UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

TEMA:

APLICACIONES PRÁCTICAS DE MICROCONTROLADORES A TRÁVES DE LA PLATAFORMA DE PROGRAMACIÓN VIRTUAL LABVIEW

Previa la obtención del Título

INGENIERO EN TELECOMUNICACIONES

ELABORADO POR:

Manuel Dillon Álvarez

Christian Villa Vásquez

TUTOR

MsC. Edwin F. Palacios Meléndez

Guayaquil, 30 de Agosto del 2014

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

CERTIFICACIÓN

Certifico que el presente trabajo fue realizado en su totalidad por los Sres. **Manuel Dillon Álvarez** y**Christian Villa Vásquez**como requerimiento parcial para la obtención del título de INGENIERO EN TELECOMUNICACIONES.

Guayaquil, 30 de Agosto del 2014

TUTOR

_____

MsC. Edwin Palacios Meléndez

DIRECTOR DE CARRERA

_____

MsC. Miguel A. Heras Sánchez

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

INGENIERÍA EN TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

MANUEL DILLON ÁLVAREZ

CHRISTIAN VILLA VÁSQUEZ

DECLARAMOS QUE:

El proyecto de tesis denominado “Aplicaciones prácticas de Microcontroladores a través de la plataforma de programación virtual LabView” ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros conforme las citas que constan al pie de las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de nuestra autoría.

En virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance científico del proyecto de grado en mención.

Guayaquil, 30 de Agosto del 2014

LOS AUTORES

MANUEL DILLON ÁLVAREZ

CHRISTIAN VILLA VÁSQUEZ

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

INGENIERÍA EN TELECOMUNICACIONES

AUTORIZACIÓN

Nosotros, MANUEL DILLON ÁLVAREZ

CHRISTIAN VILLA VÁSQUEZ

Autorizamos a la Universidad Católica de Santiago de Guayaquil, la publicación, en la biblioteca de la institución del proyecto titulado: "Aplicaciones prácticas de Microcontroladores a través de la plataforma de programación virtual LabView", cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría.

Guayaquil, 30 de Agosto del 2014

LOS AUTORES

MANUEL DILLON ÁLVAREZ

CHRISTIAN VILLA VÁSQUEZ

DEDICATORIA

Dedicamos este trabajo de titulación esta tesis a todas las personas que nos apoyaron, en especial a nuestros familiares quienes fueron un gran apoyo emocional durante el tiempo en que se desarrollo este trabajo de titulación.

A nuestros padres quienes nos apoyaron toda nuestra vida universitaria tanto moral como económicamente.

A nuestros maestros quienes nunca desistieron en transmitirnos sus conocimientos, aun sin importar los obstáculos que se presentaron durante todo este tiempo.

LOS AUTORES

MANUEL DILLON ÁLVAREZ

CHRISTIAN VILLA VÁSQUEZ

AGRADECIMIENTO

En primer lugar agradecemos a DIOS todo poderoso por habernos enfocado a seguir esta carrera y darnos sabiduría y fuerza para culminarla.

También agradecemos a nuestras familias en especial a nuestros padres por apoyarnos económicamente, emocionalmente, mentalmente, ya con su ayuda en el transcurso del tiempo nos motivaron a seguir nuestros estudios días tras días.

Y un agradecimiento muy especial a la Universidad Católica de Santiago de Guayaquil y sus autoridades, que nos dio la oportunidad de ser unos dignos profesionales, y a la Facultad Técnica Para el Desarrollo que en conjunto con sus excelentes docentes nos transmitieron sus conocimientos éticos y morales para un buen desenvolviendo en el campo profesional.

LOS AUTORES

MANUEL DILLON ÁLVAREZ

CHRISTIAN VILLA VÁSQUEZ

Índice General

Índice de Figuras

Capítulo 2

Capítulo 3

Capítulo 4

Índice de Tablas

Capítulo 2

Capítulo 3

Resumen

A través del presente proyecto de titulación se pretendió evidenciar la interactividad entre los microcontroladores PIC (Hardware) y la plataforma LabView (software). En la Facultad de Educación Técnica para el Desarrollo se ha utilizado en las clases de microcontroladores los dispositivos PIC'S pero no ha interactuado con alguna otra herramienta de simulación con lo es LabView. La programación o código fuente fue desarrollado bajo el lenguaje de alto nivel MikroBasic y el software Pickit 2 para cargar el archivo hex en el PIC 16F886.

CAPÍTULO 1: ASPECTOS GENERALES DEL TRABAJO DE TITULACIÓN

1.1. Introducción.

La programación virtual en Labview, así como la programación en MikroBasic de la familia de los microcontroladores tienen un crecimiento acelerado en diversidad de aplicaciones prácticas en las telecomunicaciones, telemedicina, telemática, bioingeniería, etc. Dichos avances tecnológicos permiten implementar diferentes aplicaciones que son compatibles entre LabView y los microcontroladores. La Carrera de Ingeniería en Telecomunicaciones de la Facultad de Educación Técnica para el Desarrollo (FETD) de la Universidad Católica de Santiago de Guayaquil (UCSG) cuenta con un Laboratorio de Electrónica que dispone de equipos, software y dispositivos electrónicos para realizar pruebas experimentales.

Hasta la presente fecha la programación virtual LabView, no se ha utilizado como herramienta para la enseñanza en aplicaciones afines a la Ingeniería en Telecomunicaciones. Es decir, que a través del presente trabajo de titulación se da una idea general de lo que es LabView.

1.2. Antecedentes.

A mediados del año 2009 la FETD ha venido investigando y desarrollando proyectos a través de los microcontroladores, la mayoría de proyectos han permitido que el nivel de conocimientos se incremente. De acuerdo a cambios curriculares realizados en la Carrera de Ingeniería en

Telecomunicaciones se podría considerar que tanto los Microcontroladores y las plataformas de programación MikroBasic y LabView son herramientas robustas que permiten un sinnúmero de aplicaciones.

1.3. Justificación del Problema.

A través del desarrollo de aplicaciones prácticas de programación virtual LabView con el empleo de microcontroladores, permite manejar: señales digitales (leds, botoneras, sensores infrarrojos), señales analógicas (DLR), comunicaciones OnWire, tonos (buzzer), motores DC con PWM en lazo abierto y las RPM del motor con encoder óptico.

1.4. Definición del Problema.

Hasta la presente fecha se ha venido programando en lenguajes de alto nivel para microcontroladores, sin permitir el uso de otras herramientas de programación, por eso surge la necesidad de desarrollar aplicaciones prácticas de programación virtual LabView con microcontroladores, para demostrar que estos ya guardan compatibilidad y permitirán incrementar temas de investigación a futuro.

1.5. Objetivos de la Investigación.

1.5.1. Objetivo General.

Desarrollar aplicaciones prácticas mediante programación VI en LabView con microcontroladores, para reforzar los conocimientos adquiridos y fomentar así la investigación formativa.

1.5.2. Objetivos Específicos.

1. Describir el estado del arte de la programación en LabView, MikroBasic y de los microcontroladores incluyendo a los compiladores para desarrollo de aplicaciones con lenguajes de alto nivel.
2. Diseñar las experiencias prácticas en LabView para su funcionamiento en una tarjeta de entrenamiento de microcontroladores con aplicaciones específicas.
3. Evaluar las aplicaciones prácticas mediante programación LabView.

1.6. Idea a Defender.

A través de las aplicaciones prácticas de programación virtual LabView, permitirá mejorar el proceso de aprendizaje a los estudiantes en Telecomunicaciones de la FETD, es decir, que adopten otras herramientas virtuales de programación como lo es LabView; y a la vez contribuirá a proponer temas de investigación a nivel de trabajos de titulación (pregrado y posgrado) o de proyectos de investigación semilla (SINDE).

1.7. Metodología de Investigación.

El Proyecto de Titulación utiliza el método de comprobación y de observación, conocido como cuasi experimental

CAPÍTULO 2: Estado del Arte del Entorno Virtual - LabView.

Para el presente capítulo de fundamentará al entorno virtual de LabView, el mismo que permite realizar diferentes aplicaciones en el ámbito de las ciencias aplicadas, tales como Telecomunicaciones, Electrónica, Electricidad, etc.

2.1. Introducción a LabView.

Existen diferentes definiciones del entorno virtual LabView, cuyo acrónimo es *Laboratory Virtual Instrument Engineering Workbech*, considerado como un lenguaje de programación de alto nivel orientada a objetos, es decir, programación gráfica que permite la creación de diversas aplicaciones en áreas de las Ciencias Aplicadas, Medicina, etc., de una manera sencilla y rápida.

National Instruments es la empresa desarrolladora y propietaria de LabView, fundada en 1976 en Austin, Texas. Sus primeros productos eran dispositivos para el bus de instrumentación GPIB. En abril de 1983 comenzó el desarrollo de lo que sería su producto estrella: LabView, que vería la luz en octubre de 1986 con el lanzamiento de LabView 1.0 para Macintosh (ordenadores muy populares en esa década y que disponían de interfaz gráfica) y en 1990 la versión 2. Para Windows habría que esperar a septiembre de 1992. En la tabla 1.1 se detallan los principales hitos de LabView (Lajara V. & Pelegrí S., 2011).

LabView permite recoger, analizar y monitorizar los datos dentro de un entorno de programación gráficoen el que se ensamblan objetos llamados instrumentos virtuales (VIs) paraformar el programa de aplicación con el que interactuará el usuario y que sedenomina instrumento virtual(Acevedo L. & Rueda B., 2010).

Tabla 2. 1: Historia de la evolución de la plataforma LabView.

Fecha	Hito
Abril de 1983.	Comienza el desarrollo de LabView.
Octubre de 1986.	LabView 1.0 para Macintosh.
Enero de 1990.	LabView 2.0.
Septiembre de 1992.	LabView para Windows.
Octubre de 1992.	LabView para Sun.
Octubre de 1993.	LabView 3.0 multiplataforma.
Abril de 1994.	LabView para Windows NT.
Octubre de 1994.	LabView para Power Macintosh.
Octubre de 1995.	LabView para Windows 95.
Mayo de 1997.	LabView 4.0.
Marzo de 1998.	LabView 5.0.
Febrero de 1999.	LabView 5.1, LV para Linux y LV Real – Time.
Agosto de 2000.	LabView 6i.
Enero de 2002.	LabView 6.1.
Mayo de 2003.	LabView 7 Express, LabView PDA y FPGA.
Mayo de 2004.	LabView 7.1.
Mayo de 2005.	LabView DSP.
Junio de 2005.	LabView Embedded.

Fecha	Hito
Octubre de 2005.	LabView 8.
Agosto de 2006.	LabView 8.20 (Edición especial por el 20° aniversario)
Agosto de 2007.	LabView 8.5.
Agosto de 2008.	LabView 8.6.
Agosto de 2009.	LabView 2009.
Agosto de 2010.	LabView 2010.

Fuente: **(Lajara V. & Pelegrí S., 2011)**

Originalmente la plataforma de programación gráfica LabView, fue creada y desarrollada para diferentes aplicaciones tales como electrónica, controles de instrumentación, sistemas de instrumentación, etc. (Santamaría, 2009)(Acevedo L. & Rueda B., 2010). En la figura 2.1 se muestra la ventana para crear o abrir proyectos en LabVIEW 2013.

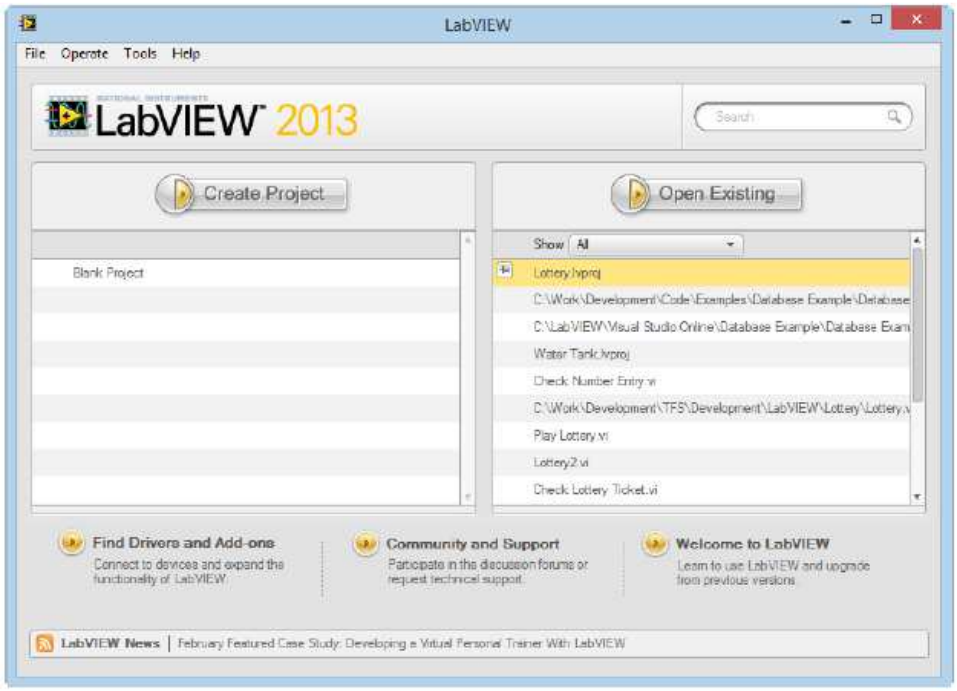

Figura 2. 1: Ventana para crear o abrir proyectos en LabVIEW 2013.
Fuente: http://www.ni.com/gettingstarted/labviewbasics/esa/environment.htm

2.2. Ventajas al utilizar LabView

Seguidamente se van a describir las ventajas de usar este tipo de lenguaje de programación:

a. La primera ventaja de usar LabView es que es compatible con herramientas de desarrollo similares y puede trabajara la vez con programas de otra área de aplicación, como Matlab o Excel. Además se puede utilizar en muchos sistemas operativos, incluyendo Windows y UNIX, siendo el código transportable de uno a otro.

b. Otra de las ventajas más importantes que tiene este lenguaje de programación es que permite una fácil integración con hardware, específicamente con tarjetas de medición, adquisición y procesamiento de datos (incluyendo adquisición deimágenes).

c. Es muy simple de manejar, debido a que está basado en un nuevo sistema de programación gráfica, llamado lenguaje G.

d. Es un programa enfocado hacia la instrumentación virtual, por lo que cuenta con numerosas herramientas de presentación, en gráficas, botones, indicadores y controles, los cuales son muy esquemáticos y versátiles. Estos serían complicados de realizar en bases como C++ donde el tiempo para lograr el mismo efecto sería muchas veces mayor.

e. Es un programa que contiene librerías especializadas para manejos de DAQ (tarjetas de adquisición de datos), Redes, Comunicaciones, Análisis Estadístico, Comunicación con Bases de Datos (útil para una automatización de una empresa a nivel total).

f. Como se programa creando subrutinas en módulos de bloques, se pueden usar otros bloques creados anteriormente como aplicaciones por otras personas.

En la tabla 2.2 se muestran cada una de las ventajas de los instrumentos virtuales sobre el instrumento tradicional.

Tabla 2. 2: Ventajas entre instrumentos virtuales y tradicional.

Instrumento Tradicional	**Instrumento Virtual**
Definido por el fabricante.	Definido por el usuario.
Funcionalidad específica, con conectividad limitada.	Funcionalidad ilimitada, orientado a aplicaciones, conectividad amplia.
Hardware es la clave.	Software es la clave.
Alto costo/función.	Bajo costo/función, variedad de funciones, reusable.
Arquitectura cerrada	Arquitectura abierta.
Lenta incorporación de nuevas tecnologías.	Rápida incorporación de nuevas tecnologías, gracias a la plataforma PC.
Bajas economías de escala, alto costo de mantenimiento.	Altas economías de escala, bajos costos de mantenimiento.

Fuente: Los Autores.

2.3. Aplicaciones de LabView.

LabView tiene su mayor aplicación en sistemas de medición, como monitoreo de procesos (como en el caso de este proyecto, ya que se representan las curvas de fluidez y viscosidad de diversos fluidos) y para aplicaciones de control. Además, LabView se utiliza bastante en el procesamiento digital de señales, en el procesamiento en tiempo real de aplicaciones biomédicas, manipulación de imágenes y audio, automatización, diseño de filtros digitales, generación de señales, entre otras, etc.

2.4. Programación de Flujo de Datos.

Molina M., J. M. & Ruiz C., A. (2010) sostienen que el lenguaje de programación utilizado en LabVIEW, también conocido como G, es un lenguaje de programación de flujo de datos. Su ejecución está determinada por la estructura gráfica de un diagrama de bloques (el código LV-fuente) en el que el programador conecta diferentes nodos de función a través de cables. Estos propagan las variables y cualquier nodo puede ejecutar tan pronto como se disponga de todos los datos de entrada.

Dado que podría darse el caso de múltiples nodos simultáneamente, G es intrínsecamente capaz de la ejecución en paralelo. El procesamiento múltiple y subprocesos múltiples de hardware es explotada de forma automática por el programador incorporado, que se encarga de multiplexar múltiples hebras de SO en los nodos listos para su ejecución.

2.5. Programación Gráfica.

LabVIEW vincula la creación de interfaces de usuario (llamados paneles frontales) en el ciclo de desarrollo. Los programas o subrutinas de LabVIEW se denominan instrumentos virtuales (VIs). Cada VI tiene tres componentes: un diagrama de bloques, un panel frontal y un panel de conectores. El último se utiliza para representar la VI en los diagramas de bloques de otra, llamado VIs.

Según lo indica Lajara y Pelegrí (2011), los controles e indicadores del panel frontal permiten al usuario introducir datos obtenidos a partir de un instrumento virtual en ejecución. Sin embargo, el panel frontal también puede servir como una interfaz de programación. Así, un instrumento virtual o bien se puede ejecutar como un programa, con el panel frontal que sirve como una interfaz de usuario, o, cuando se deja caer como un nodo en el diagrama de bloques, el panel frontal define las entradas y salidas para el nodo dado a través del panel de conectores.

2.6. Entorno Virtual.

De acuerdo a Lajara y Pelegrí (2011): LabView es una herramienta de programación gráfica. Originalmente este programa estaba orientado para aplicaciones de control de equipos electrónicos usados en el desarrollo de sistemas de instrumentación lo que se conoce como Instrumentación Virtual.Es por esto que los programas creados en LabView se guardarán en ficheros llamados VI *(Virtual Instrument)* y con la misma extensión. Los

programas no se escriben, sino que se dibujan, facilitando su comprensión. Al tener ya pre-diseñados una gran cantidad de bloques, se le facilita al usuario la creación del proyecto. Cada VI consta de dos partes diferenciadas(Palacios, 2012): Panel de control y diagrama de bloques.

2.4.1. Panel Frontal.

Esla interfaz con el usuario para el VI[1]. Según Ciscar (2010) se utiliza para interactuar con el usuario cuando el programa se está ejecutando. Los usuarios podrán observar los datos del programa actualizados en tiempo real. En esta interfaz se definen los controles e indicadores. En la figura 2.2 se muestra como ejemplo una aplicación y a la derecha el panel frontal.

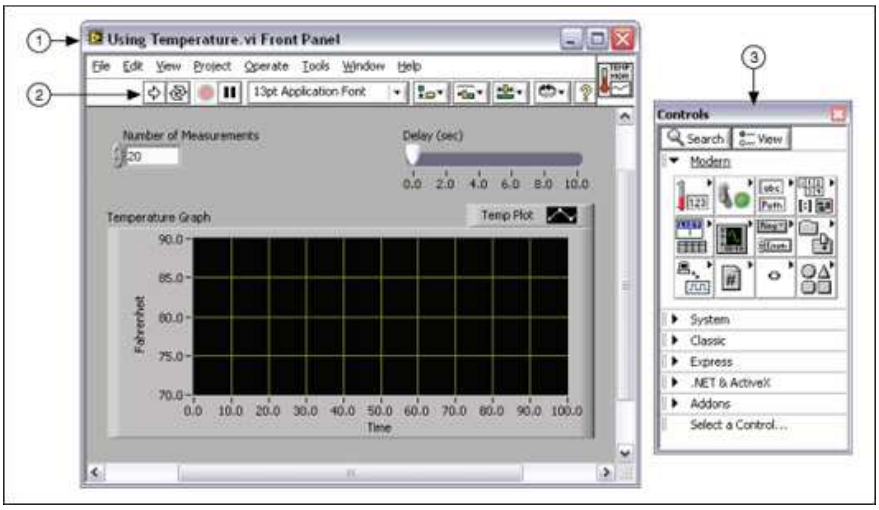

Figura 2. 2: Ventana del Panel Frontal.
Fuente: http://www.ni.com/gettingstarted/labviewbasics/esa/environment.htm

De la figura 2.2se puede ver a lado de la ventana del panel frontal otra ventana llamada *Controls*, esta ventana se la conoce como “paleta de controles”, la misma se divide en varias categorías, para lo cual el

[1] Disponible online y recuperado de la página web: http://www.ni.com/gettingstarted/labviewbasics/esa/environment.htm

programador o usuario se encarga de exponer algunas o todas las categorías (véase figura 2.3). Es decir, que la “paleta de controles” contiene varios indicadores que se visualizan en tablas, gráficos en dos dimensiones (2D) o gráficos en tres dimensiones (3D), etc., además de los controles que son booleanos, numéricos, string&path, arreglos matriciales, ring&enum, contenedores, dispositivos de entrada y salida (I/O), entre otros tales como Silver, System, Classic, Express, Control Design & Simulation, etc.

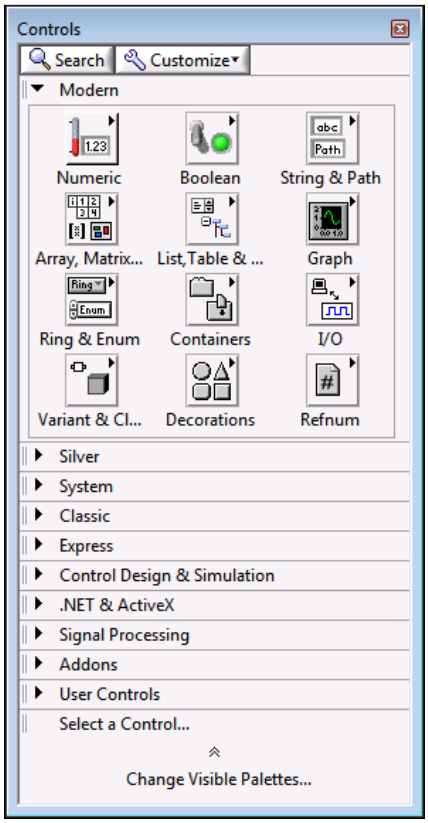

Figura 2. 3: Ventana de la paleta de controles – LabView.
Fuente: http://www.ni.com/gettingstarted/labviewbasics/esa/environment.htm

Como ya indicamos el panel frontal define los controles e indicadores, que serían las terminales de entrada y salida interactivas de la instrumentación virtual (VI). Es decir, que los controles son representados como perillas, botones, diales y demás dispositivos de entrada. Los

indicadores simulan salidas de instrumentos (representados mediante gráficas, luces y demás dispositivos de visualización) y suministra datos al diagrama de bloques del VI. Mientras que los controles simulan los dispositivos de entrada de instrumentos y los datos de suministro para el diagrama de bloques del VI.Por ejemplo, el valor "Numeric" puede ser un "control numérico" o un "indicador numérico", como se observa en la figura 2.4.

Figura 2. 4: Ejemplo para controles e indicadores numéricos.
Elaborado: Por el Autor

Al seleccionar *"Numeric Control"*se podrá cambiar fácilmente a*"Numeric Indicador "*a través de *"Change to Indicator"* tal como se muestra en la figura 2.5.

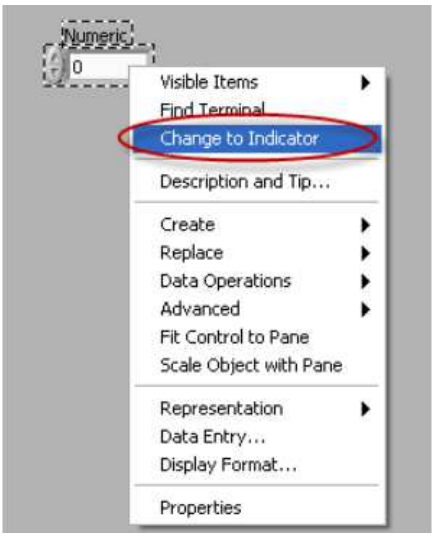

Figura 2. 5: Forma de cambiar los controles numéricos a indicadores numéricos.
Elaborado: Por el Autor

La diferencia entre controles numéricos e indicadores numéricos, es que para el primero se introducen valores, mientras que para el segundo solamente se puede leer (observar), es decir, que no se puede modificar, tal como se ilustra en la figura 2.6.

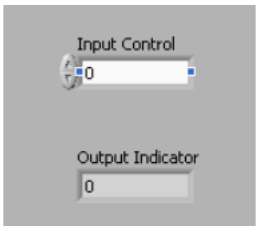

Figura 2. 6: Diferencias entre controles e indicadores numéricos.
Elaborado: Por el Autor

2.4.2. Diagrama de Bloques.

Es donde se realiza toda la programación, en la cual definimos su funcionalidad, es aquí donde seleccionamos los iconos con determinadas funciones a realizar y son interconectadas con otros elementos(Palacios, 2012). En la figura 2.7 se muestra la ventana del diagrama de bloques de un ejemplo cualquiera.

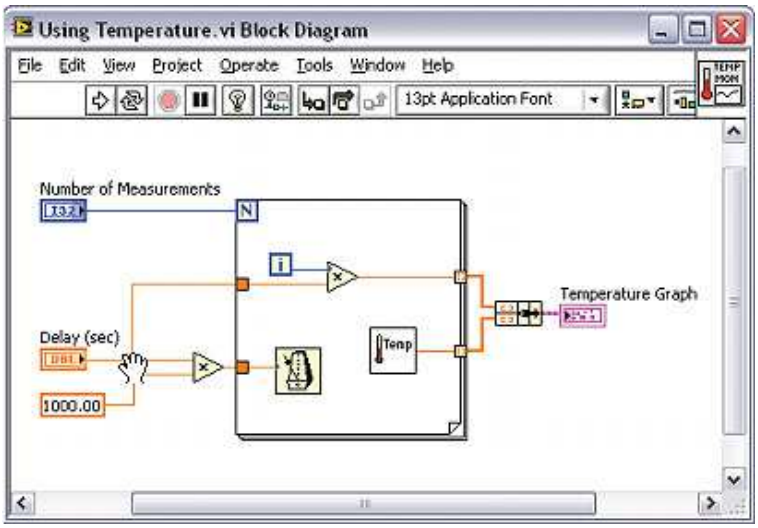

Figura 2. 7: Ventana del diagrama de bloques.
Fuente: http://www.ni.com/gettingstarted/labviewbasics/esa/environment.htm

En la figura 2.8 se muestra el comportamiento de las dos ventanas, es decir, del panel frontal y del diagrama de bloques.

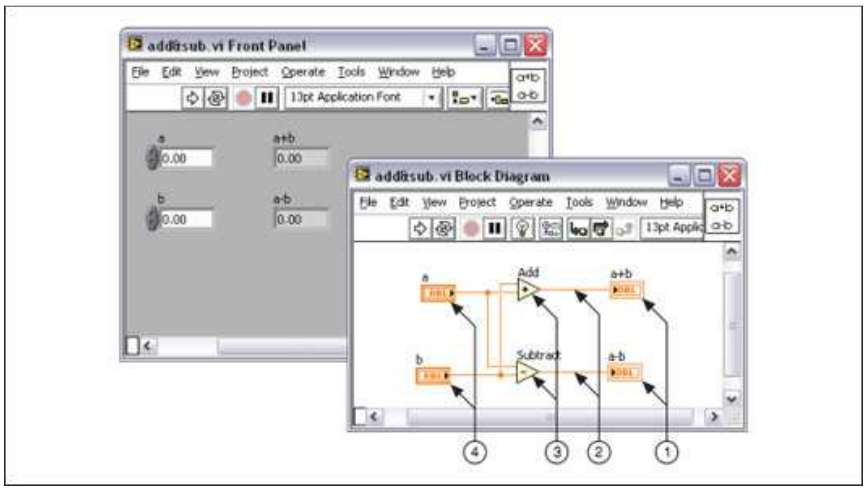

Figura 2. 8: Comportamiento del diagrama de bloques y del panel frontal.
Fuente: http://www.ni.com/gettingstarted/labviewbasics/esa/environment.htm

En la tabla 2.3 se muestran los símbolos para los diversos tipos de terminales de control e indicadores. Para González C., R., & Pradines P., R., (2007) el color y el símbolo de cada terminal indican el tipo de datos del control o indicador. Los terminales de control tienen un borde más grueso que los terminales de los indicadores. Además, aparecen flechas en los terminales del panel frontal para indicar si el terminal es un control o un indicador. Aparecerá una flecha a la derecha si el terminal es un control, y una flecha que aparece a la izquierda si el terminal es un indicador.

Tabla 2. 3: Tipos de terminales de control e indicadores.

Control	Indicator	Data Type	Color	Default Values
SGL	SGL	De precisión simple de punto flotante numérico	Naranja	0.0
DBL	DBL	De doble precisión de punto flotante numérico	Naranja	0.0
EXT	EXT	De precisión extendida de coma flotante numérica	Naranja	0.0
CSG	CSG	De precisión compleja de punto flotante numérico	Naranja	0.0 + i0.0

Control	Indicator	Data Type	Color	Default Values
		De doble precisión compleja de punto flotante numérico	Naranja	0.0 + i0.0
		De precisión extendidad compleja de punto flotante numérico	Naranja	0.0 + i0.0
		De 8 bits- número entero con signo	Azul	0
		De 16 bits- número entero con signo	Azul	0
		De 32 bits- número entero con signo	Azul	0
		De 8 bits- número entero sin signo	Azul	0
		De 16 bits- número entero sin signo	Azul	0
		De 32 bits- número entero sin signo	Azul	0
		<64.64> bits - marca de tiempo	Marrón	date and time (local)
		Tipo de enumerado	Azul	—
		Booleano	Verde	FALSE
		Cadena	Rosado	empty string
		Arrays (Matrices): Encierra el tipo de datos de sus elementos entre corchetes y toma el color de ese tipo de dato.	Varia de color	—
		Cluster: Encierra varios tipos de datos. Los tipos de cluster de datos son marrón si todos los elementos del grupo son numéricos o rosa si los elementos del grupo son de diversos tipos.	Marrón o rosado	—
		Trayectoria o ruta	Aguamarina	<Not A Path>
		Dinámicos	Azul	—
		Forma de onda: Cluster de elementos que transporta los datos, la hora de inicio, y Δt de una forma de onda.	Marrón	—
		Forma de onda digital	Verde oscuro	—

Fuente: http://www.ni.com/gettingstarted/labviewbasics/esa/environment.htm

2.4.3. Paleta de Controles.

La paleta de controles sólo está disponible en el panel frontal tal como se mencionó en el acápite 2.4.1, donde la paleta de controles contiene los

controles e indicadores que se utilizan para construir el panel frontal. A continuación se observa en la figura 2.9 las sub paletas numéricas y booleanas, mientras que la figura 2.10 se muestra la sub paleta cadena y trayectoria.

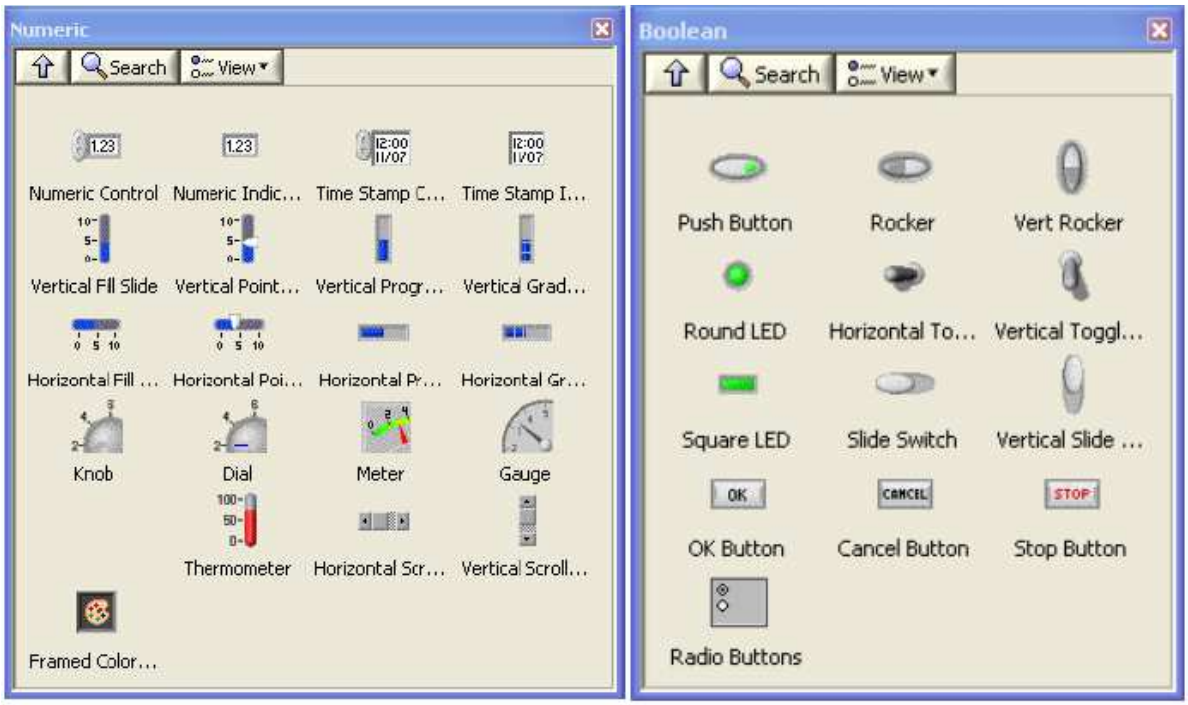

Figura 2. 9: Sub paletas del tipo numérico y booleano.
Elaborado: Por el Autor

Figura 2. 10: Sub paleta cadena y trayectoria.
Elaborado: Por el Autor

2.7. Características principales.

Hoy en día, científicos, ingenieros, técnicos y estudiantes utilizan LabView para desarrollarsoluciones que respondan a sus interrogantes más exigentes, es por ello que damos fe que suprincipal característica es la facilidad de uso que posee. También resulta válido para personas conpocos conocimientos en programación, ya que pueden realizar programas relativamentecomplejos, imposibles para ellos y a veces hasta para uno mismo con los lenguajes tradicionales. En la tabla 2.4 se muestran algunas interfaces de comunicación que funcionan con LabView.

Tabla 2. 4: Interfaces de comunicación para LabView.

Interfaz de comunicación	Descripción
Puerto Serie	Los puertos seriales se refiere a los datos enviados mediante un solo hilo: los bits se envían uno detrás del otro
Puerto Paralelo	La transmisión de datos paralela consiste en enviar datos en forma simultánea por varios canales (hilos).
GPIB	Es un estándar de conexión que permite la comunicación de un ordenador con instrumentos electrónicos de medida, como pueden ser generadores de funciones, osciloscopios, etc. En la figura 2.11 se observa el dispositivo GPIB.
USB	Es un bus estándar industrial que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre computadoras, periféricos y dispositivos electrónicos.
PXI	PXI es una plataforma de despliegue de alto rendimiento y bajo costo para aplicaciones tales como pruebas de manufactura, militares y aeroespaciales, monitoreo de máquinas, automotrices y pruebas industriales.
VXI	Permite realizar pruebas y mediciones automatizadas tanto para las necesidades y futuras aplicaciones.
UDP	Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera.
Data Socket	National Instruments desarrolló DataSocket, que permite la transferencia fácil de datos sobre muchos protocolos diferentes (DSTP, OPC, MIRADOR, HTTP, FTP, y de acceso a archivos locales).
Irda (infrarrojo)	Transmisión y recepción de datos por rayos infrarrojos. Esta tecnología se basa en rayos luminosos que se mueven en el espectro infrarrojo. Soportan una amplia gama de dispositivos electrónicos y comunicaciones.

Elaborado: Por los Autores

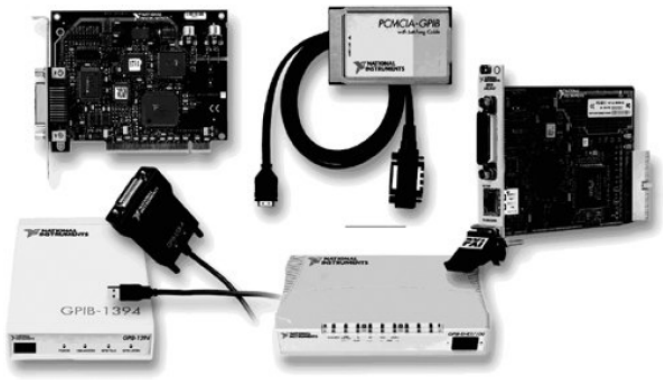

Figura 2. 11: Tarjetas GPIB de NI.
Elaborado: Por el Autor

En diferentes trabajos de tesinas o de titulación, sostienen que LabViewbrinda la posibilidad de interactuar con diferentes lenguajes de programación y dispositivos electrónicos, tal como se muestra en la tabla 2.5. (Borja S., J. E., & Jiménez M., R. R., 2011) (Pincay S., E., 2014)

Tabla 2. 5: Plataformas de programación y dispositivos electrónicos compatibles con LabView.

DLL (librerías de funciones), .NET, ActiveX, MultiSim, Matlab/Simulink, AutoCAD, SolidWorks, etc.
Herramientas gráficas y textuales para el procesado digital de señales.
Visualización y manejo de gráficas con datos dinámicos.
Adquisición y tratamiento de imágenes.
Control de movimiento (combinado incluso con todo lo anterior).
Tiempo Real estrictamente hablando.
Programación de FPGAs para control o validación.
Sincronización entre dispositivos.

Fuente: http://www.ni.com/gettingstarted/labviewbasics/esa/environment.htm

2.8. LabVIEW en Real-Time, FPGA, PDA y Embebidos.

Hay unos pocos módulos adicionales especiales para LabVIEW que debemos mencionar. Estos son los de LabVIEW Real-Time, LabVIEW FPGA, LabVIEW PDA, y los módulos de LabVIEW Embebidos *(Embedded)*, que le permite ejecutar VIs de LabVIEW en otros destinos de ejecución.

El módulo LabVIEW Real-Time, es una combinación de hardware y software que le permite tomar partes de su código de LabVIEW, y descargarlos para ser ejecutado en un tablero de control independiente con su propio sistema operativo en tiempo real. Esto significa que usted puede garantizar que ciertas piezas de su programa de LabVIEW sigan funcionando con precisión, aunque la interfaz de usuario bloquea la máquina anfitriona y sus chillidos de computadora a un punto muerto.

Los módulos de LabVIEW FPGA y LabVIEW PDA permiten orientar a los programas de LabVIEW para ejecutarse en una FPGA *(Field Programmable Gate Array)* o PDA *(Personal Digital Assistant)*, respectivamente. El módulo LabVIEW FPGA proporciona la capacidad para aprovechar la naturaleza intrínsecamente paralela de programación de flujo de datos con el dispositivo de lógica programable inherentemente paralelo, al chip FPGA. El módulo LabVIEW PDA permite a los desarrolladores desplegar aplicaciones de LabVIEW en los dispositivos de mano tales como los que ejecutan Palm OS y Pocket PC, para la creación, sistemas de adquisición de datos de mano portátiles, conectados en red.

Mientras que el módulo LabVIEW Embedded, permite compilar los VIs de LabVIEW y los ejecuta en cualquier plataforma de microprocesador de 32 bits a través de la integración de LabVIEW con cadenas de herramientas de terceros. Esto incluye los compiladores de GNU C++ (gcc), eCos, Wind River Tornado/VxWorks, y muchas otras plataformas que permiten crear su propia capa de soporte de cadena de herramientas, por lo que el cielo es el límite.

A medida que avanzamos hacia el futuro, esperamos ser capaces de ejecutar los VIs LabVIEW… sí... en todas partes! "LabVIEW en todas partes" es la filosofía que permite desarrollar instrumentos virtuales de LabVIEW y desplegarlos en casi cualquier hardware, incluso volver a configurar el hardware. National Instruments está comprometido con esta filosofía y, sin duda, ofrecer más y mejores herramientas y módulos para el funcionamiento de "LabVIEW todas partes".

CAPÍTULO 3: MICROCONTROLADORES PIC.

El presente capítulo se explicará y describirá a los microcontroladores de la familia Microchip.

3.1. Microcontroladores PIC16.

Los microcontroladores (uC) PIC16 tienen las siguientes utilidades al momento de realizar aplicaciones en cualquier ámbito de la Ingeniería tanto en Electrónica como en Telecomunicaciones:

- funciones MCU
- La ejecución del programa
- Registros de archivos RAM
- Otros dispositivos PIC

La unidad de microcontrolador (MCU), en actualidad es grande (operatividad), o más bien pequeña en tamaño. Es uno de los avances más significativos en la continua miniaturización de los dispositivos electrónicos. Ahora, incluso los productos triviales (insignificantes), como una tarjeta de cumpleaños musical o etiqueta de precio electrónica, pueden incluir unmicrocontrolador (uC).

Que son un factor importante en la digitalización de los sistemas analógicos, tales como sistemas de sonido o la televisión. Además, proporcionan un componente esencial de los sistemas más grandes, tales

como automóviles, robots y sistemas industriales. No hay escapatoria posible de los microcontroladores, por lo que es muy útil saber cómo funcionan.

El ordenador o controlador digital tiene tres elementos principales: los dispositivos de entrada y salida, que se comunican con el mundo exterior; un procesador, para hacer cálculos y manejar las operaciones de datos; y la memoria, para almacenar programas y datos. En la figura 3.1 se muestra un ejemplo de lo explicado.

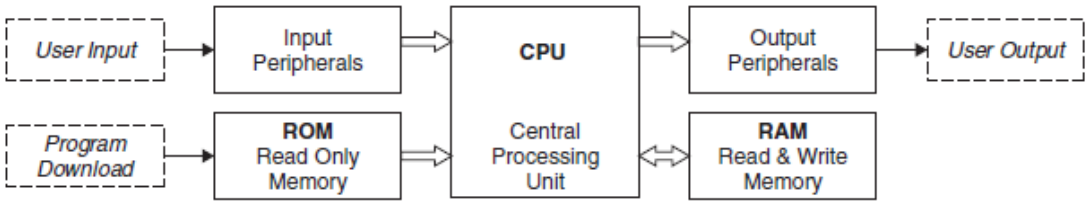

Figura 3. 1: Elementos de un controlador digital.
Fuente:

A diferencia delos sistemas de microprocesadores convencionales (tal como un PC), que dispone de muchos chips separados en una placa de circuito impreso, el microcontrolador contiene todos estos elementos en un solo chip. El uC, es esencialmente una computadora en un chip; sin embargo, todavía necesita dispositivos de entrada y salida, tales como un teclado, sensores (por ejemplo infrarrojos y movimiento), LDCs, GLCDs, etc., para así formar un sistema de trabajo.

El microcontrolador almacena su programa en una ROM (memoria de sólo lectura). Anteriormente, se utilizaba UV (ultravioleta) para borrar la ROM

programable (EPROM). Los chips ROM programables, son programados en las etapas finales de la producción, mientras que la EPROM puede ser programada por el usuario.

Las Flash ROM actualmente suele utilizarse para prototipos y poco volumen de producción. Esto puede ser programado en el circuito por el usuario después de que el circuito ha sido construido. El ciclo de creación de los prototipos es más rápido, y las variaciones de software son más fáciles de acomodar. Todos estamos ahora familiarizados con la ROM flash que se utiliza en dispositivos de memoria USB, memoria de la cámara digital, y así sucesivamente, con capacidades comunes de Gb $(10^9\ byte\ )$.

La gama de microcontroladores disponibles se está expandiendo rápidamente.El primero en ser utilizado ampliamente, fue el microcontrolador 8051 de Intel, que fue desarrollado junto al resto de procesadores de Intel en las primeras PCs, talcomo la Intel 8086. Éste dispositivo ha dominado el campo, desde hace algún tiempo.

Otros microcontroladores, surgieronlentamente, principalmente en la forma compleja de los procesadores, para aplicaciones tales como los sistemas de gestión del motor. Estos dispositivos son relativamente caros, por lo que fueron justificadas sólo en productos de alto valor. El potencial de los microcontroladores parece haberse dado cuenta muy lentamente.

El desarrollo de Flash ROM ayudó a abrir el mercado, y Microchip fue uno de los primeros en tomar ventaja. El PIC16F84 barato y reprogramable se convirtió en el más conocido, convirtiéndose rápidamente en el número uno de dispositivo para estudiantes y aficionados. En la parte posterior de este éxito, el producto Microchip se desarrolló rápidamente y de rangos diversificados.

El sistema MPLAB apoyo al desarrollo de los PICs, éste se distribuye de forma gratuita, lo que ayudó a los PICs a dominar el mercado de gama baja, aunque en la actualidad han surgido diferentes plataformas de programación de alto nivel, tales como: PicBasic, PicC, MikroBasic, MikroPascal y MikroC.

Por eso Flash ROM, es uno de los dispositivos de mejor desarrollotécnico, que se hicieron experimentando sobre microsistemas más fáciles y más interesantes. El software interactivo de diseño de circuitos electrónicos, fue otra herramienta útil para el éxito de los PICs. ISIS Proteus Profesional es la plataforma (software) que permite realizar simulaciones gracias a la disponibilidad de librerías (familia de los PICs).

También existe Ares Proteus para el diseño de tarjetas electrónicas. El proceso de diseño conjunto es ahora mucho más transparente, de modo que los sistemas de trabajo son más rápidamente alcanzables para principiantes. El bajo costo en los circuitos de depuración, es otra técnica que ayuda

finalmente a que el hardware funcione rápidamente, con sólo un gasto modesto en las herramientas de desarrollo(Gridling, 2007).

3.1.1. Características de los Microcontroladores PIC16.

La gama de los microcontroladores disponibles, ocurrió debido a que las características de los microcontroladores utilizados en cualquier circuito en particular, requerían de un alto grado de concordancia posible a las necesidades reales de cualquier aplicación. Algunas de las principales características a tener en cuenta son:

- Número de entradas y salidas.
- Tamaño de la memoria de programa.
- Tamaño de la memoria RAM de datos.
- Memoria de datos no volátil.
- Máxima velocidad de reloj.
- Rango de interfaces.
- Costo y disponibilidad.

Por ejemplo, el PIC16F877A es de gran utilidad como dispositivo de referencia, ya que tiene un sistema de instrucción mínima, pero una gama completa de características periféricas.El enfoque general para el diseño de aplicaciones a través de microcontroladores, es desarrollar diseños utilizando únicamente PICs (chip) que tienen capacidad de sobra, luego de seleccionado un dispositivo relacionado que tiene el conjunto de características más adaptada a los requisitos de la aplicación.

Si es necesario, se puede utilizar otra escala o rango de los PICs, tales como el PIC10 (serie de 12 pines), o si se hace evidente la necesidad de mayor potencia, podemos subir a un chip de especificación más alta, como el PIC18 (serie de 24 pines). Esto es posible ya que todos los dispositivos tienen los mismos conjuntos de arquitectura de núcleo e instrucciones compatibles.

La variación más significativa entre los PICs de Microchip, es el tamaño o número de instrucciones, que puede ser de 12, 14, o 16 bits. El sufijo A (al final de la numeración del PIC) indica que el chip tiene una velocidad máxima de reloj de 20 MHz, la actualización principal del dispositivo original sería 16F877.

Estos chips de otro modo pueden ser considerados como idénticos, el sufijo siendo opcional para la mayoría de propósitos. El pinout 16F877A se observa en la figura 3.2 y la arquitectura interna se muestra en la figura 3.3. Esta última es una versión algo simplificada del diagrama de bloques definitivo en la ficha técnica.

3.1.2. Ejecución del programa.

El chip tiene 8 kb (8096×14 bits)de memoria de programa (Flash ROM), el cual tiene que ser programado a través de los pines de programación de serie PGM, PGC y PGD. La longitud fijaInstrucciones contienen tanto el código de operación y operando (datos inmediatos, la

dirección de registro, o saltar de direcciones). El PIC de gama media tiene un número limitado de instrucciones (35) y por lo tanto se clasifica como un procesador (reducida ordenador de conjunto de instrucciones) RISC.

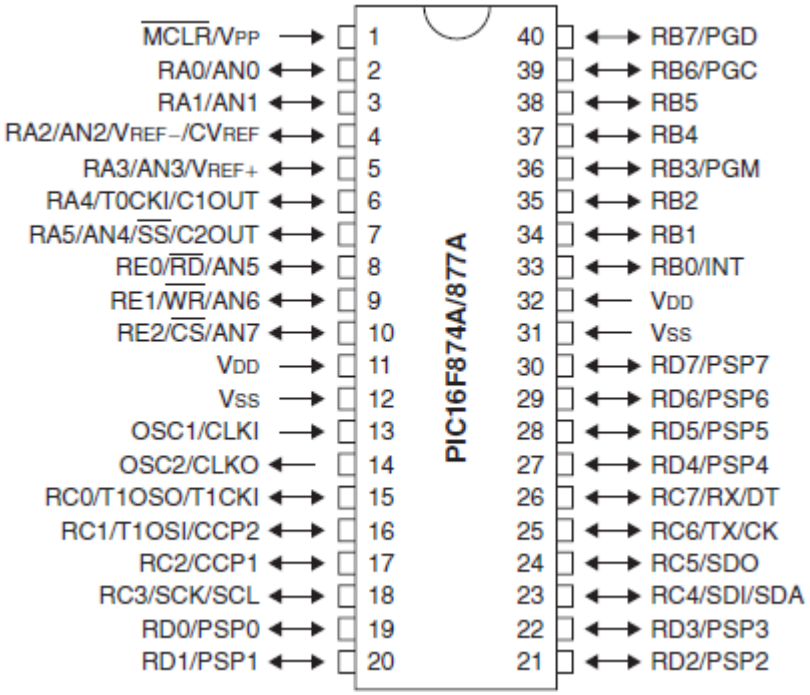

Figura 3. 2: Configuración de pines 16F877
Fuente: Data Sheet de Microchip

En cuanto a la arquitectura interna, podemos identificar los bloques que participan en la ejecución del programa. La memoria de programaROM contiene el código máquina, en la ubicación numerada de 0000h a 1FFFh (tamaño 8 kb).El contador de programa contiene la dirección de la instrucción en curso y se incrementa o se modifica después de cada paso. El restablecimiento o encendido, que se pone a cero y la primera instrucción en la dirección 0000 se carga en el registro de instrucción, decodificador, y ejecutado.

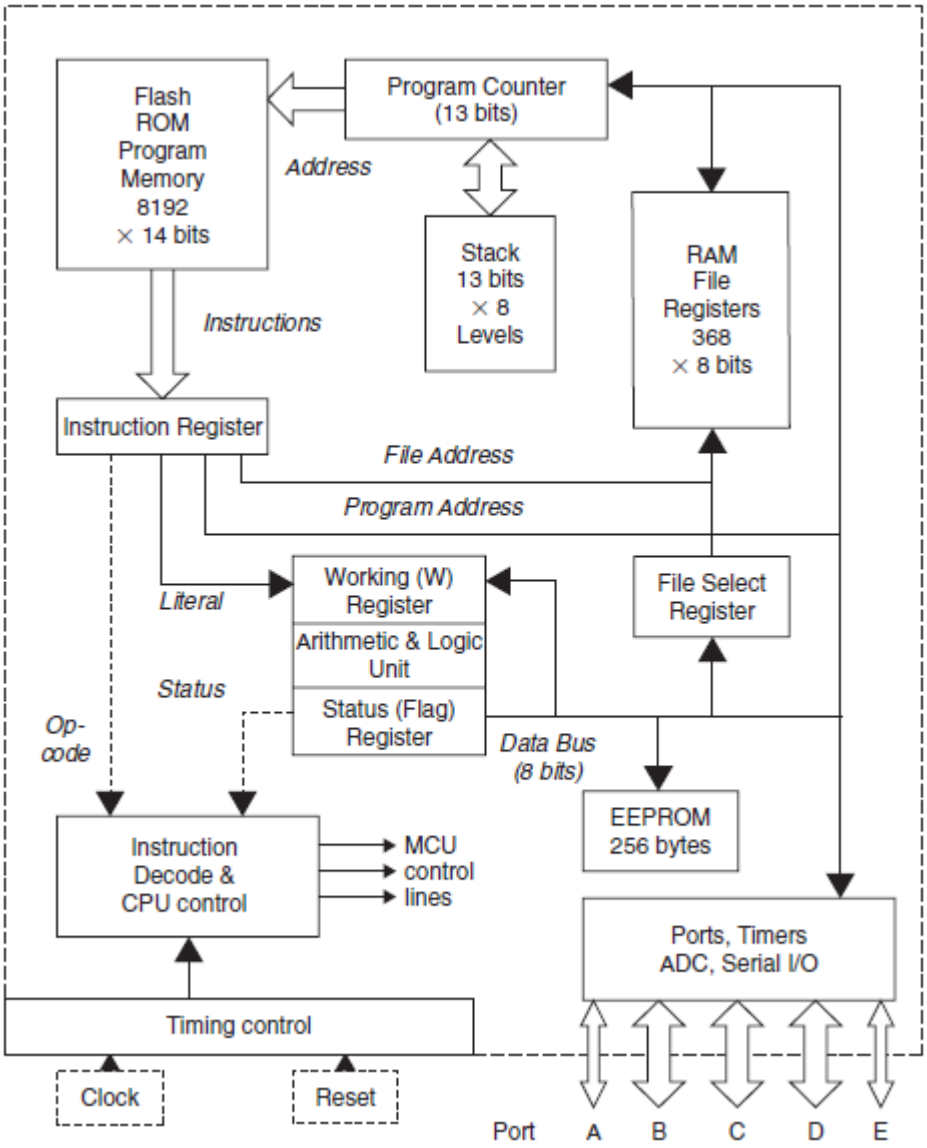

Figura 3. 3:Diagrama de bloques PIC16F877 MCU.
Fuente: Data Sheet de Microchip

El programa prosigue entonces en secuencia, que operan en los contenidos de los registros de archivos (000-1FFh), la ejecución de instrucciones de movimiento de datos, para transferir de datos entre los puertos y los registros de archivo o la aritmética y las instrucciones lógicas para procesarlas. La CPU tiene un registro principal de trabajo (W), a través del cual deben pasar todos los datos.

Si se decodifica una instrucción de salto (salto condicional), se lleva a cabo una prueba de bits; si el resultado es verdadero, la dirección de destino incluida en la instrucción se carga en el contador de programa para forzar el salto. Si el resultado es falso, la secuencia de ejecución continúa sin cambios. En lenguaje ensamblador, cuando se implementan en la práctica se utilizan CALL y RETURN.

Subrutinas, un proceso similar se produce. La pila se utiliza para almacenar direcciones de retorno, de modo que el programa puede volver automáticamente a la posición original del programa. Sin embargo, este mecanismo no es utilizado por el compilador CCS C, ya que limita el número de niveles de subrutina (o funciones C) a ocho, que es la profundidad de la pila. En cambio, una instrucción GOTO sencilla se utiliza para las llamadas de función y devoluciones, con la dirección de retorno calculada por el compilador.

Tabla 3. 1:PIC16F877 archivo simplificado Registro Mapa.

Bank 0 (000–07F)		Bank 1 (080–0FF)		Bank 2 (100–180)		Bank 3 (180–1FF)	
Address	Register	Address	Register	Address	Register	Address	Register
000h	Indirect	080h	Indirect	100h	Indirect	180h	Indirect
001h	Timer0	081h	Option	101h	Timer0	181h	Option
002h	Prog. count. low	082h	Prog. count. low	102h	Prog. count. low	182h	Prog. count. low
003h	Status reg	083h	Status reg	103h	Status reg	183h	Status reg
004h	File select	084h	File select	104h	File select	184h	File select
005h	Port A data	085h	Port A direction	105h	—	185h	—
006h	Port B data	086h	Port B direction	106h	Port B data	186h	Port B direction

007h	Port C data	087h	Port C direction	107h	—	187h	—
008h	Port D data	088h	Port D direction	108h	—	188h	—
009h	Port E data	089h	Port E direction	109h	—	189h	—
00Ah	Prog. count. high	08Ah	Prog. count. high	10Ah	Prog. count. high	18Ah	Prog. count. high
00Bh	Interrupt control	08Bh	Interrupt control	10Bh	Interrupt control	18Bh	Interrupt control
00Ch–01Fh	20 peripheral control registers	08Ch–09Fh	20 peripheral control registers	10Ch–10Fh	4 peripheral control registers	18Ch–18Fh	4 peripheral control registers
				110h–11Fh	16 general purpose registers	190h–19Fh	16 general purpose registers
020h–06Fh	80 general purpose registers	0A0h–0EFh	80 general purpose registers	120h–16Fh	80 general purpose registers	1A0h–1EFh	80 general purpose registers
070h–07Fh	16 common access GPRs	0F0h–0FFh	Accesses 070h–07Fh	170h–17Fh	Accesses 070h–07Fh	1F0h–1FFh	Accesses 070h–07Fh

Fuente: Data Sheet de Microchip

3.1.3. Registros RAM de archivos.

El bloque principal de memoria RAM (ver la tabla 3.1) es un conjunto de 368 registros de archivo de 8 bits, incluyendo los registros de funciones especiales (SFR), que tienen una función específica, y los registros de propósito general (GPRS). Cuando se crean las variables en C, que se almacenan en el GPRS, comenzando en la dirección 0020 h.

Los registros de archivos se dividen en cuatro bloques, que se registran en los bancos 0 a 3. Los SFRsestán ubicados en direcciones bajas de cada banco de la memoria RAM.

Algunos registros son direccionables a través de las fronteras de los bancos; por ejemplo, el registro de estado se puede acceder en todos los bloques en la dirección correspondiente en cada banco. Otros son direccionables en sólo una página específica, por ejemplo, Port A es un registro de datos. Algunas direcciones de registro no se implementan físicamente.

Dado que algunos registros son accesibles en múltiples bancos, la conmutación del banco puede ser minimizado por el compilador cuando se monta el código de máquina, ahorrando así espacio del código de programa y también el tiempo de ejecución. Para más detalles sobre el conjunto de registros de archivos, consulte la ficha de datos (data Sheet de cada uno de los microcontroladores PIC).

El contador de programa utiliza dos registros de 8 bits para almacenar una dirección de memoria de programa de 13 bits. Sólo el byte bajo en la dirección002h es directamente direccionable. El registro de estado 003hse registra su resultado en la ALU (unidad aritméticalógica), las operaciones, tales como cero y de acarreo. Los registros seleccionados indirectos y de archivo se utilizan para el direccionamiento indexado de la GPRS. El Timer0 es el temporizador/contador de registro disponible en todo microcontrolador PIC, mientras que los registros Timer1 y Timer2 se encuentran en el bloque periférico.

Los registros del puerto se encuentran en el banco 0, mediante las direcciones 05h (Port A) hasta 09h (Port E) con el registro de dirección de datos para cada uno, en el lugar correspondiente del banco 1. Podemos ver que un total de $80 + 16 + 80 + 96 + 96 = 368$ GPRs están disponibles para su uso como la memoria RAM de datos. Hay que tener en cuenta que el número de registros utilizados para cada variable C, depende del tipo de variable y puede oscilar de 1 a 32 bits (1-4 GPRS).

3.1.4. Otros PIC chips.

En cualquier diseño embebido, las características de la MCU tienen que ser adaptado a los requisitos de la aplicación. El fabricante debe asegurarse de que, como las aplicaciones se vuelven más exigentes, un más potente dispositivo de un tipo conocido es disponible. Podemos ver este proceso en acción donde Microchip comenzó la producción de los chips básicos como el 16C84, luego desarrollado la gama de productos para satisfacer el creciente mercado.

Los microcontroladores PIC están disponibles actualmente en grupos distintos, designados las series como 10, 12, 16, 18, y 24. Sus características generales se resumen en la Tabla 2.2.Las 16 series de dispositivos originales CMOS fueron designados como 16Cxx. Cuando se introdujo la memoria flash, se convirtieron en 16Fxxx. Actualmente, un número limitado de dispositivos esta disponible en las gamas baja de la cantidad de pines (LPC) (series 10/12), mientras que los rangos de potencia

se están expandiendo rápidamente. Además son los que figuran en el rango 24HXXXX, que corre a 40 MIPS, y el dsPIC (procesador de señal digital) Campo de alta especificación.

Tabla 3. 2:Tipos de microcontroladores PIC.

uC PIC	Pines	Palabras de datos (bits)	Memoria de programa (bytes)	Conjunto típico de instrucciones	Velocidad MIPS	Descripción
10FXXX	6	8	≤ 512 B	33 x 12 bits	≤ 2	Pocos pines, de pequeño formato, barato, sin EEPROM, no de bajo consumo. programa ensamblador
12FXXX	8	8	≤ 2 kB	12 / 14 bits	≤ 0.5	Pocos pines, de pequeño formato, barato, con EEPROM, ADC de 10 bits, algunos de baja potencia, programa ensamblador
16FXXX	≤ 64	8	≤ 14 kB	35 x 14 bits	≤ 5	Gama media, UART, I2C, SPI, muchas de baja potencia, programa ensamblador o C
18FXXXX	≤ 100	8	≤ 128 kB	75 x 16 bits	≤ 16	De alta gama, CAN, serie J USB suministro de 3V, programa C
24FXXXX	≤ 100	16	≤ 128 kB	76 x 24 bits	16	Gama de potencia, suministro de 3V, sin EEPROM, memoria RAM de datos ≤ 8kB, programa C

Fuente: (Verle, 2010)

3.2. Configuración de los microcontroladores PIC16.

Al programar el microcontrolador PIC, ciertos modos de funcionamiento se deben establecer antes de la descarga principal del programa. Estos son controlados por los bits individuales en un registro de configuración especial

separado del bloque de memoria principal. Las principales opciones (que se describirán posteriormente) son las siguientes.

- tipos de oscilador de reloj
- Perro guardián, encendido, temporizadores de salida
- Programación de bajo voltaje
- Código de protección
- Modo de depuración de circuito

3.2.1. Opciones de reloj.

El chip PIC16F877 tiene dos modos de reloj principal, CR y XT. El modo de CR necesita un circuito RC (resistivo/capacitivo) sencillo unido a CLKIN, cuya constante de tiempo $(R \times C)$determina el período de la señal de reloj. La resistencia R debe estar entre$3\ k\Omega$ y $100\ k\Omega$, y capacitancia C mayor que$20\ pF$. Por ejemplo, si $R = 10\ k\Omega$y $C = 10$ nF, el periodo de reloj será de alrededor de $2 \times C \times R = 200\ \mu s$(calculados a partir de CR del tiempo de subida/bajada) y cuya frecuencia sería igual a $5\ kHz$. Esta opción es aceptable cuando lasincronización del programa no sea crítica.

El modo XT comúnmente es el más utilizado, dado que el costo del componente extra es pequeño en comparación con el coste del mismo y la sincronización exacta del chip es a menudo una necesidad. Un cristal externo y dos condensadores están disponibles en los pines CLKIN y CLKOUT.

La frecuencia del cristal en este modo puede ser de 200 kHz a 4 MHz y es típicamente una precisión mejor que 50 ppm (partes por millón) o 0,005%.Un valor conveniente es 4 Mz, ya que es la máxima frecuencia posible con un cristal estándar, y ofrece un tiempo de ejecución de instrucciones de $1.000\mu s$ (1 millón de instrucciones por segundo, o 1 MIPS).

Un cristal de baja velocidad puede ser utilizado para reducir el consumo de energía, que es proporcional a la velocidad de reloj en los dispositivos CMOS. El modo LP (baja potencia) es compatible con la frecuencia de reloj rango 32-200 kHz. Para lograr la máxima velocidad de reloj de 20 MHz, se necesita una alta velocidad (HS) de cristal, con un aumento correspondiente en el consumo de energía.

Los fusibles de configuración MCU se deben establecer en el modo de reloj requerido cuando se programa el chip. Numerosos chips de los microcontroladores PIC, tienen ahora un oscilador interno, en la cual ya no necesita componentes externos. Es más preciso que el la señal de reloj RC, pero menos preciso que un cristal. Normalmente funciona a 8 MHz y se puede calibrar en la fase de configuración del chip para proporcionar una fuente de temporización más precisa.

3.2.2. Opciones de configuración.

Aparte de las opciones de reloj, se deben seleccionar varias opciones de hardware.

a. Watchdog Timer

Cuando está activado, el temporizador watchdog (WDT) restablece automáticamente el procesador después de un período determinado (por defecto 18 ms). Esto permite, por ejemplo, una aplicación para escapar de un bucle sin fin, debido a un error de programación o condición de tiempo de ejecución no prevista por el diseñador del software o código fuente.

Para mantener el funcionamiento normal, el WDT debe ser desactivado o restablecer en el bucle del programa antes de que expire el período de tiempo de espera establecido. Por tanto, es importante establecer los bits de configuración del microcontrolador para desactivar el WDT si no se pretende utilizar esta característica. De lo contrario, el programa es susceptible de comportarse mal, debido a la puesta a cero al azar del PIC.

b. Encendido del temporizador

El temporizador de encendido (PuT), ofrece un 72 ms nominal de retardo entre la tensión de alimentación para alcanzar el valor de operación y el inicio de la ejecución del programa. Esto asegura que la tensión de alimentación sea estable antes de que el reloj se ponga en marcha.

c. Puesta en marcha del temporizador

Después de transcurrir el tiempo de encendido, un nuevo retraso permite que el reloj se estabilice antes de que comience la ejecución del

programa. Cuando se selecciona uno de los modos de reloj de cristal, la CPU espera 1.024 ciclos antes de habilitar la CPU.

d. Restablecimiento (Brown-out Reset, BOR).

Es posible para una caída de tensión de alimentación transitoria, o brown-out, para interrumpir la ejecución del programa del microcontrolador. Cuando se activa, el circuito de detección brown-out mantiene al uC en reinicio, mientras la tensión de alimentación es inferior a un umbral y la libera cuando el suministro se ha recuperado. En CCS C, un bajo voltaje detecta la función de activar una interrupción que permite al programa se reinicie de forma ordenada.

e. Protección de código (CP)

El chip puede configurarse durante la programación para evitar que el código de máquina que está siendo leído de nuevo desde el PIC, para proteger el código de valor comercial o seguro. Opcionalmente, partes de código son seleccionadas del programa para ser protegidos contra escritura.

f. Programación y depuración en un circuito.

La mayoría de los uC (chips) PIC ahora soportan programación y depuración en un circuito, que permite que el código de programa pueda ser descargado y probado en el hardware de destino, bajo el control del sistema de acogida. Esto proporciona una etapa de prueba final después de la

simulación de software que se ha utilizado para eliminar la mayoría de los errores en el programa.

Por ejemplo, MPLAB permite la misma interfaz que se utilizapara la depuración, tanto en la simulación y en modo circuito. La ligera desventaja, de esta opción es que se debe tener cuidado de que cualquier circuito de aplicación conectado a las patillas de programaciónno interfiera con la operación de estas características. Es preferible dejar estos pines para el uso exclusivo del sistema de la CIPD. Además, se requiere una pequeña sección de memoria de programa para ejecutar el código de depuración.

g. Modo de programación de bajo voltaje

El modo de programación de bajo voltaje puede ser seleccionado durante la programación de manera que no se necesita (12 V) alto voltaje de programación habitual, y el chip se puede programar en $V_{vv}(+5V)$. La desventaja es que el pin de programación no puede ser utilizado para los pines de E/S digitales. En cualquier caso, se recomienda aquí que los pines de programación no sean utilizados para dispositivos de E/S.

h. Memoria de sólo lectura eléctricamente programable y borrable

Muchos microcontroladores PIC tiene un bloque de memoria de usuario no volátil donde los datos pueden ser almacenados durante el apagado. Estos datos pueden, por ejemplo, ser el código de seguridad para una

cerradura electrónica o lector de tarjetas inteligentes. La memoria EEPROM, puede ser reescrita por ubicación individual, a diferencia de la memoria de programa ROM flash. El PIC16F877, tiene un bloque de 256 bytes, que es un valor bastante típico. Hay una secuencia especial de lectura/escritura para evitar la sobre escritura accidental de los datos.

3.2.3. Configuración en lenguaje de alto nivel C.

Los directiva #fuses del preprocesador se utiliza para configurar los fusibles de configuración de la plataforma de programación C orientado a los microcontroladores PIC. Por ejemplo, una declaración típica sería:

```
#fuses XT,PUT,NOWDT,NOPROTECT,NOBROWNOUT
```

Las opciones definidas en el archivo de encabezado estándar de CCS C para el PIC16F877 son:

Clock Type Select	LP, XT, HS, RC
Watchdog Timer Enable	WDT, NOWDT
Power Up Timer Enable	PUT, NOPUT
Program Code Protect	PROTECT, NOPROTECT
In Circuit Debugging Enable	DEBUG, NODEBUG
Brownout Reset Enable	BROWNOUT, NOBROWNOUT
Low Voltage Program Enable	LVP, NOLVP
EEPROM Write Protect	CPD, NOCPD
Program Memory Write Protect (with percentage protected)	WRT_50%, WRT_25%, WRT_5%, NOWRT

La condición predeterminada para los fusibles si no se incluye una directiva, sería equivalente a escribir lo siguiente:

```
#fuses RC,WDT,NOPUT,BROWNOUT,LVP,NOCPD,NOWRT
```

Esto corresponde a todos los bits del registro de configuración siendo de alta prioridad.

3.3. Periféricos del microcontrolador PIC16.

En esta sección se describirá algunos de los periféricos del PIC16:

- Dispositivos E/S digitales.
- Temporizadores (Timers)
- Convertidores A/D
- Comparadores
- Puerto esclavo paralelo
- Interrupciones

3.3.1. Dispositivos E/S digitales.

Las entradas y salidas (E/S) digitales básicas en los microcontroladores, utilizan un pin cuyo puerto es bidireccional. La configuración por defecto es generalmente, pin de entrada digital, ya que es la opción más segura si algún error se ha hecho en las conexiones externas.Para configurar el pin como salida, el correspondiente bit de dirección de datos debe ser despejado en el registro de dirección de datos del puerto (por ejemplo, TRISD). Nótese, sin embargo, que los pines conectados al convertidor A/D por defecto son para modos de entrada analógica.

El hardware básico de E/S se ilustra de forma simplificada en la figura 3.4, con posibilidad de entrada analógica. El manual de referencia de la serie 16, muestra en detalle los circuitos equivalentes para pines individuales. Para la entrada, la salida del controlador de corriente se desactiva mediante la carga del bit de dirección de datos con un 1, que desconecta la puerta de tres estados. Los datos se leen en la entrada de retención de datos, desde el mundo exterior cuando su línea de control es impulsada por la CPU durante un registro puerto de instrucción de lectura. Los datos se copian en el registro de la CPU de trabajo para su procesamiento.

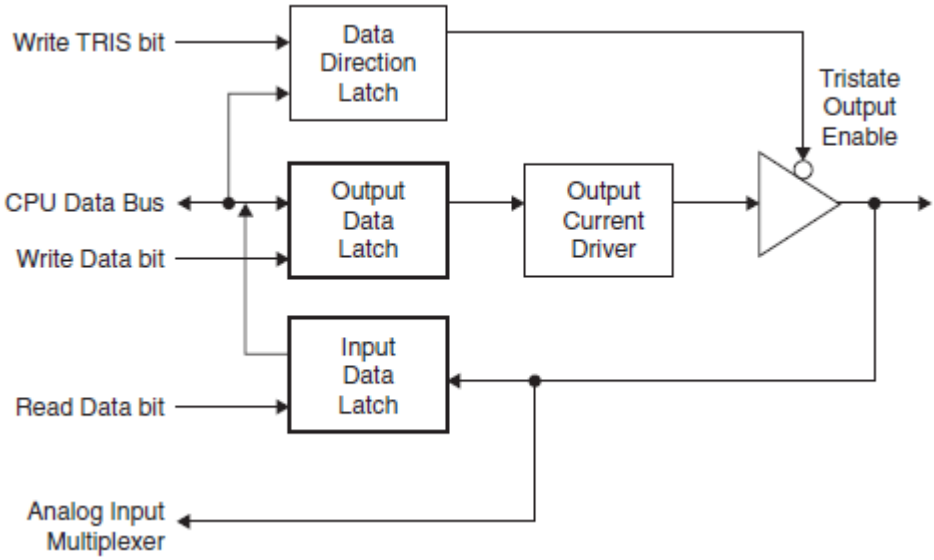

Figura 3. 4: E/S Pin Operación.
Fuente: Bates, M. P. (2010)

Cuando el puerto está configurado para la salida, un 0 se carga en el bit de dirección de datos, permitiendo la salida de corriente. Los datos de salida se cargan en los de retención de datos de la CPU. A los datos de 1 en

la salida permite que el conductor de corriente a la fuente de hasta 25 mA a 5 V, o lo que la tensión de alimentación es (2-6 V). Un dato 0, permite que el pasador se hunda una corriente parecida a 0 V.

El 16F877 tiene los siguientes puertos de E/S digitales disponibles:

Port A	RA0–RA5	6 bits
Port B	RB0–RB7	8 bits
Port C	RC0–RC7	8 bits
Port D	RD0–RD7	8 bits
Port E	RE0–RE2	3 bits
Total digital I/O available		33 pins

La mayoría de los pines (patillas) tienen funciones alternativas, que se describen más adelante.

3.3.2. Timers.

La mayoría de los microcontroladores proporcionan contadores binarios de hardware que permiten una medición de intervalo de tiempo o cuentan, para ser llevado a cabo por separado de la ejecución del programa. Por ejemplo, un período de tren de impulsos de salida fija puede ser generado mientras el programa continúa con otra tarea. Las características de los temporizadores que se encuentran en los microcontroladores PIC, típicamente son representados en la figura 3.5, pero ninguno de los PIC16F877 tiene todas las características mostradas.

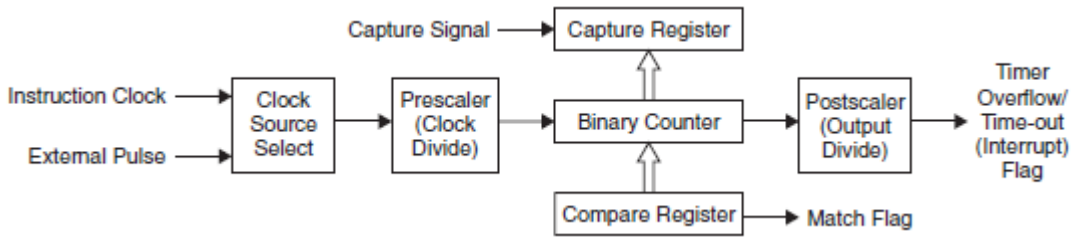

Figura 3. 5:General de Operación del temporizador.

Fuente: (Verle, 2010)

El registro de conteo (contador) más comúnmente utilizado, es operado por conducción desde el reloj de instrucciones interna para formar un temporizador. Esta señal se ejecuta en un cuarto de la frecuencia de reloj; es decir, una instrucción toma cuatro ciclos a ejecutar. Por lo tanto, con un reloj de 4 MHz, el temporizador cuenta en microsegundos (reloj instrucción 1-MHz). El número de bits en el temporizador (8 o 16)

Determina el número máximo (256 o 65 536, respectivamente). Cuando se desborda el registro del temporizador y vuelve a cero, un bit indicador de desbordamiento se establece. Esta bandera se puede consultar (probado) para comprobar si se ha producido un desbordamiento o una interrupción generada, para desencadenar la acción requerida.

Para modificar el periodo de cuenta, el registro del temporizador puede ser cargado con un número dado. Por ejemplo, si un registro de 8 bits se precargado con el valor 156, un tiempo de espera se produce después de $256 - 156 = 100$ relojes. Muchos módulos permiten la precarga del temporizador cada vez que se reinicia, en cuyo caso el valor requerido se almacena en un registro de precarga durante la inicialización del temporizador automático.

Un prescaler típicamente permite que la frecuencia de entrada del temporizador que se divide por 2, 4, 8, 16, 32, 64, o 128 esto se extiende el número máximo proporcionalmente, pero a expensas de la precisión del temporizador. Por ejemplo, el temporizador de 8 bits accionado a 1 MHz con un valor de 4 preescala recuentos de hasta $256 \times 4 = 1024\ \mu s, a\ \mu s$ por bit. Un postscaler tiene un efecto similar, conectado a la salida del contador.

En el modo de comparación, un período separado registro almacena un valor que se compara con la cuenta actual después de cada reloj y el flag de estado cuando coinciden. Este es un método más elegante de modificar el período de tiempo de espera, que se puede utilizar en la generación de una modulada (PWM) de salida de ancho de pulso. Una aplicación típica es para controlar la potencia de salida a una carga de corriente, tal como un pequeño motor de corriente continua-más sobre esto más adelante.

En el modo de captura, la cuenta del temporizador es capturado (con copia a otro registro) en el punto en el tiempo cuando una señal de los cambios externos en uno de los pernos de MCU. Esto puede ser usado para medir la longitud de un impulso de entrada o el período de una forma de onda.

El '877 tiene tres registros de contador / temporizador. Timer0 tiene un contador de 8 bits y 8 bits prescaler. Se puede velocidad de reloj desde el reloj instrucción o una señal externa aplicada a RA4. El prescaler también se

puede utilizar para extender el intervalo de temporizador de vigilancia (véase más adelante), en cuyo caso no está disponible para su uso con Timer0. Timer1 tiene un contador de 16 bits y prescaler y puede ser ajustado internamente o externamente como por Timer0. Ofrece captura y

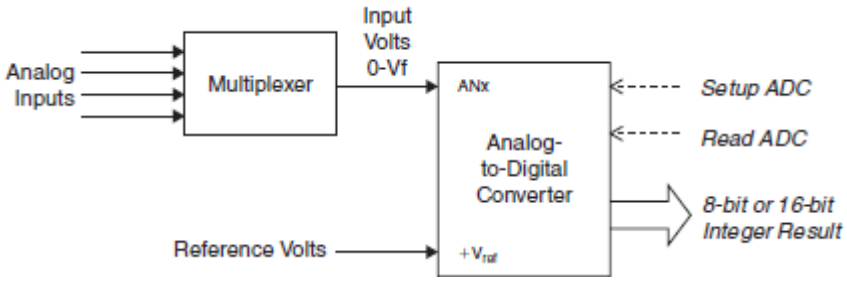

Figura 3. 6:Operación ADC.
Fuente: (Verle, 2010)

Comparar los modos de funcionamiento. Timer2 es otro contador de 8 bits, pero tiene tanto un prescaler y postscaler (hasta 01:16) y comparar registrarse para el control época.

Se proporcionan más detalles en la interconexión Microcontroladores PIC por el autor y los libros de datos MCU. Al programar en C, es necesario un conocimiento limitado de la operación del temporizador, como las funciones de C generalmente se encargan de los detalles.

3.3.3. A/D Converter.

Ciertos pines del PIC se pueden configurar como entradas a un convertidor de analógico al digital (ADC). El '877 tiene ocho entradas analógicas, que están conectados al puerto A y puerto E. Cuando se utiliza en este modo, se les conoce como AD0-AD7. Los registros de control

necesarios se inicializan en CCS C usando un conjunto de funciones que permiten el modo de operación ADC e insumos a ser seleccionados.

Una instrucción adicional "dispositivo" en la parte superior del programa establece la resolución del ADC. Una tensión analógica presentado en la entrada se convierte a continuación en binario y se invoca el valor asignado a una variable entera cuando la función para leer el ADC.

El rango de entrada por defecto es fijado por la oferta (nominalmente 0-5 V). Si se utiliza un suministro de batería (que cae en el tiempo) o se necesita una precisión adicional, una tensión de referencia separado puede ser alimentado en al AN2 $(+V_{ref})$ y opcionalmente AN3 $(-V_{ref})$. Si tan sólo Se utiliza $+V_{ref}$, el límite inferior se mantiene 0 V, mientras que la superior se establece por la tensión de referencia. Esta se suministra típicamente usando un diodo zener y divisor de tensión.

El 2.56 V derivado de un zener 2V7 da un factor de conversión de 10 mV por bit para una conversión de 8 bits. Para una entrada de 10 bits, una referencia de 4,096 V puede ser conveniente, dando una resolución de 4 mV por bit. Los elementos esenciales de la operación ADC se ilustran en la figura 3.6.

3.3.4. Comparador.

El comparador (véase la figura 3.7) es un tipo alternativo de entrada analógica se encuentra en algunos microcontroladores, tales como el 16F917 utilizado en el tablero de la mecatrónica se describe más adelante.

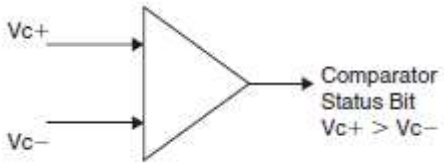

Figura 3. 7:comparador Operación.
Fuente: Ejemplos del Software Microcode Pic Basic

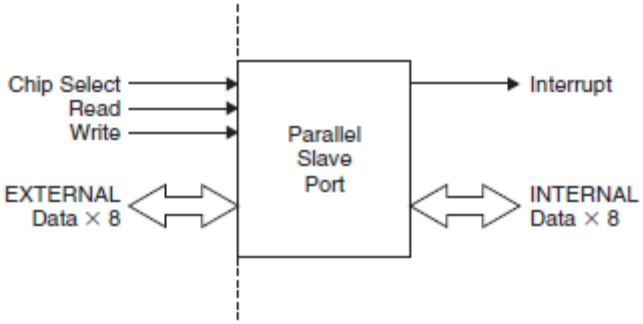

Figura 3. 8: Operación del puerto paralelo esclavo.
Fuente: Ejemplos del Software Microcode Pic Basic

En él se compara la tensión en un par de entradas, y un bit de estado se establece si el $C+$ pin es mayor que $C-$. El bit de estado comparador también puede ser monitoreado en un pin de salida. El '917 tiene dos de tales módulos de comparación; están habilitados utilizando una función del sistema para ajustar el modo de operación. El '877 no tiene comparadores, por lo que el ADC debe ser utilizado en su lugar.

3.3.5. Puerto Paralelo Esclavo.

El puerto esclavo paralelo en el '877 chip está diseñado para permitir las comunicaciones paralelo con un sistema de bus de datos de 8 bits externa o periférica (Figura 3.8). Port D proporciona los pines de datos de ocho de E / S, y el puerto E tres líneas de control: lectura, escritura, y selector de chip. Si los datos han de ser de entrada al puerto, la dirección de datos pin tiene el valor adecuado y los datos presentados a Port D. La selección de chip de entrada se debe establecer bajo y los datos enclavados en el registro de datos del puerto, tomando la línea baja de escritura. Por el contrario, los datos pueden ser leídos desde el puerto usando la línea de lectura. Cualquiera de operación se puede iniciar una interrupción.

3.3.6. Interrupciones.

Las interrupciones pueden ser generados por diversos eventos de hardware internos o externos. Se estudian con más detalle más adelante en relación con la programación de las operaciones periféricas. Sin embargo, en esta etapa, es útil tener una idea acerca de las opciones de interrupción previstas en el MCU. Tabla 3.3 se enumeran los dispositivos que pueden configurarse para generar una interrupción.

La manera más efectiva de la integración de las operaciones de temporizador en un programa de aplicación es mediante el uso de una interrupción de temporizador. La figura 3.9 muestra una secuencia de programa que se ejecuta un temporizador para generar un intervalo de pulso de salida. Una rutina de interrupción (ISR) se ha escrito y se asigna a la

interrupción del temporizador. El temporizador se configura durante el inicio del programa y comenzó por la precarga o borrarlo.

Tabla 3. 3Interrupciones Fuentes del PIC16F877.

Interrupt Source	Interrupt Trigger Event	Interrupt Label
Timers		
Timer0	Timer0 register overflow	INT_TIMER0
Timer1	Timer1 register overflow	INT_TIMER1
CCP1	Timer1 capture or compare detected	INT_CCP1
Timer2	Timer2 register overflow	INT_TIMER2
CCP2	Timer2 capture or compare detected	INT_CCP2
Ports		
RB0/INT pin	Change on single pin RB0	INT_EXT
Port B pins	Change on any of four pins, RB4-RB7	INT_RB
Parallel Slave Port	Data received at PSP (write input active)	INT_PSP
Analog Converter	A/D conversion completed	INT_AD
Analog Comparator	Voltage compare true	INT_COMP
Serial		
UART Serial Port	Received data available	INT_RDA
UART Serial Port	Transmit data buffer empty	INT_TBE
SPI Serial Port	Data transfer completed (read or write)	INT_SSP
I^2C Serial Port	Interface activity detected	INT_SSP
I^2C Serial Port	Bus collision detected	INT_BUSCOL
Memory		
EEPROM	Nonvolatile data memory write complete	INT_EEPROM

Fuente:(Reyes, 2008)

El principal programa de conteo y el contador a continuación, proceder al mismo tiempo, hasta que se produce un tiempo de espera y se genera la interrupción. El programa principal se suspende y el ISR ejecutado. Al terminar, el programa principal se reanuda en el punto inicial. Si el ISR contiene una instrucción para cambiar un bit de salida, una onda cuadrada se podría obtener con un periodo de dos veces el retraso del temporizador.

Cuando se utilizan las interrupciones en los programas en lenguaje ensamblador, es más fácil predecir el efecto, ya que el programador tiene un control más directo sobre la secuencia exacta de la ISR.

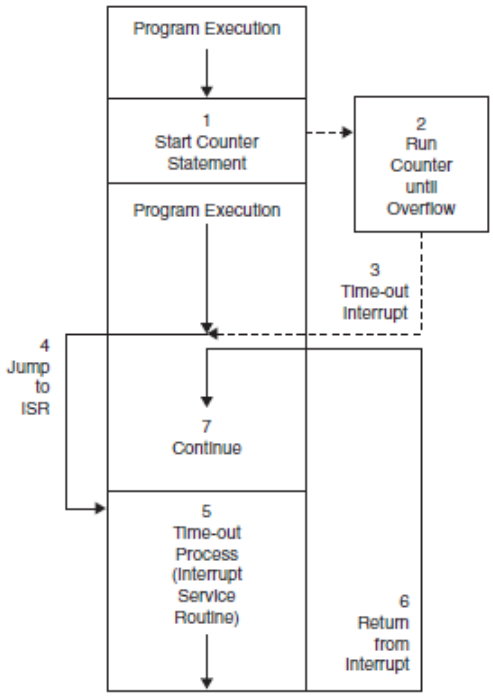

Figura 3. 9:Proceso de Interrupción de temporizador.
Fuente: Ejemplos del Software Microcode Pic Basic

Un programa en C se genera automáticamente por el compilador, por lo que el momento preciso que resulta de una interrupción es menos obvio. Por esta razón, el uso de un sistema operativo en tiempo real (RTOS) a veces se prefiere en el entorno de C, especialmente cuando los programas se hacen más complejos. De hecho, C fue desarrollado originalmente para precisamente este propósito, para escribir sistemas operativos para ordenadores.

3.4. Componente y operadores en Pic Basic.

- USART enlace asíncrono
- SPI bus síncrono
- I2C bus síncrono

Conexiones de datos en serie son útiles porque sólo se necesitan uno o dos cables de señal, en comparación con al menos ocho líneas de datos para unas más señales de bus de control paralelos. La típica

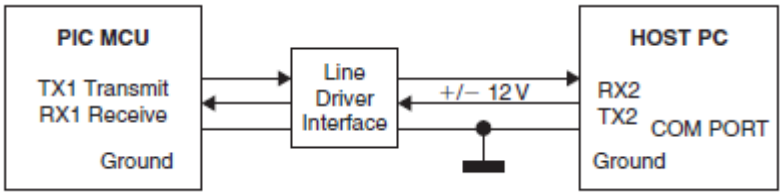

Figura 3. 10:USART Operación.
Fuente: Ejemplos del Software Microcode Pic Basic

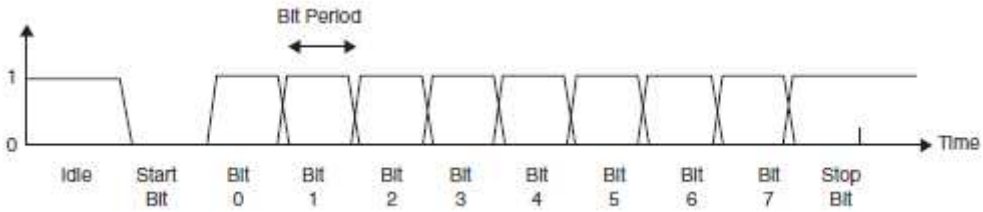

Figura 3. 11:USART RS232 Señal.
Fuente: Ejemplos del Software Microcode Pic Basic

Microcontrolador PIC ofrece una selección de interfaces seriales. El mejor para cualquier canal de comunicación dada depende de la distancia entre los nodos, la velocidad y el número de conexiones de hardware requerido.

3.4.1. USART.

El síncrono / asíncrono universal de recibir transmisión (USART) del dispositivo se utiliza normalmente en modo asincrónico para cumplir off-board, uno-a-uno las conexiones. El término asíncrono significa que se necesita ninguna señal de reloj separada a tiempo de la recepción de datos, por lo que sólo un envío de datos, recepción de datos, y se necesitan cables de tierra. Es rápido y simple de implementar si un ancho de banda limitado de datos es aceptable.

Una aplicación común es conectar el chip PIC a un PC host para cargar los datos adquiridos por el subsistema de MCU (ver figura 3.10). El enlace USART puede enviar datos de hasta 100 metros mediante la conversión de la señal a los niveles de mayor tensión (normalmente $\pm 12V$). La señal digital se invierte y se desplaza para convertirse en bipolar (simétrica alrededor de 0 V, línea negativa cuando inactivo) para la transmisión.

El PIC 16F877 tiene un puerto dedicado RS232 hardware, pero CCS C permite en un punto para ser configurado como un puerto RS232, que proporciona funciones para generar las señales en el software. La forma básica de la señal tiene 8 bits de datos y una parada y bit de inicio. El período de bit es activado por la tasa de baudios. Un valor típico es de 9600 baudios, que está a unos 10 k bits por segundo. El período de bit es entonces aproximadamente 100 μ s, alrededor de 1 byte por milisegundo, o 1 K byte por segundo.

Los datos se transfieren entre registros de desplazamiento que operan a la misma velocidad de bits; el receptor tiene que ser inicializado a los mismos baudios de ajuste que el transmisor. Suponiendo que estamos viendo los datos de nivel TTL, en estado de reposo, la línea es alta. Cuando se pasa a nivel bajo, se inicia el reloj del receptor, los datos se muestrean en el centro de cada siguiente periodo de bit de datos, y los datos se desplazan en el registro de recepción (véase la figura 3.11).

RS232 se utiliza para acceder a la pantalla LCD serial estándar, en cuyo caso, los conductores de línea no se requiere necesariamente. Caracteres ASCII y códigos de control se envían a operar la pantalla, que tiene su propio MCU con una interfaz en serie para recibir y descodificar los datos. Se acciona entonces la matriz de píxeles para visualizar caracteres alfanuméricos. La mayoría de las pantallas LCD también se pueden configurar para mostrar gráficos de mapas de bits simples. En el modo de simulación, una terminal virtual RS232 proporciona una manera conveniente de generar de entrada alfanumérico en el MCU para la prueba. Los códigos ASCII se listan en la Tabla 2.5.

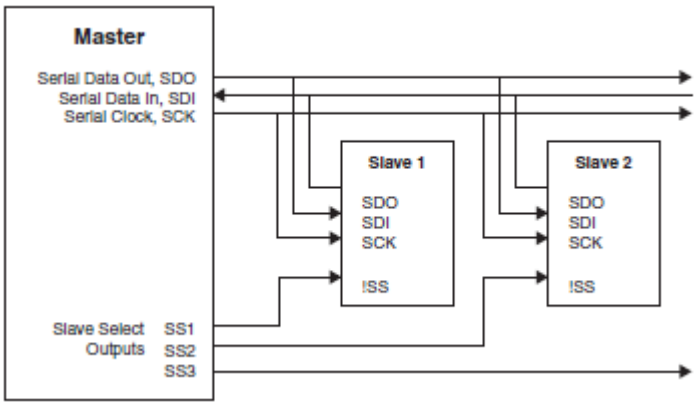

Figura 3. 12:Conexiones SPI.
Fuente: Ejemplos del Software Microcode Pic Basic

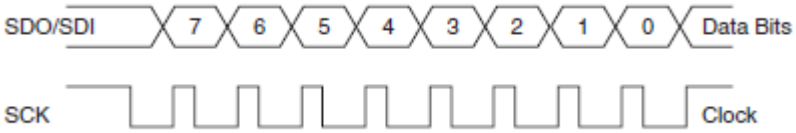

Figura 3. 13:Señales SPI.
Fuente: Ejemplos del Software Microcode Pic Basic

CAPÍTULO 4: DESARROLLO EXPERIMENTAL

En el presente capítulo se desarrollará la parte demostrativa experimental de las aplicaciones con microcontroladores mediante la plataforma de programación virtual LabVIEW. Esto nos ha permitido diseñar algo novedoso que hasta la presente jamás se ha realizado en la Facultad de Educación Técnica para el Desarrollo, que beneficiará a los estudiantes en incrementar sus conocimientos en hardware (dispositivos electrónicos) y software (LabVIEW y MikroBasic).

4.1. Práctica 1: Comunicación serial.

En esta práctica se realizará el envío de datos mediante comunicación serial. Para lo cual se procederá a programar el módulo de entrenamiento MEI&T04 para que envíe mediante puerto serial el conteo ascendente de una variable de 8 bits (0-255), dicho envío se realizará cada 500 milisegundos.

Posteriormente, se monitoreael dato recibido en código ASCII por el AccesPort.El corazón del módulo de entrenamiento programableMEI&T04, es el microcontrolador PIC16F886, por tanto, debemos considerar lo importante de conocer las pines de conexión del uC PIC16F886 (véase la figura 4.1), tales como:

- Programación serial ICSP: MCLR (Pin1), ICSPDAT (Pin28), ICSPCLK (Pin27), V_{SS} (GND), V_{DD} (V_{CC}).

- Entradas analógicas: AN0, AN1, AN2, AN3, AN4, AN8, AN9, AN10, AN11, AN12, AN13.
- Modulación de Ancho de Pulso (PWM): CCP1 y CCP2.
- Comunicación serial: TX y RX.

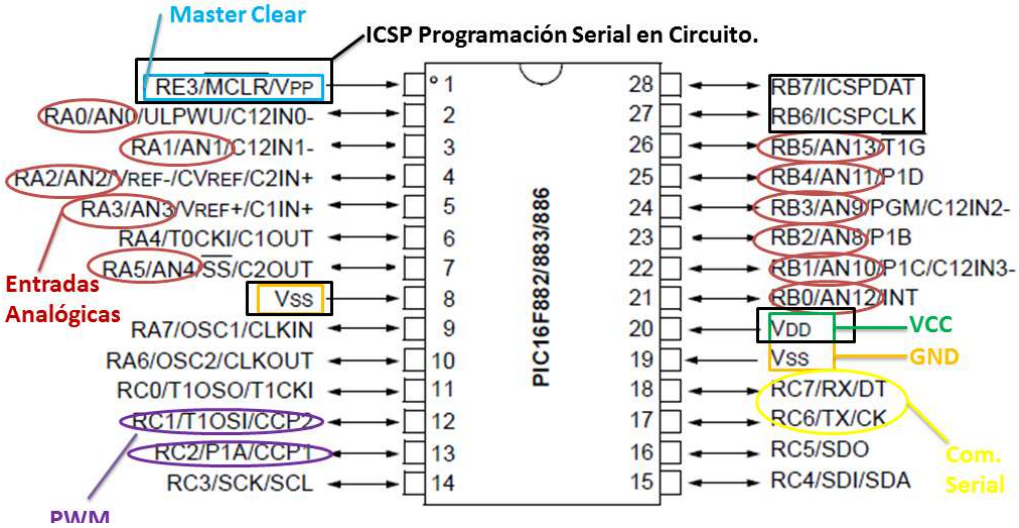

Figura 4. 1: Configuración de los pines del uC PIC16F882/883/886.
Fuente: Los autores

Es necesaria la configuración del microcontrolador, es decir, configurar los registros (véase la figura 4.2), entre los cuales se destacan:

- Registro TRIS: registro de 8 bits que permite configurar los puertos (A, B, C, E) del microcontrolador pin a pin como entrada o salida.
- Registro ANSEL y ANSELH: Nos permite configurar como entradas digitales o analógicas de los pines que poseen la habilidad de digitalizar entradas analógicas.
- Registro PORT: Nos permite obtener el estado lógico booleano de las entradas digitales, sin embargo a los pines que son salida nos permite poner un estado lógico booleano.

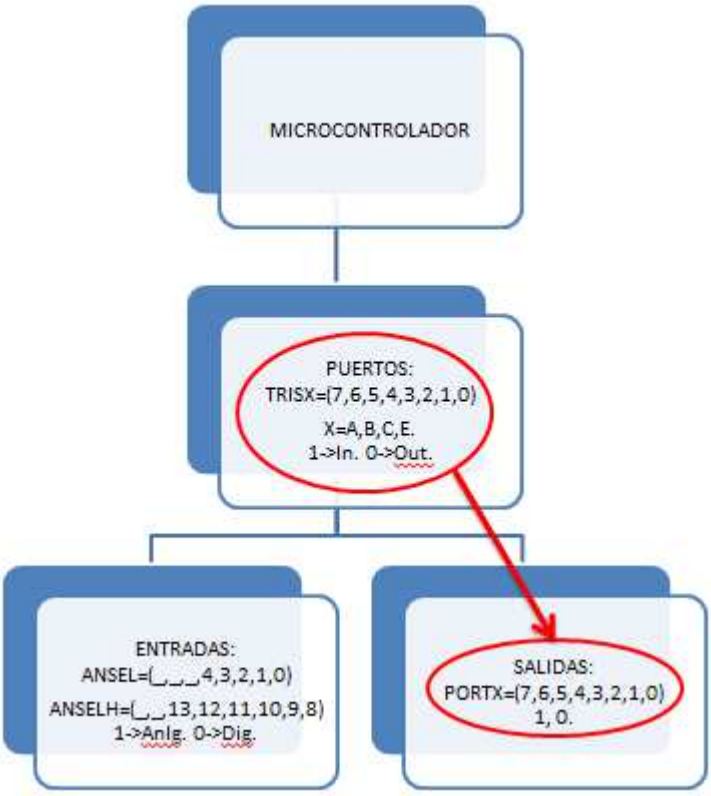

Figura 4. 2: Configuración de los registros del uC PIC16F886.
Fuente: Los autores

En la figura 4.3 se resaltan los elementos necesarios de la tarjeta de entrenamiento MEI&T04 para que poder realizar la comunicación serial.

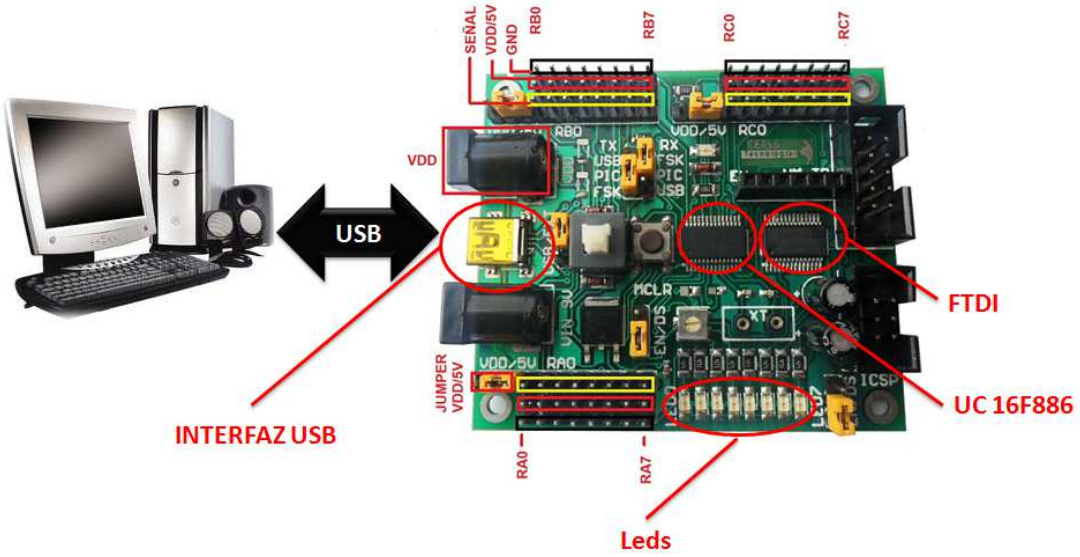

Figura 4. 3: Elementos requeridos para la práctica 1.
Fuente: Los autores

A continuación, se muestra el código de programación que permite configurar al PIC16F886 de acuerdo a lo descrito con anterioridad.

```
program ejemplo2
' Declarations section
'1)----------SYMBOL.----------
SYMBOL LEDS=PORTB
'2)-----CREAR Y SET VARIABLES-------
DIM CNT AS BYTE
DIM txt as string[3]
CNT=0
'3)------CONFIG IN (1)- OUT (0)------
TRISA=%00000001 '<7,6,5,4,3,2,1,0>
TRISB=%00000000 'Leds -> OUT
TRISC=%00000000 '<7,6,5,4,3,2,1,0>
TRISE=%00000000 'RE3 -> IN BOTONERA
'4)--CONFIG IN: DIG (0)- ANALOG (1)---
ANSEL= %00000000 '<_,_,_,An4,An3,An2,An1,An0>
'An0->A0, An1->A1, An2->A2, An3->A3, An4->A5
ANSELH= %00000000
'<_,_,An13,An12,An11,An10,An9,An8>
'An8->B2, An9->B3, An10->B1, An11->B4, An12->B0,
An13->B5

'5)--- COMUNICACIÓN-------
UART1_Init(9600)
' Initialize UART module at 9600 bps
Delay_ms(50)

main:
'   Main program
   while(1)
     inc(CNT)
     LEDS=CNT
     ByteToStr(CNT, txt)
     UART1_Write_text(txt)' Dato
     delay_ms(500)
   wend
end.
```

Para poder cargar el firmware, se deberá realizar lo siguiente:

1. Conectar el programador de microcontroladores al módulo de entrenamiento MEI&T04, tal como se muestra en la figura 4.4.
2. Conectar el hardware (programador) al computador y haciendo uso del software (programación de alto nivel) MikroBasic, y luego importar el archivo “.HEX”.
3. Proceder a programar el módulo de entrenamiento MEI&T04.

Es importante que luego de la programación del código con comunicación serial UART, debamos cargar el código en el módulo de entrenamiento. Resultó ser una excelente experiencia práctica, para comprobar que el código utilizado se comunique mediante el módulo de

entrenamiento(ver figura 4.4). De fácil conexión al computador y de verificación, la comunicación serial con tramas de comunicación, para lo cual usaremos el (Hiperterminal) AccesPort.

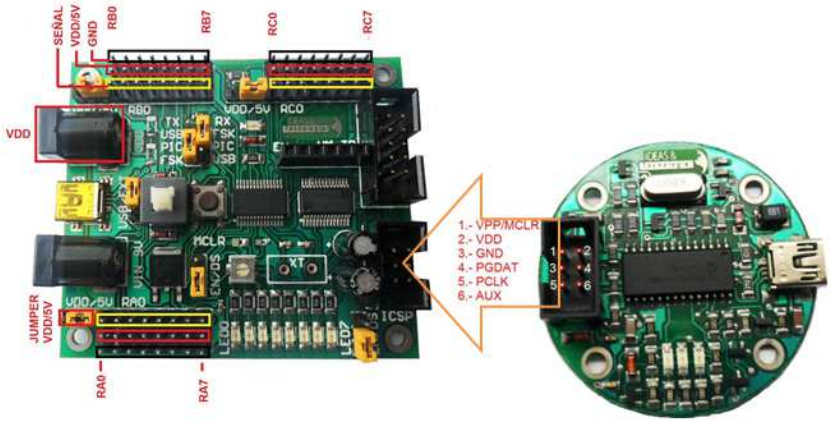

Figura 4. 4: Conexión y configuración entre los pines del programador y entrenador.
Fuente: Los autores

Para la recepción de datos por comunicación serial, se procederá con la creación de un VI (véase la figura 4.5) en Labview, el que nos permita monitorear el dato recibido en código ASCII por el AccesPort.

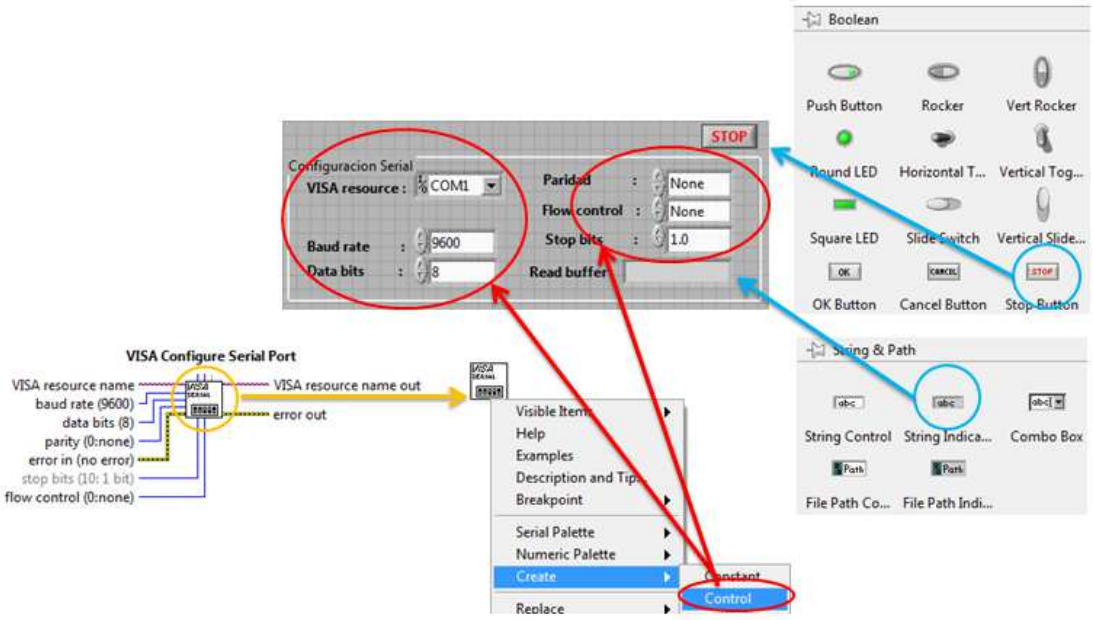

Figura 4. 5: Programación gráfica de la comunicación serial.
Fuente: Los autores

En la figura 4.6 se muestra la configuración en detalle del VI visualizado en la figura 4.5, el mismo que nos demuestra cómo se realiza la comunicación serial.

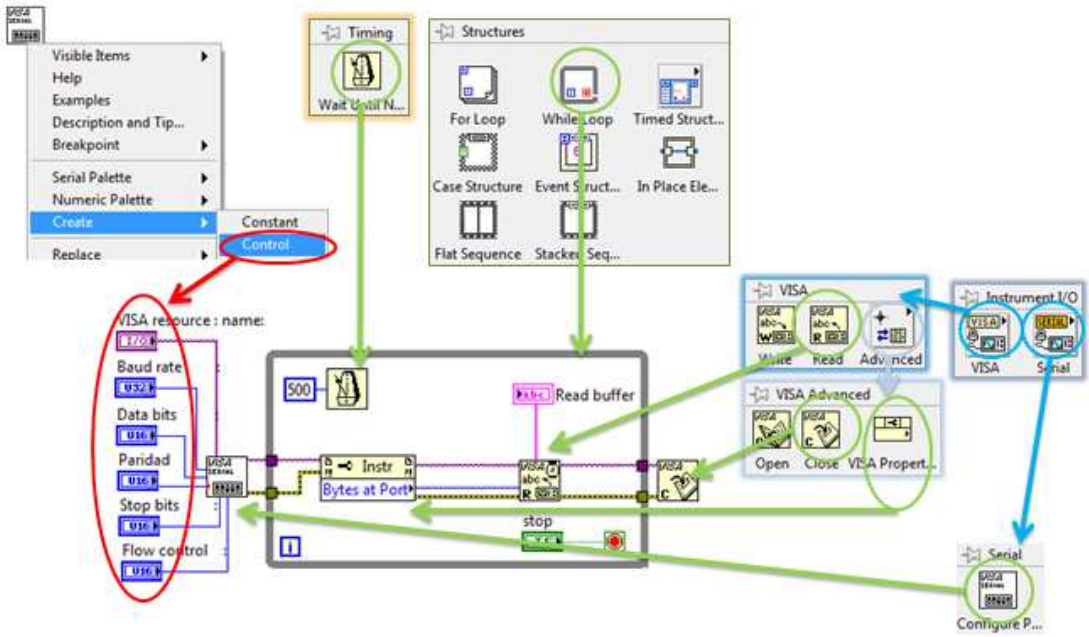

Figura 4. 6: Configuración en detalle de la comunicación serial.
Fuente: Los autores

De igual forma que el hardware usaremos el AccesPort para verificar las tramas de comunicación con el Software Labview, pero debido a que ambos están en el computador, utilizaremos el “Virtual Serial Por Emulator” para la creación de dos puertos de comunicación serial virtual y hacer las pruebas de la comunicación serial, tal y como se muestra en la figura 4.7.

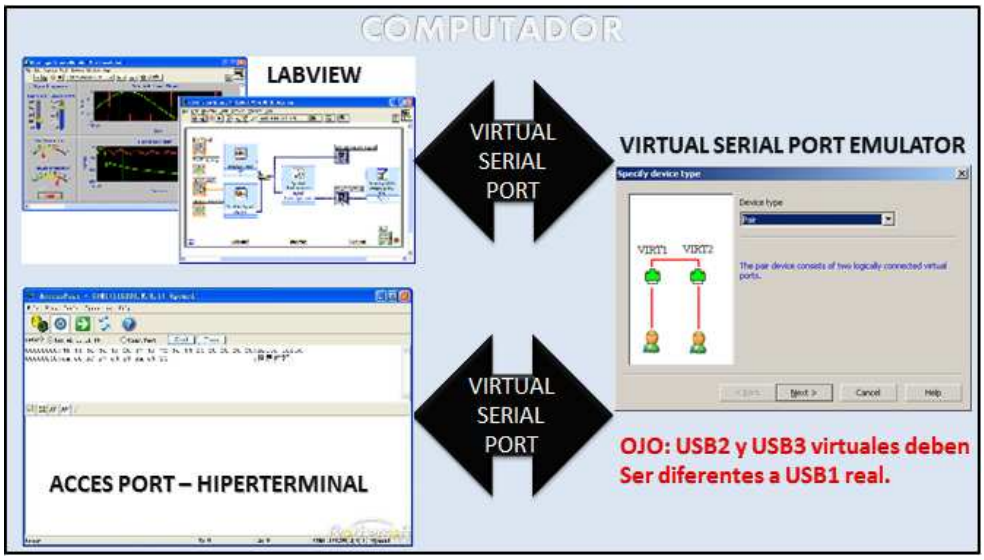

Figura 4. 7: Comprobación de la comunicación serial.
Fuente: Los autores

Finalmente luego de las pruebas independientes tanto de hardware y software, conectamos con seguridad el hardware del módulo de entrenamiento MEI&T04 con LabView, haciendo para ello la conexión con el puerto USB el computador y el micro USB del módulo de entrenamiento.

4.2. Práctica 2: Salidas Digitales.

Para esta práctica se realizará el manejo de ocho LEDs Seriales, para lo cual programaremos el módulo de entrenamiento MEI&T04, para que envíe por puerto serial el valor en byte (Trama de datos de 1 byte) de correspondiente al estado lógico de encendido o pagado de cada uno de los ocho leds. El control se realizará en LabView mediante el testeo de ocho Botones.

Así como en la primera práctica, es necesaria la configuración de los registros (véase la figura 4.8), entre los cuales se destacan:

- Registro TRIS: registro de 8 bits que permite configurar los puertos (A, B, C, E) del microcontrolador pin a pin como entrada o salida.
- Registro ANSEL y ANSELH: Nos permite configurar como entradas digitales o analógicas de los pines que poseen la habilidad de digitalizar entradas analógicas.
- Registro PORT: Nos permite obtener el estado lógico booleano de las entradas digitales, sin embargo a los pines que son salida nos permite poner un estado lógico booleano.

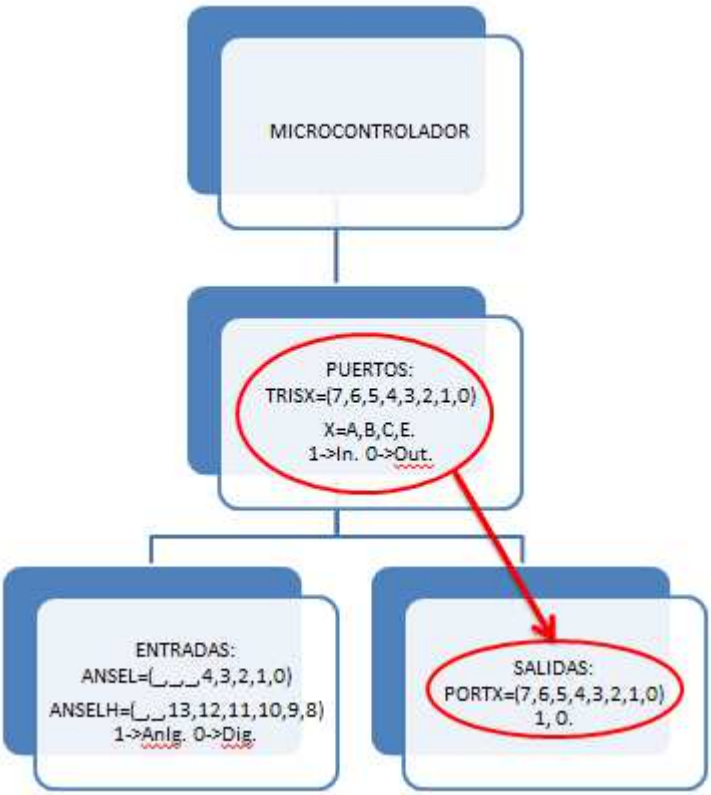

Figura 4. 8: Configuración de los registros del uC PIC16F886.
Fuente: Los autores

En la figura 4.9 se muestra el módulo (hardware) de entrenamiento (ver figura 4.4) para visualizar el encendido de LEDs.

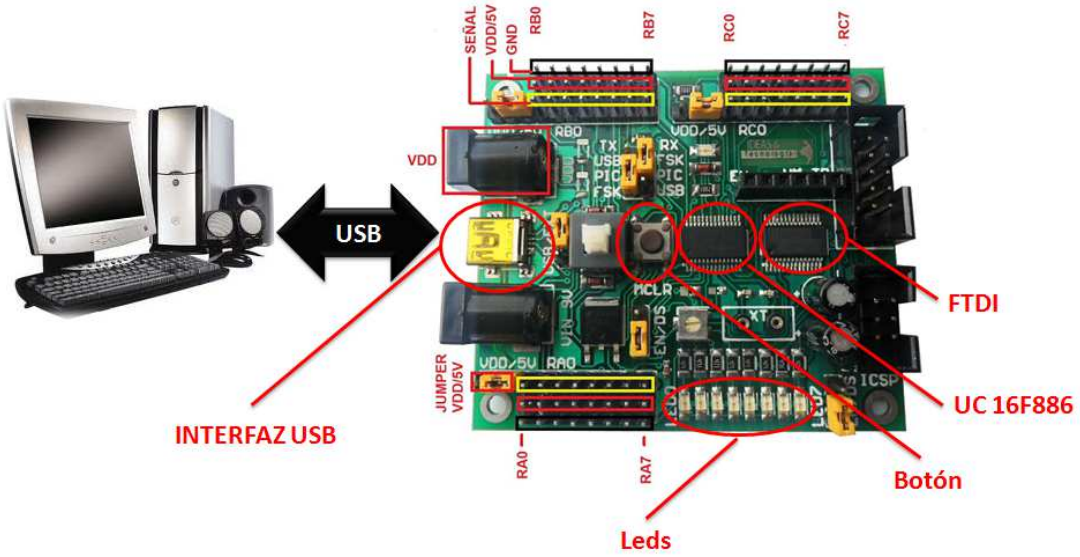

Figura 4. 9: Elementos requeridos para la práctica 2.
Fuente: Los autores

A continuación, se muestra el código de programación que permite al PIC16F886 manejar LEDs seriales cuando recibe la trama de datos desde labview.

```
program MANEJO_LEDS_LABVIEW
'----------SYMBOL.----------
SYMBOL LEDs=PORTB
'-----CREAR VARIABLES-------
DIM DATO AS BYTE

main:
'------CONFIG IN (1)- OUT (0)------
TRISA=%00000000 'Sin Uso
TRISB=%00000000 'Leds -> OUT
TRISC=%10000000 'C7 -> Rx Uart
TRISE=%00001000 'RE3 -> IN BOTONERA
 'MCLR is external / Digital input -> Edit Proyect

'—Set de valores booleanos en salidas de registros---
PORTA= 0X00 '-en binario-%00000000
PORTB = 0X00
PORTC = 0X00
PORTE = 0X00

'--CONFIG IN: DIG (0)- ANALOG (1)---
ANSEL= %00000000 '<_,_,_,An4,An3,An2,An1,An0>
'An0->A0, An1->A1, An2->A2, An3->A3, An4->A5
ANSELH= %00000000
'<_,_,An13,An12,An11,An10,An9,An8>
'An8->B2, An9->B3, An10->B1, An11->B4, An12-
>B0, An13->B5

'-----SET DE VARIABLES----
DATO=0

'------COMUNICACION SERIAL----
UART1_Init(9600)
PORTB=%10101010
DELAY_MS(200)
PORTB=%01010101
DELAY_MS(200)

while (1)
'—Validamos si hay datos listos para leer--
  IF(UART1_Data_Ready =1)THEN
     DATO = UART1_Read()
     LEDs=DATO
  END IF
wend
end.
```

En la figura 4.10 se muestra el VI utilizado para la presente práctica, en la cual se visualiza el LED para comprobar el encendido del mismo.

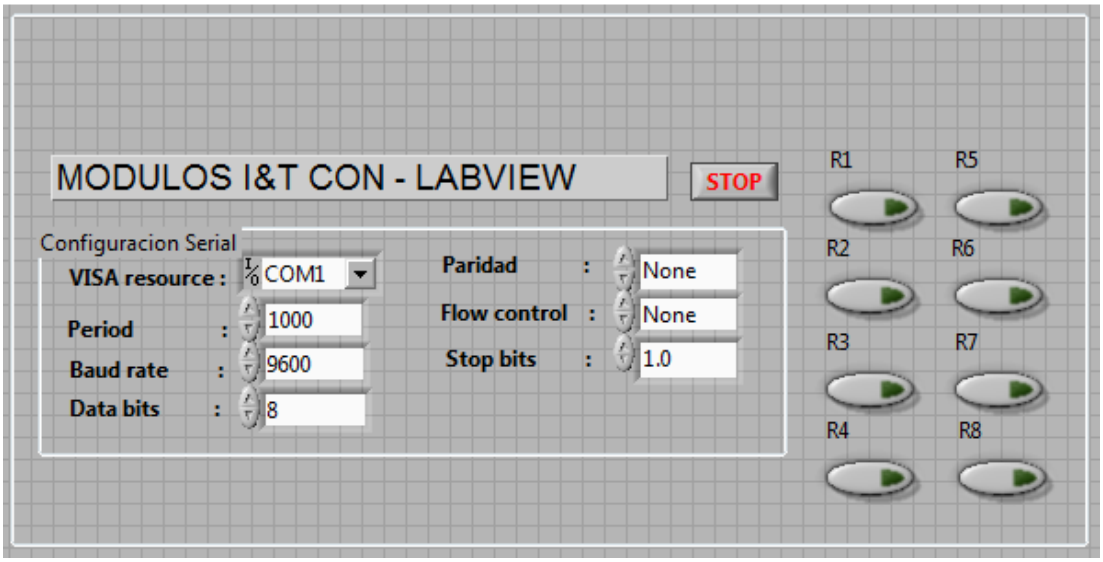

Figura 4. 10: Programación gráfica de la comunicación serial.
Fuente: Los autores

En la figura 4.11 se muestra la configuración en detalle del VI visualizado en la figura 4.10.

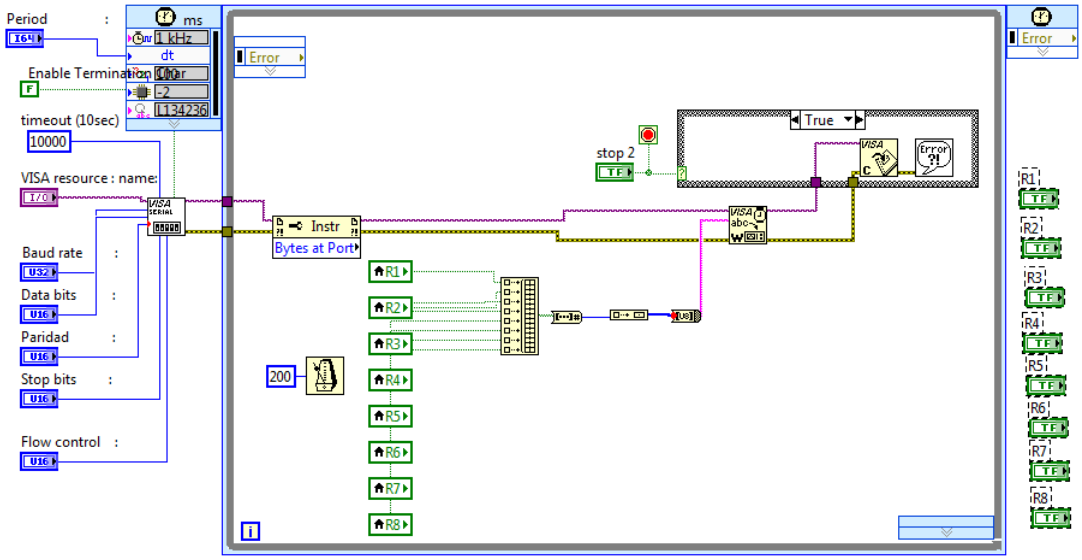

Figura 4. 11: Configuración en detalle del manejo LEDs serial.
Fuente: Los autores

CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES.

5.1. Conclusiones.

- A través de la fundamentación teórica también conocida como Estado del Arte, nos permitió comprender la importancia de los microcontroladores que permiten realizar diferentes aplicaciones relacionadas a las Ciencias Aplicadas, tanto para Carreras como Ingeniería en Telecomunicaciones como la de Electrónica en Control y Automatismo, donde se pueden integrar la programación de instrumentación virtual VIs.

- Se describió las plataformas de programación tanto de LabVIEW como lenguaje de alto nivel; y MikroBasic la cual permitió realizar el código fuente para el microcontrolador PIC16F886 (hardware).

5.2. Recomendaciones.

- Es necesario que la Facultad de Educación Técnica para el Desarrollo, compren licencias profesionales de algunos programas que permiten desarrollar proyectos mediante los microcontroladores, tales como Pic Basic, Mikro Basic, Microcode Estudio y Proteus Profesional.

REFERENCIAS BIBLIOGRÁFICAS

Acevedo L., C., & Rueda B., R. (2010). Implementación de LabView como Sistema Scada para la arquitectura de control SNAC PAC OPTO 22, mediante una aplicación OPC. Bucaramanga: Universidad Pontificia Bolivariana.

Gridling, G. y. (2007). Introduction to Microcontrollers.Viiena, Austria: Vienna University of Technology.

Hermosa Donate, A. (2010). Electrónica digital fundamental y programable: curso profesional teoría-práctica. Barcelona: Marcombo.

Lajara V., J. R., & Pelegrí S., J. (2011). LabView: Entorno gráfico de programación (Segunda ed.). Barcelona: Marcombo, S.A.

Mandado Pérez, E., Menéndez Fuertes, L., Fernández Ferreira, L., & López Matos, E. (2007). Microcontroladores PIC: Sistema Integrado para el Autoaprendizaje. Barcelona, España: Marcombo.

Reyes, C. A. (2008). Microcontroladores PIC Programación en Basic. Quito, Ecuador: RISPERGRAF.

Santamaría, J. (2009). Herramientas Computacionales para Control Sesión LabView. Bucaramanga: UPB - Facultad de Ingeniería Electrónica.

Staff, U. (2011). Microcontroladores: Funcionamiento, Programación y Usos Prácticos. Buenos Aires: USERS.

Valdés Pérez, F. E., & Pallás Areny, R. (2007). Microcontroladores: fundamentos y aplicaciones con PIC. Cataluña, España: Marcombo.

Valverde Villarán, A. (2005). Sistema de Desarrollo para el Microcontrolador PIC18F452. Sevilla: Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación.

Verle, M. (2010). PIC Microcontrollers - Programming in Basic. Belgrado: mikroElektronika.

Bates, M. P. (2010). *Programming 8-Bit PIC Microcontrollers in C with Interactive Hardware Simulation*. Editorial Newnes.

González C., R., & Pradines P., R., (2007). *Análisis de Software para desarrollo entorno gráfico LabVIEW y propuesta de implementación para Laboratorio en el Instituto de Electricidad y Electrónica en Universidad Austral de Chile.* Trabajo de Titulación para optar al Título de Ingeniero Electrónico. Universidad Austral de Chile.

Molina M., J. M. & Ruiz C., A. (2010). *Automatización y telecontrol de sistemas de riego*. Marcombo S.A., Barcelona – España.

Borja S., J. E., & Jiménez M., R. R. (2011). *Diseño de un sistema de monitoreo de alarmas para un edificio hospitalario bajo la plataforma LabView.* Tesina de Seminario para obtener el Título de Ingeniero en Electrónica y Telecomunicaciones. Facultad de Ingeniería en Electricidad y Computación – ESPOL, Guayaquil.

Pincay S., E. (2014). *Sistema SCADA para medición de tensión y puentes con sistemas embebidos para Laboratorio de Control de la Facultad de Educación Técnica.* Trabajo de Titulación para obtener el Título de Ingeniero en Eléctricomecanica. FETD – UCSG, Guayaquil.