UNIVERSIDAD DE ORIENTE
Fundada en 1947
Santiago de Cuba, República de Cuba

Santiago de Cuba

Facultad de Ingeniería Eléctrica

Departamento de Ingeniería Automática

TÍTULO: Sistema de adquisición de datos para monitoreo en pequeñas centrales hidroeléctricas.

Tesis en opción al grado de Ingeniero en Automática

AUTOR: *Luis Emilio Rosario Navarrete*

TUTOR: *MSc. Ing. Julio Fong Barrio.*

Curso 2008-2009

Pensamiento:

De la forma en que el hombre utilice las energías de nuestro planeta, dependerá el futuro del mismo.

Luis Emilio Rosario Navarrete.

Dedicatoria:

Este trabajo va dedicado a todas las personas que de una manera u otra me han ayudado a lograr a realizar mis estudios.

Agradecimientos:

A mis padres que fueron los pilares fundamentales para que yo lograra un título universitario, a mi tutor Msc. Juilio Fong Barrio por la ayuda prestada durante el dasarrollo de la tésis, al Msc. José Antonio Pullés que no escatima a la hora de ayudar a un alumno, a mi tía Alicia Navarrete que sin su ayuda este trabajo no hubiese sido posible y a todos los profesores del Departamento de Ingeniería Automática.

Resumen

Este trabajo forma parte de los estudios que viene realizando la Facultad de Ingeniería Eléctrica para el desarrollo de energías renovables.El objetivo del mismo es diseñar un sistema de adquisición de datos que permita el monitoreo de parámetros eléctricos en pequeñas centrales hidroeléctricas, parámetros que serán medidos y procesadas por un microcontrolador PIC18F4520 y transmitidas para su supervisión en una PC , utilizando el LabWindows/CVI de la *National Instruments* .También se proponen los circuitos acondicionadores para la medición de dichos parámetros. Los sistemas de adquisición de datos se encaminan a utilizar configuraciones de modo que puedan implementar, con la información obtenida, acciones de control y monitoreo de los datos y cambios detectados.

Abstract

The present research work is part of the different studies that are being carried out at the Faculty of Electric Engineering to develop renewable energy. It aims at designing a data acquisition system which permits the monitoring of electrical variables in small hydro electrical mills. These variables will be measured and processed by a microcontroller PIC18F4520, and they will be transmitted to a PC for its supervision, using the LabWindows/CVI from National Instruments. The circuits for the measurement of these variables are also included in this work. The data acquisition systems are devoted to use configurations in such a way that, with the information obtained, they can apply controlling actions and monitoring data and detected changes.

Índice de contenidos:

Ilustraciones:

Introducción:

La revolución científico-técnica que se lleva a efecto en nuestra época ha impulsado vertiginosamente el uso y desarrollo de las computadoras y microprocesadores en el control automático de cualquier proceso productivo y en todas las ramas de la ingeniería eléctrica.
Para aumentar la calidad y eficiencia de los sistemas de control automático y por tanto de los procesos productivos, se requiere incrementar la exactitud de las mediciones y en particular las mediciones eléctricas.
El desarrollo y perfeccionamiento de los medios computacionales que tiene lugar en la actualidad requiere a su vez el perfeccionamiento de los principios, métodos y medios de medición que satisfagan las velocidades y posibilidades que brindan los microprocesadores y las computadoras electrónicas.
La ciencia de las mediciones eléctricas ha dado un salto cualitativo y cuantitativo con la introducción de los microprocesadores y computadoras electrónicas, lográndose una mayor exactitud, flexibilidad y confiabilidad en los resultados de las mediciones. Además el uso de computadoras permiten mayor velocidad en la obtención y procesamiento de los resultados de las mediciones y brinda la posibilidad de realizar simultáneamente las mediciones de un gran número de magnitudes diferentes.
Un campo de aplicación de estos sistemas puede extenderse a la medición, control y monitoreo de las principales magnitudes de pequeñas centrales hidroeléctricas, además posibilita el estudio e identificación de estas instalaciones caracterizadas por regímenes de trabajo sumamente adversas.
Estas pequeñas centrales hidroeléctricas, generalmente aprovechan el potencial hidroenergético de ríos de poco caudal, donde es importante el ahorro de agua, el mejor aprovechamiento de la potencia eléctrica instalada, fundamentalmente en sistemas eléctricos aislados, así como la explotación más racional y eficiente de toda la instalación. Además estas instalaciones están sometidas a fuertes perturbaciones, tales como cambios bruscos de la potencia demandada por los consumidores, variaciones del caudal y presión de agua, etc.
Un problema que presenta la mayoría de estas instalaciones en nuestro país, es la insuficiente y baja calidad de medios de medición; en muchos casos solo disponen de algunos instrumentos para la medición de voltaje, corriente y potencia.
El objetivo de este trabajo consiste en el desarrollo de un sistema de medición en el que se aprovecha las potencialidades de las computadoras y microcontroladores, el cual permite realizar las mediciones y registro de las diferentes magnitudes de interés en una pequeña central hidroeléctrica. El sistema permite mostrar los resultados en pantalla mediante gráficos, indicación de los valores efectivos, siendo esta una herramienta poderosa para el estudio e identificación dinámica de las instalaciones hidroenergéticas o de cualquier otro proceso, lo que resultaría complejo, trabajoso y de menor exactitud si se emplean medios convencionales.

El procedimiento empleado requiere de pocos medios técnicos (Hardware) lo que reduce el costo, aumenta la fiabilidad y facilita su construcción.

Problema de la investigación:

Limitación de la adquisición de datos para monitorear parámetros eléctricos en pequeñas centrales hidroeléctricas.

Objeto de la investigación:

Sistema de adquisición de datos para monitoreo en pequeñas centrales hidroeléctricas.

Objetivo de la investigación:

Diseño y caracterización de un sistema de adquisición de datos para la medición y monitoreo de parámetros eléctricos, usando un microcontrolador PIC18F4520.

Campo de acción:

Sistema de adquisición de datos para el monitoreo de parámetros eléctricos en pequeñas centrales hidroeléctricas usando el microcontrolador PIC18F4520.

Hipótesis:

Si se diseña el sistema de adquisición de datos para el monitoreo en pequeñas centrales hidroeléctricas, se lograría un procesamiento de las señales de medición así como su transmisión hacia una PC, aumentando las facilidades de supervisión de dichos parámetros.

Tareas de la investigación:

- Estudio de documentos relacionados con los sistemas de adquisición de datos y la energía renovable.
- Caracterizar los sistemas de adquisición de datos.
- Caracterizar el PIC18F4520.
- Diseño del hardware del sistema de adquisición de datos.
- Programación del PIC18F4520.
- Diseño y programación de la interfaz grafica de usuario usando el LabWindows/CVI.

Métodos teóricos empleados en la investigación:

- Método histórico – lógico.
- Método de análisis – síntesis.
- Método de inducción-deducción.

Métodos empíricos

- La observación.

El presente trabajo de diploma consta de resumen, introducción, dos capítulos, conclusiones, recomendaciones, bibliografía y anexos. En el primer capítulo se da una breve reseña histórica de los sistemas de adquisición de datos, así como la descripción de los elementos que los conforman, se presentan las ecuaciones necesarias para el cálculo de los parámetros eléctricos , se describe el microcontrolador PIC18F4520 y el protocolo de comunicación serie RS232.

En el segundo capítulo diseñan los diferentes circuitos acondicionadores para la medición y transmisión, se diseña el software para la programación del PIC18F4520 y del LabWindows/CVI. Asimismo, contiene una valoración económica del proyecto y la valoración de expertos.

.

Capitulo 1: Fundamentos teóricos de la investigación.

En este capítulo se hace referencia a las características de la adquisición de datos, además de una breve reseña histórica, se habla de la importancia del procesamiento digital para la rama de la energía. Se caracterizó un sistema de adquisición de datos señalando con una respectiva explicación cada componente de dicho sistema.

1.1. Reseña histórica de la adquisición y distribución de señales

La adquisición de datos, consiste en la toma de muestras del mundo real (sistema analógico) para generar datos que puedan ser manipulados por un ordenador (sistema digital). Consiste, en tomar un conjunto de variables físicas, convertirlas en tensiones eléctricas y digitalizarlas de manera que se puedan procesar en una computadora o PAC. Se requiere una etapa de acondicionamiento, que adecua la señal a niveles compatibles con el elemento que hace la transformación a señal digital. En 1927 Nyquist determinó que al muestrear una señal, la frecuencia de muestreo debe ser mayor que dos veces el ancho de banda de la señal de entrada, para poder reconstruir la señal original de forma exacta a partir de sus muestras. En caso contrario, aparecerá el fenómeno del Aliasing que se produce al infra-muestrear. Si la señal sufre aliasing, es imposible recuperar el original. Velocidad de muestreo recomendada:

–2*frecuencia mayor (medida de frecuencia).

–10*frecuencia mayor (detalle de la forma de onda).

Nyquist publicó sus resultados en el artículo "Certain topics in Telegraph Transmission Theory (1928)". Esta regla es ahora conocida como el teorema de muestreo de Nyquist-Shannon.

1.2. Caracterización de la adquisición de datos.

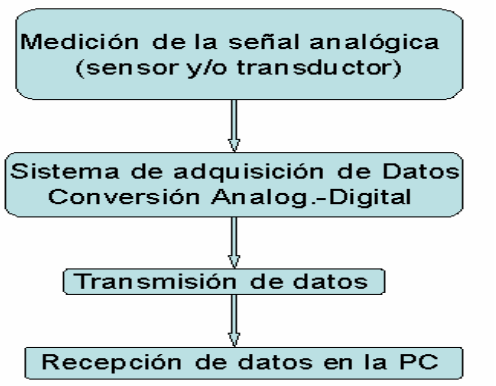


Figura. 1.1: Sistema clásico de adquisición de datos.

Todos los sistemas de medición electrónicos están compuestos de instrumentos y componentes interconectados para poder realizar una función de medición. Cada componente del sistema no solo debe realizar su función individual correctamente sino también trabajar efectivamente con los demás componentes para que todo el sistema opere correctamente.

Un sistema de adquisición y monitoreo moderno se compone de varias etapas. La primera consiste en la recolección de datos por medio de sensores que traducen valores físicos a señales eléctricas, mayormente estas señales son análogas.
Las señales recolectadas son enviadas por medio de cableado. Estas señales pueden ser filtradas de perturbaciones o ruido. Una segunda etapa se compone de los módulos convertidores analógicos a digital (ADC) provenientes de los sensores o transductores, y de los módulos de convertidor digital a analógico (DAC) que van hacia los actuadotes del sistema. Las señales de los convertidores pasan por una interfaz, que entabla comunicación con el procesador. Esta interfaz es llamada también tarjeta de adquisición de datos. Los datos son sincronizados por un reloj. De ésta interfaz los datos pasan a un bus de comunicación, de donde a su vez pasan los datos del procesador hacia la interfase. En la ultima etapa del trayecto de la información esta el computador. Aplicaciones de software traducen a nivel de usuario la información recolectada. Estos valores pueden ser procesados con otras herramientas, brindando una gran versatilidad y facilidad de usos. Los datos adquiridos se visualizan, analizan, y almacenan en un ordenador, ya sea utilizando el proveedor de software suministrado u otro software. Los controles y visualizaciones se pueden desarrollar utilizando varios lenguajes de programación de propósito general como VisualBASIC, C++, Fortran, Java, Lisp, Pascal. Los lenguajes especializados de programación utilizados para la adquisición de datos incluyen EPICS, utilizada en la construcción de grandes sistemas de adquisición de datos. LabWindows, ofrece un entorno gráfico de programación optimizado para la adquisición de datos. Estos entornos de adquisición proporcionan un lenguaje de programación además de bibliotecas y herramientas para la adquisición de datos y posterior análisis.

Las interfaces entre la señal y una PC podría ser en forma de módulos que pueden ser conectados a la computadora de los puertos (paralelo, serie, USB, etc...) o ranuras de las tarjetas conectadas a (PCI, ISA) en la placa madre. Por lo general, el espacio en la parte posterior de una tarjeta PCI es demasiado pequeño para todas las conexiones necesarias, de modo que una ruptura de caja externa es obligatoria. El cable entre este recuadro y el PC es cara debido a los numerosos cables y el blindaje necesario y porque es exótico. Las tarjetas DAQ a menudo contienen múltiples componentes (multiplexores, ADC, DAC, TTL-IO, temporizadores de alta velocidad, memoria RAM). Estos son accesibles a través de un bus por un microcontrolador, que puede ejecutar pequeños programas. El controlador es más flexible que una unidad lógica dura cableada, pero más barato que una CPU de modo que es correcto para bloquear con simples bucles de preguntas.

1.3. Descripción de la instalación hidrogeneradora.

La instalación hidrogeneradora consiste de un generador eléctrico acoplado a una turbina hidráulica destinada a la producción de energía eléctrica aprovechando la energía potencial del agua en su caída desde un nivel superior a otro inferior. El flujo de agua es conducido por una tubería conductora desde el embalse hasta la válvula que regula el flujo de agua que llega a la Turbina. Al actuar el agua sobre los álabes de la Turbina produce un momento que hace girar el eje a una determinada velocidad. Acoplado al eje de la Turbina se encuentra el generador eléctrico, el cual entrega en sus terminales de salida una Potencia Eléctrica con un determinado nivel de voltaje y de frecuencia que dependerán de la excitación eléctrica, de la velocidad de giro de la Turbina y de la corriente eléctrica que demanda la carga. En la Figura1.2 se muestra el diagrama simplificado de esta instalación [2].

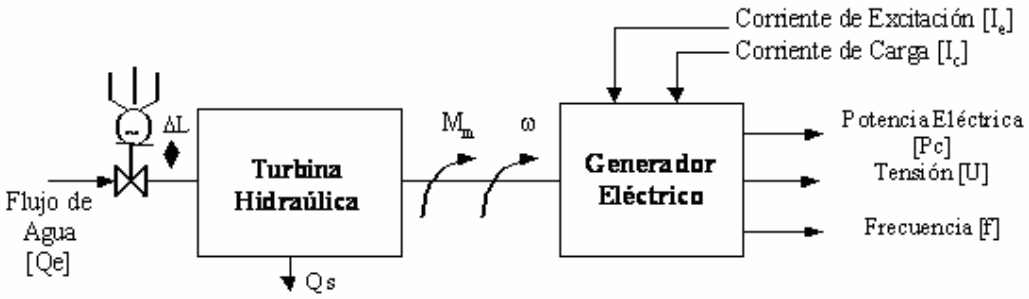


Figura 1.2. Diagrama en bloques de una instalación hidrogeneradora.

donde:

- ΔL : Desplazamiento del vástago de la válvula reguladora de agua.
- Qe: Flujo de agua de entrada a la válvula reguladora de agua.
- Qs: Flujo de agua de salida de la Turbina Hidráulica.
- Mm: Momento motriz de la Turbina Hidráulica.
- w : Velocidad de giro de la Turbina Hidráulica.
- Ie: Corriente de excitación del generador eléctrico.
- Ic: Corriente de carga.
- Pe: Potencia eléctrica suministrada a la carga.
- U : Voltaje eléctrico de salida del generador.
- f : Frecuencia de la salida del generador.

Generalmente, estas instalaciones denominadas Micro, Mini o Pequeñas Centrales Hidroenergéticas, según el nivel de energía eléctrica que produzcan, van equipadas con generadores sincrónicos, los cuales son máquinas robustas y de simple control. Estas máquinas generan energía eléctrica cuya frecuencia y la velocidad son sincrónicas, de ahí su denominación [2].

1.4. Medición y cálculo de los parámetros eléctricos.

El principio de medición de las diferentes magnitudes se basa en las definiciones fundamentales que aparecen en las bibliografías especializadas sobre este tema y a partir de éstas se condicionan para ser procesadas mediante procedimientos computacionales.

1.4.1 Medición de la frecuencia.

La frecuencia, en un sistema eléctrico de corriente alterna, está directamente relacionada con la velocidad de giro, es decir, con el número de revoluciones por minuto de los alternadores. Esto es[10]:

$$n = 60\frac{f}{p} \quad \text{[rpm]} \qquad (1)$$

Donde:
n = velocidad en rpm del generador.
f = frecuencia en Hz.
p = número de pares de polos.
Las variaciones de velocidad se traducen en variaciones de la frecuencia y de la tensión a la salida del generador. En los grandes sistemas eléctricos, dado que la frecuencia es común a toda la red, los generadores conectados a ella girarán de manera sincrónica a la misma velocidad angular eléctrica. La frecuencia nominal de la tensión de la red en nuestro país es de 60Hz. Para poder medir dicha frecuencia es necesario convertir la sinusoide en un tren de pulsos equivalente a la misma. Para una simple señal digital, como la descrita en la figura 1.3, el periodo es directamente el tiempo entre flancos de subida, o entre flancos de bajada.

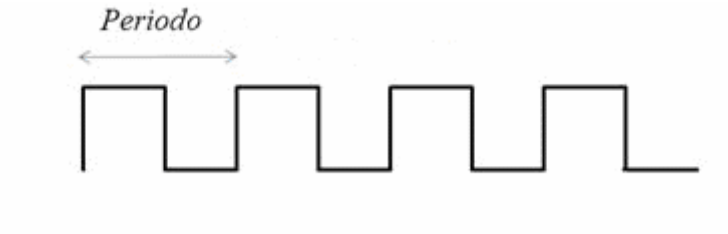


Figura1.3. Forma de Onda Digital.

Si el tiempo entre flancos de subida o de bajada varia ligeramente, se puede promediarlos sobre un gran número de muestras para determinar la frecuencia.

Para una adquisición de frecuencia digital, el proceso es bastante simple. Para señales de baja frecuencia, es suficiente emplear un contador o un tiempo base. El flanco de subida de la señal de entrada dispara el número de veces que el tiempo base debe ser contado. Ya que el tiempo base es de una frecuencia conocida, se puede calcular fácilmente la frecuencia de la señal de entrada (Figura 1.4).

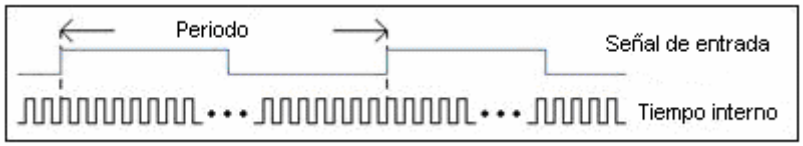


Figura 1.4. Señal Digital con Respecto al Tiempo Base Interno.

1.4.2. Medición de voltaje.

Las mediciones instantáneas de las señales de voltaje y corriente se toman directamente por las entradas analógicas del microcontrolador. Los valores efectivos se calculan a partir de los valores instantáneos aplicando la fórmula [10]:

$$U = \sqrt{\frac{1}{T}\int_0^T u^2 dt} \qquad (2)$$

donde U : valor efectivo de la señal
u : valor instantáneo de la señal
T : período de la señal.

En su versión discreta:

$$U = \sqrt{\frac{1}{N}\sum_{i=1}^{N} u_i^2} \qquad (3)$$

donde u_i : valor muestreado en cada instante
N : número de muestreo en un período o en un número multiplo del período.

1.4.3. Medición de corriente.

La corriente I que se desea medir se convierte en una voltaje U = f(I) mediante un acondicionador de señal. El valor instantáneo de la corriente es proporcional al valor instantáneo del voltaje Us a la salida del acondicionador de señal. Por tanto, el valor efectivo de la corriente es [10]:

$$I = K\sqrt{\frac{1}{N}\sum_{i=1}^{N} u_i^2} = Us \qquad (4)$$

La medición de U_s se realiza con el microcontrolador del mismo modo que la medición de voltaje explicado anteriormente proceso de medición de corriente y voltaje es similar, diferenciándose solamente en el tipo de acondicionador de señal.

1.4.4. Cálculo de potencias y factor de potencia.

El cálculo de la potencia activa se realiza a partir de los valores instantáneos de los voltajes y corrientes por la fórmula [10]:

$$P = \frac{1}{T}\int_0^T u\,i\,dt \qquad (5)$$

En su versión discreta:

$$P = \frac{1}{N}\sum_{k=1}^{N} u_k i_k \qquad (6)$$

El cálculo de la potencia aparente, potencia reactiva y factor de potencia se realiza a partir de los valores efectivos de los voltajes y las corrientes por las siguientes fórmulas [10]:

- La potencia aparente S

$$S = U_{rms} \cdot I_{rms} \qquad (7)$$

- Potencia reactiva Q

$$Q = \sqrt{S^2 - P^2} \qquad (8)$$

- Factor de potencia FP

$$FP = \cos\varphi = \frac{P}{S} \qquad (9)$$

1.5. Mediciones en un sistema trifásico tetrafilar con conexión en estrella.

En la figura 1.5 se muestra el esquema de medición de voltajes, corrientes, potencias y factor de potencia en un generador trifásico con conexión en estrella y neutro accesible [10].

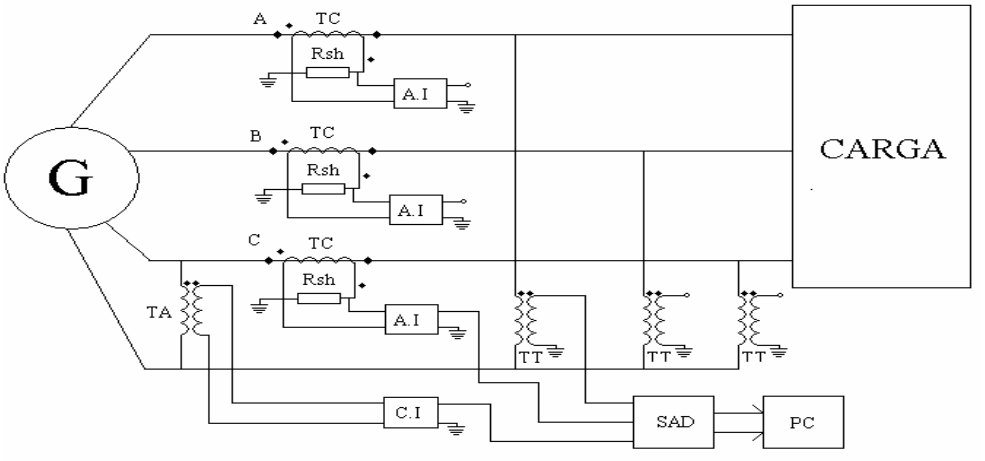


Figura 1.5. Esquema de conexiones del sistema de adquisición.

En cada una de las líneas se conecta un acondicionador de corriente para la medición de las corrientes de líneas y entre cada línea y el neutro se conecta un acondicionador de voltaje para la medición de los voltajes de fase. Se miden los valores instantáneos de los voltajes y corrientes y se calculan los valores efectivos por fase y las potencias por fase. Las potencias trifásicas totales se calculan entonces por las sumas de las potencias por fase [10].

$$P_T = P_{fA} + P_{fB} + P_{fC} \quad (10)$$

$$S_T = S_{fA} + S_{fB} + S_{fC} \quad (11)$$

$$Q_T = \sqrt{S^2 - P^2} \quad (12)$$

$$FP_T = \cos\varphi = \frac{P_T}{S_T} \quad (13)$$

La medición de las magnitudes en sistemas trifásicos con cualquier conexión puede realizarse haciendo uso de los esquemas de conexiones convencionales, requiriendo solo cambios en las conexiones de los acondicionadores y en el programa, sin modificaciones sustanciales en el hardware. También, es posible obtener los valores medios de corriente y tensión y los valores instantáneos de la corriente, voltaje y potencia por fase mediante cambios en el programa [10].

1.6. Características del Microcontrolador PIC18F4520

La arquitectura del procesador sigue el modelo Harvard*:* en esta arquitectura, la CPU se conecta de forma independiente y mediante buses distintos con la memoria de instrucciones y con la de datos.

De esta forma la CPU puede acceder simultáneamente a las dos memorias, como se observa en la figura.1.6 [6].

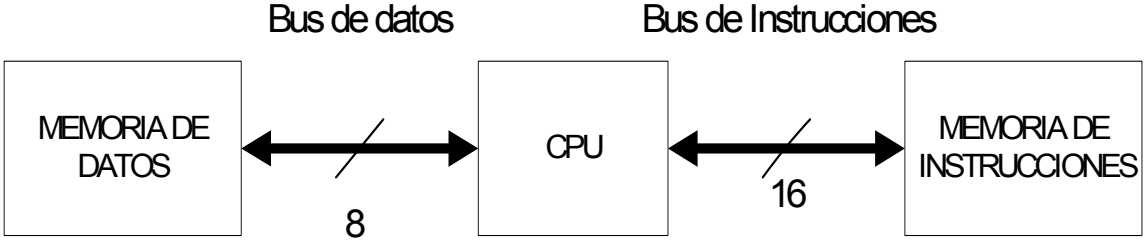


Figura 1.6. Arquitectura del microcontrolador PIC18F4520.

La familia *PIC18FXXXX* admite la *Programación Serie En Circuito* (*ICSP*), que permite la grabación del programa sobre el microcontrolador una vez que se haya colocado en la tarjeta del producto final. Los microcontroladores del modelo *PIC18F4520* son circuitos *RISC* de 77 instrucciones. El tamaño de cada instrucción es de 16 bits, o sea, una palabra de dos bytes aunque hay cuatro instrucciones cuyo tamaño es de dos palabras de 16 bits. El vector de *RESET* ocupa la dirección 0x0000 y las interrupciones las direcciones 0x0008 y 0x0018.
La arquitectura interna del *PIC18F4520* es de tipo *Harvard.* La estructura *Harvard* se caracteriza por la independencia entre la memoria de código y la memoria de datos. La utilización de buses diferentes para cada tipo de memoria posibilita el trabajo en paralelo de las dos memorias y proporciona gran velocidad que se traduce como una alta eficiencia. Los microcontroladores del modelo *PIC18F4520* ofrecen la posibilidad de funcionamiento en condición de reposo o de bajo consumo y pueden trabajar en modo maestro o en modo esclavo [6] .

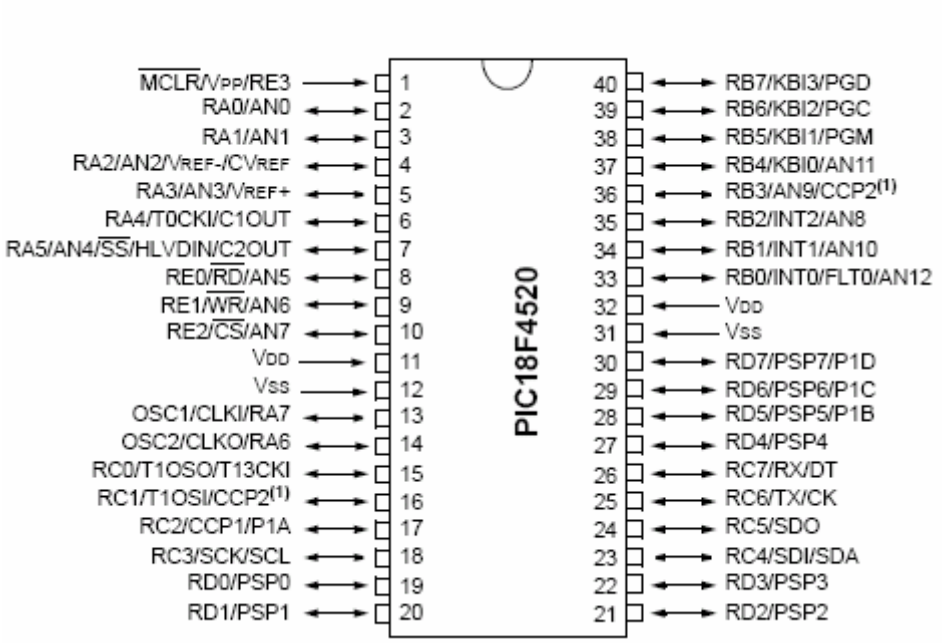


Figura1.7.Diagrama de pines del PIC18F4520.

Los dispositivos *PIC18F452* poseen las siguientes características [6]:

- Frecuencia máxima de operación de 40 Mhz de DC.
- Memoria de programa (bytes) de 32 K.
- Memoria de programa (instrucciones) de 16384.
- Memoria de datos (bytes) de 1536.
- Memoria de datos *EEPROM* (bytes) de 256.
- 18 fuentes de interrupción.
- 5 Puertos de E/S (A, B, C, D, E).
- 4 Temporizadores.
- 2 Módulos de captura, comparación y temporización (*CCP1, CCP2*).
- Comunicación serie *MSSP, USART* direccionable.
- Comunicación paralelo *PSP.*
- Conversor A/D de 10 bits con 8 canales de entrada.

- Instrucción de reset pila llena, desbordamiento de pila *RESET*, POR, BOR.
- Temporizadores *RESET, PWRT* y *OST.*
- Programación con voltaje bajo.
- Detección de baja tensión.
- Juego de instrucciones.
- Encapsulados *DIP 40, PLCC 40, TOFP 40, IW 40.*

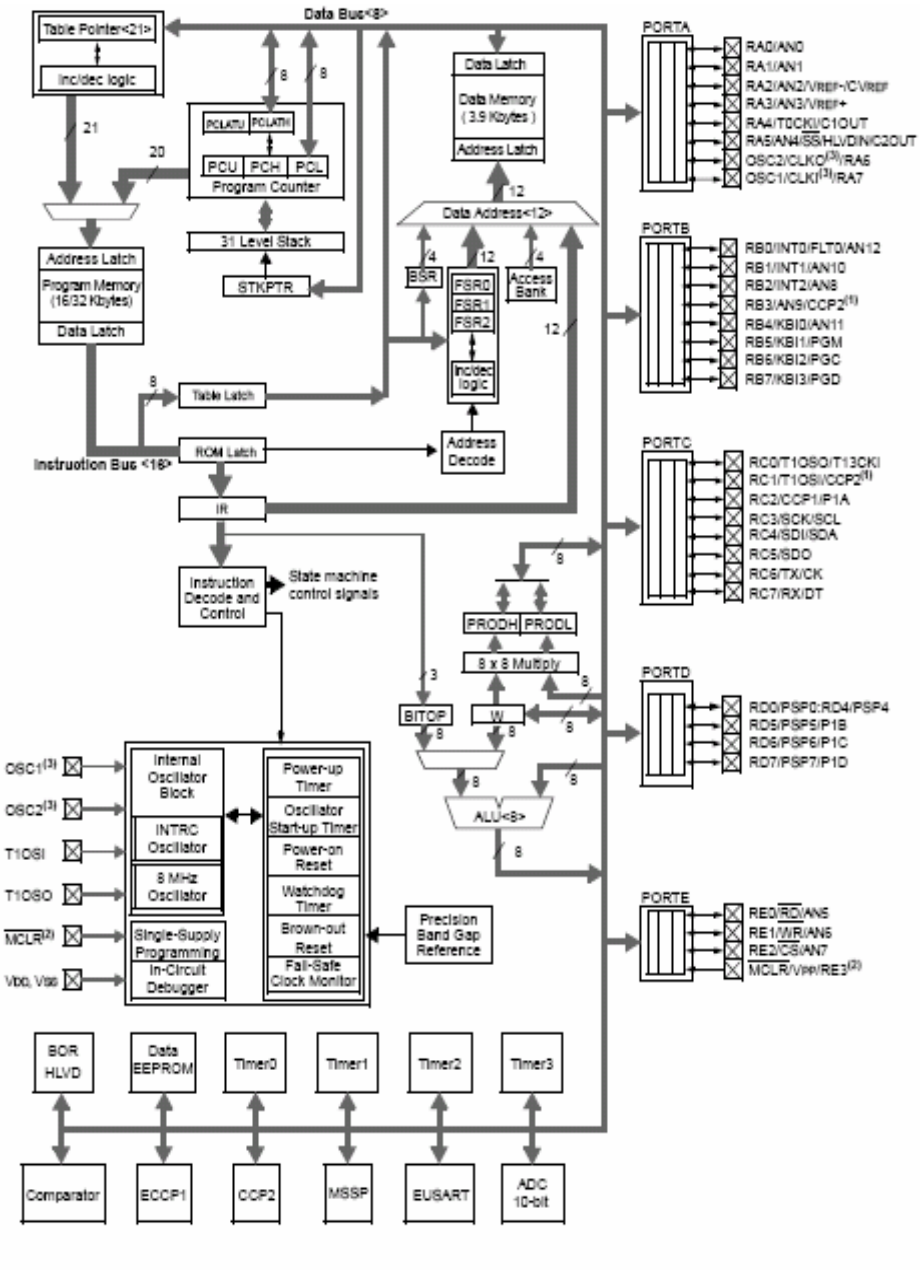


Figura 1.8.Diagrama en bloques del PIC18F4520.

Las características periféricas del microcontrolador PIC18F4520 son:

•*Timer 0*: Temporizador de 8-bit/16-bit, configurable como temporizador/contador con un divisor de frecuencia programable (*prescaler*) de 8-bit.
• *Timer 1*: Temporizador de 16-bit, configurable como temporizador /contador.
• *Timer 2*: Temporizador de 8-bit configurable como temporizador /contador con un registro de período de 8-bit (base de tiempo para PWM).
• Dos módulos *CCP* de Captura, Comparación y Modulación por Ancho de Pulso (*PWM*).
- Modo Captura de 16-bit, su máxima resolución es de 6.95 ns.
- Modo Comparación de 16-bit, su máxima resolución es de 100 ns.
- *PWM* con resolución máxima de 10-bit.
• Conversor analógico/digital *(CAD)* multicanal con resolución de 10-bit.
• Puerto Serie Sincrónico (*MSSP*) con los modos de operación: *SPI* y *I2C*.
• Receptor Universal Sincrónico y Asincrónico (*USART/SCI*) con 9-bit de detección de dirección.
• Puerto Paralelo Esclavo (*PSP*) de 8-bits, con controles externos (*RD, WR y CS*).
• Circuito de detección de *Brown-out* para el *Brown-out Reset* (*BOR*).
- Timer 3.

El *Timer 3* es un temporizador/ contador de 16 bits que consta de dos registros de 8 bits cada uno (*TMR3H* y *TMR3*), los cuales pueden leerse y escribirse [6].

Organización de la memoria.

La memoria del microcontrolador está compuesta por tres bloques:

- Memoria de programa
- Memoria de datos *RAM*
- Memoria de datos *EEPROM*

La memoria de programa y la memoria de datos utilizan buses separados [7].

Puertos de entrada/ salida.

Algunos pines de E/S están multiplexados con alguna función alternativa para los periféricos del microcontrolador. En general cuando los periféricos son habilitados, estos pines no se emplean como E/S de propósito general. El dispositivo *PIC18F4520* posee cinco puertos de entrada/salida: A, B, C, D, E. Cada uno de estos puertos tiene tres registros de operaciones. El procedimiento para el trabajo con dichos registros es el mismo para los del microcontrolador. Los registros son los siguientes [6]:

- El registro de direcciones (*TRIS).*
- El registro *PORT* que lee los niveles de los pines del dispositivo.
- El registro de salida (*LAT).*

Temporizadores.
Módulo Timer 0.
El *Timer 0* tiene las siguientes características:

- Puede operar como contador de 8 bits.
- Es leíble y escribible.
- Divisor de frecuencia programable (*prescaler*) de 8 bits
- Reloj selector interno y externo.
- Interrupción de "*overflow*" desde la FFh a 00h.
- Selector de flanco para el reloj externo.

El modo temporizador se selecciona borrando el bit *T0CS (T0C0N*<5>). El *Timer 0*, se incrementará en todos los ciclos de instrucción (sin *prescaler*). Si se escribe el registro *TMR0*, el incremento es deshabilitado durante los 2 siguientes ciclos de instrucciones [7]. El usuario puede trabajar con el mismo, escribiendo un valor ajustado en el registro *TMR0* [6].

El modo contador se selecciona al poner a "1" el bit *T0CS*. En este modo, el temporizador se incrementará en todos los flancos de subida o bajada del pin RA4/T0CK1, donde el incremento del flanco se determina por el bit *T0SE (T0C0N* <4>), haciendo *T0SE*= "0" se selecciona el flanco de subida [6].

Módulo Timer 1.

El *Timer 1* es un temporizador/contador de 16 bits que consta de dos registros de 8 bits cada uno (*TMR1H* y *TMR1L*), los cuales pueden leerse y escribirse. Este par de registros se incrementan desde la 0x0000 a la 0xFFFF y vuelve a la 0x 0000. El *Timer1* puede ser habilitado o deshabilitado, seteando o borrando el bit de control *TMR1ON (T1CON*<0>). Cuando el oscilador del *Timer1* es habilitado (*T1OSCEN*="1"), los pines *RC1/T1OSI/CCP2* y *RC0/T1OSO/T1CKI*, se convierten en entradas. La interrupción del *TMR1*, si se habilita, es generada por un desbordamiento (*overflow* desde 0xFFFF a 0x0000), y muestreada por la bandera de interrupción *TMR1IF (PIR1*<0>). Esta interrupción puede ser habilitada o deshabilitada, seteando o borrando el bit de habilitación del *T1OSCEN (PIE1*<0>). El *Timer 1* opera de dos modos diferentes. El modo de operación se determina por el bit *TMR1CS (T1CON*<1>).
Los modos de operación del *Timer 1* son los siguientes:

- Como temporizador.
- Como Contador.
  - Contador Asincrónico.
  - Contador Sincrónico.

En el modo temporizador el *Timer 1* se incrementa en cada ciclo de instrucción. Como contador, se incrementa en cada flanco de subida del reloj externo. También cuenta con un *Reset* interno de entrada, el cual es generado por cualquiera de los módulos *CCP*. O sea el valor de *TRISC*<1:0>, es ignorado. El contador del *prescaler* se borra con la escritura en el registro *TMR1L*.

Módulo Timer 2.

El *Timer 2* es un temporizador de 8 bits con *prescaler* y *postscaler*. Puede ser usado como base de tiempo para el modo *PWM* de los módulos *CCP*. El registro *TMR2* puede leerse y escribirse, se borra con cualquier *Reset*. El reloj de entrada (*Fosc/4*) tiene opciones de prescala de 1:1, 1:4, 1:16, que se escogen con el bit de control *T2CKPS1:T2CKPS0 (T2CON*<1:0>).

El *Timer 2* tiene un registro y un periodo de 8 bits, *PR2*. Este temporizador se incrementa desde la 0x00 hasta 0xFF, y vuelve a la 0x 00 en el próximo ciclo de incremento. PR2 se puede leer y escribir. La salida del *TMR2* va a través de un *postscaler* de 4 bits para generar la interrupción *TMR2*, que es muestreada en el bit *TMR2IF (PIR1*<1>). El *Timer 2* puede ser borrado haciendo "0" el bit de control *TMR2ON (T2CON*<2>), para minimizar el consumo de energía.

Módulo Timer 3.

El *Timer 3* es un temporizador/ contador de 16 bits que consta de dos registros de 8 bits cada uno (*TMR3H* y *TMR3*), los cuales pueden leerse y escribirse. El módulo *Timer 3* trabaja de forma similar al módulo *Timer 1* pues poseen la mismas características. Para seleccionar este modo el *Timer 1* y/o el *Timer 3* deben estar en modo temporizador o en modo contador sincrónico. En el registro *T3CON* se selecciona al temporizador para usar cada módulo *CCP.*

El perro guardián.

El perro guardián *(WDT)* es como una corrida libre del *on-chip* del oscilador *RC* que no requiere ningún componente externo. Este oscilador *RC* está separado del oscilador *RC* del pin *OSC1/CLKI*. Eso significa que el *WDT* correrá, aun cuando el reloj en los pines *OSC1/CLKI* y *OSC2/CLKO/RA6* del dispositivo se hayan detenido, por ejemplo, para la ejecución de la instrucción *SLEEP* [7].
El *WDT* puede habilitarse o deshabilitarse con el bit de configuración *SWDTEN*. Durante la operación normal, el *Time-out* del *WDT* genera un *Reset* al dispositivo (*Reset* del *Timer* del *WDT*). Si el dispositivo está en modo *SLEEP*, el *Time-out* del *WDT* provoca que el dispositivo se levante y continúe con la operación normal (levantamiento del *Timer* del *WDT*). El bit *TO* en el registro *RCON* será borrado por un *Time-out* del *Watchdog* [6].

Conversor Analógico-Digital.

El conversor analógico-digital (*A/D*) tiene 8 entradas. La entrada analógica carga un capacitor "*Sample and Hold*". La salida del capacitor es la entrada al convertidor. El convertidor genera una salida digital de 10 bits aproximada al nivel analógico. El *CAD* tiene voltajes de referencia alto y bajo, determinado por la combinación de V_{DD}*, Vss, RA2, RA3* [8]. El *CAD* tiene 5 registros [6]:

1. -Registro que contiene la parte alta del resultado *A/D (ADRESH).*
2. -Registro que contiene la parte baja del resultado *A/D (ADRESL).*
3. -Registro de Control 0 de *A/D (ADCON0).*
4. -Registro de Control 1 de *A/D (ADCON1).*
5. -Registro de Control 2 de A/D (*ADCON2*).

Pasos para efectuar una conversión de *A/D*:

1. Configurar el módulo *A/D.*
   • Configurar los pines analógicos / la referencia de voltaje / y la I/O digital (*ADCON*1).
   • Seleccionar el canal de entrada del *A/D (ADCON0).*
   • Seleccionar el reloj de conversión del *A/D (ADCON0).*

• Encender el módulo *A/D (ADCON0).*

2. Configurar las interrupciones del *A/D*:
   • Borrar el bit *ADIF.*
   • Setear el bit *ADIE.*
   • Setear el bit *GIE .*
3. Esperar el tiempo de adquisición requerido.

1.7. Comunicación serie.

Cuando se transmite información a través de una línea serie es necesario utilizar un **sistema de codificación** que permita resolver los siguientes problemas [5]:

1. Sincronización de bits: El receptor necesita saber dónde comienza y termina cada bit en la señal recibida para efectuar el muestreo de la misma en el centro del intervalo de cada símbolo (bit para señales binarias).

2. Sincronización del caracter: La información serie se transmite por definición bit a bit, pero la misma tiene sentido en palabras o bytes.

3. Sincronización del mensaje: Es necesario conocer el inicio y fin de una cadena de caracteres por parte del receptor para, por ejemplo, detectar errores en la comunicación de un mensaje.

La **velocidad de transmisión de datos** es expresada en bits por segundo o baudios. El baudio es un concepto más general que bit por segundo. El primero queda definido como el número de estados de la señal por segundo, si sólo existe dos estados (que pueden ser representados por un bit, que identifica dos unidades de información) entonces baudio es equivalente a bit por segundo. Baudio y bit por segundo se diferencian cuando es necesario más de un bit para representar más de dos estados de la señal.

La velocidad de transmisión queda limitada por el ancho de banda, potencia de señal y ruido en el conductor de señal. La velocidad de transmisión queda básicamente establecida por el reloj. Su misión es examinar o muestrear continuamente la línea para detectar la presencia o ausencia de los niveles de señal ya predefinidos. El reloj sincroniza además todos los componentes internos.

Cuando se establece la comunicación es necesario implementar una base de tiempo que controle la velocidad. En un microcontrolador se utilizaría la **base de tiempos del reloj** del sistema, si bien, en términos genéricos se utilizaría uno de los siguientes métodos [5]:

a. Mediante la división de la base de reloj del sistema, por ejemplo mediante un contador temporizador programable.

b. A través de un oscilador TTL, para cambiar frecuencia hay que cambiar el cristal.

c. Generador de razón de baudios, existen diferentes dispositivos especializados que generan diferentes frecuencias de reloj.

Se pueden establecer **canales (líneas)** para la comunicación de acuerdo a 3 técnicas, siempre tomando al microcontrolador como referencia (transmisor) y al periférico como destino (receptor) [5]:

- *Simplex*: En ella la comunicación serie usa una dirección y una línea de comunicación. Siempre existirá un transmisor y un receptor, no ambos. La ventaja de este sistema consiste en que es necesario sólo un enlace a dos hilos. La desventaja radica en que el extremo receptor no tiene forma de avisar al extremo transmisor sobre su estado y la calidad de la información que se recibe. Por esta razón generalmente no se utiliza.

- Semi *duplex*: La comunicación serie se establece a través de una sola línea, pero en ambos sentidos. En un momento el transmisor enviará información y en otro recibirá, por lo que no se puede transferir información en ambos sentidos de forma simultánea. Este modo permite la transmisión desde el extremo receptor de la información, sobre el estado de dicho receptor y sobre la calidad de la información recibida por lo que permite así la realización de procedimientos de detección y corrección de errores.

- Totalmente *duplex (Full duplex)*: Se utilizan 2 líneas (una transmisora y otra receptora) y se transfiere información en ambos sentidos. La ventaja de este método es que se puede transmitir y recibir información de manera simultánea. La mayoría de los dispositivos especializados para la comunicación pueden transferir información en *full duplex* como en *half duplex* (el modo *simplex* es un caso especial dentro de *half duplex*).

1.7.1. Modos de transmisión.

Existen dos modos básicos para realizar la transmisión de datos y son [5]:

1. Modo asincrónico: Las transmisiones asíncronas son aquellas en que los bits que constituyen el código de un caracter se emiten con la ayuda de impulsos suplementarios que permiten mantener en sincronismo los dos extremos.

Cuando se opera en modo asíncrono no existe una línea de reloj común que establezca la duración de un bit y el caracter puede ser enviado en cualquier momento. Esto conlleva que cada dispositivo tiene su propio reloj y que previamente se ha acordado que ambos dispositivos transmitirán datos a la misma velocidad.

En un sistema digital un reloj es normalmente utilizado para sincronizar la transferencia de datos entre las diferentes partes del sistema; definirá el inicio y fin de cada unidad de información así como la velocidad de transmisión. Si no existe reloj común, algún modo debe ser utilizado para sincronizar el mensaje.

La frecuencia con que el reloj muestrea la línea de comunicación es mayor que la cadencia con que llegan los datos, por ejemplo, si los datos están llegando a una cadencia de 2400 bps, el reloj examinará la línea unas 19200 veces por segundo, es decir, ocho veces la cadencia binaria. La gran rapidez con que el reloj muestrea la línea, permite al dispositivo receptor detectar una transmisión de “1” a “0” ó de “0” a “1” muy rápidamente, y mantener así la mejor sincronización entre los dispositivos emisor y receptor.

El tiempo por bit en una línea en que se transfiere la información a 2400 bps es de unos 416 microsegundos (1 seg/2400). Una frecuencia de muestreo de 2400 veces por segundo permitirá muestrear el principio o el final del bit.

En ambos casos se detectará el bit, sin embargo, no es extraño que la señal cambie ligeramente, y permanezca la línea con una duración un poco más larga o más corta de lo normal. Una frecuencia de muestreo lenta no sería capaz de detectar el cambio de estado de la señal a su debido tiempo, y esto daría lugar a que la estación terminal no recibiera los bits correctamente.

En la transmisión asíncrona un caracter a transmitir es encuadrado con un indicador de inicio y fin de caracter. La forma estándar de encuadrar un caracter es a través de un bit de inicio y un bit de parada.

Durante el intervalo de tiempo en que no son transferidos caracteres, el canal debe poseer un "1" lógico. Al bit de parada se le asigna también un "1". Al bit de inicio del caracter a transmitir se le asigna un "0" (figura 8), por lo que un cambio de nivel de "1" a "0" lógico le indicará al receptor que un nuevo caracter será transmitido .

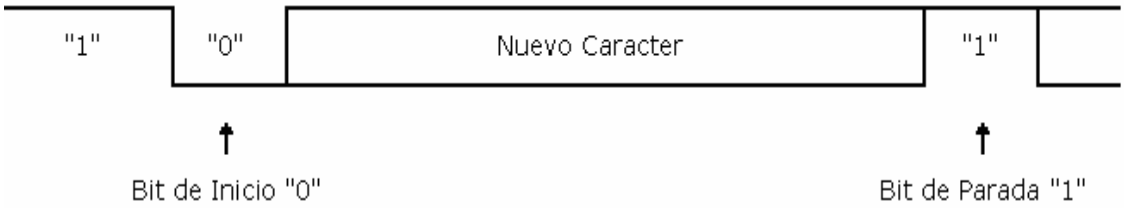


Figura 1.9.Formato de transmisión asincrónica.

Esta transmisión es definida por el protocolo RS232, que se basa en las siguientes reglas [5]:

a. Cuando no se envían datos por la línea, ésta se mantiene en estado alto “1”.

b. Cuando se desea transmitir un caracter, se envía primero un bit de inicio que pone la línea a estado bajo “0” durante el tiempo de un bit.

c. Durante la transmisión si la línea está a nivel bajo, se envía un “0” y a nivel alto un “1”.

d. A continuación se envían todos los bits del mensaje a transmitir con los intervalos que marca el reloj de transmisión. Por convenio se transmiten entre 5 y 8 bits.

e. Se envía primero el bit menos significativo, siendo el más significativo el último.

f. A continuación del último bit del mensaje se envía el bit (o bits) del final que hace que la línea se ponga a “1” por lo menos durante el tiempo mínimo de un bit. Estos bits pueden ser un bit de paridad para detectar errores y el bit o bits de stop, que indican el fin de la transmisión de un caracter.

Los datos codificados por esta regla, pueden ser recibidos siguiendo los pasos siguientes:

a. Esperar la transición “1” a “0” en la señal recibida.

b. Activar el reloj con una frecuencia igual a la del transmisor.

c. Muestrear la señal recibida al ritmo de ese reloj para formar el mensaje.

d. Leer un bit más de la línea y comprobar si es “1” para confirmar que no hay error en la sincronización.

La característica fundamental del formato de transmisión asíncrono es su capacidad de manejar datos en tiempo real, con un intervalo de longitud arbitraria entre caracteres sucesivos.

2. Modo sincrónico: los caracteres se transmiten consecutivamente, no existe bit de inicio ni de parada entre caracteres, la corriente de estos está dividida en bloques, enviándose una secuencia de sincronización al inicio de cada bloque (figura 1.10) [5].

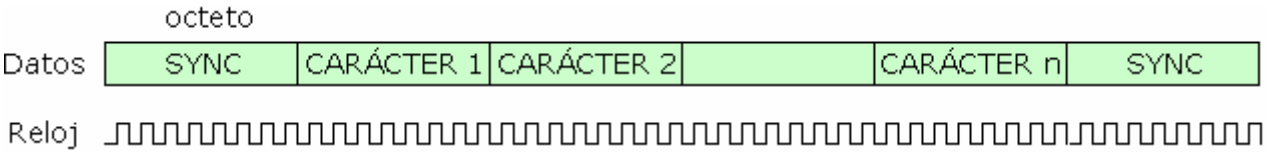


Figura 1.10.Formato de transmisión sincrónica.

La transmisión síncrona es un método más eficiente de comunicación en cuanto a velocidad, no existe información adicional entre caracteres a ser transmitidos.

Cuando se transmite de manera síncrona lo primero que se envía es un octeto de sincronismo ("*sync*"). El octeto de sincronismo realiza la misma función que el bit de inicio en la transmisión asíncrona, indicando al receptor que va ha ser enviado un mensaje. Este caracter, utiliza la señal local de reloj para determinar cuándo y con qué frecuencia será muestreada la señal, permite sincronizar los relojes de los dispositivos transmisor y receptor. La mayoría de los dispositivos de comunicación llevan a cabo una resincronización contra posibles desviaciones del reloj, cada 1 ó 2 segundos, insertando caracteres del tipo "*sync*" periódicamente dentro del mensaje.

Los caracteres de sincronismo deben diferenciarse de los datos del usuario para permitir al receptor detectar los caracteres "*sync*", por ejemplo, el código ASCII utiliza el octeto 10010110. Existen ocasiones en que son definidos dos caracteres de sincronismo, ello puede ser necesario si, por cualquier motivo el caracter "*sync*" original se desvirtuara, el siguiente permitirá la reinicialización del receptor. En segundo lugar, puede ocurrir que el equipo receptor necesite un tiempo adicional para adaptarse a la señal entrante.

Cuando se transmite de forma síncrona, es necesario mantener el sincronismo entre transmisor y receptor cuando no se envían caracteres, para ello son insertados caracteres de sincronismo de manera automática por el dispositivo que realiza la comunicación (figura 1.11) [5].

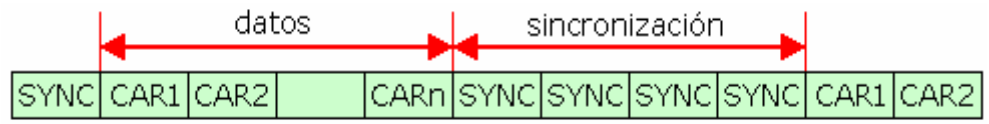


Figura 1.11.Inserción automática de caracteres de sincronismo.

El receptor/transmisor síncrono debe indicar además cuándo el sincronismo ha sido logrado por parte del receptor.

1.7.2 .Errores en la comunicación.

Cuando se escriben o envían datos, pueden producirse errores, entre otras cosas, por ruidos inducidos en las líneas de transmisión de datos. Es necesario comprobar la integridad de los datos transmitidos mediante métodos que permitan determina si se ha producido un error. En un caso típico, si al transmitirse un mensaje se determina que se ha producido un error, el receptor solicita de nuevo el mensaje al emisor [5].

Se pueden detectar errores de acuerdo con la forma de transmisión:

1. Transmisión asíncrona: - Paridad - Sobre escritura -Error de encuadre (*framing*)

2. Transmisión síncrona: - Paridad - Sobre escritura

Generadores y detectores de paridad

Un error en una transmisión serie solamente suele afectar a un bit, uno de los métodos más comunes para detectar errores es el control de la paridad, consiste en añadir un bit, de paridad, a los datos que se envían o escriben. La paridad puede ser par o impar.

Paridad par: El bit de paridad será cero, cuando el número de bit "unos" que contienen los datos a transmitir sea un número par, y el bit de paridad será uno cuando los datos que se mandan contienen un número impar de unos.

Dato	Paridad
0000 0001	1
0101 0001	1
0101 0101	0
0000 0000	0

La suma de los bits que son unos, contando datos y bit de paridad dará siempre como resultado un número par de unos.

Paridad impar: el número de unos (datos+paridad) siempre debe ser impar.

Dato	Paridad
0000 0001	0
0101 0001	0
0101 0101	1
0000 0000	1

Puede existir el caso en que, por ejemplo, se alteren dos bits en un caracter transmitido y si se ha implementado la comprobación de paridad, el error no será detectado.

El **método *checksum*** puede ser utilizado en la transmisión síncrona como en la asíncrona. Se basa en la transmisión, al final del mensaje, de un byte (o bytes) cuyo valor sea el complemento a dos de la suma de todos los caracteres que han sido transmitidos en el mensaje. El receptor implementará una rutina que suma todos los bytes de datos recibidos y al resultado se le sumará el último byte (que posee la información en complemento a dos de la suma de los caracteres transmitidos), si la recepción del mensaje ha sido correcta, el resultado debe ser cero.

1.8. Características de la norma de comunicación serie RS232.

La norma RS232 es una de las más populares que se utilizan en la comunicación serie, y es la que se utiliza las computadoras, si bien hoy día está ampliamente superada por la transmisión serie a través de USB, de manera que está remitiendo su uso (por ejemplo, ya no se implementa en ordenadores portátiles). Se desarrolló en la década de los 60 para gobernar la interconexión de terminales y MODEM.

La norma RS232 resuelve tres aspectos en la comunicación que se establece entre el **DTE**, Equipo Terminal de Datos, por ejemplo un PC y el **DCE**, Equipo para la comunicación de datos, por ejemplo el microcontrolador:

1. **Características eléctricas de la señal:** Se establece que la longitud máxima entre el DTE y el DCE no debe ser superior a los 15 metros y la velocidad máxima de transmisión es de 20.000 bps. Los niveles lógicos no son compatibles

TTL, considerando:

a. 1 lógico entre -3V y -15V.
b. 0 lógico entre +3V y +15V.

2. **Características mecánicas de los conectores:** Se utiliza un conector 25 pines, DB 25, o de 9 pines, DB 9, donde el conector macho identifica al DTE y el conector hembra al DCE.
3. **Descripción funcional de las señales usadas:** Las señales están básicamente divididas en dos grupos:

a. Señales primarias, que son normalmente utilizadas para las transferencias de datos.
b. Señales secundarias, utilizadas para el control de la información que será transferida.

La norma RS232 está definida tanto para la transmisión sincrónica como para la asincrónica.

Velocidad de transmisión.

La velocidad está estandarizada según la norma RS 232C en baudios:

a. 75
b. 110
c. 150
d. 300
e. 600
f. 1200
g. 2400
h. 4800
i. 9600
j. 19200

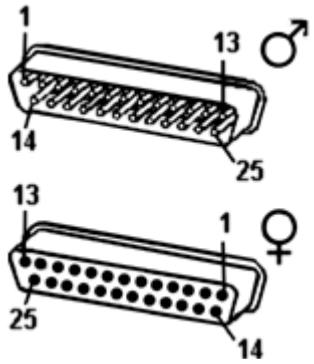


Figura 1.12.Conectores macho y hembra DB25 .

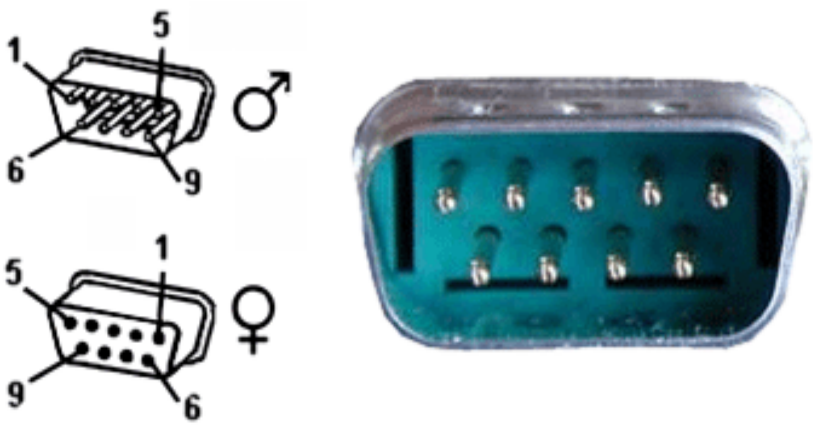


Figura 1.13.Conectores macho y hembra DB9.

Ambos conectores son totalmente compatibles entre sí y existen adaptadores para pasar de un conector a otro[5].

Descripción de los terminales.

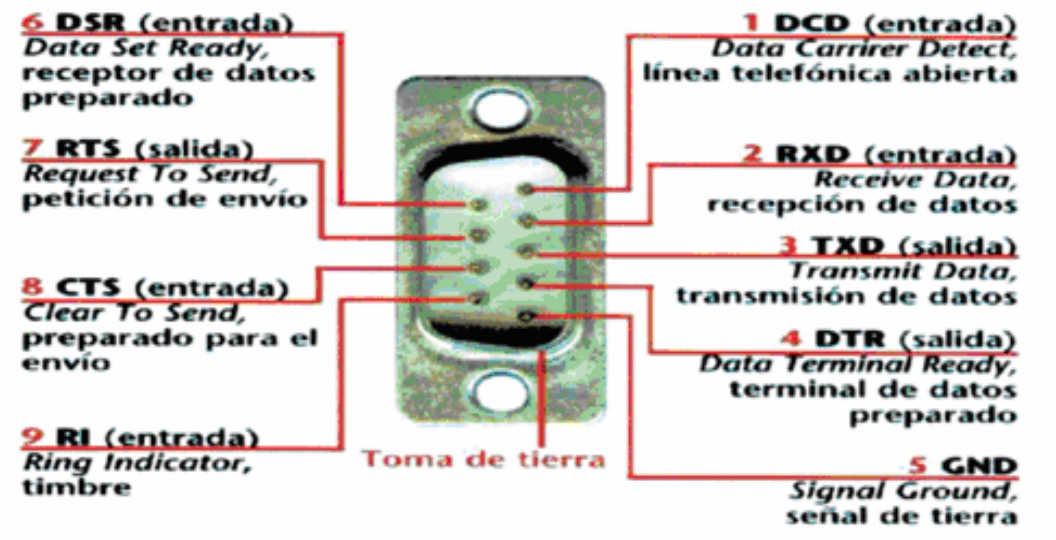


*Figura 1.14.*Descripción *de los terminales de un conector DB9.*

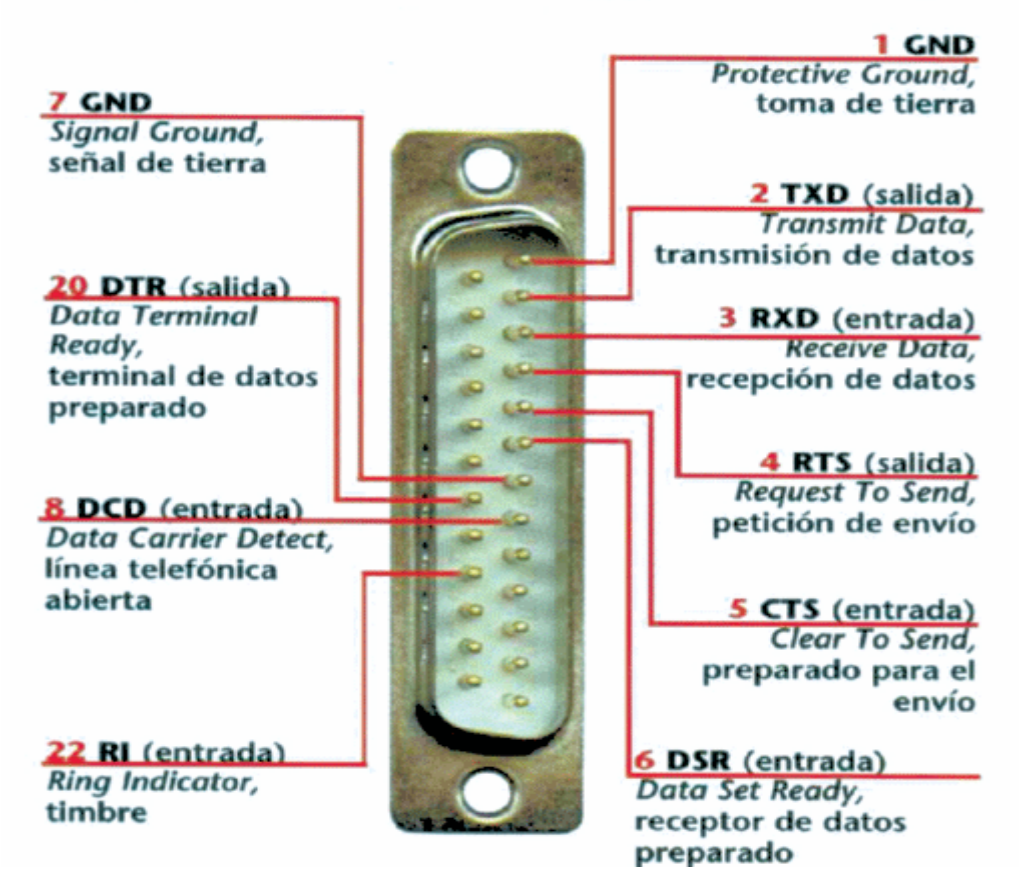


Figura 1.15.Descripción de los terminales de un conector DB25.

Para ilustrar mejor el significado de cada terminal, se describirá el conector terminal DB25 (Figura 1.15) [5].

- **TXD (Transmit Data, transmisión de datos, salida, pin. 2):** Señales de datos que se transmiten del DTE al DCE. En principio, los datos no se pueden transmitir si alguno de los terminales RTS, CTS, DSR ó DTR está desactivado.
- **RXD (Receive Data, recepción de datos, entrada, pin. 3):** Señales de datos transmitidos desde el DCE al DTE.
- **DTR (Data Terminal Ready, terminal de datos preparado, salida, pin. 20):** Señal del DTE que indica que está conectado, generalmente en "0" indica que el DTE está listo para transmitir o recibir.
- **DSR (Data Set Ready, dispositivo preparado, entrada, pin. 6):** Señal del DCE que indica que el dispositivo está en modo de transmisión de datos.
-**RTS (Request To Send, petición de envío, salida, pin. 4):** Señal del DTE al DCE, notifica al DCE que el DTE dispone de datos para enviar. Se emplea en líneas semiduplex para controlar la dirección de transmisión. Una transición de 1 a 0 avisa al DCE que tome las medidas necesarias para prepararse para la transmisión.
- **CTS (Clear To Send, preparado para transmitir, entrada, pin. 5):** Señal del DCE al DTE indicando que puede transmitirle datos.
- **CD (Carrier Detect, detección de portadora, entrada, pin. 8):** Señal del DCE que ha detectado la señal portadora enviado por un modem remoto o que la línea telefónica está abierta.

- **RI (Ring Indicator, timbre o indicador de llamada entrante, entrada, pin. 22):** Señal del DCE indicando que está recibiendo una llamada por un canal conmutado.
- **SG (GND) (System Ground ó Signal Ground, masa de señal, pin. 7):** Masa común para todos las líneas.
- **FG (GND) (Shield ó Protective Ground, tierra de protección, pin. 1):** El conductor esta eléctricamente conectado al equipo. Una secuencia normal, a través de la RS232, es la siguiente:

1. Ambos dispositivos son alimentados, indicando encendido (si ha sido establecido en el equipo). El DTE activa el terminal DTR y el DCE activa el terminal DSR. Una interfase RS232 bien diseñada no comunicará hasta que estos dos terminales estén activos. El DTE esperará la activación del terminal DSR y el DTE la activación del terminal DTR. Aunque DTR y DSR algunas veces pueden ser utilizados para el control del flujo, estos terminales solo indican que los dispositivos están conectados.

2. El DTE pregunta al DCE si este está listo. El DTE activa la línea RTS. El DCE si está listo, responde activando la línea CTS. Puestos de acuerdo ambos equipos, se puede entrar a comunicar.

3. Los datos son transferidos en ambos sentidos. El DTE envía información al DCE a través del terminal TXD. El DCE envía información al DTE a través del Terminal RXD.

Interfaz TTL-RS232.

Para una comunicación full duplex (Figura 1.16 y 1.17) desde la USART de un microprocesador o microcontrolador deben conectarse un mínimo número de señales, concretamente TXD y RXD así como la masa (GND, SG o Signal Ground). Sin embargo una interfaz típica RS232 requiere al menos 7 señales [5].

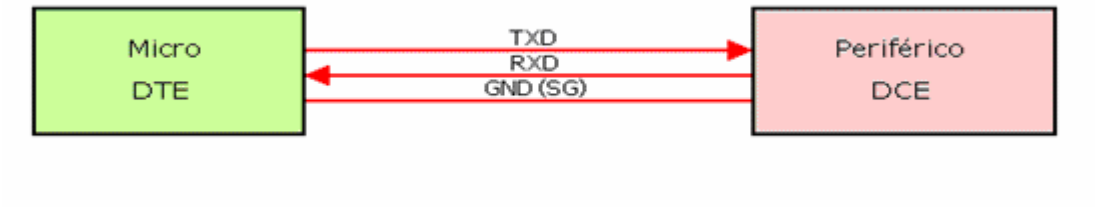


Figura 1.16.Requerimientos de señales mínimas para una comunicación duplex.

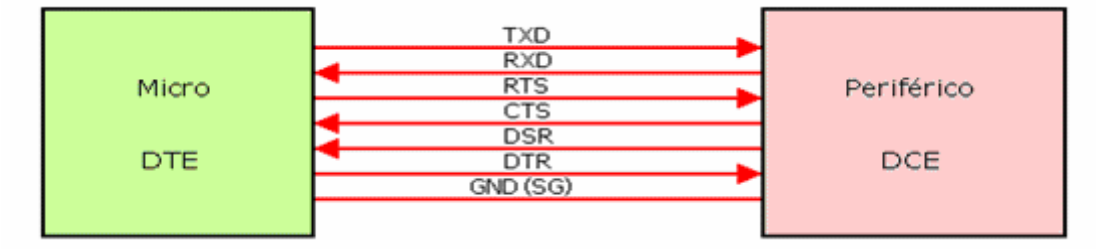


Figura 1.17.Requerimientos de señales típicas para una comunicación full duplex.

Las líneas adicionales se utilizan para la puesta de acuerdo entre el DTE y el DCE El terminal para transmitir datos (TXD) es utilizado para transferir datos del DTE al DCE, por lo que debe ser conectado a la línea receptora serie del periférico. De manera idéntica la línea receptora de datos (RXD) debe ser conectada a la línea transmisora del periférico. Para convertir TTL a RS232 se pueden usar circuitos típicos de transistores y diodos discretos o los circuitos integrados MC1488 y MC1489, sin embargo, existe un circuito integrado muy popular que permiten esta conversión. El MAX232 es un conversor de nivel TTL/RS232. Sólo es necesario este circuito integrado y 4 condensadores.

Conexión de un microcontrolador al puerto serie de una PC.
Para conectar el PC (Figura1.18) a un microcontrolador por el puerto serie se utilizan las señales TXD, RXD y GND. La PC utiliza la norma RS232, por lo que los niveles de tensión de las patillas están comprendidos entre +15 y -15 voltios. Los microcontroladores normalmente trabajan con niveles TTL (0-5v). Es necesario por tanto intercalar un circuito que adapte los niveles(es muy conveniente usar el MAX232 que se había mencionado anteriormente) [5].

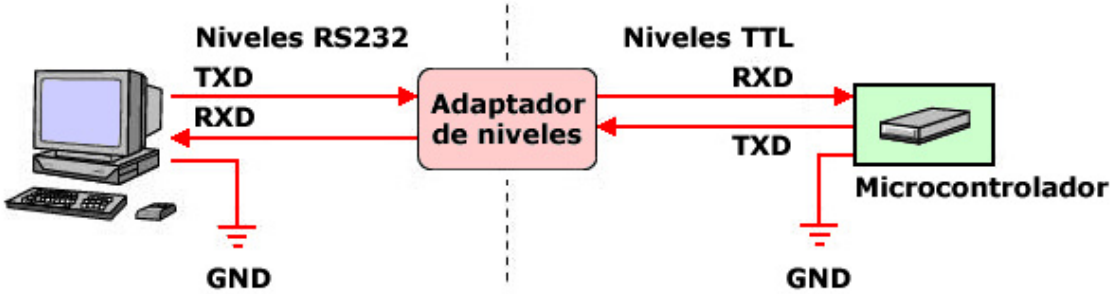


Figura 1.18.Conexión de una PC con un microcontrolador.

Cable de conexión.
Para realizar la conexión entre el PC y un microcontrolador circuito se puede usar diferentes alternativas. Una manera es utilizar un cable serie macho-hembra no cruzado, y en el circuito un conector hembra DB9 para circuito impreso (Figura 1.19).

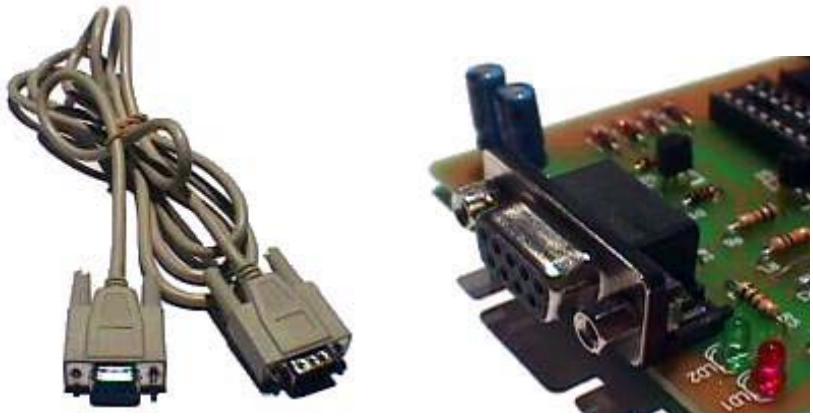


Figura 1.19.Cable de conexión.

En la placa de circuito impreso donde se encuentra el PIC y donde se colocará el conector DB9 hembra sería conveniente realizar la interconexión entre pines que se describe en la siguiente figura (1.20) [5].

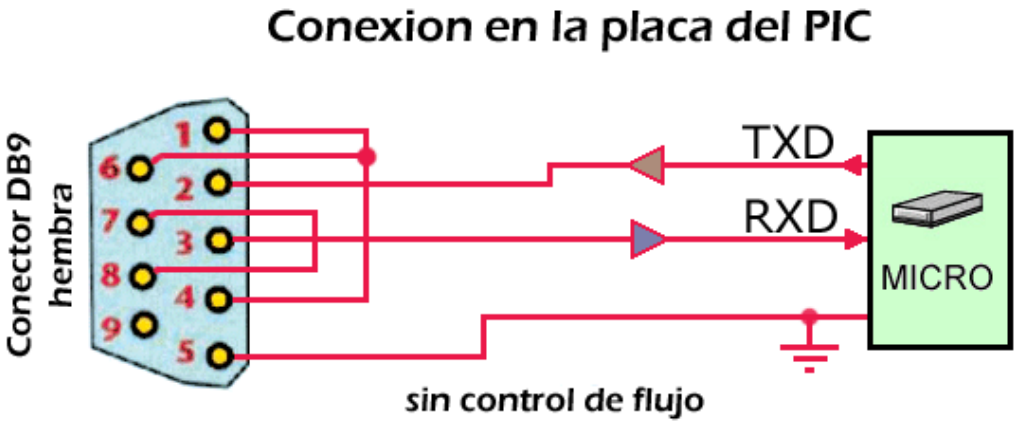


Figura 1.20. Conexión en la placa del PIC.

Conclusiones parciales.

Al caracterizar los sistemas de adquisición de datos se observa que resulta necesaria la presencia de sensores o transductores que midan las señales analógicas, una etapa de conversión de las señales analógicas en digitales, un protocolo de transmisión, y su recepción por una PC.

El microcontrolador cumple con los requerimientos para diseñar el sistema de adquisición de datos ya que posee los canales necesarios para la medición de los parámetros eléctricos en las pequeñas centrales hidroeléctricas.

El protocolo de transmisión serie RS232 se adecua a las necesidades de la transmisión de datos por su fácil implementación ya que existen en los compiladores utilzados, librerías para su uso. Además, los cables de conexión son relativamente fáciles de obtener.

Capitulo 2: Diseño del sistema de adquisición de datos.

En este capitulo se proponen los diseños de los circuitos acondicionadores para la medición de frecuencia , voltaje , corriente eléctrica, de una pequeña central hidroeléctrica que van acoplados al microcontrolador PIC18F4520,el circuito para acoplar la trasmisión serie entre la PC y el microcontrolador, así como los algoritmos correspondiente para la programación del microcontrolador y la programación en el LabWindows.

2.1 Esquema en bloques del hardware.

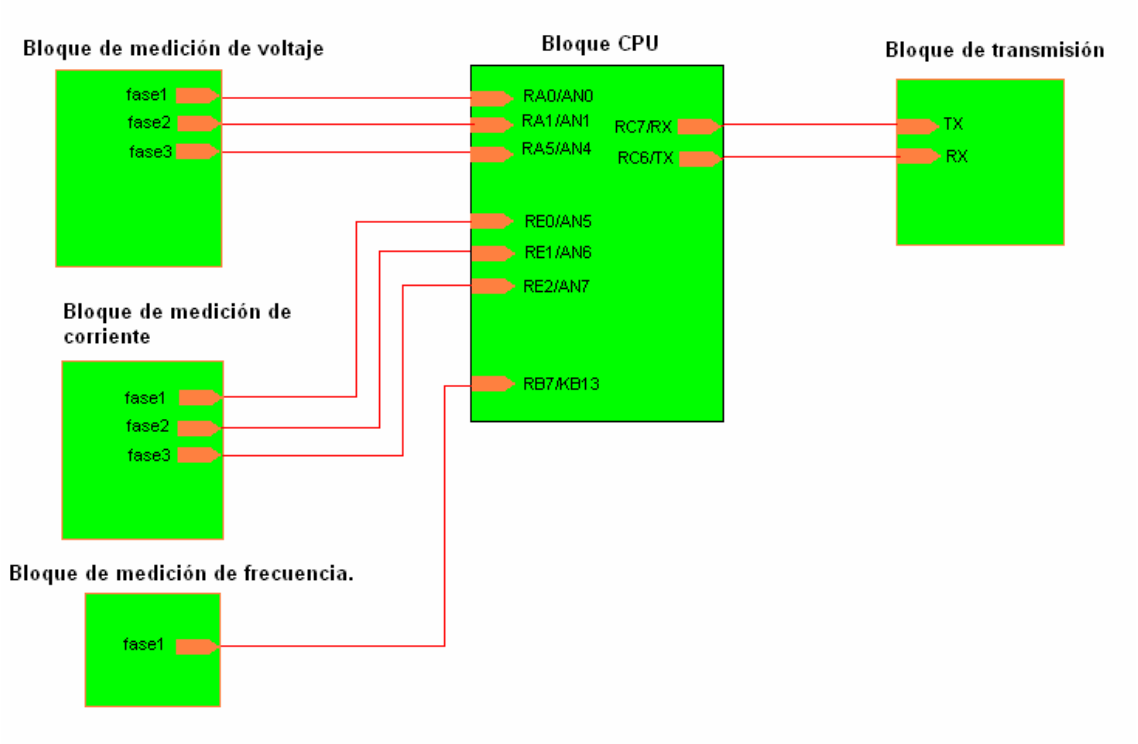


Figura 2.1.Esquema en bloques del sistema de adquisición de datos.

El sistema de adquisición de datos propuesto está constituido por los bloques funcionales siguientes:

- Circuitos acondicionadores para la medición de voltaje.
- Circuitos acondicionadores para la medición de corriente.
- Circuito acondicionador para la medición de frecuencia.
- Circuito acondicionador para la transmisión por el puerto serie.
- Microcontrolador PIC18F4520.

2.2 Fuentes de alimentación.

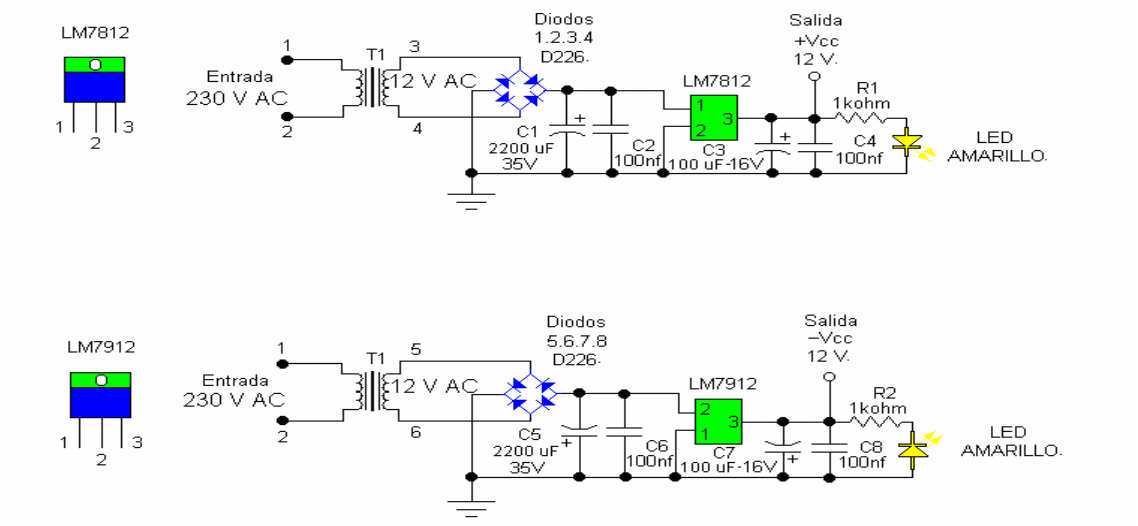


Figura 2.2.Fuentes de alimentación.

La fuente de alimentación, (Figura 2.2) garantiza las tensiones de 12 y -12 V necesarias para alimentar los dispositivos que lo requieren. La fuente de alimentación se conecta entre una línea y el neutro del sistema de tensión alterna a través de un transformador reductor para la adaptación, de los niveles de voltaje requeridos por los circuitos electrónicos. Esta tensión es rectificada, filtrada y estabilizada. En las fuentes de alimentación se utilizan estabilizadores de voltajes LM 7812 para el voltaje positivo y LM 7912 para el negativo. Los capacitores a la entrada del regulador pueden ser electrolíticos de 1000 µF y de polietileno de 100 nF, para el filtraje de la señal rectificada. Los capacitores de salida del regulador pueden ser no electrolíticos de 10 µ F y 100 nF, en paralelo, para mejorar la respuesta transitoria y el rechazo al ruido. Para las fuentes de 5V y -5V se emplean los mismos esquemas con estabilizadores de tensión LM7805 y LM.

2.3. Circuito acondicionador para la medición de frecuencia.

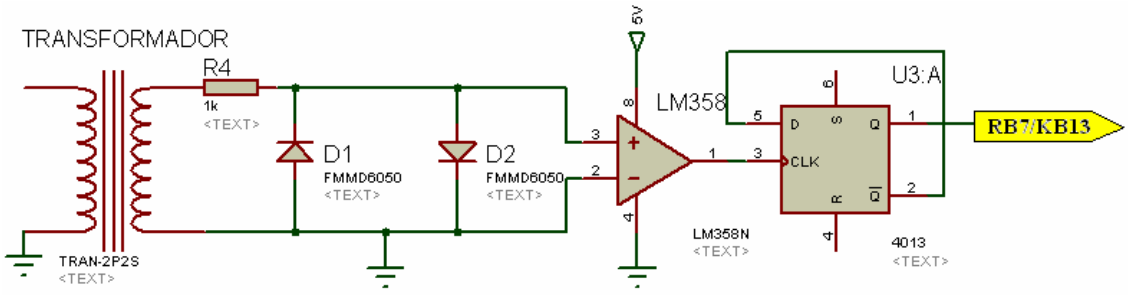


Figura 2.3.Circuito acondicionador para la medición de frecuencia.

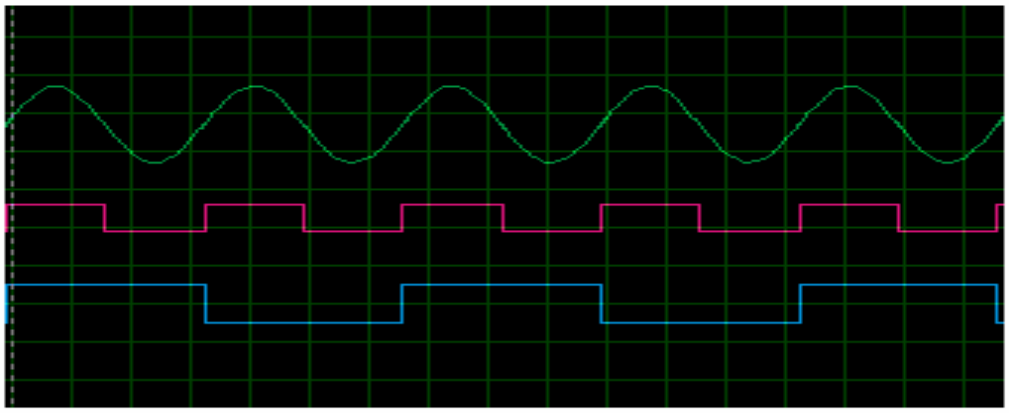


Figura 2.4.Oscilograma.

El circuito acondicionador de la señal para la medición de frecuencia, (Figura 2.3) toma una muestra de la señal de la tensión de la red a través de un transformador reductor con relación U_F/5 V, el cual se conecta entre una línea cualquiera y el neutro. La señal del secundario del transformador se aplica a la entrada de un comparador (LM 358) a través de una resistencia R4. A la entrada del comparador se conectan dos diodos, en paralelo opuesto, para limitar la tensión a la entrada del comparador. A la salida del comparador se tiene un tren de pulsos rectangulares con nivel lógico 1 o 0, según la polaridad de la señal alterna que se muestrea. La señal de salida pasa a un biestable tipo D (CI 4013) con el objetivo de dividir por dos la frecuencia del tren de pulsos y por consiguiente obtener nivel alto con duración igual al período de la señal de entrada y nivel bajo de igual duración.

2.4. Circuito acondicionador para la medición de voltaje.

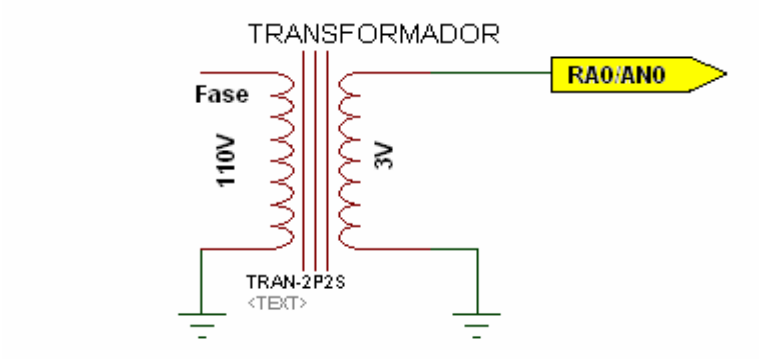


Figura 2.5.Circuito acondicionador para la medición de voltaje.

Para la medición de voltaje se utiliza simplemente un transformador por cada fase (Figura 2.5) el cual reduce el voltaje de la misma a 3V como voltaje nominal. Los voltajes de cada fase irán a los pines RA0/AN0, RA1/AN1 y RA5/AN4 del microcontrolador.

2.5. Circuito acondicionador para la medición de corriente.

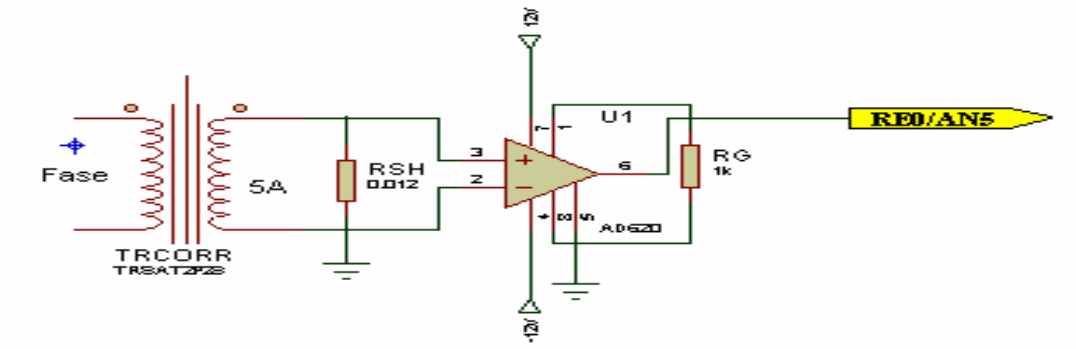


Figura 2.6.Circuito acondicionador para la medición de corriente.

Para la medición de corriente se usa un transformador de corriente para cada fase, con relación de transformación Ki=5, debido a que en mini hidroeléctricas la corriente nominal es aproximadamente 25A, por lo tanto se reduce la misma a 5A, luego con una resistencia Shunt de 0.012 ohm en paralelo, se transforma en un voltaje de 60 mV , se utiliza una resistencia tan pequeña porque los transformadores de corriente deben trabajar en modo de cortocircuito, luego con un amplificador de instrumentación AD620 con ganancia G=50 se eleva el voltaje a 3V, para las fases 1,2 y 3 se distribuyen dichas señales por los pines RE0/AN5 , RE1/AN6 y RE2/AN7 respectivamente para su posterior conversión en señales digitales (Figura 2.6).

2.6. Circuito acondicionador para la transmisión por el puerto serie.

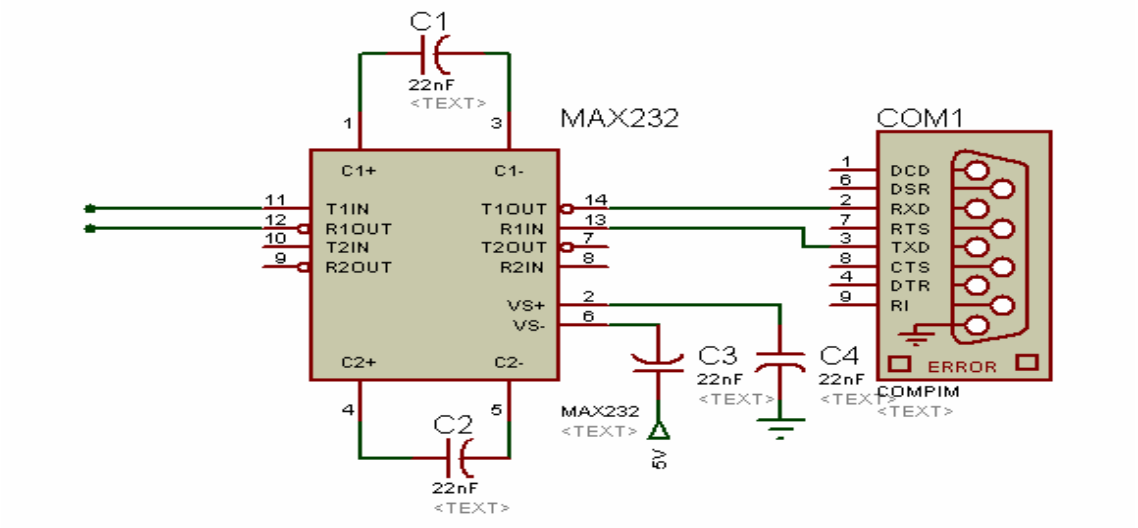


Figura 2.7.Circuito acondicionador para la comunicación serie.

Para conectar la PC a un microcontrolador por el puerto serie se utilizan las señales TXD, RXD y GND. La PC utiliza la norma RS232, por lo que los niveles de tensión de los pines están comprendidos entre +15 y -15 voltios. Los microcontroladores normalmente trabajan con niveles TTL (0-5v). Es necesario por tanto intercalar un circuito que adapte los niveles, para eso se utiliza el MAX232 y un conector DB9.

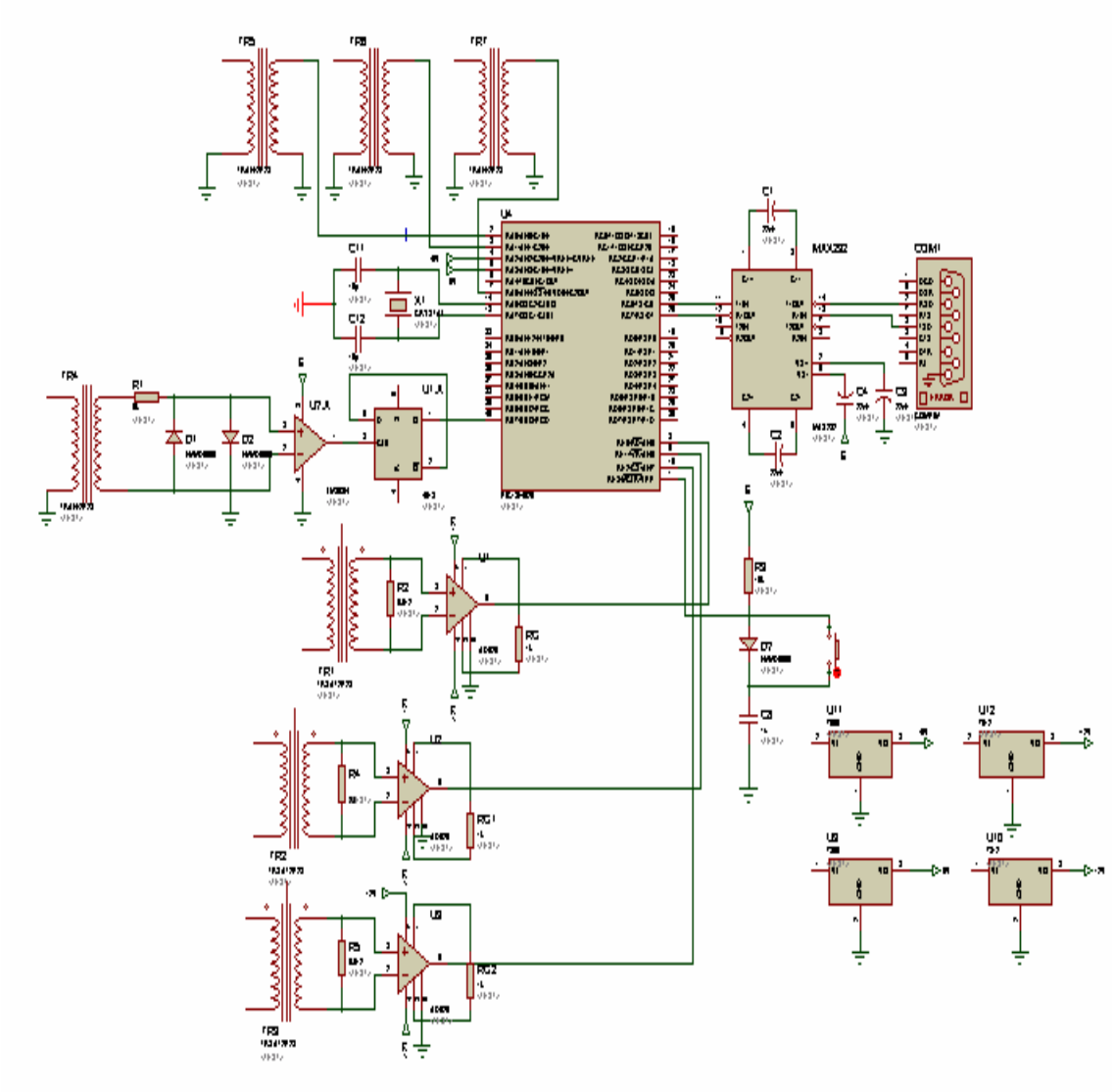


Figura 2.8. Diseño general.

2.7. Diseño del Software.

La familia de los microcontroladores PICs tienen la ventaja de que estos pueden ser programados en lenguaje ensamblador o en lenguaje C, o en ambos lenguajes simultáneamente.

Esta característica hace que estos microcontroladores sean fáciles de programar, dando la posibilidad a los programadores de escoger en cual lenguaje realizar su programa.

En este caso, se diseñó un software en lenguaje C teniendo la ventaja que es más cómodo y mas fácil de utilizar.

Durante el estudio de las herramientas de programación para la gama media se ha buscado distintas herramientas gratuitas que ofrezcan soporte para el lenguaje C. Microchip ofrece los compiladores MPLAB-C17 y MPLAB-C18, limitadas a la gama alta (PIC17CXX y PIC18CXX, respectivamente).El compilador que se utilizó fue el MPLAB-C18 , debido que el microcontrolador usado es de la familia que este comprende.

Diagrama de flujo para la programación del Pic18f4520.

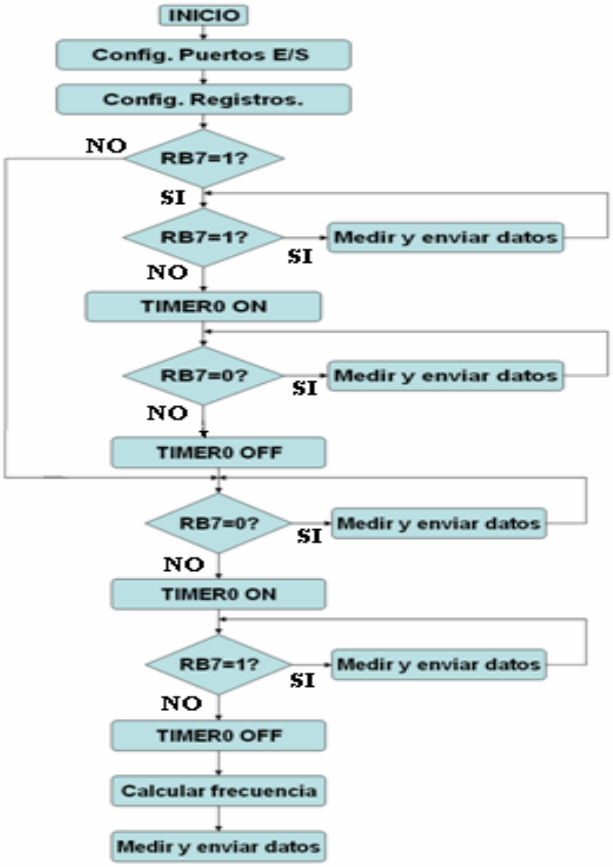


Figura 2.9.Algoritmo de medición.

Configuración de los puertos de entrada/salida: Esta función tiene como objetivo la configuración de hardware interno del PIC, es decir la forma de trabajo de los periféricos utilizados en la aplicación así como los la forma de utilización de los pines de los puertos.

Configuración del registro TRISA:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	1	1	1	1	1	1	1

El puerto A, de 7 bits, es bidireccional y tiene asociados para su control dos registros. El registro *TRISA* que permite la configuración de sus pines como entrada (si el pin correspondiente está en “1”) o salida (si el pin correspondiente está en “0”) y el registro *PORTA* que permite el intercambio de información con los dispositivos conectados a dicho puerto mediante la lectura o escritura de este registro.

Los pines de este puerto están asociados con los canales de entrada del conversor análogo-digital (CAD). La configuración de estos pines está determinada por el registro *ADCON1 (A/D Control Register1)*. Para trabajar con los canales del conversor se deben configurar los pines del puerto en entrada

escribiendo un “1” en los bits correspondientes del registro *TRISA,* como en la aplicación se utilizarán de este puerto 5 entradas analógicas para el uso del CAD y 2 entradas digitales para el oscilador, todos los bits del registro TRISA deben ponerse a “1”.

Configuración del registro TRISB:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	X	X	X	X	X	X	X

El puerto B, de 8 bits, es bidireccional y tiene asociados para su control dos registros. El registro *TRISB* que permite la configuración de sus pines en entrada (si el pin correspondiente está en “1”) o salida (si el pin correspondiente está en “0”) y el registro *PORTB* que permite el intercambio de información con los dispositivos conectados a dicho puerto mediante la lectura o escritura de este registro. En la aplicación se usará el pin RB7 como entrada digital para la medición de frecuencia por lo que se debe poner el bit 7 de este registro a “1”, los demás bits son arbitrarios.

Configuración del registro TRISC:

Bit 7	Bit 6	Bit 5	Bit 5	Bit 2	Bit 1	Bit 0
1	0	X	X	X	X	X

El puerto C, de 7 bits, es bidireccional y tiene asociados para su control dos registros. El registro *TRISC* que permite la configuración de sus pines en entrada (si el pin correspondiente está en “1”) o salida (si el pin correspondiente está en “0”) y el registro *PORTC* que permite el intercambio de información con los dispositivos conectados a dicho puerto mediante la lectura o escritura de este registro.

En los pines 7 y 6 se encuentran las líneas de datos de recepción y transmisión del puerto serie interno del PIC, dedicados a la transferencia de datos con otros periférico, fundamentalmente con la PC, por lo que el bits 7(RX) se debe configurar como entrada y el bits 6(TX) como salida.

Configuración del registro TRISD:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
X	X	X	X	X	X	X	X

El puerto D, tiene 8 bits con buffer *Schmitt Trigger* de entrada, es bidireccional y tiene asociados para su control dos registros. El registro *TRISD* que permite la configuración de sus pines en entrada (si el pin correspondiente está en “1”) o salida (si el pin correspondiente está en “0”) y el registro *PORTD* que permite el intercambio de información con los dispositivos conectados a dicho puerto

mediante la lectura o escritura de este registro. Este puerto no se utiliza, por lo tanto es arbitraria la forma en que se configure.

Configuración del registro TRISE:

Bit 3	Bit 2	Bit 1	Bit 0
X	0	0	0

Los tres primeros pines de este puerto deben configurarse como entradas analógicas del CAD, cuando son configuradas como tales se leen como “0”.
El pin 4 es la entrada de reset externa (MCLR), y como se ha configurado esta opción mediante los bits de configuración de la CPU, no importa se como se ponga este pin.

Configuración del modo de trabajo del conversor digital/analógico:
Mediante la configuración de tres registros se puede establecer la forma de trabajo del conversor, de acuerdo con la tarea que se requiere que este realice según la aplicación, en esta se utilizan 8 canales analógicos para la conversión de datos (canales 0…7), se deben configurar los voltajes de referencia del CAD referido a RA2/Vref- =- 5V y RA3/Vref+= 5V, también la cantidad de canales a utilizar. El tiempo de adquisición que en este caso es automático seleccionando la red RC interna del conversor. Además de como será ajustados los datos al concluir la conversión, que será justificado a la derecha.

Registros a configurar con los valores:
ADCON0 = 0B00000000;
ADCON1 = 0B00110110;
ADCON2 = 0B10101111;

Configuración del modo de trabajo del USART:
Teniendo en cuenta la aplicación para la que ha sido diseñado el microcontrolador, será la forma de transferencia de datos y por consiguiente la configuración de este periférico.

Pasos para la configuración de USART en modo asincrónico:

1ro Configuración de las líneas RX y TX:

- RCSTAbits.SPEN=1; Puerto serie activado (Se configuran TX y RX como líneas de transmisión y recepción del puerto serie).

2do Establecer el modo asincrónico:

- TXSTAbits.SYNC=0

3ro Establecer la velocidad de comunicación configurando adecuadamente los valores de los bits BRGH y BRG16 de los registros TXSTA y BAUDCON y los registros SPBRGH y SPBRG, este caso la velocidad de comunicación será de 9600 baudios.

TXSTAbits. BRGH=1;
BAUDCONbits. BRG16=0;

```
SPBRGH=0x00;
SPBRG=0x81;
```

4to Habilitar la transmisión y recepción mediante los bits SREN y TXEN respectivamente.

```
RCSTAbits.SREN=0;
TXSTAbits, TXEN=1;
```

Configuración del TIMER0.

El TIMER0 puede operar como temporizador o como contador de impulsos, el modo es seleccionado con el bit TOCS del registro T0CON, en este caso el timer se configura como temporizador, poniendo este bit a ”0” , además se debe configurar como temporizador de 16 bits, con el bit T08BIT a “0” , y con pre-escala de 4 ,poniendo en los registros T0PS2…0 la combinación “001”, o sea cada 4 ciclos de máquina (Fosc/4) cambia el bit menos significativo de los registros ADRESL y ADRESH.

Algoritmo para la medición.

Para medir la frecuencia se encuesta por cualquier estado (0 ó 1) del pin RB7/KB13, se espera a que cambie el mismo para tener la seguridad que es principio del pulso, en este tiempo se realizan las mediciones de todas las variables por los distintos canales del CAD así como la transmisión de las mismas por el USART ,acto seguido se habilita el Timer0 y se vuelve a esperar que cambie el estado del bit RB7/KB13, en este intervalo se vuelve a medir y enviar las variables, transcurrido este tiempo se deshabilita el timer y para de contar, en los registros ADRESL y ADRESH se guarda el conteo que dividiendo Fosc/4*conteo*pre-escala se obtiene la frecuencia ,luego se envían por última vez las mediciones por el puerto serie, este algoritmo de medir y enviar 3 veces los datos por el puerto serie es necesario para aprovechar los intervalos de tiempo en que el microcontrolador está esperando por los cambios de estado del pin RB7/KB13 para medir la frecuencia. .

2.8. Caracterización del entorno de desarrollo LabWindows/CVI.

LabWindows/CVI es un ambiente completo de desarrollo ANSI C de la National Instruments para la creación de aplicaciones de instrumentación virtual. Se basa fundamentalmente en un entorno interactivo para desarrollo de programas y unas librerías de funciones para crear aplicaciones de adquisición de datos y control de instrumentos. LabWindows/CVI contiene además un conjunto de herramientas software para la adquisición, análisis y presentación. LabWindows/CVI viene con librerías para adquisición, análisis y visualización, una interface simplificada de edición para usuario con arrastre y colocación de elementos y herramientas automatizadas de generación de código con las cuales usted puede probar dicho código interactivamente entes de adicionarlo a su proyecto. Una gran variedad de industrias emplean LabWindows/CVI, incluyendo la militar y de defensa, telecomunicaciones, manufacturación y aeroespacial.

Las herramientas que básicamente forman este entorno de desarrollo son las siguientes:

• Un editor de interfaces gráficas de usuario (GUI).

• Una ventana para editar el código fuente de nuestros programas ANSI C.

• Paneles de funciones para la ejecución interactiva y generación de código de forma automática.

• Compilador, Reubicador y Depurador integrados para el desarrollo y mantenimiento de proyectos.

Contiene:

1. Cinco librerías para adquisición de datos:

- **Instrument library:** Esta librería contiene drivers GPIB, VXI y RS232 para instrumentos tales como osciloscopios, multímetros, generadores de funciones etc.

- **GPIB/GPIB 488.2 library**: Esta librería contiene funciones para control de instrumentación conectada a un bus GPIB.

- **Data Acquisition library**: Esta librería se proporciona con las tarjetas de adquisición de *National Instruments* y consiste en un conjunto de funciones de alto nivel que permitirá controlar dichas tarjetas.

- **RS232 library.:** Conjunto de funciones de alto nivel para comunicaciones a través del puerto serie.

- **VXI library**

2. Dos librerías para análisis de datos:

- **Formatting and I/O library:** Librería formada por un conjunto de funciones para almacenar y recuperar datos de ficheros y para manipular y formatear los datos de nuestros programas.

- **Advanced Analysis library**: Esta librería consiste en un conjunto de funciones para realizar operaciones matemáticas complejas, generación de señales, operaciones con números complejos, procesamiento de señal, probabilidad etc.

3. Una librería para presentación de datos:

- **User Interface library:** LabWindows/CVI permite crear interfaces gráficos de usuario (GUI) mediante la herramienta User Interface Editor.

4. Una librería de utilidades

- **Utility library:** Contiene funciones para realizar distintas operaciones de carácter general: temporizaciones, teclado, utilidades de manejo de ficheros, operaciones de E/S, interrupciones, memoria, sonido, etc.

5. Dos librerías para redes y comunicación entre procesos:

- **Dinamic Data Exchange (DDE) library.**

- **Transmission Control Protocol (TCP) library.**

Además, tal y como se comentó anteriormente, la librería completa ANSI C también está integrada en el entorno de desarrollo de LabWindows/CVI.

Aplicaciones en que se utiliza:

- **Real Time**: Programación para componentes dedicados a tiempo real.
- **Signal Processing**: Tratamiento avanzado de la señal.
- **Vision**: Tratamiento de las imágenes, reconocimiento de formas, OCR.
- **PID Control**: Funciones para el control.
- **SPC**: Herramientas de control estadístico de los procesos para Solaris.
- **Enterprise Connectivity**: Control estadístico de los procesos, comunicación con las bases de datos y publicación Internet [4].

Diagrama de flujo para la programación del LabWindows/CVI .

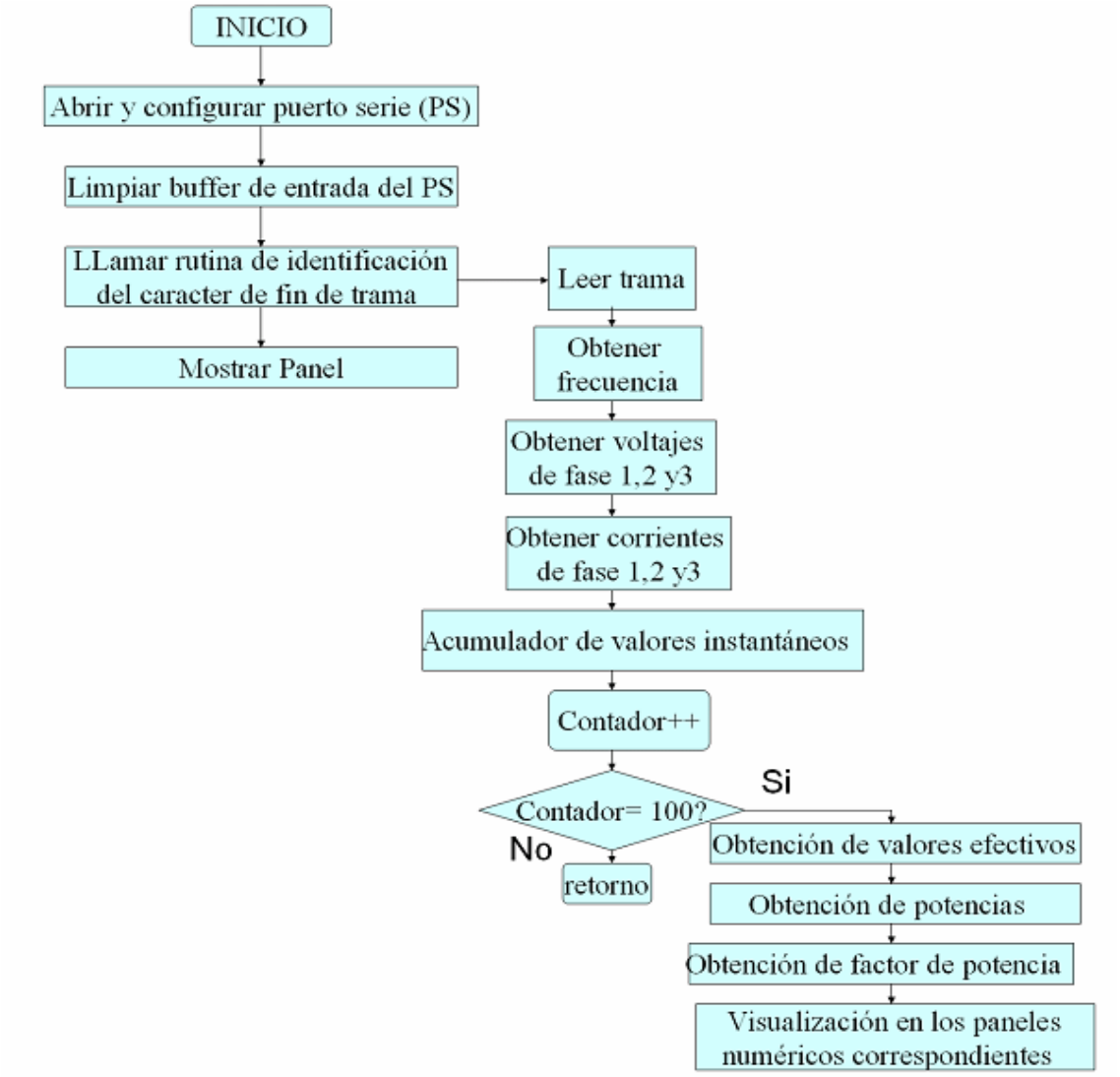


Figura 2.10.Algoritmo de cálculo y visualización.

Para la programación del LabWindows/CVI primero se se abre el puerto serie y se configura con 9600 baudios 8 bits de datos 1 bit de parada y sin paridad, se limpia el buffer de entrada luego se llama la rutina de identificación de caracter de fin de trama, esta cuando llega un caracter prefijado salta a la función Event_Char_Detect_Func que atiende este evento. En esta función se separan los valores instantáneos de la frecuencia , los voltajes y corrientes de las 3 fases, para ello se enviaron dichos valores previamente del microcontrolador separados por el caracter "!" ,estos datos se guardan como cadena de caracteres por lo que se convierten a valores numéricos, luego se van guardando estos valores instantáneos en acumuladores hasta contar 100 ,que es el número de mediciones y se obtienen los valores de voltajes y corrientes efectivas de cada fase , los valores de las potencias trifásicas y factor de potencia, luego se visualizan en los paneles correspondientes(Figura 2.9).

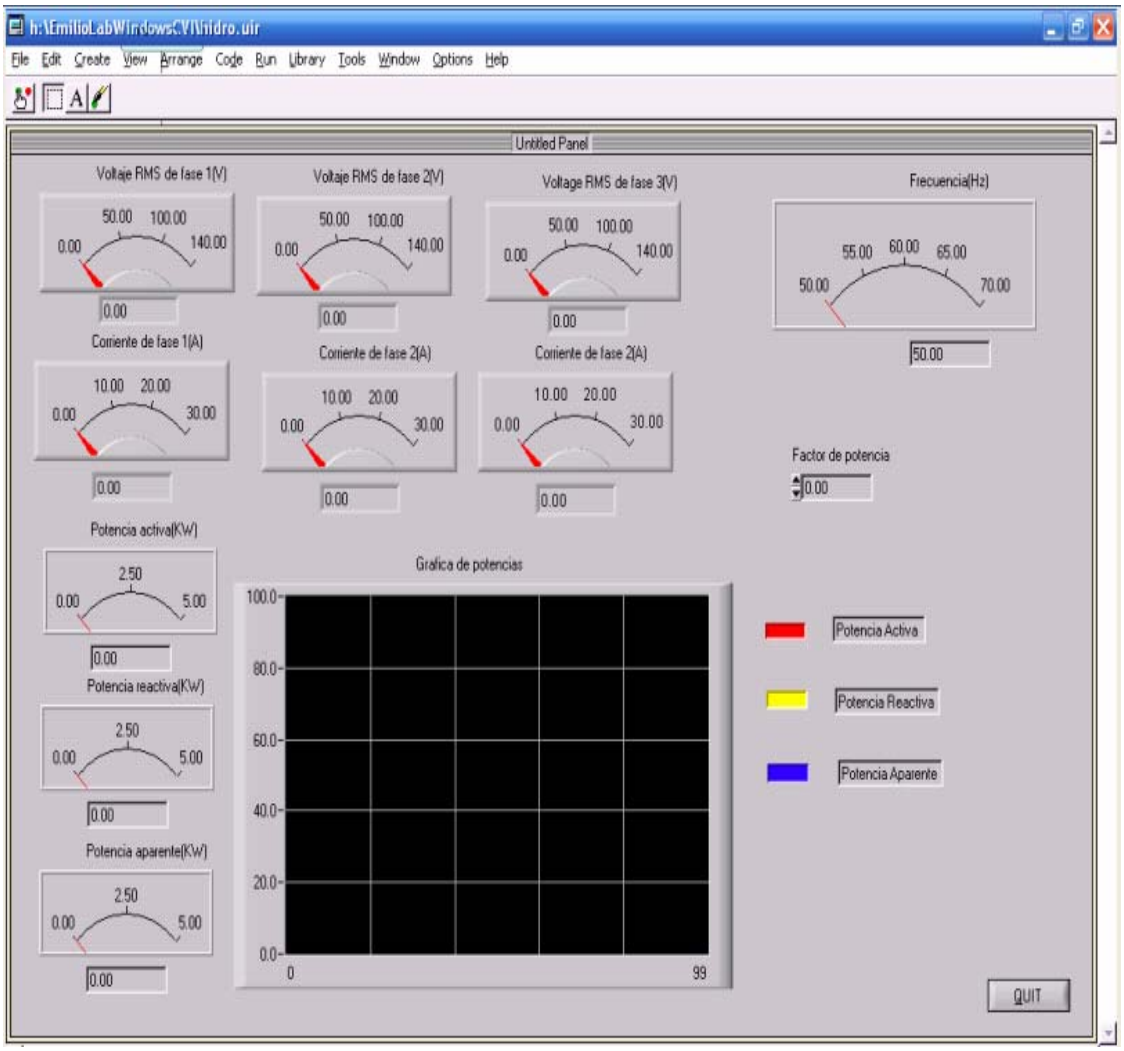


Figura 2.11.Panel de visualización.

2.9 .Valoración económica.

Con la instalación en pequeñas centrales hidroeléctricas del sistema de adquisición de datos propuesto, se puede prescindir de equipos profesionales para realizar las mediciones que encarecen la producción de energía eléctrica.

En cualquier sistema de adquisición de datos en base a microcontroladores se logran mejores resultados en sus características y desempeños y a la vez suelen ser muy baratos, la utilización de compiladores y software gratuitos reducen mucho mas el costo del proyecto, dado que los softwares en el mercado mundial ascienden a los miles de dólares.

Componentes	Cantidad	Precio por unidad	Precio sub-total
R 1K	4	0.04 USD	0.16 USD
R 10K	1	0.04 USD	0.16 USD
R 51K	1	0.04 USD	0.16 USD
C 2200 uF	4	0.12 USD	0.48 USD
C 22 nF	4	0.10 USD	0.40 USD
C 100 nF	8	0.9 USD	7.2 USD
C 22 pF	2	0.10 USD	0.20 USD
CI 4013	1	1.05 USD	1.05 USD
CI LM 358	1	2.33 USD	2.33 USD
CI LM7805	1	1.63 USD	1.63 USD
CI LM7812	1	1.82 USD	1.82 USD
CI LM7905	1	1.63 USD	1.63 USD
CI LM7912	1	1.85 USD	1.85 USD
CI PIC18F4520/JW (40)	1	27.78 USD	27.78 USD
LED AMARILLO	4	0.16 USD	0.64 USD
CXT 20MHz	1	2.23 USD	2.23 USD
D D6050	2	0.03 USD	0.6 USD
CI D Bridge	4	0.33 USD	0.33 USD
Cable con conectores DB9	1	3.71 USD	3.71 USD
Transf de corriente	3	2.85 USD	8.55 USD
Transf 110/5 V	3	2.05 USD	6.15 USD
Trans 230/ 12V	2	2.80 USD	5.60 USD
MAX232	1	4.0 USD	4.0 USD
Total de componentes	**42**		
Precio total en dólares			**78.05**

2.10. Valoración de expertos.

Conclusiones parciales.

En este capítulo se diseñó el sistema de adquisición de datos a través de la propuesta de los circuitos acondicionadores, la programación del microcontrolador PIC18F4520 y del LabWindows/CVI resultando válido en la simulación y pruebas realizadas.

Conclusiones generales.

Luego de haber dado cumplimiento a las tareas investigativas propuestas para el desarrollo de este trabajo, se concluye:

La utilización de sistemas hidroeléctricos para producir energía eléctrica tiene una serie de ventajas prácticas y económicas, siendo la principal el empleo de un recurso energético limpio, renovable y barato.

Los sistemas de adquisición de datos resultan de vital importancia no sólo para el monitoreo de los parámetros eléctricos de una pequeña central hidroeléctrica sino también para el control de los mismos.

El diseño propuesto resulta muy económico en cuanto a su elaboración comparado con lo costoso que resultan los instrumentos profesionales. Además son de fácil implementación.

Este sistema es capaz de medir, procesar y visualizar varios parámetros a la vez, gracias al uso del microcontrolador el cual incrementa también la velocidad de procesamiento.

El sistema que se propone facilita y agiliza el trabajo de supervisión en las mini-hidroeléctricas.

Recomendaciones.

- Construir el prototipo propuesto y ponerlo en funcionamiento en mini-hidroeléctricas que lo requieran.

- Realizar el cálculo de los errores introducidos por cada componente.

- Continuar perfeccionando la interfaz gráfica para la supervisión de los parámetros eléctricos.

- Utilizar otros protocolos de comunicación en dependencia de las condiciones objetivas con que se cuente en cada caso.

Bibliografía.

[1] Angulo J, Ma. Microprocesadores. Curso sobre aplicaciones en sistemas industriales. Rev.. 1984.

[2] García, L., Martínez, R., Domínguez, H., Fong, J., Pino, R. "La energía hidráulica mundial y la situación hidroenergética en Cuba". Energía y Desarrollo, no. 16. pp 29-32, CINER, Bolivia, abril 2000

[3] http://www.microchip.com

[4] Jenning . D. Welcome to the Small Scale Hydroelectric Home Page, 1997.Disponible en:

http://starfire.ne.uiuc.edu.ne201/course/old_web_projects/jennings/hydro.htm

[5] Manual de la comunicación serie formato pdf.

[6] Manual del PIC18F4520 en formato pdf, Microchip.

[7] Manuales del LabWindows/CVI v.5.0 en formato pdf, National Instruments, 1998

[8] Mañalich, R. "Manual del SOTRE". MinBas, Cuba, 1981.

[9] Martínez, I, Pastor F. Manrique M., Munoz M., Fresneda A. Manual de Minicentrales Hidroeléctricas. Cuaderno de Energías Renovables. Instituto para la Diversificación y Ahorro de la Energía (IDEA), España, 1992.

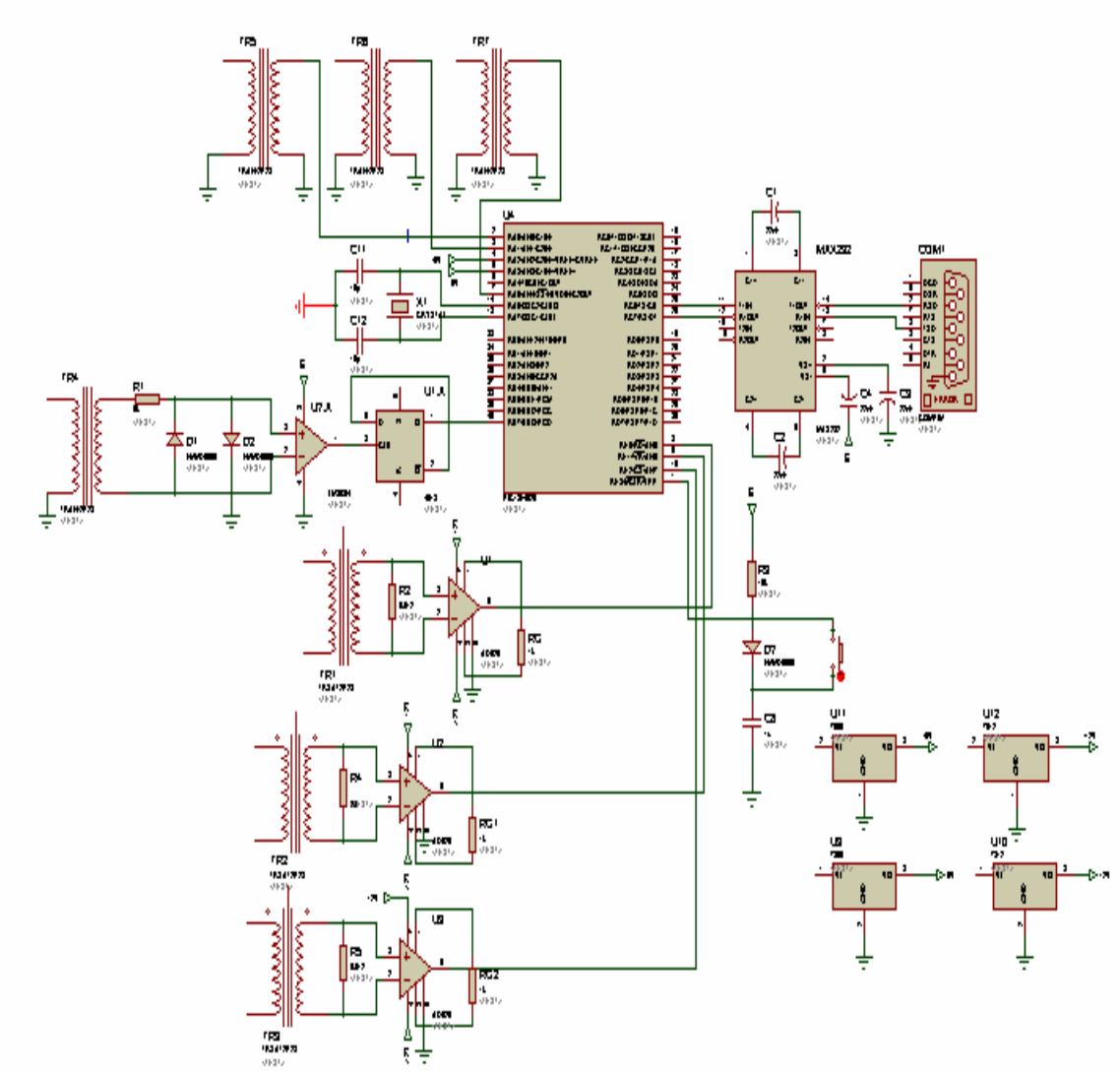
[10] Martinez,R , Fong J, Domínguez,H , Pino,R García ,Luis. Sistema de medición y monitoreo para una hidroeléctrica. Colombia 2004.

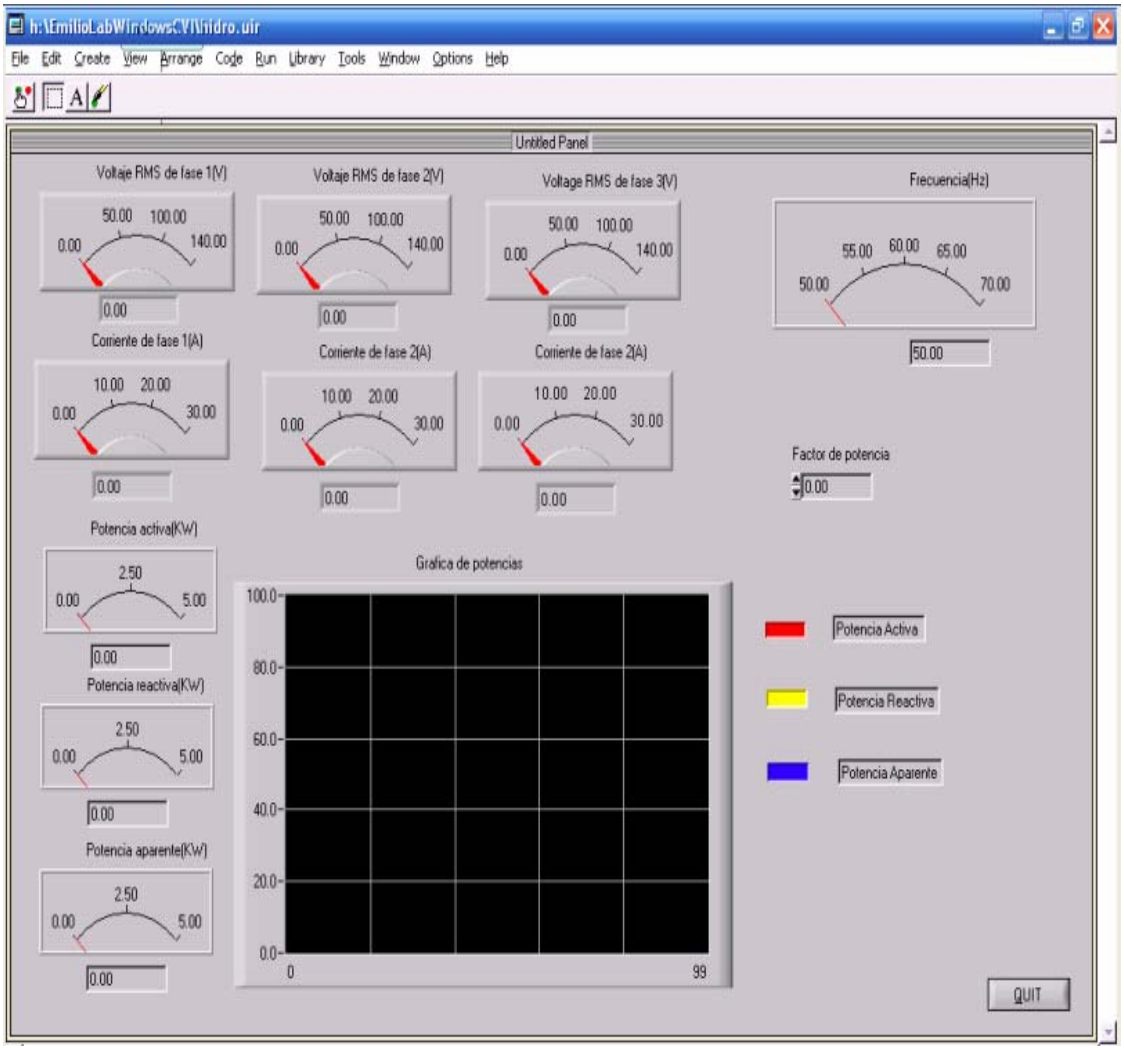
[11] NATIONAL INSTRUMENTS. "IEEE-488 Control, Data Acquisition and Analysis for Your Computer". 1990-1995

[12] PC-Lab Card, User's Manual. Adavntech Co. Ltd. 1990.

[13] Salazar, A y Fong, J. Mediciones Eléctricas. Edit. Pueblo y Educación. Cuba. 1992.

Anexos.




h:\EmilioLabWindowsCVI\nidro.uir
File Edit Create View Arrange Code Run Library Tools Window Options Help
Untitled Panel
Voltaje RMS de fase 1(V)
Voltaje RMS de fase 2(V)
Voltage RMS de fase 3(V)
Frecuencia(Hz)
Corriente de fase 1(A)
Corriente de fase 2(A)
Corriente de fase 2(A)
Factor de potencia
Potencia activa(KW)
Potencia reactiva(KW)
Potencia aparente(KW)
Grafica de potencias
Potencia Activa
Potencia Reactiva
Potencia Aparente
QUIT

```
/*********************************************************************
*************************************************************

**********************PROGRAMACIUÓN DEL PIC18F4520**********

*
        ********************************************************************
*********************************************************************/

#include "p18F4520.h"                          // definiciones de la librería
p18F4520.

#include "delays.h"                            // definiciones de la librería delays

#include "adc.h"                               // definiciones de la librería adc

#include "usart.h"                             // definiciones de la librería usart

#include "timers.h"                            // definiciones de la librería timers

#include "stdio.h"                          // definiciones de la librería stdio



/*********************************************************************
*************************************************************

*       BITS DE CONFIGURACIÓN DE LOS REGISTROS ESPECIALES


*       Micro PIC18F4520


*********************************************************************
*************************************************************/


        #pragma config OSC = HS                        //Oscillator Selection: XT

        #pragma config  OSCS = OFF          //Osc. Switch Enable:Disabled

        #pragma config  PWRT = ON             //Power-up Timer: Enabled
```

```
        #pragma config  BOR = ON          //Brown-out Reset:Enabled
    #pragma config  BORV = 42          // Brown-out Voltage:4.2V
    #pragma config  WDT = OFF          //Watchdog Timer:Disabled
    #pragma config  CCP2MUX = ON       //CCP2 MUX:Enable (RC1)
    #pragma config  STVR = ON          //Stack Overflow Reset:Enabled
    #pragma config  LVP = OFF          //  Low Voltage ICSP:Disabled
    #pragma config  DEBUG = OFF        //Background Debugger Enable: Disabled
    #pragma config  CP0 = OFF          //Code Protection Block 0:Disabled
    #pragma config  CP1 = OFF          //Code Protection Block 1: Disabled
    #pragma config  CP2 = OFF          //Disabled Code Protection Block 2:
    #pragma config  CP3 = OFF          //Code Protection Block 3: Disabled
    #pragma config  CPB = OFF          // Boot Block Code Protection: Disabled
    #pragma config  CPD = OFF          //Data EEPROM Code Protection: Disabled
    #pragma config  WRT0 = OFF         //Write Protection Block 0: Disabled
    #pragma config  WRT1 = OFF         // Write Protection Block 1:Disabled
    #pragma config  WRT2 = OFF         //Write Protection Block 2: Disabled
    #pragma config  WRT3 = OFF         //Write Protection Block 3: Disabled
    #pragma config  WRTB = OFF         //Boot Block Write Protection: Disabled
    #pragma config  WRTC = OFF         //Configuration Register Write Protection:
Disabled
    #pragma config  WRTD = OFF         // Data EEPROM Write Protection: Disabled
    #pragma config  EBTR0 = OFF        // Table Read Protection Block 0: Disabled
    #pragma config  EBTR1 = OFF        //Table Read Protection Block 1: Disabled
    #pragma config  EBTR2 = OFF        //Table Read Protection Block 2: Disabled
    #pragma config  EBTR3 = OFF        //Table Read Protection Block 3:Disabled
    #pragma config  EBTRB = OFF        // Boot Block Table Read Protection: Disabled
```

```
//*****************************************************************//
 //    **********************//Variables globales//**************
//*****************************************************************
unsigned int frecuencia ;
float volt_LSB = 0.17,corr_LSB=0.03;
int  volt_fase1,volt_fase2,volt_fase3,corr_fase1,corr_fase2,corr_fase3;
unsigned long int frecH=0,frecL=0,fr=0,fr1=0;
void Medir_Enviar (void);       //Fución medir y enviar datos
void SetupHardInterno (void);  //Configurar hardware interno
//*****************************************************************
*****************************************************************
//*       FUNCIÓN: SetupHardInterno
                                *
//*****************************************************************
*****************************************************************
void SetupHardInterno (void)
{       //INICIALIZACION DE LOS PINES DE LOS PUERTOS Y REGISTROS////
        TRISA= 0b11111111;                      //puerto A como entrada
        TRISB= 0b10000000;                      //RB7 como entrada
        TRISC= 0b10000000;      //RC6/TX Se configura como salida,
                                        //RC7/RX Se configura como entrada
        TRISD= 0b000000000;
        TRISE= 0b0000;                          // Puerto E como entrada


        //INICIALIZACION DE LOS TIMERS
```

```
        T0CON=0b00000001;           // TIMER0 como temporizador y pre- escala 4

          //INICIALIZACION DEL USART

          RCSTAbits.SPEN=1;       //Puerto serie activado(Se configuran TX y RX

                                        //como lineas del puerto serie)

    OpenUSART( USART_TX_INT_OFF             //Se config. el USART sin
interrupcion por TX

          &USART_RX_INT_OFF                       //Se config. el USART sin
interrupcion por RX

          &USART_ASYNCH_MODE                      //Se config. el USART en
modo Asincronico

          &USART_EIGHT_BIT                    //Se config. el USART con 8 Bits
sin paridad y un bits de parada

          &USART_CONT_RX                          //Se config. el USART con
recepcion contunua de datos

          &USART_BRGH_HIGH,129 );             //Se config. el USART a 9600
baud

          //INICIALIZACION DEL ADC


          ADCON0 = 0B00000000; //Configurado el canal 0, conversor habilitado.

          ADCON1 = 0B00110111; //Configuración voltage de referencia en RA2 y RA3

                                                  //y 8 canales analogicos,

    ADCON2 = 0B10101110;          //Justificación a la derecha,red RC interna,12 TAD

}
```

```
//*********************************************************************
******//

 //*                FUNCIÓN Medir_Enviar

/*********************************************************************
*****//

void Medir_Enviar (void)

{

        ADCON0bits.ADON=1;          // se habilita el ADC

       ConvertADC();                    //Inicio de        Conversión

        while (BusyADC());              //esperar fin de conversión

        volt_fase1=ReadADC()*volt_LSB-179.6; // calcular voltaje de fase 1


       SetChanADC( ADC_CH1 );              //  Seleccionar canal 1

          ADCON0bits.ADON=1;

        ConvertADC();

        while (BusyADC());

        volt_fase2=ReadADC()*volt_LSB-179.6;


        SetChanADC( ADC_CH4 );

          ADCON0bits.ADON=1;

        ConvertADC();

        while (BusyADC());

        volt_fase3=ReadADC()*volt_LSB-179.6;

        SetChanADC( ADC_CH5 );

              ADCON0bits.ADON=1;
```

```
        ConvertADC();
        while (BusyADC());
        corr_fase1=ReadADC()*corr_LSB-35;

        SetChanADC( ADC_CH6 );
        ADCON0bits.ADON=1;
        ConvertADC();
        while (BusyADC());
        corr_fase2=ReadADC()*corr_LSB-35;

        SetChanADC( ADC_CH7 );
        ADCON0bits.ADON=1;
        ConvertADC();
        while (BusyADC());
        corr_fase3=ReadADC()*corr_LSB-35;
        CloseADC();                                  // cerrar conversor.

printf("%d",frecuencia);        //enviar frecuencia.
printf("!");                    //enviar caracter de separación.
printf("%i",volt_fase1);        //enviar voltaje fase1.
printf("!");                    //enviar caracter de separación
printf("%i",volt_fase2);   //enviar voltaje fase 2.
printf("!");                    //enviar caracter de separación
printf("%i",volt_fase3);         //enviar voltaje fase 3.
printf("!");                                 //enviar caracter de separación
```

```
printf("%i",corr_fase1);         //enviar corriente de fase 1.

printf("!");                              //enviar caracter de separación

printf("%i",corr_fase2);        //enviar corriente fase 2.

printf("!");                              //enviar caracter de separación.

printf("%i",corr_fase3);   //enviar corriente de fase 3.

printf("!");                       //enviar caracter de separación.

printf("A");                       //enviar caracter de fin de trama.


}


/*******************************************************************
***********************************************************


MAIN

                    *****
       ************************************************************
****************************************************************/


void main (void)

{




while (1)


{ SetupHardInterno ();

  if(PORTBbits.RB7)
```

```
{
TMR0H=0x00;                    //limpiar registros  del timer
TMR0L=0x00;
while(PORTBbits.RB7)           //esperar por el canbio de estado(1 a 0)
Medir_Enviar();                        // medir y enviar datos
T0CONbits.TMR0ON=1;        //habilitar TIMER0
while(PORTBbits.RB7==0)        //esperar cambio de estado para la medición del
periodo del pulso
Medir_Enviar();                        //medir y enviar datos.
T0CONbits.TMR0ON=0;        //deshabilitar TIMER0
}
if(PORTBbits.RB7==0)           //  Esta rutina es la misma que la anterior ,
{                              //   pero con estados del pulso diferentes
TMR0H=0x00;                     //
TMR0L=0x00;                    //
while(PORTBbits.RB7==0)         //
Medir_Enviar();                //
T0CONbits.TMR0ON=1;            //
while(PORTBbits.RB7)           //
Medir_Enviar();                //
T0CONbits.TMR0ON=0;            //
}                              //
        frecL=TMR0L;                    // leer registr TMR0L;
    frecH= TMR0H*0x100;    // leer registr TMR0H y
                           //correr valor 8 bits a la derecha;
 fr=frecH+frecL;           //obtener conteo en una sola variable.
```

```
 fr1=fr*0x04;                    //multiplicar por pre-escala.
frecuencia=5000000/fr1;      //obtener frecuencia
   Medir_Enviar();               // medir y enviar datos
}



}
```

```
*********/PROGRAMACION DEL LABWINDOWS/CVI*********************

#include <ansi_c.h>

#include <formatio.h>

#include <math.h>

#include <rs232.h>

#include <cvirte.h>

#include <userint.h>

#include "hidro.h"

static int panelHandle;

void CVICALLBACK Event_Char_Detect_Func (int portNo,int eventMask,void
*callbackData);

int cont=0;

double
fact_pot_3f=0,P_react_3f=0,P_ap_3f=0,P_act_3f=0,fact_pot_fase1=0,fact_pot_
fase2=0,fact_pot_fase3=0;

double
P_react_fase1=0,P_react_fase2=0,P_react_fase3=0,P_ap_fase1=0,P_ap_fase
2=0,P_ap_fase3=0,P_act_fase1=0;

double  P_act_fase2=0,P_act_fase3=0,corr_efect_fase1=0,
corr_efect_fase2=0,corr_efect_fase3=0,volt_efect_fase1=0;

double  volt_efect_fase2=0,volt_efect_fase3=0;



int main (int argc, char *argv[])

{       if (InitCVIRTE (0, argv, 0) == 0)

                return -1;      /* out of memory */

        if ((panelHandle = LoadPanel (0, "hidro.uir", PANEL)) < 0)

                return -1;
```

```
        OpenComConfig (1, "", 9600, 0, 8, 1,30, 30);

        SetCTSMode (1, LWRS_HWHANDSHAKE_OFF);

        FlushInQ (1);

      InstallComCallback (1, LWRS_RXFLAG, 0, 65 , Event_Char_Detect_Func,
0);

        DisplayPanel (panelHandle);

        RunUserInterface ();

        DiscardPanel (panelHandle);

        return 0;

}

 void CVICALLBACK Event_Char_Detect_Func (int portNo,int eventMask,void
*callbackData)

{  char buf[200]={'\0'},frecuencia[10];

   char volt_fase1[10]={'\0'}, volt_fase2[10]={'\0'},
volt_fase3[10]={'\0'},corr_fase1[10]={'\0'},corr_fase2[10]={'\0'},

   corr_fase3[10]={'\0'};

   double
volt_fase1_conv,volt_fase2_conv,volt_fase3_conv,corr_fase1_conv,corr_fase2
_conv,corr_fase3_conv,frecuencia_conv,graf[3];

   int j=0,k=0,l=0, p=0,w=0,q=0,f=0;

   ComRdTerm(1,buf,20,65);  //*Se cargan en buf los caracteres del quenue de
entrada

    while (buf[j] != '!')

                          { frecuencia[j]= buf[j];

                            j++;

                            if(j>4)

                            break;

                          }
```

```
                j=j+1;
while (buf[j] != '!')
                { volt_fase1[f]= buf[j];
                  j++;
                  f++;
                  if(f>4)
                  break;
                }
                  j=j+1;
   while (buf[j] != '!')
                { volt_fase2[k]= buf[j];
                  j++;
                  k++;
                  if(k>4)
                  break;
                }
                 j=j+1;
   while (buf[j] != '!')
                { volt_fase3[l]= buf[j];
                  j++;
                  l++;
                  if(l>4)
                  break;
                }
                j=j+1;
```

```
while (buf[j] != '!')
{corr_fase1[p]= buf[j];
  j++;
  p++;
  if(p>4)
  break;
}
j=j+1;
while (buf[j] != '!')
{corr_fase2[q]= buf[j];
  q++;
  p++;
  if(q>4)
  break;
}
j=j+1;
while (buf[j] != '!')
{corr_fase3[w]= buf[j];
  j++;
  w++;
  if(w>4)
  break;
}
//*Converción de caracter a double.....
```

```
        frecuencia_conv=atof(frecuencia);
        volt_fase1_conv=atof(volt_fase1);
        volt_fase2_conv=atof(volt_fase2);
        volt_fase3_conv=atof(volt_fase3);
        corr_fase1_conv=atof(corr_fase1);
        corr_fase2_conv=atof(corr_fase2);
        corr_fase3_conv=atof(corr_fase3);
        //*1er paso para calcular potencia activa por cada fase//
        P_act_fase1=P_act_fase1+volt_fase1_conv*corr_fase1_conv;
        P_act_fase2=P_act_fase2+volt_fase2_conv*corr_fase2_conv;
        P_act_fase3=P_act_fase3+volt_fase3_conv*corr_fase3_conv;

          //*1er paso para calcular voltaje y corriente efectivas de cada fase//
        volt_efect_fase1=volt_efect_fase1+pow(volt_fase1_conv,2);
  volt_efect_fase2=volt_efect_fase2+pow(volt_fase2_conv,2);
  volt_efect_fase3=volt_efect_fase3+pow(volt_fase3_conv,2);
  corr_efect_fase1=corr_efect_fase1+pow(corr_fase1_conv,2);
  corr_efect_fase2=corr_efect_fase2+pow(corr_fase2_conv,2);
  corr_efect_fase3=corr_efect_fase3+pow(corr_fase3_conv,2);
  cont++;



if(cont==100)
  { //*2do paso para calcular voltaje y corriente efectivas de cada fase//
```

```
volt_efect_fase1=sqrt(volt_efect_fase1/cont);
volt_efect_fase2=sqrt(volt_efect_fase2/cont);
volt_efect_fase3=sqrt(volt_efect_fase1/cont);
corr_efect_fase1=sqrt(corr_efect_fase1/cont);
corr_efect_fase2=sqrt(corr_efect_fase2/cont);
corr_efect_fase3=sqrt(corr_efect_fase3/cont);
```

//*2do paso para calcular potencia activa por cada fase//

```
P_act_fase1=P_act_fase1/cont;
P_act_fase1=P_act_fase2/cont;
P_act_fase1=P_act_fase3/cont;
//*Calculo de potencia aparente para cada fase
P_ap_fase1=volt_efect_fase1*corr_efect_fase1;
P_ap_fase2=volt_efect_fase2*corr_efect_fase2;
P_ap_fase3=volt_efect_fase3*corr_efect_fase3;
//*Potencia reactiva para cada fase
P_react_fase1=pow(P_ap_fase1,2) - pow(P_act_fase1,2);
P_react_fase2=pow(P_ap_fase2,2) - pow(P_act_fase2,2);
P_react_fase3=pow(P_ap_fase3,2) - pow(P_act_fase3,2);
P_react_fase1=sqrt(P_react_fase1);
P_react_fase2=sqrt(P_react_fase2);
P_react_fase3=sqrt(P_react_fase3);
//* Factor de potencia para cada fase
 fact_pot_fase1= P_act_fase1/P_ap_fase1;
 fact_pot_fase2= P_act_fase2/P_ap_fase2;
 fact_pot_fase3= P_act_fase3/P_ap_fase3;
```

```
//*Potencia activa trifasica
P_act_3f=P_act_fase1+P_act_fase2+P_act_fase3;
//*Potencia aparente trifasica
P_ap_3f=P_ap_fase1+P_ap_fase2+P_ap_fase3;
//*Potencia reactiva trifasica
P_react_3f=pow(P_ap_3f,2) -pow( P_act_3f,2);
P_react_3f=sqrt(P_react_3f);
//*Factor de poencia trifasica
fact_pot_3f=P_act_3f/P_ap_3f;
```

//Mostrar valores efectivos de las 3 fases potencias trifásicas y factor de potencia

```
SetCtrlVal (panelHandle, PANEL_NUMERICMETER, volt_efect_fase1);
SetCtrlVal (panelHandle, PANEL_NUMERICMETER_2, volt_efect_fase2);
SetCtrlVal (panelHandle, PANEL_NUMERICMETER_3, volt_efect_fase3);
SetCtrlVal (panelHandle, PANEL_NUMERICMETER_4, corr_efect_fase1);
SetCtrlVal (panelHandle, PANEL_NUMERICMETER_5, corr_efect_fase2);
SetCtrlVal (panelHandle, PANEL_NUMERICMETER_6, corr_efect_fase3);
SetCtrlVal (panelHandle, PANEL_NUMERICMETER_7, frecuencia_conv);
SetCtrlVal (panelHandle, PANEL_NUMERICMETER_8, P_act_3f/1000);
SetCtrlVal (panelHandle, PANEL_NUMERICMETER_9, P_ap_3f/1000);
SetCtrlVal (panelHandle, PANEL_NUMERICMETER_10, P_react_3f/1000);
SetCtrlVal (panelHandle, PANEL_NUMERIC, fact_pot_3f);

graf[0]=P_act_3f/1000;
graf[1]=P_ap_3f/1000;
graf[3]=P_react_3f/1000;
```

```
 //Graficar potencias

 PlotStripChart (panelHandle, PANEL_STRIPCHART, graf, 3, 0, 0,
VAL_DOUBLE);

     fact_pot_3f=0;

    P_react_3f=0;

    P_ap_3f=0;

    P_act_3f=0;

    fact_pot_fase1=0;

    fact_pot_fase2=0;

    fact_pot_fase3=0;

    P_react_fase1=0;

    P_react_fase2=0;

    P_react_fase3=0;

    P_ap_fase1=0;

    P_ap_fase2=0;

    P_ap_fase3=0;

    P_act_fase1=0;

    P_act_fase2=0;

    P_act_fase3=0;

    corr_efect_fase1=0;

    corr_efect_fase3=0;

    corr_efect_fase3=0;

    volt_efect_fase1=0;

    volt_efect_fase2=0;

    volt_efect_fase3=0;
```

```
     cont=0;


    }


    return;
}
int CVICALLBACK QuitCallback (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        switch (event)
                {
                case EVENT_COMMIT:
                        QuitUserInterface (0);
                        break;
                }
        return 0;
}
```