UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Sistema inalámbrico de medición, control y monitoreo de flujo de líquidos no viscosos.

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO ELÉCTRICO Y ELECTRÓNICO

PRESENTA:

EFRAÍN FLORES FLORES

Director de tesis:

M.I. José Castillo Hernández

MÉXICO, D.F. a 3 de Diciembre de 2014.

A mis padres, Judith flores y Guillermo Barrios

AGRADECIMIENTOS

A la **Universidad Nacional Autónoma de México,** por permitirme formar parte de ella, y por formarme como profesionista, por todas las cosas y oportunidades que me ha brindado. ¡Gracias!

A mis compañeros de clases, que el destino hizo que tuviera la oportunidad de conocer y convivir con ellos durante mi estancia en la universidad.

A todos mis profesores, por sus enseñanzas, su dedicación, y su tiempo, ya que ellos son la parte más importante en mi formación académica, y son el alma de la universidad.

A mis amigos de la facultad, que son lo más valioso que he tenido estos años, por todos esos momentos en los que conté con ellos incondicionalmente, por reconfortarme, por escucharme, por pasar buenos y malos momentos juntos.

A Cecilia Fernanda Giménez Treviño, y Gerardo Esteban Chaparro, por brindarme su apoyo, su cariño, y su amistad.

A la Ing. Evelyn Salazar Guerrero y al M.I. Luis Arturo Haro Ruiz por aceptar formar parte de este proyecto y colaborar conmigo.

A mis queridos colegas Mauro Gilberto López, y Servando Rafael por contagiarme de su entusiasmo y su alegría.

A la gente que trabaja en el laboratorio de electrónica del CCADET, al M.I. José Castillo Hernández por permitirme realizar el servicio social con él, y por dirigir mi tema de tesis, a los M.I. Juan Ricardo Damián Zamacona, y Sergio Quintana Thierry, por su grata compañía durante mi estancia en el laboratorio.

JURADO

Presidente:	M.I. Luis Arturo Haro Ruiz
Vocal:	M.I. José Castillo Hernández
Secretario:	M.I. Sergio Quintana Thierry
1er. Suplente:	M.I. Juan Ricardo Damián Zamacona
2do. Suplente:	ING. Evelyn Salazar Guerrero

Director de Tesis:

M.I. José Castillo Hernández

Índice

Capítulo 1 Introducción

Capítulo 2 Conceptos básicos

Capítulo 3 Instrumentación del sensor de flujo

Capítulo 4 Comunicación inalámbrica

Capítulo 5 Desarrollo del panel de control virtual mediante LabView

Capítulo 6 Comunicación del PIC con LabView

Capítulo 7 Resultados

Conclusiones……86

Capítulo 1

Introducción

1.1. Planteamiento del problema

En la sociedad actual la vida de las personas gira en torno a una necesidad de consumir productos y recursos de todo tipo, dentro de los más importantes están los líquidos, algunos de los cuales son altamente demandados, consumidos, y necesarios para mantener el estilo de vida de las personas, y contribuyen a la economía del país. Tal es el caso del agua potable y la gasolina.

La importancia de la medición y control de líquidos, radica en la elaboración de productos y materiales que para su producción, requieren de procesos en los que es necesario controlar el volumen de los materiales líquidos que intervienen.

En la industria una importante variable a medir es el flujo volumétrico de líquidos que intervienen en diferentes procesos, la correcta medida, control, y monitoreo de esta variable, es crucial para garantizar la calidad de los productos que en su proceso de fabricación lo involucran.

Las industrias invierten mucho dinero en sistemas que permiten cuantificar la transferencia de productos líquidos para su venta. El no hacerlo adecuadamente podría traducirse en pérdidas económicas. La importancia de la medición, y control del flujo volumétrico, no solo radica en la facturación de importación o exportación de productos líquidos. Sino que también nos brinda un importante parámetro para conocer la eficiencia en procesos, la relación de reacciones químicas, razones de producción, y balance de materiales líquidos en diversos procesos industriales, entre otras cosas.

Para medir el flujo volumétrico de líquidos existen diversos dispositivos en el mercado, que operan bajo diferentes principios físicos, la elección del dispositivo correcto depende de las características del líquido con el que se trabaje, de la cantidad de volumen por segundo, o caudal, que se manejé, y de la precisión que se requiere en el sistema.

En la actualidad existen medidores de caudal que no interfieren con el flujo del líquido que miden, denominados no invasivos, y su medición se lleva a cabo a través de ondas de ultrasonido, la precisión con la que miden el caudal es muy buena, y suele ser costosos. Los medidores ultrasónicos son ideales para medir el volumen y caudal de materiales líquidos con características corrosivas e inflamables, ya que pueden no estar en contacto directo con el material que miden, y brindan una precisión en las mediciones muy buena en comparación con otros medidores con principio de operación distinto.

1.2. Objetivo

Aplicar los conocimientos adquiridos durante mi formación como ingeniero eléctrico electrónico en la facultad de ingeniería, y usarlos en la solución de un problema real, a través del diseño y construcción de un sistema electrónico funcional que pueda ser implementado físicamente.

Objetivos específicos:

- Diseñar y construir el prototipo de un sistema de medición electrónico, capaz de controlar y monitorear el flujo de líquidos. El sistema debe medir el flujo de volumen y el caudal de un líquido, y transmitir los datos inalámbricamente hacia una computadora, donde se mostrara la información sobre las variables medidas a través de un panel virtual de control y monitoreo implementado en Labview. El sistema permitirá al usuario controlar el volumen del líquido utilizado, de forma remota, a través de una conexión inalámbrica y con la ayuda del panel virtual de control.

- Implementar un sistema de instrumentación electrónico, digital, con la ayuda de distintos periféricos del micro controlador PIC 16F877, que sea capaz de comunicarse con una computadora, con el fin de crear una Interfaz Hombre Máquina (HMI), que permita una mayor versatilidad en la operación e instalación del sistema de medición y control de flujo de líquidos.

- Desarrollar una aplicación para computadora utilizando el software de instrumentación virtual Labview, que permita remplazar controles e indicadores físicos, por controles e indicadores virtuales ubicados en un tablero de control en la pantalla de una computadora.
- Crear una interfaz electrónica de envío y recepción de datos digitales, entre un micro controlador PIC, y el programa de instrumentación virtual Labview, con el fin de evitar el uso de una tarjeta de adquisición de datos especializada, que elevarían el costo en la implementación del sistema.

- Desarrollar una tarjeta electrónica, necesaria para instrumentar un sensor ultrasónico de flujo, que permita el envío de información de forma inalámbrica hacia un circuito receptor, el cual transferirá los datos hacia la computadora para su procesamiento.
- Diseñar y fabricar los circuitos impresos PCBS que forman parte de las tarjetas electrónicas correspondientes al circuito de instrumentación del sensor de flujo, y al circuito receptor del sistema.

1.3. Descripción del método de solución

Para construir el sistema que permita medir el flujo de líquidos, se utilizó un medidor de flujo ultrasónico UF25B100 de la marca **CYNERGY**. Este dispositivo tiene la capacidad de medir el volumen del líquido que pasa a través de él, y entregar señales eléctricas en forma de pulsos. Entregando un pulso por cada mililitro de líquido que detecta.

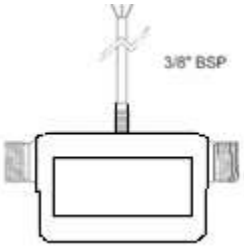


Figura 1.1 Medidor de flujo ultrasónico UF25B100.

El sensor se instrumentó a través de un micro controlador PIC16F877, el cual detecta e interpreta los pulsos, también calcula el caudal. Posteriormente los datos son mostrados en una pantalla LCD, y enviados a un módulo inalámbrico Xbee, quien se encarga de transmitir inalámbricamente los datos hacia un sistema receptor formado por otro Xbee.

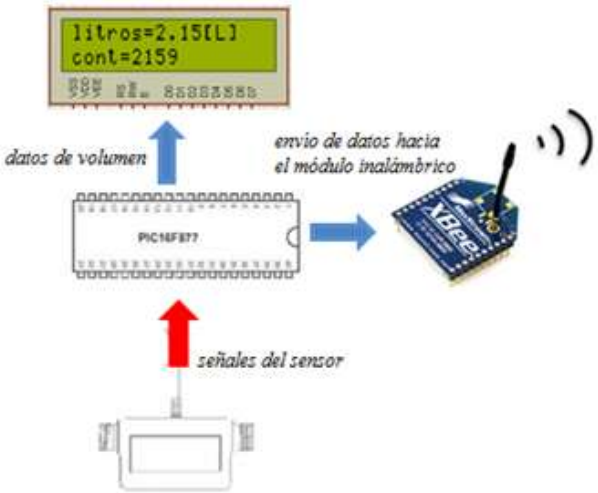


Figura 1.2 Diagrama de instrumentación del sensor de flujo, y módulo inalámbrico Xbee.

El módulo Xbee se encarga de transmitir los datos procesados en el microcontrolador, a través del protocolo de comunicación serial, y sustituye una conexión con cable, por una inalámbrica, de esta forma el sistema será fácil de instalar en cualquier entorno. El Xbee que forma parte del circuito de instrumentación, también es capaz de recibir las señales de

los comandos, que son enviados desde una computadora, donde se ha implementado un panel virtual que permita controlar el sistema.

El módulo receptor se encarga de recibir los datos de volumen y caudal enviados desde el micro controlador, los datos son recibidos empleando otro módulo Xbee, y un driver que permite la comunicación con una computadora laptop en donde se encuentra implementado el panel de control virtual.

Figura 1.3 Recepción de datos, y acondicionamiento de la señal para ser procesada en la computadora.

Las laptop no cuentan con puerto de comunicación serial, por lo que es necesario utilizar un cable de conversión Serial-USB, como se indica en la figura 1.3. Para que este cable funcione requiere de la instalación previa de los drivers apropiados en la computadora, y el circuito integrado MAX233 que sirve para conseguir los niveles de tensión apropiados en el protocolo de comunicación serial utilizados por la computadora.

El panel de control es una aplicación para computadora desarrollada en LabView, la cual permite el monitoreo del volumen y del caudal, además cuenta con controles que permiten detener y reanudar el flujo del líquido que pasa por el sensor. El fluido con el que el sistema trabaja, es agua, por lo que el flujo es controlado con una bomba de agua como actuador del sistema, como se indica en la figura 1.4.

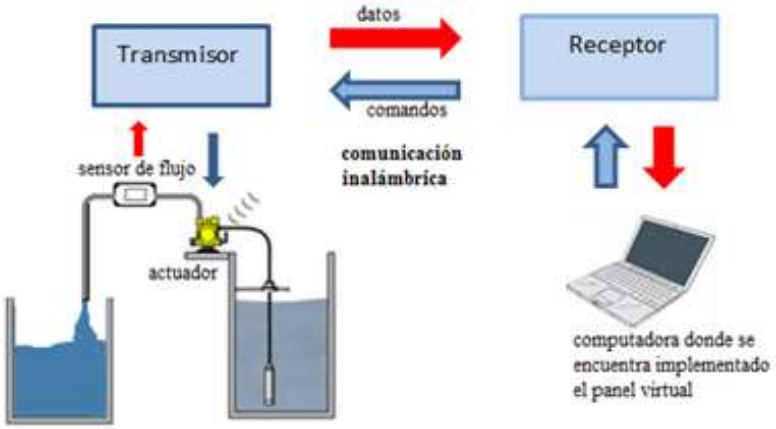


Figura 1.4 Diagrama general del sistema.

El sistema completo permite al operador monitorear el volumen del líquido que pasa de un contenedor a otro, y saber a qué razón lo hace.

Con la ayuda del panel de control virtual el operador puede detener y reanudar el flujo del líquido, mediante un control de encendido y apagado manual. O a través de un modo automático, en donde el sistema interrumpe el flujo cuando detecta que el valor de volumen fijado por el operador ha pasado de un contenedor a otro.

Durante el desarrollo de este prototipo se hizo uso de las siguientes herramientas informáticas: el simulador ISIS PROTEUS, y el compilador C CCS, para la programación del microcontrolador y para la verificación del programa implementado en él, a través de simulaciones. Y el programa de instrumentación virtual LabView, para el desarrollo del panel de control virtual.

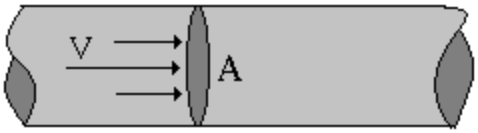
Capítulo 2

Conceptos básicos

2.1 Flujo volumétrico

El flujo volumétrico o caudal es la determinación de las unidades de volumen de un fluido, que se mueven a través de un volumen determinado, por unidad de tiempo. En física el gasto volumétrico o caudal se denota por la letra Q, y sus unidades en el sistema internacional son metro cubico por segundo $[m^3/s]$.

Si se conoce el área transversal "A" en $[m^2]$ del volumen de control que se muestra en la figura 2.1, y la velocidad "V" en $[m/s]$ a la que se mueve el fluido a través del área "A", el producto A·V, determina el caudal "Q" o gasto volumétrico en $[m^3/s]$.



$$Q[m^3/s] = (A[m^2])(V[m/s])$$

Figura 2.1 Flujo volumétrico

En la industria para conocer la razón de desplazamiento de materiales líquidos o gaseosos se utiliza el gasto volumétrico, y cuando se trabaja con sólidos se emplea el gasto másico.

2.2. Tipos de medidores de flujo ultrasónicos

Los medidores de flujo ultrasónicos son dispositivos que utilizan ondas de ultrasonido para determinar la velocidad de flujo de un fluido. Las ondas ultrasónicas tienen frecuencias superiores a los 20 [KHz] aproximadamente, y están sujetas a las mismas leyes básicas de movimiento ondulatorio, pero ofrecen varias ventajas en comparación con las ondas con frecuencias inferiores.

Las ondas de ultrasonido tienen longitudes de onda más cortas, por lo cual presentan una menor difracción al encontrarse frente a obstáculos, y son más energéticas, esto les permite poder atravesar fácilmente y sin problemas las paredes de tubos y recipientes de diversos materiales.

Los medidores ultrasónicos son dispositivos no invasivos, ya que pueden formar parte del circuito como un elemento de acoplamiento de tuberías con el mismo diámetro, o pueden ser montados en el exterior de las tuberías que contienen a los fluidos que son medidos.

Al no estar en contacto directo con los fluidos que miden, los medidores de flujo ultrasónico son ideales para medir fluidos hostiles, o sea aquellos con propiedades corrosivas, radioactivas, explosivas, o inflamables.

Existen dos principios físicos de funcionamiento para los medidores de caudal ultrasónicos, los de efecto Doppler, y los de medición por tiempo de tránsito de ondas ultrasónicas.

Medidor de efecto Doppler:

En los medidores de efecto Doppler un transductor emite una señal ultra sónica de frecuencia conocida, la cual es reflejada por las partículas del fluido que pasa, la señal reflejada es captada por otro transductor receptor, las frecuencias son comparadas, y el cambio de frecuencia entre la señal emitida y la recibida, determina la velocidad del fluido.

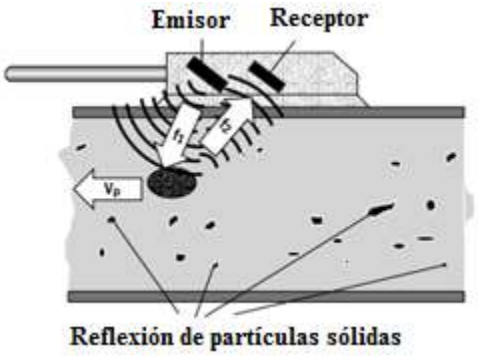


Figura 2.2 Funcionamiento del medidor de flujo por efecto Doppler.

Los medidores de efecto Doppler son poco fiables cuando el fluido tiene solidos suspendidos, ya que estos reflejan las ondas ultrasónicas, causando un error en la medición, ver figura 2.2.

Medidor por tiempo de tránsito de ultrasonidos:

El segundo tipo de medidor de caudal ultrasónico, es el de tiempo de tránsito de ultrasonidos. En estos medidores dos transductores son colocados en posiciones opuestas formando un ángulo entre ellos, los transductores emiten y reciben ondas ultrasónicas. Cuando una onda ultra sónica viaja en contra de la dirección del flujo del líquido tarda más tiempo en llegar al otro transductor, que cuando viaja en el sentido del flujo. Estos tiempos de tránsito pueden ser medidos con precisión, y de este modo se determina la velocidad del fluido, el cual es proporcional al flujo.

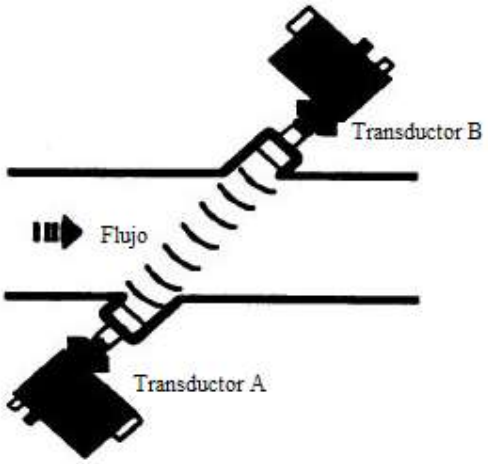


Figura 2.3 Medidor de flujo por tiempo de tránsito de ondas ultrasónicas.

A diferencia del medidor de efecto Doppler que utiliza ondas ultrasónicas continuas, el medidor por tiempo de tránsito en la figura 2.3, emite ondas ultrasónicas en forma de pulsos.

Las principales ventajas de los medidores de flujo ultrasónicos son:

- No interfieren en el flujo, por lo que son, no invasivos.
- Son ideales para medir fluidos hostiles. Como ácidos y combustibles líquidos.
- Proporcionan una alta precisión en las mediciones que realizan.

Una de las desventajas más notables, es que este tipo de dispositivos se utilizan principalmente en fluidos limpios, que no contiene partículas sólidas suspendidas o burbujas de aire, ya que podrían causar la dispersión de las ondas ultrasónicas, y provocar errores en la medición

2.3. Funcionamiento de un medidor de flujo ultrasónico por tiempo de tránsito

El medidor de flujo por tiempo de tránsito utiliza dos transductores, A, y B, colocados en posiciones opuestas en torno a la tubería por donde pasa el líquido. De tal forma que las ondas de sonido que pasan entre ellos forma un ángulo "α", y la dirección de la velocidad del líquido es la que se indica (ver la figura 2.4).

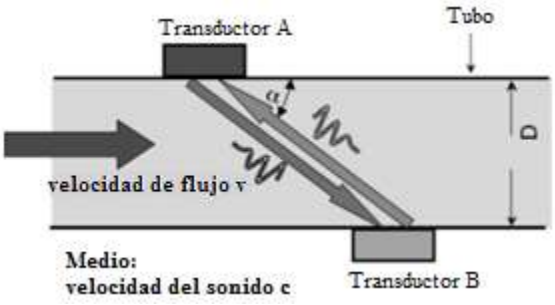


Figura 2.4 Funcionamiento del medidor de flujo por tiempo de tránsito de ondas ultrasónicas.

Si se conoce el diámetro "D" del tubo en donde los dos transductores están montados, se puede medir el tiempo que las ondas ultrasónicas tardan en pasar de un transductor a otro. El tiempo que tarda en pasar una onda ultra sónica, del transductor A, al transductor B, que es en la dirección del flujo, como se indica en la figura 2.4, está determinado por:

$$T_{A\to B} = \frac{D}{\sin\alpha} \cdot \frac{1}{(\mathrm{c} + v \cdot \cos\alpha)}$$

Donde "α" es el ángulo que forman los transductores, "D" es el diámetro del tubo, y "c" es la velocidad del sonido en el líquido.

Cuando la onda ultra sónica se mueve en dirección opuesta al flujo, el tiempo que emplea para pasar del transductor B, al transductor A, es determinado por:

$$T_{B\to A} = \frac{D}{\sin\alpha} \cdot \frac{1}{(\mathrm{c} - v \cdot \cos\alpha)}$$

Al final la diferencia de tiempos de tránsito se determina con:

$$T_{B\to A} - T_{A\to B} = v \cdot \frac{T_{B\to A} \cdot T_{A\to B} \cdot \sin(2\alpha)}{D}$$

Y se obtiene la velocidad mediante:

$$v = \frac{D}{\sin(2\alpha)} \cdot \frac{T_{B\to A} - T_{A\to B}}{T_{B\to A} \cdot T_{A\to B}}$$

Una vez conocida la velocidad del líquido, el caudal o flujo, se determina mediante:

$$q = v \cdot A = \pi \cdot \frac{D^2}{4}$$

De forma general; Cuando una onda ultra sónica se mueve en dirección del flujo del líquido, esta lo hará a la velocidad del sonido en el líquido, más una contribución debida a la velocidad de flujo. Por el contrario, si se mueve en la dirección opuesta, la velocidad a la que lo hará, será una velocidad debida a la velocidad del sonido en el líquido, menos la velocidad del flujo. Al final la diferencia entre estos dos valores se determina a través de medios electrónicos, y representa la velocidad del fluido, que es proporcional al flujo.

El objetivo de los medidores de flujo por tiempo de tránsito, es medir el caudal a través de la cuantificación de la velocidad de flujo del líquido. Por lo que se trata de un medidor de caudal indirecto.

2.4. Microcontrolador PIC 16f877

El PIC 16F877, es un microcontrolador fabricado por **Microchip,** pertenece a una subfamilia denominada de gama media, estos dispositivos se caracterizan por emplear 35 instrucciones de 14 bits en su bus de instrucciones. Su bus de datos es de 8 bits, cuenta con un diseño de CPU tipo RISC, la arquitectura que tiene es del tipo Harvard, y la tecnología que emplea es CMOS.

Este microcontrolador cuenta con gran diversidad de periféricos, es fácil de programar y es uno de los más usados en el desarrollo de prototipos.

El microcontrolador PIC 16F877, es un dispositivo que opera con 5[Vdc], tiene un encapsulado tipo DIP de 40 pines, que ofrece cinco puertos de entrada y salida digitales, distribuidos en 33 pines. El puerto A con 6 bits, mientras que los puertos B, C, y D, tienen 8 bits, y por último el puerto E, solo cuenta con 3 bit, esto nos brinda una gran cantidad de entradas y salidas digitales. Pero los pines de este microcontrolador no solo se limitan a realizar funciones de entrada y salidas digitales, estos también pueden realizar otras funciones, siempre que se les programe para ello.

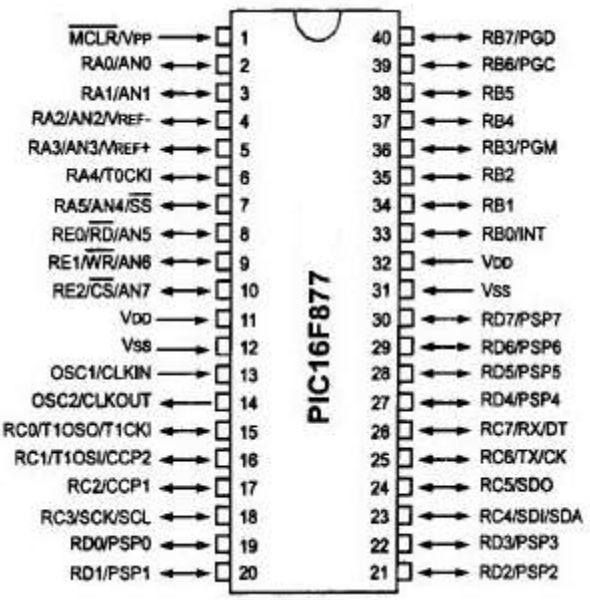


Figura 2.5 Microcontrolador PIC16F877
y su distribución de pines.

Es importante resaltar que el consumo de energía del dispositivo está en función de la velocidad a la que trabaja, el voltaje de operación, y sobre todo de la carga conectada a sus terminales.

Cuando los terminales del dispositivo se programan como entradas digitales (o analógicas en el caso de entrada al convertidor AD), se dice que se encuentran en modo sumidero, por el contrario, si son salidas, se denomina modo fuente. En ambos casos las líneas de puerto tienen una tolerancia límite de corriente de entrada o salida que puede pasar a través de ellas.

Para cada línea de puerto (pin), la tolerancia máxima de corriente es de 25 $[mA]$, y para cada puerto se tiene una tolerancia máxima que se muestra en la siguiente tabla.

Modo	Puerto A	Puerto B	Puerto C	Puerto D
Sumidero	150 $[mA]$	2000 $[mA]$	2000 $[mA]$	2000 $[mA]$
Fuente	150 $[mA]$	2000 $[mA]$	2000 $[mA]$	2000 $[mA]$

Tabla 1.1 Capacidad máxima de corriente que
soportan los puertos del PIC.

El PIC 16F877 cuenta con varios periféricos que lo hacen poderoso y versátil, a continuación se describen alguno de ellos, sobre todo los que se emplean en el desarrollo del prototipo.

Interrupción por evento externo RB0:

Es un periférico que la mayoría de los microcontroladores PIC poseen, y se utiliza para detectar cambios en los niveles de tensión en el pin 0 del puerto B. Cuando se detecta un cambio en el nivel de tensión del pin RB0, la ejecución del programa principal del microcontrolador se interrumpe, y se ejecuta el segmento de código que define a la interrupción. Una vez se ha terminado de ejecutar el código para la interrupción, el microcontrolador vuelve a ejecutar el código contenido en la función principal del programa.

El flanco para la detección del cambio de nivel de tensión puede ser de subida, o de bajada, la elección se define durante la configuración de la interrupción en el programa. Este periférico se utilizó para detectar los pulsos que el sensor de flujo emite, cuando por él pasa un mililitro de líquido.

Interrupción del TIMER0:

Los *timers* o temporizadores, son módulos que posee el microcontrolador, con los cuales se incrementar un contador localizado en un registro especial, y de esta forma puede contar eventos externos o internos a él.

Cuando éste módulo cuenta eventos externos hablamos de un contador, y si cuenta eventos internos como pulsos del reloj, se denomina temporizador. En ambos casos la frecuencia del ciclo de conteo es: la frecuencia de la fuente de oscilación dividida entre cuatro.

El TIMER0 es un contador (registro) de 8 bits que se auto incrementa en cada ciclo de conteo, tiene la capacidad de interrumpir la ejecución del programa principal cuando ocurre un desbordamiento del registro, con una capacidad máxima de conteo de 255 ciclos.

El desbordamiento en el registro del TIMER0 puede ser programado para lograr intervalos de tiempos determinados, con la ayuda de *prescaler* que divide la frecuencia del ciclo de conteo en 2,4,8,16,32,64,128 o 256, y calculando el valor con el que el registro debe ser cargado.

La interrupción del TIMER0 es utilizada en el prototipo para generar un intervalo de tiempo determinado, que sirve para determinar la relación "*litros / segundo",* obteniendo el caudal en el sistema.

Módulo USART (Transmisor-Receptor Serie Síncrono-Asíncrono Universal):

El módulo USART, es un periférico con el que cuenta el microcontrolador PIC 16F877, el cual permite la comunicación con otro dispositivo a través del protocolo de comunicación serial.

Este periférico es quizás el más importante en el desarrollo del prototipo, ya que es usado para transmitir los datos de volumen y caudal procesados en el microcontrolador, a través del protocolo de comunicación serial, hacia una computadora donde son desplegados en los indicadores correspondientes.

Interrupción por recepción de datos RDA:

Es un periférico que el microcontrolador utiliza cuando establece una comunicación a través del protocolo de comunicación serial con otro dispositivo. Este permite interrumpir la ejecución del programa principal, y ejecutar una sub rutina cada vez que un dato enviado desde un dispositivo externo, llega al módulo USART del micro controlado.

En el prototipo la interrupción RDA es utilizada para decodificar los comandos enviados desde el panel de control implementado en una computadora, y que contiene las instrucciones que el microcontrolador ejecutará.

Capítulo 3

Instrumentación del sensor de flujo

3.1. Sensor de flujo UF25B100, y su acondicionamiento

El sensor UF25B100 de la marca **CYNERGY,** es un medidor de flujo ultrasónico que opera bajo el principio de: *medición de tiempo de tránsito de ondas ultrasónicas*.

Su diseño proporciona una alta precisión, ya que es un medidor no invasivo, es decir, no cuenta con elementos mecánicos que interfieran con el flujo, y la trayectoria que sigue el fluido dentro del dispositivo minimiza las pérdidas de presión.

También puede permitir que los contaminantes solidos pasen a través de él, sin afectar su rendimiento. Cuenta además con compensación automática de viscosidad, y temperatura, lo cual le permite trabajar con diferentes líquidos.

Las señales de salida del sensor pueden ser digitales (por pulsos), o analógicas (por voltaje, y corriente).

Figura 3.1 Sensor de flujo UF25B100.

Sus características técnicas son:

Flujo máximo	25 [L/min]
Flujo mínimo	0.2 [L/min]
Tensión de alimentación	8-24 [Vdc]
Presión máxima	10 [bar]
Medición por pulsos	1000 pulsos/Litro

Tabla 1.2 Características técnicas del sensor de flujo ultrasónico UF25B100.

Conexiones eléctricas:

Cable	Descripción
Rojo	Alimentación 8-24 [Vdc]
Negro	Tierra común (GND)
Verde	Tierra (conectado internamente con negro)
Azul	Salida por pulsos NPN
Blanco	Salida por pulsos PNP
Anaranjado	Salida por voltaje de 0-5 [Vdc]
Café	Salida por corriente 4-20 [mA]
Amarillo	"**este cable no debe conectarse**"

Tabla 1.3 Conexiones eléctricas del sensor de flujo ultrasónico UF25B100.

Para el acondicionamiento de la señal del sensor, se utiliza el cable "*azul*" de salida de pulsos a colector abierto NPN. Para que este cable sea capaz de transmitir los pulsos enviados por el sensor, es necesario conectar una resistencia de *pull-up* que complete el circuito como se indica en la figura 3.2.

Figura 3.2 Configuración colector abierto.

La topología de colector abierto permite la conexión entre dispositivos que operan a diferentes niveles de tensión. Esto es importante ya que para la instrumentación del sensor se utiliza un microcontrolador PIC, que tiene un nivel de tensión de operación de 5[Vdc]. Mientras que el sensor de flujo UF25B100, debe ser alimentado con niveles de tensión de 8-24[Vdc].

Para que la conexión entre el sensor y el microcontrolador, sea correcta, se emplea una resistencia de *pull-up*, conectada a la tensión de alimentación del microcontrolador, de 5 [Vdc], y a la terminal de salida de pulsos del sensor (cable azul). El sensor es alimentado con un nivel de tensión de 9 [Vdc], como se indica en la figura 3.3.

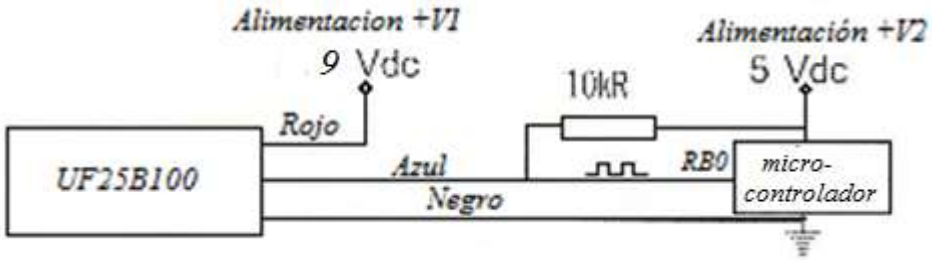


Figura 3.3 Conexión entre el sensor, y el microcontrolador usando diferentes niveles de tensión.

Una vez que el circuito a colector abierto, en la salida del sensor, ha sido completado como se indica en la figura 3.2, la terminal de salida (cable azul) debe ser conectada en el pin RB0 (pin 33) del microcontrolador como se indica en la figura 3.3. Esta conexión sencilla entre el sensor de flujo, y el microcontrolador, permite detectar los pulsos, utilizando la interrupción por evento externo RB0 del microcontrolador PIC.

3.2. Programa para medir volumen

Después de haber acondicionado la conexión entre el sensor de flujo, y el microcontrolador, es necesario implementar un programa que permite que el PIC, detecte, interprete, y muestre, la razón de volumen que pasa por el sensor.

En este programa se utiliza la interrupción por evento externo RB0, para implementar un contador que se incrementa a medida que recibe pulsos eléctricos del sensor de caudal ultrasónico.

Dado que el sensor ultrasónico está diseñado para entregar un pulso cuando un mililitro de líquido para a través de él, se puede encontrar la razón de volumen si se cuenta cuantos pulsos se han producido, y posteriormente se expresan en unidades correspondientes de volumen.

Para realizar esta tarea se emplea una simulación del dispositivo mediante ISIS PROTEUS, y el compilador CCS.

El primer paso es implementar la interrupción RB0, a través de la programación del microcontrolador PIC. Para ello es necesario crear un nuevo archivo fuente en el compilador CCS, y escribir los siguientes segmentos de código, que son explicados brevemente, a continuación.

Las directivas de cabecera que se utilizan son las siguientes:

```
#include <16f877.h>
#fuses HS,NOWDT
#use delay(clock=20M)
#use fast_io(B)    //configuración rápida del puerto B
#include <lcd.c>  // para usar el modulo LCD
```

En la cabecera del programa se define la directiva que permite la configuración rápida del puerto B, en el que se encuentra el pin RB0, y se incluye el archivo "*lcd.c"*, el cual es una aplicación que permite el uso de funciones para manejar el módulo LCD.

Después de declarar las directivas de cabecera, se define una variable de tipo entero sin signo, de 16 bit de longitud, ésta variable será capaz de representar números desde 0 hasta 65535, a la cual se ha llamado "*contador*", la variable debe ser definida antes de declarar la interrupción.

Posteriormente se declara la interrupción con el siguiente formato: Se coloca la directiva #INT_EXT, y en el siguiente renglón se coloca el nombre de la interrupción, terminando con paréntesis. Por último se coloca entre llaves el código de programa a ejecutar durante la interrupción.

```
int16 cont=0;                //variable de16 bits,  de 0 a 65545
#INT_EXT                          //directiva para interrupción RB0
Void interrup_ex()           // interrupción
  {
cont=cont+1;             //  "rutina de interrupción
  }                          // "contador" se incrementa
```

Una vez definida la interrupción, es necesario establecer en la función principal "*main()*", si el flanco que activa la interrupción será de bajada (cambio de nivel alto a bajo), o de subida (cambio de nivel bajo a alto). Para el propósito que se requiere es indistinto, se puede elegir cualquiera, en este caso se eligió que la interrupción sea por flanco de subida.

ext_int_edge(L_TO_H);
Establece la interrupción por flanco de subida.

Para que el código de interrupción funcione, es necesario habilitar las interrupciones globales y la interrupción particular de evento externo RB0, mediante las instrucciones que se muestran.

```
void main(){                    // función principal
set_tris_B(0x01);               // se define pin RB0 como entrada
port_B_pullups(TRUE);           // se habilita resistencia pullup en pin RB0
ext_int_edge(L_TO_H);           //se define flanco de subida
enable_interrupts(GLOBAL); // habilita interrupciones globales
enable _interupts(INT_EXT);  //habilita interrupción particular RB0
```

El ciclo infinito "*while()*", se encarga de ejecutar el programa principal de forma permanente, por eso dentro de esta función se realizará la conversión del número de pulsos contados por la interrupción RB0, y se transforman a su equivalente en litros. Posteriormente el resultado se muestra en un *display LCD*.

Es necesario inicializar el módulo LCD, y declarar una variable tipo flotante en donde se guardará el resultado de la conversión de pulsos leídos, a su equivalente en litros, antes de entrar al ciclo infinito "*while()*".

```
Lcd_init();                          //función que inicializa el display LCD
float L=0;                           //declaración de variable flotante L
while(true){                         // ciclo infinito
L=cont/1000.0;                 //se realiza la equivalencia de mililitros a litros
printf(lcd_putc,"Litros=%f [L] ",L);       //se imprime el resultado en el LCD
printf(lcd_putc, " \n cont=%LU",cont);  // se pasa a la línea siguiente y se
                                              //  muestra el valor de "cont"
            }  //fin de while
}              //fin de main()
```

La variable "*cont*" almacena el número de pulsos (mililitros) leídos en el sensor, por lo que para convertirla a su equivalente en litros, basta con dividirla entre 1000.

Para verificar el buen funcionamiento del programa, se utiliza una simulación mediante ISIS PROTEUS, con el fin de detectar errores, y corregirlos, antes de la implementación real del código en el microcontrolador.

Para realizar la simulación del funcionamiento del programa con ISIS PROTEUS, se alambra el siguiente circuito.

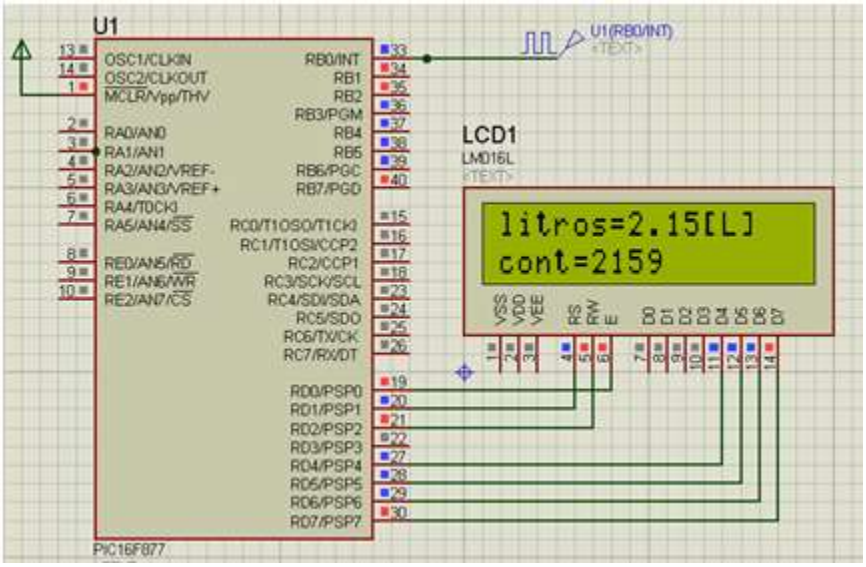


Figura 3.4 Simulación del programa para medir volumen, mediante ISIS PROTEUS.

En el diagrama del circuito, se utilizan los componentes de *ISIS PROTEUS*: PIC16F877 y LM016L. La señal de entrada del sensor de flujo, es simulada mediante un generador de señales de reloj "*DCLOCK*", como se aprecia en la figura 3.4.

En la simulación se observa como el programa implementado en el microcontrolador se encarga de contar los pulsos de la señal que es mandada por el sensor de flujo (simulado mediante DCLOCK). Mostrando el contador, y su equivalente en litros, en el módulo LCD.

3.3. Programa para medir flujo

Como parte del prototipo se incorpora un medidor de caudal, con el fin de poder monitorear la velocidad a la que se mueve el volumen del líquido dentro del sensor, de esta forma el operador del sistema podrá conocer éste parámetro para tomar decisiones, y aplicar alguna medida de control. Para desarrollar este medidor se utiliza la interrupción por desbordamiento del TIMER0 del microcontrolador PIC16F877.

Dado que el sensor nos entrega un pulso por cada mililitro de líquidos que pasa por él, podemos medir cuantos mililitros han pasado en cierta unidad de tiempo. Con la ayuda del TIMER0, se genera una base de tiempo determinada, y se compara contra el número de pulsos (mililitros) que han pasado. De este modo se obtiene la relación "*volumen/tiempo"***,** o caudal.

Para el desarrollo de este programa se utilizará el código del programa anterior, visto en el subcapítulo 3.2, que determina el volumen que pasa por el sensor, a través de la interrupción externa RB0.

Las directivas de cabecera que se utilizan, se muestran en el siguiente segmento de código, y son las mismas empleadas en el programa anterior.

```
#include <16f877.h>
#fuses HS,NOWDT
#use delay(clock=20M)
#use fast_io(B)    //configuración rápida del puerto B
#include <lcd.c>  // para usar el modulo LCD
```

Después de haber declarado las directivas de cabecera correspondientes, se declara la interrupción por evento externo RB0, que registra mediante un contador, los mililitros que pasan por el sensor. En la interrupción se define una nueva variable llamada "*R*", la cual se incrementa junto con el contador "*cont*", y que será usada para calcular el caudal en la interrupción del TIMER0.

```
int16 cont=0,R=0;       //variables
#INT_EXT                //directiva de interrupción externa RB0
void ext_isr(){     //interrupción
cont=cont+1;     // registra número de pulsos
R=R+1;              //registra número de pulsos
}                   //fin de interrupción RB0
```

Con la interrupción del TIMER0, se generan intervalos de tiempo de 10[ms], en los cuales se ejecuta la interrupción de desborde del *timer*. El intervalo de tiempo incrementa un contador dentro de la rutina de interrupción, que al llegar a 50 veces, permite el cálculo de caudal. Obteniendo el valor de caudal cada 500[ms].

Segmento de código para la interrupción de desbordamiento del TIMER0:

```
int j=0;
float v;                    // variable para guardar el dato de caudal
#int_TIMER0                 //directiva para interrupción del TIMER0
void timer0_isr(void){   //interrupción por desborde de TIMER0
j=j+1;                      //contador para intervalo de 500 [ms]
if(j==50){                  //crea nuevo intervalo de tiempo de 500[ms]
v=R*0.12;                   // cálculo de caudal en L/minuto
j=0;                        //reinicio de contador.
R=0;              //reinicio de la cuenta de volumen
}
set_timer0(0x3d);        //carga del registro del TIMER0
}
```

En el segmento de código anterior, el valor del caudal, es guardado en la variable "v", y se calcula multiplicando el valor del contador "*R*", en la interrupción RB0, por el factor "0.12", que convierte al caudal en razón de *mililitros/milisegundos,* a *litros/minuto*.

"*R*" es número de mililitros contados, y el intervalo de tiempo en el que se cuentan es de 500[ms]. Por lo que se obtiene la relación de:

$$\frac{R[mililitros]}{500[mili\ segundos\ ]}$$

Para encontrar el facto de proporción, se aplican las siguientes relaciones:

$$\frac{R[ml]}{500[ms]} = \frac{R[ml]}{0.5[s]}\left(\frac{1[L]}{1000[ml]}\right)\left(\frac{60[s]}{1[minuto]}\right)$$

$$= R\left(\frac{(1)(60)}{(0.5)(1000)(1)}\right)\left[\frac{L}{minuto}\right]$$

$$= R(0.12)\left[\frac{L}{minuto}\right]$$

Donde, al multiplicarse el factor "0.12" y el contador de mililitros "*R*" se obtiene el valor de caudal, en razón de *litros/minuto*.

El valor con el que se carga el registro del *timer*, para obtener una interrupción cada 10[ms], es de 61, o 0x3D en hexadecimal, y se calcula usando un *prescaler* de 256, una frecuencia de oscilación de 20[MHz], y la siguiente relación:

$$T = T_{Cm} \cdot P(256 - C_{tm0})$$

Dónde:

T= Tiempo de desborde. (10[ms])
P=Prescaler. (256)
C_{tm0} =Valor de carga del TIMER0.
T_{Cm}= Ciclo máquina que se puede calcular como $T_{Cm} = \frac{4}{F_{Osc}}$

Después de haber definido las interrupciones a utilizar, estas son configuradas y habilitadas con las instrucciones correspondientes dentro de la función principal "*main()*", como se muestra a continuación.

```
void main(){                    //función principal

set_tris_B(0x01);          //pin 0 del puerto B es entrada
port_b_pullups(TRUE);   //se habilita resistencia de pull-up en el pin de entrada

enable_interrupts(GLOBAL);     //habilita interrupciones globales

enable_interrupts(int_ext);   //habilita interrupción externa RB0
ext_int_edge(L_TO_h);      //la interrupción es por flanco de subida

setup_TIMER_0(RTCC_INTERNAL|RTCC_DIV_256); //configuración del TIMER0
set_timer0(0x3d);                          //carga del registro del TIMER0
enable_interrupts(INT_TIMER0);        //habilita interrupción del TIMER0

lcd_init();       //inicializa el modulo LCD.
```

Antes de entrar al bucle infinito *"while()"*, se declaran las variables a usar, y se inicializa el módulo LCD, en donde serán mostrados los datos de volumen, y caudal.

```
lcd_init();      //inicializa el modulo LCD
float L=0;      //variable para volumen
while(true){          //bucle infinito
L=cont/1000.0;      //se convierte el volumen a litros
lcd_gotoxy(1,1);       //posiciona al cursor
printf(lcd_putc,"Vol=%0.2f[L]",L);      //muestra el volumen
printf(lcd_putc,"\n\rCau=%0.1f[L/min]",v);    //muestra el caudal
      }    //fin de while
}        //fin de main
```

Con el segmento de código anterior, se muestra en el LCD, el volumen leído del sensor transformado a litros, y el caudal en *litros/minuto.*

Para verificar el funcionamiento correcto del programa desarrollado, se utiliza una simulación en ISIS PROTEUS y el compilador CCS, como se muestra en la siguiente figura.

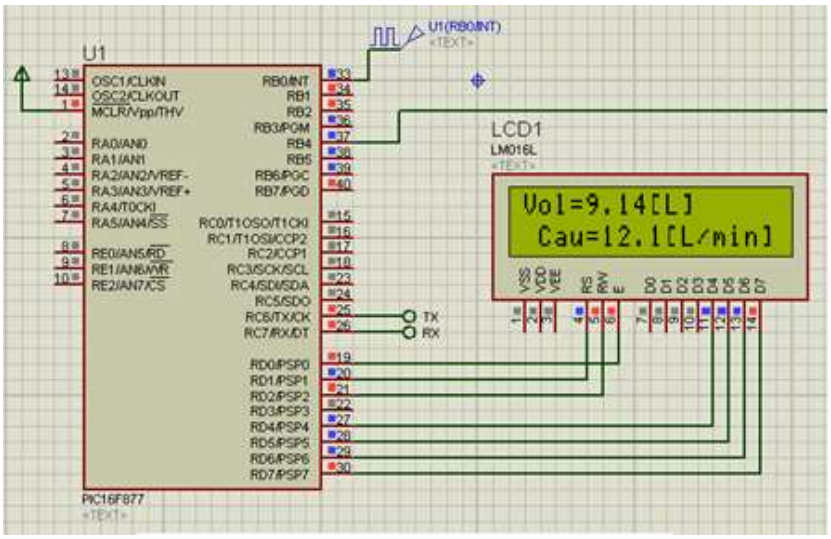


Figura 3.5 Simulación del programa para medir volumen y caudal, mediante ISIS PROTEUS.

En la figura 3.5, se muestra el funcionamiento del código implementado en el microcontrolador, el cual muestra el volumen que pasa por el sensor, y el caudal. La señal de salida del sensor es simulada mediante un generador de señales de reloj en ISIS *PROTEUS.*

Capítulo 4

Comunicación inalámbrica

4.1. El módulo de comunicación inalámbrica Xbee

El módulo Xbee es un dispositivo electrónico que se encarga de realizar la conexión inalámbrica entre dos o más dispositivos similares. Estos módulos sirven para transmitir datos de un punto a otro sin necesidad de emplear cables físicos, ya que su comunicación se lleva a cabo en radio frecuencia.

Los módulos inalámbricos Xbee utilizan un protocolo de comunicación denominado *Zegbee***.** Dicho protocolo fue creado por un grupo de grandes empresas para lograr comunicar redes de sensores en entornos industriales, médicos, y sobre todo domóticos.

El alcance de los módulos depende de la potencia de transmisión que manejen, ya que existen distintos modelos de Xbee, como los de la serie uno, serie dos, y la serie Xbee pro, esta última se caracteriza por lograr un mayor alcance de comunicación.

La utilidad de los módulos Xbee es muy amplia y variada, son utilizados para formar amplias redes de dispositivos, capaces de transmitir datos hacia un solo dispositivo denominado coordinador, quien se encarga de administrar el tráfico de datos. O para sustituir un cable de comunicación entre dos dispositivos, por una conexión inalámbrica.

Los módulos pueden ser utilizados con cualquier microcontrolador que utilicé comunicación serial, solo se tiene que realizar una serie de configuraciones antes de poder emplearlos. La configuración de los módulos Xbee se lleva a cabo a través de un software proporcionado por el fabricante de forma gratuita, llamado **X-CTU**, o a través de un emulador de terminal serial como *Hiperterminal* de Windows, mediante comandos **AT.**

Dentro de las ventajas más apreciables que ofrecen los módulos Xbee, se encuentran: que son módulos de bajo consumo de energía, son capaces de operar en un modo de Ultra bajo consumo, utilizan bandas de radio libres que no requieren licencias, su instalación e implementación es simple, y tienen la capacidad de crear redes flexibles y extensibles.

Los Xbee son módulos micro procesados, con lo que se ha logrado solucionar el problema de fallo de trama, y ruidos.

4.2. Adaptador de niveles de tensión del Xbee

El módulo Xbee requiere de una fuente de alimentación de 3.3 [V], y emplea niveles lógicos de tensión entre, 0[V] y 3.3 [V] en la transmisión y recepción de datos. Por este motivo se requiere de un circuito regulador de tensión, que garantice el nivel de voltaje nominal del módulo. Existen circuitos integrados diseñados para cumplir con tal propósito, como el LM3940 de Texas Instruments. La figura 4.1 muestra el circuito básico que se requiere para que el Xbee funcione, una vez que ha sido configurado correctamente.

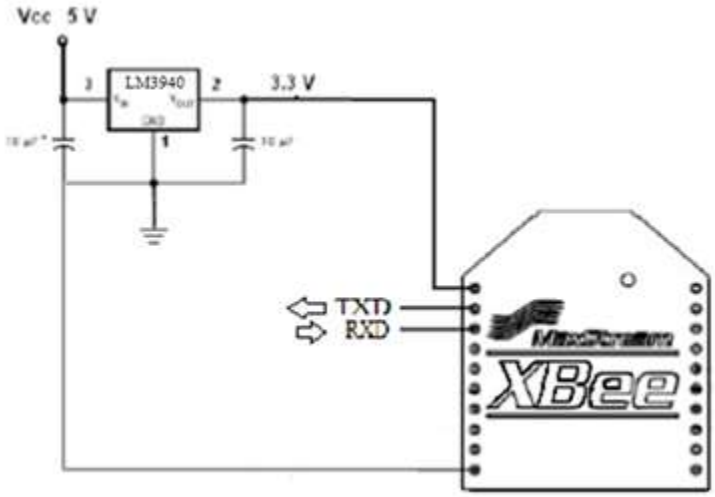


Figura 4.1, circuito básico de alimentación del módulo Xbee.

Los niveles de tensión en el pin de transmisión de datos TXD (pin 2) del módulo Xbee, son de: 0[V] para el cero lógico, y 3.3 [V] para el uno lógico. Estos niveles de tensión deben coincidir con los del pin de recepción de datos RXD (pin 3) para no dañar el módulo.

Es indispensable construir una interfaz electrónica que permita la configuración del módulo Xbee a través de una computadora, utilizando el protocolo de comunicación serial. Dicha interfaz está compuesta por un circuito integrado para adaptar los niveles de tensión que los dispositivos usan en el protocolo de comunicación serial, y un cable convertidor Serial-USB.

El circuito integrado MAX233, es un dispositivo empleado para adaptar los niveles de tensión entre dispositivos que manejan datos en niveles TTL, y los niveles lógicos de tensión que maneja una computadora en el protocolo de comunicación serial, que son de +15[V] a -15[V].

Para utilizar el circuito integrado, se debe polarizar con 5 [Vdc], de esta forma se establecen los niveles de tensión TTL en su pin de recepción de datos T1IN (pin 2), y en el pin de transmisión de datos R1OUT (pin 3). Es decir 0[V] para el cero lógico, y 5[V] para el uno lógico, como se indica en la figura 4.2.

Se tiene que tener en cuenta que el envío de datos a través del pin R1OUT (pin 3) del integrado, tendrá un nivel máximo de tensión de 5[V], que no puede aplicarse directamente al pin RXD de recepción de datos del módulo Xbee, ya que éste opera con 3.3 [V] y podría dañarse, para evitarlo se construye un adaptador de nivel de tensión mediante un divisor de voltaje, empleado tres resistencias de $10[K\Omega]$ conectadas en serie, como se muestra en la figura 4.2.

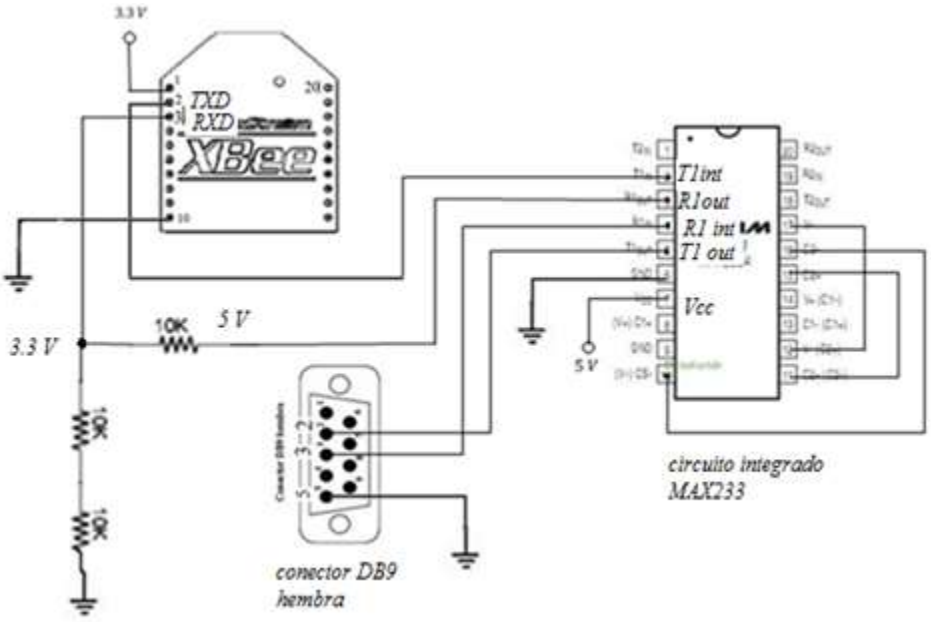


Figura 4.2 Circuito adaptador de niveles de tensión, para el protocolo de comunicación serial.

De esta forma el voltaje en el pin 3 del Xbee siempre será de 2/3 el voltaje máximo (uno lógico =5[V]) del pin 3 del integrado MAX233, y sé tendrá que 5(2/3)=3.3 [V], garantizando un nivel adecuado para la entrada de datos del módulo Xbee.

El Xbee envía datos con niveles lógicos de 0 [V] a 3.3 [V], que el integrado MAX233 no tiene problemas en identificar, ya que 3.3 [V] está dentro del rango del uno lógico que puede decodificar. Por esta razón se conectan directamente las terminales TXD (pin2) del Xbee, y T1IN (pin 2) del integrado MAX233, como se muestra en la figura 4.2.

Para realiza la conexión del MAX233 con la PC, se conecta la terminal T1OUT (pin 5) del integrado, con la terminal 2 de un conector DB9 hembra, y R1INT (pin 4) del integrado con la terminal 3 del DB9. La terminal 5 del conector DB9 debe ser conectada a la tierra del circuito adaptador de niveles de tensión, como se indica en la figura 4.2.

Finalmente se completa la interfaz de conexión entre el módulo Xbee y la PC, mediante un conector DB9 macho, o si se emplea una computadora portátil se requiere de un cable de conversión de protocolo de comunicación Serial a USB. Dicho cable necesita la instalación de un controlador en la computadora para que pueda funcionar, el fabricante lo proporciona, o se puede conseguir en internet.

NOTA

Ésta interfaz tiene dos funciones dentro de la construcción del prototipo:

1. Sirve para poder configurar los dos módulos inalámbricos que se emplean a través de una computadora, mediante el protocolo de comunicación Serial, y una serie de comandos denominados **AT**.
2. Es el mismo circuito que se emplea como receptor, un vez que los módulos Xbee han sido configurados correctamente.

4.3. Configuración del módulo Xbee

Los módulos Xbee emplean dos modos de comunicación, uno es conocido como modo transparente, y se utiliza principalmente para sustituir la conexión a través de cables, por una conexión inalámbrica. El segundo es denominado modo **API**, que se utiliza para formar redes de dispositivos, y es más complejo.

El modo de comunicación que utilizan los módulos Xbee empleados en el prototipo es el modo trasparente, con una topología conocida como *punto a punto*. Esta configuración permite remplazar un cable de comunicación serial, por una conexión inalámbrica entre los dos dispositivos.

Para llevar a cabo la configuración requerida, es necesario utilizar el circuito adaptador de niveles de tensión en el módulo Xbee, que se vio en el sub capítulo 4.2, correspondiente al tema.

La configuración de los módulos Xbee se realiza conectando la interfaz compuesta por el circuito integrado MAX233, y el cable de conversión Serial-USB, al puerto de la computadora.

Una vez identificado el dispositivo Serial-USB en la computadora, se configura una nueva conexión entre el dispositivo, e *Hiperterminal*, y se ingresa al modo AT del módulo Xbee.

El modo de comandos AT permite configurar el módulo Xbee para que pueda operar de la forma requerida. Para ingresar al modo AT se escriben los caracteres +++ sin espacios en la pantalla de *Hiperterminal*. Si la conexión es correcta, el módulo responderá enviando la palabra "OK", indicando que está listo para recibir los comandos AT de configuración.

Si después de haber ingresado los caracteres +++ de acceso a modo de comandos AT, el módulo no muestra respuesta alguna, esto puede deberse a una diferencia en las velocidades de *baudaje* (bits por segundo) entre el módulo Xbee, e *Hiperterminal*.

Para solucionar el problema es necesario asegurarse que las velocidades de *baudaje* entre el módulo Xbee, e *Hiperterminal* son las mismas. El módulo viene por Defecto configurado con una velocidad de 9600[bps].

Los comandos AT permiten la configuración de los parámetros del módulo Xbee, como velocidad de baudaje, especificar una dirección para la red de área personal, elegir un canal de comunicación, etc.

La sintaxis que deben tener los comandos AT, para que logren modificar algún parámetro del módulo es la siguiente:

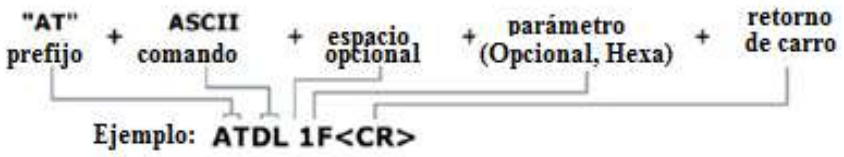


Figura 4.3 Sintaxis de los comandos AT.

Se escribe el prefijo "AT" en primer lugar, para indicar al módulo que los caracteres siguientes pertenecen a un comando de configuración ASCII, se agrega el parámetro, y finalmente se ejecuta el comando, enviar un carácter "**CR**" (Retorno de carro) oprimiendo la tecla "**enter**".

El módulo responderá con la palabra "OK" si el comando y el parámetro son válidos, en caso contrario entregará "ERROR".

Si no se ingresa un comando AT valido durante un periodo determinado, el módulo Xbee saldrá automáticamente del modo de comandos AT.

A continuación se muestra un breve ejemplo sobre la sintaxis de los comandos AT. La siguiente imagen es un ejemplo de lo que se aprecia en la pantalla de *Hiperterminal* durante la configuración del módulo Xbee.

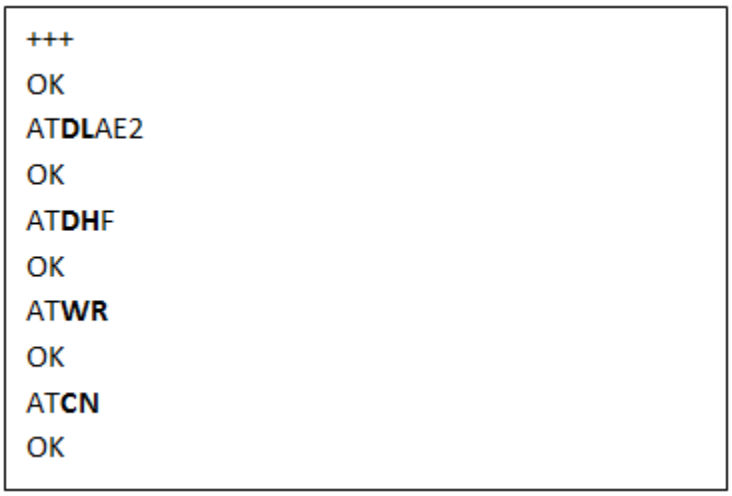


Figura 4.4 Ejemplo del uso de los comandos AT.

Se escriben los caracteres +++ de acceso al modo de comandos AT, el módulo confirma el acceso con un "OK" y con una nueva línea, en la que se escribe el prefijo "AT" seguido del comando DL, con parámetro AE2. El módulo responde con un "OK", que confirma que el comando, y parámetro ingresados son válidos. Del mismo modo se ingresa el comando DH, con el parámetro F, y el módulo responde con "OK". Los comandos WR y CN son utilizados para escribir las modificaciones, y para salir del modo AT respectivamente, y no requieren parámetros, pero el módulo confirma su validez respondiendo con "OK".

A continuación se muestra una breve lista de los comandos AT que se utilizan en la configuración de los módulos Xbee que forman parte del prototipo:

Comando AT	Descripción
CN	Sale de modo de comandos AT.
CH	Permite seleccionar canal de comunicación. Defecto=0x0C
DL	Ajusta los 32 bits menos significativos del direccionamiento.
DH	Ajusta los 32 bits más significativos del direccionamiento.
ID	Ajusta la dirección PAN del módulo. Defecto=0x3332.
RE	Restaura los valores de los parámetros a los valores por defecto que vienen de fábrica.
SL	Entrega los 32 bits menos significativos del número de serie.
SH	Entrega los 32 bits más significativos del número de serie.
WR	Escribe los valores de los parámetros.

Tabla1.4 Lista de comandos AT utilizados durante la configuración de los módulos Xbee del prototipo.

4.4. Transmisor

Para el desarrollo del prototipo es necesario construir un circuito eléctrico que actué como transmisor de datos inalámbrico, el cual permitirá poder enviar los datos de volumen y caudal del sensor de flujo, desde el microcontrolador, hacia otro circuito que funcionará como receptor. Para lo cual se emplean dos módulos Xbee básicos 802.15.4 (Serie1).

No debe confundirse los módulos inalámbricos Xbee Serie 1, con los Xbee serie 2, ya que no son compatibles, y es imposible lograr una comunicación entre ellos, por lo que es indispensable contar con dos módulos similares.

Cada dispositivo tiene una dirección pre establecida por el fabricante de 64 bits (dividida en dos registros de 32 bits), esta dirección es muy importante, ya que es el nombre que usa el módulo dentro de cualquier red de la que forme parte, es única, y no se puede modificar.

Para poder conocer el número de serie del dispositivo, se observar que en la parte inferior del módulo Xbee, cuenta con una etiqueta en la que se especifica el número de serie, dividido en dos registros de 32 bits cada uno. Si por alguna razón la dirección de 64 bits no se puede apreciar, o no aparece, es posible consultarla a través de los comandos AT de configuración del módulo Xbee.

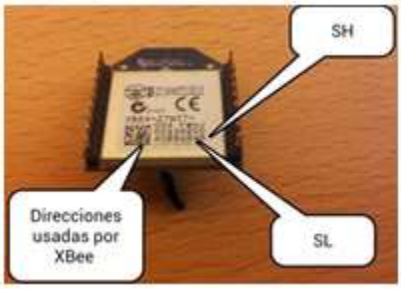


Figura 4.5 Dirección del módulo Xbee.

Para consultar la dirección del dispositivo mediante comandos AT, se conecta el módulo Xbee con *Hiperterminal,* como se hizo anteriormente en el subcapítulo 4.3, correspondiente al tema de configuración con comandos AT.

Una vez conectado se ingresan los caracteres de acceso +++, a lo que el módulo responde con un "OK" y una línea nueva, posteriormente se ingresa el prefijo "AT", seguido del comando "SH", y se presiona la tecla "**enter**".

El módulo responde mostrando la parte alta (32 bits más significativos) del registro de 64 bits (ver figura 4.6), en éste caso corresponde a "13A200". Al ejecutar el comando "ATSL" el módulo entrega la parte baja del número de serie de 64 bits, que corresponde a "4090EE8A". Finalmente se sale del modo de comandos ejecutando "ATCN". De esta forma se obtiene el registro 64 bits de los módulos Xbee empleados.

En la siguiente figura se muestra el procedimiento descrito, y la pantalla que se observa en *Hiperterminal.*

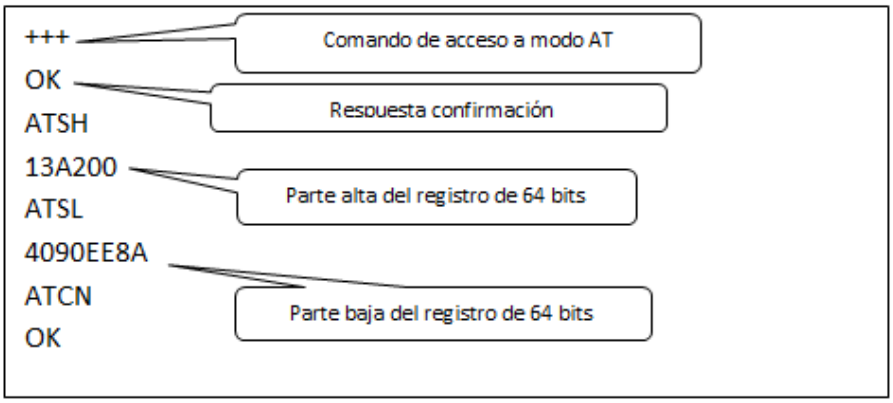


Figura 4.6 Obtención del número de serie del módulo transmisor, mediante comandos AT.

Para que los módulos Xbee logren establecer una comunicación entre sí, se utiliza una de las configuraciones más sencillas que soportan. Se denomina modo transparente, y es el que viene configurado por defecto. Este modo permite que los datos que entran por el pin 3 (D/INT) sean guardados en un buffer de entrada, y posteriormente transmitidos, enviándolos al módulo deseado. Cuando el módulo recibe un paquete de datos, estos son guardados en un buffer de salida, y enviados hacia el pin 2 (D/OUT) del Xbee.

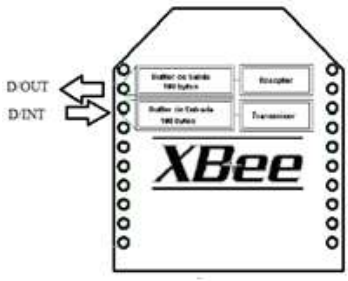


Figura 4.7 Buffers de entrada y salida.

Existen cuatro topologías del modo transparente, la diferencia radica en el número de nodos o puntos de acceso a la red inalámbrica, la topología empleada en el prototipo es conocida como conexión *punto a punto*.

La conexión *punto a punto* es la más sencilla que el módulo Xbee soporta, permite remplazar un cable de comunicación serial, por una conexión inalámbrica. Para lograrlo, solo basta con configurar 3 parámetros que definen a la red.

El primer parámetro es el ID de la PAN (Personal Area Network- Red de Área Personal), ya que todos los módulos con el mismo ID pertenecen a la misma PAN. El segundo parámetro es el canal que emplearan los dispositivos para comunicarse, por lo que es necesario definirlo, y finalmente es necesario escribir en cada módulo el número de serie del dispositivo con el que establecerá la conexión inalámbrica. Este número de serie se encuentra especificado en la parte inferior de cada módulo (o se puede obtener mediante los comandos AT como se hizo).

El ID de la PAN (Personal Area Network- Red de Área Personal) se configura con el comando "**ATID** HHHH", en donde H es un dígito hexadecimal, el parámetro "HHHH" puede tomar un valor de 0x0 a 0xFFFF, este parámetro es arbitrario y se puede establecer cualquier valor dentro del rango, el valor que se coloque aquí, deberá ser el mismo en el módulo con el que se pretende establecer la comunicación inalámbrica.

El segundo parámetro es el canal de comunicación que emplean los dispositivos Xbee. Los módulos nos permiten elegir de entre 16 posibles canales. Para que la comunicación entre dos dispositivos se establezca, deben estar configurados en el mismo canal. El comando "**ATCH**XX", permite configurar cualquiera de los 16 canales disponibles, el parámetro "XX" es un número hexadecimal con rango de 0x0B a 0x1A (de 11 a 26). La tabla siguiente muestra los 16 canales disponibles, y la forma para configurarlos mediante el comando "ATCH".

Canal	Hexadecimal	Comando AT
11	0x0B	**ATCH**0B
12	0x0C	**ATCH**0C
13	0x0D	**ATCH**0D
14	0x0E	**ATCH**0E
15	0x0F	**ATCH**0F
16	0x10	**ATCH**10
17	0x11	**ATCH**11
18	0x12	**ATCH**12
19	0x13	**ATCH**13
20	0x14	**ATCH**14
21	0x15	**ATCH**15
22	0x16	**ATCH**16
23	0x17	**ATCH**17
24	0x18	**ATCH**18
25	0x19	**ATCH**19
26	0x1A	**ATCH**1A

Tabla 1.5 Canales disponibles para la comunicación de los módulos Xbee.

Una vez configurados el ID de la PAN y el canal de comunicación que emplearan los dispositivos, se debe escribir en cada uno de ellos el número de serie del dispositivo con el que se pretende establecer la comunicación. Es decir, en el transmisor se debe escribir el número de serie del receptor, y en el receptor el número de serie del transmisor.

Para ello cada módulo cuenta con un par de registros internos de 32 bits, denominados DH y DL, en los que se escribe la parte alta y la parte baja del número de serie del módulo con el que se desea comunicarse. Los comandos empleados para escribir en los registros son: "ATDHXXXXXXXX", y "ATDLXXXXXXXX". En donde los parámetros "XXXXXXXX", son la parte alta y la parte baja respectivamente del número de serie del módulo Xbee receptor.

La figura 4.8, muestra la configuración que se hace en el módulo Xbee que actuara como transmisor del sistema.

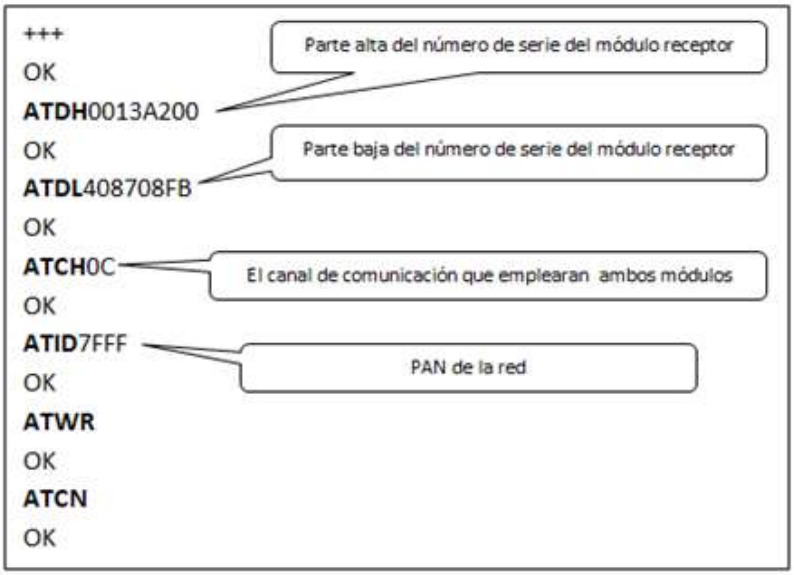


Figura 4.8 Configuración del módulo transmisor a través de Hiperterminal y los comandos AT.

En la configuración del módulo transmisor se ha guardado el número de serie del receptor, en los registros DH y DL. Se empleará el canal 12, y la PAN es 7FFF, estos dos últimos parámetros deben ser iguales al configurar el módulo receptor, como se verá más adelante.

Los módulos Xbee tiene una tensión de operación de 3.3 [V], y no debe sobre pasarse por ningún motivo, o el módulo podría sufrir daños permanentes. Dado que el sistema transmisor utiliza un microcontrolador PIC que trabaja con una tensión nominal de 5 [v], es importante poner especial atención en implementar un acoplador de niveles de tensión.

La idea es utilizar la comunicación serial del microcontrolador PIC, para transmitir datos y remplazar el cable por una comunicación inalámbrica con los módulos Xbee. El microcontrolador PIC emplea niveles lógicos de tensión de 0 a 5 [V] en su módulo USART, mientras que el Xbee emplea de 0 a 3.3 [V]. Para solucionar el problema de una manera sencilla, se emplea el siguiente circuito de la figura 4.9.

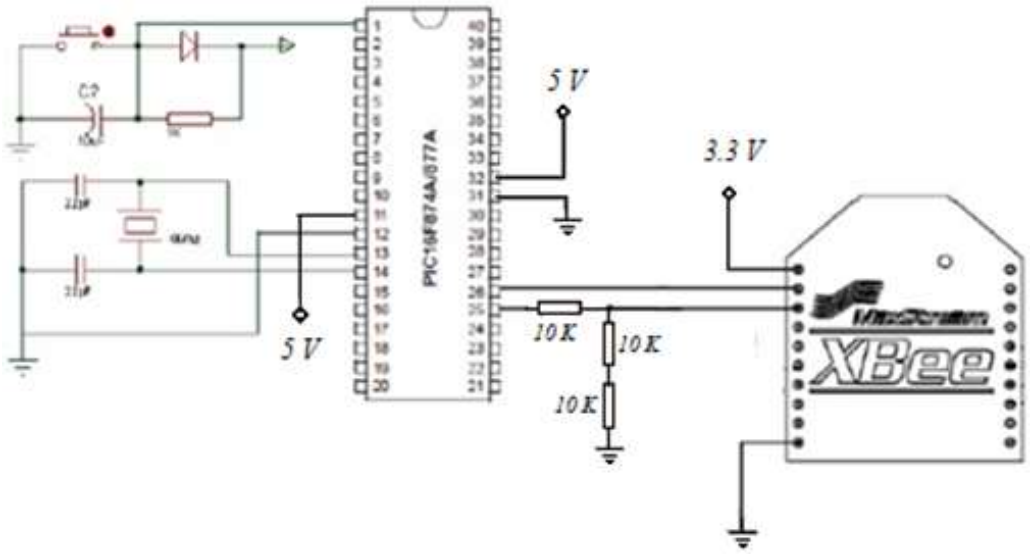


Figura 4.9 Circuito adaptador de niveles de tensión para el módulo transmisor.

El divisor de voltaje en la salida de datos del Microcontrolador (pin 25), es totalmente necesario, ya que la tensión en la entrada de datos del módulo Xbee (pin 3), no debe sobrepasar los 3.3 [V] o el módulo podría sufrir daños. De esta forma siempre se obtendrá 2/3 del voltaje de la salida del PIC, y como el máximo valor es 5 [V], entonces 5(2/3)=3.3 [V], así se obtiene un nivel de tensión adecuado para la entrada de datos del Xbee.

El diagrama de la figura 4.9, corresponde al circuito transmisor que se empleará en el prototipo de sistema de control de flujo. Los datos de volumen y caudal serán enviados por el microcontrolador al módulo Xbee, quien se encargara de transmitirlos inalámbricamente hacia un módulo receptor para su interpretación.

El circuito de la figura 4.9 también actuará como receptor, cuando los comandos de control enviados desde la computadora lleguen al microcontrolador, quien se encargará de ejecutarlos.

4.5. Receptor

El prototipo desarrollado cuenta con un circuito receptor que es capaz de recibir los datos enviados desde el microcontrolador, para después ingresarlos a la computadora donde serán procesados. Este receptor se construye usando un módulo Xbee de la serie 1, compatible con el módulo Xbee del transmisor.

Los dos dispositivos se encargan de establecer la transmisión de datos de forma inalámbrica, entre el circuito de instrumentación del sensor y la computadora en donde se ha implementado un panel virtual de monitoreo y control. La configuración del módulo receptor, se realiza utilizando la interfaz desarrollada para permitir la comunicación del Xbee con la computadora, y con la ayuda de los comandos AT en *Hiperteminal*.

Para que los módulos transmisor y receptor puedan establecer una comunicación inalámbrica, se debe tener en cuenta que tiene que pertenecer a la misma red de área personal PAN, y deben emplear el mismo canal de comunicación. Además se tiene que escribir en los registros correspondientes, el nombre del dispositivo con el que se pretende comunicarse.

Para configurar el módulo receptor se utiliza el comando "ATID7FFF" (ver figura 4.10), el cual configura la misma red de área personal empleada en el módulo transmisor, que es 7FFF. Con el comando "ATCH0C", se establece que el canal utilizado por el módulo receptor será el 12, que coincide con el canal configurado anteriormente en el módulo transmisor. En la figura 4.10, se muestra la forma en que se realiza la configuración del módulo receptor mediante comandos AT, e *Hiperteminal.*

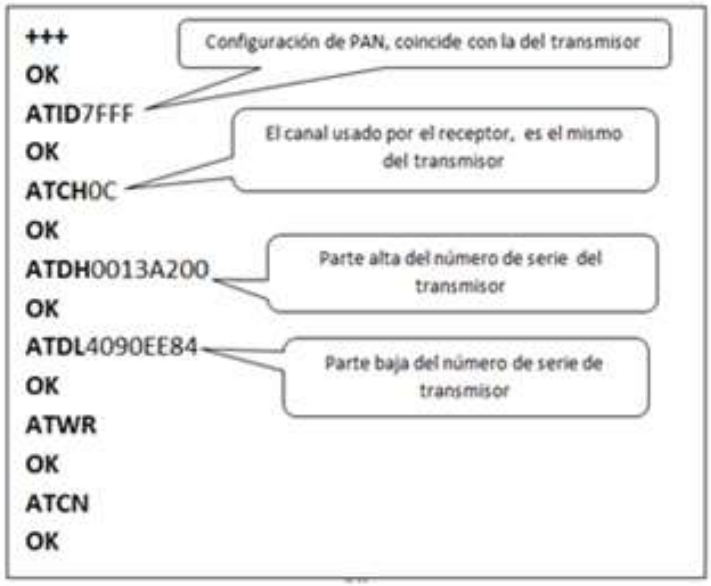


Figura 4.10, configuración del módulo receptor a través de Hiperterminal y los comandos AT.

Es necesario definir el nombre del dispositivo con el que se hará la conexión, escribiendo en los registros DH y DL la parte alta y la parte baja respectivamente del número de serie del módulo transmisor. En la figura 4.10 se observa que la parte alta del número de serie del módulo transmisor es; "0013A200", y es guardada en el registro **DH,** utilizando el comando "**ATDH**0013A200", mientras que el comando "ATDL4090EE84" hace lo mismo para la parte baja del número de serie del módulo transmisor. De este modo el enlace entre los módulos, transmisor y receptor, queda totalmente definido, y no debe de presentar problema alguno al comunicarse entre sí.

Después de haber configurado correctamente el módulo Xbee receptor, este debe utilizar un circuito adaptador de niveles de tensión para comunicarse con la computadora. El circuito empleado se muestra en la figura 4.11.

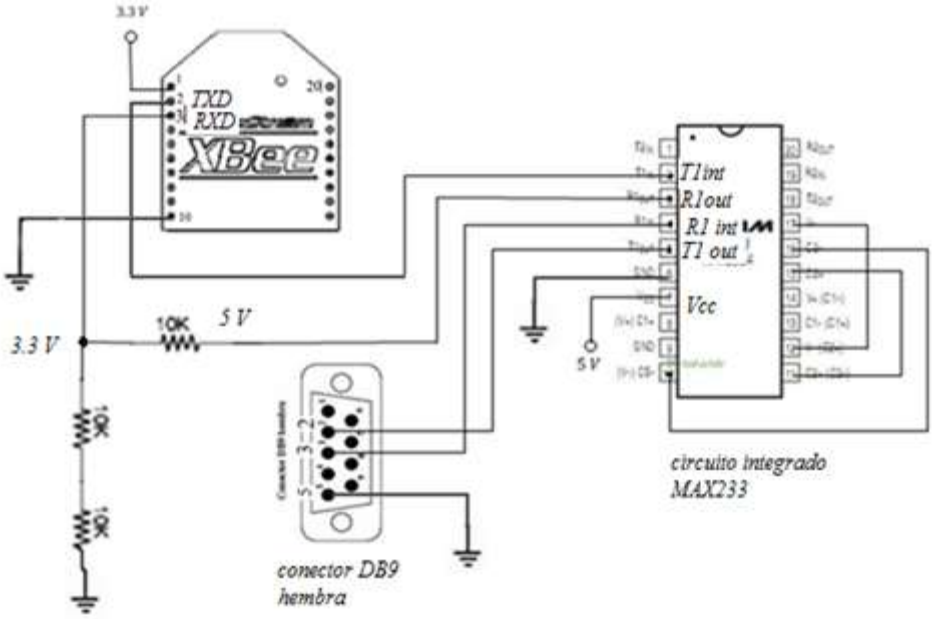


Figura 4.11 Circuito receptor en el sistema.

El circuito de la figura 4.11 es utilizado para adaptar los niveles de tensión que utiliza el módulo Xbee en el protocolo de comunicación serial, que son de 0[V] a 3.3 [V], con los niveles empleados en una computadora que son de +15[V] a -15[V]. Este circuito es el mismo que se utiliza para la configuración de los módulos Xbee.

Capítulo 5

Desarrollo del panel de control virtual mediante LabView

5.1. Uso de los bloques VISA para comunicación serial

El prototipo de sistema de monitoreo y control de flujo, utiliza el puerto serie de la computadora para poder comunicarse con el microcontrolador que forma parte del circuito de instrumentación del sensor de flujo. La comunicación entre el microcontrolador y la computadora es de envío y recepción de datos, es decir, ambos pueden enviar y recibir datos.

El microcontrolador envía los datos de volumen y caudal medidos en el sensor hacia la computadora, en donde la información es mostrada mediante indicadores virtuales. Pero también tiene que recibir los datos y comandos enviados desde el panel de control en la computadora, para su ejecución.

Para lograr establecer una comunicación entre el microcontrolador y la computadora se utiliza el software de instrumentación virtual, LabView. En este software se diseñó e implementó el panel de control virtual que permitirá operar el sistema.

Antes que nada es necesario configurar un puerto de comunicación entre LabView y el microcontrolador, la comunicación entre ambos es a través del protocolo de comunicación serial, para poder habilitar y configura el puerto de comunicación, se hace uso de los bloques VISA (Arquitectura de Software de Instrumento Virtual) de Labview.

VISA puede comunicarse y controlar dispositivos seriales utilizando los driver apropiados, sin tomar en cuenta el tipo de interface que utilicen. Para establecer la comunicación serial con otro dispositivo se tiene que configurar el puerto de comunicación mediante los bloques NIVISA.

Para encontrar los bloques NIVISA es necesario acceder en la siguiente dirección, ubicada en el espacio de diagrama de bloques en LabView: ***Functions>>Instrument>>VISA.***

En la dirección anterior se encuentra un menú con diferentes bloques que cumplen con tareas específicas, como la lectura y la escritura sobre algún puerto de comunicación.

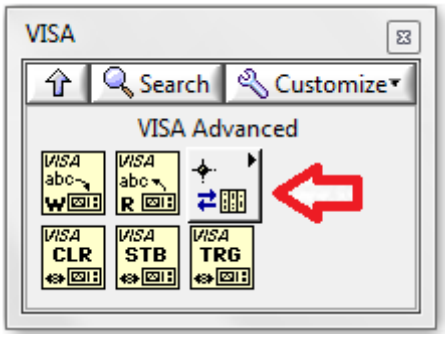
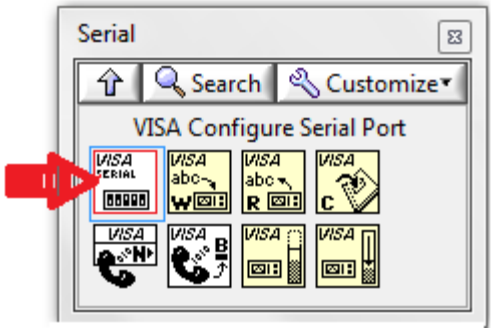


Figura 5.1 Menú de bloques VISA.

LabView cuenta con un bloque que permite la configuración del puerto de comunicación serial que se utilizará. Para encontrar el bloque se accede a la siguiente dirección: **functions>> Instrument I/O>> VISA advance>>** ***Bus/Interface Specific*****>>** ***VISA Configure Serial Port***.



5.2 Menú de bloques para manejar comunicación serial.

Como se observa en la figura 5.3 en el bloque "*VISA Configure Serial Port*", se requiere configurar los parámetros de: velocidad de *baudaje* (bits por segundo), los bits de datos, y la paridad, entre otros. Para establecer la comunicación vía serial con otro dispositivo, es necesario que sus parámetros sean compatible con los que se configuran en este bloque.

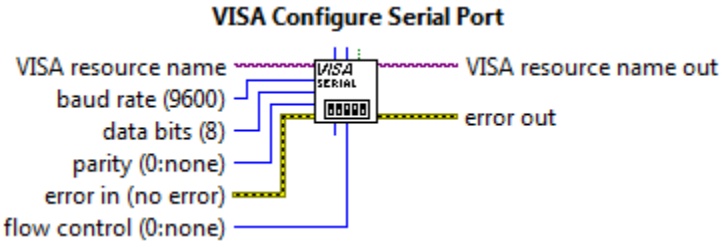


Figura 5.3 Bloque de configuración del puerto serie.

El parámetro *VISA resource name* del bloque de configuración del puerto serie, en la figura 5.3, es el nombre del puerto físico en la computadora que LabView utilizará para comunicarse con el dispositivo conectado ahí. Al crear un control en este parámetro, en el panel frontal de LabView se mostrará un menú con todos los puertos COM disponibles en la computadora.

Para terminar con la configuración del puerto, se establece la velocidad de datos empleada en la comunicación, mediante una constante igual a 9600. Los demás parámetros del bloque de configuración del puerto se dejan establecidos con los valores que tienen por defecto, como se observa en la figura 5.4.

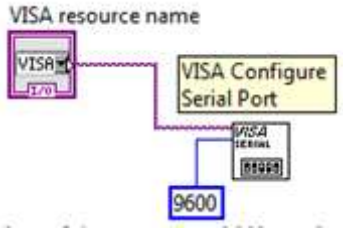


Figura 5.4 Parámetros del bloque de configuración
Del puerto serie.

Una vez definidas las características del puerto físico, primero se debe asegurar que no haya ningún evento anterior que este pendiente, para eso se utiliza el bloque "*VISA Discard Events"*, que se encuentra en la siguiente dirección: ***functions>>Instrument I/O>>VISA advance>>Event Handling>>VISA Discard events.***

Figura 5.5 Bloque para descartar eventos.

Este bloque es conectado directamente al bloque de configuración de puerto serial, como se observa en la figura 5.6, ya que cumplirá con la función de descartar cualquier evento pendiente, y de esta forma se evitarán errores cada vez que el puerto serial se habilite.

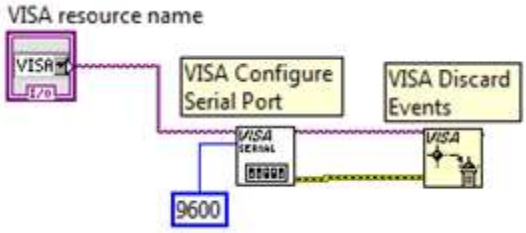


Figura 5.6 Configuración del puerto serie,
y descarte de eventos pendientes.

Después de haber descartado cualquier evento anterior que pudiera generar errores, se debe habilitar nuevamente los eventos que surjan durante la transmisión de datos actuales, para lo cual se emplea el bloque "VISA *EnableEeven*". Localizado en: ***functions>>Instrument I/O>>VISA advance>>Event Handling>>VISA Enable Events.***

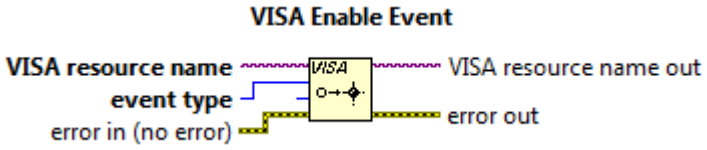


Figura 5.7 Bloque para habilitar eventos.

El bloque "*Enable Event"* habilita las notificaciones acerca de algún tipo de evento seleccionado en su parámetro *event type*. Como el módulo de instrumentación del sistema envía los datos del sensor de flujo mediante caracteres ASCII, se selecciona la opción de carácter serial (*Serial Character*). Este bloque se conecta con los dos anteriores, con el de configuración de puerto, y con el de descarte de evento, como se muestra en la figura 5.8.

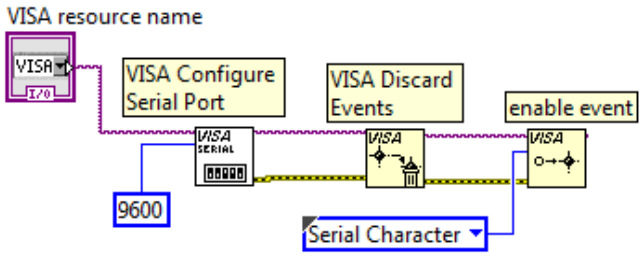


Figura 5.8 Conexión de los bloques de configuración de puerto, descarte de evento, y habilitación de evento.

Después de haber configurado el puerto descartado cualquier error, y habilitado las notificaciones del tipo de datos a recibir por el puerto; se utiliza una estructura de control de casos (*Case Structure*) con el fin de poder descartar errores que surjan durante la comunicación con el dispositivo transmisor de datos.

La estructura de control tiene dos casos, el primer caso (true) se utiliza para recibir y procesar los datos que llegan por el puerto serial. El segundo caso (false) se ejecuta cuando surge algún error de comunicación.

Para asignar los dos casos a la estructura de control, se conecta en la línea de control de error del bloque "*Enable Event*" un elemento llamado "*Unbundle By Name*" (control de estados) y un operador "*not*", con el fin de convertir el estado de error del puerto, en

valores booleanos, y emplearlos en la estructura de casos. Como se muestra en la figura 5.9.

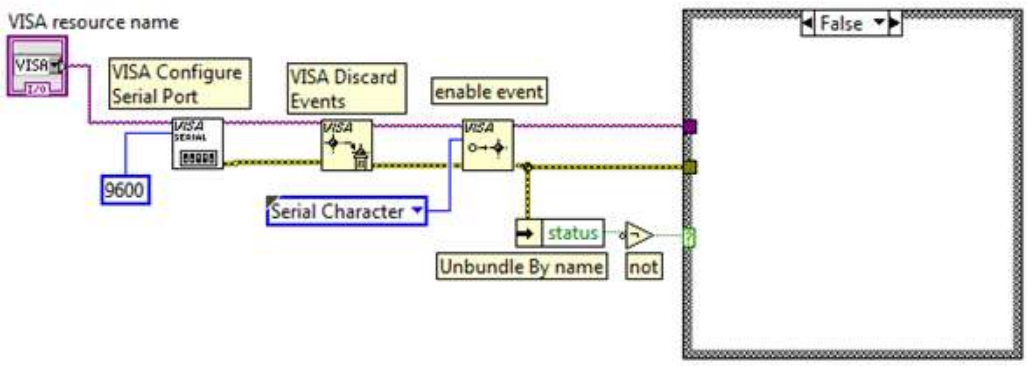


Figura 5.9 Asignación de casos mediante
Un control de estados, y un operador not.

Cuando se presenta un error en el puerto de comunicación serial, la estructura de control ejecuta el caso "**false**", en el que se debe canalizar hacia alguna acción de control de errores. Para ello se utilizan los bloques "*disable event*" para deshabilitar el evento que ha causado el error, "*discad events*" para eliminar el error, y finalmente se cierra el puerto para protegerlo de algún disturbio eléctrico con el bloque "VISA *Close*". Como se observa en la siguiente figura 5.10.

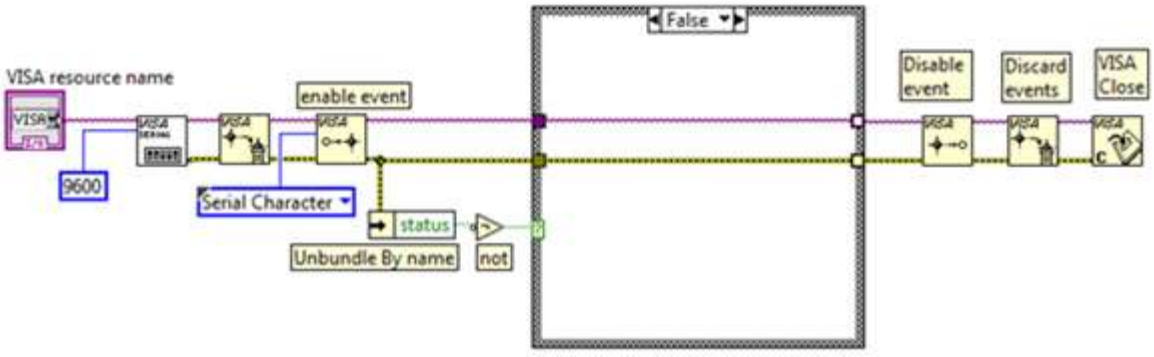


Figura 5.10 Manejo de error a través del caso "false", con los bloques de:
deshabilitación de evento, descarte de evento, y cierre del puerto.

Los bloques "*Disable Event*" y "*Discard event*", se encuentran en la siguiente dirección: **Functions>>InstrumentI/O>>VISA>>Visa Advance>>Event Handling>>Disable Event/Discard event.**

Y el bloque "VISA Close" se encuentra en la siguiente dirección: **Functions>>InstrumentI/O>>VISA>>Visa Advance>>VISA Close.**

5.2. Lectura por puerto serial

Para realizar la lectura de datos en el puerto de comunicación serial, es necesario utilizar el segundo caso (**True**) de la estructura de control, que se observa en la figura 5.10 del sub capitulo anterior, y los dos bloque que se describen a continuación.

El bloque *VISA wait on Event*, espera a que el evento de recepción de carácter ASCII se cumpla antes del tiempo que se le especifica en el parámetro *Timeout()*. Si esto no ocurre, se genera un error que es manejado con los bloques que se vieron anteriormente. El bloque se encuentra en la dirección: **Functions>>Instrument I/O>>VISA>>VISA advanced>>Event Handling>> VISA Wait on Event.**

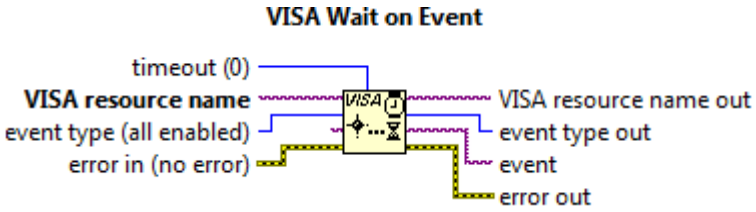


Figura 5.11 Bloque de espera de evento.

LabView cuenta con un bloque llamado "*VISA Read*", el cual permite la lectura de datos en el puerto de comunicación serial, el único parámetro que se configura es *byte count,* en donde se especifica cuantos bytes debe llegar del puerto serie antes de mandarlos al buffer de lectura, en donde se entrega una cadena de caracteres. El bloque *VISA Read* se encuentra en la dirección: **Functions>>Instrument I/O>>VISA>>VISA Read.**

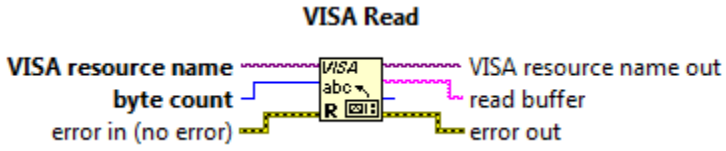


Figura 5.12 Bloque de lectura en el puerto.

El parámetro *Timeout* en el bloque "*Wait on Event*" es cargado con un valor de 1000[ms]. Es decir que cuando el puerto de comunicación serial se habilite, el bloque esperará a que se reciba un carácter ASCII en un tiempo máximo de 1 [s], de lo contrario generará un error.

El parámetro *byte count* del bloque "*VISA Read*", es cargado con el valor de 7, que significa que el bloque esperará una trama de datos de 7 bytes antes de leerlos.

Después de haber definido los parámetros en los bloques, estos se coloca dentro de un bucle infinito *while* para realizar la lectura de forma continua. Como se indica en la figura 5.13.

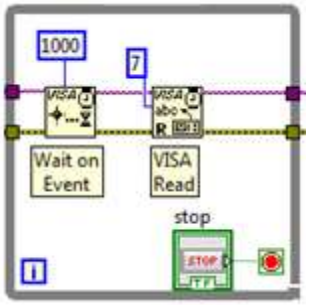


Figura 5.13 Lectura continua mediante el uso del ciclo infinito while.

El bucle infinito *while* que contiene a los dos bloques empleados para realizar la lectura a través del puerto, debe ser colocado dentro del caso "**True**" de la estructura de control que se vio en el subcapítulo 5.1, como se indica a en la figura 5.14.

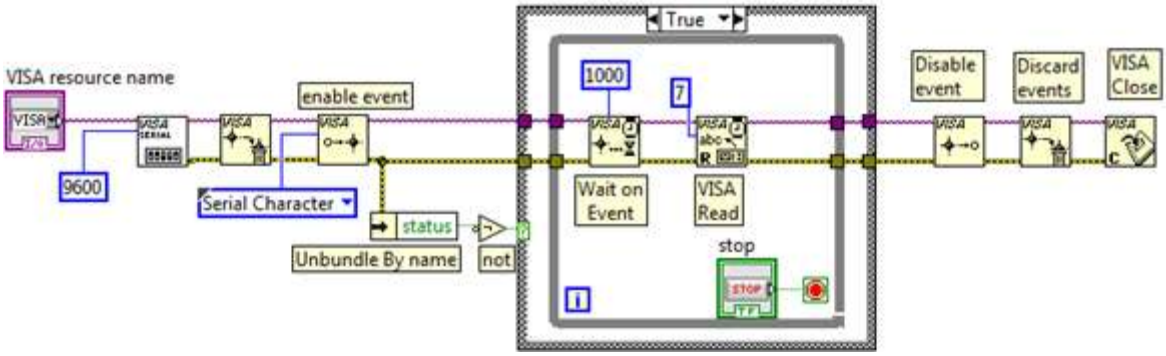


Figura 5.14 La lectura del puerto se lleva a cabo en el caso "True" de la estructura de control.

Con el diagrama de bloques de la figura 5.14 se obtiene el programa que sirve para leer los datos del puerto de comunicación serial. Ahora solo falta establecer el formato con el que los datos deben ser enviados, para que al ser leídos puedan ser interpretados.

En el buffer de lectura se espera recibir los datos de volumen y caudal que pasa a través del sensor de flujo. Estos datos son concatenados por el microcontrolador antes de ser enviados con el siguiente formato.

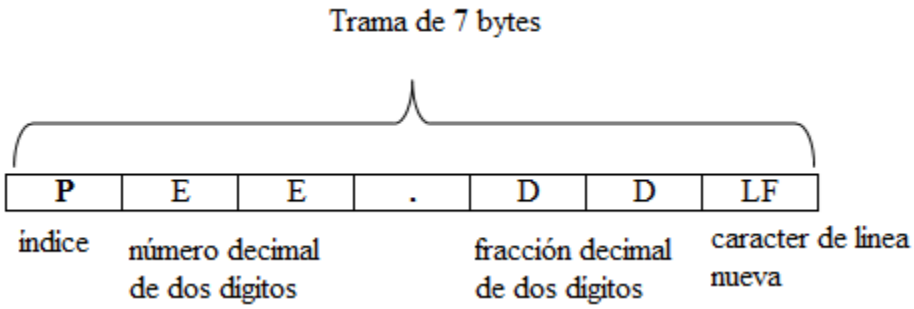


Figura 5.15 Formato de los datos que se reciben por el puerto serie de la computadora.

En donde P: es el índice con valores de "0" o "1", que indica si el dato enviado corresponde al volumen leído por el sensor, o a la caudal. La lectura hecha por el sensor debe ser un valor con dos dígitos enteros y dos dígitos después del punto decimal (parámetro: E E.D D en la figura 5.15).

LF: Es el carácter de nueva línea (10 en el código ASCII) que se concatena al final de la cadena, ya que indica el final del dato.

El microcontrolador lee el volumen que pasa por el sensor, este dato se envía con un índice igual a "0", y el valor máximo que puede tener es de 65.53 litros.

La velocidad a la que se mueve el volumen del líquido dentro del sensor se calcula en el microcontrolador , y se envía a través del puerto de comunicación serial con un índice igual a "1". El valor máximo que puede alcanzar es de 25.00 litros/minuto, ya que este es un parámetro límite del sensor.

Para lograr leer correctamente los datos concatenados enviados desde el módulo transmisor, se requiere separarlos una vez hayan sido recibidos. Primero se separa el índice del resto de la trama, y se compara, dependiendo del valor que tenga, "0" o "1", se sabrá si el dato corresponde al volumen o al caudal, y será mostrara en el panel de control con un indicador adecuado.

Al leer los datos del puerto de comunicación serial, el bloque *VISA Read* entrega en el buffer de lectura una cadena de caracteres, de la cual se tiene que separar el índice que indica a que variable corresponde el dato contenido en ella.

Para separar la cadena se utiliza el bloque "*String Subset*". El bloque entrega una sub cadena, que corresponde a la cadena de entrada comenzando desde el offset, y que contiene el número de caracteres indicados en el parámetro longitud (length).

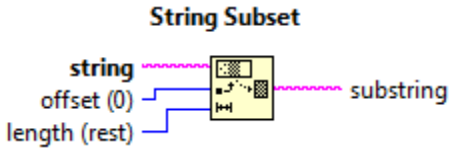


Figura 5.16 Bloque para separar una cadena de caracteres.

El bloque "String subset" se encuentra en la dirección: **functions>>String>>String Subset.**

La cadena de datos concatenados proveniente del módulo transmisor, es separada por éste bloque para leer el índice. De este modo el parámetro offset debe ser "0", y la longitud (length) "1", a si la sub cadena de salida corresponde al carácter que representa al índice, que posteriormente es comparado para saber a qué indicador mandar el dato. Como se aprecia en la figura 5.17.

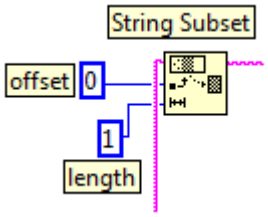


Figura 5.17 Separación del índice de la cadena de caracteres.

Para poder comparar el valor del índice, se requiere convertir la sub cadena entregada por el bloque *Substrig Subset,* y que contiene al índice, en un número entero, con el bloque "*Decimal String To Number*". Este bloque convierte los caracteres numéricos en una cadena, a un número decimal entero.

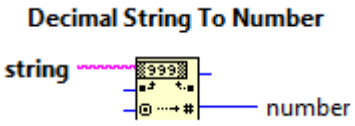


Figura 5.18 Bloque para convertir una cadena de caracteres, a un número entero.

El bloque "*Decimal String To Number*" se encuentra en la dirección: **functions>>String>>String/Number Conversion>>Decimal String To Number.**

El índice de la cadena de datos, señala a que indicador corresponde el valor enviado desde el transmisor. En el panel frontal del programa en Labview se muestran dos indicadores.

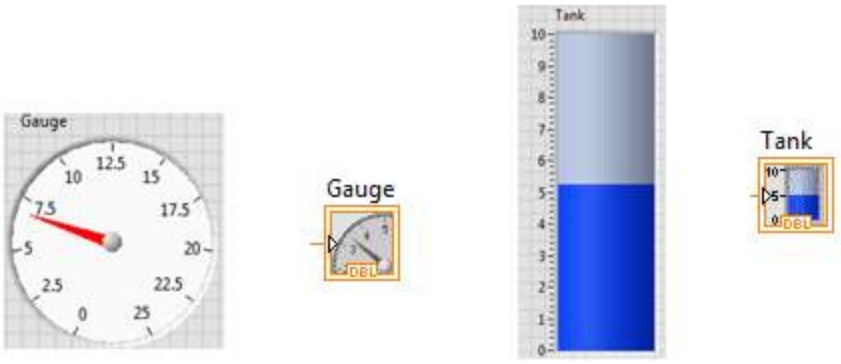


Figura 5.19 Indicadores para caudal y volumen, con su representación en el diagrama de bloques.

Éstos indicadores deben recibir el dato apropiado ya que tienen comportamientos y escalas distintas. El índice "0" de la cadena de caracteres, asigna el dato al indicador de volumen (Tank), mientras que el "1" lo asigna al indicador de caudal (Gauge).

Para poder discriminar adecuadamente el dato enviado a través del puerto serial, se utiliza una estructura de control de casos. En la que el caso a ejecutar es asignado con la ayuda de un arreglo de elementos "*enum constant*".

Para formar el arreglo de "*enums"* se siguió el siguiente procedimiento:

Primero se elige un elemento "*enum*" localizado en la dirección **functions>>Numeric>>Enum Constant.** Al presionar el botón secundario del mouse sobre el elemento, se despliega un menú en el que aparece la opción **Edit Items…** al acceder a él, aparece la ventana en la figura 5.20, en la que se debe escribir el nombre de los casos que posteriormente se asignaran a la estructura de control de casos. Al presionar el botón OK, el elemento "*enum*" queda configurado con los dos casos.

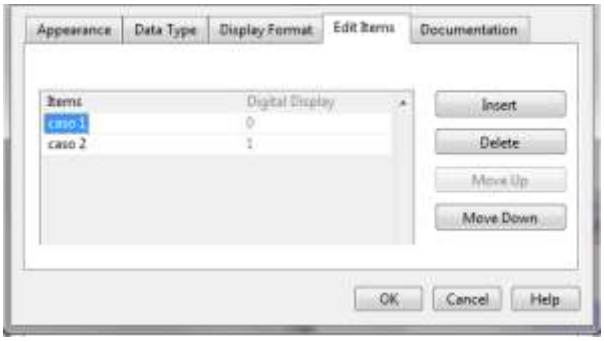


Figura 5.20 Ventana de configuración de casos.

Después se conecta el elemento "*enum*" a la estructura de casos, al hacerlo los casos configurados son asignados automáticamente a la estructura de control de casos como se muestra en la figura 5.21.

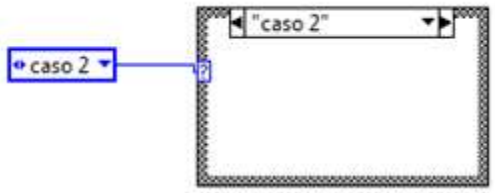


Figura 5.21 Asignación de casos en la Estructura de control.

El arreglo se construye desconectando el "*enum*" de la estructura de casos, y arrastrándolo al interior del bloque "*Array Constant*", que se encuentra en: **Functions>>Array>>Array Constant.** A continuación se agrega otro elemento al arreglo, arrastrando el extremo inferior del arreglo hacia abajo. Por último se habilitan los dos elementos del arreglo con cada caso configurado en el "*enum*", como se muestra en la figura 5.22.

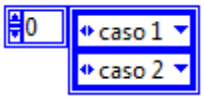


Figura 5.22 Construcción del arreglo de casos.

Para que este arreglo pueda asignar los casos a la estructura de control, se emplea el bloque "*Index Array*", que se encuentra en: **Functions>>Array>> Index Array.** Éste bloque recibe un arreglo en la entrada, y emplea un índice para devolver el elemento del arreglo indicado por el índice. En la entrada se conecta el arreglo de "enums" anterior, y la salida se conecta con la estructura de control de casos, como se muestra en la figura 5.23.

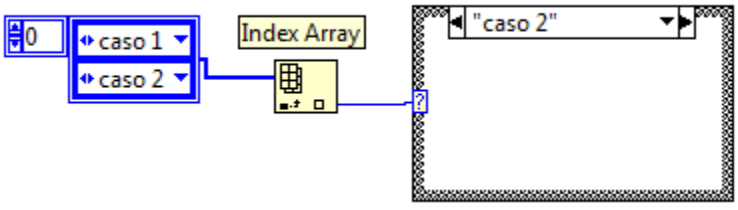


Figura 5.23 Asignación de casos en la estructura A través del arreglo y el índice de la cadena.

De este modo la asignación de casos se realiza a través de un arreglo, que en su "elemento 0" tiene el caso 1 y en el "elemento 1" tiene el "caso 2". Los elementos del arreglo

coinciden con los valores que utiliza el índice de la cadena de datos enviada desde el módulo transmisor.

Para completar la asignación de casos se emplean los bloques vistos anteriormente, "*String subset*" para separar el índice de la cadena de caracteres. Este bloque tiene como parámetros *offset* igual con "0", y longitud (*length)* igual con"1" (ver figura 5.24).

Una vez separado el índice del resto de la trama de datos, el bloque "*Decimal String To Number*" convierte el carácter del índice a un número decimal, que se utiliza para seleccionar el caso correspondiente del arreglo de "*enums*", como se muestra en la figura 5.24.

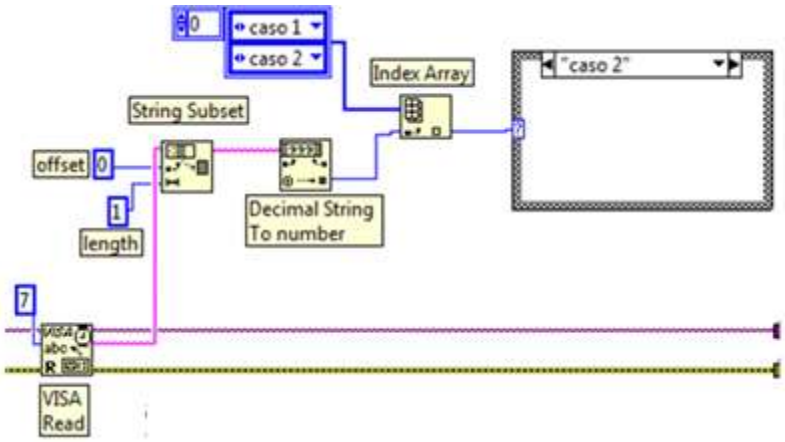


Figura 5.24 Separación del índice de la cadena de datos, y asignación de casos en la estructura de control.

Ahora solo se colocan los indicadores correspondientes dentro de la estructura de control de casos, y se hace llegar el resto de la trama de datos enviada por el transmisor. Para lo cual se hace uso del bloque "*Fract/Exp String To Number*" localizado en: **Functions>> String>> String /Number Conversion>> Fract/Exp String To Number.**

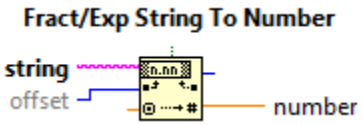


Figura 5.25 Bloque que convierte un valor contenido en una cadena de caractere, a un número punto flotante.

El bloque "*Fract/Exp String To Number*" interpreta los caracteres del 0 al 9 y el punto decimal contenidos en una cadena, comenzando por el offset, y los convierte a un numero punto flotante. Éste bloque se coloca dentro de la estructura de casos, y se encarga de

separar el resto de la trama de datos que contiene el valor correspondiente a cada indicador. El valor del parámetro **offset es** "1" para evitar leer la cadena de caracteres desde la posición 0, la cual contiene a el índice que asigna los casos.

Finalmente los indicadores de volumen y caudal son colocados en ambos casos de la estructura de control, de este modo el índice de la cadena de caracteres identificarán a cuál de ellos será enviado el dato transmitido desde el microcontrolador.

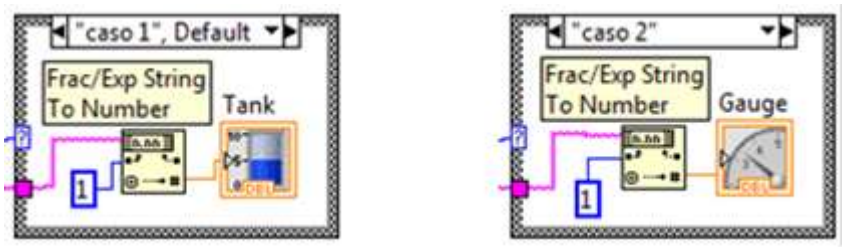


Figura 5.26 Los indicadores de volumen y caudal, son colocados en los casos de la estructura de control.

Los indicadores se encuentran en la siguiente dirección del panel frontal de LabView **Controls>>Modern>>Numeric>>Tank/Gauge.**

En la figura 5.27 se observa el diagrama de bloques dentro del ciclo infinito "*while*" que se utiliza para la lectura de datos del puerto serie.

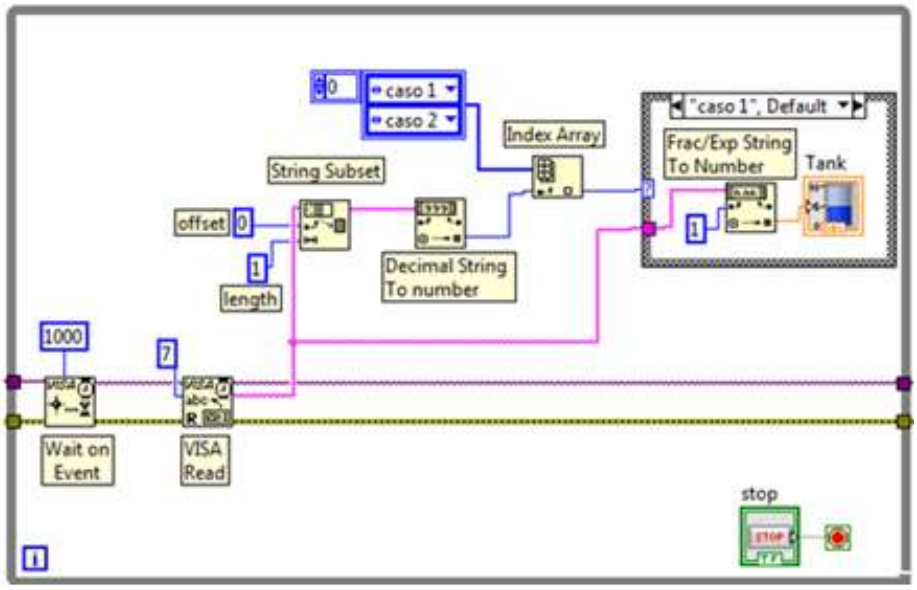


Figura 5.27 Diagrama de bloques completo para leer los datos de volumen y caudal envidos a través del puerto serie.

Es importante que en la trama de datos aparezca el carácter de nueva línea "LF" al final (10 en código ASCII), ya que indica el final del dato enviado, y el bloque "*Fract/Exp String To Number*" lo requiere para poder convertir el dato en la cadena a un número punto flotante.

5.3. Escritura por puerto serial

Para poder controlar el sistema desde el panel virtual, es necesario poder escribir los comandos de control en el puerto de comunicación serial, que serán decodificados por el microcontrolador posteriormente. Estos comandos permiten controlar el volumen de líquido que pasa a través del sensor de flujo.

LabView cuenta con un bloque que permite escribir datos en el puerto de comunicación serial, el bloque "*VISA write*" recibe una cadena de datos y la escribe en el puerto serial. Se localiza en la dirección: **Functions>>Instrument I/O>>VISA>> VISA Write.**

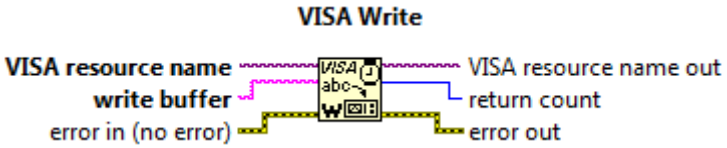


Figura 5.28 Bloque utilizado para escribir en el puerto serie.

Para poder interpretar correctamente los comandos que se envían del panel de control virtual, hacia el microcontrolador quien se encarga de ejecutarlos, se establece el siguiente formato con el que los datos tendrán que enviarse.

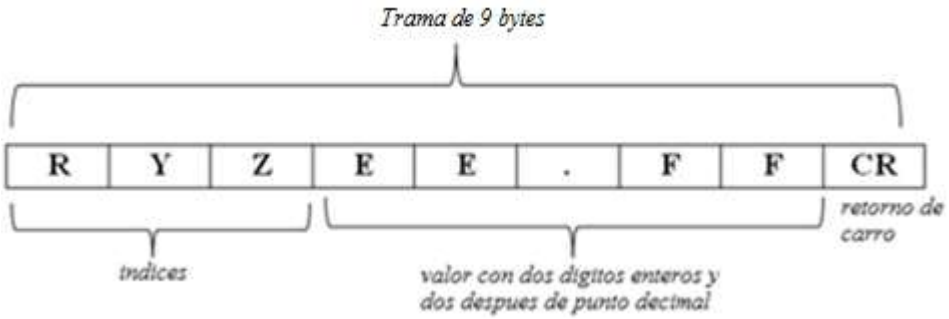


Figura 5.29 Formato para el envío de datos.

En donde R, Y, Z son tres índices que tienen dos estados lógicos, y se utilizan para saber la acción a ejecutar en el microcontrolador.

El parámetro E E . F F en la figura 5.29, es un valor con dos dígitos en la parte entera y dos dígitos después del punto decimal. Se utiliza para establecer un volumen cuando el sistema opera de forma automática. En la última posición el carácter "CR" **de** retorno de carro (13 en código ASCII), indica el final de la trama de datos.

Para obtener el formato requerido se utiliza el bloque de LabView "*Concatenate Strings*", este bloque permite concatenar las cadenas de caracteres en la entrada, y devuelve una sola cadena compuesta por todas las entradas. El bloque se encuentra en la dirección: **Functions>>String>>Concatenate Strings.**

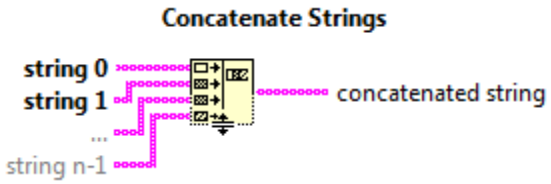


Figura 5.30 Bloque para concatenar cadenas de caracteres.

El bloque permite agregar más entradas. Para hacerlo, se coloca el cursor en la parte inferior o superior del bloque, y se arrastra con el mouse, esta acción agrega o quita entradas del bloque.

Para formar la trama de datos con el formato especificado, se requiere de cinco entradas. La primera para el índice "R", la segunda para el índice "Y", la tercera para el índice "Z", la cuarta entrada espera recibir un valor con dos dígitos en la parte entera y dos después del punto decimal, es decir, de cinco bytes contado el punto. Finalmente en la quinta entrada se coloca el carácter *de retorno de carro* "**CR"** que indica el final de la trama.

Para establecer los dos estados de los índices "R, Y, Z" se usa el elemento "*Select*" que se encuentra en: **Functions>>Comparison>>Select.** Este elemento permite asignar un carácter a cada estado lógico de su entrada.

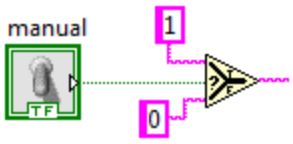


Figura 5.31 Asignación de caracteres 1 y 0,
a los estados lógicos de entrada.

Para la entrada del elemento "*Select*" se utiliza un control booleano. Cuando el control está en estado lógico "True", entrega un carácter "1", y cuando está en "false" entrega un "0", estos son los estados lógicos que definen al índice "R". Es importante recordar que el carácter "0" corresponde al valor 48 en ASCII, y el "1" es el 49 ASCII, se tiene que tener en cuenta ya que éstos son los valores que serán enviados al microcontrolador.

De una forma similar se utilizan los caracteres “2 y 3” para definir el índice “Y”, mientras que el índice “Z” queda definido con los caracteres “4 y 5”. Como se muestra en la figura 5.32.

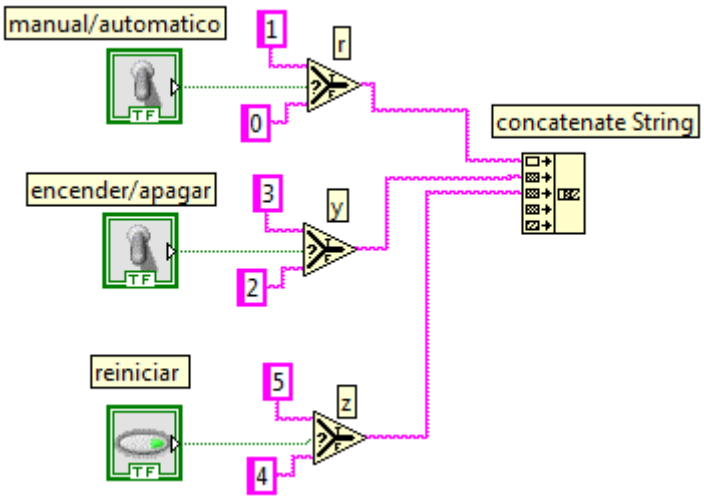


Figura 5.32 Asignación de caracteres a los estados lógicos de los índices R, Y, Z.

El estado lógico del índice “R” nos permite habilitar el modo manual o automático del sistema. Si se encuentra en el modo manual, el índice “Y” nos ayuda a encender o apagar la bomba que hace que el líquido pase a través del sensor de flujo. Por último el índice “Z” es controlado por un *push button* en el panel de control, que al estar en estado lógico “True” envía un carácter 5 (53 en código ASCII), que hace que el contador con el que se mide el volumen de líquido en el microcontrolador se reinicie, permitiendo así, una nueva medición del volumen.

Nota: Es importante recordar que los caracteres utilizados son enviados en su representación ASCCII. Por lo que en el microcontrolador se trabaja con los valores ASCII correspondientes a cada carácter.

El modo automático del sistema permite fijar un valor para el volumen que pasa a través del sensor, antes de interrumpir el flujo. De esta forma el sistema controla el volumen de forma automática.

El volumen que se desea, que pase por el sistema, se indica a través de un control virtual llamado “*Dial*”. Éste control entrega valores numéricos que no pueden ser concatenados directamente a la trama de caracteres, por lo que se requiere convertir el dato numérico a una cadena de caracteres.

El control virtual “*Dial*” se encuentra en la dirección del panel frontal: **Controls>>Modern>>Numeric>>Dial.**

El bloque "*Number To fractional String*" convierte un número fraccional, en una cadena de caracteres. El parámetro "*precision*" indica el número de dígitos después del punto decimal que el bloque debe entregar. El bloque se encuentra en la dirección: **Functions>>String>>String Number Conversion>> Number To fractional String.**

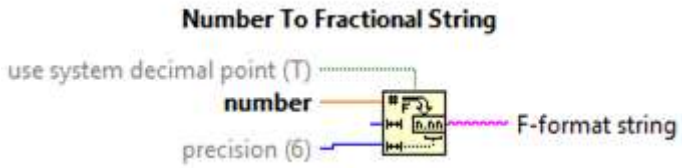


Figura 5.33 Bloque para convertir un número
Punto flotante, a una cadena de caracteres.

El valor numérico entregado por el control "*Dial",* es convertido a una cadena de caracteres con el bloque "*Number To fractional String*", en el bloque se especifican dos dígitos después del punto decimal (ver figura 5.34). El valor numérico convertido a caracteres, ahora puede ser concatenado con el resto de la trama de datos.

El carácter constante de *retorno de carro* es el último elemento que se agrega a la cadena concatenada, ya que es el que indica al microcontrolador el final de la trama de datos enviados. En la figura 5.34 se ilustra el diagrama de bloques para la escritura en el puerto.

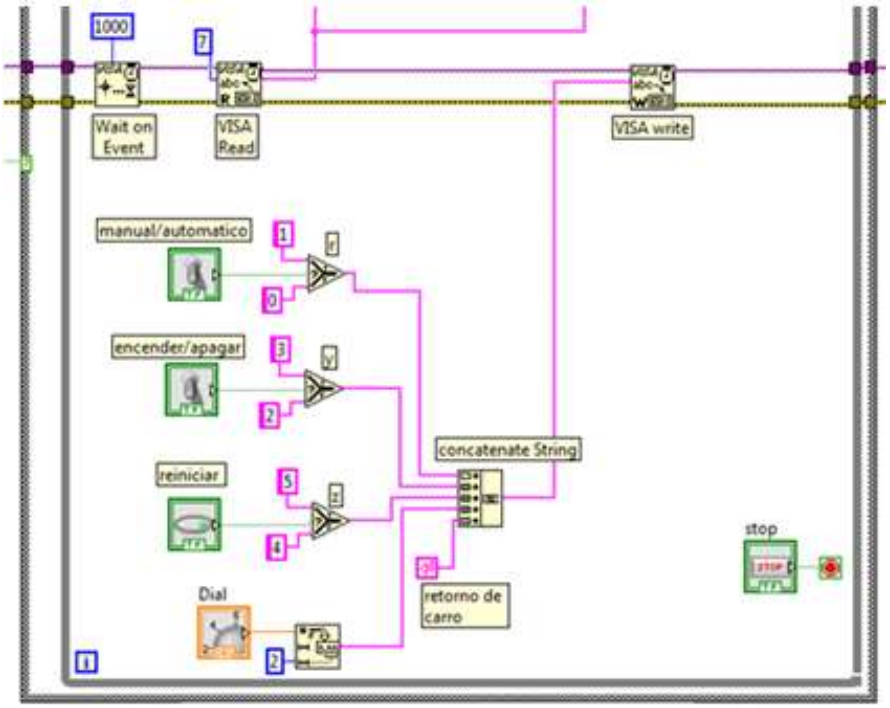


Figura 5.34 Diagrama de bloques completo para
La escritura de datos en el puerto serie.

5.4. Descripción del funcionamiento del panel de control

Una vez concluida la programación en diagrama de bloques. En el panel frontal de LabView aparecen todos los controles e indicadores empleados, y es hora de modificar sus parámetros, como el nombre, la escala, y las acciones mecánicas, entre otros. También se pueden agregar etiquetas para indicar las unidades apropiadas, adornos, etc.

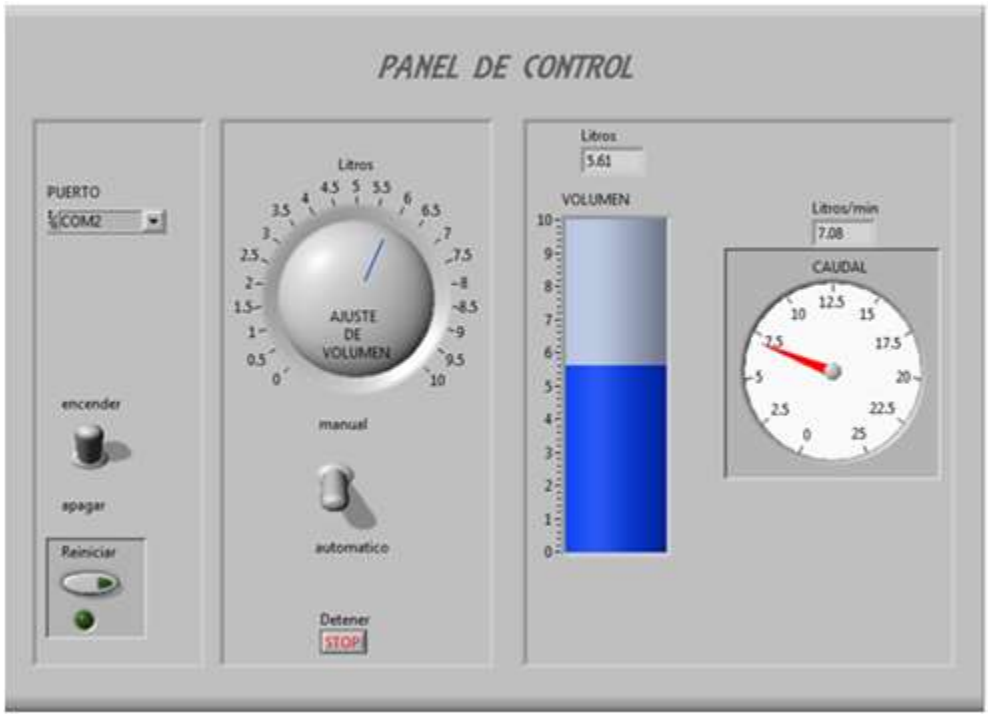


Figura 5.35 Panel de control del sistema.

En el panel de control se han puesto todos los controles en la parte izquierda, los indicadores se encuentran en la parte derecha, y se agregaron indicadores numéricos con el fin de tener una mayor precisión en la lectura de datos.

La función principal del panel virtual es controlar el volumen del líquido que pasa a través del sensor de flujo, esta tarea se complementa con los indicadores que monitorean y muestran en tiempo real, los valores del volumen y caudal. De este modo el operador del sistema podrá tener referencias validas que le ayuden a actuar sobre el sistema.

Para utilizar el panel de control, antes que nada se debe seleccionar el puerto serial que se utilizará para comunicarse con el módulo transmisor. Para ello se utiliza el elemento del panel de control con título *PUERTO*. Al darle clic se despliega una lista con todos los puertos disponibles en la computadora en donde se ejecuta el programa, de aquí se elige el COM que corresponde al puerto de comunicación serial del sistema, y que se ha configurado previamente.

Figura 5.36 Menú "PUERTO". Despliega los
COM disponibles en la computadora.

El panel de control permite encender y apagar manualmente la bomba de agua empleada como actuador que controla el flujo del líquido, mediante un control ON/OFF. Este control es ejecutado a través de un *switch virtual*, con estados lógicos de *encender/apagar*.

Figura 5.37 Control "encender/apagar", del panel de control.

Para que el switch virtual "*encender/apagar"* funcione, es necesario habilitarlo con el switch "*manual/automático"* que se muestra en la figura 5.37*,* este último tiene que estar en el modo manual. Si se encuentra en modo automático, el switch "*encender/apagar"* no causará ningún efecto sobre el sistema.

Figura 5.37 Control "manual/automático", del panel de control.

Cuando el switch "*manual/automático"* se encuentra en modo "*automático"*, el sistema toma el valor numérico del control "*ajuste de volumen"* y lo manda al microcontrolador, en donde el volumen leído por el sensor, se compara contra este parámetro.

Figura 5.38 Control "ajuste de volumen", usado para fijar un valor de volumen.

Si el volumen leído en el sensor es menor que el parámetro establecido por el control "*ajuste de volumen*" de la figura 5.38, el sistema mantiene encendida la bomba con el fin de permitir que el volumen del líquido alcance el valor establecido por el control. Una vez que el volumen leído y el volumen establecido por el control "ajuste de volumen" son iguales, el sistema apaga la bomba de agua. De este modo el sistema deja pasar solo el volumen establecido por el control "*ajuste de volumen*" de forma automática.

Si se desea reiniciar el sistema para comenzar una nueva medición del volumen, se utiliza el botón "*reiniciar*", éste botón tiene la acción mecánica de un *push butoon,* y solo actúa mientras se mantenga presionado. Al presionarse el botón se manda un comando hacia el microcontrolador, que hace que el contador con el que se mide el volumen se reinicie, permitiendo iniciar una cuenta nueva en la medida del volumen de líquido con el que se esté trabajando.

Figura 5.39 Botón para reiniciar el conteo de volumen.

Cuando el programa presenta algún problema en su ejecución, o si por alguna razón se interrumpe la comunicación en el puerto serial, se debe detener su ejecución con el botón "*Detener*" del panel de control. Al presionarlo la ejecución de programa se detiene, permitiendo solucionar los posibles problemas que causaron el disturbio.

Figura 5.40 Botón para detener la ejecución del programa.

Finalmente en el panel de control se muestran los indicadores de *nivel de líquido* y de *caudal,* estos indicadores son gráficos, y permiten al usuario visualizar en tiempo real los valores de volumen, y de caudal en el sistema.

Figura 5.41 Indicador de volumen.

El indicador *de "volumen"* muestra con un gráfico en forma de tanque, el avance del volumen que pasa por el sensor de flujo. Es importante saber que este indicador representa el volumen que ha pasado por el sensor. Y que no es el nivel del líquido en el contenedor que se emplea.

El indicador solo muestra el volumen que pasa por el sensor desde que se inicia la cuenta, por lo que no se puede saber qué cantidad de líquido había previamente en el contenedor que se esté empleando.

Por último el indicador "*Caudal"* muestra la velocidad a la que se mueve el volumen dentro del sensor de flujo. La escala máxima que se presenta es de 25 [L/m], ya que esta es la máxima velocidad a la que el sensor puede medir volumen. Si se presenta algún cambio en la velocidad del volumen, el indicador lo mostrara casi al instante.

Figura 5.42 Indicador de flujo.

Capítulo 6

Comunicación del PIC con Labview

6.1. Envío de datos del PIC a LabView

Para establecer una comunicación por puerto serial con el panel de control del sistema. El microcontrolador PIC utiliza el módulo de comunicación serie ***USART*** (Transmisor-Receptor Serie Síncrono-Asíncrono Universal), o también conocido como ***SCI*** (Interfaz de Comunicación Serie), que permite la comunicación con la computadora en modo *full-duplex* asíncrono.

Para usar el módulo *USART* con el compilador CCS y el microcontrolador PIC, es necesaria una configuración genérica en la directiva de cabecera:

#USE RS232(opciones)

En donde "*opciones"* configura distintos parámetros del *USART*, tales como: velocidad de transmisión, pins utilizados, paridad, etc. Las opciones de configuración para el módulo *USART* empleadas se muestran en la tabla 1.6.

Opción	Parámetro que establece
BAUD=X	Velocidad de Baudios.
XMIT=pin	Pin de transmisión
RCV=pin	Pin de recepción

Tabla 1.6 Opciones de configuración del módulo USART.

Las opciones de configuración que se colocan dentro de los paréntesis, deben ser separadas por comas. En el siguiente ejemplo se observa que la velocidad de baudios se establece en 9600, y que la transmisión, y recepción de datos son asignadas a los pines 6 y 7 respectivamente del puerto C del microcontrolador.

#USE RS232(BAUD=9600, XMIT=PIN_C6, RCV=PIN_C7**)**

Para que el módulo *USART* funcione correctamente, la directiva "#USE RS232( )" debe ser definida después de la directiva que configura al reloj de oscilación "#USE DELAY( )". Como se muestra en el siguiente ejemplo.

```
#INCLUDE<16f877.h>
#FUSES HS,NOWDT
#USE DELAY (CLOCK=20M)   //directiva del reloj de oscilación
#USE RS232(BAUD=9600,XMIT=PIN_C6,RCV=PIN_C7)   //directiva de USART
```

En el ejemplo se muestra la configuración del módulo *USART* a través de la directiva "#USE RS232()". Después de haber escrito las directivas de cabecera que se muestran, se pueden usar las funciones para escribir en el puerto serie. Como se verá a continuación.

El compilador CCS cuenta con funciones que facilitan la tarea de enviar y recibir datos a través del módulo *USART*. Las funciones utilizadas para escribir datos en el puerto de comunicación serial son:

putc(dato) o **putch(dato)**

Estas funciones son equivalentes, y envían el carácter especificado en su parámetro "*dato*" hacia el puerto serial. El parámetro "*dato"* es un numero decimal con rango de 0 a 255 y tiene asignado un carácter ASCII.

puts(Cadena)

Envía una cadena de caracteres. El parámetro "*Cadena*" es un arreglo de una dimensión, que contiene al "0" en la última posición. El carácter "0" (48 en ASCII) indica a la función, el final de la cadena.

La función "*puts(Cadena)"* envía uno a uno los caracteres contenidos en su parámetro "*Cadena*", y al finalizar envía el carácter de retorno de carro "*RETUR*" (13 en ASCII).

prinf("Cadena",valor)

Esta función envía una cadena de caracteres en la que se pueden concatenar datos, el parámetro *"valor"* es la variable que se desea concatenar a la cadena, y puede ser cualquiera de los tipos de datos que maneja el compilador CCS, char, int, float, etc. La posición en la que se desea concatenar la variable se indica con un "%n", en donde, "n" es el índice correspondiente a cada tipo de dato.

Para enviar los datos de volumen y caudal, al panel virtual implementado en LabView, se utiliza el programa desarrollado con anterioridad, en el subcapítulo 3.3, el cual se encarga de medir el volumen, y calcula el gasto que pasa a través del sensor de flujo.

El volumen se almacena en la variable “L”, y el gasto en la variable “v”. Estos son los datos que deben ser enviados al panel de control a través del puerto de comunicación serial, siguiendo el formato establecido en el capítulo 5.2 durante el desarrollo del programa para la lectura por puerto serial en LabView, el cual se muestra en la figura 6.1.

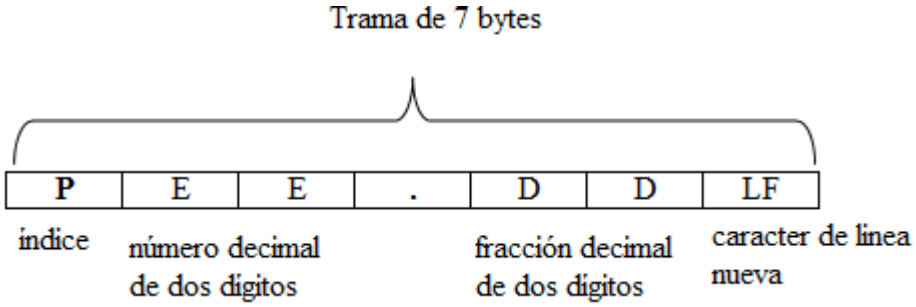


Figura 6.1 Formato para el envío de datos del PIC, hacia LabView.

LabView espera recibir una cadena de datos (trama) de 7 bytes, en la que la primera posición es asignada al índice, quien discrimina si el dato enviado corresponde al volumen, o al caudal. El índice puede tomar los valores “0” o “1”.

Después del índice, los caracteres de la posición 2 a la 6 (parámetro E E .D D) de la figura 6.1, representan el dato de volumen, o caudal. Por último el carácter de nueva línea “LF” (10 en ASCII) debe colocarse en la última posición de la cadena de caracteres, ya que le indica a LabView el final de la trama de datos.

Una vez que la trama de datos es enviada a la computadora, el programa desarrollado en LabView compara el valor del índice. Si el índice toma el valor “0”, la aplicación en LabView envía el dato al indicador de volumen, y cuando es “1” lo envía al indicador de caudal.

El siguiente segmento de código envía por el puerto serie la trama de datos correspondiente al volumen.

```
printf("0");                    // escribe el  índice 0
    printf("%0.2f",L);  //escribe el dato de volumen L
       putc(10);      //carácter de avance de línea "LF", indica fin de la trama
```

En donde:

prinf(“0”) escribe el carácter “0”, que corresponde al índice que representa al volumen.

printf(“%0.2f”,L) escribe el valor de la variable L, en la que se almacena el volumen leído, con solo 2 dígitos después del punto decimal.

putc(10) Se encarga de escribir el carácter ASCII cuyo valor decimal corresponde a 10, es decir el carácter de nueva línea “LF”, que indica el final de la trama de datos.

Para enviar el dato de gasto por el puerto serie, se emplea el siguiente segmento de código:

```
printf("1");                //índice “1”
   printf("%0.2f",v);     //dato de caudal
      putc(10);      // carácter “LF” que indica fin de la trama
```

En donde:

printf(“1”) escribe un índice igual a “1”, con el que se indica que el dato corresponde al caudal.

Printf(“%0.2f”v) escribe el valor de la variable “v”, con la que se mide el caudal dentro del sensor, empleando 2 dígitos después del punto decimal.

putc(10) escribe el carácter “LF”(10 en ASCII), que indica el final de la trama de datos.

Los dos segmentos de código anteriores, deben ser ejecutados en el bucle infinito *while* para que el envío de datos a través del puerto serial sea constante, como se verá posteriormente.

6.2. Recepción de datos enviados de Labview, al PIC

Con la ayuda del panel de control virtual el operador del sistema envía comandos al microcontrolador PIC, quien se encarga de recibirlos para posteriormente decodificarlos, y ejecutar las instrucciones correspondientes.

Para poder realizar la lectura del puerto de comunicación serial, se utiliza el módulo *USART* del microcontrolador, que como se ha visto anteriormente, se configura a través de la directiva “#USE RS232()”.

Después de haber definido las directivas de cabecera, y configurado el módulo *USART*, es necesario incluir el archivo “stdlib.h”, como se muestra en el siguiente segmento de código.

```
#INCLUDE<16f877.h>
#FUSES HS,NOWDT
#USE DELAY (CLOCK=20M)
#USE RS232(BAUD=9600,XMIT=PIN_C6,RCV=PIN_C7)  //configuración del  módulo USART
#INCLUDE <stdlib.h>  // archivo.h
```

El archivo "*stdlib.h*" incluido en las directivas de cabecera, permite el uso de funciones para poder manipular datos que llegan a través del puerto de comunicación serial. Las funciones incluidas en el archivo "*stdlib.h*" que se utilizan en el desarrollo del proyecto se describen a continuación:

gets(arreglo);

Recibe caracteres del puerto serial, hasta que se recibe el carácter de retorno de carro "CR" (13 en ASCII). Con los caracteres recibidos genera un "**arreglo**". No hay ECO.

x= atof(arreglo);

Convierte la cadena de caracteres que se almacenó en el "**arreglo**" en un número con formato punto flotante.

LabView en vía hacia el PIC una trama de datos con de 9 bytes de longitud, con el formato que se indica en la figura 6.2.

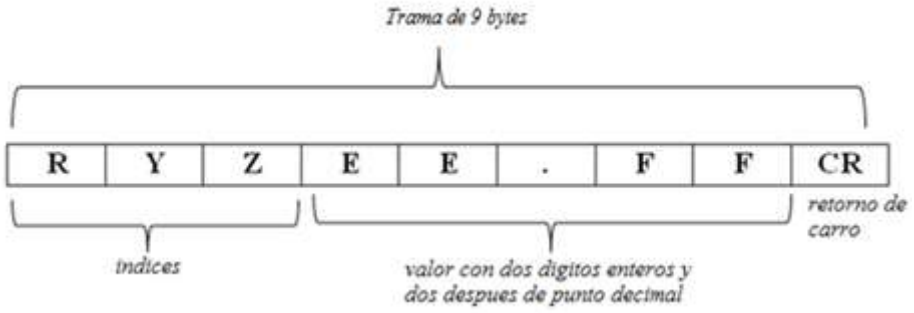


Figura 6.2 Formato con el que LabView envía los datos hacia el PIC.

Los tres primeros caracteres de la cadena anterior (R, Y, Z), corresponden a los índices que identifican la instrucción que el microcontrolador ejecutará. Los caracteres de la posición 4 a la 8 (parámetro "E E . F F"), corresponden al dato de ajuste de volumen que el operador del sistema fijará a través del control ("*ajuste de volumen"*) en el panel virtual, por último se indica el final de la trama de datos mediante el carácter de retorno de carro "CR".

Para recibir la cadena de caracteres que llega a través del puerto serie, se emplea la interrupción por recepción de datos RDA, y la función para generar arreglos de caracteres "*gets(arreglo)*", como se muestra en el siguiente segmento de código.

```
char cadena[9];
#INT_RDA          // interrupción por recepción de datos RDA
void interrups_rda(){
gets(cadena);        // "se almacena la trama de datos en un arreglo de
}                     //   de caracteres"
```

En este segmento de código se utiliza la interrupción por recepción de datos RDA para recibir la cadena de 9 caracteres, que representa un comando enviado por el usuario a través de LabView. La cadena es almacenada en un arreglo llamado "*cadena*", con la ayuda de la función *gets()*.

Posteriormente los índices son separados mediante la asignación directa de los elementos 0, 1, y 2 de la cadena, con las variables r, y, z, las cuales guardan el valor de los índices, que son usados para identificar el comando a ejecutar. Como se muestra a continuación.

```
   r=cadena[0];              // "separación de los índices para
   y=cadena[1];                 // identificar el comando "
   z=cadena[2];

for(i=0;i<=4;i++){  // separación del dato de volumen
 arr[i]=ch[i+3];
 }
  x=atof(arr);   //convierte la cadena en un numero punto flotante
```

En donde los índices R, Y, Z, de la figura 6.2, son separados del resto de la trama de datos. El dato de ajuste de volumen que se encuentra localizado en las posiciones de 4 a 8 de la trama, es separado y almacenado en un arreglo mediante el ciclo *for()*.

El arreglo generado en el ciclo *for(),* es convertido a un número punto flotante a través de la función "*atof()*", y después es guardado en la variable "x", este número será usado para fijar el valor de volumen que tiene que pasar de un contenedor a otro cuando el sistema opera de forma automática.

6.3. Descripción del software implementado en el PIC

El programa implementado en el microcontrolador PIC, es capaz de leer e interpretar los pulsos enviados por el sensor de flujo, calcula el caudal, y envía los datos a través del puerto de comunicación serial hacia Labview.

También permite que el microcontrolador reciba comandos desde el panel de control en Labview. Los comandos están contenidos en una cadena de caracteres que es interpretada para ejecutar las instrucciones correspondientes.

Los comandos permiten controlar el flujo de líquido dentro del sistema, a través de un modo manual, en el que el operador decide el momento de detener o reanudar el flujo del líquido. O a través de un modo automático, en donde el operador fija un valor para el volumen que debe pasar de un contenedor a otro, antes de que el programa detenga el flujo.

Para implementar lo antes descrito, se desarrolló el siguiente programa en el compilador CCS. Y el funcionamiento de cada segmento de código se describe a continuación:

Primero se definen la cabecera del programa, con las directivas de pre procesado y los archivos ".c" que contiene las funciones, y los ".h" que contienen las librerías usadas en el programa.

```
#include <16f877.h>
#fuses HS,NOWDT  // fusibles empleados
#use delay(clock=20M)  // frecuencia del cristal
#include <stdlib.h>    //incluye librerías para manejo de cadenas de caracteres
#use rs232(baud=9600,xmit=pin_c6,rcv=pin_c7)     // configura el módulo USART
#use fast_io(B)     //configuración rápida del puerto B
#include <lcd.c>  // incluye funciones para manejo del LCD
```

Después se declaran todas las interrupciones, y se definen las rutinas que ejecutarán. Primero la interrupción por recepción de datos RDA.

```
///interrupción RDA/////

char ch[9];     //arreglo donde se guardan los datos enviados desde Labview .
#INT_RDA       //directiva de la interrupción.
void interrups_rda(){  // nombre a la interrupción.
gets(ch);  //guarda los datos recibidos en un arreglo.
}
```

En la rutina de interrupción anterior se reciben y almacenan los datos que llegan a través del puerto serial, en un arreglo de caracteres con 9 elementos, llamado “ch”.

A continuación, se declara y se define la interrupción usada para medir el volumen que pasa por el sensor de flujo.

```
/// Interrupción RB0-///
int16 cont=0,R=0;   // se declaran e inicializa a 0, dos variables de 16 bits
#INT_EXT            // directiva de la interrupción por evento externo
void ext_isr(){     //se asigna nombre a la interrupción y se especifica como vacía
cont=cont+1;   // cont cuenta los pulsos del sensor
R=R+1;          // variable auxiliar usada para calcular caudal
}
```

En la rutina de interrupción por evento externo RB0, la variable de 16 bits “cont”, se incrementa a medida que el microcontrolador detecta los pulsos enviados al pin RB0 por el sensor de flujo. La cuenta representa el volumen que ha pasado por el sensor, ya que cada pulso equivale a un mililitro.

El contador “R”, se incrementa a la par con el contador de volumen “cont”, pero “R”, será reiniciado después de un intervalo de tiempo definido por el TIMER0, permitiendo obtener la relación de “*volumen/tiempo*”, o gasto volumétrico.

Para generar el intervalo de tiempo que se utiliza en el cálculo del caudal, se emplea la interrupción del TIMER0.

```
// Interrupción del TIMER0//

int j=0;
float v;
#int_TIMER0       //directiva de interrupción del TIMER0.
void timer0_isr(void){      //nombre de la interrupción.
j=j+1;             //contador para generar intervalo.
if(j==50){  // se genera intervalo de 500[ms].
v=R*0.12;     // se calcula caudal en Litros/minuto.
j=0;      // se reinicia contador para generar intervalos de 500[ms].
R=0;  //se reinicia contador de pulso.
}
set_timer0(0x3d); // carga  en el  registro del TIMER0  para generar intervalo 10[ms].
}      //fin de la rutina de interrupción.
```

El cálculo de caudal dentro de la interrupción del TIMER0, se hace siguiendo el procedimiento descrito en el subcapítulo 3.3 correspondiente al tema, y en el cual se describe la configuración del *timer,* y el factor de proporción utilizado para convertir el valor de caudal en una razón de *litros/minuto.*

Para utilizar la interrupción por evento externo RB0, es necesario definir al pin 0 del puerto B como una entrada digital, y habilitar la resistencia de *pull-up*, esto permitirá que el microcontrolador pueda recibir los pulsos que el sensor de flujo envía. Como se indica en el siguiente segmento de código.

```
/// ////Configuración del pin de entrada///////
set_tris_B(0x01);     // define pin 0 de puerto B como entrada digital
port_b_pullups(TRUE);    //habilita resistencia de pull-up
```

Para que las interrupciones RDA, RB0, y la del TIMER0 funcionen, es necesario habilitarlas dentro de la función principal *main().* En el siguiente segmento de código se muestra cómo se hizo.

```
/// Función principal ///
void Main(){     //función principal es vacía.

enable_interrupts(GLOBAL);  //habilita interrupciones globales.
setup_TIMER_0(RTCC_INTERNAL|RTCC_DIV_256);   //configura interrupción del TIMER0.
set_timer0(0x3d);              // carga valor al registro del TIMER0.
enable_interrupts(INT_TIMER0);  // habilita interrupción particular del TIMER0.

enable_interrupts(int_rda); // habilita interrupción particular de recepción de datos.
enable_interrupts(int_ext); // habilita la interrupción por evento externo RB0.
ext_int_edge(L_TO_h);  // la interrupción RB0 es por flanco de subida.
```

El TIMER0 es configurado usando un *preescaler* que divide la frecuencia de conteo en 256, y el valor que se carga en el registro ha sido calculado previamente para que la interrupción se genere cada 10[ms].

Antes de entrar al bucle infinito *while()* deben ser definidas todas las variable que se utilizarán dentro de él, y debe inicializarse el módulo LCD.

Por el pin 4 del puerto B en el microcontrolador, se enviará la señal de control al actuador, que interrumpe el flujo de líquido a través del sensor. Por esta razón cada vez que inicie la ejecución del programa, se debe asegurar que se encuentre apagado.

```
//////////////////////////función principal ////

void main(){            //inicio de función principal
char arr[7];           // se define un arreglo de caracteres que contendrá 7 elementos
int r=0,y=0,z=0,i;         // se definen las variables para los índices.
float L=0,x=0;            // variables que contendrán datos de volumen

output_low(pin_b4);           // pin 4 de puerto B apagado al iniciar el programa
lcd_init();      // inicializa modulo LCD
```

Las variables enteras: r, y, z, son empleadas para guardar los índices con los que se decodifican los comandos enviados desde el panel de control virtual en LabView. Y las variables punto flotante, "L", y "x", almacenan la cuenta de volumen, y el ajuste de volumen que se hace desde el panel de control, respectivamente.

En el bucle infinito "*while(true)*", se realiza la decodificación de los comandos enviados desde el panel de control, con la ayuda de los índices que son separados en el siguiente segmento de código, y con la función "*atof(arreglo)*" para convertir cadenas de caracteres a un número punto flotante.

```
while(true){   // bucle infinito

    L=cont/1000.0;                  //  conversión de mililitros contados, a Litros
    r=ch[0];          //  índice en posición 0, de la cadena de comandos recibida
    y=ch[1];          //  índice en posición 1, de la cadena de comandos recibida
    z=ch[2];          // índice en posición 2, de la cadena de comandos recibida

  for(i=0;i<=4;i++){   //  "se separan los caracteres de la posición 4 a la 7 de la
   arr[i]=ch[i+3];       //     cadena de comandos. Y se guardan en el arreglo"
   }
    x=atof(arr);        // se convierte el arreglo a un número punto flotante
```

Cuando un comando es enviado desde el panel de control implementado en LabView, la cadena de datos que contiene dicho comando es separada. En ella se encuentran los índices que determinan la acción a ejecutar por el microcontrolador.

Si el valor en ASCII del índice "r" es 49, se entra en modo de control manual. En donde el índice "y" define dos estados lógicos en el pin de salida RB4 (pin 4 del puerto B): "*encendido*" o "*apagado"*.

En el pin RB4 del microcontrolador se genera la señal que enciende o apaga el actuador que controla el flujo de líquido en el sensor. Si el índice r=49, y el índice y=51, el actuador está encendido. Si el índice r=49, y y=50 el actuador se apaga.

El estado lógico del actuador, y el modo en el que opera, se despliegan en el LCD implementado en circuito de instrumentación del sistema con la ayuda del siguiente segmento de código.

```
if(r==49){      //  modo  manual

          if(y==51){                    // encendido
           output_high(pin_b4);
           printf(lcd_putc,"\n\rON  MANUAL");
           }
            else{                          //apagado
               output_low(pin_b4);
             printf(lcd_putc,"\n\rOFF  MANUAL");
             }
```

Cuando el sistema opera en modo automático, el índice "r= 48". En este modo de operación la variable "L", que mide el volumen que pasa por el sensor, es comparada contra la variable "x", que almacena el valor de volumen que el operador del sistema fija a través del panel de control, y que corresponde al volumen que debe pasar por el sensor antes de interrumpir el flujo del líquido.

Si el valor de volumen que ha pasado por el sensor, es menor al volumen fijado por el usuario, el actuador esta encendido. Cuando los dos valores se igualan, el actuador se apaga. Como se muestra en el siguiente segmento de código.

```
if(r==48){        //automático

  if(L<=x){  //compara  volumen leído contra el volumen fijado por el control.
     output_high(pin_b4);  //actuador encendido

          }
     else                          // si L=x
       output_low(pin_b4);   //actuador apagado

          }
```

Cuando se requiere volver a iniciar una nueva medición de volumen, la instrucción es mandada desde el panel virtual. El comando correspondiente asigna al índice z el valor de 53, que es comparado en el microcontrolador. Cuando la condición se cumple, el contador de volumen "*cont*" y la variable con la que se calcula el caudal ("R"), se reinician a cero. Esto permite volver a comenzar la cuenta de volumen. Lo anterior se muestra en el siguiente segmento de código.

```
if(z==53){       //instrucción de reinicio
      cont=0;  // reinicia cuenta de volumen
      R=0;       // reinicia cálculo de caudal
      delay_ms(1000);   // tiempo de guarda
      }
```

Después de haber reiniciado la cuenta de volumen, se espera un intervalo de tiempo de 1 [s], antes de comenzar a contar nuevamente.

Finalmente, los datos de volumen leído, y caudal calculado, son enviados a través del puerto de comunicación serial hacia LabView, para ser desplegados en los indicadores correspondientes.

En la pantalla LCD, que forma parte del circuito de instrumentación del sistema, se despliega el valor de volumen que pasa por el sensor, y el volumen que el usuario fija cuando el sistema opera de modo automático. El siguiente segmento de código realiza lo antes descrito.

```
lcd_gotoxy(1,1);        // inicia el despliegue de datos en las primeras posiciones del LCD.
        printf(lcd_putc,"L=%f[L] ",L);          // muestra en el LCD el volumen
       printf(lcd_putc,"\n\rb=%0.2f ",x);   //muestra el volumen fijado por el usuario

   ////////////manda volumen/////////
       printf("0");              // manda índice 0 a labview.
       printf("%0.2f",L);    //manda dato de volumen a labview.
        putc(10);           //caracter de avance de linea "LF", indica el termino de dato.

    ////////////manda caudal////////
       printf("1");             //manda índice  1 a labview.
       printf("%f",v);       //manda dato de caudal a labview.
       putc(10);           // carácter LF, indica  fin de dato.
```

Los datos que se envían hacia LabView, son identificados con el índice “0” o “1”. Cuando el índice tiene un valor de “0” el dato corresponde al volumen, cuando es “1”, el dato es de caudal, y Labview se encarga de desplegarlos en los indicadores adecuados.

El programa que el microcontrolador usa, está compuesto por todos los segmentos de código anteriores. Este programa se ha verificado mediante una simulación del sistema, antes de su implementación física en el microcontrolador PIC16F877. Como se verá en el siguiente capítulo.

Capítulo 7

Resultados

7.1. Simulación del funcionamiento del sistema

Como parte de los resultados, se verifica que la interacción entre el programa que utilizará el microcontrolador, y la aplicación para el panel virtual desarrollado en Labview funcionen correctamente, a través de una simulación en ISIS PROTEUS. La simulación permite detectar errores, y verificar el desempeño del sistema antes de ser implementado físicamente. Para llevar a cabo la simulación, se alambró el circuito que se muestra en la figura 7.1, en un archivo de PROTEUS.

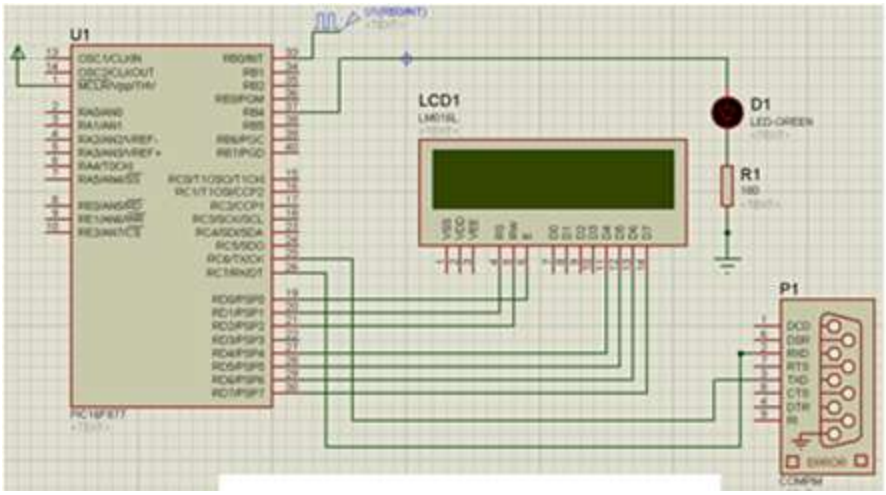


Figura 7.1 Simulación del circuito transmisor en ISIS PROTEUS.

Los componentes de PROTEUS que se emplearon en la simulación son: *PIC16F877, RES, LM016L, LED-GREN, COMPIM.*

El elemento de PROTEUS, *COMPIM*, permite la simulación de un puerto de comunicación serial. Para usar este elemento se requiere de un programa emulador de puertos virtuales instalado previamente, que permite la conexión entre ISIS PROTEUS y LabView.

En el circuito de la figura 7.1 se observan algunos elementos que simulan conexiones físicas dentro del sistema de medición de flujo. Como el puerto de comunicación serial (COMPIM), y la señal de entrada que el sensor de flujo enviará al microcontrolador por el pin RB0 (pin33).

En el led conectado en el pin RB4 (pin37) del microcontrolador, se presenta la señal de control que se encargara de encender, y apagar la bomba de agua que controla el flujo en el sistema real.

En la simulación se aprecia el comportamiento del sistema cuando el usuario interactúa con él, a través del panel de control. La señal del sensor de flujo es simulada con un generador de señales de reloj en *PROTEUS*. El valor de la cuneta de pulsos se transforma a unidades de volumen, y es enviado hacia LabView para ser mostrado en un indicador de volumen. Como se observa en la figura 7.2.

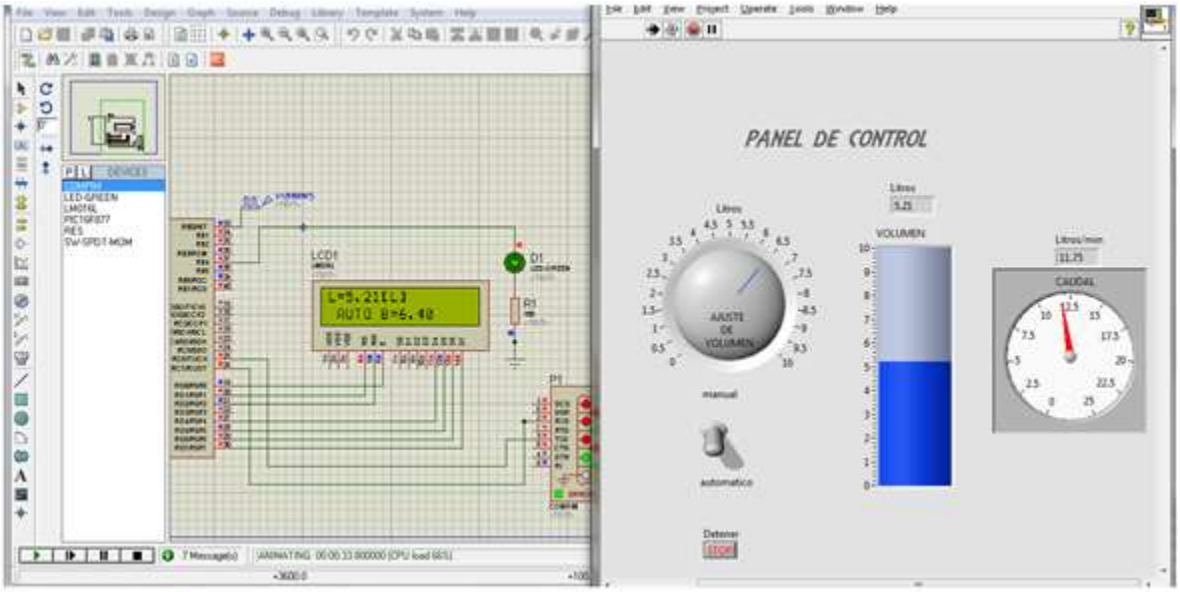


Figura 7.2 Simulación del circuito a través de PROTEUS y LabView.

A través de esta simulación es posible verificar el comportamiento del sistema, y se comprueba que el microcontrolador ejecuta las instrucciones que el operador le indica a través del panel virtual.

En el panel virtual, en el lado derecho de la figura 7.2, se observan los indicadores para las variables de volumen, y caudal, que el microcontrolador envía, estos indicadores sirven para que el operador pueda monitorear ambas variables, y si es necesario tome alguna medida de control.

También se observan los controles con los que el usuario podrá operar el sistema. El funcionamiento detallado del panel de control se describe en el capítulo 5 en la parte correspondiente a este tema.

La simulación nos brinda un resultado muy aproximado del comportamiento del sistema, pero no incluye la conexión inalámbrica, que es parte esencial del prototipo que se ha desarrollado. Por lo que a continuación se describe y se muestra la implementación física en el sistema real.

7.2. Implementación física del sistema

El siguiente prototipo de medición, control, y monitoreo de flujo de líquidos, fue ensamblado y probado en el laboratorio de electrónica del **CCADET** (Centro de Ciencias Aplicadas y desarrollo Tecnológico) de la UNAM.

Para implementar el sistema se utilizó un medidor de flujo ultrasónico ***UF25B100*** fabricado por "CYNERGY", el cual fue instrumentado a través de un microcontrolador PIC 16F877, y un módulo LCD. Como se vio en el capítulo 3 de este trabajo.

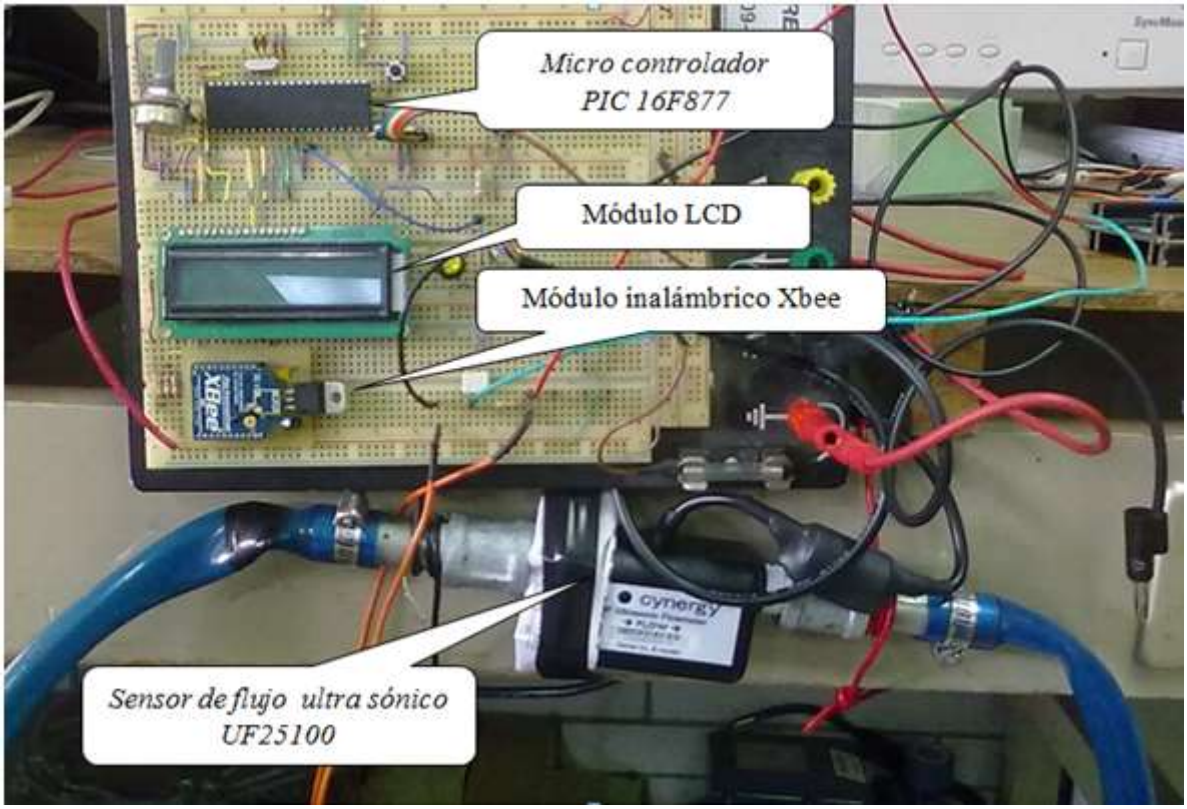


Figura 7.3 Circuito de instrumentación del sensor de flujo.

La figura 7.3 muestra el circuito de instrumentación para el sensor ultrasónico de flujo durante la etapa de pruebas de funcionamiento, esta parte es esencial durante la implementación física del sistema, ya que sirve para comprobar el buen estado físico de los componentes que se emplean en el montaje del circuito impreso PCB.

El programa que utiliza el microcontrolador corresponde al descrito en el capítulo 6. El circuito de instrumentación fue probado físicamente antes de ser montado en un circuito impreso PCB. Durante las pruebas se comprobó que el circuito de instrumentación es capaz

de medir las variables volumen, y caudal de agua, y desplegar la información en el módulo LCD que se encuentra integrado en él, ver figura 7.3.

El sensor de caudal es ultrasónico, y opera bajo el principio de *medición por tiempo de tránsito de ondas ultrasónicas*. La resolución que tiene es de un mililitro, y entregar en su terminal de salida un pulso por cada mililitro de líquido que pasa a través de él.

A diferencia de la medición de volumen que es entregada directamente por el sensor, el caudal se mide de forma indirecta, ya que resulta de la estimación de la cantidad de volumen que pasa por el sensor en cierta unidad de tiempo. La base de tiempo utilizada para el cálculo de caudal es generada internamente por el microcontrolador.

Durante el funcionamiento del sensor, se comprobó que presenta errores en las mediciones si el líquido con el que se trabaja contiene burbujas de aire. Por lo que es indispensable verificar que no existan burbujas en el sistema, de lo contrario se debe realizar una purga.

Para comunicar el circuito de instrumentación con el panel de control virtual implementado en LabView, se usaron dos módulos Xbee serie 1. Los cuales fueron acondicionados, y configurados, conforme al procedimiento descrito en el capítulo 4.

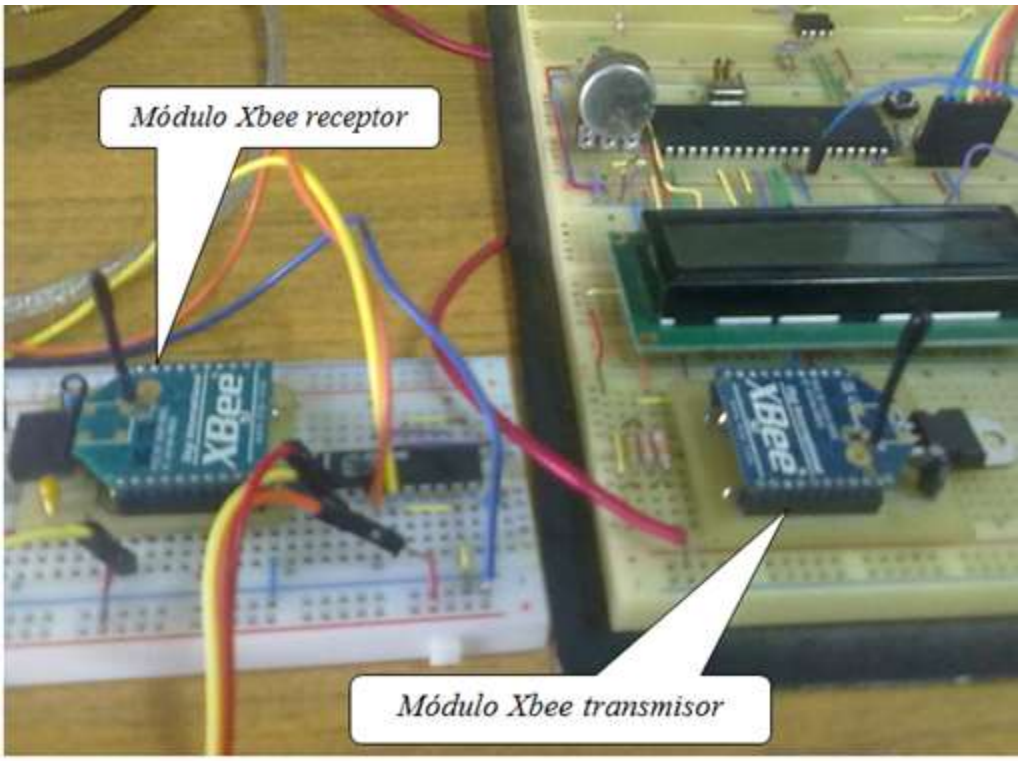


Figura 7.4 Módulos inalámbricos Xbee, transmisor y receptor.

El módulo Xbee transmisor fue integrado al circuito de instrumentación del sensor, usando los debidos adaptadores de nivel de tensión para el puerto de comunicación serie entre el micro controlador PIC y el módulo inalámbrico Xbee, que se vieron en el subcapítulo 4.4.

Para construir la planta, en la cual se monitorea y controla el flujo de agua, se empleó una bomba de agua como actuador del sistema. La bomba de agua es la encargada de detener o reanudar el flujo de agua que pasa a través del sensor, y para controlarla se hace uso de un relevador de estados sólido formado por un TRIAC, y un opto acoplador. La señal de control de encendido y apagado se obtiene a través del pin RB4 (pin 37) del microcontrolador PIC.

El opto acoplador MOC3041 en la figura 7.6 es totalmente necesario, ya que sirve para aislar la señal de control proveniente del circuito de instrumentación con niveles de tensión de 5[Vdc], del circuito de potencia que trabaja con un voltaje nominal de 127 [Vac].

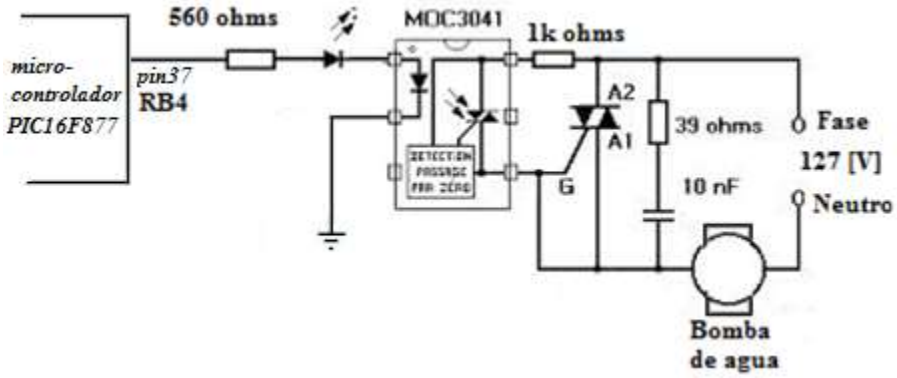


Figura 7.6 Relevador de estado sólido.

El circuito de la figura 7.6 muestra la forma en que el circuito de control es aislado del circuito de potencia a través de un "*foto triac"* contenido dentro del opto acoplador MOC3041. Este circuito permite encender y apagar la bomba de agua que forma parte del sistema, a través del de la señal emitida por el microcontrolador, sin correr el riesgo de una posible falla eléctrica que pudiese dañar al circuito de instrumentación.

El circuito de potencia para encender y apagar la bomba de agua, el módulo inalámbrico Xbee, y el circuito de instrumentación del sensor de flujo, fueron integrados en una sola placa PCB, en la cual se incluyó una fuente de alimentación con tres niveles de voltaje, como se muestra en la figura 7.7.

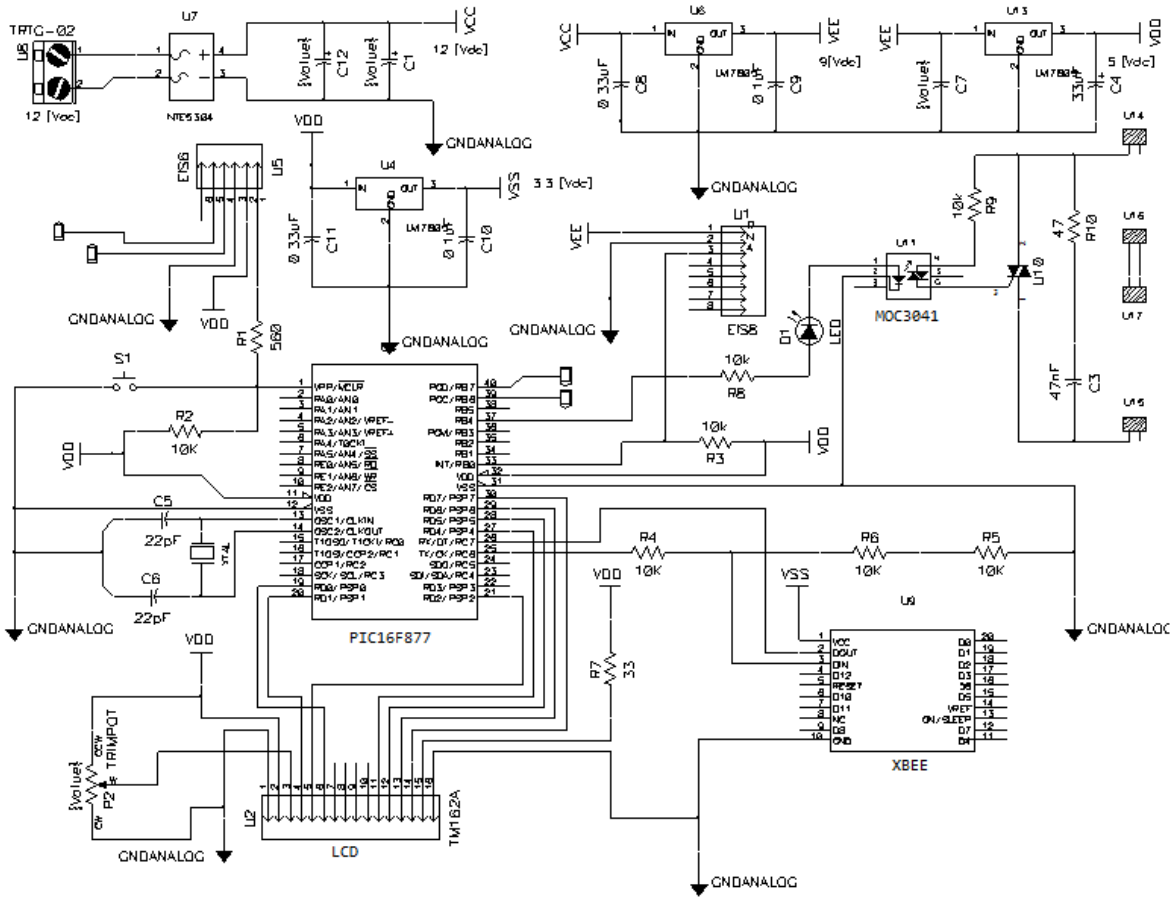


Figura 7.7 Diagrama del circuito de instrumentación.

En la figura 7.7 se muestra el diagrama eléctrico de la palca PCB que se construyó para la implementación física del sistema. En el diagrama eléctrico se aprecia el circuito de instrumentación del sensor, el cual consta del microcontrolador PIC, y el módulo LCD. En la placa PCB se han incluido el relevador de estado sólido, el módulo inalámbrico Xbee, y una fuente de alimentación con tres niveles de voltaje para alimentar los distintos componentes que forman parte del circuito.

El circuito eléctrico de la figura 7.7 debe ser alimentado con 12 [Vac] provenientes de un transformador externo. La señal de entrada es rectificada a través de un puente de diodos, y filtrada con la ayuda de dos capacitores electrolíticos para poder mantener un nivel de tensión constante, de este modo se obtiene un voltaje de entrada de 12 [Vdc] con el que se alimenta a la tarjeta electrónica.

Para que el sensor de flujo ultrasónico funcione debe ser alimentado con un voltaje de 8-20[Vdc], por lo que se ha colocado un regulador de voltaje de 9[Vdc], que es un valor de voltaje dentro de dicho rango para alimentar al sensor.

El microcontrolador, y el módulo LCD, funcionan con un voltaje nominal de 5[Vdc] por lo que un regulador de voltaje de 5[V] se ha conectado en cascada con la salida del regulador de 9[V] para evitar una caída de voltaje mayor, que provocaría que el regulador de 5[V] se caliente.

De una forma similar un regulador de voltaje de 3.3 [V] fue conectado en cascada con la salida del regulador de 5[V] para alimentar el módulo inalámbrico Xbee, que opera con un voltaje nominal de 3.3 [V].

Después de haber alambrado, y verificado el buen funcionamiento del circuito de la figura 7.7 en la tabla de proyectos *protoboart*, se diseñó el circuito impreso PCB con el fin de interconectar todos los dispositivos y elementos electrónicos en una sola placa, la cual sirve como soporte mecánico, de esta forma se obtiene una tarjeta electrónica que es más fácil de instalar dentro de un gabinete, y le da una mejor presentación al proyecto.

Para diseñar el circuito impreso PCB, se utilizó el software editor de circuitos impresos P-CAD, el cual ofrece un paquete con diferentes aplicaciones que contienen distintas herramientas que facilitan la implementación de circuitos impresos.

Para diseñar un PCB con P-CAD, se hace uso de librerías que contiene la información del dispositivo electrónico a interconectar. El software contiene librerías de diversos dispositivos de distintos fabricantes. En caso de no encontrar el elemento o dispositivo requerido dentro del diseño, existen aplicaciones dentro del paquete P-CAD, las cuales permiten crear una representación esquemática del elemento o dispositivo, y asociarla con un patrón para la huellas en la placa de cobre, la cual contiene la forma y las dimensiones reales del dispositivo.

Después de haber definido las librerías a utilizar dentro del diseño, se utiliza la aplicación de P-CAD llamada "*Schematic"* la cual permite crear el diagrama eléctrico, y en el cual se especifican todas las conexiones entre elementos. En la figura 7.7 se muestra el diagrama eléctrico de la placa PCB a construir, y corresponde al esquemático diseñado en la aplicación de P-CAD "*schematic"*.

Una vez creado el esquemático del circuito utilizando los dispositivos correctos, P-CAD nos permite generar una *netlist* (lista de conexiones), la cual vincula las terminales de cada dispositivo con las conexiones correspondientes, de este modo el software le indica al diseñador que pines de los dispositivos empleados deben conectarse entre sí, disminuyendo la posibilidad de cometer errores en las conexiones.

Con el *netlist* generado, éste es importado a la aplicación de P-CAD llamada "*PCB"*, en la que finalmente los componentes incluidos dentro del esquemático son presentados en la pantalla donde se diseñan las pistas de cobre, con la forma y dimensiones reales.

Dentro de la aplicación "*PCB"* de P-CAD, se pueden colocar y distribuir los componentes en la posición que se requiera, para posteriormente usar la herramienta de ruteo que permite generar las pistas de cobre que interconectan a los componentes. En la figura 7.8 se muestra el diseño terminado de la placa PCB a implementar.

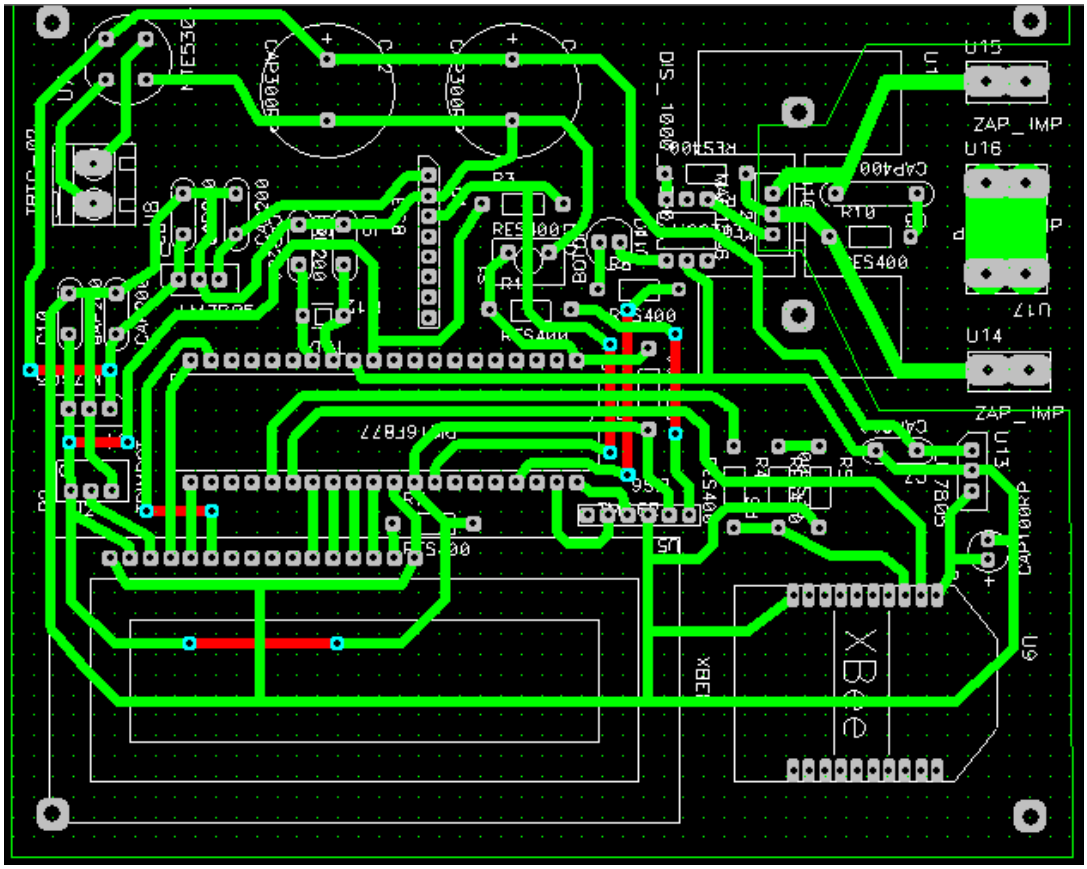


Figura 7.8 diseño del circuito impreso en la aplicación PCB de P-CAD.

El circuito impreso de la figura 7.8, fue revelado mediante una máquina de control numérico (CNC), la cual se encargó de construir las pistas de cobre, y hacer los barrenos (perforaciones) que permiten soldar los componentes sobre la placa de baquelita.

Después de haber revelado la placa, los componentes previamente probados en la tabla de proyectos *protoboart* se colocaron en su lugar correspondiente, al final se obtuvo la placa del circuito impreso que se muestra en la figura 7.9.

La figura 7.9 muestra la placa PCB terminada, con los componentes electrónicos colocados en sus lugares correspondientes, en esta pala PCB o tarjeta electrónica se incorpora el circuito de instrumentación del sensor de flujo, el módulo inalámbrico Xbee, el circuito de potencia formados por el relevador de estado sólido, y la fuente de alimentación del circuito.

La tarjeta electrónica fue construida con una sola capa de cobre, para completar las pistas se utilizaron puentes formados por segmentos de alambre colocados en la parte superior de la placa.

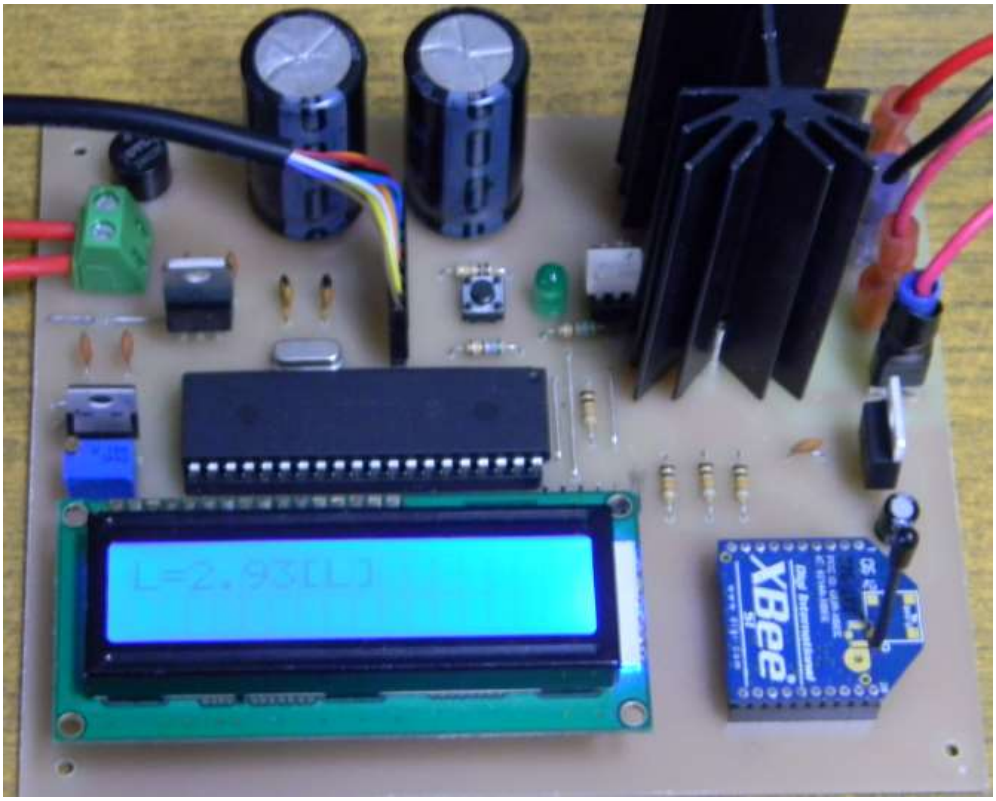


Figura 7.9 Tarjeta electrónica terminada.

En la tarjeta electrónica se han incluido distintos tipos de conectores, que permiten colocar dispositivos y elementos eléctricos externos. Como el transformador de 12[Vac] necesario para alimentar el circuito, el sensor de flujo que se conecta a la placa mediante un conector de 8 pines, y los cables de conexión de la bomba de agua que controla el flujo del líquido en el sistema.

El circuito de potencia formado por el relevador de estado sólido, el cual controla el encendido y apagado de la bomba de agua, y que se ha incluido en la tarjeta electrónica, fue aislado del circuito de instrumentación mediante un opto acoplador MOC3041.

La tarjeta electrónica implementada es capaz de medir el volumen de agua que pasa por el sensor, calcular el caudal, desplegar la información en el display LCD, y mandar los datos al módulo inalámbrico Xbee para ser recibidos por el circuito receptor del sistema, posteriormente los datos son desplegados en el panel de control implementado en LabView.

La tarjeta también tiene la función de recibir los comandos enviados por el operador desde el panel de control, y detener o reanudar el flujo del líquido en el sistema, apagando o encendiendo la bomba de agua con la ayuda del relevador de estado sólido incluido en ella.

Para poder recibir y procesar los datos en la computadora, el módulo Xbee receptor utiliza una interfaz electrónica, la cual consiste en un circuito integrado adaptador de niveles de tensión MAX233, y un cable de conversión de protocolo Serial-USB. En la figura 7.10 se muestra el diagrama eléctrico usado en la aplicación "*schematic"* de PCAD para la fabricación del circuito impreso del módulo receptor del sistema. En él no aparece el cable de conversión Serial-USB ya que éste es un dispositivo externo.

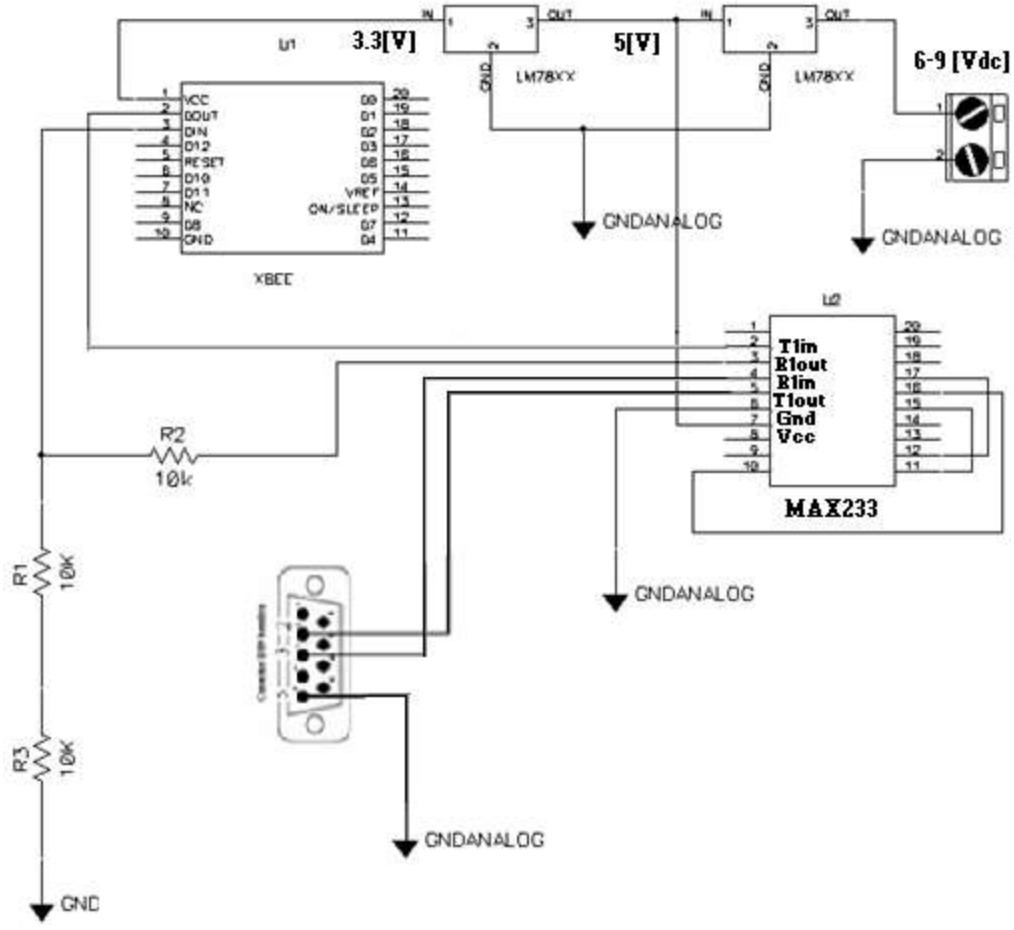


Figura 7.10 Diagrama eléctrico del módulo receptor del sistema.

El módulo receptor del sistema está constituido por dos reguladores de tensión, uno de 3.3 [Vdc] conectado en cascada con la salida del segundo regulador de 5[Vdc] que alimenta al circuito integrado MAX232, el cual se encarga de cambiar los niveles de tensión lógicos empleados por el módulo Xbee, que son de 0 a 3.3 [Vdc], a niveles de 5 a -5 [Vdc], los cuales están dentro del rango de los niveles lógicos de tensión utilizados por las computadoras en el protocolo de comunicación serial.

Cuando el circuito integrado MAX232 envía datos hacia el módulo Xbee, los niveles lógicos de tensión son de 0 a 5 [Vdc], que no pueden ingresarse directamente en la terminal de recepción de datos del módulo Xbee, o podría dañarse. Para lo cual se emplea un adaptador de niveles de tensión construido con un arreglo de resistencias conectadas en serie como se aprecia en la figura 7.10. Dicho arreglo es un divisor de voltaje, el cual ajusta el nivel lógico "1" del integrado MAX232, con nivel de 5[Vdc], a 3.3 [Vdc] que es nivel empleado por el Xbee.

Para construir el circuito impreso del módulo receptor del sistema, se usó nuevamente del software PCAD, la figura 7.10 corresponde al diagrama eléctrico utilizado en la aplicación *schematic* de P-CAD. Después de haber generado la lista de conexiones (*netlist*), con la aplicación "*PCB"* de P-CAD se distribuyeron los componentes electrónicos, y se ruteo su conexión como se muestra en la figura 7.11.

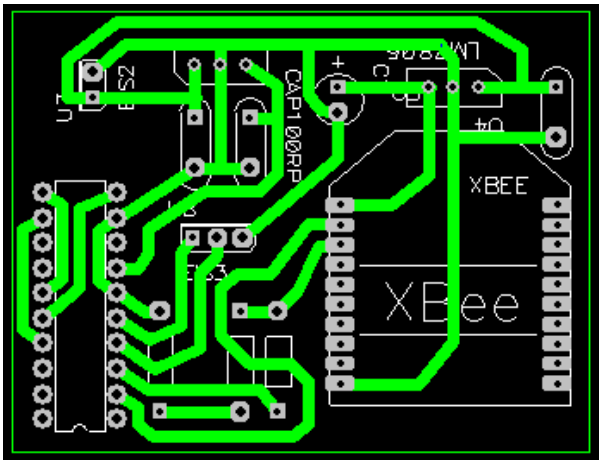


Figura 7.11 Diseño del circuito impreso PCB para el módulo receptor del sistema.

Luego de haber terminado con el diseño del circuito impreso PCB en P-CAD, la placa fue revelada con la ayuda de una máquina de control numérico (CNC) la cual construyó las pistas de cobre, e hizo los barrenos para los pines de los distintos componentes electrónicos, dejando la placa PCB terminada.

Después de haber fabricado el circuito impreso PCB, los componentes fueron colocados y soldados en sus lugares correspondientes, la figura 7.12 muestra la tarjeta electrónica terminada, la cual funcionando como módulo receptor del sistema de medición y control de flujo de líquidos.

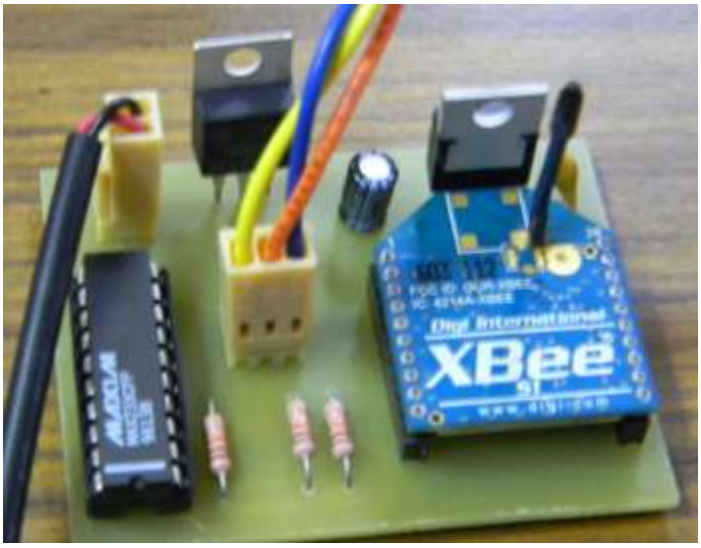


Figura 7.12 Tarjeta electrónica para el módulo receptor del sistema.

En la tarjeta electrónica del módulo receptor del sistema, se incluyeron dos conectores, uno es para conectar una fuente de voltaje externa que alimenta el circuito. La fuente debe ser con niveles de tensión de 6 a 9 [Vdc]. El segundo conector es de tres terminales y se utiliza para un conector DV9 hembra como se indica en la figura 7.10.

Con el conector DV9 hembra el circuito se conecta al cable de conversión de protocolo de comunicación Serial-USB, ya que la laptop en la que se ha implementado el panel de control virtual no cuenta con puerto de comunicación Serial.

El cable de conversión Serial-USB es un elemento externo, y por eso no se incluyó dentro del diagrama del circuito impreso PCB del módulo receptor. Para que el cable de conversión Serial-USB funcione es necesario instalar previamente los controladores correspondientes del dispositivo en la computadora.

El panel de control virtual fue programado e implementado a través de LabView, como se describe en el capítulo 5 correspondiente al tema. Para ejecutar la aplicación hecha en LabView se utilizó una laptop. En la figura 7.13 se aprecia el circuito receptor del sistema conectado a través del cable de conversión Serial-USB a la computadora.

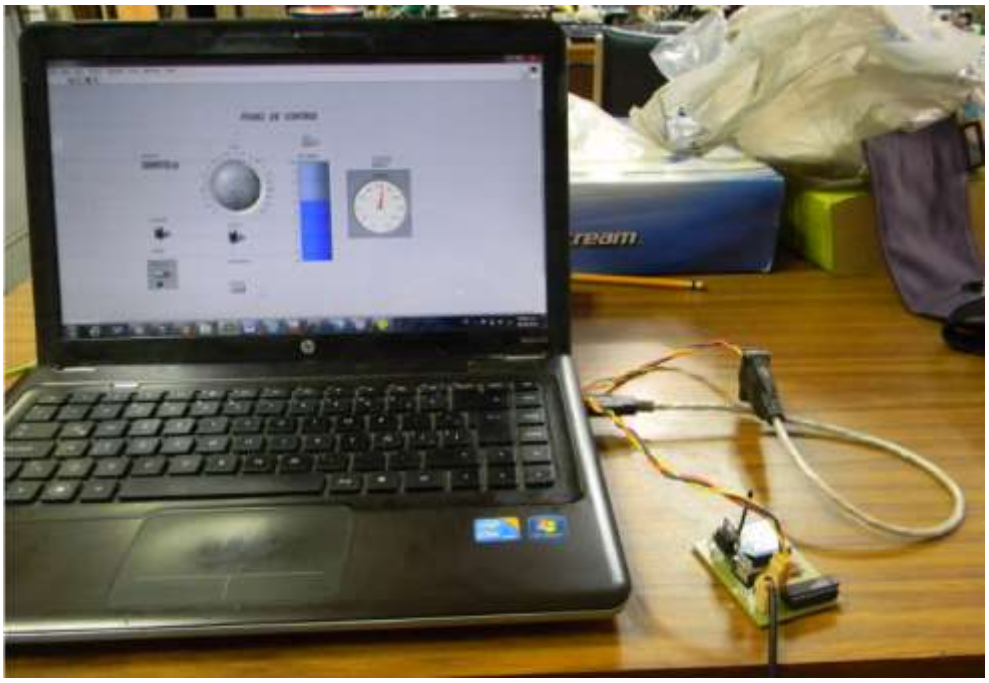


Figura 7.13 Panel virtual de control, implementado en una computadora laptop.

El funcionamiento del panel de control fue puesto a prueba, y se demostró que funciona de forma correcta conforme a los requerimientos establecidos. El panel permite monitorear el volumen y el caudal a través de los indicadores virtuales con los que cuenta, y lo hace de forma inalámbrica gracias a la conexión entre los módulos Xbee.

En el panel de control se incluyeron controles virtuales que permiten controlar el flujo de líquido en la planta que se construyó para probar el sistema. Los controles virtuales permiten al usuario encender o apagar la bomba de agua que controla el flujo de líquido, con una opción manual y otra automática.

Cuando el sistema opera de forma manual, el usuario es quien decide cuando detener o reanudar el flujo de líquido, cambiando el estado del control virtual correspondiente en el panel de control. Si el sistema opera de forma automática, el usuario tiene que fijar un valor de volumen con la ayuda de la perilla virtual localizada en el panel de control, y el sistema permitirá el flujo de líquido hasta alcanzar el valor de volumen antes fijado por el usuario.

El panel de control cuenta con los indicadores de volumen, y gasto, que permiten al operador del sistema monitorear el flujo de líquidos en el sistema y tomar alguna medida de control. El funcionamiento detallado del panel de control se ha explicado en el tema 5 de este trabajo correspondiente al tema.

Para probar el funcionamiento del sistema de medición y control de líquidos desarrollado, fue necesario construir una pequeña planta en la cual se pudieran realizar las pruebas de medición, control, y monitoreo de flujo de volumen de agua.

La planta se construyó utilizando una bomba de agua como actuador del sistema, un contenedor principal de 20[L], un contenedor secundario de 4[L], y el sensor ultrasónico de flujo, como se muestra en la figura 7.14.

Para revisar el funcionamiento de la comunicación inalámbrica, se comprobó que el actuador (bomba de agua) respondiera ante el accionamiento de encendido y apagado del control virtual ubicado en el panel de control. Para realizar esta prueba se utilizó el modo manual del sistema, en donde el operador es quien toma la decisión de detener o reanudar el flujo del líquido, basando su decisión mediante el monitoreo del volumen con la ayuda del indicador correspondiente en el panel de control.

Al final se encontró que la bomba de agua responde correctamente a las instrucciones de encendido y apagado hechas por el operador del sistema desde el panel de control implementado en la computadora. Los datos de volumen y caudal de líquido son enviados desde el circuito de instrumentación hacia la computadora, y desplegados en sus respectivos indicadores, con lo cual se comprobó que la comunicación inalámbrica funciona correctamente.

Figura 7.14 Prueba de control de volumen.

Para revisar el funcionamiento del modo automático del sistema se utilizó el contenedor de 4[L] como volumen patrón, con el cual se comparó el control de volumen hecho por el sistema. La prueba consistió en fijar un valor de 4 [L] a través del panel de control y permitir que el sistema se encargara de transmitir dicha cantidad de volumen de agua del contenedor principal con capacidad de 20 [L], hacia el volumen utilizado como patrón de comparación de 4[L]. Como se muestra en la figura 7.14.

Debido a que el sensor de flujo proporciona medidas de volumen más precisas, durante las pruebas, las mediciones de volumen se tomaron del módulo LCD que forma parte del circuito de instrumentación del sensor. La finalidad de la prueba fue comprobar si el sistema es capaz de controlar un volumen determinado por el operador, y verificar la precisión de este. La siguiente tabla muestra los resultados de la prueba que se repitió diez veces.

Prueba	*Volumen leído. [L]*	*Promedio [L]*	*Desviación [L]*
1	4.15	4.14	0.01
2	4.13	4.14	0.01
3	4.15	4.14	0.02
4	4.15	4.14	0.01
5	4.16	4.14	0.02
6	4.14	4.14	0.00
7	4.15	4.14	0.01
8	4.13	4.14	0.01
9	4.13	4.14	0.01
10	4.15	4.14	0.01
			Imprecisión=0.011

Tabla 1.7 de resultados de la prueba de control de volumen.

La tabla muestra los resultados de la prueba de control de volumen hecha por el sistema cuando este opera de forma automática, en ella se observa que el promedio del volumen leído es de 4.14 [L], el cual es superior por 140[ml] aproximadamente al volumen establecido por el operador en el panel de control, por lo que se concluye que el sistema es inexacto debido al actuador empleado y su algoritmo de control de tipo ON/OFF.

En la cuarta columna de la tabla de resultados se ha calculado la desviación individual de cada medición con respecto a la media aritmética de las lecturas hechas, con el fin de obtener la *imprecisión* del sistema. Si la *imprecisión* es pequeña el sistema es más preciso.

Al sacar el promedio de las desviaciones individuales en la cuarta columna se obtuvo la *imprecisión* de 0.011[L], la cual es pequeña con respecto a la media aritmética de las lecturas hechas. Con esto se demuestra que el sistema tiene una buena precisión a pesar de que es inexacto.

Conclusiones

- Al finalizar el proyecto se logró la implementación física de un sistema que permite, medir, controlar, y monitorear el flujo de líquidos. El sistema es funcional, y fue probado exitosamente utilizando agua como líquido de trabajo.

- A pesar que el sensor de flujo utilizado puede trabajar con diferentes líquidos con características de viscosidad distintas a las del agua, en este prototipo solo fue probado con agua debido a la bomba que se utilizó como actuador. Para manejar líquidos viscosos es necesario utilizar un actuador adecuado que controle el flujo del líquido utilizado.
- En las pruebas de funcionamiento se demostró que el sistema es inexacto en el control de volumen, debido al actuador y a la acción de control implementada (ON/OFF). Por lo que si se requiere una mayor exactitud en el sistema, es indispensable utilizar un actuador al cual se le pueda aplicar un algoritmo de control más avanzado.
- El sistema diseñado puede ser instalado fácilmente en cualquier entorno de trabajo, ya que no cuenta con ningún cable físico de comunicación de datos entre el circuito de instrumentación y el panel de control implementado en la computadora, gracias a la conexión inalámbrica que se le ha integrado.
- El sensor de flujo presenta errores en sus mediciones cuando el líquido con el que se trabaja contiene burbujas de aire, por lo que se requiere de una purga en el sistema antes de ser utilizado.
- Durante la prueba de control de volumen realizada se observó la exactitud y la precisión del sistema. Al tomarse una muestra de 10 mediciones y sacar su media aritmética, se observó que esta es superior por 140[ml] a la referencia de volumen hecha por el operador de sistema desde el panel de control.

  Al analizar los resultados se concluyó que el sistema es inexacto ya que en todas las mediciones se registraron valores distintos a los que el operador indicó en el panel de control. Sin embargo al hacer un análisis sencillo sobre la precisión del sistema, se encontró que este es capaz de reproducir las mediciones realizadas bajo las mismas condiciones, y que la desviación de estas con respecto a la media aritmética de las lecturas es pequeña. Por lo que se concluyó que el sistema es precisó a pesar de que este es inexacto.
- El prototipo desarrollado funciona con un circuito de instrumentación y un receptor, diseñados para permitir la comunicación con el software de instrumentación virtual LabView, sin emplear alguna tarjeta de adquisición de datos especializada que provocaría que el costo del proyecto se elevara.

Bibliografía

[1] Eduardo García Breijo, Simulador proteus para microcontroladores PIC, 2008.

[2] José Rafael Lajara Vizcaino y José Pelegrí Sebastián, Labview entorno gráfico de programación, 2007.

[3] Luis Thayer Ojeda, (recuperación Mayo 2014) Guía del usuario XBEE series 1.

disponible en red:

http://www.olimex.cl/pdf/Wireless/ZigBee/XBee-Guia_Usuario.pdf

[4] Junior Figueroa Olmedo, (recuperación Mayo 2014) Teoría y programación módulos Xbee, disponible en red:

http://es.scribd.com/doc/58980339/Teoria-y-Programacion-Módulos-XBEE

[5] Cynergy, (recuperación Mayo 2014) Ultrasonic Flow Meter UF25B,

disponible en red:

http://www.farnell.com/datasheets/1833198.pdf

[6] Fredrich Hofmann, (recuperación Mayo 2014) Fundamentals of ultrasonic flow measurement for industrial applications,

disponible en red:

http://www.investigacion.frc.utn.edu.ar/sensores/Caudal/HB_ULTRASONIC_e_144.pdf