.

Aquí entonces radica este error. Al intentar traducir una señal que puede tener infinitos valores en otra que solo puede tener valores finitos, esta claro que se pierde información. La cantidad de valores posibles (o estados posibles) que puede tener la señal digital está relacionada con la cantidad de bits con la cual ésta se representa. Sin embargo la cantidad de bits es una cantidad finita. La única forma de hacer que este error sea nulo, es haciendo que la cantidad de bits con la que se representa el valor digital sea infinito. Esto implica que cierto valor digital representa a muchos (de hecho, infinitos) valores analógicos posibles. De esta forma, un valor analógico produce una salida digital, y esa salida digital si se vuelve a convertir a un valor analógico, puede que no corresponda con el valor original. Cierta información se ha perdido en este proceso, y esto se conoce como error de cuantización.

En el caso de un conversor ideal, donde la función de salida puede creerse como una escalera perfecta, el error entre la señal real de entrada y su correspondencia con la salida digital, tiene una función de densidad de probabilidad uniforme en el caso de que se asume que la entrada es totalmente aleatoria. Este error varía en el rango de ±1/2 LSB, o bien, ±q/2, donde q es el ancho de un escalón, tal se muestra en la siguiente figura.

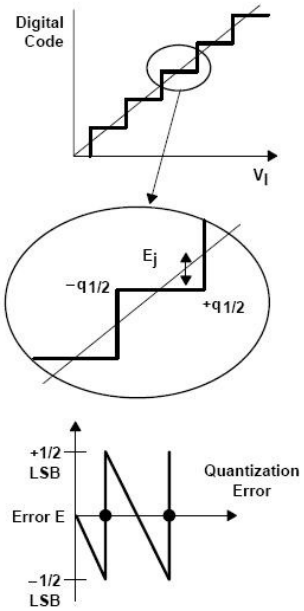


Figura 2.20: Error de cuantización.

2.1.13.7 Error absoluto

El error absoluto en la exactitud del proceso de conversión es la diferencia máxima que se encuentra entre el valor analógico de la señal de entrada con el valor medio que se define para ese mismo código binario de salida. Este error es absoluto y es por ello que incluye a todos los errores citados previamente (compensación, ganancia, no linealidad e, incluso, cuantización), según muestra la figura 2.21.

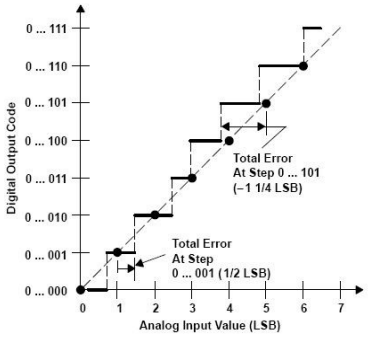


Figura 2.21: Error de absoluto.

2.1.12 Parámetros de una hoja de datos para un osciloscopio digital

En la hoja de datos de un osciloscopio comercial, se pueden ver parámetros más significativos, que ayudarán a tener una idea sobre la precisión de los resultados que se obtendrán con el dispositivo. Estos datos se pueden separar en tres bloques: sistema de deflexión vertical, sistema de deflexión horizontal, y disparo o trigger. Si el sistema además se conecta a un computador, también es necesario indicar los requerimientos mínimos de éste. A continuación se muestra la hoja de datos de un osciloscopio comercial

Sistema de deflexión vertical

Entrada	2 canales entrada, 1 canal de disparo externo / salida de compensación del generador
Impedancia de entrada	1 MOhm/20 pF
Rango del factor de deflexión	10 mV/div a 5 V/div en secuencia 1-2-5
Precisión	+- 2.5 %
Resolución	8 bits (0.4 %)
Tensión máxima de entrada	+- 100 V

Sistema de deflexión horizontal

Rango de la base de tiempos	Tiempo real - 1 µs/div a 2 s/div Repetitivo - 10 ns/div a 2 s/div
Frecuencias de muestreo	Tiempo real - 1 kHz a 50 MHz Repetitivo - 1kHz a 5 GHz (Frecuencia de muestreo equivalente)
Otros	Base de tiempos dual

Disparo

Fuente de disparo	Canal A, Canal B, Externo
Selección del umbral	Canal A, Canal B en todo el tango de presentación, Externo fijado sobre 1.2 V
Opciones de disparo	- Disparo por flanco ascendente o descendente - Tras un número definido de muestras después de la condición de disparo (Filtro digital) - Tras un número definido de condiciones de disparo
Tamaño de almacenamiento	Máx 32000 muestras para la base de tiempo más rápida. [Para bases de tiempo lentas (50KS/s) está limitado por la memoria del ordenador]

Requerimientos del sistema

Computador	Procesador Pentium con puerto usb 1.1 o superior
Sistema operativo	Windows 98 / Windows 2000 / Windows XP

2.2 El puerto USB

2.2.1 Introducción

USB (Universal Serial Bus) corresponde a un estándar creado con el fin de conectar dispositivos externos a un computador de forma serial de alta velocidad en comparación con el puerto serie convencional y el puerto paralelo, los cuales dominaron durante muchos años [12].

Este estándar fue creado por el USBIF (USB Implementers Forums), la cual está compuesta por distintas empresas de computación y electrónica tales como Intel, HP y Microsoft.

Se han desarrollado hasta el momento tres versiones de USB, mejorando cada vez más la tasa de transferencia (sin dejar de ser compatible una versión con sus anteriores), lo cual implica la posibilidad de crear dispositivos cada vez más veloces compatibles con el estándar tales como almacenamiento masivo o video.

En cuanto a las tasas de transferencia, es posible mencionar lo siguiente:

Low speed	–	1.5 Mbit/s
Full Speed	–	12 Mbit/s
High Speed	–	480 Mbits/s

En cuanto al diseño, es posible decir que USB posee un controlador principal "host" al cual se le pueden conectar distintos dispositivos en cadena, análogamente a los Hubs utilizados para redes de computadores.

Por lo tanto se crea una estructura de árbol, la cual tiene un límite máximo de 127 dispositivos funcionando a la vez, conectados a la raíz.

2.2.2 Funcionamiento y topología

Los dispositivos USB están asociados a *pipes*, canales lógicos unidireccionales, los cuales conectan al host controlador con un *endpoint*, que representa una sección del dispositivo que satisface un propósito específico tal como recibir o transmitir datos [13].

Para la transferencia de datos, la información es encapsulada en paquetes de diversos tamaños de largo relacionado con potencias de 2.

Tanto los endpoints como los pipes son enumerados del 0 al 15 en cada dirección (entrada y salida). Esto significa que un dispositivo puede tener hasta 32 endpoints. Esta dirección está vista tomando como referencia el host controlador. Esto quiere decir que un endpoint de salida corresponde a un canal que transmite desde el host controlador hacia el dispositivo.

Un endpoint tiene asociada sólo una dirección, salvo el endpoint 0 (bidireccional), el cual está reservado para el bus de control.

Al conectar un dispositivo USB, se crea el denominado proceso de enumeración, en el cual el host controlador asigna una dirección de 7 bits que se usa a modo de identificación del dispositivo (similar a la función de una dirección IP en Internet). Posteriormente, el host controlador va consultando a cada dispositivo en orden de su identificación preguntando si tiene algo que mandar, con lo cual se evitan las colisiones [12].

Además, si se quiere acceder a un endpoint, el dispositivo tiene asociado un parámetro llamado descriptor el cual guarda el estado (activo, inactivo) en uno o más descriptores de configuración. Dichos descriptores a su vez poseen descriptores de interfaz, los cuales poseen configuraciones por defecto y finalmente los endpoints como se aprecia en la figura:

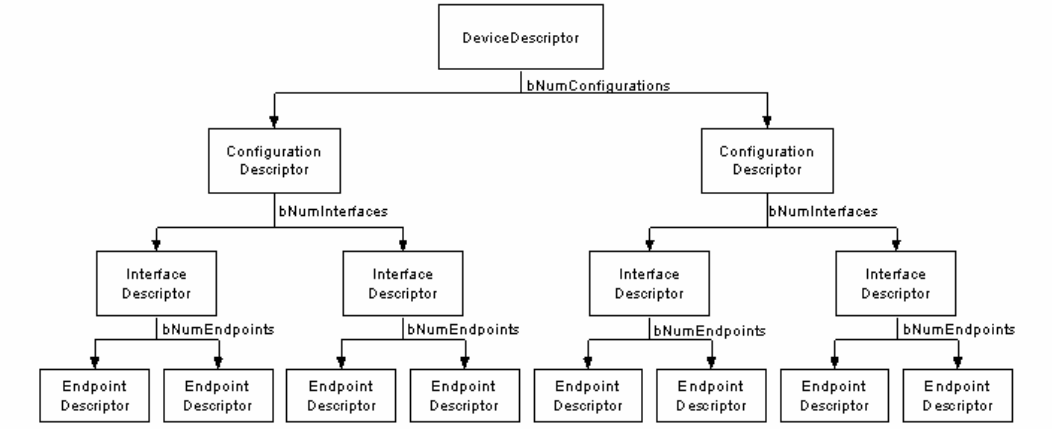


Figura 2.22*: Jerarquía de descriptores y endpoints.*

Existen cuatro tipos principales de transferencia en una conexión USB, que son los siguientes:

Transferencia de control

Utilizada para comandos cortos y simples, además es el tipo de transferencia utilizado por el pipe 0.

Transferencia isócrona

Consiste en dejar un ancho de banda fijo para la transferencia con posibilidad de pérdidas de datos. Generalmente es utilizada para aplicaciones de audio y video.

Transferencia HID

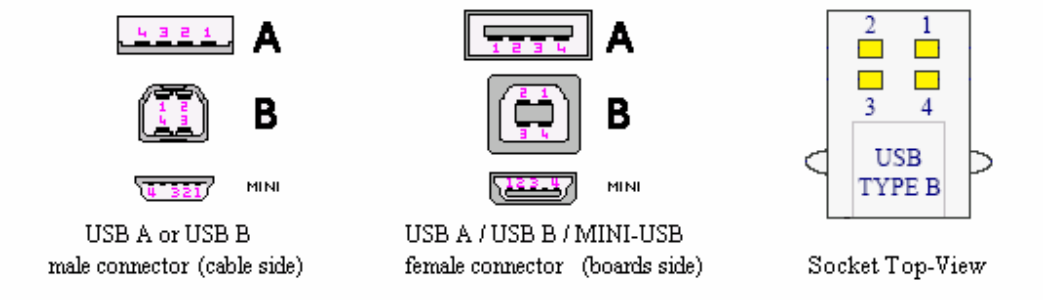
Utilizada para dispositivos de interacción humana (HID) como por ejemplo un mouse o teclado.

Transferencias Bulk o masivas

Se utiliza en transferencias grandes utilizando todo el ancho de banda disponible, pero sin garantías de velocidad. En caso de que se conectara otro dispositivo, la transferencia disminuirá ya que el ancho de banda no está garantizado

2.2.3 Pinout USB

En el conector USB se distinguen 4 pines, 2 de polarización y 2 de datos, como se muestra en la figura 2.23:



Pin	Name	Cable color	Description
1	VCC	Red	+5 VDC
2	D-	White	Data -
3	D+	Green	Data +
4	GND	Black	Ground

Figura 2.23: Pinout USB

La información transmitida viaja por los pines D+ y D- del conector USB utilizando señalización diferencial half-duplex minimizando el ruido electromagnético en tramos largos.

Se permite un largo máximo de 5 metros para no perder calidad en la señal y así evitar pérdidas en la información.

Hay conectores Standard y mini. Los primeros son los que normalmente se encuentran en los computadores y se clasifican en tipo A y B. Los tipo A (planos) se encuentran del lado del host controlador mientras que los B (cuadrados) están del lado del dispositivo. Para el caso de los conectores mini se respeta la misma clasificación A y B sólo que de menor tamaño.

2.2.4 Potencia

El voltaje suministrado por el puerto USB es de 5 volts regulados entre los pines 1 y 4, según se mostró en el pinout anterior. De haber una fuente externa, el voltaje no puede ser menor que 4.375 V ni mayor que 5.25 V. La corriente máxima total suministrada a los dispositivos por puerto conectado es de 500 mA.

Al momento de conectar un dispositivo, se reporta al host controlador cuanta potencia va a consumir. Así, el host controlador lleva una tabla con las necesidades de cada puerto y cuando eventualmente un dispositivo supera su límite generalmente se apaga, cortándole el suministro de corriente, no afectando al resto de los dispositivos [13].

El estándar exige que los dispositivos se conecten en un modo de bajo consumo (100 mA máximo) y luego le comuniquen al host controlador cuanta corriente precisan, para luego cambiar a un modo de alto consumo (si el host se lo permite). Los dispositivos que superen los límites de consumo deben utilizar su propia fuente de poder.

Los dispositivos que no cumplan con los requisitos de potencia y consuman más corriente de la negociada con el host puede dejar de funcionar sin previo aviso ,o en algunos casos, dejar todo el bus sin servicio.

2.2.5 Librería de funciones USB

Como se mencionó anteriormente, para utilizar las funciones USB, se utilizó una API que trabaja con la familia PIC18Fxx5x. Dichas funciones que se pueden utilizar son las siguientes:

MPUSBOPEN (INSTANCE, PVID_PID, PEP, DWDIR, DWRESERVED)

Devuelve el acceso al pipe del Endpoint con el VID_PID asignado. Su entrada es un número de dispositivo para abrir. Es importante tener en cuenta que el driver lo comparten distintos dispositivos. El número devuelto por el MPUSBGetDeviceCount tiene que ser igual o menor que el número de todos los dispositivos actualmente conectados y usando el driver genérico.

Al llamar la función tiene que haber un mecanismo que intente llamar MPUSBOpen() desde 0 hasta MAX_NUM_MPUSB_DEV. Se tiene que contar el número de llamadas exitosas. Cuando este número sea igual al número devuelto por MPUSBGetDeviceCount, hay que dejar de hacer las llamadas porque no puede haber más dispositivos con el mismo VID_PID.

pVID_PID es un string que contiene el PID&VID del dispositivo objetivo (identificador unico dentro del bus USB). El formato es “vid_xxxx&pid_yyyy”. Donde xxxx es el valor del VID y el yyyy el del PID, los dos en hexadecimal.

pEP es un string con el número del Endpoint que se va a abrir. El formato es "\\MCHP_EPz" o "\MCHP_EPz" dependiendo del lenguaje de programación. Donde z es el número del Endpoint en decimal.

MPUSBREAD (HANDLE, PDATA, DWLEN, PLENGTH, DWMILLISECONDS)

Sus parámetros son: handle, el cual identifica la pipe del Endpoint que se va a leer, el pipe unida tiene que crearse con el atributo de acceso MP_READ; pData es un puntero al buffer que recibe el dato leído de la pipe; dwLen: Especifica el número de bytes que hay que leer de la pipe; pLenght: Puntero al número de bytes leídos. MPUSBRead pone este valor a cero antes de cualquier lectura o de chequear un error; dwMilliseconds especifica el intervalo de time-out en milisegundos. La función vuelve si transcurre el intervalo aunque no se complete la operación.

MPUSBWRITE (HANDLE, PDATA, DWLEN, PLENGTH, DWMILLISECONDS)

Sus parámetros son: handle, la cual identifica la pipe del Endpoint que se va a escribir. La pipe unida tiene que crearse con el atributo de acceso MP_WRITE; pData es un puntero al buffer que contiene los datos que se van a escribir en la pipe; dwLen especifica el número de bytes que se van a escribir en la pipe; pLenght es un puntero al número de bytes que se escriben al llamar esta función. MPUSBWrite pone este valor a cero antes de cualquier lectura o de chequear un error; y finalmente dwMilliseconds la cual especifica el intervalo de time-out en milisegundos. La función vuelve si transcurre el intervalo aunque no se complete la operación.

MPUSBCLOSE (HANDLE)

Cierra una determinada unión. Su parámetro único es handle, el cual identifica la pipe del Endpoint que se va a cerrar.

2.2.6 Estructura del paquete USB

Básicamente los paquetes USB se clasifican en tres categorías: los paquetes de Token, los cuales son los que especifican la identificación, dirección y endpoint de una conexión, además de añadirle un CRC para evitar errores; también se encuentran los paquetes de datos, los cuales una vez definidos los parámetros de conexión envían el dato propiamente tal; y finalmente los paquetes de handshacking, los cuales establecen que la conexión se realizó correctamente, de acuerdo a lo mostrado en la figura 2.24:

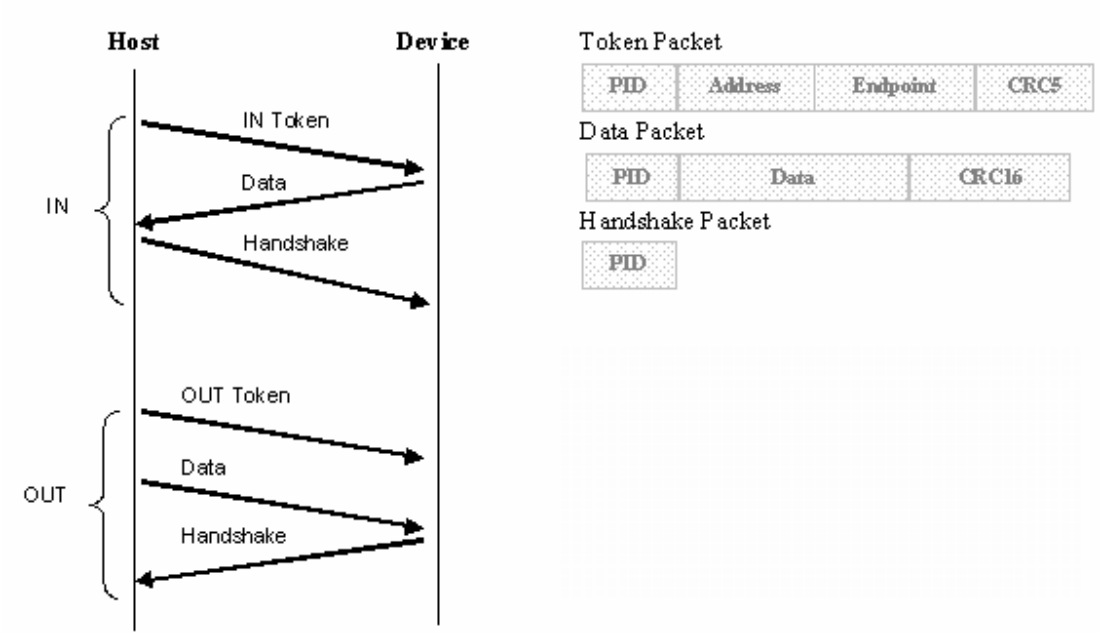


Figura 2.24: Tipos de paquetes USB

2.3 Conectividad de red

2.3.1 Introducción

Ethernet es la tecnología más popular de soporte físico de redes LAN en uso hoy. Esto es porque logra un buen balance entre velocidad, costo y facilidad de instalación. Estos puntos, combinados con una amplia aceptación en el mercado informático y la habilidad de soportar virtualmente la mayoría de los protocolos populares en la red, hacen de Ethernet una tecnología de red ideal para la mayoría de los usuarios de computadoras hoy en día.

El estándar Ethernet está definido por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) como el estándar IEEE 802.3. Este estándar define las reglas para configurar una Ethernet y especifica a su vez cómo interactúan entre si los elementos de una red Ethernet, adhiriendo a este estándar equipamiento y protocolos de red [19].

Por otro lado, TCP/IP corresponde a un conjunto de protocolos de comunicación para redes de computadores, en el cual Internet está basado. Se denomina TCP/IP ya que TCP (Transmisión control protocol) e IP (Internet protocol) son los más utilizados de éste conjunto. Puede utilizarse prácticamente en todo tipo de redes de computadores, debido a su flexibilidad de adaptación a distintos sistemas operativos.

2.3.2 Modelo OSI

Para entender mejor el traspaso de información en una red, se recurre al denominado modelo OSI, el cual explica el funcionamiento a base de siete capas independientes, las cuales tienen funciones determinadas en el proceso de transmisión de datos [23].

Los niveles superiores de este modelo son más cercanos a la interacción con el usuario, mientras que las capas inferiores, se relacionan más con la transmisión física de éstos [21].

El modelo OSI, si bien explica esta transferencia desde un punto de vista más teórico, en la práctica se usa una simplificación de éste en cinco capas denominado modelo TCP, el cual es más abstracto de entender, por lo que se deja a OSI como referencia para entenderlo.Un esquema del modelo OSI es el mostrado en la figura 2.25:

Figura 2.25: Modelo OSI.

Resumiendo las funciones de cada capa de dicho modelo, se tiene lo siguiente:

1º CAPA FISICA

Se encarga de las características eléctricas, mecánicas, funcionales y de procedimiento que se requieren para mover los bits de datos entre cada extremo del enlace de la comunicación.

2º CAPA DE ENLACE

Asegura con confiabilidad del medio de transmisión, ya que realiza la verificación de errores, retransmisión y control fuera del flujo que se utilizan en la capa de red.

3º CAPA DE RED

Proporciona los medios para establecer, mantener y concluir las conexiones conmutadas entre los sistemas del usuario final. Por lo tanto, la capa de red es la más baja, que se ocupa de la transmisión de extremo a extremo.

4º CAPA DE TRANSPORTE

Esta capa proporciona el control de extremo a extremo y el intercambio de información con el nivel que requiere el usuario. Representa el corazón de la jerarquía de los protocolos que permite realizar el transporte de los datos en forma segura.

5º CAPA DE SESION

Administra el diálogo entre las dos aplicaciones en cooperación mediante el suministro de los servicios que se necesitan para establecer la comunicación, flujo de datos y conclusión de la conexión.

6º CAPA DE PRESENTACIÓN

Permite a la capa de aplicación interpretar el significado de la información que se intercambia. Esta realiza las conversiones de formato mediante las cuales se logra la comunicación de dispositivos.

7º CAPA DE APLICACIÓN

Se entiende directamente con el usuario final, al proporcionarle el servicio de información distribuida para soportar las aplicaciones y administrar las comunicaciones por parte de la capa de presentación.

2.3.3 Protocolos

En esta sección, se mencionaran algunos aspectos de los principales protocolos estudiados para el diseño de la transmisión remota de la información adquirida por el osciloscopio

2.3.3.1 Protocolos de red

2.3.3.1.1 El protocolo IP

El protocolo de IP (Internet Protocol) es la base fundamental de Internet. Porta datagramas de la fuente al destino. El nivel de transporte parte el flujo de datos en datagramas. Durante su transmisión se puede partir un datagrama en fragmentos que se montan de nuevo en el destino [24].

Las principales características de este protocolo son: protocolo orientado a no conexión; fragmenta paquetes si es necesario; direccionamiento mediante direcciones lógicas IP de 32 bits; si un paquete no es recibido, este permanecerá en la red durante un tiempo finito; realiza el "mejor esfuerzo" para la distribución de paquetes; tamaño máximo del paquete de 65635 bytes; sólo se realiza verificación por suma al encabezado del paquete

IP proporciona un servicio de distribución de paquetes de datos orientado a no conexión de manera no fiable. La orientación a no conexión significa que los paquetes de información, que serán emitido a la red son tratados independientemente, pudiendo viajar por diferentes trayectorias para llegar a su destino. El término no fiable significa principalmente que no se garantiza la recepción del paquete.

La unidad de información intercambiada por IP es denominada datagrama. Tomando como analogía los marcos intercambiados por una red física los datagramas contienen un encabezado y un área de datos. IP no especifica el contenido del área de datos, ésta será utilizada arbitrariamente por el protocolo de transporte.

Para que en una red dos computadoras puedan comunicarse entre sí ellas deben estar identificadas con precisión Este identificador puede estar definido en niveles bajos (identificador físico) o en niveles altos (identificador lógico) dependiendo del protocolo utilizado. TCP/IP utiliza un identificador denominado dirección IP, cuya longitud es de 32 bytes. La dirección IP identifica tanto a la red a la que pertenece una computadora como a ella misma dentro de dicha red. En la figura 2.26 se observa la estructura de un paquete IP de forma grafica:

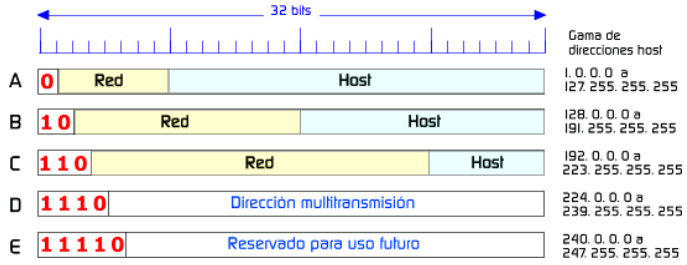


Figura 2.26: Encabezado IP.

Tomando tal cual está definida una dirección IP podría surgir la duda de cómo identificar qué parte de la dirección identifica a la red y qué parte al nodo en dicha red. Lo anterior se resuelve mediante la definición de las "Clases de Direcciones IP". Para clarificar lo anterior veamos que una red con dirección clase A queda precisamente definida con el primer octeto de la dirección, la clase B con los dos primeros y la C con los tres primeros octetos. Los octetos restantes definen los nodos en la red específica.

2.3.3.2 Protocolos de transporte

2.3.3.2.1 El protocolo TCP

El fin de TCP es proveer un flujo de bytes confiable de extremo a extremo sobre red no confiable. TCP puede adaptarse dinámicamente a las propiedades de la red y manejar fallas de muchas clases [25].

La entidad de transporte de TCP puede estar en un proceso en que parte un flujo de bytes en trozos y los manda como datagramas de IP. Para obtener servicio de TCP, el emisor y el receptor tienen que crear los puntos terminales de la conexión (sockets).

La dirección de un socket es la dirección de IP del host y un número de 16 bits que es local al host (el puerto). Se identifica una conexión con las direcciones de socket de cada extremo; se puede usar un socket para conexiones múltiples a la vez.

Cuando una aplicación manda datos a TCP, TCP puede mandarlos inmediatamente o almacenarlos. Una aplicación puede solicitar que TCP manda los datos inmediatamente a través del flag de PUSH.

TCP también apoya los datos urgentes. TCP manda datos con el flag URGENT inmediatamente. En el destino TCP interrumpe la aplicación (la manda una señal), que permite que la aplicación pueda encontrar los datos urgentes. En la figura 2.27 se observa la estructura de un paquete TCP de forma grafica:

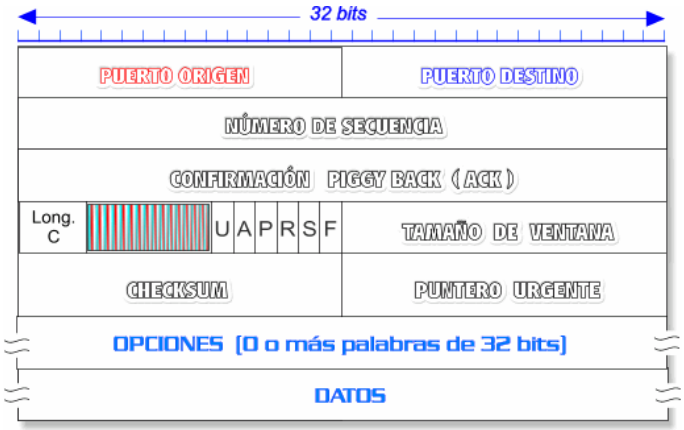


Figura 2.27: Encabezado TCP.

2.3.3.2.2 El protocolo UDP

El grupo de protocolos de Internet también maneja un protocolo de transporte sin conexiones, el UDP (User Data Protocol, protocolo de datos de usuario). El UDP ofrece a las aplicaciones un mecanismo para enviar datagramas IP en bruto encapsulados sin tener que establecer una conexión [26].

Muchas aplicaciones cliente-servidor que tienen una solicitud y una respuesta usan el UDP en lugar de tomarse la molestia de establecer una conexión con handshaking. El UDP se describe en el RFC 768. Un segmento UDP consiste en una cabecera de 8 bytes seguida de los datos. Los dos puertos sirven para lo mismo que en el TCP: para identificar los puntos terminales de las máquinas origen y destino. El campo de longitud UDP incluye la cabecera de 8 bytes y los datos. La suma de comprobación UDP incluye la misma pseudocabecera de formato, la cabecera UDP, y los datos, rellenados con una cantidad par de bytes de ser necesario. En la figura 2.28 se observa la estructura de un paquete UDP de forma grafica:

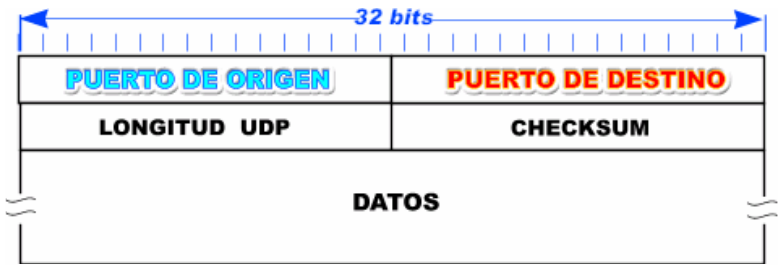


Figura 2.28: Encabezado UDP.

UDP no admite numeración de los datagramas, factor que, sumado a que tampoco utiliza señales de confirmación de entrega, hace que la garantía de que un paquete llegue a su destino sea mucho menor que si se usa TCP. Esto también origina que los datagramas pueden llegar duplicados y/o desordenados a su destino. Por estos motivos, el control de envío de datagramas, si existe, debe ser implementado por las aplicaciones que usan UDP como medio de transporte de datos, al igual que el reensamble de los mensajes entrantes.

Es por ello un protocolo del tipo best-effort (máximo esfuerzo), porque hace lo que puede para transmitir los datagramas hacia la aplicación, pero no puede garantizar que la aplicación los reciba.

Tampoco utiliza mecanismos de detección de errores. Cuando se detecta un error en un datagrama, en lugar de entregarlo a la aplicación destino, se descarta.

Cuando una aplicación envía datos a través de UDP, éstos llegan al otro extremo como una unidad. Por ejemplo, si una aplicación escribe 5 veces en el puerto UDP, la aplicación al otro extremo hará 5 lecturas del puerto UDP. Además, el tamaño de cada escritura será igual que el tamaño de las lecturas.

2.3.3.3 Standard de enlace

2.3.3.3.1 Ethernet

Los creadores de este protocolo de capa de enlace fueron Digital, Xerox e Intel, donde posteriormente la IEEE lo definió como Standard Ethernet 802.3 y hasta el día de hoy, es el método de conexión que más ha persistido [26].

El protocolo de red Ethernet fue diseñado originalmente por Digital, Intel y Xerox por lo cual, la especificación original se conoce como Ethernet DIX. Posteriormente, IEEE ha definido el estándar Ethernet 802.3. La forma de codificación difiere ligeramente en ambas definiciones.

Su velocidad de transmisión es de 10Mbit/s. Para el caso del protocolo Ethernet/IEEE 802.3, el acceso al medio se controla mediante un sistema definido como CSMA/CD (Carrier Sense Multiple Access with Collision Detection o Detección de Portadora con Acceso Múltiple y Detección de Colisiones).

Su principio de funcionamiento consiste en que una estación, para transmitir, debe detectar la presencia de una señal portadora y, si existe, comienza a transmitir. En el caso de que dos estaciones empiecen a transmitir al mismo tiempo, se produce una colisión y ambas deben repetir la transmisión, para lo cual esperan un tiempo aleatorio antes de repetir, evitando de este modo una nueva colisión, ya que el tiempo de espera escogido por ambas será distinto.

Hay una gran variedad de implementaciones de IEEE 802.3. Para distinguir entre ellas, se ha desarrollado una notación. Esta notación especifica tres características de la implementación: la tasa de transferencia de datos en Mb/s; el método de señalamiento utilizado; y la máxima longitud de segmento de cable en cientos de metros del tipo de medio.

Formato de la trama o frame Ethernet.

Los campos de trama Ethernet e IEEE 802.3 son los siguientes:

Preámbulo: la secuencia de unos y ceros alternados les indica a las estaciones receptoras que una trama es Ethernet o IEEE 802.3. La trama Ethernet incluye un byte adicional que es el equivalente al campo Inicio de trama (SOF) de la trama IEEE 802.3.

Inicio de trama (SOF): el byte delimitador de IEEE 802.3 finaliza con dos bits 1 consecutivos, que sirven para sincronizar las porciones de recepción de trama de todas las estaciones de la LAN. SOF se especifica explícitamente en Ethernet.

Direcciones destino y origen: vienen determinadas por las direcciones MAC únicas de cada tarjeta de red (6 bytes en hexadecimal). Los primeros 3 bytes de las direcciones son especificados por IEEE según el proveedor o fabricante. El proveedor de Ethernet o IEEE 802.3 especifica los últimos 3 bytes. La dirección origen siempre es una dirección de broadcast única (de nodo único). La dirección destino puede ser de broadcast única, de broadcast múltiple (grupo) o de broadcast (todos los nodos).

Tipo (Ethernet): el tipo especifica el protocolo de capa superior que recibe los datos una vez que se ha completado el procesamiento Ethernet.

Longitud (IEEE 802.3): la longitud indica la cantidad de bytes de datos que sigue este campo.

Datos (Ethernet): una vez que se ha completado el procesamiento de la capa física y de la capa de enlace, los datos contenidos en la trama se envían a un protocolo de capa superior, que se identifica en el campo tipo. Aunque la versión 2 de Ethernet no especifica ningún relleno, al contrario de lo que sucede con IEEE 802.3, Ethernet espera por lo menos 46 bytes de datos.

Datos (IEEE 802.3): una vez que se ha completado el procesamiento de la capa física y de la capa de enlace, los datos se envían a un protocolo de capa superior, que debe estar definido dentro de la porción de datos de la trama. Si los datos de la trama no son suficientes para llenar la trama hasta una cantidad mínima de 64 bytes, se insertan bytes de relleno para asegurar que por lo menos haya una trama de 64 bytes (tamaño mínimo de trama).

Secuencia de verificación de trama (FCS): esta secuencia contiene un valor de verificación CRC (Control de Redundancia Cíclica) de 4 bytes, creado por el dispositivo emisor y recalculado por el dispositivo receptor para verificar la existencia de tramas dañadas.

Cuando un paquete es recibido por el destinatario adecuado, les retira la cabecera de Ethernet y el checksum de verificación de la trama, comprueba que los datos corresponden a un mensaje IP y entonces lo pasa a dicho protocolo (capa de red-Internet) para que lo procese.

Hay que destacar que las direcciones utilizadas por Ethernet no tienen nada que ver con las direcciones de Internet. Las de Internet se le asignan a cada usuario, mientras que las de Ethernet vienen de incluidas de fábrica en la tarjeta de red (NIC).

El formato de trama Ethernet que se utiliza en redes TCP/IP es algo diferente del estándar IEEE 802.3:

Aquí el campo Longitud no existe (las tarjetas son capaces de detectar automáticamente la longitud de una trama) y en su lugar se emplea el campo Tipo.

Los medios físicos más utilizados son:

	10Base5	**10Base2**	**10Base-T**	**10Base-FL**
Cable	Coaxial grueso	Coaxial delgado	UTP Cat 3/5	Fibra 62,5/125
Pares	1	1	2/2	2
Full dúplex	No	No	Sí/Sí	Sí
Tipo Conector	N	BNC	RJ-45/RJ-45	ST
Topología	Bus	Bus	Estrella/Estrella	Estrella
Dist. Seg.	500, máx 2500 m	185, máx 925 m	100, máx 500 m	2 km.
Nº Nodos/seg.	100	30	1024/1024	1024

Tabla 2.1: Comparación de medios físicos.

En Ethernet, como en todas las redes locales, la transmisión es realizada de manera asincrónica. Por esto, se utiliza un sincronismo implícito en los datos mediante el uso de códigos que incorporan cierto nivel de redundancia. Ethernet usa el código Manchester, que utiliza dos voltajes e identifica el bit 0 como una transición alto-bajo y el 1 como una transición bajo-alto.

El código Manchester es poco eficiente, pero resulta sencillo y barato de implementar. Su mayor inconveniente resulta ser la elevada frecuencia de la señal, lo que complicó bastante las cosas cuando se adaptó Ethernet para UTP.

Los errores de CRC en una red Ethernet funcionando correctamente deberían ser casi nulos, salvo los originados por la conexión y desconexión de equipos. Debido a la elevada confiabilidad del medio físico, el protocolo MAC de Ethernet no realiza ningún tipo de verificación, ya que la probabilidad de que un frame no llegue a su destino es tan baja que esto sería perjudicial para el rendimiento de la red. Todo lo anterior se resume en la figura 2.29 correspondiente a un frame ethernet.

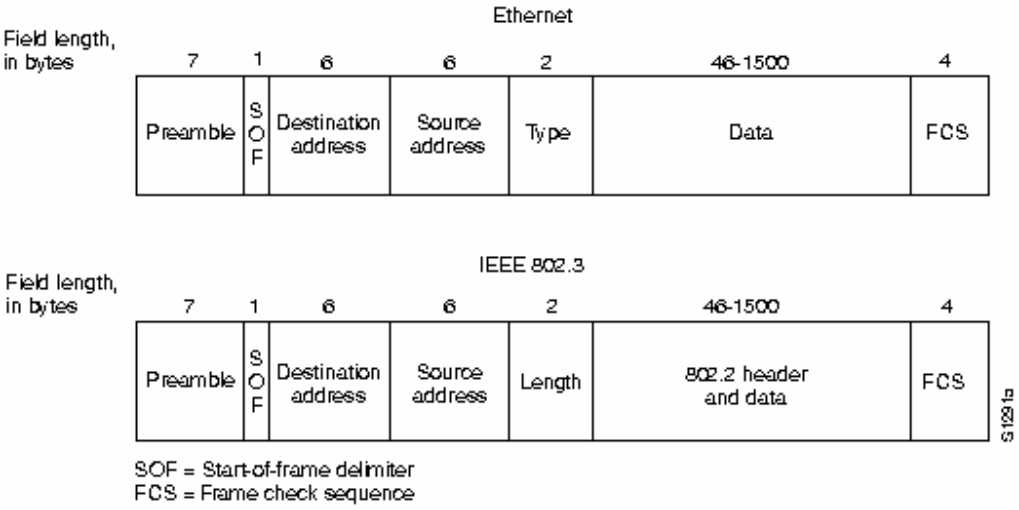


Figura 2.29: Encabezado Ethernet y 802.3

2.3.4 Capa física

La capa física de OSI, tal y como está normalizado por ISO, se define como la capa del modelo de referencia encargada de las funciones relacionadas con la transmisión de datos, desde el punto de vista de la gestión de las características eléctricas y mecánicas para una adecuada transferencia del canal. Como en este proyecto, se utilizará par trenzado, se mencionarán aspectos de este medio de transmisión

Para enviar una señal Ethernet de un punto a otro, es necesario que estén conectados a través de un sistema compuesto de componentes hardware dados por el estándar y que estén adaptados al sistema de cableado.

Sin embargo, la tendencia evolutiva de Ethernet a lo largo de los años ha ido dirigida a una implementación independiente del medio. La primera implementación de este tipo fue AUI (Attachment Unit Interface) dirigida a sistemas Ethernet de 10Mbps. Posteriormente se desarrollaron nuevos estándares como MII (Fast Intenet) o GII (Gigabit Ethernet).

AUI fue diseñada como una parte del sistema original Ethernet de 10 Mbps. Éste primer diseño fue implementado sobre cable coaxial (10BASE5). Sin embargo, posteriormente fueron ganando terreno nuevos medios de transmisión tales como la fibra óptica o el par trenzado, llegando éste último a ser el método más popular de implementación de dicha red (10BASE-T).

Al ser independiente del medio, esta implementación no requiere cambios en la electrónica de la red Ethernet, ya que es ajena al tipo de medio usado para la transmisión. Este sistema se compone de diferentes dispositivos que se describirán a continuación:

DTE: Son las siglas de "Data Terminal Equipment". Un DTE es un dispositivo direccionable que sirve como punto de origen o destino de los datos. Un PC o computador

equipado con Ethernet posee una interfaz DTE, que no es más que la electrónica necesaria para realizar el MAC (control de acceso al medio) requerido para enviar y recibir tramas a través del canal.

2.3.4.1 Codificación de señales ethernet

Las señales enviadas sobre los sistemas de 10 Mbps (incluyen sistemas de fibra óptica, coaxial y par trenzado) usan un esquema de codificación Manchester. Éste tipo de codificación combina señales de reloj y datos en símbolos, lo que proporciona una transición de reloj en medio de la duración de cada bit.

Además se divide cada período o duración de bit en dos partes, cambiando la polaridad en la segunda parte respecto de la primera. Por tanto podemos distinguir dos casos:

-Si la primera parte está en alto y la segunda en bajo tenemos un 0 binario.
-Si la primera parte está en bajo y la segunda en alto tenemos un 1 binario.

La transición en la mitad del período de bit proporciona la opción de sincronizar la estación receptora con los datos que le llegan. Este esquema de codificación requiere que, para una secuencia de 10 Mbps de unos y ceros, se requiera una frecuencia de 20 MHz sobre el cable.

Diferentes métodos de señalización de línea son usadas para enviar las señales codificadas en Manchester dependiendo del tipo de medio de transmisión usado. Por ejemplo las señales Ethernet sobre un coaxial se transmiten a través de dos corrientes sobre dicho cable: una corriente de OFFSET y una corriente de señalización que cambia en amplitud para representar 1 y 0. Los voltajes generados pertenecen al rango [0,-2]V, siendo en este caso para el caso coaxial. Estos valores cambiarán según el medio usado.

2.3.4.2 Sistemas con par trenzado (10BASE-T)

El sistema 10BASE-T fue el primer sistema conocido ampliamente en la implementación de Ethernet. La invención de 10BASE-T en la década de los noventa produjo una rápida masificacion de Ethernet en los computadores.

Los sistemas 10BASE-T fueron diseñados para soportar la transmisión a 10Mbps sobre el par de categoría 3 (usado para telefonía). Sin embargo, actualmente la gran mayoría del cableado por par trenzado usa categoría 5 debido a su mayor calidad de señal transmitida y a su alto rendimiento con 10 BASE-T.

2.3.4.3 Elementos del sistema 10BASE-T

Un sistema 10BASE-T está compuesto por: una interfaz Ethernet que incluye un transceiver 10BASE-T, un Cable AUI, Transceiver externo 10BASE-T AUI, Hub equipado con puertos 10BASE-T, Cable UTP de par trenzado, categoría 3 o superior

La interfaz 10BASE-T ya lleva incorporado un transceiver interno que permite realizar una conexión directa con par trenzado a través de un conector RJ45.

2.3.4.4 Polaridad de señal y polaridad reversible

Las señales correspondientes a los datos transmitidos y recibidos sobre el par trenzado 10BASE-T están polarizadas, con un cable portando la señal + y otro la -. Muchos sistemas 10BASE-T están provistos de la opción "polaridad reversible". Lo que significa que, a pesar de que las conexiones de los cables no estén bien realizadas (en lo que a un cambio de polaridad se refiere), es el mismo sistema el que se encarga de solucionar este error vía hardware, sin tener que modificar el cableado. El pinout de un conector RJ45 se muestra en la figura 2.30.:

Pin	Label
1	RD+
2	RD-
3	TD+
4	NC
5	NC
6	TD-
7	NC
8	NC

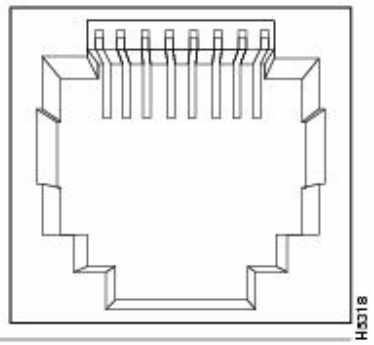


Figura 2.30: Pinout RJ45.

CAPITULO 3: IMPLEMENTACION DEL PROYECTO

3.1 Características de componentes

3.1.1 Conversión análogo-digital y microcontrol

Microchip lanzó el 2006 su microcontrolador PIC18F2550, el cual tiene la ventaja de ser compatible con el Standard USB 2.0 con posibilidades de low y full speed (1.5 Mb /s y 12 Mb/s respectivamente) [5].

Este chip soporta tanto transferencias de control, interrupción, isócronas y bulk. Además de tener una RAM interna de 1 Kb para almacenar datos provenientes de USB. Cabe mencionar también que la polarizacion entre Vdd y Vss es de 5 volts continuos.

También, como es común en la familia de microcontroladores PIC, incluye una serie de pines de propósito general y dos para la transmisión y recepción de datos seriales.

El pinout de este dispositivo se muestra en la figura 3.1:

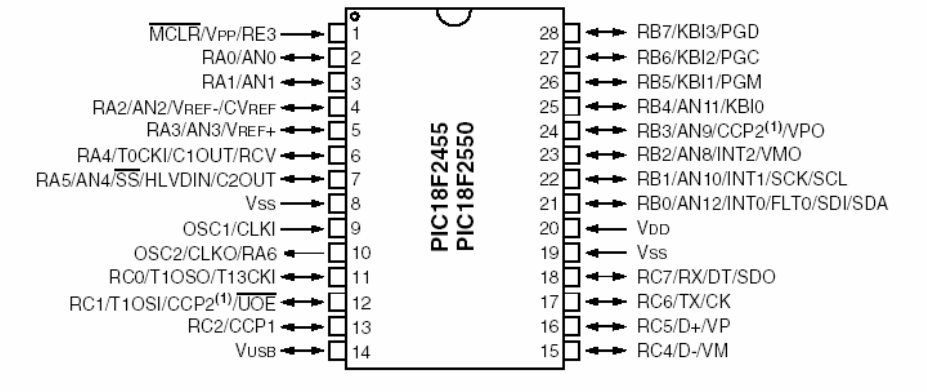


Figura 3.1: Pin-out del microcontrolador PIC18F2550.

Este microcontrolador soporta frecuencias de reloj de hasta 48 Mhz e incluye ocho puertos conversores análogos digital de 10 bits de resolución máxima. El modelo de este bloque se aprecia en la figura 3.2:

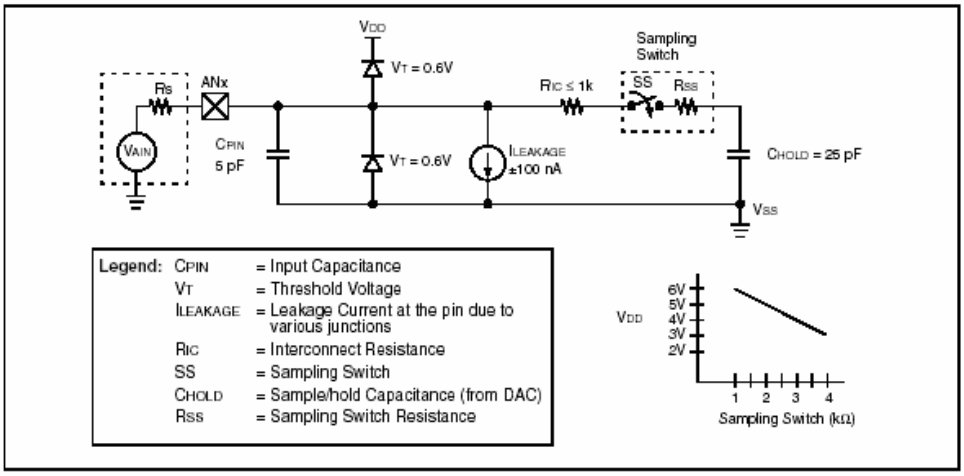


Figura 3.2: Modelo del conversor análogo-digital del PIC18F2550.

Como se aprecia en la figura 3.2, dentro del ADC, se incluyen condensadores de retención y diodos de protección, los cuales ayudan al microcontrolador a soportar eventuales transcientes cuando se ingresa una señal.

Viéndolo desde un punto de vista más general, este ADC contiene ocho canales para el muestreo y la cuantizacion de la información análoga, y la forma en que funciona este intercambio de canales, es mediante la multiplexión de los mismos, mediante un clock específico para el conversor, definido por software (firmware en este caso) como se muestra en la figura 3.3:

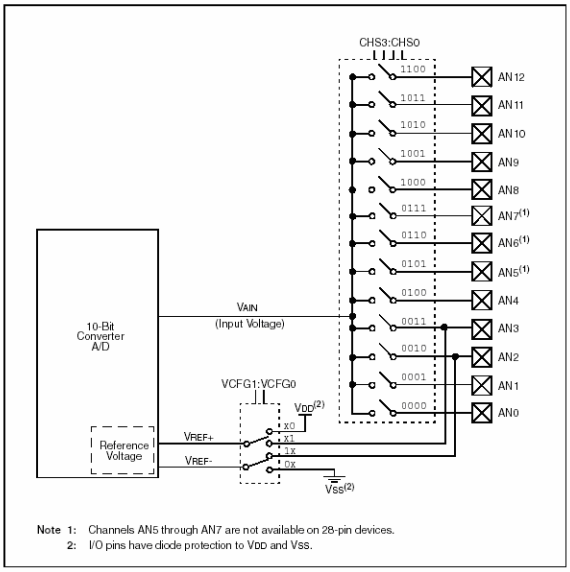


Figura 3.3: Multiplexion de canales análogo-digital del PIC18F2550.

En esta imagen se aprecia el hecho de que existe un único conversor ADC y para que soporte ocho canales, se multiplexan estos. Además, se tienen pines específicos para definir una referencia para la conversión. Por ejemplo, si se quiere medir voltajes alternos, conviene dejar la referencia en la mitad del rango máximo del ADC.

Cabe mencionar que el conversor tiene un rango específico de variables en las que puede operar bien, lo cual se indica en la siguiente tabla:

Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
A01	NR	Resolution	—	—	10	bit	ΔVREF ≥ 3.0V
A03	EIL	Integral Linearity Error	—	—	<±1	LSb	ΔVREF ≥ 3.0V
A04	EDL	Differential Linearity Error	—	—	<±1	LSb	ΔVREF ≥ 3.0V
A06	EOFF	Offset Error	—	—	<±1.5	LSb	ΔVREF ≥ 3.0V
A07	EGN	Gain Error	—	—	<±1	LSb	ΔVREF ≥ 3.0V
A10	—	Monotonicity	Guaranteed(1)			—	VSS ≤ VAIN ≤ VREF
A20	ΔVREF	Reference Voltage Range (VREFH – VREFL)	1.8 3	— —	— —	V V	VDD < 3.0V VDD ≥ 3.0V
A21	VREFH	Reference Voltage High	VSS	—	VREFH	V	
A22	VREFL	Reference Voltage Low	VSS – 0.3V	—	VDD – 3.0V	V	
A25	VAIN	Analog Input Voltage	VREFL	—	VREFH	V	
A30	ZAIN	Recommended Impedance of Analog Voltage Source	—	—	2.5	kΩ	
A50	IREF	VREF Input Current(2)	— —	— —	5 150	µA µA	During VAIN acquisition. During A/D conversion cycle.

Note 1: The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.
2: VREFH current is from RA3/AN3/VREF+ pin or VDD, whichever is selected as the VREFH source.
VREFL current is from RA2/AN2/VREF-/CVREF pin or VSS, whichever is selected as the VREFL source.

Tabla 3.1: Características del conversor análogo-digital del PIC18F2550.

De esta tabla se desprende que la conversión ADC involucra una serie de errores, los cuales fueron comentados en el capítulo 2. Entre estos están: el error de linealidad integral y diferencial, error de offset y error de Ganancia. Todos estos errores están definidos siempre y cuando se cumpla una diferencia entre los voltajes de referencia superior a 3 Volts. En cuanto a los requerimientos de la conversión, se tiene la tabla 3.2:

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
130	TAD	A/D Clock Period	PIC18FXXXX	0.7	25.0(1)	µs	TOSC based, VREF ≥ 3.0V
			PIC18LFXXXX	1.4	25.0(1)	µs	VDD = 2.0V, TOSC based, VREF full range
			PIC18FXXXX	TBD	1	µs	A/D RC mode
			PIC18LFXXXX	TBD	3	µs	VDD = 2.0V, A/D RC mode
131	TCNV	Conversion Time (not including acquisition time)(2)		11	12	TAD	
132	TACQ	Acquisition Time(3)		1.4 TBD	— —	µs µs	-40°C to +85°C 0°C ≤ to ≤ +85°C
135	TSWC	Switching Time from Convert → Sample		—	(Note 4)		
137	TDIS	Discharge Time		0.2	—	µs	

Legend: TBD = To Be Determined

Note 1: The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

2: ADRES registers may be read on the following TCY cycle.

3: The time for the holding capacitor to acquire the "New" input voltage when the voltage changes full scale after the conversion (VDD to VSS or VSS to VDD). The source impedance (*Rs*) on the input channels is 50Ω.

4: On the following cycle of the device clock.

Tabla 3.2: Requerimientos del conversor análogo-digital del PIC18F2550.

En cuanto a la tabla anterior, se puede mencionar que existen cinco parámetros que se desprenden del proceso de digitalizar un dato: reloj del conversor, el tiempo de conversión, el tiempo de adquisición, el tiempo de muestreo y finalmente el tiempo de descarga. Entonces se puede definir la siguiente expresión para la digitalización de datos:

$$T_{total} = T_{conversion} + T_{adquisicion} + T_{switching} + T_{descarga}$$

Además se tiene que para el PIC18F2550, el reloj del ADC si se usa un oscilador externo, el período se encontrará dentro del rango de 7 a 25 microsegundos. El tiempo de conversión, entre 11 y 12 veces este clock.

En cuanto al tiempo de adquisición, para temperaturas comprendidas entre -40 y +85 ºC, se mantendrá en 1.4 microsegundos, a su vez que el tiempo de descarga en 0.2 microsegundos.

También conviene considerar la función de transferencia el ADC, de la cual se puede analizar el error de cuantización que se tendrá.

En la figura 3.4, se observa la relación entre la salida digital y el LSB (byte menos significativo). Se observa en este caso, un error de cuantizacion de +-q/2

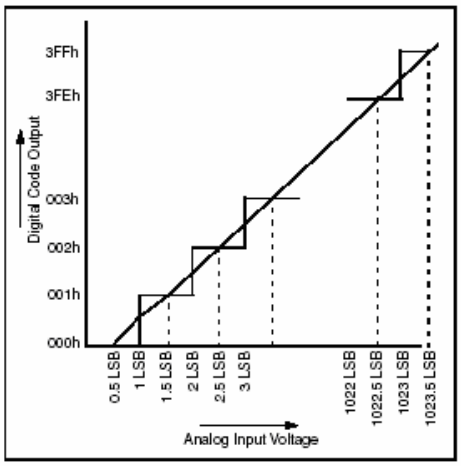


Figura 3.4: Requerimientos del conversor análogo-digital del PIC18F2550.

Entonces, si se tiene un rango de 0 a 5 volts de entrada y una resolución de 8 bits, el paso mínimo será de

$$\frac{5}{2^8} = 0{,}01953125 \text{ Volts}$$

Con lo cual, el error de cuantificación será entonces de

$$\frac{0{,}01953125}{2} = 0{,}009765625$$

Otro dato interesante de analizar es el diagrama de tiempo del ADC. En la figura 3.5 se muestran el reloj de referencia, los datos de entrada, el límite entre un dato actual y el siguiente y los tiempos de simple&hold respectivos.

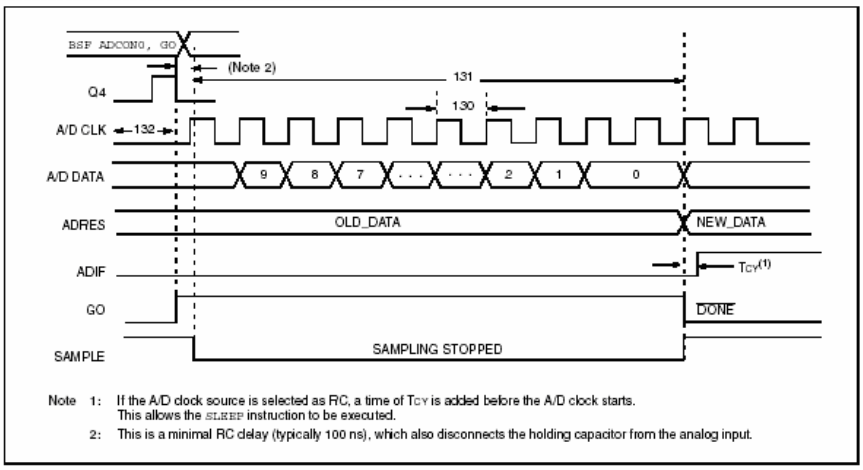


Figura 3.5: Diagrama de tiempo del conversor.

Por otro lado, cuando el dato ya está digitalizado, en este proyecto se envía a la memoria RAM del bloque USB para transportar la información hacia el computador. El diagrama de bloques para esta sección es el mostrado en la figura 3.6:

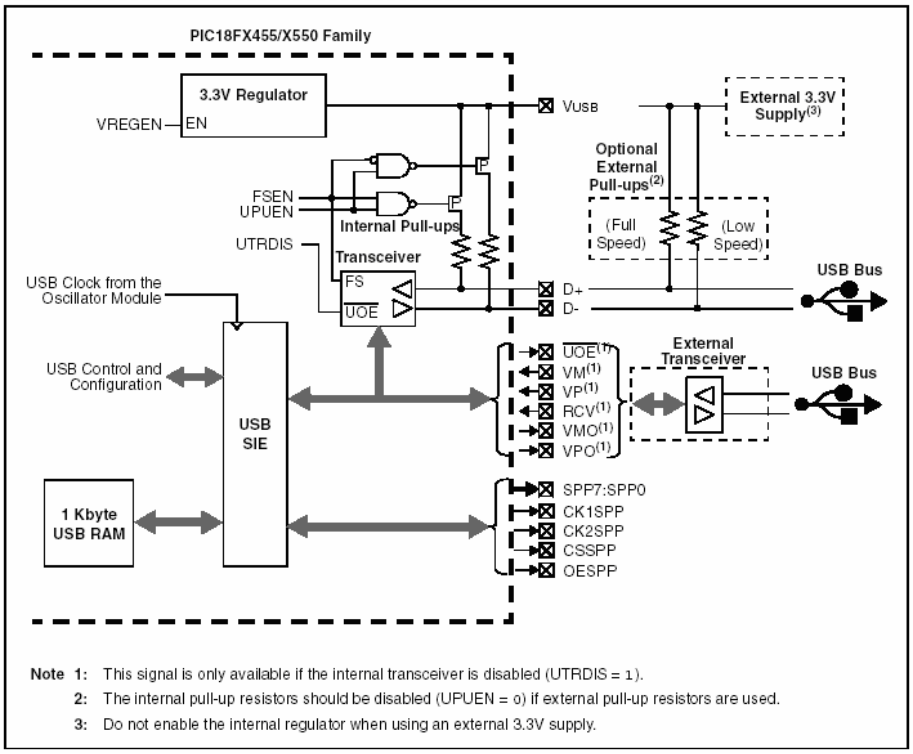


Figura 3.6: Diagrama de bloques USB.

Acá se observa que la memoria donde se guardan los datos es de 1 Kb. Una vez ahí, se pasa al bloque USB SIE (Serial Interface Engine), el cual es el encargado de la comunicación con el computador.

Para esta comunicación, se dispone de un transceiver, el cual define la capa física de comunicación USB, es decir, define la información en niveles de voltaje D+ y D-, los cuales asimila el host controlador USB del computador.

Además de enviar la información digitalizada, previamente, el SIE se encarga de inicializar la transacción, es decir, los procesos de init, task y enumeración del dispositivo, donde se define el VID y el PID, además de establecer los pipes y endpoints por donde viajarán los datos.

En cuanto a la forma en que viaja la señal, se tiene el diagrama de tiempo entre las señales diferenciales D+ y D- como se muestra en la figura 3.7:

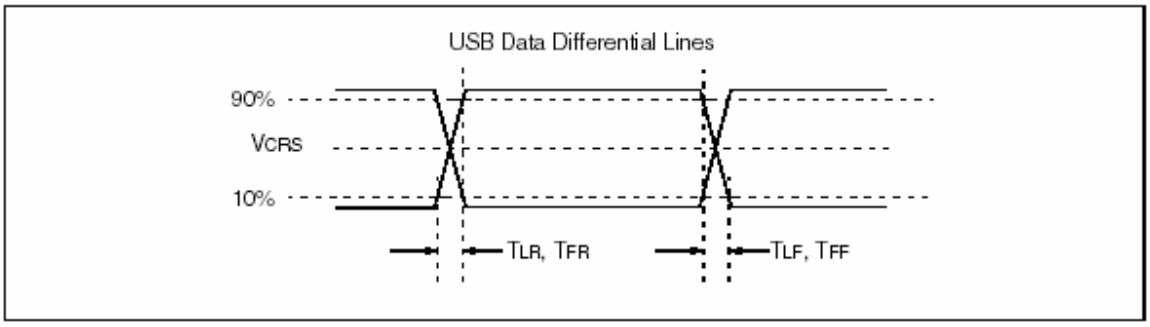


Figura 3.7: Líneas de datos diferenciales USB.

Se definen también los parámetros TxR (Transition rise time), TxF (Transition fall time), donde x puede ser L (low speed) o F (full speed).

Para Low Speed, se tienen rangos tanto de subida como bajada de mínimo 75 y máximo 300 ns. En el caso de Full Speed, serán un mínimo de 4 y un máximo de 20 ns para que defina un dato de otro.

3.1.2 Amplificación de ganancia programable

Para el bloque de amplificación programada, es necesario construir un sistema en que se tenga un amplificador, con un conjunto de resistencias que definan niveles discretos de ganancia, con el fin de poder mostrar un rango más amplio de voltajes y no limitarse al rango dinámico del conversor, según se aprecia en la figura 3.8.

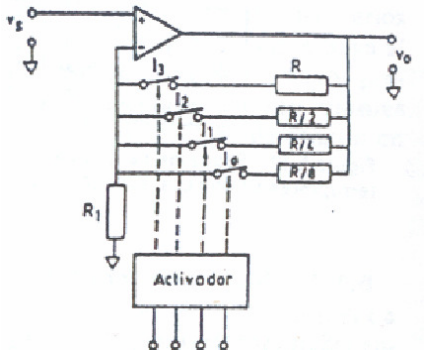


Figura 3.8: Amplificador de ganancia programable.

El sistema de control se puede resumir en el diagrama de bloques mostrado en la figura 3.9:

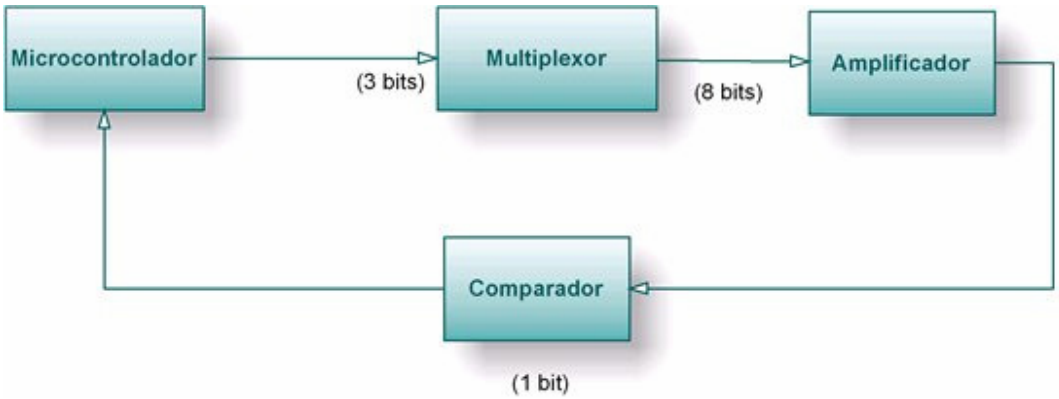


Figura 3.9: Opción 2 para el diseño del amplificador de ganancia programable.

En este caso, se utilizarían como multiplexión, el CMOS4051, y para la amplificación y comparación, amplificadores operacionales. Ahora se analizarán las características de estos elementos.

Para el CMOS4051 (figura 3.10), se tiene un amplio rango de voltajes de operación (3 a 15 volts), además de una resistencia interna del interruptor aproximada de 80 Ohms (varía con la polarización y temperatura) y una alta impedancia cuando está apagado (se asume circuito abierto). También tiene baja disipación de potencia (1 microWatt) [9].

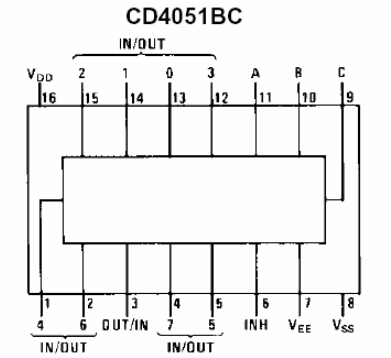


Figura 3.10: Pin-out del multiplexor CMOS4051.

En cuanto a la resistencia interna del multiplexor, en la hoja de datos, se puede observar la tabla 3.3:

Symbol	Parameter	Conditions		−55°C		+25°			125°C		Units
				Min	Max	Min	Typ	Max	Min	Max	
R_{ON}	"ON" Resistance (Peak for $V_{EE} \le V_{IS} \le V_{DD}$)	R_L = 10 kΩ (any channel selected)	V_{DD} = 2.5V, V_{EE} = −2.5V or V_{DD} = 5V, V_{EE} = 0V		800		270	1050		1300	Ω
			V_{DD} = 5V, V_{EE} = −5V or V_{DD} = 10V, V_{EE} = 0V		310		120	400		550	Ω
			V_{DD} = 7.5V, V_{EE} = −7.5V or V_{DD} = 15V, V_{EE} = 0V		200		80	240		320	Ω

Tabla 3.3: Rangos de operación del multiplexor CMOS4051.

Acá se desprende que la resistencia interna será de 270 Ohms para el funcionamiento entre +-2.5v, 120 Ohms para +-5 volts y 80 Ohms para +-7.5 volts, todos en el caso de una temperatura de 25º C.

En cuanto al amplificador operacional, conviene utilizar un LM324 (Figura 3.11). Este integrado contiene cuatro amplificadores operacionales en su interior, además de que su polarización no es simétrica, lo que equivale a decir que perfectamente se puede alimentar con tensiones de 0 y 5 volts, eliminando así la necesidad de una fuente simétrica de alimentación [8].

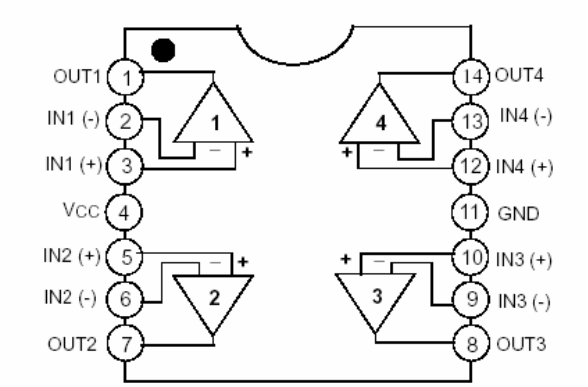


Figura 3.11: Pin Out LM324.

Pero como se desean medir tensiones alternas, éstas tendrán componentes tanto positivas como negativas. Entonces, será necesario incluir una etapa sumadora utilizando uno de los bloques interiores del LM324. Por lo tanto, se sumaran 2.5 volts la señal que varia en el rango +-2.5 volts para hacerla coincidir en el rango de 0 a 5 volts.

En cuanto a la linealidad de la relación entrada-salida, según lo investigado en el datasheet, se asegura una linealidad en el rango de temperaturas entre 0 y 70 grados Celsius.

3.1.3 Controlador Ethernet

El Microchip ENC28J60 (Figura 3.12) es un controlador Ethernet 10Base-T (10Mbps en cables), cercano al estándar IEEE 802.3.

Esta constituido por un modulo PHY (nivel físico), un modulo MAC (subnivel MAC), una memoria RAM de 8kbyte para almacenar los paquetes en recepción y en transmisión, una serie de registros de configuración y un modulo para la comunicación de serie SPI.

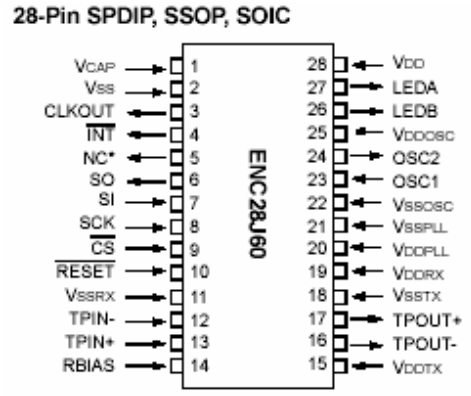


Figura 3.12: Pin Out ENC28J60

El controlador está diseñado para trabajar con 25Mhz, por lo tanto es necesario un cuarzo de esta frecuencia entre los polos OSC1 y OSC2, más dos condensadores cerámicos conectados a tierra. Además la polarizacion de este chip es a 3.3 volts. Su diagrama de bloques se muestra en la figura 3.13:

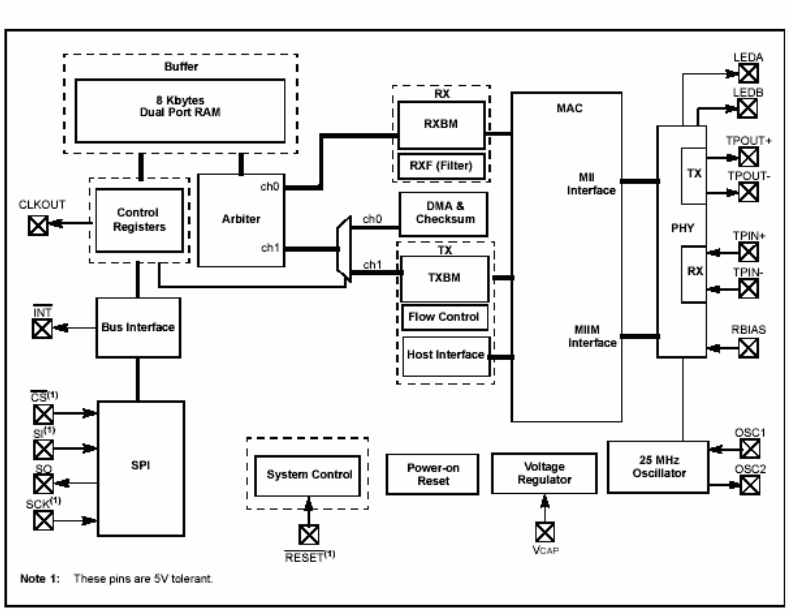


Figura 3.13: Diagrama de bloques ENC28J60.

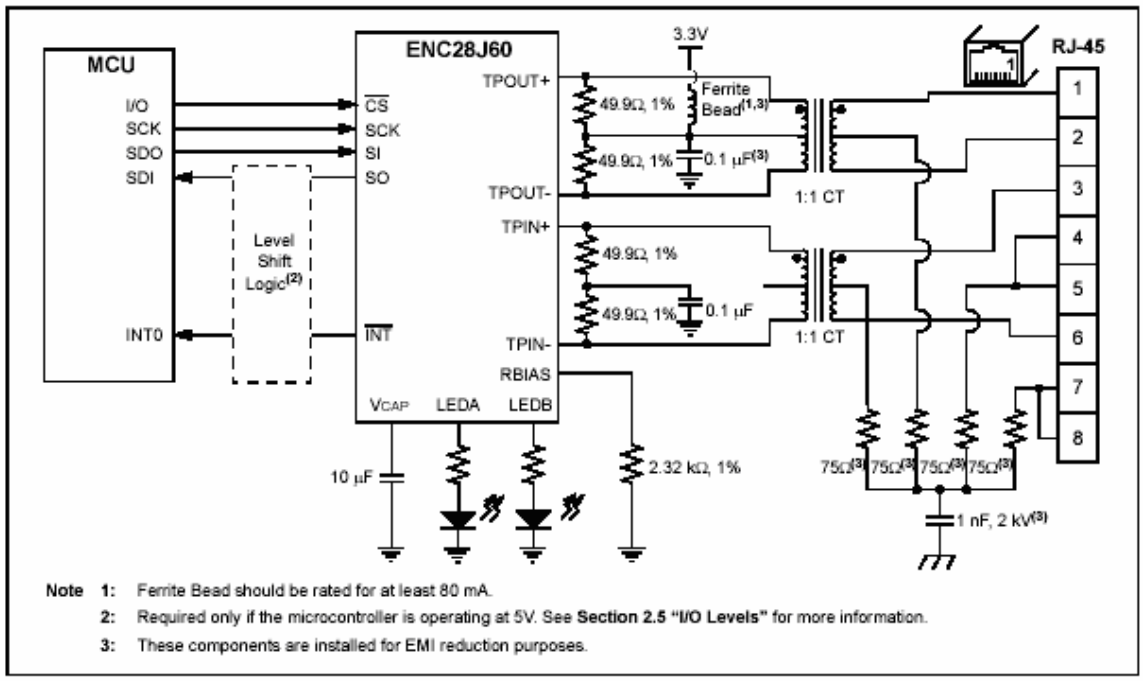


Figura 3.14 Conexiones ENC28J60.

Como se aprecia en la figura 3.14, existen tres bloques principales: La unidad de microcontrol (MCU), el controlador Ethernet y el transductor de niveles de tensión y conector RJ45.

La comunicación entre el microcontrolador y el ENC28J60 sucede por los pines SDI y SDO, mientras que un clock sincroniza ambos, y opcionalmente existe un Chip Select, el cual puede conectarse al microcontrolador. En este caso, lo lógico es conectarlo directamente a un nivel bajo, puesto a que su entrada está negada.

Existen cuatro salidas diferenciales: Tpout (positivo y negativo), y Tpin (positivo y negativo), las cuales respectivamente se conectan a transformadores y finalmente al conector.

3.2 ETAPA DE ENTRADA Y PROTECCIONES

Como se dijo en la revisión bibliográfica, en un divisor de tensión se tendrá un voltaje de entrada y otro de salida para poder ajustar la señal a los requerimientos del conversor ADC.

La tensión de salida Vs (Voltaje entre A y B) en función del voltaje de entrada y las resistencias R1 y R2 se puede calcular entonces como:

$$V_S = \frac{V_E \cdot R_2}{(R_1 + R_2)}$$

Para no poner en riesgo el circuito es conveniente fijar un voltaje máximo permisible a la entrada y éste llevarlo a 5 volts mediante un divisor de tensión. Por ejemplo, si se quiere medir 220 volts en el osciloscopio, conviene diseñar un divisor que soporte una tensión máxima de 350 volts como se muestra en la figura 3.15.

$$350v \Rightarrow 5v = \frac{350 \cdot R_2}{R_1 + R_2} \Rightarrow R1 = \frac{350 \cdot 10k\Omega}{5} - 10k\Omega = 690k\Omega$$

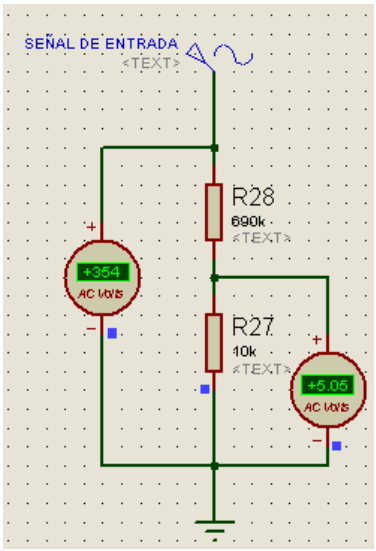


Figura 3.15: Divisor de tensión.

La figura 3.16 muestra una señal de amplitud 350 Volts ya atenuada, además de resaltar el nivel de 5 Volts y 0 Volts. Por lo tanto el resultado es una señal que oscila entre +2.5 y -2.5 Volts

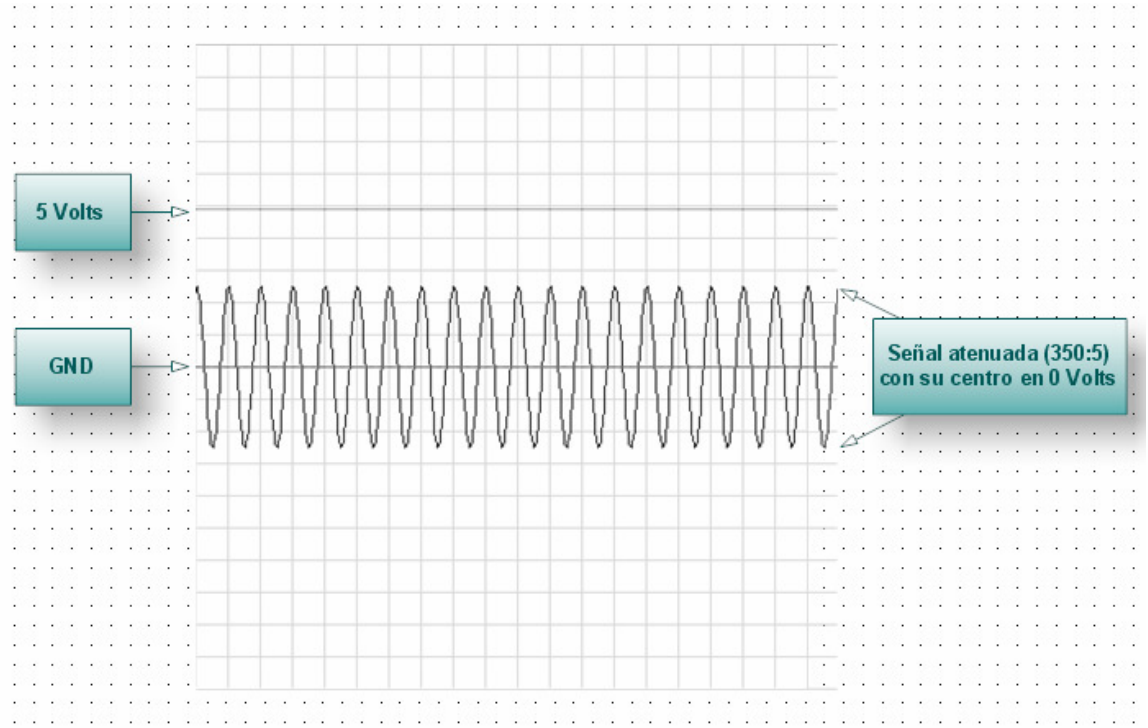


Figura 3.16: Señal atenuada.

Con este divisor, si se quieren medir 220 volts a la salida quedará el siguiente voltaje:

$$Vs = \frac{220 \cdot 10k\Omega}{690k\Omega + 10k\Omega} = \frac{220 \cdot 10k\Omega}{700k\Omega} = 3.14v$$

Por lo tanto, el ADC en ese rango de voltajes no sufrirá daños. Cabe mencionar que esta señal está referenciada a 0 volts, lo que significa que las amplitudes máxima y mínima serán de +2.5 y -2.5 volts respectivamente con una entrada de 350 volts máximo permisible.

La entrada del conversor permite un rango de 0 a 5 volts, es decir, no acepta voltajes negativos. Para solucionar este problema, se puede recurrir a un amplificador operacional en modo de sumador.

De acuerdo a lo visto en el capítulo 2 y con la información llevada hasta el momento, entonces la solución es agregar una componente continua de 2.5 volts para cambiar la referencia y en vez de estar en el rango de -2.5 y 2.5 volts, cambiar a un rango de 0 a 5 volts. Con esto, una señal de 0 volts entraría al ADC como 2.5 volts. En la figura 3.17 se muestra el bloque sumador y en la figura 3.18 el desplazamiento resultante de la señal ya atenuada..

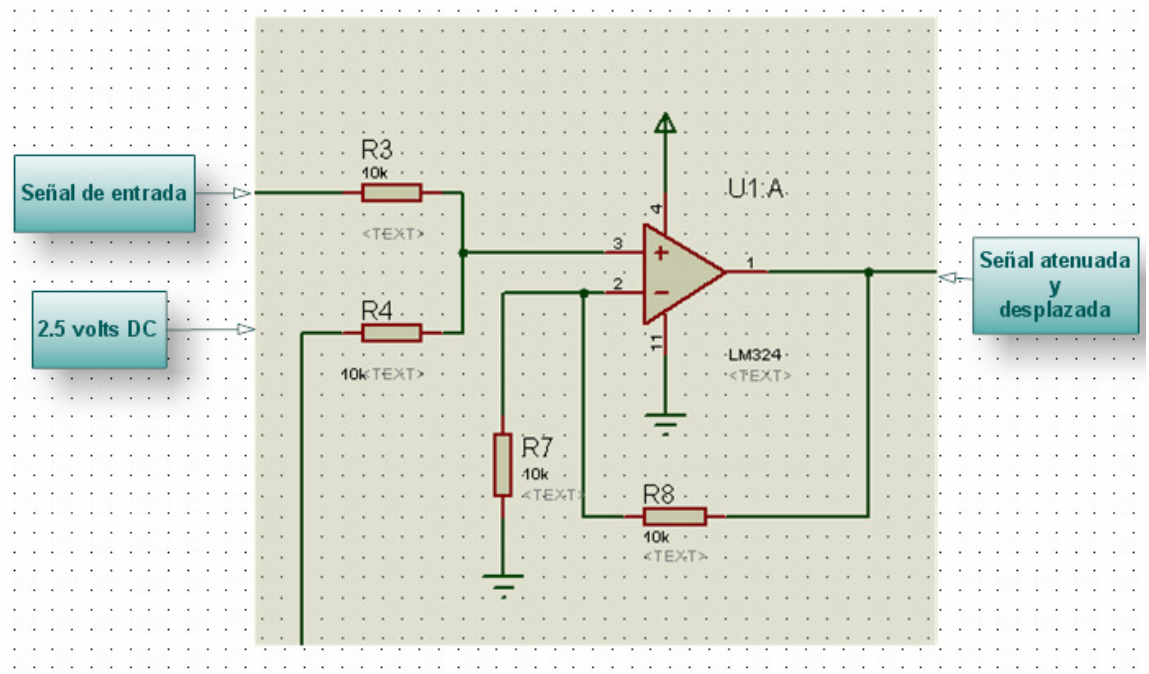


Figura 3.17: Sumador de componente continua.

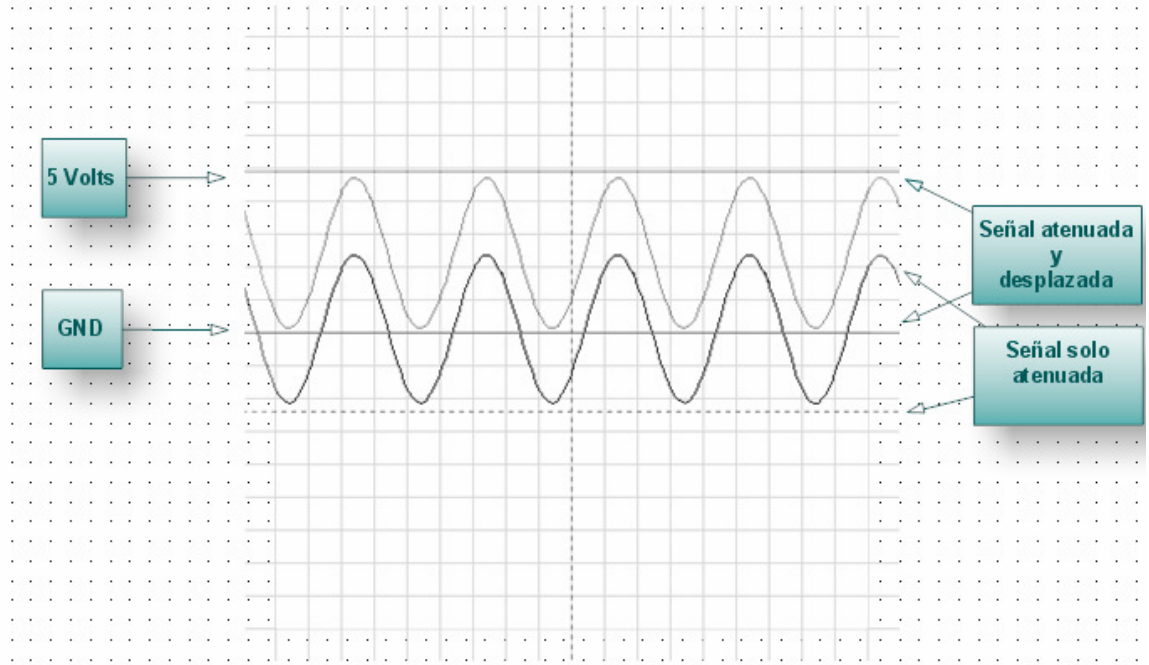


Figura 3.18: Atenuación y desplazamiento.

Con esta configuración a la entrada, si se introducen voltajes muchos menores, el sistema pierde calidad debido al error de cuantización. El mínimo voltaje posible a muestrear en estas condiciones se calcula de la siguiente manera:

$$V\min = \frac{350}{256} = 1{,}3671875 \text{ Volts}$$

$$e = \frac{V\min}{2} = 0{,}68359375 \text{ Volts}$$

El valor de 'e' corresponde al margen de valores en torno a una cantidad discreta al variar el bit menos significativo o LSB. Como el error es considerable si se quiere medir también voltajes pequeños, conviene añadir una etapa amplificadora de ganancia programable para auto ajustar la señal.

Primeramente, se puede ver un modelo de amplificador no inversor cuya configuración se muestra en la figura 3.19:

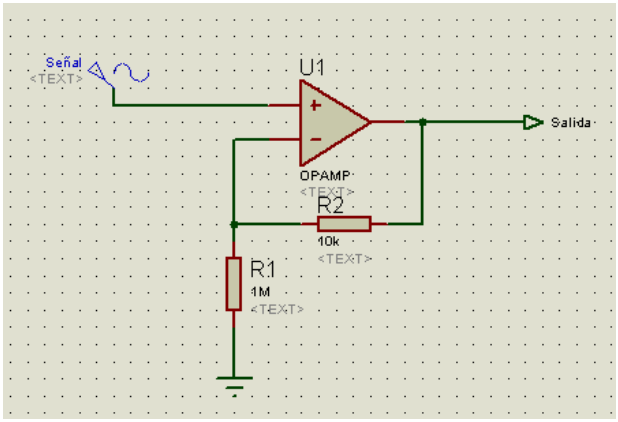


Figura 3.19: Amplificador no inversor.

Donde se cumple que:

$$Vout = Vin \cdot \left(\frac{R1 + R2}{R1} \right)$$

Se puede definir la ganancia como:

$$G = \frac{Vout}{Vin} = \frac{R1 + R2}{R1}$$

Ahora se puede ver también un modelo de este tipo de amplificadores, pero con un selector de ganancias variables accionado mecánicamente y que actúa directamente sobre R2 y cuya referencia es 2.5 volts para mantener la señal desplazada de GND y así no procesar voltajes negativos como se observa en la figura 3.20:

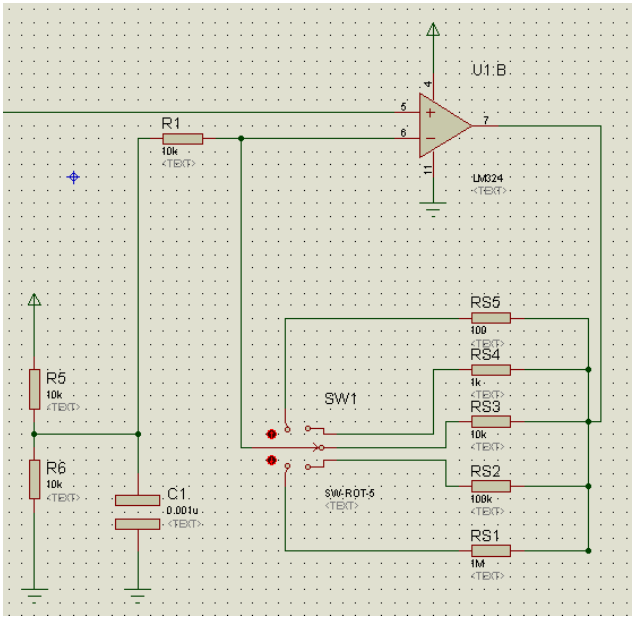


Figura 3.20: Amplificador no inversor con selector de ganancia manual.

El objetivo de esta sección es diseñar un sistema que proteja al conversor análogo-digital, además que para señales pequeñas respecto al máximo, no haya pérdida considerable de precisión debido al aumento del error de cuantizacion para estos valores de entrada, mediante la utilización de ganancias programables y retroalimentación.

El siguiente paso es reemplazar el selector mecánico por uno accionado digitalmente (figura 3.21) mediante un arreglo de resistencias controladas como se explicará a continuación.

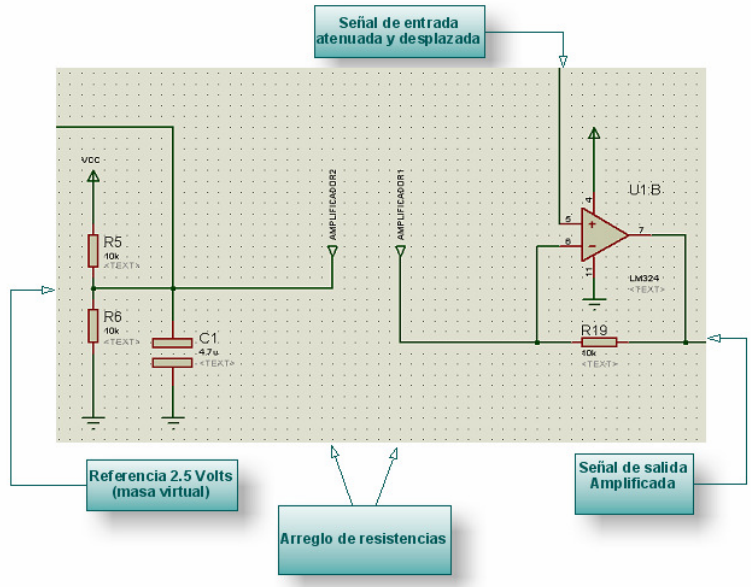


Figura 3.21: Diseño de amplificador de ganancia programable.

Se han etiquetado los marcadores Amplificador 1 y 2 con el fin de dejar ordenado el circuito, y es en este punto donde se introduce el arreglo de resistencias, operado mediante un multiplexor CMOS 4051.

Como se mencionó en el ítem de selección de componentes, éste integrado tiene tres entradas de control, una entrada común y ocho salidas que se alternan según la información que haya en el puerto de control.

Con esta información, se puede diseñar un potenciómetro digital de ocho estados discretos, los cuales afectarán la ganancia del amplificador operacional según se ve en la figura 3.22.

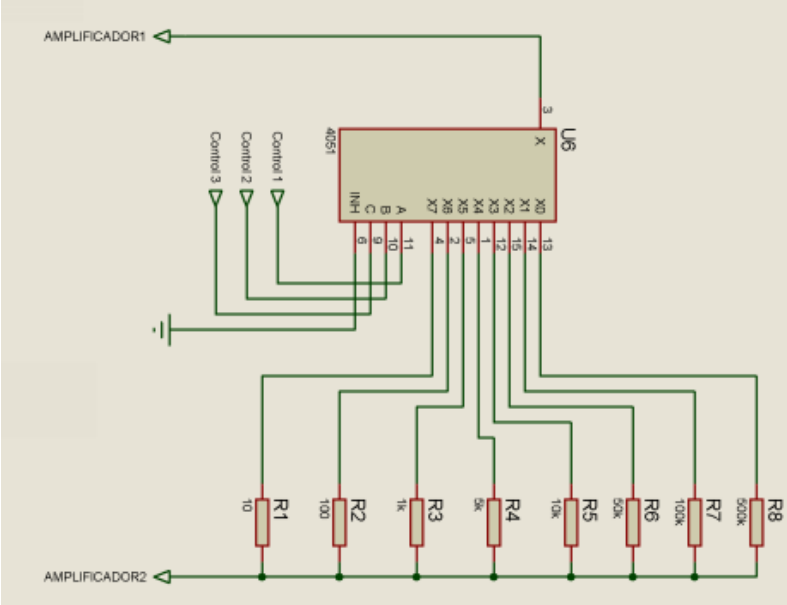


Figura 3.22: Arreglo de resistencias seleccionadas mediante multiplexión.

La ganancia del sistema se verá afectada por la resistencia de encendido del switch del multiplexor. Esta resistencia estará en serie con la resistencia externa que definirá la ganancia del amplificador. Considerando este efecto, se obtiene un nuevo cálculo para este parámetro:

$$G' = \frac{Vout}{Vin} = \frac{R1 + Rin + R2}{R1 + Rin}$$

Como se variará la resistencia de retroalimentación R2, entonces se puede despejar en función de R1, Rin y G':

$$R2 = R1 \cdot (G' - 1) - Rin \cdot (G' - 1)$$

Considerando que la señal se atenuó en un divisor de tensión con relación 350:5, es decir con una atenuación de 70 veces en la entrada:

$$Vin' = \frac{Vin}{70}$$

Entonces, para definir los valores de ganancia hay que considerar esta atenuación y compensarla. Con esta información se puede deducir la siguiente ecuación para el cálculo de los niveles discretos de ganancia en función de niveles máximos de voltaje de entrada:

$$G = \frac{350}{V_{escala}}$$

Por lo tanto, es posible calcular los valores para las resistencias que irán conectadas al multiplexor en función de Rin, R2 y un voltaje máximo para determinada escala. Estos dos últimos parámetros son variables libres.

Teniendo esto en cuenta, conviene definir la siguiente escala para los voltajes máximos en cada escala y, por lo tanto, es posible calcular valores para sus respectivas ganancias y resistencias como se muestra en la tabla 3.4.

Nivel	Voltaje máximo [V]	Resistencia R1 [Ohm]	Ganancia G'	Mínimo voltaje detectado [V]	Error de cuantización
1	2,5	152900	140	0,009765625	0,004882813
2	5	75900	70	0,01953125	0,009765625
3	25	14300	14	0,09765625	0,048828125
4	50	6600	7	0,1953125	0,09765625
5	100	2750	3,5	0,390625	0,1953125
6	200	825	1,75	0,78125	0,390625
7	250	440	1,4	0,9765625	0,48828125
8	350	0	1	1,3671875	0,68359375

Resolución [bits]	8
R1 [Ohms]	1000
Rin[Ohms]	100

Tabla 3.4: Valores de resistencias.

Como se observa, estos valores son factibles de crear mediante combinaciones serie-paralelo de resistencias. Para aumentar la precisión del instrumento, se usarán resistencias de 1% de tolerancia.

En resumen, la etapa de entrada y protecciones puede ser modelada por el siguiente diagrama de flujo (figura 3.23):

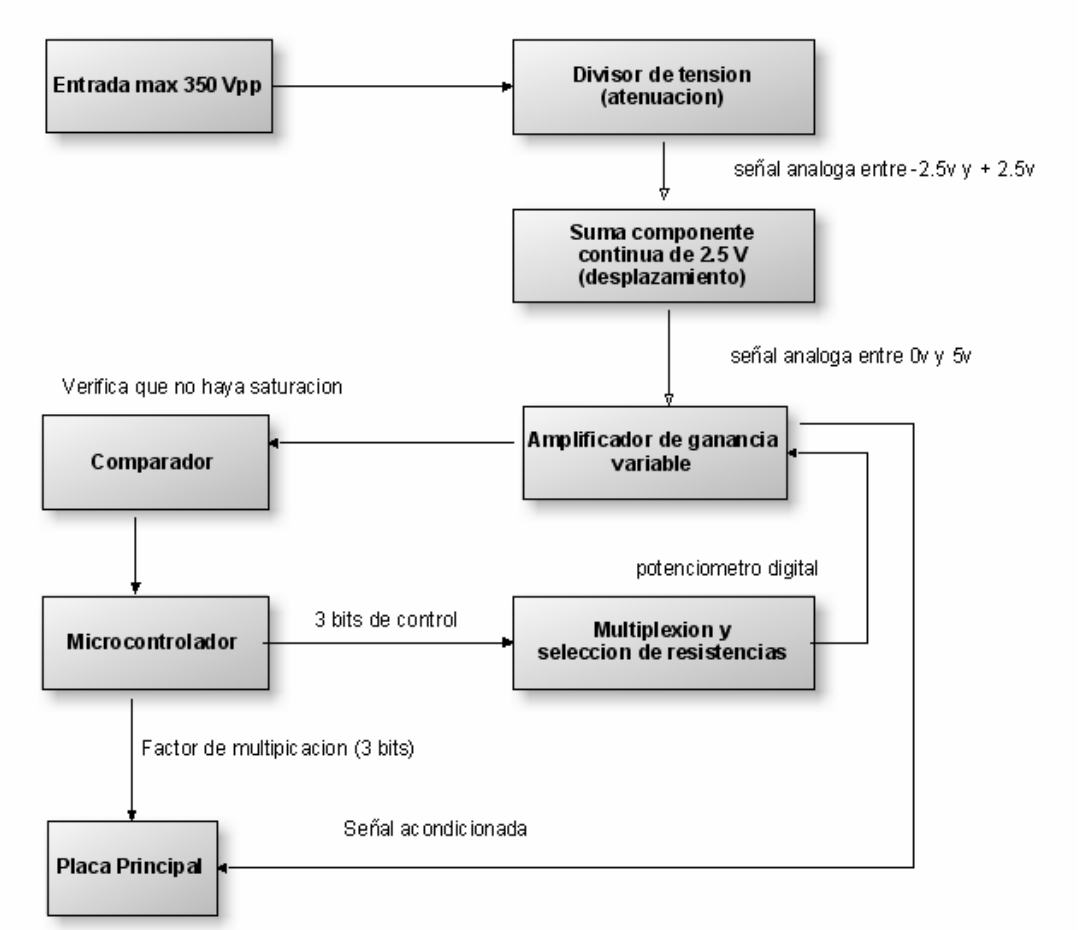


Figura 3.23: Diagrama de flujo para el acondicionamiento de una señal

Para el resto de los valores, se puede ver la gráfica entre la señal atenuada, desplazada y amplificada en la figura 3.24:

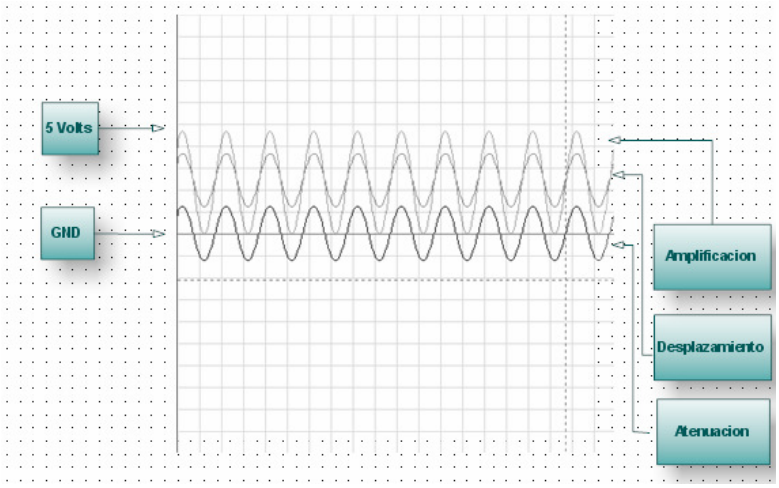


Figura 3.24: Etapas del acondicionamiento.

Como se observa, la señal una vez desplazada, pasa por el amplificador y éste calcula la ganancia óptima para el voltaje de entrada.

En cuanto a la lógica de control, se busca una amplificación máxima para las señales más pequeñas y que no exista saturación en el amplificador.

Un diseño a considerar, sería crear un lazo de realimentación en que la señal del amplificador ingresara a un puerto ADC libre y viera si está saturada o no la señal. En este caso el diagrama de flujo sería el mostrado en la figura 3.25:

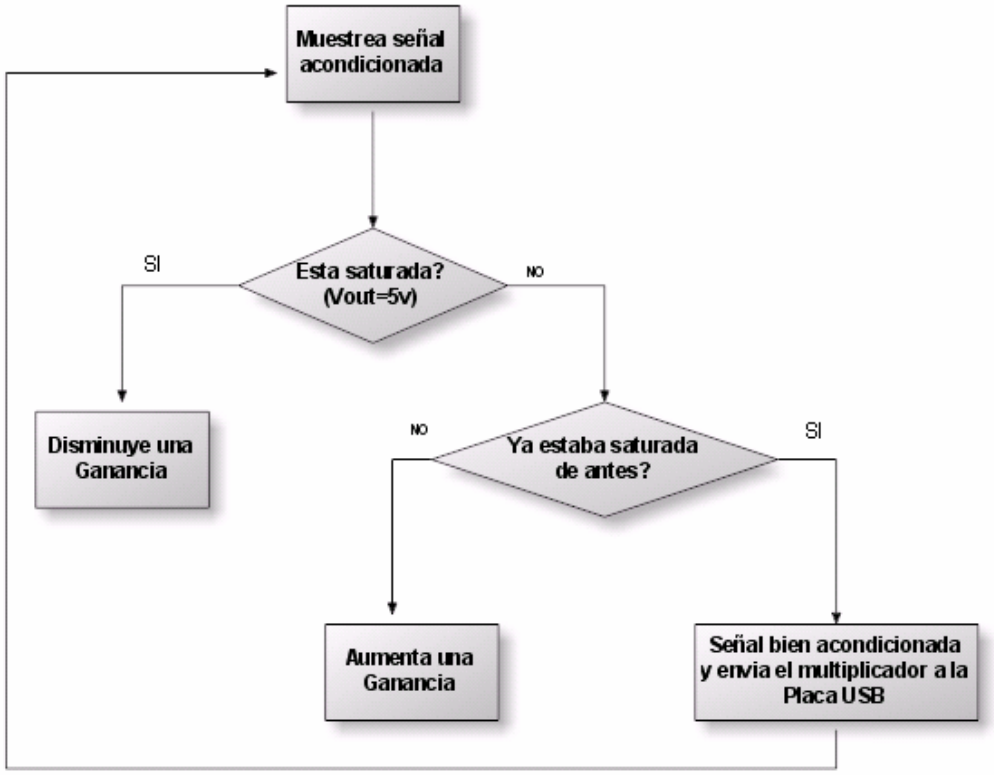


Figura 3.25: Diagrama de flujo para la amplificación de la señal acondicionada (opción 1).

Esto trae el inconveniente de que se ocupan ciclos de reloj en la interrupción ADC (se verá esto en la sección Firmware), por lo que lo ideal es utilizar un comparador que indique cuando el opamp está saturado y active una entrada GPIO del microcontrolador para indicarle que baje una ganancia y, cuando esté bien acondicionada la señal, se envía ésta al ADC y el multiplicador al microcontrolador. Con este nuevo esquema, el diagrama de flujo queda representado por la figura 3.26:

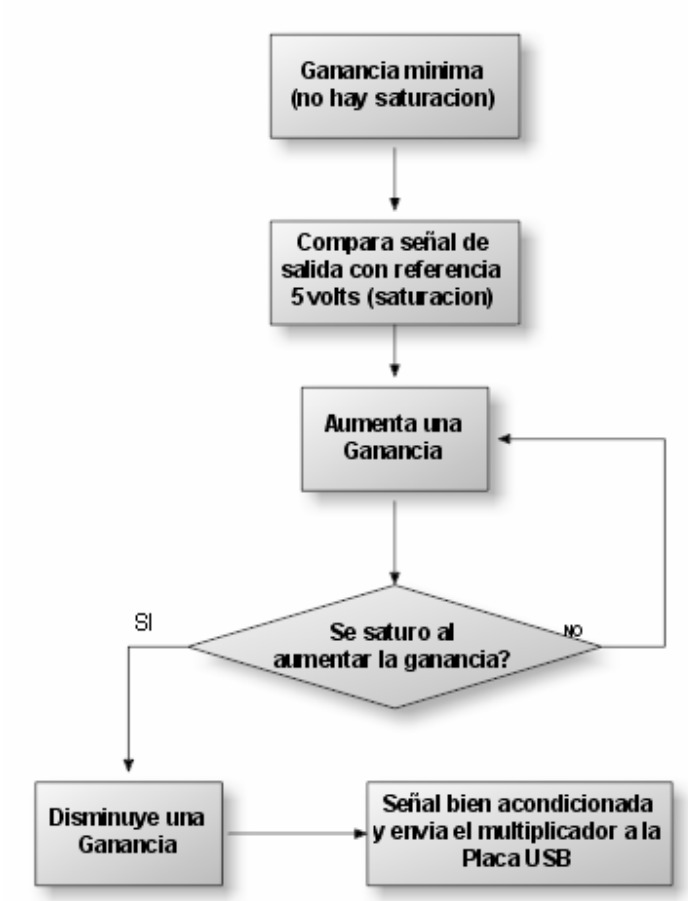


Figura 3.26: Diagrama de flujo para la amplificación de la señal acondicionada (opción 2).

El comparador también está hecho con un LM324. Éste recibe la señal que irá al ADC del microcontrolador y la compara con una referencia de 5 volts (condición de saturación). Comienza desde el punto de mínima ganancia, aumentando hasta que llega al nivel de saturación. Posteriormente, disminuye una ganancia y envía tanto la señal como el factor de multiplicación.

Cabe mencionar que se requieren 4 bits de datos para realizar este algoritmo: uno para la comparación (entrada al microcontrolador) y tres de salida de este hacia el multiplexor. El firmware con el procesamiento de la ganancia es parte de la sección anexos.

3.3 PRIMERA FASE: USB

3.3.1 HARDWARE

3.3.1.1 Conversión ADC, Microcontrol y Transmisión USB

Para la conversión y transmisión de datos, se diseñó la placa principal de este proyecto. Como se mencionó en la selección de componentes, el PIC18F2550 tiene tantos puertos de propósito general, conversores ADC, transmisión por RS232 y Transmisión USB. El diagrama de bloques indicando entradas y salidas de la placa se muestra en la figura 3.27:

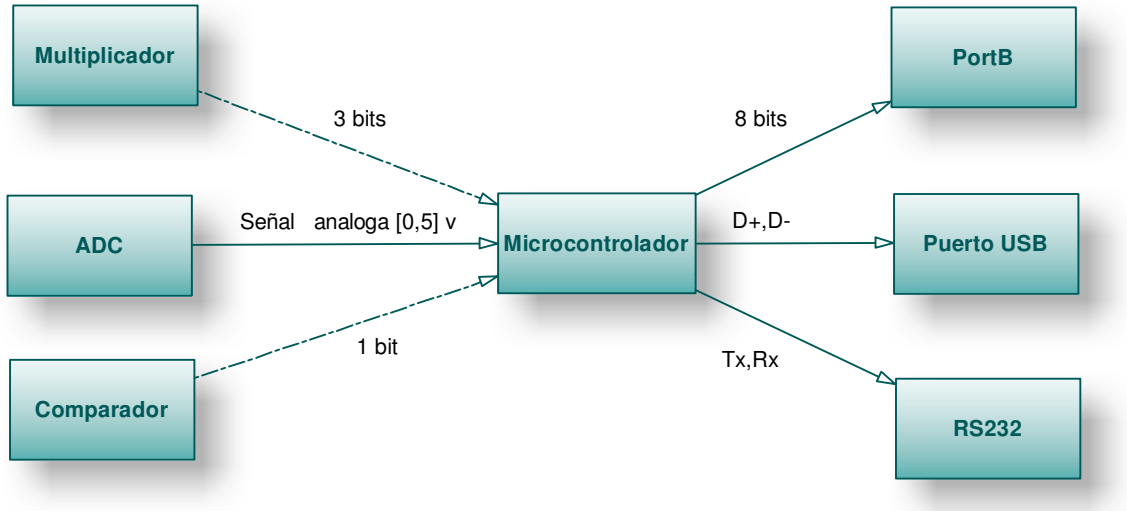


Figura 3.27: Bloques de entrada y salida al microcontrolador.

Como se ve en la imagen, la entrada principal es la señal a muestrear. Ésta es, a su vez, la salida de la etapa de acondicionamiento en que la señal no supera el rango entre 0 y 5 volts.

Además, otra entrada es la salida del comparador. Ésta se activa cuando observa que la señal de entrada está saturada con tal de que arregle la ganancia, para que la señal quede bien acondicionada.

Relacionado a esto, también está la entrada del multiplicador, el cual controla el multiplexor indicando qué resistencia determinará la ganancia del amplificador. Como el multiplexor es de 8 bits, entonces serán necesarios 3 bits para este bus de control.

En cuanto a las salidas, la principal es la transmisión USB. Como se mencionó anteriormente, funciona con señales diferenciales D+ y D-, las cuales son generadas directamente por el PIC. También, se agrega una fase de comunicación serial RS232 para monitorear el sistema en caso de un eventual debuggeo.

Por otro lado, se aprovecha también el puerto portB, el cual funciona de forma paralela y tiene la finalidad de comunicarse con el sistema de transmisión de datos basado en ethernet,

el cual se mencionará posteriormente. Básicamente, el esquemático de la placa principal es mostrado en la figura 3.28:

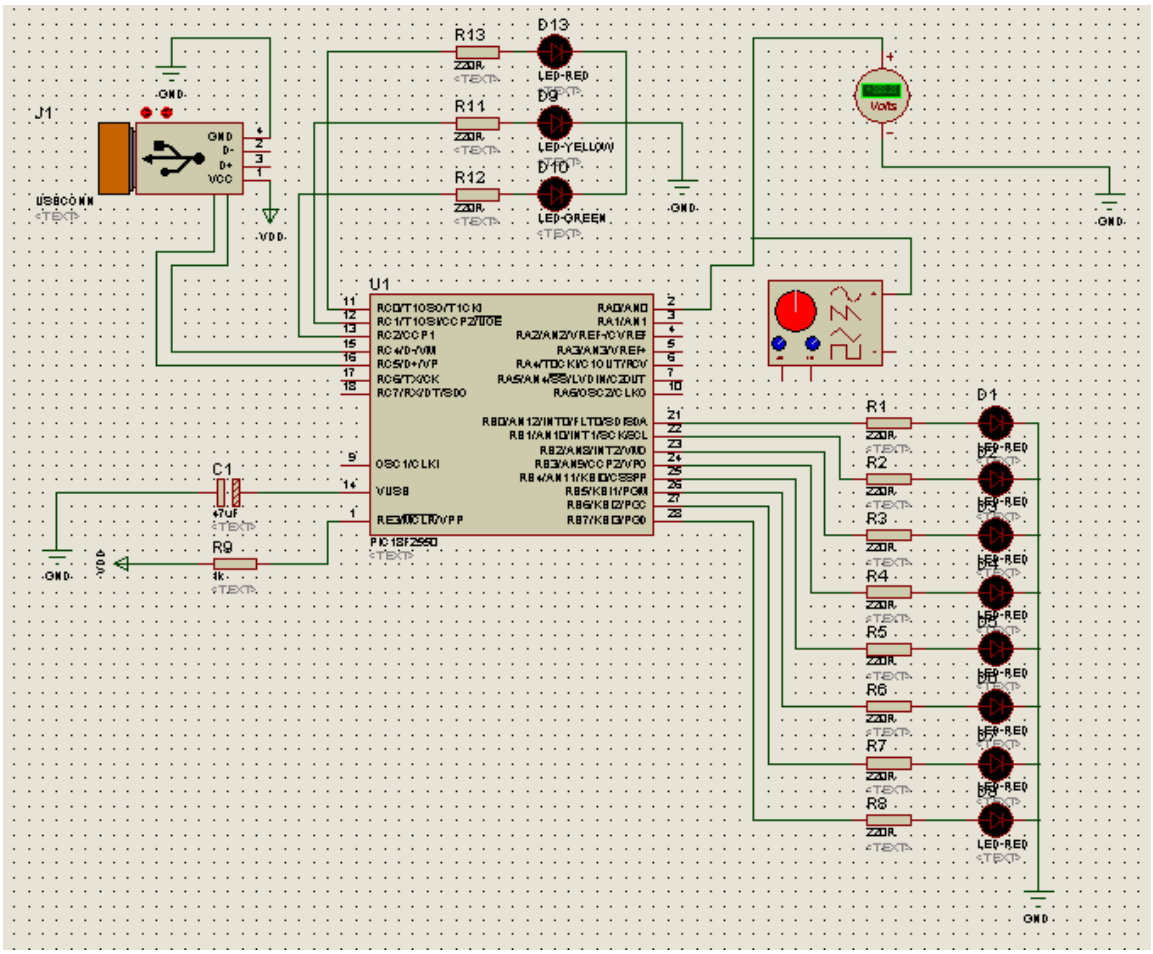


Figura 3.28: Prototipo implementado (simulación).

En cuanto al conexionado, para el núcleo del circuito se tiene acoplado un cristal de 12 MHz entre pines 9 y 10 y dos condensadores de 15 pF. Internamente, mediante un preescalamiento definido en los fuses del firmware, se aumentará el clock de funcionamiento a 48 MHz.

La polarización se hace con 5 volts regulados (ya sea externos o sacados directamente del puerto USB) entre los pines 9, 18 (GND) y 20 (Vdd).

La definición del clock más la alimentación en la imagen anterior están implícitos dentro de la pastilla por el software usado, sin embargo, en la sección posterior, se verá el mismo esquemático, pero hecho en Eagle donde se podrán apreciar estos bloques.

Los pines correspondientes al puerto USB 2.0 son el 15 (D+) y 16 (D-). Además de tener un condensador de 100 nF entre la polarizacion del USB tal de evitar eventuales peaks que puedan dañar el puerto.

También está la opción de alimentarse externamente. Esto es utilizando un regulador de tensión 7805 y condensadores que entreguen un potencial continuo y sin alteraciones.

Para la comunicación serial, se dispone de un transductor MAX232, el cual convierte señales TTL (usadas por el sistema en general) a RSS232, las cuales pueden ser +12 ó -12 y que son interpretadas por el puerto serie del computador.

3.3.1.2 PCB del proyecto

A partir del diseño mostrado anteriormente, se creó el PCB de la placa principal (Figuras 3.29 y 3.30), utilizando el software EAGLE, y Circuit CAM más BoardMaster para los Gerber que irán a la fresa para darle un aspecto más profesional a la placa.

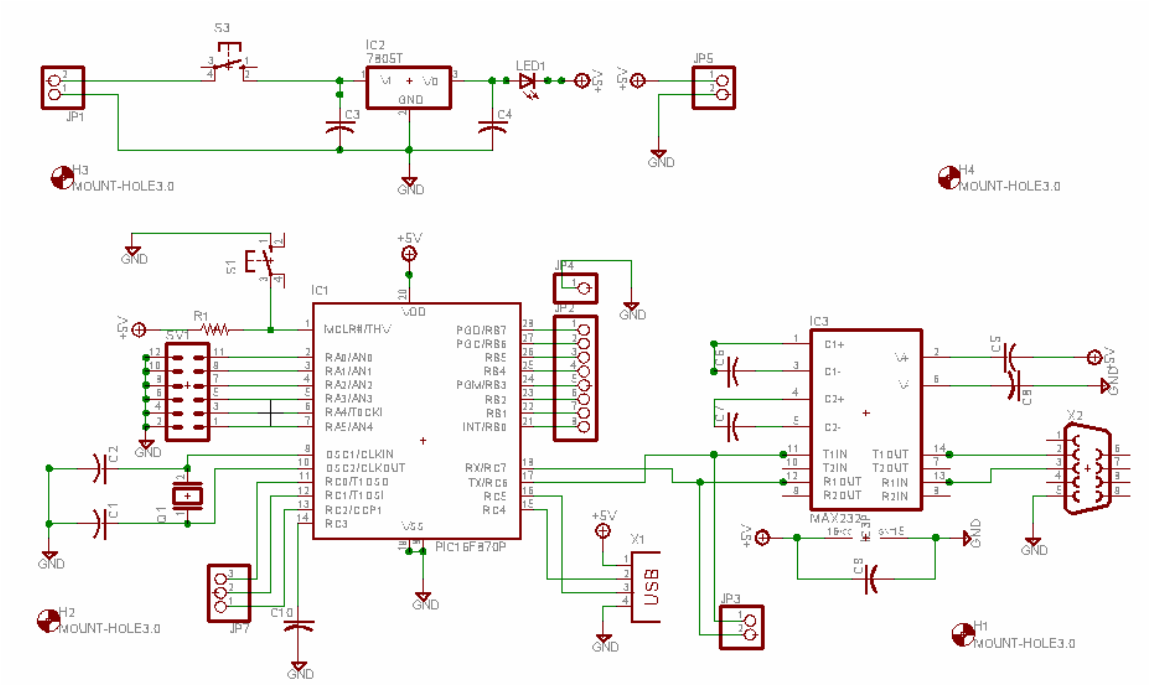


Figura 3.29: Diseño del PCB en Eagle.

Las entradas y salidas son representadas mediante Pinheaders con tal de darle mayor flexibilidad y modularidad al circuito.

Para el diseño en Eagle no se encontró el PIC18F2550 dentro de los componentes disponibles, por lo que se utilizó un PIC16F873, el cual tiene las mismas dimensiones y orden en los pines.

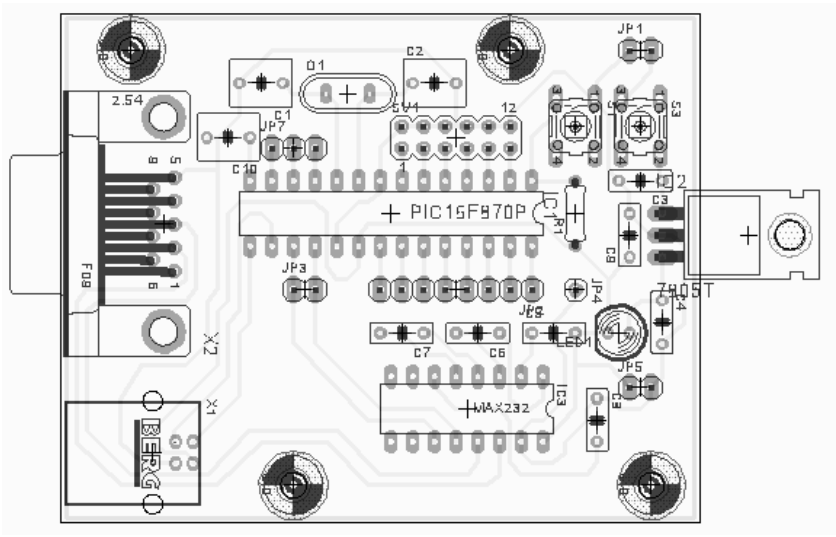


Figura 3.30*:* PCB del proyecto en EAGLE ya ruteado.

3.3.2 FIRMWARE USB

3.3.2.1 Estructura del código

Básicamente se implementó un código hecho con el compilador CCS con la finalidad de configurar el dispositivo en modo Bulk, tal de que se inicialice y enumere en el computador. Posteriormente, se lee repetitivamente el puerto AN 0 (pin1) mediante interrupciones, para tener un tiempo fijo entre muestras y enviar éstas por el puerto USB hacia el computador, como se aprecia en el diagrama de flujo de la figura 3.31:

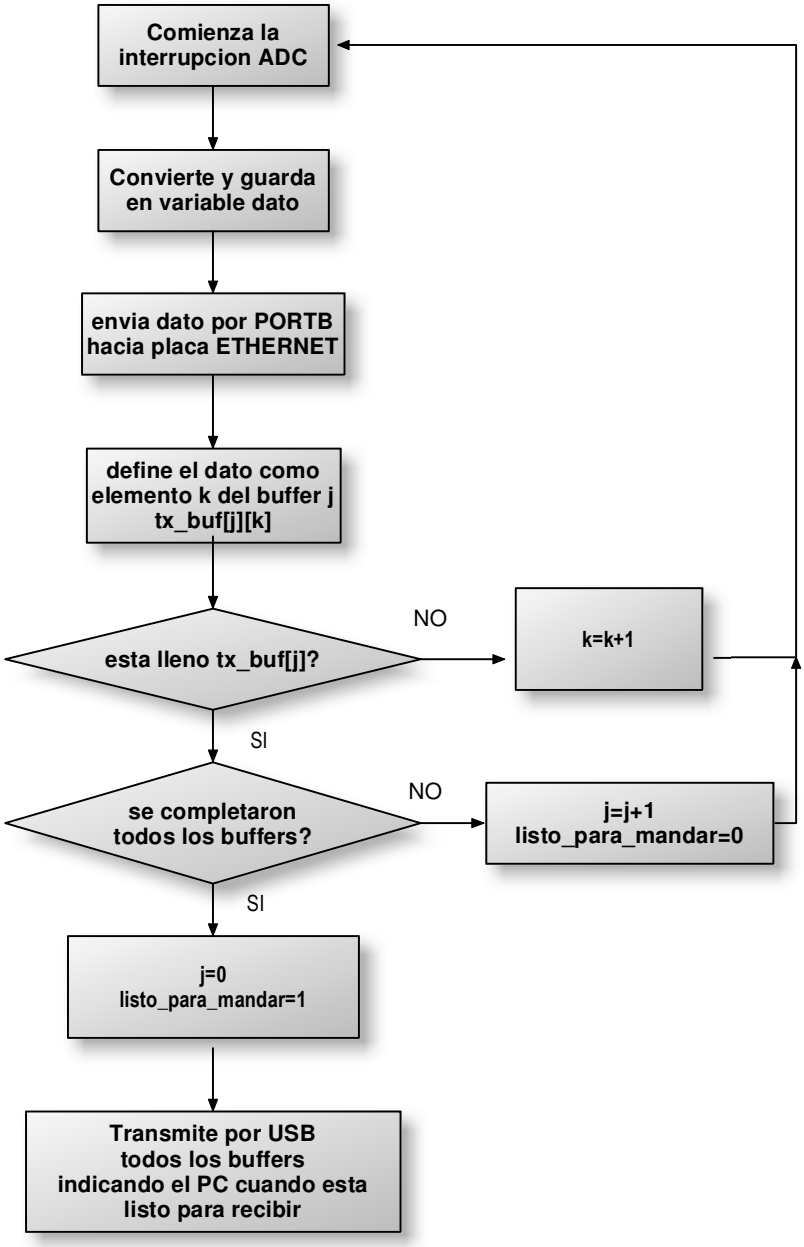


Figura 3.31: Diagrama de flujo para la conversión de datos y transmisión por USB.

Existe una restricción respecto a la transferencia de información, la cual consiste en el límite del paquete a transmitir con un máximo de 64 bytes. Para evitar que se pierdan paquetes, dentro de la interrupción del conversor análogo digital se definieron cuatro arreglos o buffers, con el fin de que no exista pérdida de información en el traspaso por USB.

Una vez que los cuatro buffers se llenaron, un flag indica cuando el computador debe recibir la información. Éste envía comandos numerados secuencialmente pidiendo el paquete respectivo. Cuando se enviaron los cuatro paquetes, entonces éstos se vuelven a actualizar con datos nuevos, teniendo así implementado un control de tiempos. Un esquema sencillo de lo explicado se muestra en la figura 3.32:

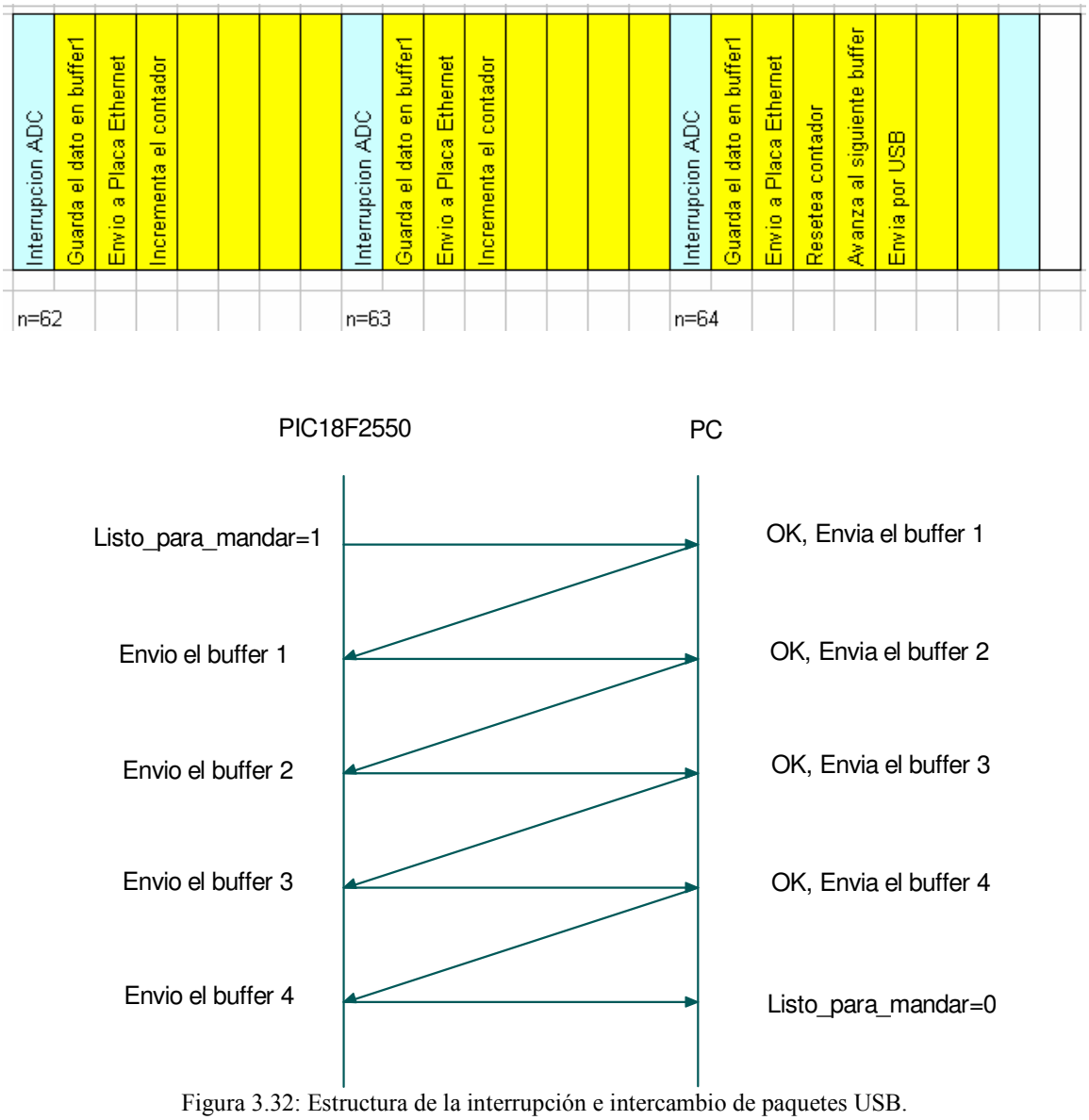


Figura 3.32: Estructura de la interrupción e intercambio de paquetes USB.

3.3.2.2 Tiempo de muestreo

Para definir el tiempo entre muestras, se utilizó interrupciones definidas por el ADC según un divisor dado por el usuario.

Para obtener dicho divisor, el compilador CCS dispone de la utilidad de visualizar tanto el código en C como el Assembler generado simultáneamente. Por lo que, recurriendo a las tablas de OPCODE disponibles en el datasheet del PIC, se puede lograr definir este número.

Se puede definir para esta familia de PICS divisores de 2, 4, 8, 16, 32 y 64 respecto al clock. Entonces la idea es ver con qué número las operaciones incluidas dentro de la interrupción no interfieren con el dato siguiente.

El código de la interrupción en C comparado con Assembler es el siguiente:

```
      ..................... #INT_AD
      ..................... AD_isr(){
[9]   .....................    dato=Read_ADC(ADC_READ_ONLY);
      *
(3)   0BFC:  BTFSC  FC2.1
(2)   0BFE:  BRA    0BFC
(1)   0C00:  MOVF   FC4,W
(1)   0C02:  CLRF   26
(2)   0C04:  MOVWF  25

[2]   .....................    output_b(dato);
(1)   0C06:  CLRF   F93
(1)   0C08:  MOVFF  25,F8A
      .....................

[3]   .....................    if(usb_enumerated()){
(1)   0C0C:  MOVF   1A,F
(2)   0C0E:  BZ    0C24

[9]   .....................       envia[k-1] = dato;
(1)   0C10:  MOVLW  01
(1)   0C12:  SUBWF  28,W
(1)   0C14:  CLRF   03
(1)   0C16:  ADDLW  27
(1)   0C18:  MOVWF  FE9
(1)   0C1A:  MOVLW  00
(1)   0C1C:  ADDWFC 03,W
(1)   0C1E:  MOVWF  FEA
(1)   0C20:  MOVFF  25,FEF
      .....................    }
      .....................

[5]   .....................    if (k==bytes_a_enviar){
(3)   0C24:  DECFSZ 28,W
(2)   0C26:  BRA    0C44
```

```
[12]  ...................        usb_put_packet(1, envia, bytes_a_enviar,
USB_DTS_TOGGLE);
(1)   0C28:  MOVLW  01
(1)   0C2A:  MOVWF  2C
(1)   0C2C:  CLRF   2E
(1)   0C2E:  MOVLW  27
(1)   0C30:  MOVWF  2D
(1)   0C32:  CLRF   30
(1)   0C34:  MOVLW  01
(1)   0C36:  MOVWF  2F
(1)   0C38:  MOVLW  02
(1)   0C3A:  MOVWF  31
(2)   0C3C:  BRA    0B54

[2]   ...................        k=1;
(1)   0C3E:  MOVLW  01
(1)   0C40:  MOVWF  28
      ...................     }

[2]   ...................     else{
(2)   0C42:  BRA    0C48

[2]   ...................        k=k+1;
(1)   0C44:  MOVLW  01
(1)   0C46:  ADDWF  28,F
      ...................     }

[1]   ...................     Read_ADC(ADC_START_ONLY);
(1)   0C48:  BSF    FC2.1
      ................... }
```

Se han incluido números entre corchetes y paréntesis antes de cada sentencia: los primeros equivalen al tiempo total de una línea de código en C y los segundos equivalen a los ciclos de reloj que toma una operación en Assembler en ejecutarse, y la agrupación de determinadas líneas en Assembler conforman una sentencia en C.

Analizando esto, se definen entonces 54 ciclos de reloj dentro de la interrupción, por lo que el divisor que no interfiere con la división siguiente es 64 (64>54).

Entonces, si se define que el tiempo de adquisición para la conversión análogo-digital ocurra una vez cada 64 ciclos de reloj, teniendo este a 48 MHz, se puede decir que:

$$Tad = \frac{64}{48.000.000} = 1.33\mu s$$

Por otro lado, según el datasheet del PIC18F2550, cada conversión de 8 bits requiere 9 veces el tiempo de adquisición (TAD) ya calculado. Lo que equivale a decir que existe $9 \cdot 1.33\mu s = 11.97\mu s$ de separación entre cada muestra. Considerando el tiempo de descarga del condensador de $2\mu s$, se tienen entonces $13.97\mu s$ entre cada muestra.

Como los paquetes se definen con el largo máximo permitido de 64 bytes, entonces cada paquete representara $64 \cdot 13.97\mu s = 894.08\mu s$.

Por teorema de Nyquist, se puede calcular entonces la máxima frecuencia que se puede muestrear como:

$$f\max = \frac{1}{2 \cdot 11.97} kHz = 41.771 kHz$$

En cuanto al ancho de banda, se puede decir entonces que éste es de 83.542 kSa/s.

3.3.3 SOFTWARE USB

Se diseñó el software en Visual Basic, utilizando el API (Application Programming Interface) desarrollado por Microchip para utilizar el puerto USB, considerando una variable entera para definir el largo de la ventana de datos a graficar, modificable mediante un scrollbar cuyo rango varía entre 1 dato y 512 datos.

El modelo por capas se muestra en la figura 3.33:

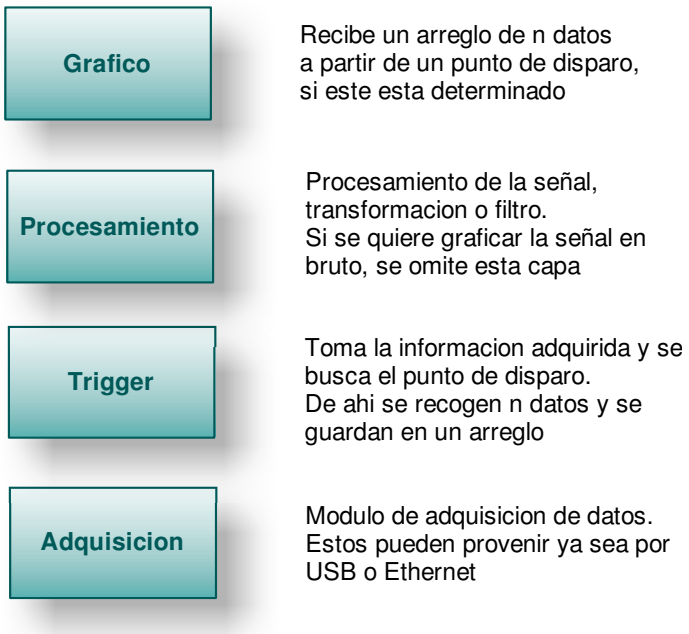


Figura 3.33: Modelo de capas para el software.

Los drivers para Windows son proporcionados por Microchip junto con la API. Ésta última contiene funciones de inicialización, recepción y transmisión de datos incorporados en mpusbapi.dll, la cual hubo que adaptar a Visual Basic, ya que originalmente fue diseñada para C.

3.3.3.1 Recepción de datos

Para la recepción de datos mediante USB, consiste básicamente en definir un arreglo de enteros de largo 64 y un timer modificable que refresque los datos cada cierto tiempo. La línea que define este arreglo es:

```
If (SendReceivePacket(tx_Buf, tx_length, rx_Buf, rx_length, 1000, 1000) = 1) Then
'Aca se procesan los datos recibidos por USB
End If
```

Como se aprecia, está dentro de una condición, en que SendReceivePacket es 1 si efectivamente llegó información desde USB y 0 en caso contrario.

La información recibida se guarda en rx_Buf y a medida que llega se guarda en el string strDatos concatenando tantos buffers de llegada como lo determine el scrollBar que define la escala horizontal.

3.3.3.2 Mecanismo de Trigger

Se diseñó un mecanismo de trigger o disparo por software, en el cual se ajusta un nivel de disparo tal que la señal quede en lo posible centrada dentro del gráfico. Una vez que se detectó el punto de disparo, entonces se guarda una ventana de datos, la cual posteriormente será graficada.

Básicamente, los datos a graficar, se guardarán en un arreglo entero de datos llamado ventana. Para el llenado de datos a partir de la información recibida por el conversor, se define un largo "n" el cual será la dimensión del arreglo.

Posteriormente, se reciben datos y se guardan en un arreglo llamado strDatos que se actualiza mientras van llegando los datos, con el fin de que si se detectó el punto de disparo, entonces éste quede en la mitad de los datos a graficar.

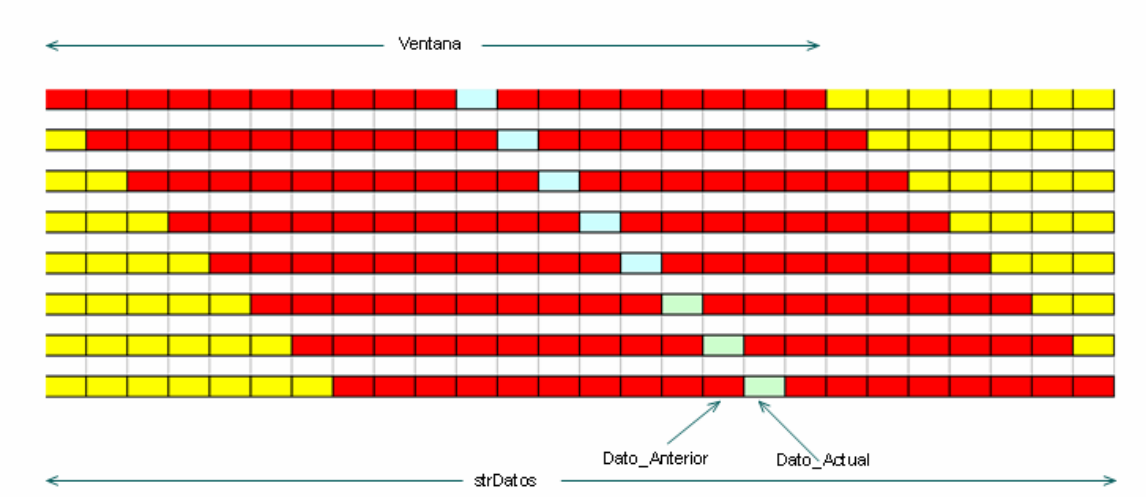


Figura 3.34 Esquema de funcionamiento del trigger.

Como se ve en la figura 3.34, se definen dos strings: primero, strDatos, recopilado directamente desde USB o Ethernet, dentro de la capa física; y segundo, ventana, el cual una vez recopilados los datos y encapsulados en el arreglo ventana, se procede a graficarlos. Esto se realiza recorriendo el arreglo.

Se definen además dos variables auxiliares: Dato_actual, la cual varía según el contador que recorre el string; y Dato_anterior, el cual varía según la posición actual menos uno. La finalidad de estas dos variables es compararlas y ver si está en flanco de subida o bajada. También, dato actual debe estar dentro del margen de disparo para dejar el flag de trigger como verdadero.

En la figura 3.35 se muestra el diagrama de flujo del mecanismo de recepción de datos y de trigger:

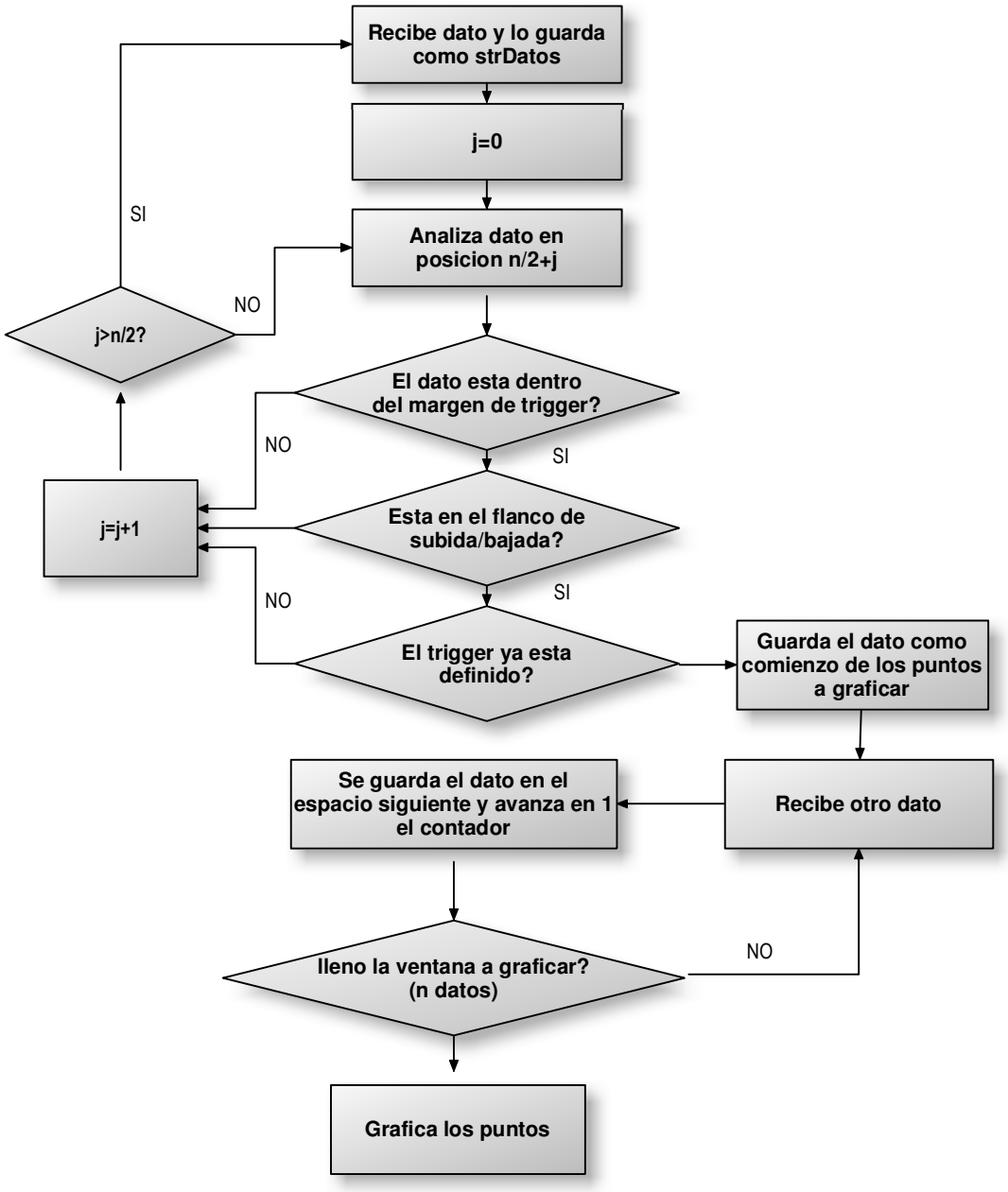


Figura 3.35 Diagrama de flujo para la adquisición de datos.

3.3.3.3 Gráfico de datos

Una vez recopilados los datos y encapsulados en el arreglo ventana, se procede a graficarlos. Esto se realiza recorriendo el arreglo completo y tomando dos puntos (x0,y0) y (x1,y2) para posteriormente unirlos mediante líneas. Un contador inicializado en 0 irá recorriendo el arreglo completo para esta función, como se aprecia en la figura 3.36

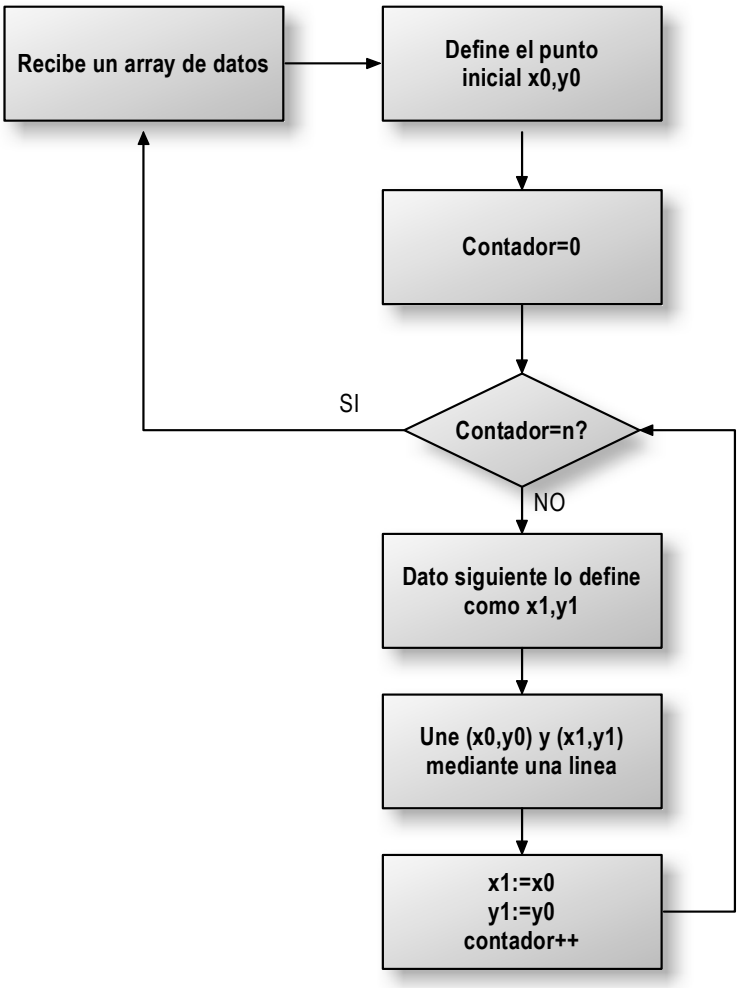


Figura 3.36 Diagrama de flujo para graficar los puntos.

Una vez compilado y ejecutado el código, se tiene la siguiente interfaz gráfica:

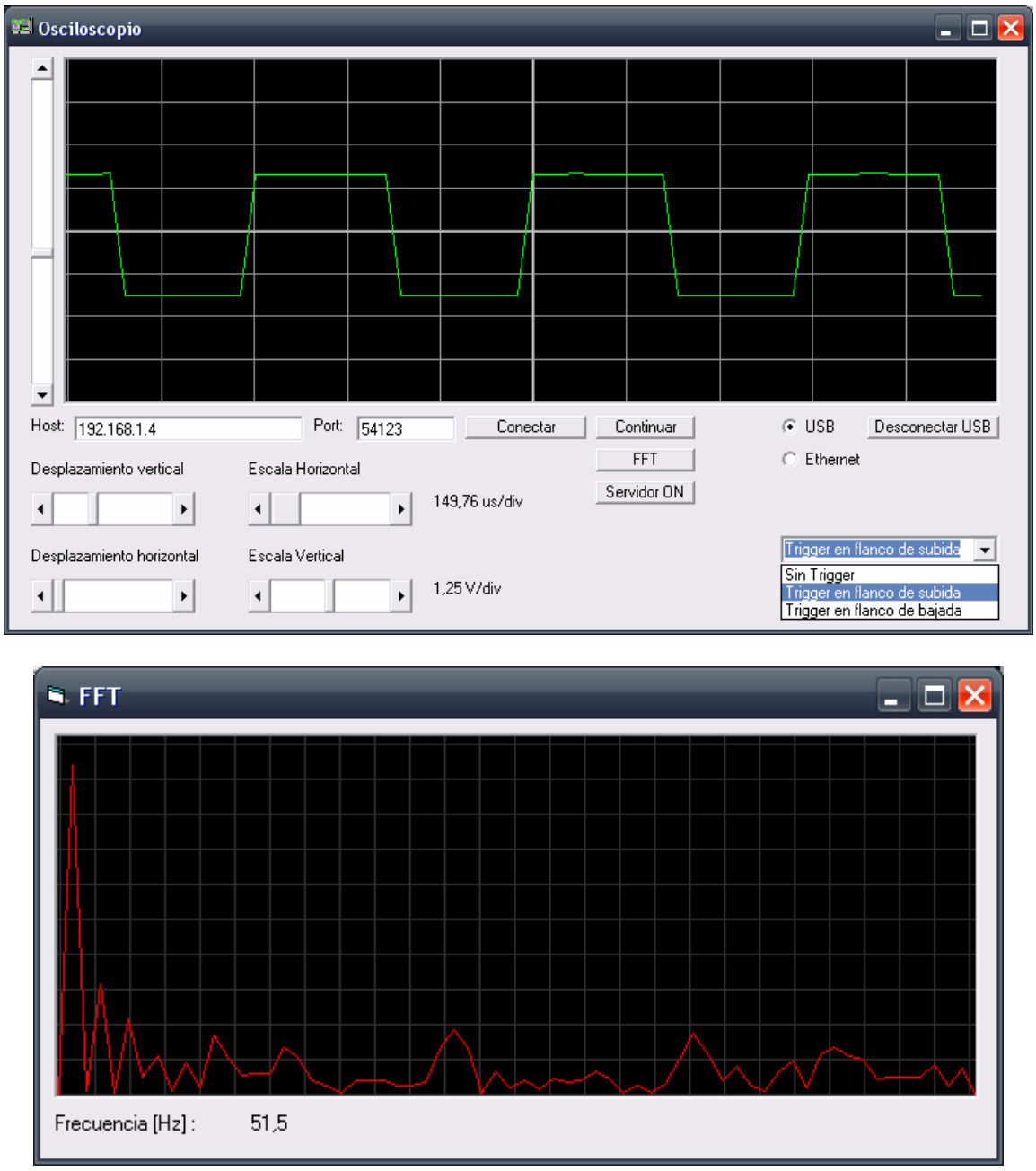


Figura 3.37 Interfaz gráfica.

En la figura 3.37 dispositivo está funcionando monitoreando un tren de pulsos obtenido de un 555 en modo estable.

3.3.3.4 Sistemas de deflexión horizontal y vertical

Para tener un control sobre el análisis de la señal a graficar, se definió una grilla de fondo de 8 divisiones verticales y 10 horizontales.

La idea detrás de esto es que mediante la cantidad de paquetes que se graficarán, se divida el eje horizontal en tantas partes como datos en total se hayan recibido.

Entonces, si se dispone de un control Picture box de ancho "Picture1.width", y en este se graficaran "n" datos, se tendrán $n \cdot 13.97 \mu s$ dentro de la pantalla y por lo tanto cada división se definirá como:

$$\frac{n}{10} \cdot 13.97[\mu s / div]$$

Ahora, en relación a la deflexión vertical, si el control Picture box se define con un alto "Picture1.height" y en éste se graficarán intensidades representadas con valores que variarán desde 1 a 256 y como se dispone de 8 divisiones, entonces cada división representara 32 niveles de intensidad.

Para el caso de un rango dinámico de 5 volts y un factor de multiplicación "f", entonces se puede decir que cada división tendrá:

$$\frac{32 \cdot 5 \cdot f}{256} = 0.625 \cdot f[V / div]$$

3.4 SEGUNDA FASE: ETHERNET

3.4.1 HARDWARE

Para darle conectividad Ethernet a la placa de adquisición de datos, este trabajo se basará en la placa PIC-WEB de Olimex, la cual trae incorporado un microcontrolador PIC18F452 y un controlador ethernet ENC28J60 más un conector RJ45.

Se eligió esta placa porque es adaptable mediante conexión paralela para recibir los datos de la otra placa en el portB. A continuación se describirán los bloques principales de la placa Ethernet.

Alimentación: básicamente, la placa funciona con tensiones de 5 volt (PIC) y 3.3 volt (Controlador Ethernet). Contiene el regulador LM117 (figura 3.38) el cual está configurado para obtener los 3.3 V y un administrador de energía MC34863AD (figura 3.39) para obtener los 5 volt. Sus esquemáticos son los siguientes:

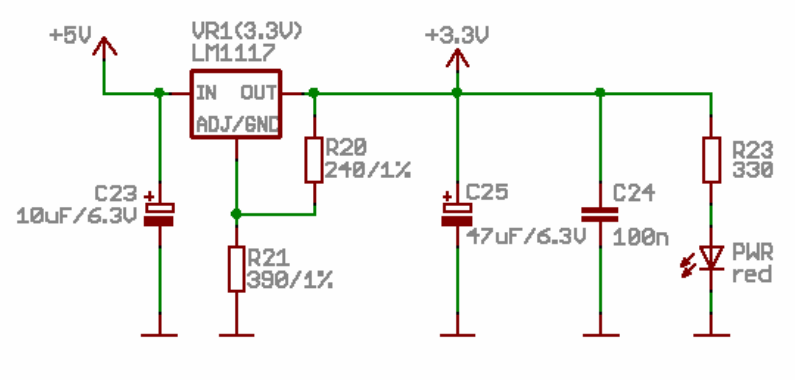


Figura 3.38: Obtención de 3.3 volts para placa PICWEB.

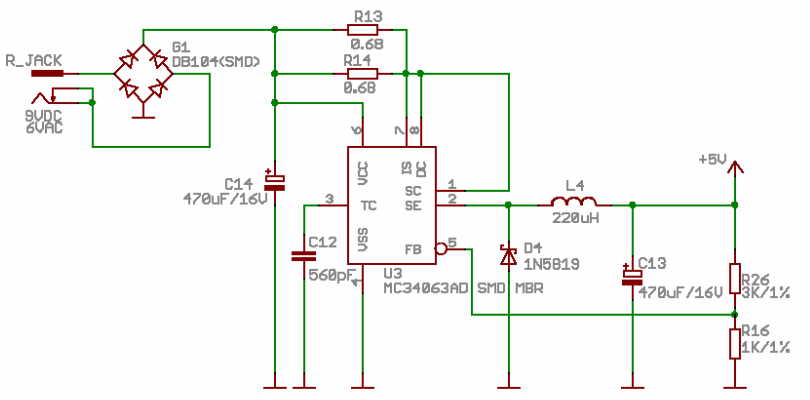


Figura 3.39: Obtención de 5 volts para placa PICWEB.

En cuanto al controlador ENC28J60 (figura 3.40) , se observan las salidas Tx+,Tx-,Rx+ y Rx- que irán al conector Rj45, además de las salidas LedA y LedB que manejan los leds de estado del conector. La comunicación entre el microcontrolador PIC y el Enc28j60 es mediante los pines 6,7y 8 los cuales son de entrada y salida de datos y un clock de sincronización.

También se aprecia el cristal de 25 Mhz con el que funcionará el controlador. Además están los pines de Chip Select siempre activo y Reset que va al pin ETH_RST relacionado con el PIC. Se observa, de igual manera, la polarización de 3.3 V.

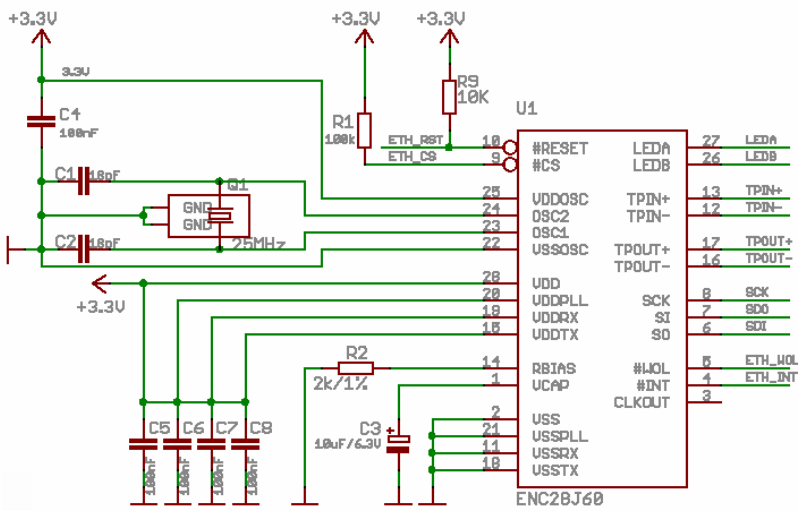


Figura 3.40: Bloque controlador Ethernet.

Por otro lado, está el conector RJ45 (figura 3.41), en el cual se observan los transformadores internos que serán los encargados de la transmisión a nivel físico de la información.

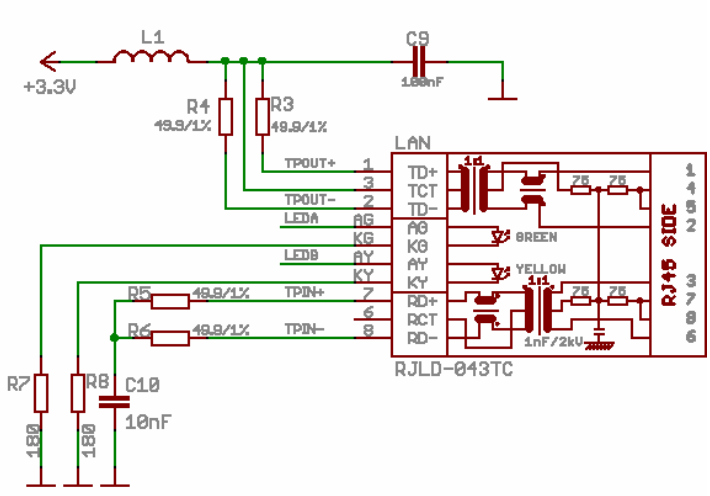


Figura 3.41: Conector RJ45.

También el microcontrolador incorpora una salida serial con la que se pueden monitorear los datos mediante un Terminal. La comunicación en esta fase, análoga a la placa USB, es mediante un chip max232 que convierte señales TTL a RS232 como se aprecia en la figura 3.42.

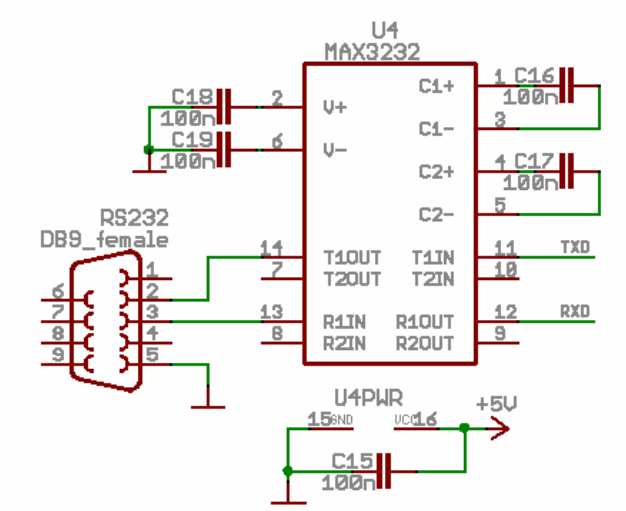


Figura 3.42: Transductor MAX232 y conector DB9.

Finalmente, está el microcontrolador PIC18F452 (figura 3.43) , el cual incorpora un conector Molex para su programación ICSP (que no requiere desmontar el integrado para su programación). Su clock principal es de 10 Mhz y también tiene varios puertos de propósito general, los cuales servirán de puente con la placa USB.

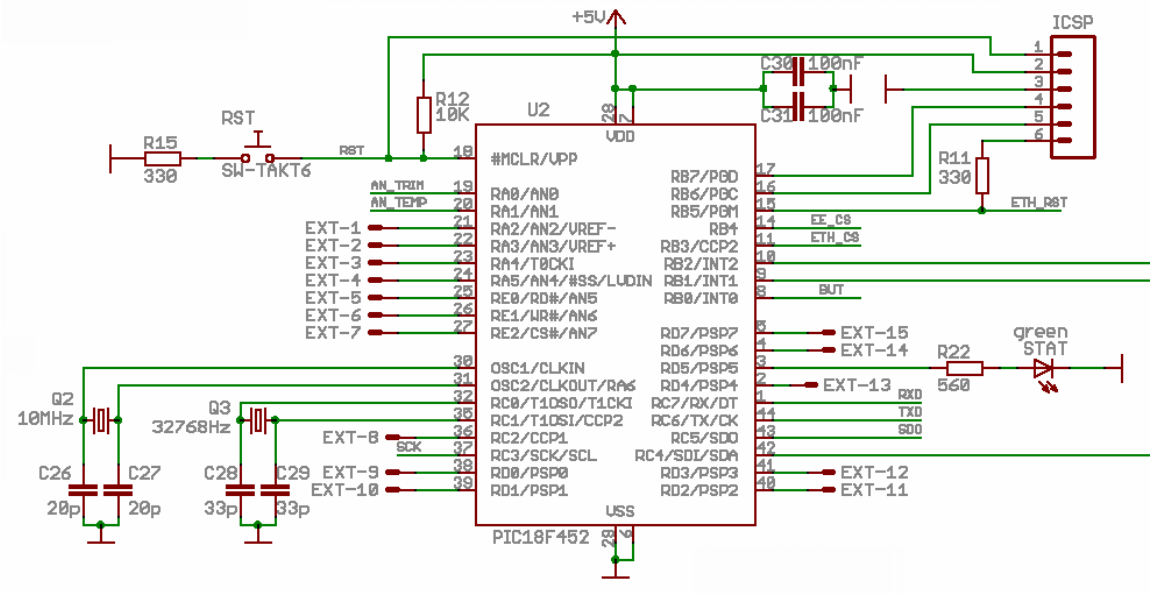


Figura 3.43: Microcontrolador PIC18F452.

3.4.2 Firmware

Para la etapa de diseño del firmware Ethernet, se tomó en cuenta los pines del conector de extensión de la placa PIC-WEB. La disposición física de los pines se muestra en la figuras 3.44 y 3.45:

EXT 2	EXT 4	EXT 6	EXT 8	EXT 10	EXT 12	EXT 14	EXT 16	EXT 18	EXT 20
A3	A5	E1	C2	D1	D3	D6	RST	3.3v	Vin
A2	A4	E0	E2	D0	D2	D4	D7	5v	GND
EXT 1	EXT 3	EXT 5	EXT 7	EXT 9	EXT 11	EXT 13	EXT 15	EXT 17	EXT 19

Figura 3.44: Pin-out Puerto EXT.

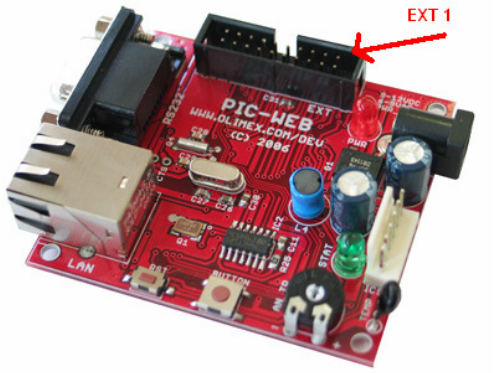


Figura 3.45: Ubicación del pin 1.

Básicamente se barajaron dos opciones en el diseño: la primera consistió en utilizar el conversor de la placa USB y posteriormente enviar los datos por el portb paralelamente hacia el conector EXT de la placa ethernet (mediante los pines destacados en amarillo). La segunda opción fue utilizar los conversores de ambas placas simultáneamente, según los datasheet, las características del modulo adc para ambos microcontroladores son idénticas.
A continuación se muestran los detalles del primer diseño:

La idea fundamental fue establecer una comunicación paralela con la Placa USB, la cual contiene el conversor ADC. La equivalencia entre puertos de las distintas placas es la mostrada en la figura 3.46:

Placa USB	B0	B1	B2	B3	B4	B5	B6	B7
Placa Ethernet	A2	A3	E0	E2	D0	D2	D4	D7

Figura 3.46: Conexionado entre placas.

Por lo tanto, a la placa Ethernet sólo le llega información individual de cada pin, la cual primero debe compactarse para dar lugar a un carácter, luego transmitirlo y en el software que lo recibirá, obtener su número equivalente a partir de su correspondiente ASCII y

finalmente graficar el punto y trazar líneas entre puntos adyacentes como se muestra en el diagrama de flujo de la figura 3.48. Para compactar la información recibida por los pines, se recurrió a la siguiente expresión:

$$Dato = 2^0 \cdot A_2 + 2^1 \cdot A_3 + 2^2 \cdot E_0 + 2^4 \cdot E_2 + 2^5 \cdot D_0 + 2^6 \cdot D_2 + 2^7 \cdot D_4 + 2^8 \cdot D_7$$

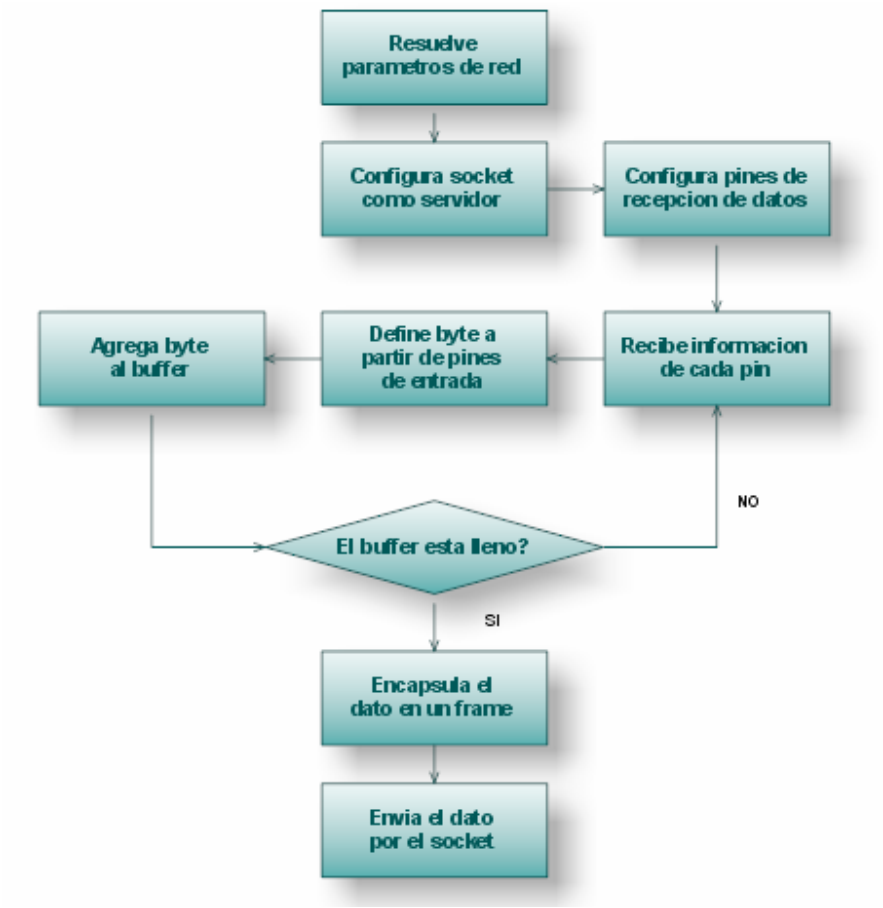


Figura 3.47: Diagrama de flujo para la adquisición de datos y transmisión por Ethernet.

En primer lugar, se definieron los pines del puerto EXT de la siguiente forma:

```
//CONECTOR EXT (PRIMERA FILA)
#define LATA2                   LATAbits.LATA2
#define LATA4                   LATAbits.LATA4
#define LATE0                   LATEbits.LATE0
#define LATE2                   LATEbits.LATE2
#define LATD0                   LATDbits.LATD0
#define LATD2                   LATDbits.LATD2
#define LATD4                   LATDbits.LATD4
#define LATD7                   LATDbits.LATD7

//CONECTOR EXT (SEGUNDA FILA)
#define LATA3                   LATAbits.LATA3
#define LATA5                   LATAbits.LATA5
#define LATE1                   LATEbits.LATE1
```

```
#define LATC2                LATCbits.LATC2
#define LATD1                LATDbits.LATD1
#define LATD3                LATDbits.LATD3
#define LATD6                LATDbits.LATD6
```

En codigo C18, las variables a usar son:

```
unsigned char byte0;
unsigned char byte1;
unsigned char byte2;
unsigned char byte3;
unsigned char byte4;
unsigned char byte5;
unsigned char byte6;
unsigned char byte7;

//se actualizan los valores para los pines del puerto ext

      if ( PORTA_RA2 == 0 ){byte0=0;} else{byte0=1;}
      if ( PORTA_RA3 == 0 ){byte1=0;} else{byte1=1;}
      if ( PORTE_RE0 == 0 ){byte2=0;} else{byte2=1;}
      if ( PORTE_RE2 == 0 ){byte3=0;} else{byte3=1;}
      if ( PORTD_RD0 == 0 ){byte4=0;} else{byte4=1;}
      if ( PORTD_RD2 == 0 ){byte5=0;} else{byte5=1;}
      if ( PORTD_RD4 == 0 ){byte6=0;} else{byte6=1;}
      if ( PORTD_RD7 == 0 ){byte7=0;} else{byte7=1;}
```

En segunda opción, se tiene que el pin EXT1 corresponde a RA2 del PIC18F452, además también corresponde al ADC AN2. Por lo tanto, se puede configurar este pin como conversor mediante los registros mostrados en la figura 3.48:

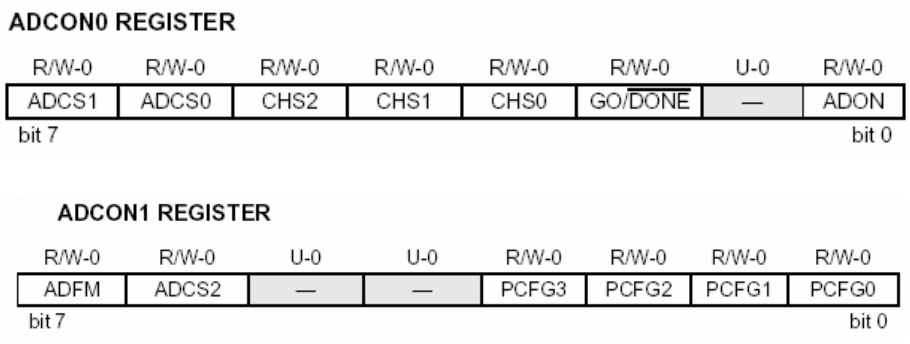


Figura 3.48: Registro de control del ADC.

Mediante estos registros de 8 bits cada uno, se pueden configurar: para ADCON0, el clock de conversión ADC, el canal seleccionado y el estado (activo/inactivo) del canal; y para ADCON1, el formato del resultado de conversión y los bits que definen si un pin es entrada o salida digital de propósito general o un canal conversor.

Al ejecutarse el clock principal de ambas placas a la misma velocidad, y configurando el tiempo de adquisición con igual parámetro para éstas, entonces la conversión de datos será simultánea. El diagrama de flujo es el mostrado en la figura 3.49:

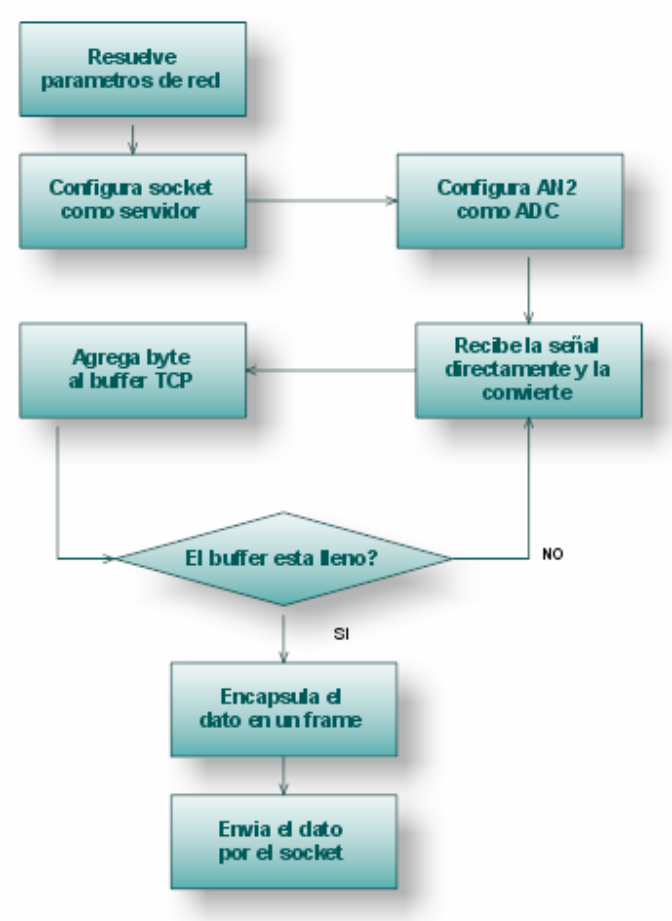


Figura 3.49: Diagrama de flujo para la segunda alternativa.

Para resolver los parámetros de red, existen dos alternativas según la red a la que se conecte: configurar la dirección IP, máscara de subred y gateway de forma estática; y por otro lado activar la opción de resolver la IP dinámicamente mediante DHCP. Ambas opciones están incluidas dentro del stack TCP/IP de Microchip.

Para enviar de forma correcta la información es conveniente diseñar un mecanismo de handshacking. Con esto se asegura un sincronismo, independientemente si se usa USB o Ethernet para la transmisión de datos.

Para implementar el handshaking se requieren dos líneas adicionales: La línea de strobe es la que utiliza la parte transmisora para indicarle a la parte receptora la disponibilidad de información; la línea de acknowledge es la que utiliza la parte receptora para indicarle a la parte transmisora que ha tomado la información y que está lista para recibir más datos. El puerto paralelo provee de una tercera línea de handshaking llamada busy, ésta la puede

utilizar la parte receptora para indicarle a la parte transmisora que está ocupada y por lo tanto la parte transmisora no debe intentar colocar nueva información en las líneas de datos.

Una típica sesión de transmisión de datos se parece a lo siguiente:

Parte transmisora:

- La parte transmisora ve la línea busy para ver si la parte receptora está ocupada. Si la línea busy está activa, la parte transmisora espera en un bucle hasta que la línea busy esté inactiva.
- La parte transmisora coloca la información en las líneas de datos.
- La parte transmisora activa la línea de strobe.
- La parte transmisora espera en un ciclo hasta que la línea acknowledge está activa.
- La parte transmisora inactiva la línea de strobe.
- La parte transmisora espera en un bucle hasta que la línea acknowledge esté inactiva.
- La parte transmisora repite los pasos anteriores por cada byte a ser transmitido.

Parte receptora:

- La parte receptora inactiva la línea busy (asumiendo que está lista para recibir información).
- La parte receptora espera en un bucle hasta que la línea strobe esté activa.
- La parte receptora lee la información de las líneas de datos (y si es necesario, procesa los datos).
- La parte receptora activa la línea acknowledge.
- La parte receptora espera en un bucle hasta que esté inactiva la línea de strobe.
- La parte receptora inactiva la línea acknowledge.
- La parte receptora repite los pasos anteriores por cada byte que debe recibir.

Hay que tener cuidado al seguir estos pasos. Tanto la parte transmisora como la receptora coordinan sus acciones de tal manera que la parte transmisora no intentará colocar varios bytes en las líneas de datos, en tanto que la parte receptora no debe leer más datos que los que le envíe la parte transmisora, un byte a la vez.

En la figura 3.50 se muestra el diagrama de tiempo en un proceso típico:

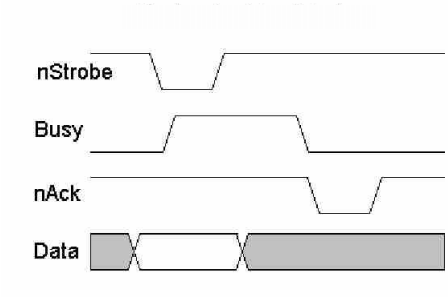


Figura 3.50: Handshacking entre placas.

Para la placa USB se definirán como pines de handshacking los pines: A1 (Busy), A2 (Acknowledge) y A3 (Strobe). Por otro lado, para la placa Ethernet, los pines de handshacking serán: D1(Busy), D2 (Acknowledge) y D6 (Strobe) los cuales están destacados en azul dentro del pinout del puerto EXT de la placa Ethernet. Considerando esto, se puede representar mediante el siguiente código:

Parte transmisora [utilizando lenguaje ccs]

```
#define busy         PIN_A1
#define acknowledge  PIN_A2
#define strobe       PIN_A3

set_tris_a(0b00000110); //a1=ack y a2=busy [entradas] a3=strobe [salida]
set_adc_channel (0);
medicion=read_adc ();

if(!input(busy)){               //si no esta ocupado

   output_b(medicion);          //pone el byte en el bus de datos
   output_high(strobe);         //le avisa al receptor que hay
disponibilidad de datos
   while(!input(acknowledge)){} //espera a que acknowledge este activo
(que rx ya haya leido el dato)
   output_low(strobe);          //ya no hay datos disponibles en tx
   while(input(acknowledge)){}  //espera a que acknowledge este inactivo
}
```

Parte receptora [utilizando lenguaje c18]

```
unsigned char busy;
unsigned char acknowledge;
unsigned char strobe;

//se definen las lineas busy, ack como salida y strobe como entrada
TRISD_RD1=0; //busy
TRISD_RD3=0; //ack
TRISD_RD6=1; //strobe


//strobe (input)
```

```
if ( PORTD_RD6 == 0 ){strobe=0;} else{strobe=1;}

LATD1 ^=0; //Se desactiva la linea busy
while(strobe=0){} //espera a que hayan datos disponibles

//ACA SE RECOPILA LA INFORMACION DE CADA PIN DENTRO DE UN UNICO CARACTER
dato_int=byte0*1+byte1*2+byte2*4+byte3*8+byte4*16+byte5*32+byte6*64+byte7
*128;

if (TCPIsConnected(tcpSocketUser)){

TCPPut(tcpSocketUser,dato_int);

TCPFlush(tcpSocketUser);
StackTask();

LATD3 ^=1;  //activa la linea acknowledge

while(strobe=1){} //espera hasta que este inactiva la linea de strobe
```

En cuanto a la velocidad de transferencia, esta dependerá del tráfico de la red. Su límite máximo según las especificaciones del datasheet del ENC28J60 corresponde al límite de velocidad en una red 10 base T, es decir 10 Mbits/s.

Esto traduciéndolo a bytes de datos por segundos se calcula convirtiendo la velocidad máxima a bytes (10000000/8=1250000) y dividiéndolo por el numero de paquetes que caben en 1250000 bytes.

El frame Ethernet se define según la figura 3.51:

Preámbulo	SOF	Destino	Origen	Tipo	Datos	Relleno	FCS
7 bytes	1 byte	6 bytes	6bytes	2 bytes	0 a 1500 bytes	0 a 46 bytes	4 bytes

Figura 3.51: Diagrama Frame Ethernet.

Por lo tanto, hay 27 bytes en cada paquete enviado (considerando 1 byte de datos y 0 de relleno). Entonces, se pueden enviar hasta 46626 paquetes por segundo.

3.4.3 Software

Principalmente, esta etapa consistió en el desarrollo de un módulo para el programa principal que brinda la conexión Ethernet al programa comentado en la fase USB.

Tiene la principal ventaja de que funciona como cliente, es decir, al tener la IP de la placa se puede conectar y recibir la información mediante un socket definido por el control MSWINSCK.OCX propio de Visual Basic 6.

Se agregaron al formulario principal dos checkbox, los cuales alternan entre la recepción USB o la recepción Ethernet. Además se incorporaron dos textbox nuevos: uno para la

definición del host servidor y otro para especificar su puerto como se aprecia en la figura 3.52.

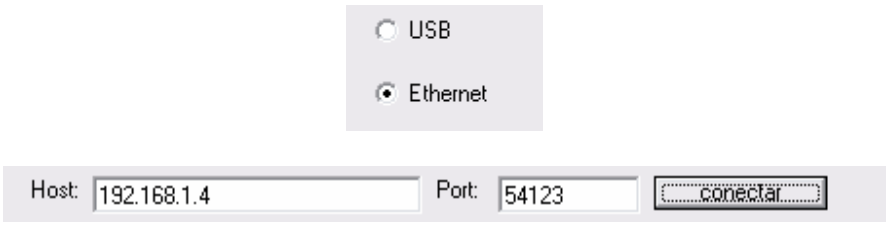


Figura 3.52: Conmutación USB-Ethernet.

Por otro lado, para lograr la conexión, se recurrió a las funciones incorporadas en el control ya mencionado.

Para definir la conexión, se tiene:

Winsock.Remotehost: Define el host al cual se conectará.
Winsock.Remoteport: Define el puerto de este host, el cual da el servicio.

Winsock.connect: Inicia la conexión.
Winsock.close: Cierra la conexión.

Winsock_dataarrival: Interrumpe el programa para avisar que llegó un dato nuevo en el socket.

El diagrama de flujo es idéntico al de la fase USB, con la única diferencia de que la adquisición del dato funciona de la forma ya mencionada. Además, se usa el mismo módulo de trigger para graficar el dato.

CAPITULO 4: DISCUSION DE RESULTADOS

En este punto se verán los cambios que se fueron dando desde un comienzo hasta la elaboración del producto final en cuanto al diseño tanto de hardware como de software.

Posteriormente, se hará una comparación entre el rendimiento de un osciloscopio convencional y el diseñado en este informe. Y, finalmente, se mencionarán algunos aspectos que se pueden mejorar a futuro para crear a partir de este proyecto un dispositivo con mejores prestaciones.

En un comienzo se diseñó el diagrama de bloques del osciloscopio a partir de la investigación realizada sobre osciloscopios convencionales. Posteriormente, se analizó cuáles bloques podían ser implementados por hardware y cuáles por software.

Se creó un sistema enfocado a la modularidad, es decir, que mediante el cambio de ciertos bloques o módulos se pudiera adaptar el sistema a distintos entornos.

4.1 Sobre la selección de componentes

En cuanto a la selección de componentes, se barajaron distintas alternativas para cada bloque, donde se dio prioridad a reducir el margen de error en las mediciones.

Por ejemplo, en el bloque de acondicionamiento de la señal, primeramente se considero un sistema basado en decodificadores, negadores e interruptores lógicos, para poder modificar el arreglo de resistencias que define la ganancia del sub-bloque amplificador.

Para el caso de interruptores digitales, se puede utilizar el CMOS4066, el cual consiste en cuatro interruptores controlados digitalmente como se ve la figura 4.1:

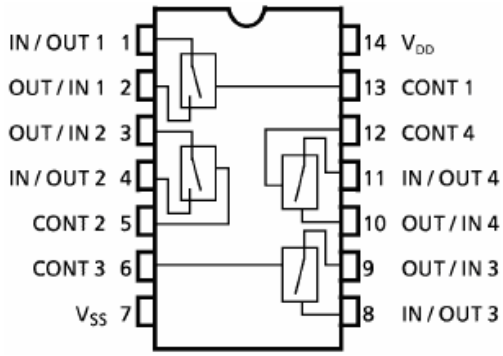


Figura 4.1: Pinout CMOS4066.

Para minimizar los bits de control, también a esto se añade un decodificador 3 a 8, con el cual, para controlar 8 niveles de ganancia, se necesitarían 3 bits en el microcontrolador. El esquema de este decodificador es el mostrado en la figura 4.2:

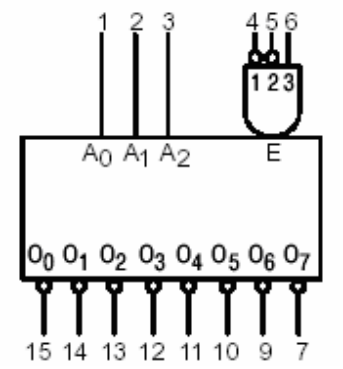


Figura 4.2: Pinout Decodificador 74hc138.

Como se observa, las señales A0, A1 y A2 controlarán una de las ocho salidas del circuito integrado. Pero un detalle muy importante, es que dichas salidas están negadas, por lo que es necesario agregar negadores antes de entrar a los interruptores lógicos. Para esto es posible utilizar el integrado 74ls04 o similar, el cual tiene seis negadores en su interior. Por lo que en ese caso se necesitarían dos pastillas para negar todas las salidas del decodificador según se observa en la figura 4.3.

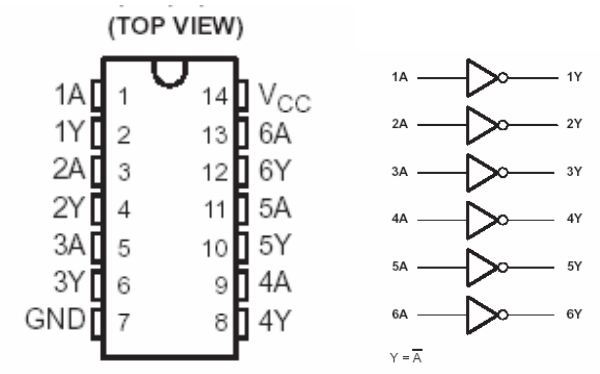


Figura 4.3: Pinout Negador 74hc04.

Resumiendo todo esto, es posible realizar el siguiente diagrama de flujo en que se observan estos componentes analizados y los bits de cada enlace.

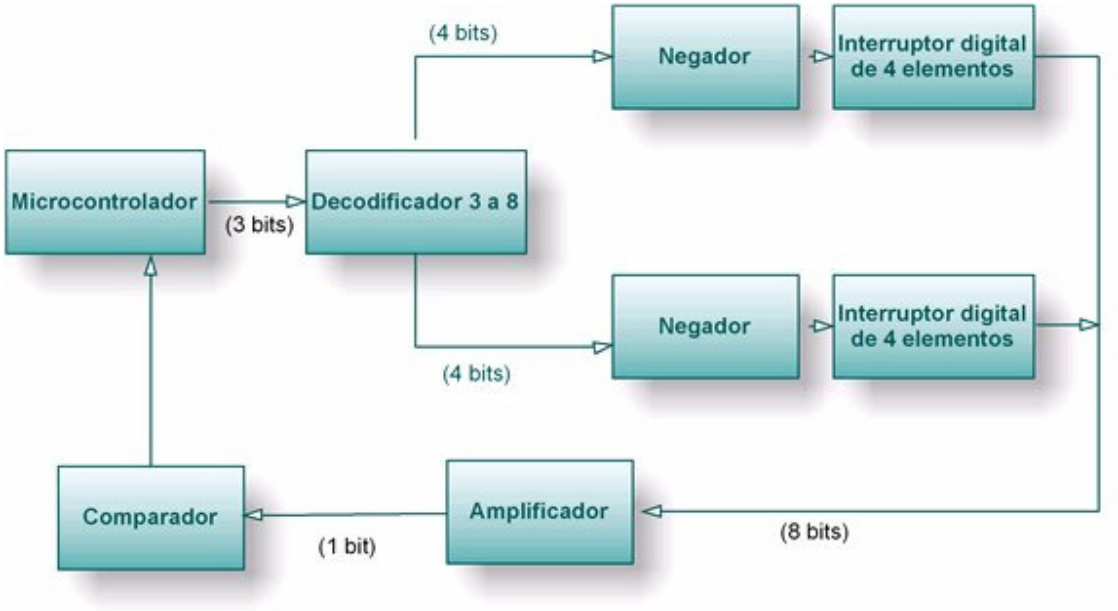


Figura 4.4: Opción 1 para el diseño del amplificador de ganancia programable.

Pero el problema que implica este diseño es que, al usar mayor número de circuitos integrados, se incrementa el error en las mediciones. Entonces se optó por utilizar directamente un solo chip multiplexor de ocho canales, que efectúa el mismo trabajo, y disminuye considerablemente el error. Además, por este mismo motivo, las resistencias utilizadas tienen un margen de tolerancia del 1%.

También cabe mencionar que en cuanto al amplificador se vio primero la posibilidad de implementarlo a partir de un LM741, pero al ser éste polarizado simétricamente, entonces sería necesario también incluir un circuito exclusivo de alimentación que suministrara voltajes negativos, lo cual no sería conveniente, ya que el resto del sistema trabaja en un nivel de entre 0 y 5 volts.

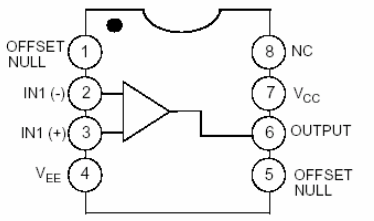


Figura 4.5: Pin Out LM741

Por esto se eligió usar un LM324, ya que el hecho de operar con una alimentación entre 0 y 5 volts lo hace mas fácil de manejar. Además, incluye cuatro amplificadores operacionales en su interior, lo cual también hace disminuir su error debido a la reducción de componentes a utilizar.

En cuanto al microcontrolador, se prefirió utilizar un PIC ya que es más resistente a ambientes industriales, interferencias, golpes, caídas, etc. Además, si eventualmente llegara a quemarse, puede ser sustituido fácilmente.

En cuanto a la interfaz controladora Ethernet, se prefirió el ENC28J60 debido a su compatibilidad con productos de microchip y su conexión con el microcontrolador.

4.3 Comunicación USB

En esta sección se verán los principales problemas que hubo con el diseño de la comunicación USB. Cabe reiterar que esta comunicación está restringida por el driver de Microchip, el cual sólo admite paquetes de tamaño máximo de 64 bytes.

Al principio, se definieron paquetes de largo 1 byte, ya que se pensó en un comienzo en una constante tasa de transferencia de USB, pero al momento de implementarlo, funcionó bien al ser el único dispositivo dentro del bus. En cambio, al conectar otro equipo al bus USB (un pendrive por ejemplo) se perdía calidad de la señal.

Analizando con un monitor USB freeware, se observó que la llegada de paquetes era irregular, esto debido a que habían paquetes del otro dispositivo intercalados. Entonces se pensó en aprovechar al máximo el tamaño de paquete, así el tiempo entre muestras sería constante, ya que la lectura del conversor análogo digital está definida por interrupciones de tiempo constante. Utilizando un osciloscopio comercial se logró el análisis de la figura 4.6:

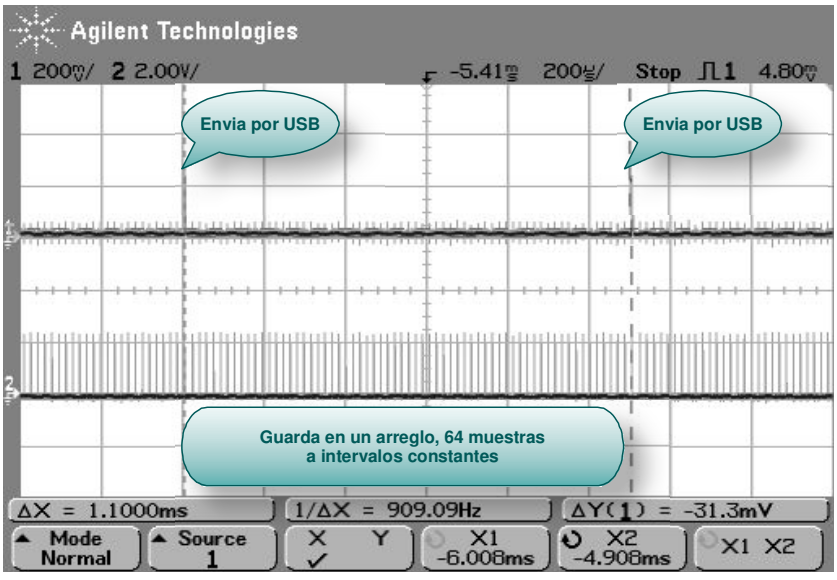


Figura 4.6: Lectura ADC y Transmisión USB.

Como se observa en la imagen, utilizando los controles “cursors” del osciloscopio, se midió en la práctica el tiempo entre envío de paquetes por USB (espacio entre líneas punteadas). El resultado fue de 1.1 ms para 64 datos con separación constante de:

$$t = \frac{1.1 \cdot 10^{-3}}{64} = 0{,}000171875 = 1.7 \cdot 10^{-4}[seg]$$

Conviene observar también cuanto tiempo consume el envío del paquete:

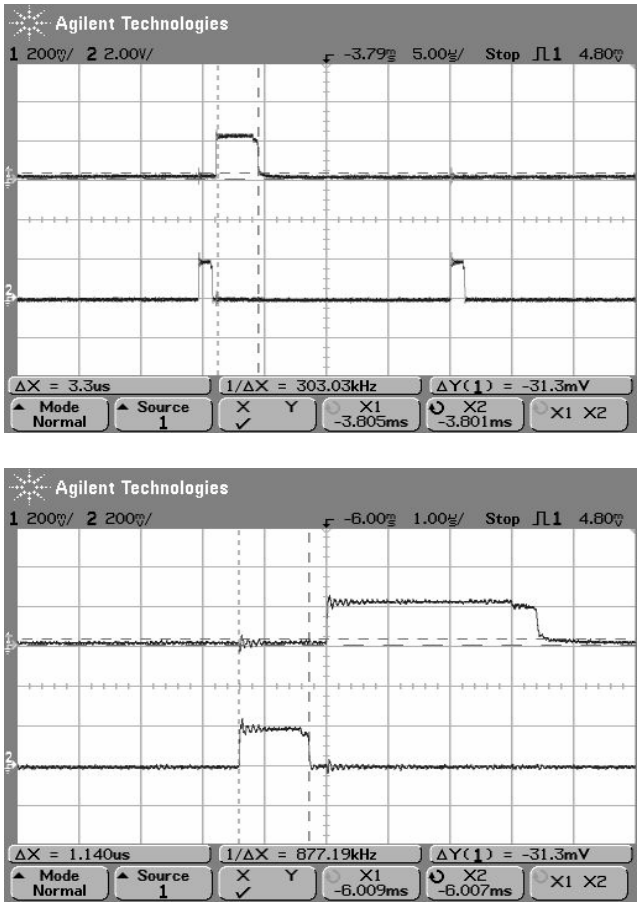


Figura 4.7: Lectura ADC y Transmisión USB (medición de tiempos).

Acá el canal inferior corresponde a la toma constante de muestras y el superior en el momento en que se envían los datos (figura 4.7). La segunda imagen es un zoom de la primera con la finalidad de medir el tiempo que toma la adquisición de un dato.

Como se aprecia en los resultados, el envío de un paquete toma 3.3 μs mientras que la adquisición del dato toma 1.140 μs, por lo cual el hecho de enviar un paquete por USB no interferirá la lectura siguiente.

También es importante evitar, a la hora de hacer un debuggeo en el sistema, que las interrupciones no se traslapen. Esto se logra añadiendo una línea que encienda y otra que

apague un pin dentro del ciclo principal. Esto entonces indicará el tiempo libre entre interrupciones. El resultado es el mostrado en la figura 4.8:

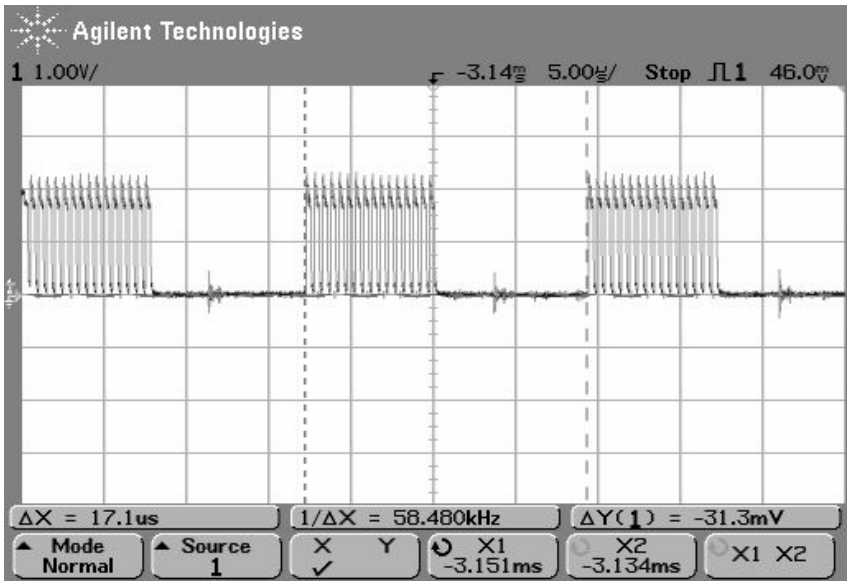


Figura 4.8: Ciclo principal.

Se observa que el ciclo principal se ejecuta en 17.1 μs, en que lo que no parpadea es el tiempo en que la interrupción se ejecuta. Como no hay traslape, ésta funciona sin problemas.

4.4 Comunicación Ethernet

La placa de desarrollo PIC WEB de Olimex, trae incorporado en su stack un código de ejemplo, el cual provee a la placa de un servidor web embebido y transmisión de la página web por FTP.

Además trae un conector RS232 para poder configurar los parámetros más importantes como son: la dirección física o MAC, el IP de la placa, el IP del Gateway y la máscara de subred. También permite el uso de DHCP si se quiere una asignación de la dirección IP de forma dinámica.

Cabe mencionar que para guardar una página web, el stack define un sistema de archivos denominado MPFS, el cual compila la página en un archivo único de extensión .bin. El menú principal de la placa se muestra en la figura 4.9:

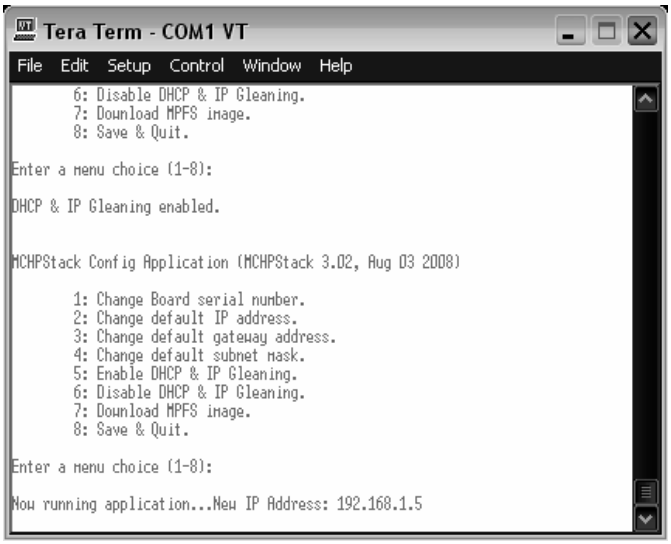


Figura 4.9: Configuración de parámetros de red.

Al conectar a la red esta placa surge una serie de negociaciones con el computador. Esto se puede ver con un sniffer de red tales como Ethereal o Wireshark, tal como se observa en la imagen 4.10:

No.	Time	Source	Destination	Protocol	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	DHCP Discover - Transaction ID 0x12233456
2	0.002188	192.168.1.1	255.255.255.255	DHCP	DHCP Offer - Transaction ID 0x12233456
3	0.015400	0.0.0.0	255.255.255.255	DHCP	DHCP Request - Transaction ID 0x12233456
4	0.017284	192.168.1.1	255.255.255.255	DHCP	DHCP ACK - Transaction ID 0x12233456
5	0.042143	192.168.1.4	192.168.1.255	UDP	Source port: 2860 Destination port: 30303

Figura 4.10: Negociación DHCP.

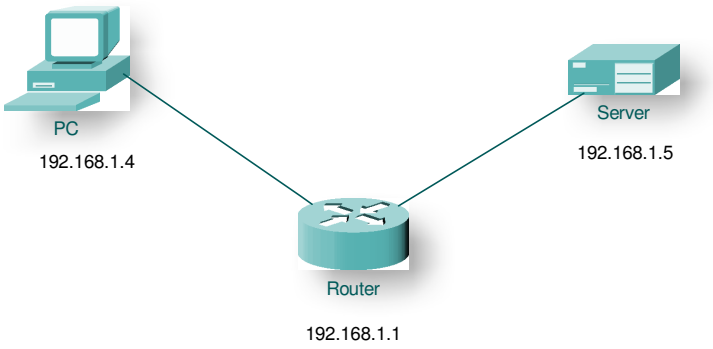


Figura 4.11: Red utilizada para pruebas.

Ahora se analizarán los resultados obtenidos tras simular la transmisión de paquetes en el escenario citado (figura 4.11) . La calidad de servicio depende de varios parámetros que a continuación se mencionan:

Ancho de banda

Determina la cantidad de información que puede circular por un medio físico de comunicación de datos por unidad de tiempo. Un mayor ancho de banda implica que, en un intervalo de tiempo dado, se pueda transmitir un volumen de información superior. El ancho de banda suele medirse en bits por segundo o en alguno de sus múltiplos (Kilo bits por segundo, Mega bits por segundo, etc.).

Latencia

Existen diferentes modos de medir la latencia de red, pero lo más habitual es calcularla como el tiempo necesario para que un paquete de información viaje desde la fuente hasta su destino. Ésta se mide en segundos o en alguno de sus submúltiplos (mili segundos, etc.).

Jitter

El jitter mide la tendencia de los paquetes a llegar en intervalos de tiempo regulares. Un jitter bajo indica que los tiempos entre llegadas consecutivas en una sesión son, más o menos, uniformes. Un jitter elevado indica que dichos tiempos son variables e irregulares. En este caso, para simplificar, el jitter es medido como el tiempo que transcurre entre dos llegadas consecutivas de paquetes.

Pérdidas

La capa de red no garantiza la entrega fiable de paquetes. Esto significa que, ante determinadas circunstancias como la congestión en la red, es posible que se pierdan paquetes. Aunque el nivel de red no ofrece garantía alguna en este sentido, la capa de transporte (en particular el protocolo TCP) permite ofrecer un servicio fiable orientado a conexión.

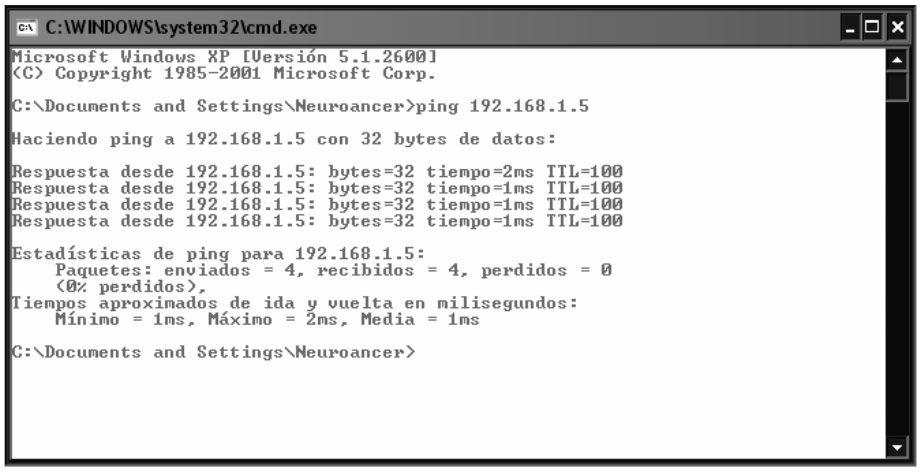


Figura 4.12: Resultados de PING.

No.	Time	Source	Destination	Protocol	Info
10	0.295166	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=1532 Ack=1 Win=1024 Len=512
11	0.295246	192.168.1.4	192.168.1.5	TCP	3427 > 54123 [ACK] Seq=1 Ack=2044 Win=64516 Len=0
12	0.368507	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=2044 Ack=1 Win=1024 Len=508
13	0.441146	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=2552 Ack=1 Win=1024 Len=512
14	0.441232	192.168.1.4	192.168.1.5	TCP	3427 > 54123 [ACK] Seq=1 Ack=3064 Win=65535 Len=0
15	0.514536	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=3064 Ack=1 Win=1024 Len=508
16	0.587174	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=3572 Ack=1 Win=1024 Len=512
17	0.587254	192.168.1.4	192.168.1.5	TCP	3427 > 54123 [ACK] Seq=1 Ack=4084 Win=64515 Len=0
18	0.660591	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=4084 Ack=1 Win=1024 Len=508
19	0.733229	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=4592 Ack=1 Win=1024 Len=512
20	0.733307	192.168.1.4	192.168.1.5	TCP	3427 > 54123 [ACK] Seq=1 Ack=5104 Win=65535 Len=0
21	0.806614	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=5104 Ack=1 Win=1024 Len=508
22	0.879257	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=5612 Ack=1 Win=1024 Len=512
23	0.879346	192.168.1.4	192.168.1.5	TCP	3427 > 54123 [ACK] Seq=1 Ack=6124 Win=64515 Len=0
24	0.952604	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=6124 Ack=1 Win=1024 Len=508
25	1.025289	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=6632 Ack=1 Win=1024 Len=512
26	1.025382	192.168.1.4	192.168.1.5	TCP	3427 > 54123 [ACK] Seq=1 Ack=7144 Win=65535 Len=0
27	1.098621	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=7144 Ack=1 Win=1024 Len=507
28	1.171265	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=7651 Ack=1 Win=1024 Len=512
29	1.171349	192.168.1.4	192.168.1.5	TCP	3427 > 54123 [ACK] Seq=1 Ack=8163 Win=64516 Len=0
30	1.244603	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=8163 Ack=1 Win=1024 Len=508
31	1.317289	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=8671 Ack=1 Win=1024 Len=512
32	1.317376	192.168.1.4	192.168.1.5	TCP	3427 > 54123 [ACK] Seq=1 Ack=9183 Win=65535 Len=0
33	1.390634	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=9183 Ack=1 Win=1024 Len=508
34	1.463280	192.168.1.5	192.168.1.4	TCP	54123 > 3427 [ACK] Seq=9691 Ack=1 Win=1024 Len=512

Tabla 4.1: Análisis de paquetes y handshacking TCP/IP mediante sniffer.

Mediante el sniffer de red Wireshark se realizó una captura de paquetes en el momento en que la información estaba siendo transferida al computador.

Para calcular el jitter, se tomaron los tiempos entre los paquetes 10 y 30 según la tabla 4.1 y se calculó la diferencia de tiempos para los paquetes de origen 192.168.1.5 ya que los paquetes con origen 192.168.1.4 correspondientes a los que envía el PC son la respuesta a que el paquete llegó bien. Entonces:

Nº	Tiempo	Origen	Destino	Diferencia de tiempo [s]
10	0,295166	192.168.1.5	192.168.1.4	
11	0,295246	192.168.1.4	192.168.1.5	0,00008
12	0,368507	192.168.1.5	192.168.1.4	0,07326
13	0,441146	192.168.1.5	192.168.1.4	0,07264
14	0,441232	192.168.1.4	192.168.1.5	0,00009
15	0,514536	192.168.1.5	192.168.1.4	0,07330
16	0,587174	192.168.1.5	192.168.1.4	0,07264
17	0,587254	192.168.1.4	192.168.1.5	0,00008
18	0,660591	192.168.1.5	192.168.1.4	0,07334
19	0,733229	192.168.1.5	192.168.1.4	0,07264
20	0,806614	192.168.1.4	192.168.1.5	0,07339
21	0,879257	192.168.1.5	192.168.1.4	0,07264
22	0,879257	192.168.1.5	192.168.1.4	0,00000
23	0,879346	192.168.1.4	192.168.1.5	0,00009
24	0,952604	192.168.1.5	192.168.1.4	0,07326
25	1,025382	192.168.1.5	192.168.1.4	0,07278
26	1,025382	192.168.1.4	192.168.1.5	0,00000
27	1,098621	192.168.1.5	192.168.1.4	0,07324
28	1,171265	192.168.1.5	192.168.1.4	0,07264
29	1,171349	192.168.1.4	192.168.1.5	0,00008
30	1,224603	192.168.1.5	192.168.1.4	0,05325

Tabla 4.2: Cálculo de jitter.

El ancho de banda está sujeto tanto a condiciones de congestión y característica de la red, entre otros. Entonces es posible calcular una estimación de éste respecto a los resultados dados por el sniffer.

Se consideran entonces los paquetes provenientes de 192.186.1.5 que contienen 512 datos, cada uno separado con un tiempo constante del dato anexo. Por lo tanto, el jitter no será entre datos sino que entre paquetes, de otra forma existe un jitter entre el último dato de un paquete y el primero del paquete siguiente.

El jitter entre paquetes será equivalente al promedio de la diferencia de llegada de paquetes, o sea, 0.073 seg.

Para calcular el ancho de banda para este caso, se puede tomar un espacio de tiempo y ver cuantos paquetes fueron enviados, lo cual está en directa relación con el número de bytes. Tomando desde el paquete 12 al 28 (cada uno con un campo de datos de 512 bytes).

Entonces el ancho de banda será:

$$Ancho\_de\_banda = \frac{512 \cdot 10}{1.171265 - 0.368507} = \frac{5120}{0.802578}[bytes/seg] = 6.378[Kbyte/seg]$$

Se muestra a continuación el detalle de un paquete capturado mediante Wireshark (figura 4.13). Al final del análisis se menciona la longitud del paquete de datos como 512 bytes.

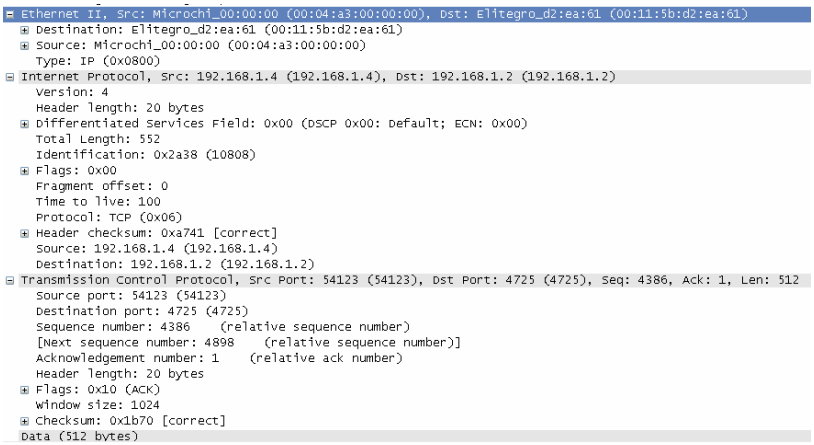


Figura 4.13: Estructura de un paquete TCP.

4.5 Puesta en marcha

Para poner a prueba el funcionamiento del osciloscopio, se ingresó la señal de salida del 555 tanto a un osciloscopio agilent como al dispositivo diseñado. A continuación se muestran las comparaciones de resultados:

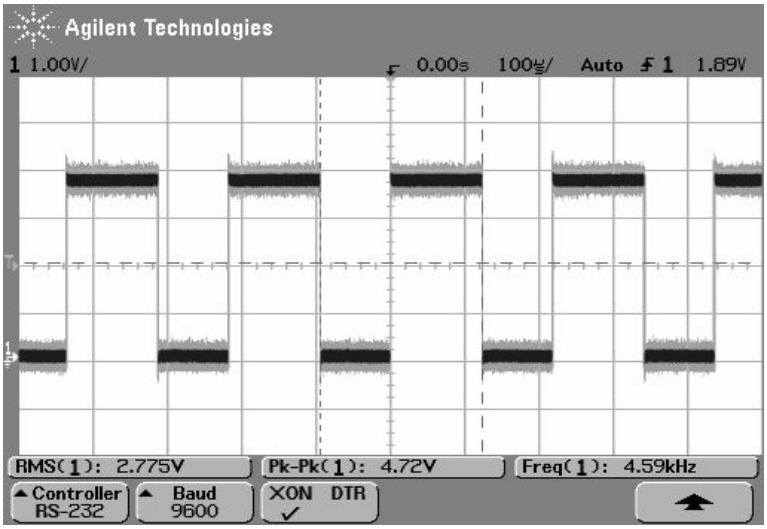


Figura 4.14: Tren de pulsos de 4.59 kHz y 2.775 Vrms

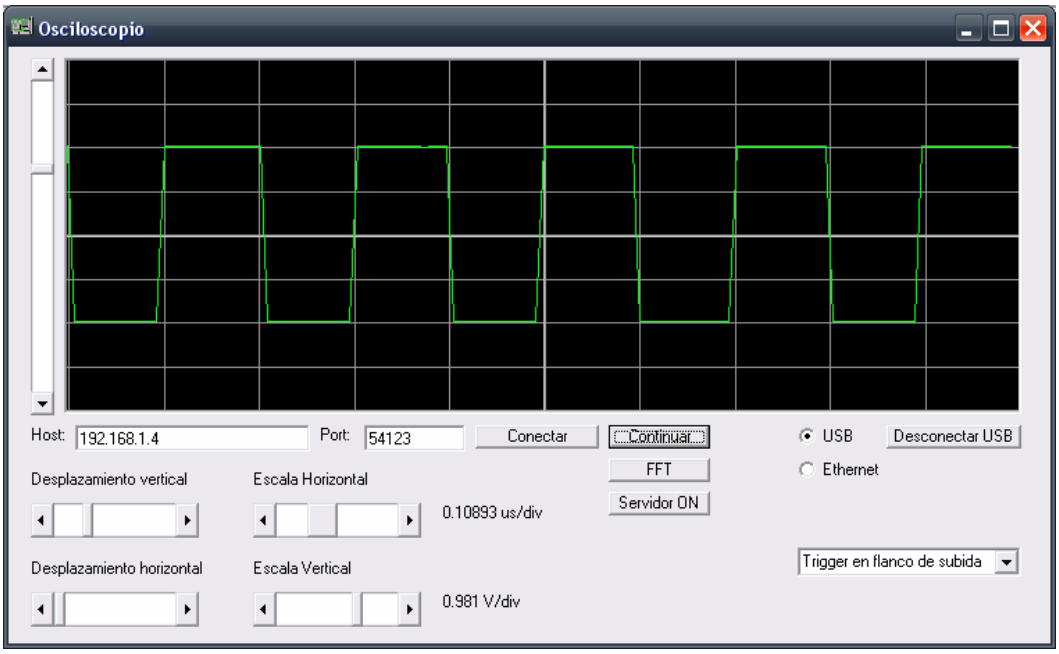


Figura 4.15: Equivalente en el equipo implementado

Para comprobar la presicion de los datos obtenidos en el dispositivo, en la figura 4.15 se observa que el scrollbar de deflexión horizontal esta ajustado en 0.10893 ms/div y el vertical en 0.981 V/div, con lo cual indirectamente es posible calcular frecuencia y voltaje RMS. Cabe mencionar que al hacer estos ajustes con los scrollbar, un ciclo completo del tren de pulsos quedo ajustado en un espacio de 4 divisiones verticales por 2 horizontales. Por lo tanto, el periodo del tren será de:

$$2[div] \cdot 0.10893[ms/div] = 0.21786[ms]$$

La frecuencia es el inverso del periodo, con lo cual se obtiene:

$$f = \frac{1}{0.21786 \cdot 10^{-3}} = 4.5897 \cdot 10^{3}[Hz]$$

En cuanto a la deflexión vertical, al ser 4 divisiones en este ejemplo, se tiene que la amplitud peak to peak del tren de pulsos se calcula como:

$$4[div] \cdot 0.981[V/div] = 3.99[Vpp]$$

A partir de este resultado, entonces el voltaje RMS es:

$$V_{RMS} = V_{PP} \cdot 0.707 = 3.99 \cdot 0.707 = 2{,}82093[\mathrm{V}]$$

Lo cual se asemeja bastante a la información dada en la figura 4.14

CAPITULO 5: CONCLUSIONES

En este trabajo de título se presentó el diseño de un sistema de adquisición de datos y posterior procesamiento y gráfica de éstos en forma de un osciloscopio digital básico.

Se buscó principalmente un sistema de envío y recepción de datos basados en el bus USB y la transmisión de datos por Ethernet.

El sistema fue diseñado con un enfoque modular, es decir, que si en un futuro alguien necesita desarrollar una aplicación sólo por USB, se pueda sacar este bloque sin inconvenientes y poder adaptarlo a otros contextos, así también con el módulo Ethernet.

En cuanto al software, al ser modelado en forma de capas, se puede utilizar con una amplia variedad de recepción de datos, sólo será necesario cambiar la capa de recepción. Por ejemplo, además de USB y Ethernet, también se probó con RS232.

Respecto a la metodología, se puede decir que, a pesar de haber elaborado una carta Gantt con la planificación del proyecto, de la teoría a la práctica hay una gran diferencia. Por ejemplo, el hecho de aprender nuevos lenguajes de programación y el funcionamiento de USB.

Los principales problemas en el diseño fueron lograr la conectividad USB mediante la API de Microchip, y que no existiera pérdida de información en el traspaso del dato convertido, limitándose a las restricciones del canal de comunicación.

Como trabajo futuro se sugiere modificar la API de Microchip para que soporte más de 1 pipe, la idea es recolectar un mayor número de muestras por paquete, de esta manera se reduciría considerablemente el tráfico perdido en encabezados.

Por otro lado, se implementó un trigger por software. Se podría optimizar el mecanismo de disparo, implementando éste por firmware y enviando exclusivamente los datos que serán graficados y no una trama en que se descartarán datos.

También, en la fase de acondicionamiento, el sistema de amplificación de ganancia programable puede reducirse en componentes, directamente usando un potenciómetro digital para modificar la ganancia, tal como el X9C102P.

Otra cosa importante de mencionar, es que hubiese sido ideal implementar de cero el hardware Ethernet, pero lamentablemente por problemas de importación no se pudo conseguir el controlador ENC28J60, por lo que se trabajó en base a la placa de desarrollo PIC-WEB de Olimex, la cual contiene el controlador y el conector RJ45 incluidos.

Finalmente se adjunta la hoja de datos del dispositivo diseñado:

Sistema de deflexión vertical

Entrada	1 canal de entrada
Impedancia de entrada	1 MOhm/5 pF
Rango del factor de deflexión	62.5 mV/div a 6.25 V/div
Precisión	+- 1 %
Resolución	8 bits
Tensión máxima de entrada	+- 220 V (con etapa de acondicionamiento) 5 V (sin fase de acondicionamiento)

Sistema de deflexión horizontal

Rango de la base de tiempos	149.76 µs/div a 599.04 us/div
Frecuencias de muestreo	40 kHz

Disparo

Fuente de disparo	Software
Selección del umbral	Scrollbar
Opciones de disparo	- Disparo por flanco ascendente o descendente - Tras un número definido de muestras después de la condición de disparo (Filtro digital)
Tamaño de almacenamiento	256 muestras

Requerimientos del sistema

Computador	Procesador Pentium con puerto usb 1.1 o superior
Sistema operativo	Windows 98 / Windows 2000 / Windows XP

Dimensiones

Placa USB	6x7.5 cm.
Placa Ethernet	6x6.5 cm.

BIBLIOGRAFIA

Osciloscopios

[1] Técnica de medidas con osciloscopio: principios y aplicaciones de los modernos osciloscopios de rayos catódicos /J.
Czech; trad. por Miguel Molinos Calvo. Czech, J.
U.CHILE Fac. de Ciencias 621.38 C998

[2] Typical oscilloscope circuitry / Tektronix, Inc.
Datos de Publicación : Beaverton, Or. : Tektronix, Inc., 1961.
U.CHILE Ingeniería Eléctrica 621.317. 7:621.38 T267

[3] Ruiter, Jacob H.
Modern oscilloscopes and their uses / by Jacob H. Ruiter.
U.CHILE Fac. de Ciencias 621.34 R934

[4] Jose M. Drake, Laboratorio de instrumentación. Departamento de electrónica y computadores – Universidad de Cantabria – Octubre 2004

[5] Sergi Bernat Jordana de Buen, "Unidad procesadora de señales para osciloscopio" Memoria de Ingeniero Civil Electricista.
Santiago: Depto. de Ingeniería Eléctrica, Universidad de Chile, 1981.

[6] Osciloscopios analógicos y digitales
https://www.cs.tcd.ie/courses/baict/bac/jf/labs/scope/oscilloscope.html [consulta: 1 marzo 2008]

Conversión Análogo/Digital

[7] Microchip, 2001 conversores análogo/digital
http://ww1.microchip.com/downloads/en/devicedoc/adc.pdf

[8] Steven W. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing" – 2º edition – California Technical
Publishing – San Diego, California 1997-1999
www.dspguide.com

Acondicionamiento de la señal

[9] Pablo Estévez – Apuntes de Análisis de redes II

[10] Ron Mancini, Editor in chief – Op Amps for Everyone –Texas Instruments 2002

[11] National Semiconductor - Op Amp Circuit Collection, Application note 31 Feb. 1978

USB

[12] Especificación USB 2.0
http://www.usb.org/developers/docs/

[13] BeyondLogic,2001-2007 USB in a nutshell
http://www.beyondlogic.org/usbnutshell/usb1.htm

Datasheets

[14] Microchip, 2006 Datasheet PIC18F2550
http://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf

[15] Fairchild, 2002 Datasheet LM324
http://www.datasheetcatalog.org/datasheet/fairchild/LM324.pdf

[16] Texas instruments, 1999 Datasheet Multiplexor CMOS4051
http://www.datasheetcatalog.org/datasheets/120/107571_DS.pdf

[17] Microchip, 2006 Datasheet PIC18F452
http://ww1.microchip.com/downloads/en/DeviceDoc/39564c.pdf

[18] Microchip, 2006 Datasheet ENC28J60
http://ww1.microchip.com/downloads/en/DeviceDoc/39662b.pdf

Ethernet

[19] IEEE standard 802.3 (Ethernet)
http://standards.ieee.org/getieee802/802.3.html

[20] Microchip TCP/IP Stack, AN833
http://ww1.microchip.com/downloads/en/AppNotes/00833b.pdf

[21] Microchip, 2006 Ethernet theory of operation – AN1120
http://ww1.microchip.com/downloads/en/AppNotes/01120a.pdf

[22] C. Saavedra, "Implementación del protocolo HTTP en un microcontrolador PIC". Memoria de Ingeniero Civil Electricista.
Santiago: Depto de Ingeniería Eléctrica, Universidad de Chile, 2003. T 2003 Sa12i c.2

[23] Tanenbaum, Andrew S., 1944-
Redes de computadoras / Andrew S. Tanenbaum; traducción David Morales P.; revisión técnica Gabriel Guerrero Reyes
Ingeniería Bca. Central 004.6 T155r.E

[24] Ray, John
Edicion especial TCP/IP / John Ray.
U.CHILE Ingeniería Bca. Central 004.62 R211 1999

[25] Casad, Joe
TCP/IP in 24 hours / Joe Casad, Bob Willsey.
U.CHILE Fac. de Ciencias 004.62 C334

[26] Comer, Douglas.
Internetworking with TCP/IP / Douglas E. Comer.
U.CHILE Ingeniería Bca. Central 005.2 C734 1995 V.1

Lenguajes

[27] Microchip, 2004 MPLAB C18 C Compiler Getting Started
ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_C18_Getting_Started_51295d.pdf

[28] Custom Computer Services (CCS), 2001 - C Compiler reference manual.

[29] Ceballos Sierra, Fco. Javier (Francisco Javier)
Enciclopedia de Microsoft Visual Basic 6 / Fco. Javier Ceballos Sierra.
U.CHILE Ingeniería Bca. Central 005.133ViB C321 2000

ANEXOS

A.1 FIRMWARE USB

```
//se incluyen las librerías para el PIC utilizado
#include <18F2550.h>

//se configuran los fuses para que a partir de un cristal de 12 MHz se
//tenga un clock de 48 MHz mediante preescalers
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL3,CPUDIV1,VREGEN

//definición del ADC operando a 8 bits y el clock final a 48 MHz
#device ADC=8
#use delay(clock=48000000)

#define bytes_a_enviar 64

//acá se define el modo de transferencia BULK o masivo, deshabilitando el
//modo de transferencia HID y definiendo el tamaño del paquete que se
//enviara por USB
#define USB_HID_DEVICE     FALSE
#define USB_EP1_TX_ENABLE  USB_ENABLE_BULK
#define USB_EP1_RX_ENABLE  USB_ENABLE_BULK
#define USB_EP1_TX_SIZE    bytes_a_enviar
#define USB_EP1_RX_SIZE    1

//se incluyen tanto la capa física dada por Microchip como los
//descriptores del dispositivo

#include <pic18_usb.h>
#include <OsciloscopioUSB.h>
#include <usb.c>

#define LED0    PIN_C0
#define LED1    PIN_C1
#define LED2    PIN_C2
#define LED_ON  output_high
#define LED_OFF output_low


//el array envía se define para guardar el dato de acuerdo a los bytes
que
//se quieran enviar
int8 dato;
int8 listo_para_mandar;
int8 crear_paquete;
int8 envia[4][bytes_a_enviar];


int8 k;
int8 j;
int8 comando;
void iniciarUSB(void);
```

```
#INT_AD
AD_isr(){
   dato=Read_ADC(ADC_READ_ONLY);
   output_b(dato);

   if(crear_paquete==1){
    if(k<64){
      envia[j][k]=dato;
      k++;
    }
    else{
      k=1;
      if(j<4){
         j++;
         listo_para_mandar=0;
      }
      else{
         j=0;
         listo_para_mandar=1;
      }
    }
   }
   Read_ADC(ADC_START_ONLY);
}

//empieza el main
void main(void) {
   iniciarUSB();

   Do
   {
   if (listo_para_mandar==1 && usb_enumerated()){
      if(usb_kbhit(1)){
       usb_get_packet(1,comando,1);
       if (comando==1){
         usb_put_packet(1, envia[0], bytes_a_enviar, USB_DTS_TOGGLE);
         crear_paquete=0;
       }
       if (comando==2){
         usb_put_packet(1, envia[1], bytes_a_enviar, USB_DTS_TOGGLE);
         crear_paquete=0;
       }
       if (comando==3){
         usb_put_packet(1, envia[2], bytes_a_enviar, USB_DTS_TOGGLE);
         crear_paquete=0;
       }
       if (comando==4){
         usb_put_packet(1, envia[3], bytes_a_enviar, USB_DTS_TOGGLE);
         crear_paquete=1;
       }


      }
   }

   }while(1);
}
```

```
void iniciarUSB(void){

  //se configura el puerto AN 0 como ADC en vez de GPIO
  //y se activan las interrupciones del ADC
  setup_adc (ADC_CLOCK_DIV_64);
  setup_adc_ports (AN0);
  enable_interrupts(INT_AD);
  enable_interrupts(GLOBAL);
  set_adc_channel (0);

   //inicializa el contador
   k=1;
   j=0;
   listo_para_mandar=0;
   crear_paquete=1;

   Read_ADC(ADC_START_ONLY);

   //tambien en el circuito se definen 3 leds auxiliares (rojo,amarillo y
   //verde) tal de que muestren en que etapa de configuracion del USB
dentro
   //del computador esta,ademas de definir LED_ON y LED_OFF para apagar o
   //encender los leds

   //se inicializa el dispositivo USB y enciende el led rojo
   usb_init();
      LED_ON(LED0);
      LED_OFF(LED1);
      LED_OFF(LED2);

   //habilita el periferico y sus interrupciones, ademas de apagar el led
   //rojo y encender el amarillo
   usb_task();
      LED_OFF(LED0);
      LED_ON(LED1);
      LED_OFF(LED2);

   //se espera a que el dispositivo este finalmente reconocido por el
   //computador y ennumerado dentro del resto de los dispositivos USB
   //ademas de apagar el led amarillo y encender el verde
   usb_wait_for_enumeration();
      LED_OFF(LED0);
      LED_OFF(LED1);
      LED_ON(LED2);
}
```

A.2 SOFTWARE

A.2.1 VENTANA PRINCIPAL (FORM1.FRM)

```
Option Explicit

Dim Host As String
Dim Puerto As String

Dim datorecibido As String
Dim dato As Integer
Dim NoTeCuelgues As Integer

Dim j, k As Integer

Dim contFFT As Integer

Private Sub Command1_Click()
  CloseMPUSBDevice
End Sub


Private Sub HScroll2_Change()
Label_V = 500 / (8 * HScroll2.Value) & " V/div"
End Sub

Private Sub timer1_timer()
    If Option1.Value = True Then
     If BotonPausa.Caption = "Pausa" Then

            recibe
            Seleccion_Trigger
            dibuja (ventana)

            If (Winsock2.State = sckConnected) Then
             Winsock2.SendData strDatos
            End If

            If Form2.Visible = True Then
                FFT (ventana)
            End If

     End If
    End If
End Sub

Private Sub BotonFFT_Click()
    If BotonFFT.Caption = "FFT" Then
        BotonFFT.Caption = "Cerrar FFT"
        Form2.Show
    Else
        BotonFFT.Caption = "FFT"
        Form2.Hide
    End If
End Sub
```

```
Private Sub Form_Load()

    OpenMPUSBDevice
    Inicializa_Trigger
    n = 512
    contFFT = 0
    ReDim ventana(256) As Integer

    Label_V = 500 / (8 * HScroll2.Value) & " V/div"
    Label_H.Caption = HScroll1.Value * 83.2 * 9 / 5 & " us/div"
End Sub


Private Sub Inicializa_Trigger()
    Combo1.Clear
    Combo1.AddItem "Sin Trigger"
    Combo1.AddItem "Trigger en flanco de subida"
    Combo1.AddItem "Trigger en flanco de bajada"
    Combo1.Text = "Sin Trigger"
    encontrotrigger = False
End Sub

Private Sub BotonPausa_Click()
    If BotonPausa.Caption = "Pausa" Then
        BotonPausa.Caption = "Continuar"
    Else
        BotonPausa.Caption = "Pausa"
    End If
End Sub


Private Sub HScroll1_Change()
    n = HScroll1.Value * 64
    ReDim ventana(n) As Integer
    Label_H.Caption = HScroll1.Value * 93 & " us/div" '83.2 us por
paquete
End Sub

Private Sub Command2_Click()

    Host = Text3.Text
    Puerto = Text4.Text
    j = 0
    k = 0
    If Command2.Caption = "Conectar" Then
        With Winsock1
            .Close
            .RemoteHost = Host
            .RemotePort = Puerto
            .Connect
        End With
        Command2.Caption = "Desconectar"
    Else
    Winsock1.Close
    Command2.Caption = "Conectar"
    End If
```

```
End Sub

Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
    On Error Resume Next


    Winsock1.GetData strDatos
    If BotonPausa.Caption = "Pausa" Then


            Seleccion_Trigger
            dibuja (ventana)

            If Form2.Visible = True Then
                FFT (ventana)
            End If

     End If
End Sub

Private Sub Command3_Click()

    If Command3.Caption = "Servidor ON" Then
        Command3.Caption = "Servidor OFF"

        With Winsock2
            .Close 'Si Winsock1 esta abierto, lo cierra
            .LocalPort = 4444 'puerto del servidor para la comunicación
            .Listen 'a la escucha de posibles envíos de datos
        End With

    Else
        Command3.Caption = "Servidor ON"
        Winsock2.Close

    End If




End Sub

Private Sub Winsock2_ConnectionRequest(ByVal requestID As Long)
 Winsock2.Close 'cierra el socket si esta abierto
 Winsock2.Accept (requestID) 'acepta la conexión de la antena
End Sub
```

A.2.2 FUNCIONES USB

```
Attribute VB_Name = "USB"
Option Explicit

Public Declare Function MPUSBGetDLLVersion Lib "mpusbapi.dll" () As Long
Public Declare Function MPUSBGetDeviceCount Lib "mpusbapi.dll" (ByVal
pVID_PID As String) As Long

Public Declare Function MPUSBOpen Lib "mpusbapi.dll" (ByVal instance As
Long, ByVal pVID_PID As String, ByVal pEP As String, ByVal dwDir As Long,
ByVal dwReserved As Long) As Long

Public Declare Function MPUSBClose Lib "mpusbapi.dll" (ByVal handle As
Long) As Long

Public Declare Function MPUSBRead Lib "mpusbapi.dll" (ByVal handle As
Long, ByVal pData As Long, ByVal dwLen As Long, ByRef pLength As Long,
ByVal dwMilliseconds As Long) As Long

Public Declare Function MPUSBWrite Lib "mpusbapi.dll" (ByVal handle As
Long, ByVal pData As Long, ByVal dwLen As Long, ByRef pLength As Long,
ByVal dwMilliseconds As Long) As Long

Public Declare Function MPUSBReadInt Lib "mpusbapi.dll" (ByVal handle As
Long, ByVal pData As Long, ByVal dwLen As Long, ByRef pLength As Long,
ByVal dwMilliseconds As Long) As Long

Public Const INVALID_HANDLE_VALUE = -1
Public Const ERROR_INVALID_HANDLE = 6&

Public Declare Function GetLastError Lib "kernel32" () As Long


Public Const vid_pid = "vid_04d8&pid_0011"
Public Const out_pipe = "\MCHP_EP1" 'We don't want two \\ in VB... \\ is
for C
Public Const in_pipe = "\MCHP_EP1"

Public Const MPUSB_FAIL = 0
Public Const MPUSB_SUCCESS = 1

Public Const MP_WRITE = 0
Public Const MP_READ = 1

'Declare the IN PIPE and OUT PIPE Public variables
Public myInPipe As Long, myOutPipe As Long


Sub Initialize()
myInPipe = INVALID_HANDLE_VALUE
myOutPipe = INVALID_HANDLE_VALUE
End Sub

Sub OpenMPUSBDevice()
Dim tempPipe As Long
```

```
Dim count As Long

tempPipe = INVALID_HANDLE_VALUE
count = MPUSBGetDeviceCount(vid_pid)

If count > 0 Then
    myOutPipe = MPUSBOpen(0, vid_pid, out_pipe, MP_WRITE, 0)
    myInPipe = MPUSBOpen(0, vid_pid, in_pipe, MP_READ, 0)
   If myOutPipe = INVALID_HANDLE_VALUE Or myInPipe = INVALID_HANDLE_VALUE
Then
        MsgBox Str(myOutPipe) + Str(myInPipe) + "Error al abrir pipes"
        myOutPipe = myInPipe = INVALID_HANDLE_VALUE

    End If
Else
   MsgBox "El dispositivo no esta conectado"
End If

End Sub

Sub CloseMPUSBDevice()
If myOutPipe <> INVALID_HANDLE_VALUE Then
    MPUSBClose (myOutPipe)
    myOutPipe = INVALID_HANDLE_VALUE
End If
If myInPipe <> INVALID_HANDLE_VALUE Then
    MPUSBClose (myInPipe)
    myInPipe = INVALID_HANDLE_VALUE
End If
End Sub


Function SendReceivePacket(ByRef SendData() As Byte, SendLength As Long,
_
                           ByRef    ReceiveData()    As    Byte,    ByRef
ReceiveLength As Long, _
                           ByVal SendDelay As Long, ByVal ReceiveDelay As
Long) As Long


    Dim SentDataLength As Long
    Dim ExpectedReceiveLength As Long

    ExpectedReceiveLength = ReceiveLength

    If    (myOutPipe    <>    INVALID_HANDLE_VALUE    And    myInPipe    <>
INVALID_HANDLE_VALUE) Then

        If    (MPUSBWrite(myOutPipe,    VarPtr(SendData(0)),    SendLength,
SentDataLength, SendDelay) = MPUSB_SUCCESS) Then

            If        (MPUSBRead(myInPipe,        VarPtr(ReceiveData(0)),
ExpectedReceiveLength, ReceiveLength, ReceiveDelay) = MPUSB_SUCCESS) Then

                If (ReceiveLength = ExpectedReceiveLength) Then
                   SendReceivePacket = 1    '// Success!
                   Exit Function
```

```
                ElseIf (ReceiveLength < ExpectedReceiveLength) Then
                    SendReceivePacket  =  2      '//  Partially  failed,
incorrect receive length
                    Exit Function
                End If

            Else
                CheckInvalidHandle
            End If
        Else
            CheckInvalidHandle
        End If
    End If

    SendReceivePacket = 0  '// Operation Failed

End Function

Sub CheckInvalidHandle()
 If (GetLastError() = ERROR_INVALID_HANDLE) Then


        CloseMPUSBDevice
 Else
        'MsgBox "Error Code : " + Str(GetLastError())
 End If
End Sub
```

A.2.3 ADQUISICION DE DATOS

```
Option Explicit

Global tx_Buf(1) As Byte
Global rx_Buf(64) As Byte
Global rx_length As Long
Global tx_length As Long

Global strDatos As String

Dim i, k As Integer

Public Function recibe()
   tx_length = 1
   rx_length = 64
   strDatos = ""

    tx_Buf(0) = 0
    If  (SendReceivePacket(tx_Buf,  1,  rx_Buf,  rx_length,  100,  100)  =  1)
Then
    End If

     For k = 0 To UBound(rx_Buf) - 1
      strDatos = strDatos & Chr(rx_Buf(k))
     Next k
```

```
    tx_Buf(0) = 1
    If (SendReceivePacket(tx_Buf, 1, rx_Buf, rx_length, 100, 100) = 1)
Then
    End If

     For k = 0 To UBound(rx_Buf) - 1
      strDatos = strDatos & Chr(rx_Buf(k))
     Next k

    tx_Buf(0) = 2
    If (SendReceivePacket(tx_Buf, 1, rx_Buf, rx_length, 100, 100) = 1)
Then
    End If

     For k = 0 To UBound(rx_Buf) - 1
      strDatos = strDatos & Chr(rx_Buf(k))
     Next k

    tx_Buf(0) = 3
    If (SendReceivePacket(tx_Buf, 1, rx_Buf, rx_length, 100, 100) = 1)
Then
    End If

     For k = 0 To UBound(rx_Buf) - 1
      strDatos = strDatos & Chr(rx_Buf(k))
     Next k
End Function
```

A.2.4 GRAFICOS

```
Option Explicit

Dim x0, y0, x1, y1 As Integer
Dim Max_x, Max_y As Integer
Dim AjusteV, AjusteH As Double
Dim Desp_v As Integer
Dim Escala_V, Escala_H As Double

Dim i As Integer

Global ventana As Variant
Global n As Integer

Public Function dibuja(ventana As Variant)             'Aplicado sobre
Form1.Picture1

    n = UBound(ventana)

    Max_x = Form1.Picture1.ScaleWidth
    Max_y = Form1.Picture1.ScaleHeight

    Desp_v = Form1.HScroll3.Value

    Escala_V = Form1.HScroll2.Value / 100
    Escala_H = Max_x / n
```

```
    Form1.Picture1.Cls

    dibuja_grilla1

    Form1.Picture1.ForeColor = RGB(0, 255, 0)

    x0 = 0
    y0 = Max_y - Escala_V * ventana(0) - Desp_v

    For i = 0 To UBound(ventana) - 1
        x1 = Escala_H * i
        y1 = Max_y - Escala_V * ventana(i) - Desp_v
        Form1.Picture1.Line (x0, y0)-(x1, y1)
        x0 = x1
        y0 = y1
    Next i

End Function


Public Function dibujaFFT()

    Form2.Picture2.Cls

    dibuja_grilla2

    Form2.Picture2.ForeColor = RGB(255, 0, 0)

    AjusteH = Form2.Picture2.ScaleWidth / UBound(ResultadoFFT)
    AjusteV = (0.05 * Form2.Picture2.ScaleHeight) / 350

    x0 = 0
    y0 = Form2.Picture2.ScaleHeight - AjusteV * ResultadoFFT(0)

    For i = 0 To UBound(ResultadoFFT)
        x1 = AjusteH * i
        y1 = Form2.Picture2.ScaleHeight - AjusteV * ResultadoFFT(i)
        Form2.Picture2.Line (x0, y0)-(x1, y1)
        x0 = x1
        y0 = y1
    Next
End Function

Public Function dibuja_grilla1()

    Max_x = Form1.Picture1.ScaleWidth
    Max_y = Form1.Picture1.ScaleHeight

    Form1.Picture1.ForeColor = RGB(150, 150, 150)

    For i = 1 To Max_y Step Max_y / 8
     Form1.Picture1.Line (0, Max_y - i)-(Max_x, Max_y - i)
    Next i

    For i = 1 To Max_x Step Max_x / 10
     Form1.Picture1.Line (i, 0)-(i, Max_y)
```

```
    Next i

    Form1.Picture1.ForeColor = RGB(200, 200, 200)
    Form1.Picture1.Line (0, Max_y / 2)-(Max_x, Max_y / 2)
    Form1.Picture1.Line (Max_x / 2, 0)-(Max_x / 2, Max_y)

End Function

Public Function dibuja_grilla2()

    Max_x = Form2.Picture2.ScaleWidth
    Max_y = Form2.Picture2.ScaleHeight

    Form2.Picture2.ForeColor = RGB(50, 50, 50)

    For i = 1 To Max_y Step 20
     Form2.Picture2.Line (0, Max_y - i)-(Max_x, Max_y - i)
    Next i

    For i = 1 To Max_x Step 20
     Form2.Picture2.Line (i, 0)-(i, Max_y)
    Next i

End Function
```

A.2.5 TRIGGER

```
Option Explicit

Global encontrotrigger, triggerchk As Boolean
Dim mintrigger, maxtrigger As Integer
Dim j, flanco, centrotrigger As Integer
Dim strVentana, Dato_Actual, Dato_Anterior As String


Public Function Seleccion_Trigger()
    If Form1.Combo1.Text = "Sin Trigger" Then
        Sin_Trigger

    Else
     If Form1.Combo1.Text = "Trigger en flanco de subida" Then
        flanco = 20
     End If

     If Form1.Combo1.Text = "Trigger en flanco de bajada" Then
        flanco = -20
     End If
     Busca_Trigger
    End If
End Function


Public Function Busca_Trigger()
On Error Resume Next
```

```
    encontrotrigger = False
    mintrigger = Form1.VScroll1.Value - 10
    maxtrigger = Form1.VScroll1.Value + 10

    If Form1.VScroll1.Value < 10 Then
      mintrigger = 0
    End If

    If Form1.VScroll1.Value > 245 Then
     maxtrigger = 255
    End If
    '*************************

    j = 1
    centrotrigger = n / 2 + Form1.HScroll4.Value


    Do While encontrotrigger = False And j < n / 2

        Dato_Actual = Mid(strDatos, centrotrigger + j, 1)
        Dato_Anterior = Mid(strDatos, centrotrigger + j - 1, 1)


        If  Asc(Dato_Actual)  >  mintrigger  And  (Asc(Dato_Actual)  -
Asc(Dato_Anterior) > flanco) Then
            encontrotrigger = True 'sale del while
            Exit Do
        Else
            encontrotrigger = False
            Dato_Anterior = Dato_Actual
        End If
        j = j + 1
     Loop

     strVentana = Mid(strDatos, j, n)


     For j = 1 To UBound(ventana)
        ventana(j - 1) = Asc(Mid(strVentana, j, j + 1))
     Next j

End Function


Public Function Sin_Trigger()
On Error Resume Next
    Dim i As Integer
    For i = 1 To UBound(ventana)
        ventana(i - 1) = Asc(Mid(strDatos, i, i + 1))
    Next i
End Function
```

A.2.6 ANALISIS DE FOURIER

```
Option Explicit

Dim Rex(512), Imx(512) As Integer
Dim PI As Double
Dim n As Integer
Dim nm1, nd2 As Integer
Dim i, j, k, l As Integer
Dim tr, ti As Double
Dim ur, ui, sr, si As Double
Dim m As Integer
Dim le, le2 As Integer
Dim ip As Integer


Public Sub FFT()
    PI = 3.14159265
    nm1 = n - 1
    nd2 = n / 2
    j = nd2
    m = Log(n) / Log(2)

    ' Bit reversal
    For i = 1 To n
        If i < j Then
            tr = Rex(j)
            ti = Imx(j)
            Rex(j) = Rex(i)
            Imx(j) = Imx(i)
            Rex(i) = tr
            Imx(i) = ti
        End If

        k = nd2

        While ((k <= j) And (k > 0))
            j = j - k
            k = k / 2
        Wend
        j = j + k
    Next i
    'fin Bit reversal

    For l = 1 To m
     le = 2 ^ l
     le2 = le / 2
     ur = 1
     ui = 0
     sr = Cos(PI / le2)
     si = -Sin(PI / le2)

     For j = 1 To le2
        For i = j - 1 To nm1 Step le
            ip = i + le2
            tr = Rex(ip) * ur - Imx(ip) * ui
```

```
            ti = Rex(ip) * ui + Imx(ip) * ur
            Rex(ip) = Rex(i) - tr
            Imx(ip) = Imx(i) - ti
            Rex(i) = Rex(i) + tr
            Imx(i) = Imx(i) + ti
        Next i

        tr = ur
        ur = tr * sr - ui * si
        ui = tr * si + ui * sr
     Next j
    Next l
End Sub
```

A.2.7 GUARDAR IMAGEN

```
Option Explicit

Dim replace
Dim Imagen As String

Public Function Guardar_Imagen()

   On Error Resume Next


   Form1.CommonDialog1.CancelError = True
   Form1.CommonDialog1.DialogTitle = "Ingrese el nombre de destino"
   Form1.CommonDialog1.Filter  =  "Archivo  BMP  (*.bmp)|*.bmp|Todos  los
archivos (*.*)|*.*"
   Form1.CommonDialog1.FileName = "imagen.bmp"
   Form1.CommonDialog1.ShowSave

   If Err = cdlCancel Then Exit Function
      Imagen = Form1.CommonDialog1.FileName

      If Err Then
         MsgBox Error$, 48
         Exit Function
      End If

   ' Guardar la imagen de la señal
   SavePicture Form1.Picture1.Image, Imagen

   Form1.CommonDialog1.CancelError = True
   Form1.CommonDialog1.DialogTitle = "Ingrese el nombre de destino"
   Form1.CommonDialog1.Filter  =  "Archivo  BMP  (*.bmp)|*.bmp|Todos  los
archivos (*.*)|*.*"
   Form1.CommonDialog1.FileName = "imagen.bmp"
   Form1.CommonDialog1.ShowSave

   If Err = cdlCancel Then Exit Function
      Imagen = Form1.CommonDialog1.FileName

      If Err Then
```

```
         MsgBox Error$, 48
         Exit Function
      End If

   ' Guardar la imagen de la FFT
   'SavePicture Form1.Picture2.Image, Imagen

End Function
```