UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

"SISTEMA DE MONITOREO PARA ACUARIO CON INTERFAZ USB"

T E S I S

QUE PARA OBTENER EL TÍTULO DE INGENIERO MECÁNICO ELECTRICISTA

P R E S E N T A :

DANIEL ANDRES FLORES BELLO

ASESOR:

ING. ARTURO OCAMPO ÁLVAREZ

México……2011

"Nunca consideres el estudio como una obligación,
sino como una oportunidad para entrar
en el bello y maravilloso mundo del saber."

Albert Einstein.

AGRADECIMIENTOS

CONTENIDO

CAPÍTULO 1: INTRODUCCIÓN 1

1.1. ANTECEDENTES DE LA ELECTRÓNICA Y LA ACUARIOFILIA 2
1.2. ACUARIO 4
1.2.1. TIPOS DE ACUARIOS 4
1.2.2. CONTROL DEL AMBIENTE EN UN ACUARIO 5
1.3. OBJETIVO GENERAL 7
1.4. OBJETIVOS PARTICULARES 7
1.5. ALCANSES Y LIMITACIONES 7

CAPÍTULO 2: MICROCONTROLADOR 8

2.1. ¿QUÉ ES UN MICROCONTROLADOR? 8
2.2. CONTROLADOR Y MICROCONTROLADOR 8
2.3. MICROPROCESADOR Y MICROCONTROLADOR 10
2.4. ARQUITECTURA DE VON NEUMANN 11
2.5. ARQUITECTURA HARVARD 12
2.6. ALIMENTACIÓN DE UN AVR ATmega8 13
2.7. PUERTOS DE ENTRADA/SALIDA 15
2.8. OSCILADOR. 16
2.8.1. OSCILADOR External Crystal/Ceramic Resonator 17
2.8.2. OSCILADOR External Low-frequency Crystal 18
2.8.3. OSCILADOR External RC Oscillator 18
2.8.4. OSCILADOR Internal RC Oscillator 18
2.8.5. RELOJ EXTERNO External Clock 18
2.9. PERIFÉRICOS BÁSICOS 19
2.9.1. LED 19
2.9.2. INTERRUPTORES Y PULSADORES 20
2.9.3. ENTRADAS DIGITALES CON OPTOACOPLADORE 21
2.10. ORGANIZACIÓN DE LA MEMORIA 22
2.10.1. ARQUITECTURA INTERNA DEL AVR-ATmega8 22
2.10.2. MEMORIA DE PROGRAMA 23
2.10.3. MEMORIA DE DATOS SRAM 24
2.10.4. UNIDAD CENTRAL DE PROCESOS (CPU) DE Atmega8(L) 27

2.10.5. PUNTERO DE PILA 27
2.11. RECURSOS ESPECIALES DEL Atmega8(L) 28
2.12.MÓDULO DEL CONVERTIDOR A/D 28
2.12.1. CONTROL DEL CONVERTIDOR A/D 30
2.12.2. TIEMPO DE CONVERSIÓN A/D 34
2.13.MODULO USART 35
2.13.1. REGISTROS DE CONTROL PARA EL MODULO USART 37
2.14. INTERRUPCIONES 42
2.15. INTERRUPCIONES EN EL AVR ATmega8L 43
2.16. CONTROL DE UN LCD CON UN MICROCONTROLADOR 44
2.16.1. DIVERSIDAD DE ALGUNOS MÓDULOS LCD 44
2.16.2. ASPECTO FÍSICO 45
2.16.3. LOS CARACTERES DEL LCD 46
2.16.4. ASIGNACIÓN DE PINES 47
2.16.5. INTERPRETACIÓN DEL SIGNIFICADO DE LOS PINES DEL LCD 48
2.16.6. INTERFAZ DEL DISPLAY CON EL MUNDO EXTERIOR 50
2.16.7.CONEXIÓN DEL LCD AL MICROCONTROLADOR AVRATmega8L 51

CAPÍTULO 3: DISEÑO DE INTERFAZ 52

3.1. ¿QUÉ ES USB? 52
3.2. VELOCIDADES DE TRANSMICION 52
3.3. CABLES Y CONECTORES 53
3.4. TOPOLOGIA EN DISPOCITIBOS USB 55
3.5. FUNCIONAMIENTO 57
3.6. MODULO DE INTERFAZ USB 58
3.6.1. DIAGRAMA A BLOQUES DEL FT232BL. 62
3.6.2. DESCRIPCION DE FUNCION DE LOS BLOQUES DEL FT232 62
3.7. DIAGRAMA DEL MODULO DE INTERFAZ USB 64
3.8. ¿QUE ES UN INSTRUMENTO 65
3.9. INSTRUMENTACIÓN VIRTUAL. 65
3.10. COMPONENTES DE UN SISTEMA BASADO EN INSTRUMENTACION VIRTUAL. 69
3.10.1. TRANSDUCORES. 69

3.10.2. BLOQUES TERMINALES. 70
3.10.3. HARDAWARE DE ACONDICIONAMIENTO DE SEÑAL. 70
3.10.4. HARDWARE DE ADQUISICION DE DATOS 70
3.10.5. CABLES DE CONEXIÓN. 70
3.10.6. COMPUTADORA 70
3.10.7. SOFTWARE 71
3.11. EL SOFTWARE EN LA INSTRUMENTACIÓN VIRTUAL 71
3.12. LABVIEW COMO HERRAMIENTA PARA CREAR INSTRUMENTOS VIRTUALES 72
3.13. LABVIEW 74
3.13.1. PROGRAMACION CON LABVIEW 76
3.13.2. EJECUCIÓN DE UN PROGRAMA EN LABVIEW 77

CAPÍTULO 4: AUTOMATIZACION DE ACUARIO 79

4.1. POTENCIAL DE HIDROGENO (pH) 79
4.2. TEMPERATURA 80
4.3. DISEÑO DEL MODULO DE MEDICION Y TRANSMICION POR USB 81
4.3.1. SENSORES 81
4.3.2. PROGRAMANDO EL MICROCONTROLADOR 84
4.3.3 ANALISIS Y DISEÑO DEL ALGORITMO 86
4.3.4. PROGRAMACION Y GRABADO DEL AVR 88
4.3.5. ELABORACION DEL MÓDULO DE MONITOREO PARA ACUARIO CON INTERFAZ USB 92
4.3.6. MONITOREO DE DATOS EN LA PC 96

CONCLUSIONES 97
GLOSARIO. 100
BIBLIOGRAFIA.....105
ANEXOS….....106

CAPÍTULO 1: INTRODUCCIÓN

El estudio de la interacción sobre los parámetros y acondicionamiento del medio en un acuario así como también en la Acuicultura (conjunto de actividades, técnicas y conocimientos de cultivo de especies acuáticas vegetales y animales), nos permite proporcionar indicadores que relacionan la respuesta productiva o de mantenimiento de especies y ecosistemas acuáticos artificiales, de acuerdo al comportamiento de las diversas variables que existen en este medio. Como consecuencia, un registro y análisis, adecuados del comportamiento de las diferentes variables que intervienen para el adecuado acondicionamiento del medio, contribuyen a la optimización y respuesta productiva de las diferentes especies que se encuentren en el acuario, que así mismo contribuyen para que las especies que se encuentran tengan un mayor tiempo de vida. Este trabajo presenta el diseño de un sistema de monitoreo para ser aplicado en un acuario.

En la actualidad en diferentes sectores tanto en la industria como en comunicaciones, se ha producido con gran impacto la solución de transmitir información por USB, por ser un medio muy eficiente para realizar dicha función, en donde también entra el sector de la Acuicultura facilitando la transmisión de la información, para el proceso de cultivo de las especies acuáticas. La incorporación de esta tecnología, tiene una gran aceptación a tal grado que no solamente se limita a la aplicación de transmisión de información, sino también a todas aquellas aplicaciones relacionadas con sistemas de control.

Una de las cuestiones que llama la atención, es que en la actualidad el poder de procesamiento, la variedad, la facilidad de uso y el bajo consumo de energía de los microcontroladores, nos ayuda a diseñar soluciones económicas pero confiables y eficientes. El presente trabajo ha incorporado el uso de un microcontrolador en el módulo. En cuanto a la elección de los sensores, se encuentran como ventajas, que estos dispositivos se ubican dentro de la gama de dispositivos de mayor exactitud, bajo consumo y facilidad de acondicionamiento. Los componentes elegidos han hecho de este diseño una solución sencilla pero ajustada a los requerimientos de la aplicación.

Dado que en la actualidad existe un gran número de personas que practican la Acuariofilia (afición a la cría de peces) y que es de gran importancia el estar monitoreando diferentes parámetros del acuario, y que esto se puede realizar con métodos muy básicos, pero estos implican en muchas ocasiones una gran complejidad y tiempo para realizarlos, aunque también existen sistemas que facilitan este tipo de monitoreo, pero los cuales tienen un gran costo en el mercado y que por lo regular son equipos que solamente adquieren personas que se dedican a la Acuicultura, se hace referencia al diseño de este sistema.

El sistema beneficiará a las personas que tienen en su hogar acuarios y personas que se dedican a la crianza de peces de ornato, que en muchas ocasiones por falta de tiempo no pueden realizar con frecuencia el monitoreo de los principales parámetros (temperatura y pH), que son de las más importantes para que su acuario este en óptimas condiciones.
Ya que no solamente basta con estar monitoreando las mediciones en tiempo real y que se puedan visualizar en un display estas también son enviadas a la computadora por una interfaz USB para tener un registro de estos parámetros, de igual forma el módulo de medición se hizo portátil para su fácil manejo.

1.1. ANTECEDENTES DE LA ELECTRÓNICA Y LA ACUARIOFILIA

El acelerado desarrollo que la ciencia y la tecnología en los últimos años, ha tenido un crecimiento espectacular en la microelectrónica y en la biotecnología.

La tecnología, que busca la aplicación industrial con sus aportes principalmente en la electrónica, está demostrando su capacidad, no sólo de sustituir la fuerza y habilidad física del hombre con la robótica, sino también su capacidad de trabajo intelectual con las nuevas generaciones de computadoras que han dado lugar a hablar de la “inteligencia artificial”. Mientras que, la informática permite estar al tanto de lo que ocurre en cualquier lugar del mundo de manera casi simultánea a los hechos y participar en tomas de decisiones a distancia.

La Acuariofilia es la afición a la cría de peces y otros organismos acuáticos en un acuario, bajo condiciones controladas. Ha evolucionado tremendamente a lo largo de los siglos, desde el mantenimiento de carpas doradas con fines ornamentales en recipientes y estanques, desde hace 2000 años.

Actualmente la Acuariofilia es una afición que puede llegar a altos niveles de conocimiento y sofisticación, que traspasan la frontera de afición para convertirse en una verdadera ciencia, la Acuariología.

El origen de la Acuariofilia es muy antiguo, y va ligado al de la acuicultura. Los antecedentes de cultivo de peces, fundamentalmente carpas, se remontan a los sumerios, que ya utilizaban estanques para mantener peces vivos destinados a alimentación. Los romanos también criaban carpas destinadas al consumo.
En China, los bancales inundados para el cultivo de arroz eran utilizados para la cría de carpas, como fuente complementaria de proteína. De estos cultivos aparecieron formas coloreadas de carpines dorados y carpas koi que fueron seleccionadas por su belleza. Posteriormente fueron llevadas a Japón, donde se desarrollaron nuevas variedades.

El principio básico de la Acuariofilia moderna es la recreación de un ecosistema acuático artificial en el que puedan desarrollar un comportamiento natural todo tipo de especies acuáticas, y estabilizado a través de sistemas técnicos auxiliares. Ya no es una afición centrada en el mantenimiento exclusivo de peces, sino una afición basada en una ciencia, la acuariología como ya se había mencionado antes.

En realidad no existen muchos antecedentes entre la Electrónica y la Acuariofilia como tal, ya que fue este el principal motivo por el cual se realizó este proyecto, lo más representativo que se tiene como relación es la gran variedad de equipos de acondicionamiento para el acuario como lo son:

- Filtros.
- Lámparas.
- Bombas de suministro de aire.
- Calentadores
- Equipos de medición de Temperatura ó pH (Análogos ó Digitales)

Que serían los equipos más básicos para poder tener en óptimas condiciones un acuario y que se pueda acercar más a un ambiente natural.
Los equipos que sobresalen y que tienen una mayor relación con la electrónica serían los equipos de medición, tales como los mencionados anteriormente que serian los de pH y temperatura los cuales existen pero su costo es muy elevado, y que son ocupados por lo general en la Acuicultura ó en la Acuariología. Así como también diferentes sistemas de Control para los demás equipos.

Figura 1-1 pH-Metro con sensor de temperatura

1.2. ACUARIO

Un acuario es un recipiente capaz de contener agua, con al menos una de sus caras de algún material transparente, generalmente de vidrio o metacrilato, y dotado de los componentes mecánicos que hacen posible la recreación de ambientes subacuáticos de agua dulce, marina o salobre y albergar vida correspondiente a esos ambientes, como peces, invertebrados, plantas, etc. El diseño más básico de acuario es de planta rectangular, realizado a partir de vidrios sellados con silicona neutra.

En un acuario las condiciones ambientales son estables y controladas, y están adecuadas para la vida de los organismos que van a vivir en él. Los acuarios más sofisticados pueden albergar un auténtico arrecife marino, dotados de sistemas de iluminación especiales, bombas, generadores de olas, filtros físicos, biológicos y químicos, control de temperatura, dosificadores de elementos traza, reactores, medidores de parámetros, etc.

1.2.1. TIPOS DE ACUARIOS

De manera general, y según la concentración en sales minerales del agua, los acuarios se dividen en:

- **Acuarios de agua dulce** (concentración salina de < 0,5%), que simula un ambiente lacustre o fluvial

- **Acuarios de agua salada** (concentración salina de 5%-18%), que simula un ambiente marino u oceánico

- **Acuarios de agua salobre** (concentración salina de 0,5%-5%), que simula los ambientes intermedios en cuanto a concentración salina, como por ejemplo albuferas o estuarios.

El Acuario de Agua Dulce es el más clásico y en él las especies requieren una temperatura que oscile entre los 24 y 28 °C, el pH en el acuario debe de ser entre 6.5 y 7.5, este acuario es muy sencillo de mantener ya que sus parámetros son muy fáciles de establecer y por lo cual nuestro sistema se basa en un acuario de este tipo.

Un Acuario de Agua Salada como su nombre los dice es el cual tendrá un ambiente marino u oceánico. Los parámetros críticos de un tanque de agua salada son pH, nitrato, salinidad y temperatura. El pH de un tanque marino es uno de los parámetros más importantes. Los peces y los invertebrados marinos son especialmente sensibles a los cambios rápidos del pH, así que mantener fluctuaciones del pH dentro de un 0,2 a lo largo del día es muy crítico. Todas las criaturas marinas se sienten a gusto con un pH cerca de 8.2, con un margen de 8.0 a 8.4. El pH nunca debe bajar de 8.0.

El parámetro crítico siguiente son los nitratos. Los peces de agua salada son más tolerantes a nitratos elevados que los invertebrados (en general), pero aun así prefieren nitratos por debajo de 20 ppm (parte por millón), con menos de 5 ppm requeridos para la mayoría de los invertebrados. Los cuidadores de Arrecifes tienden a calificar como inaceptable un nivel por encima de 0,5 ppm, pero esto es una meta poco realista para los tanques de solo peces o con un mínimo de invertebrados.

El parámetro siguiente de preocupación es salinidad, o peso específico. Libremente (muy libremente), el peso específico es la cantidad de sal en el agua. Muchos acuariófilos tratan al peso específico y la salinidad como un misma cosa, pero técnicamente hablando, no lo son. El peso específico es dependiente de la temperatura y la salinidad no lo es. La mayoría de los aerómetros o densímetros (los aerómetros miden peso específico) están calibrados para leer el peso específico correcto a 15 °C. Puesto que ésta es uno poco bajo para la mayoría de tanques, los aerómetro para aficionados están generalmente corregidos para leer el peso específico correcto alrededor de una temperatura de 25 °C.

Finalmente, la temperatura de un tanque de agua salada es básicamente la misma que en un tanque de agua dulce. A cualquier temperatura entre 24 °C y 27 °C, con 25 °C siendo un buen término medio.

1.2.2. CONTROL DEL AMBIENTE EN UN ACUARIO

Si bien es posible conservar peces vivos un cierto tiempo en un poco de agua sin ninguna ayuda tecnológica, las condiciones de vida serán muy malas, por ello todo acuario se debe equipar correctamente.

- **FILTRACIÓN**

Es vital que el agua del acuario circule, se le quiten las impurezas y esté biológicamente depurada. Para hacer esto, se utiliza una bomba de agua, que abastezca correctamente a las masas de filtración, asegurando la filtración mecánica, así como la desintoxicación biológica, por la acción de bacterias o de materiales absorbentes.

La mezcla del agua implica también una función oxigenante y permite recrear ciertos medios de vida agitados.

- **ILUMINACIÓN**

Con el fin de ver bien los peces, de darles un biorritmo diario y de asegurar la fotosíntesis de las plantas, un acuario se debe iluminar correctamente.
El método aparentemente más simple es el de utilizar la luz del sol, pero esto tiene numerosos inconvenientes. La calidad de la luz es importante, ya que

contribuye entre otros factores al crecimiento sano de los peces. Lo importante para iluminar correctamente un acuario es calcular la ratio vatios por litro, siendo adecuado para los acuarios de agua dulce de 0,4 vatios por litro de agua y de 0,6 para los de agua salada. Por ejemplo, para un acuario de 60 L con plantas, necesitaremos una lámpara de 24 W.

- **CLIMATIZACIÓN**

Para mantener una temperatura tropical, lo cual conviene a los peces exóticos, se tienen que utilizar sistemas de climatización, compuestos de una resistencia calentadora y de un termostato. Un acuario de agua dulce funciona entre 21-27°C, dependiendo de las especies.

- **SUMINISTRO DE AIRE**

Es un aspecto importante el considerar un sistema que proporcione una corriente continua de aire al agua del acuario. En el mercado existen varios sistemas, como el de la bomba de diafragma o el recirculador tipo Venturi, que son ideales para mantener oxigenada el agua, oxidando los nitritos producidos y convirtiéndolos en nitratos, que son menos nocivos.

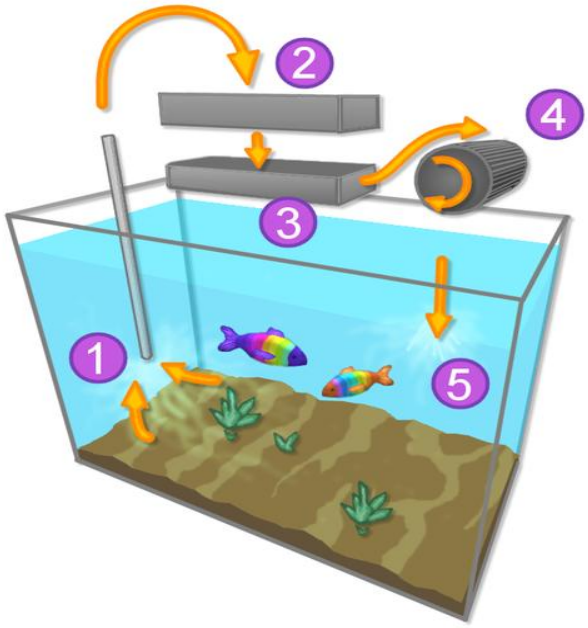

Figura 1-2 Sistema de filtración en un acuario típico:
(1) Entrada. (2) Filtración mecánica. (3) Filtración de carbono activo. (4) Medio de filtración biológica. (5) Salida a la reserva

1.3. OBJETIVO GENERAL

Diseñar un sistema para monitorear los parámetros de temperatura y pH en un acuario ya sea de un hogar ó uno de mayores dimensiones, así como también enviar la información por USB a una PC.

1.4. OBJETIVOS PARTICULARES

- Seleccionar los sensores para el sistema
- Seleccionar el Microcontrolador que se adecua a las necesidades del proyecto
- Controlar un LCD (Liquid Cristal Display)
- Realizar el algoritmo que realice las operaciones matemáticas correspondientes para poder visualizar en el LCD los datos obtenidos de acuerdo a las mediciones de los sensores.
- Seleccionar el modo de enviar la información vía USB
- Realizar un programa con instrumentación virtual para poder visualizar los datos de las mediciones en la PC.

1.5. ALCANCES Y LIMITACIONES

El alcance de este trabajo está de acuerdo con las necesidades planteadas para un acuario estándar en un hogar, ya sea de agua dulce o marina. Aunque el tema de monitoreo de variables en un acuario es muy extenso, las personas que tienen este tipo de ornamento en su hogar muchas veces puede requerir de soluciones tecnológicas para poder mantener y procurar sus acuarios, que suelen ser de una gran complejidad y de muy alto costo.

En lo que se refiere a las mediciones físicas, es de mucha importancia el monitorear del pH, temperatura, salinidad, corriente del agua entre otras, pero cabe mencionar que las variables físicas de mayor incidencia para el mantenimiento de un acuario son la temperatura y el potencial de hidrogeno pH, que son las cuales se incorporaron al sistema.

CAPÍTULO 2: MICROCONTROLADOR

2.1. ¿QUÉ ES UN MICROCONTROLADOR?

Un microcontrolador es un circuito integrado programable que contiene todos los componentes necesarios para controlar el funcionamiento de una tarea determinada como lo son: el control de una lavadora, un teclado de computadora, una impresora, un sistema de alarma, el funcionamiento de los ratones, en los teléfonos, en los hornos de microondas y los televisores de nuestro hogar, etc. Para esto, el microcontrolador utiliza muy pocos componentes asociados. Un sistema con microcontrolador debe disponer de una memoria donde se almacena el programa que gobierna el funcionamiento del mismo que, una vez programado y configurado, sólo sirve para realizar la tarea asignada. La utilización de un microcontrolador en un circuito reduce notablemente el tamaño y el número de componentes y, en consecuencia, disminuye el número de averías, el volumen y el peso de los equipos, entre otras ventajas.

El microcontrolador es uno de los inventos más notables del siglo XX. En el mercado hay gran cantidad de ellos, con multitud de posibilidades y características. Cada tipo de microcontrolador sirve para una serie de casos y es el diseñador del sistema quien debe decidir cuál es el microcontrolador más idóneo para cada uso.

Por otra parte, el microcontrolador es un sistema completo (microprocesador + puertos de entrada o salida + memoria + otros periféricos) que está contenido en el chip de un circuito integrado programable y se destina a gobernar una sola tarea con el programa que reside en su memoria. Sus líneas de entrada/salida soportan el conexionado de los sensores y actuadores del dispositivo a controlar.

Si sólo se fabricara un modelo de microcontrolador, éste debería tener muy potenciados todos sus recursos para poderse adaptar a las exigencias de las diferentes aplicaciones. En la práctica, cada fabricante de Microcontroladores oferta un elevado número de modelos diferentes, desde los más sencillos hasta los más potentes. Es posible seleccionar la capacidad de las memorias, el número de líneas de E/S (entrada o salida), la cantidad y prestaciones de los elementos auxiliares, la velocidad de funcionamiento, etc. Por todo ello, un aspecto muy importante del diseño es la selección del microcontrolador a utilizar.

2.2. CONTROLADOR Y MICROCONTROLADOR

Recibe el nombre de controlador el dispositivo que se emplea para el gobierno de uno o varios procesos. Por ejemplo, el controlador que regula el funcionamiento de un horno dispone de un sensor que mide constantemente

su temperatura interna y, cuando traspasa los límites prefijados, genera las señales adecuadas que accionan los efectores que intentan llevar el valor de la temperatura dentro del rango estipulado.

Aunque el concepto de controlador ha permanecido invariable a través del tiempo, su implementación física ha variado frecuentemente. Hace tres décadas, los controladores se construían exclusivamente con componentes de lógica discreta, posteriormente se emplearon los microprocesadores, que se rodeaban con chips de memoria y E/S sobre una tarjeta de circuito impreso. En la actualidad, todos los elementos del controlador se han podido incluir en un chip, el cual recibe el nombre de microcontrolador. Realmente consiste en una sencilla pero completa computadora contenida en el corazón (chip) de un circuito integrado.

Un microcontrolador es un circuito integrado de alta escala de integración que incorpora la mayor parte de los elementos que configuran un controlador. Un microcontrolador dispone normalmente de los siguientes componentes:

- Procesador o CPU (*Central Processing Unit*).
- Memoria RAM (*Random-Access Memory*) para contener los datos.
- Memoria para el programa tipo ROM (*Read Only Memory*)/PROM (*Programmable Read-Only Memory*)/EPROM (*Erasable Programmable Read-Only Memory*).
- Líneas de entrada o salida para comunicarse con el exterior.
- Diversos módulos para el control de periféricos (temporizadores, Puertas Serie y Paralelo, CAD (*Conversor Analógico-Digital*), CDA (*Conversor Digital-Analógico*), etc.).
- Generador de impulsos de reloj que sincronizan el funcionamiento de todo el sistema.

Los productos que para su regulación incorporan un microcontrolador disponen de las siguientes ventajas:

- Aumento de prestaciones: un mayor control sobre un determinado elemento representa una mejora considerable en el mismo.
- Aumento de la fiabilidad: al reemplazar un elevado número de elementos por un microcontrolador, disminuye el riesgo de averías y se precisan menos ajustes.

- Reducción del tamaño en el producto acabado: La integración del microcontrolador en un chip disminuye el volumen y la mano de obra.

- Mayor flexibilidad: las características de control están programadas, por lo que su modificación sólo necesita cambios en el programa de instrucciones.

El microcontrolador es en definitiva un circuito integrado que incluye todos los componentes de una computadora. Debido a su reducido tamaño es posible montar el controlador en el propio dispositivo al que gobierna. En este caso el controlador recibe el nombre de controlador empotrado (*embebed controller*).

2.3. MICROPROCESADOR Y MICROCONTROLADOR

Un microprocesador es básicamente un chip que contiene la CPU se encarga de controlar todo el sistema. Un sistema digital basado en un microprocesador es un sistema abierto, ya que su configuración defiere según la aplicación a la que se destine. Se pueden acoplar los módulos necesarios para configurarlo con las características que se desee. Para ello saca al exterior las líneas de sus buses de datos, direcciones y control, de modo que permita su conexión con la memoria y los módulos de entrada/salida. Finalmente resulta un sistema implementado por varios circuitos integrados dentro de una misma placa de circuito impreso. Figura 2-1.

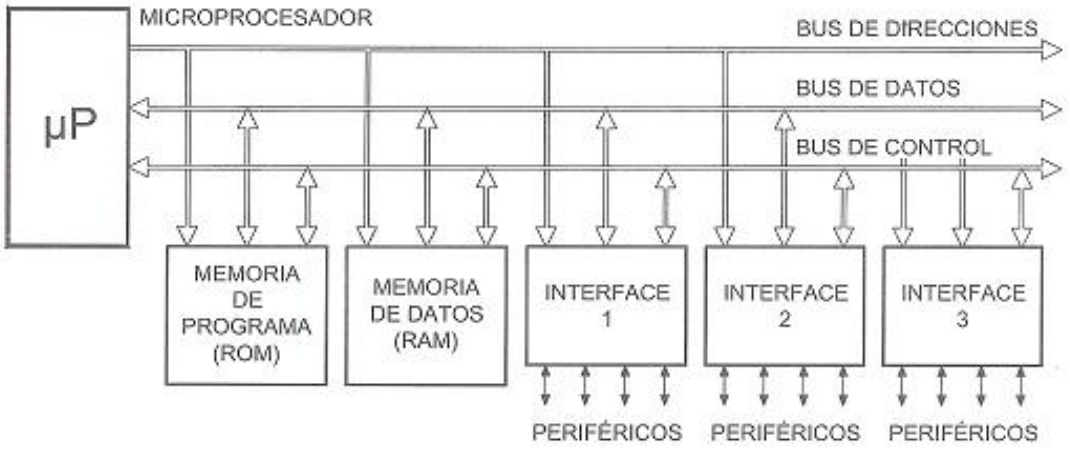

Figura 2-1 Estructura de un sistema digital basado en microprocesador

Un microcontrolador es un sistema cerrado, lo que quiere decir que en un solo circuito integrado se encierra un sistema digital programable completo. Este dispositivo se destina a gobernar una sola tarea que no se puede modificar. Los microcontroladores disponen de los bloques esenciales: CPU, memorias de datos y de programa, reloj, periféricos de entradas/salidas, etc. Figura 2-2.

En las figuras (2-1 y 2-2) se muestran las diferencias entre los sistemas digitales basados en microprocesador respecto de los basados en microcontrolador.

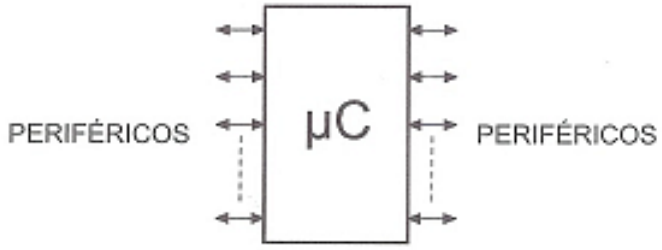

Figura 2-2 Estructura de un sistema digital basado en microcontrolador

La diferencia fundamental entre ambos es que, un sistema digital basado en un microcontrolador está formado por un solo circuito integrado, lo que reduce notablemente el tamaño y el costo, mientras que un sistema basado en un microprocesador, al estar compuesto por varios circuitos integrados para soportar las memorias y los módulos de entrada/salida, tiene mayor tamaño, más costo y menor fiabilidad.

2.4. ARQUITECTURA DE VON NEUMANN

La arquitectura tradicional de sistemas digitales programables se basa en el esquema propuesto por John Von Neumann. En este modelo, la unidad central de proceso o CPU está conectada a una memoria única que contiene las instrucciones del programa y los datos, como se puede observar en la figura 2-3.

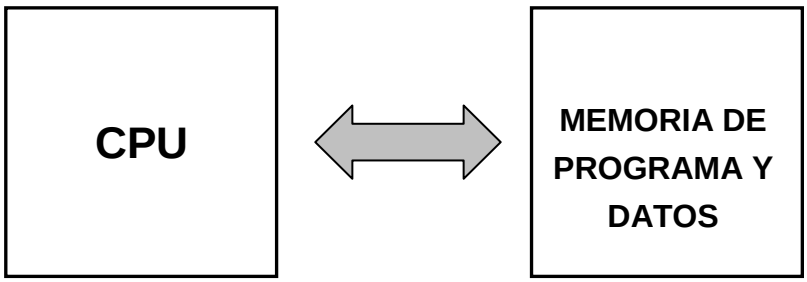

Figura 2-3 Arquitectura de Von Neumann

El tamaño de la unidad de datos o instrucciones está fijado por el ancho del bus de datos de la memoria exterior utilizada, que es de 8 bits. Un microprocesador con un bus de 8 bits que lo conecta con la memoria deberá manejar datos e instrucciones de una o más unidades de 8 bits de longitud. Cuando deba acceder a una instrucción o dato de más de un byte de

longitud, deberá realizar más de un acceso a la memoria. Por otro lado, este bus único limita la velocidad de operación del microprocesador, ya que no se puede buscar en la memoria una nueva instrucción antes de que finalicen las transferencias de datos que pudieran resultar de la instrucción anterior.

Ahora, lo que se puede observar dentro de la arquitectura tradicional o de Von Neumann son dos principales limitaciones:

- La longitud de las Instrucciones está limitada por la unidad de longitud de los datos, por lo tanto el microprocesador debe hacer varios accesos a memoria para buscar instrucciones complejas.

- La velocidad de operación está limitada por el efecto de cuello de botella que significa un único bus para datos e instrucciones, que impide superponer ambos tiempos de acceso.

Se puede rescatar una ventaja, que consiste en simplificar la lógica del microcontrolador.

2.5. ARQUITECTURA HARVARD

Tradicionalmente, los sistemas digitales programables se basaban en la arquitectura de Von Neumann, esta se caracterizaba por disponer de una única memoria en la que se almacenaba tanto los datos como las instrucciones, a esta memoria se podía acceder únicamente por un bus.

La arquitectura Harvard (figura 2-4) consiste en disponer de dos memorias independientes, a las que se conecta mediante dos grupos de buses separados:

- Memoria de datos.

- Memoria de programa.

Ambos buses son totalmente independientes y pueden ser de distintos anchos, esto permite que la CPU pueda acceder de forma independiente y simultánea a la memoria de datos y a la de instrucciones, consiguiendo que las instrucciones se ejecuten en menos ciclos de reloj.

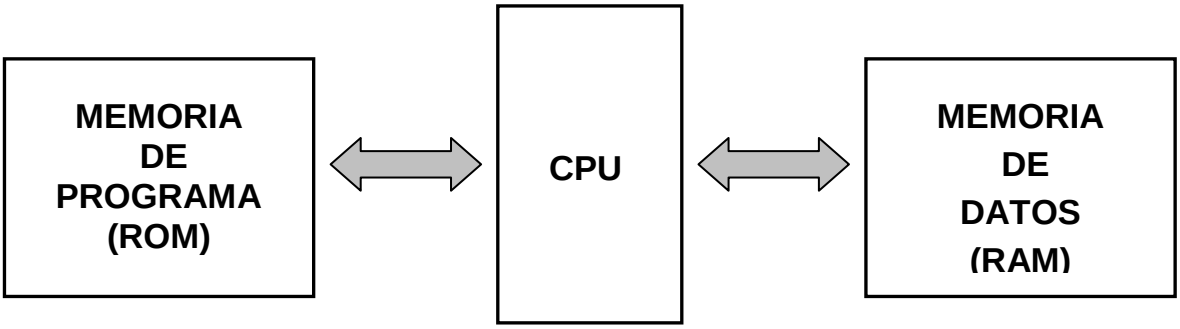

Figura 2-4 Arquitectura Harvard

Esta dualidad, de la memoria de datos por un lado y por otro la memoria de programa, permite la adecuación del tamaño de las palabras y los buses a los requerimientos específicos de las instrucciones y los datos.

Las principales ventajas de la arquitectura Harvard son:

1. El tamaño de las instrucciones no está relacionado con el de los datos, y por esta razón puede ser optimizado para que cualquier instrucción ocupe una sola posición de memoria de programa. Con esto se asegura una mayor velocidad y una menor longitud de programa.

2. El tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad de operación.

2.6. ALIMENTACION DE AVR ATmega8L

Cabe mencionar que la fuente de alimentación utilizada para el diseño del sistema es una fuente simétrica de +5v y -5v la cual se verá mas adelante con detalle, pero que es de gran importancia mencionar el tema de la alimentación del microcontrolador

Normalmente, el microcontrolador AVR-ATmega8L se alimenta con 5 volts, aplicados entre los pines VCC y GND que son, respectivamente, la alimentación y la tierra del chip.

Estos 5 volts de alimentación se pueden obtener de un circuito que suministra este voltaje a partir de una tensión continua. Este circuito se basa en el popular regulador de tensión 7805 (figura 2-5).

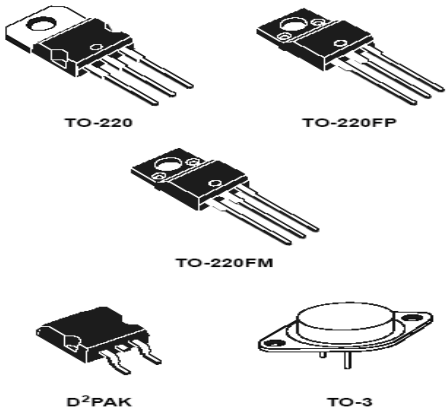

Figura 2-5 Tipos de empaquetados para el regulador de tensión 7805

Este circuito regulador es únicamente para voltajes positivos y dispone de tres terminales (figura 2-6), y su voltaje de salida es fijo, a si que si queremos disponer de 5 volts, este circuito nos los puede proporcionar.

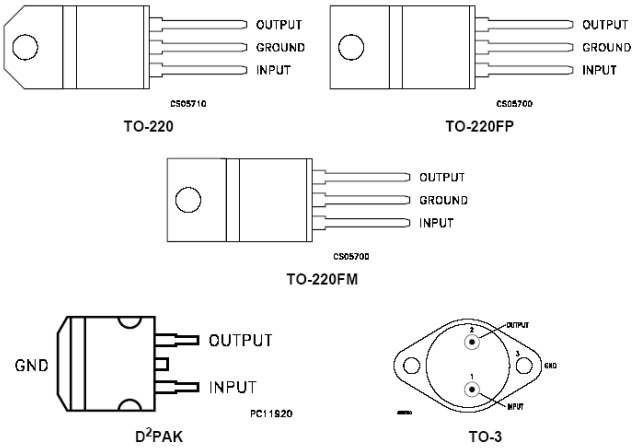

Figura 2-6 Terminales para el regulador de tensión 7805

Este regulador nos puede proporcionar la regulación necesaria para la alimentación de nuestro circuito, eliminando los problemas asociados con la distribución de tensión. Este circuito a la vez nos permite liberar hasta 1 A en la corriente de salida de este dispositivo regulador.

El consumo de corriente para el funcionamiento del microcontrolador depende de la tensión de alimentación, de la frecuencia de trabajo y de las cargas que soporten sus salidas, siendo del orden de unos pocos miliamperes.

Este regulador de voltaje (figura 2-7) será conectado de manera que el capacitor a la entrada reduzca considerablemente el rizado de la tensión de entrada que finalmente, el regulador 7805 se encargará de estabilizar a los 5 volts de alimentación.

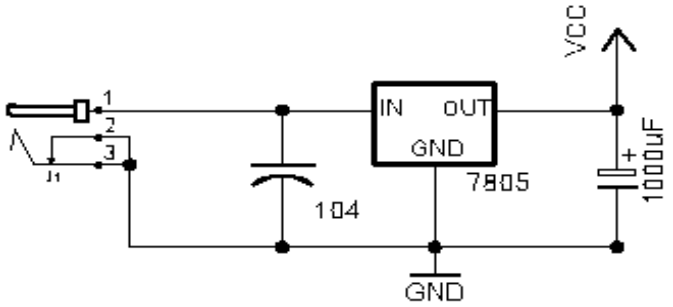

Figura 2-7 Conexión del regulador 7805

El circuito de alimentación del microcontrolador debe tratarse como el de cualquier otro dispositivo digital, debiendo conectarse un capacitor de desacoplo de unos 100nF, lo más cerca posible de los pines de alimentación.

2.7. PUERTOS DE ENTRADA/SALIDA

El microcontrolador se comunica con el mundo exterior a través de los puertos. Estos puertos están constituidos a su vez por líneas digitales de entrada/salida que trabajan entre 0 y 5 Volts. Estos puertos se pueden configurar como entradas, para recibir datos, o como salidas, para gobernar dispositivos externos.
El AVR ATmega8 tiene tres puertos, tal como se observa en la figura 2-8.

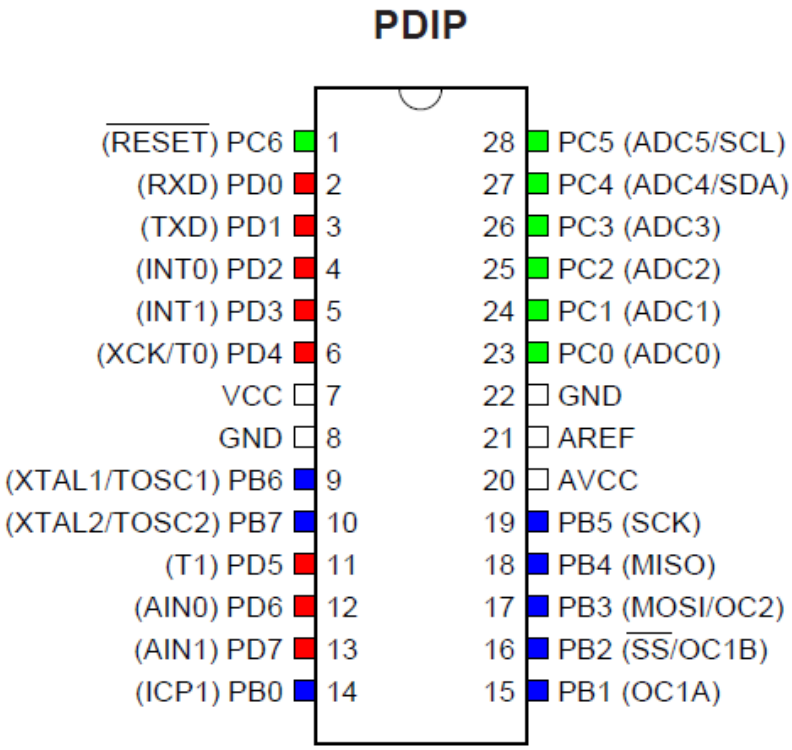

Figura 2-8 AVR ATmega8 configuración de pines de I/O

Estos puertos son:

- El Puerto B con 8 líneas, pines PB0 a PB7.
- El Puerto C con 7 líneas, pines PC0 a PC6.
- El Puerto D con 8 líneas, pines PD0 a PD7.

Cada línea puede ser configurada como entrada o como salida, independientemente unas de otras, según sea programado este dispositivo.

Las líneas son capaces de entregar niveles TTL cuando la tensión de alimentación aplicada en Vcc es de 5 Volts. La máxima capacidad de corriente de cada una de ellas es:

- 20 mA, cuando el pin está a nivel bajo, es decir, cuando consume corriente (modo *sink*). Sin embargo, la suma para todas las líneas de los puertos no debe exceder los 400mA, la suma de las líneas C0-C5 del puerto C no deben exceder los 200mA y las líneas de los puertos B0-B7,C6,D0-D7 y XTAL2 no devén exceder los 100mA
- 20 mA, cuando el pin está a nivel alto, es decir, cuando proporciona corriente (modo *source*). Sin embargo, la suma para todas las líneas de los puertos no debe exceder los 400mA, la suma de las líneas C0-C5 del puerto C no deben exceder los 100mA y las líneas de los puertos B0-B7,C6,D0-D7 y XTAL2 no devén exceder los 100mA

2.8. OSCILADOR

Todo microcontrolador requiere de un circuito que le indique la velocidad de trabajo, a este circuito se le llama oscilador o reloj. Este genera una onda cuadrada de alta frecuencia que se utiliza como señal para sincronizar todas las operaciones del sistema. Este circuito es muy simple pero de vital importancia para el buen funcionamiento del sistema. Generalmente todos los componentes del reloj se encuentran integrados en el propio microcontrolador y tan solo se requieren unos pocos componentes externos, como un cristal de cuarzo o una red RC, para definir la frecuencia de trabajo.

En el AVR ATmega8L, cuenta con diferentes configuraciones de trabajo para el oscilador:

- **External Crystal/Ceramic Resonator** Cristal externo ó resonador cerámico
- **External Low-frequency Crystal** Cristal externo de baja frecuencia

- **External RC Oscillator** Oscilador con resistencia y capacitor
- **Internal RC Oscillator** Oscilador interno con resistencia y capacitor
- **External Clock** Señal externa de reloj

2.8.1. OSCILADOR External Crystal/Ceramic Resonator

El oscilador de cristal ó resonador cerámico trabaja a una frecuencia comprendida entre 4MHz y 8MHz para el AVR ATmega8L, cuando se usa esta configuración se tiene que hacer uso de los pines **XTAL1/TOSC1** y **XTAL2/TOSC2** (figura 2-9) que son las líneas usadas para este fin, pero tiene la desventaja de que estos pines forman parte del puerto B, por lo que si se programa el dispositivo en esta función se reducirá el número de I/O del puerto B.

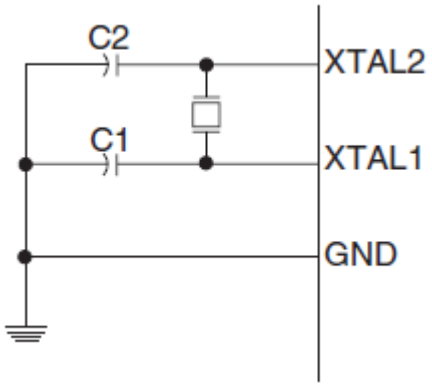

Figura 2-9 Conexión del oscilador

Para el valor de C1 y C2 debe de ser el mismo, este valor depende del cristal o resonador que se está usando, para la elección de los capacitores se puede ver la tabla 2-1

Tabla 2-1

CKOPT	CKSEL3..1	Frequency Range(MHz)	Recommended Range for Capacitors C1 and C2 for Use with Crystals (pF)
1	101(1)	0.4 - 0.9	–
1	110	0.9 - 3.0	12 - 22
1	111	3.0 - 8.0	12 - 22
0	101, 110, 111	1.0 ≤	12 - 22

2.8.2. OSCILADOR External Low-frequency Crystal

El oscilador de baja frecuencia, es un oscilador de bajo consumo el cual debe ser conectado como en la figura 1-10,trabajando entre las frecuencias entre 4MHz y 8MHz, si se programa el dispositivo en este modo se pueden permitir que los condensadores internos XTAL1 y XTAL2 funcionen, para eliminar así los condensadores externos. Los condensadores internos tienen un valor nominal de 36pF.

2.8.3. OSCILADOR External RC Oscillator

Es un oscilador de bajo costo formado por una red RC (figura 2-10). Su principal inconveniente es la baja precisión, pero como contrapartida está su bajo precio, que lo hace interesante para muchas aplicaciones en las que no importa la exactitud de tiempos.

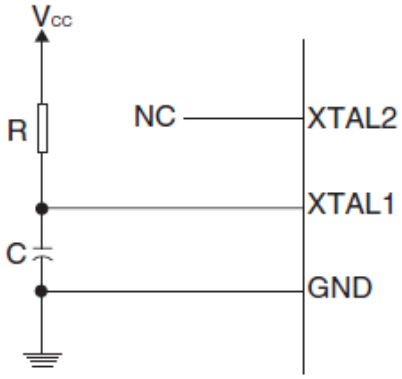

Figura 2-10 conexión del oscilador RC

La frecuencia de trabajo es estimada por la ecuación ***f=1/(3RC)***, donde C debe ser por lo menos de 22pF, los rangos de trabajo para este oscilador son de 1MHz-12MHz.

2.8.4. OSCILADOR Internal RC Oscillator

El AVR ATmega8L tiene una función de trabajar con un oscilador interno de RC el cual puede trabajar a 1.0MHz, 2.0MHz, 4.0MHz ó 8.0MHz, lo cual hace que opere sin componentes externos y **PB6 (XTAL1/TOSC1)** y **PB7 (XTAL2/TOSC2)** se pueden usar como I/O.

2.8.5. RELOJ EXTERNO External Clock

Esta posibilidad suele ser utilizada para hacer funcionar varios microcontroladores a partir de una única señal de reloj (figura 2-11). Esta configuración se utiliza en pocas ocasiones.

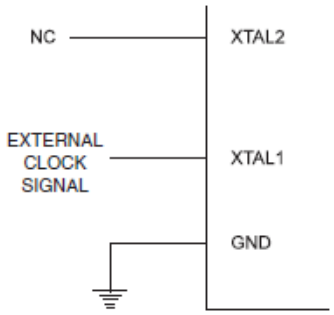

Figura 2-11 conexión de señal externa de reloj

2.9. PERIFÉRICOS BÁSICOS

2.9.1. LED

El LED (*Light Emisor Diode*) es un dispositivo que permite comprobar el funcionamiento de los circuitos, de forma cómoda, mediante la emisión de luz. Es barato y fácil de conectar a la salida de un microcontrolador. Se polariza en directo con una tensión en extremos ente 1.2 y 2.2 V (volts), según el modelo, y sólo requiere de 5 a 30 mA para su encendido.

En el AVR ATMega8 es posible manejar directamente los LED de dos formas distintas. Figura 2-12

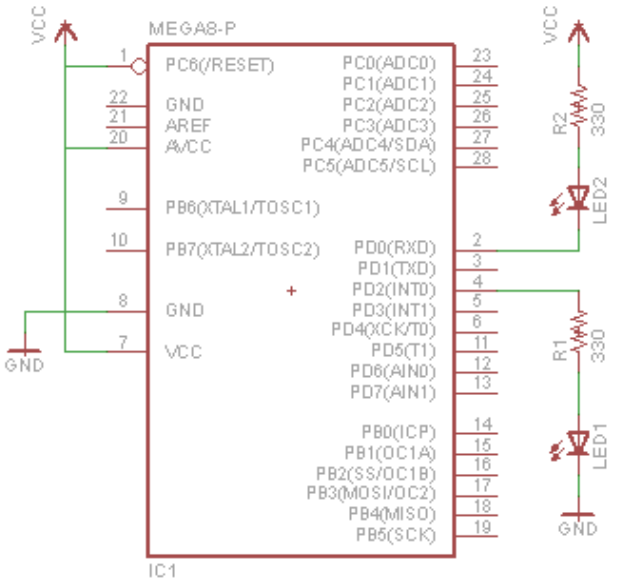

Figura 2-12 Formas de conectar un LED a un microcontrolador

Estas formas son:

- Conectando el cátodo del diodo a la salida del microcontrolador y el ánodo al positivo de la alimentación a través de una resistencia limitadora, como el LED2 de la figura 2-12. En donde, en este caso, el LED2 se iluminará solamente con un nivel bajo de salida (0 Volts).

- Conectando el ánodo del diodo a la salida del microcontrolador, a través de una resistencia limitadora, y el cátodo a tierra, como el del LED1 de la figura 2-12. En donde, en este caso, el LED1 se iluminará con un nivel alto de salida (5 Volts).

En este tipo de casos en donde se tienen que usar indicadores como lo son los LED, tenemos que optar por poner una resistencia, ya que esta hará la función de limitar la corriente a un valor adecuado para hacer que encienda el LED.El valor de la resistencia deberá tener un valor comprendido entre 220Ω y 330Ω. En la figura 2-12 se puede observar que se ha elegido un valor de 330Ω, que limita la corriente a un valor de unos 10 mA, lo cual nos proporcionará una luminosidad suficiente para la mayoría de las aplicaciones. Si se opta por que emita más luz, se puede bajar el valor de la resistencia a 220Ω.

2.9.2. INTERRUPTORES Y PULSADORES

Estos dispositivos nos permiten introducir un nivel lógico “0” ó “1”, según la posición en la que se encuentren, “cerrado” o “abierto”.

Para poder leer el estado de interruptores y pulsadores (figura 2-13 (a)) solamente basta con conectar estos dispositivos entre una entrada y tierra.

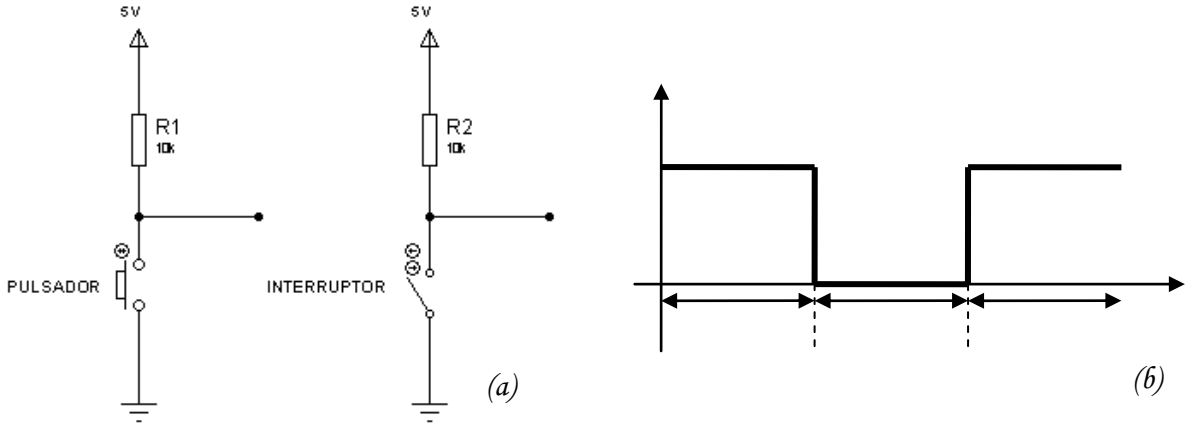

Figura 2-13: (a) pulsador e interruptor; (b) niveles de tensión de pulsador e interruptor

En la gráfica (figura 2-13 (b)) se puede observar que mientras el dispositivo se encuentra abierto, la entrada mantiene una tensión de 5 V que corresponde a un nivel lógico “1”.

Ahora que cuando se cierra, la entrada pasa a 0 V, que corresponde a un nivel lógico “0”.

También se pueden encontrar varios tipos de conmutadores, finales de carrera, detectores y sensores digitales con un funcionamiento similar a los pulsadores e interruptores.

2.9.3. ENTRADAS DIGITALES CON OPTOACOPLADORES

Alguna vez, nos vemos en la necesidad de utilizar como entradas señales de alta tensión o señales relacionadas con la tensión de la red eléctrica. Estas tensiones no se pueden aplicar directamente al microcontrolador, y es necesario aislar eléctricamente el circuito mediante un optoacoplador con un montaje como el de la figura 2-14.

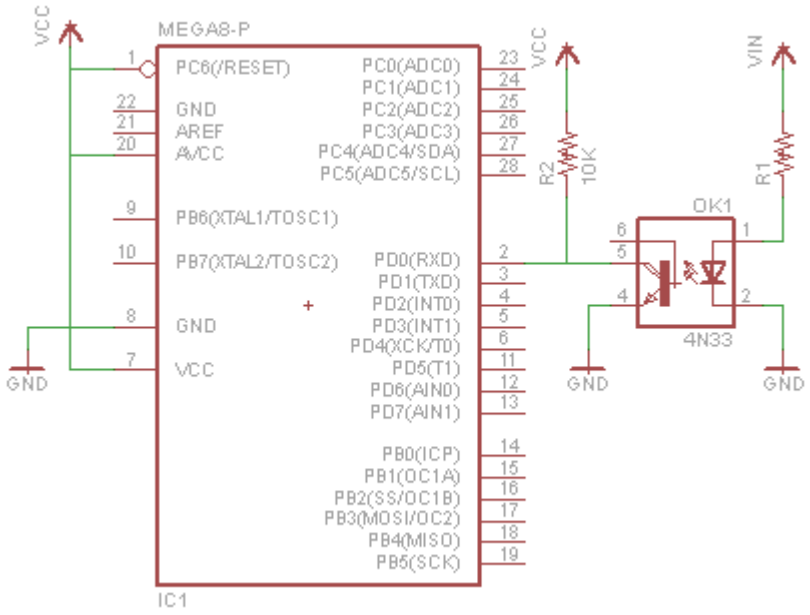

Figura 2-14 Gobierno de una entrada mediante optoacoplador 4N33

En estos casos se puede usar un optoacoplador como lo es el 4N33, el cual viene en un encapsulado DIL06, en donde se encierra un LED y un fototransistor en configuración Darlington. Su funcionamiento se puede describir de esta manera:

- Cuando se aplica una tensión $\mathbf{V_{IN}}$, circula una corriente por el LED del optoacoplador, emitiendo un haz de luz que incide sobre el transistor y lo satura. En este caso, a la entrada del microcontrolador se aplica un nivel bajo, igual que cuando estaba cerrado el interruptor de la figura ANTERIOR.

- Cuando no se aplica alguna tensión al LED, este se encuentra apagado, bloqueando así al transistor. Entonces a la entrada del microcontrolador se está aplicando un nivel alto, igual que cuando está abierto un interruptor (figura ANTERIOR (b)).

La tensión que necesita un LED en conducción es de 1.25 V, y para que se ilumine, hay que hacer circular una corriente de unos 15 mA La resistencia en serie con el LED debe permitir que circule esta intensidad.

El LED soporta una tensión máxima inversa de sólo 3 V. Ahora que para aplicaciones de corriente alterna, hay que conectar en paralelo con el LED un diodo de protección en inverso, o de otra forma utilizar un optoacoplador que ya lo lleve integrado como el H11A1.

Con esta forma de conexión de entradas digitales por medio de optoacopladores, se asegura que los dos circuitos se encuentren eléctricamente aislados; en donde la única comunicación entre ambos es la luz que emite el LED.

2.10. ORGANIZACION DE LA MEMORIA

2.10.1. ARQUITECTURA INTERNA DEL AVR-ATmega8

Como se puede observar en la tabla 2-2 se pueden destacar las siguientes características:

- Memoria FLASH 8 Kbyte
- Memoria SRAM de 1Kbytes
- Memoria EEPROM de 512 bytes
- 23 puertos de I/O

Product	Flash (KB)	EEPROM (Bytes)	RAM (Bytes)	I/O	SPI	USART	USI	TWI	PWM	On-Chip Debug		10-bit ADC	LCD
										JTAG	debugWire		
megaAVR													
ATmega 48	4	256	512	23	1	1	–	1	5	–	Y	8	–
ATmega8	8	512	1K	23	1	1	–	1	3	–	–	8	–
ATmega88	8	512	1K	23	1	1	–	1	5	–	Y	8	–
ATmega8515	8	512	512	35	1	1	–	–	3	–	–	–	–
ATmega8535	8	512	512	32	1	1	–	1	4	–	–	8	–

Tabla 2-2 Características de los dispositivos ATmega AVR

Dentro del ATmega8 se pueden mencionar 3 bloques de memoria:

- **MEMORIA DE PROGRAMA.** En donde en sus 8191 posiciones, contiene el programa con las instrucciones que gobiernan la aplicación. Es del tipo no volátil, es decir, el programa se mantiene aunque desaparezca la alimentación.

- **MEMORIA EEPRON DE DATOS.** Es una pequeña área de memoria de datos de lectura y escritura no volátil, gracias a la cual, un corte del suministro de la alimentación no ocasiona la pérdida de la información, que estará disponible al reiniciarse el programa. El espacio de EEPROM consta de 512 bytes.

- **MEMORIA DE DATOS SRAM.** Memoria para almacenar sus datos que son tratados durante el tiempo de ejecución (incluidos también los registros, pila, etc.) de memoria no volátil

2.10.2. MEMORIA DE PROGRAMA.

La memoria flash del Atmega8 tiene una capacidad de 8K x 8, las instrucciones son de 16 bits o 32 bits, por tanto esta memoria es organizado como 4Kx16 bits. Para mayor seguridad el espacio de la memoria flash(memoria de programas) está dividido en 2 áreas: arranque y de aplicación figura 1-16. Para acceder al espacio comprendido entre 0 y 4095 ($FFF) direcciones, el tamaño del registro contador de programa (PC) será de 12 bits.

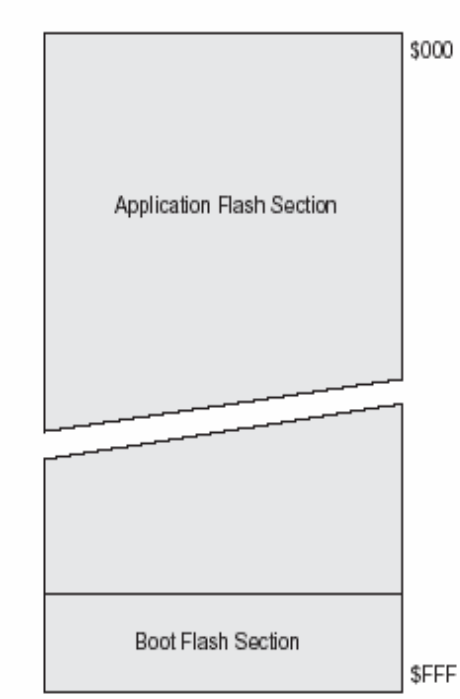

Figura 2-15 Memoria de Programas

2.10.3. MEMORIA DE DATOS SRAM.

En la memoria de datos residen los registros de propósito general (R0...R31), los registros de E/S y los registros de la SRAM interna.

La siguiente figura 2-16 muestra la organización de la memoria SRAM del Atmega8

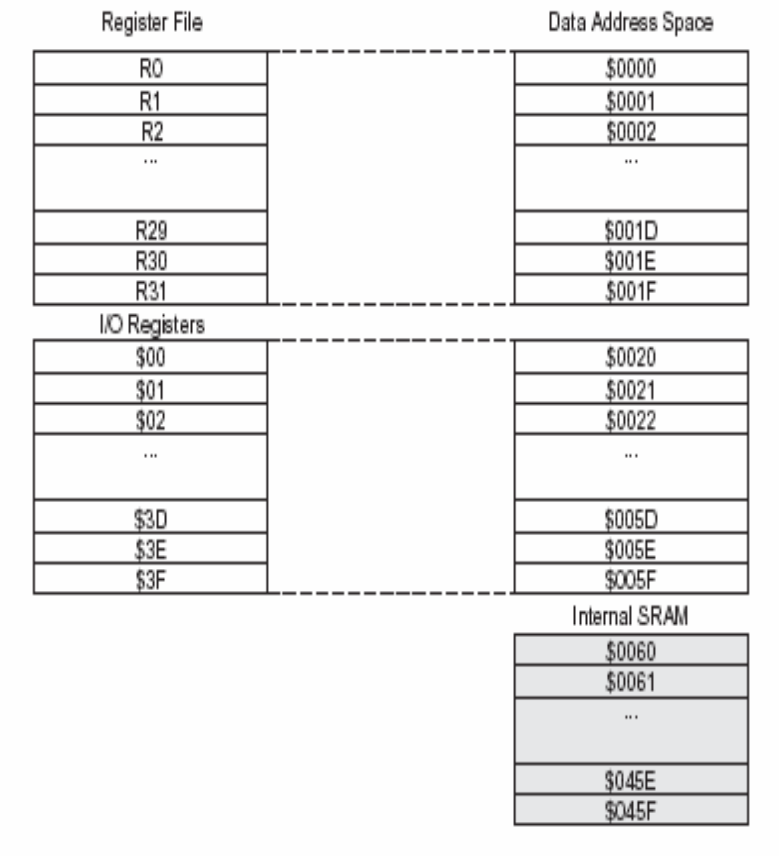

Figura 2-16 Mapa de la memoria de datos

- **REGISTROS DE PROPOSITO GENERAL**

El Atmega8(L), dispone de 32 registros de propósito general figura 2-17, Estos son registros de uso general que se pueden usar para guardar los datos temporales del programa que se esté ejecutando.

Se observa en la figura, además que a cada registro le corresponde una dirección dentro de las 32 primeras posiciones en el espacio de memoria de datos. Los registros R26..R31 tienen algunas funciones adicionales.

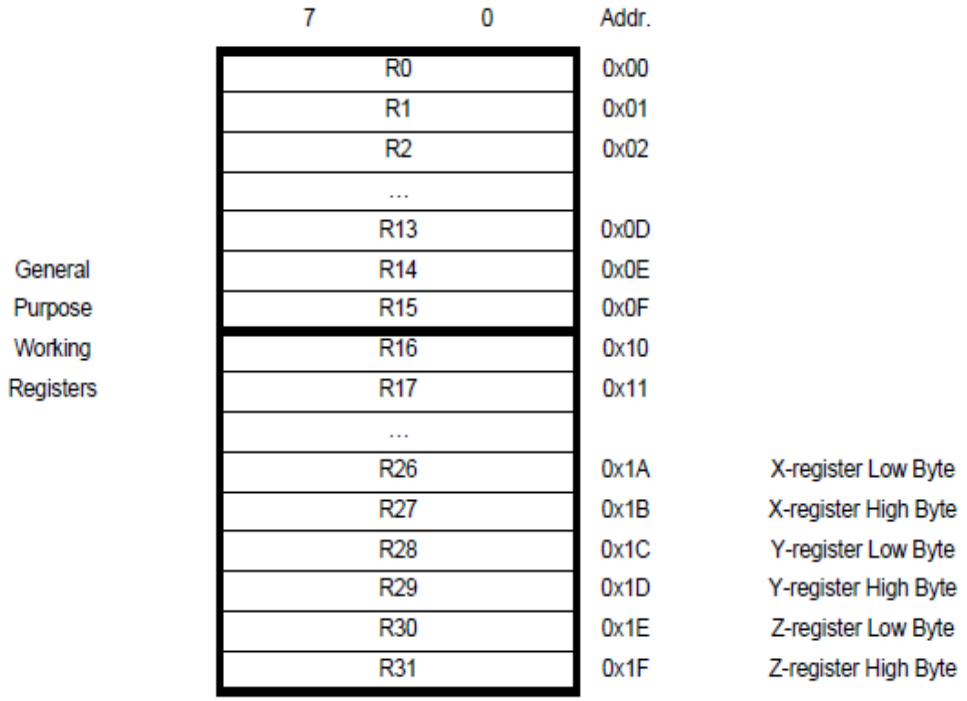

Figura 2-17 Registros de propósito general

Los Registros X, Y, Z**:** los registros R26 y R27 forman el registro X de 16 bits, los registros R28 y R29 forman el registro Y de 16 bits, los registros R30 y R31 forman el registro apuntador Z de 16 bits. Los tres registros se definen como se describe en la figura 2-18.

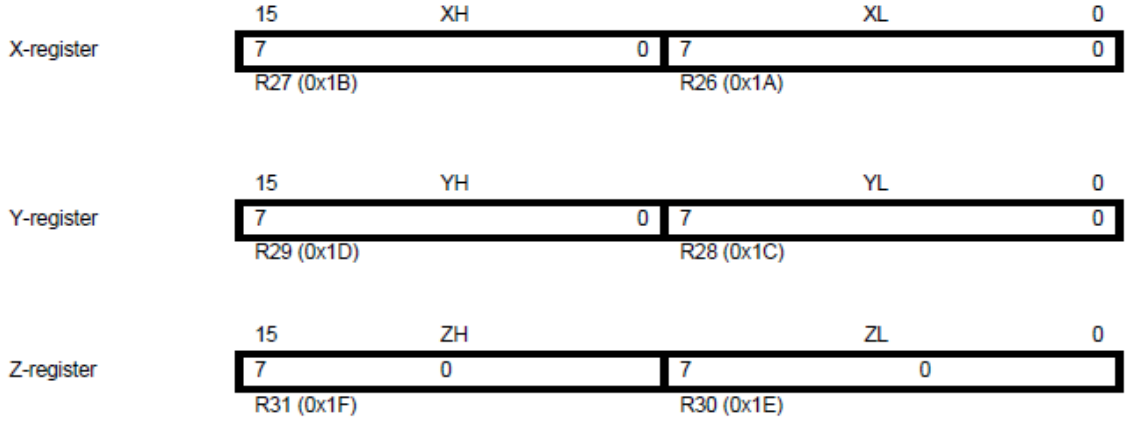

Figura 2-18 Registros X-,Y- y Z-

- **REGISTROS I/O**

Los registros de I/O son los que cumplen con un propósito especial en el control del microcontrolador que van de la dirección $0020…$005F como se muestra en la tabla 2-3.

Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	9
0x3E (0x5E)	SPH	–	–	–	–	–	SP10	SP9	SP8	11
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	11
0x3C (0x5C)	Reserved									
0x3B (0x5B)	GICR	INT1	INT0	–	–	–	–	IVSEL	IVCE	47, 65
0x3A (0x5A)	GIFR	INTF1	INTF0	–	–	–	–	–	–	66
0x39 (0x59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	70, 100, 120
0x38 (0x58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	71, 101, 120
0x37 (0x57)	SPMCR	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	210
0x36 (0x56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	168
0x35 (0x55)	MCUCR	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	31, 64
0x34 (0x54)	MCUCSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	39
0x33 (0x53)	TCCR0	–	–	–	–	–	CS02	CS01	CS00	70
0x32 (0x52)	TCNT0	Timer/Counter0 (8 Bits)								70
0x31 (0x51)	OSCCAL	Oscillator Calibration Register								29
0x30 (0x50)	SFIOR	–	–	–	–	ACME	PUD	PSR2	PSR10	56, 73, 121, 190
0x2F (0x4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	95
0x2E (0x4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	98
0x2D (0x4D)	TCNT1H	Timer/Counter1 – Counter Register High byte								99
0x2C (0x4C)	TCNT1L	Timer/Counter1 – Counter Register Low byte								99
0x2B (0x4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High byte								99
0x2A (0x4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low byte								99
0x29 (0x49)	OCR1BH	Timer/Counter1 – Output Compare Register B High byte								99
0x28 (0x48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low byte								99
0x27 (0x47)	ICR1H	Timer/Counter1 – Input Capture Register High byte								100
0x26 (0x46)	ICR1L	Timer/Counter1 – Input Capture Register Low byte								100
0x25 (0x45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	115
0x24 (0x44)	TCNT2	Timer/Counter2 (8 Bits)								117
0x23 (0x43)	OCR2	Timer/Counter2 Output Compare Register								117
0x22 (0x42)	ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	117
0x21 (0x41)	WDTCR	–	–	–	WDCE	WDE	WDP2	WDP1	WDP0	41
0x20[1] (0x40)[1]	UBRRH	URSEL	–	–	–	UBRR[11:8]				155
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	153
0x1F (0x3F)	EEARH	–	–	–	–	–	–	–	EEAR8	18
0x1E (0x3E)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	18
0x1D (0x3D)	EEDR	EEPROM Data Register								18
0x1C (0x3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	18
0x1B (0x3B)	Reserved									
0x1A (0x3A)	Reserved									
0x19 (0x39)	Reserved									
0x18 (0x38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	63
0x17 (0x37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	63
0x16 (0x36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	63
0x15 (0x35)	PORTC	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	63
0x14 (0x34)	DDRC	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	63
0x13 (0x33)	PINC	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	63
0x12 (0x32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	63
0x11 (0x31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	63
0x10 (0x30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	63
0x0F (0x2F)	SPDR	SPI Data Register								128
0x0E (0x2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	128
0x0D (0x2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	126
0x0C (0x2C)	UDR	USART I/O Data Register								150
0x0B (0x2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	151
0x0A (0x2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	152
0x09 (0x29)	UBRRL	USART Baud Rate Register Low byte								155
0x08 (0x28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	191
0x07 (0x27)	ADMUX	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	202
0x06 (0x26)	ADCSRA	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	204
0x05 (0x25)	ADCH	ADC Data Register High byte								205
0x04 (0x24)	ADCL	ADC Data Register Low byte								205
0x03 (0x23)	TWDR	Two-wire Serial Interface Data Register								170
0x02 (0x22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	170
0x01 (0x21)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	170
0x00 (0x20)	TWBR	Two-wire Serial Interface Bit Rate Register								168

Tabla 2-3 Mapa de Registros de Memoria I/O

2.10.4. UNIDAD CENTRAL DE PROCESOS (CPU) DE Atmega8(L)

La función de la CPU es controlar la operación del microcontrolador, permitiendo la ejecución correcta del programa, habilitando el acceso a las memorias, controlar los periféricos y manejar las interrupciones. El AVR, utiliza la arquitectura Harvard gracias a ella se puede acceder de forma simultánea e independiente a la memoria de datos y a la memoria de instrucciones.
Los registros de archivo de acceso rápido contienen 32 registros de propósito general de 8 bits y son accesados con un solo ciclo de reloj. Esto permite usar un solo ciclo en una operación aritmética-lógica(ALU), por ejemplo en una operación típica ALU, dos operandos son extraídos del archivo de Registros, y la operación es ejecutada, el resultado es almacenado de nuevo en el Registro de archivos(Rd), en un solo ciclo de reloj.

Rd _ Rd + Rr

De los 32 registros de 8 bits 6 pueden ser usados como 3 registros de 16 bits para direccionamiento indirecto apuntando el espacio de memoria de datos. Estos registros adicionales son X, Y, y Z, de 16 bits. La ALU, permite realizar las operaciones aritméticas y lógicas entre registros o entre un registro y una constante. Luego de una operación aritmética, el registro de estados es modificado para reflejar la información acerca del resultado de una operación. El flujo de un programa es variado por los saltos condicionales e incondicionales y las instrucciones de llamada para dirigirse a la nueva dirección en el espacio de memoria. Las instrucciones del AVR, tiene normalmente un formato de compuesto por una palabra de 16 bits. Cada dirección en la memoria de programas contiene instrucciones de 16 ó 32 bits

2.10.5. PUNTERO DE PILA

La pila es el área de espacio de memoria, utilizado temporalmente para guardar y recuperar datos y/o direcciones cuando el CPU está ejecutando una subrutina programada o una interrupción. El puntero de pila apunta (almacena la dirección) a la próxima dirección libre de la pila y que decrementa su valor en uno cada vez que se almacena un dato(de un byte) en ella, incrementándolo en uno cuando se retira este valor(de un byte). En el caso que se guarda la dirección de retorno cuando se atiende a una subrutina o una interrupción el puntero de pila es decrementado en 2 y luego de ejecutarse la última instrucción de retorno de la subrutina, el puntero de pila es incrementado en 2.

El puntero de pila en el AVR es implementado con 2 registros de 8 bits en el espacio de memoria E/S. El programador debe darle el valor inicial al puntero de pila en la memoria de datos SRAM por encima de la dirección $60 figura 2-19.

Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	**SPH**
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	**SPL**
	7	6	5	4	3	2	1	0	

Figura 2-19 Registro de Puntero de Pila

2.11. RECURSOS ESPECIALES DEL Atmega8(L).

En esta parte es fundamental mencionar que como cada fabricante de microcontroladores ofrece numerosas versiones de una arquitectura; como por ejemplo en algunas versiones se amplía la capacidad de la memoria, en otras incorporan nuevos recursos, el AVR Atmega8(L) que cuenta con:

- Dos Timer/Contadores de 8 bits con prescaler separado y modo comparación.
- Un Timer/Contador de 16 bits con prescaler separado, modo comparación y modo de captura
- Comparador analógico On-Chip.
- Timer watchdog programable con oscilador separado On-Chip.
- Interface serie SPI maestro/esclavo.
- USART serie programable.
- Contador en tiempo real con oscilador separado.
- ADC de 6 canales en el encapsulado PDIP.
4 canales de 10 bits de precisión.
2 canales de 8 bits de precisión.
- 3 canales de PWM.

Con estas herramientas se va a poner solución a este trabajo, claro está que, no todos estos recursos se van a ocupar, pero es el microcontrolador que más se ajusto a los requerimientos.

A continuación, en los siguientes subcapítulos se mencionan las herramientas de este microcontrolador que se ocuparon.

2.12. MÓDULO DEL CONVERTIDOR A/D

Como es muy frecuente el trabajo con señales analógicas, éstas deben ser convertidas a digital, y por ello muchos microcontroladores incorporan un conversor A/D (analógico-digital), el cual se utiliza para tomar datos de varias entradas diferentes que se seleccionan mediante un multiplexor. Figura 2-20.

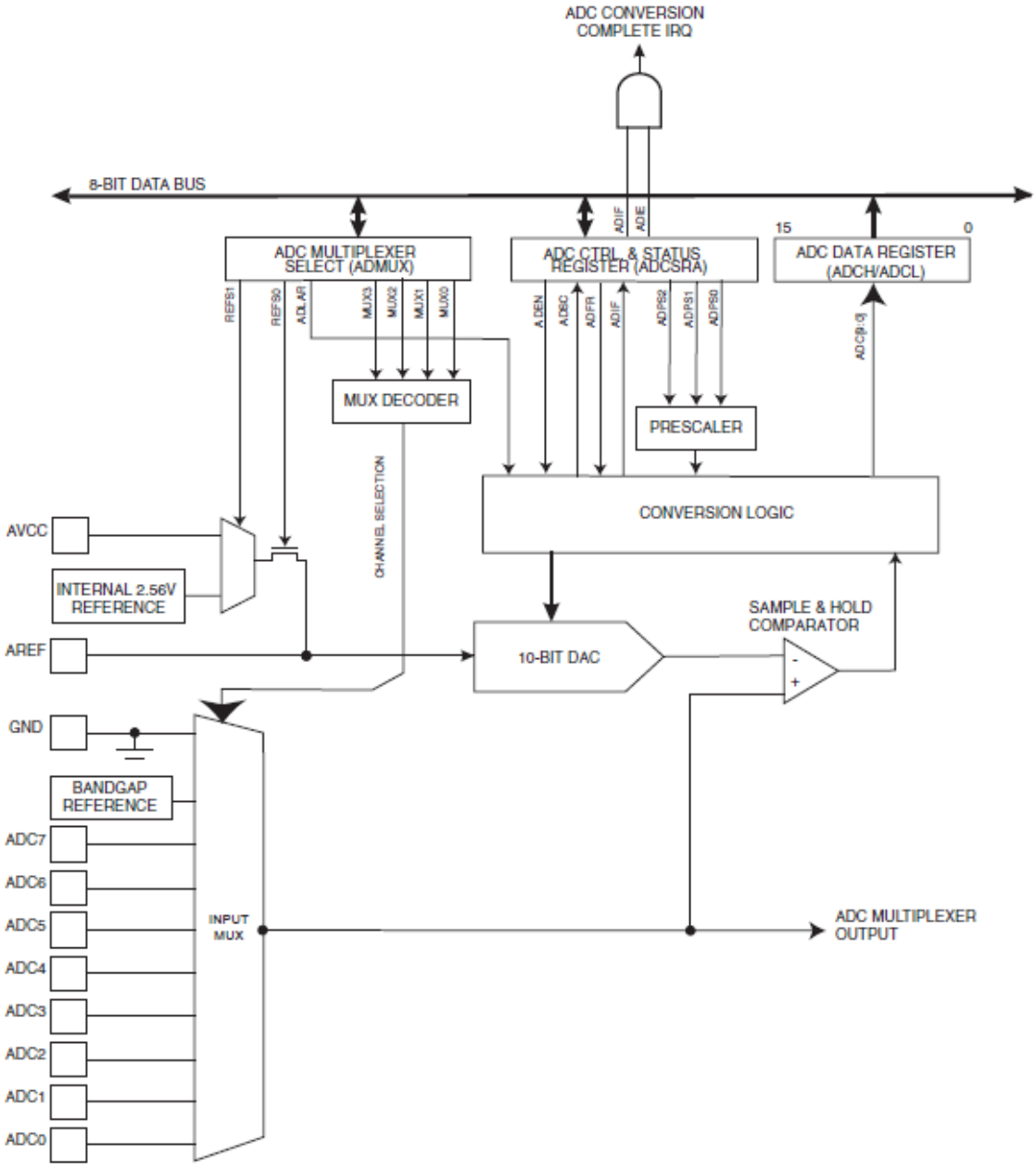

Figura 2-20 Diagrama a bloques A/D

Las resoluciones más frecuentes son 8 y 10 bits, aunque hay microcontroladores con conversores de 11 y 12 bits; para resoluciones mayores es preciso utilizar conversores A/D externos. Los conversores A/D son uno de los periféricos más codiciados en el mundo de los microcontroladores este tiene una resolución de conversión de 10bits de hasta 15kSPS, (Kilo Samples Per Second), 6 canales analógicos, ajuste a derecha e izquierda del resultado de la conversión, voltajes de referencia, 2 modos de conversión: simple y continuo.

Esto no quiere decir que cuente con 6 convertidores A/D si no que en realidad se trata de uno solo, que se puede multiplexar en seis entradas como se muestra en la figura 1-21. Este modulo se puede configurar por medio de programación.

La técnica que utiliza el µC (microcontrolador) para la conversión es la de aproximaciones sucesivas se basa en la realización de comparaciones sucesivas de manera descendente o ascendente, hasta que se encuentra la combinación que iguala la tensión entregada por el D/A y la de entrada. El rango de conversión es de 0 a 5 V, pero si hubiera que hacer alguna conversión de más voltaje, bastará con poner a la entrada del conversor un divisor de tensión correctamente calculado, o bien trabajar con alguna tensión de referencia externa al µC.

La resolución que tiene cada bit de la conversión tiene un valor que es función de la tensión de referencia externa (en caso que la hubiere), y viene dada por:

$$\mathrm{Res} = \frac{\mathbf{V_{ref+}} - \mathbf{V_{ref-}}}{\mathbf{1024}}$$

Con lo cual, por ejemplo, si la tensión de referencia positiva (Vref+) es de 5 V y la tensión de referencia negativa (Vref -) es tierra, la resolución por cada bit es de 4,8 mV por cada bit. Una vez realizada la conversión que se obtiene con:

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

Obtendremos un valor binario *0000000000* para 0 V y un valor binario *1111111111* para 5 V.

2.12.1. CONTROL DEL CONVERTIDOR A/D

Todos los módulos del AVR tienen registros asociados para su control. Para controlar el convertidor A/D (analógico/digital), se va a necesitar usar los siguientes registros:

- **ADMUX – ADC Multiplexer Selection Register – ADMUX**
- **ADCSRA – ADC Control and Status Register A**
- **ADCL and ADCH – The ADC Data Register**

Como la resolución del CAD (convertidor A/D) es de 10 bits y los registros del µC son de 8 bits, se utilizan dos registros; el ADCL y ADCH en forma concatenada (que tiene relación). Es decir, en uno de ellos usaremos los 8 bits completos y en el otro solo 2 bits para llegar a los 10. El registro ADMUX es el que nos permitirá controlar, configurar y poner en marcha nuestro convertidor A/D.

ADMUX – ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Registro 2-1 ADMUX

Bit 7:6 – REFS1:0: Reference Selection Bits: Sé utilizan para seleccionar el tipo de voltaje de referencia que se usará, si interno, externo o deshabilitado, según la tabla 2-4.

Table 22-2. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Tabla 2-4 Selección de voltaje de referencia del ADC

Bit 5 – ADLAR: ADC Left Adjust Result: Configurado con 1, ajustará el resultado de la conversión en los registros ADCH y ADCL a la Izquierda, caso contrario con un 0 lo hará a la derecha, esto lo veremos más adelante en la figura 2-21-(a)-(b).

Bits 3:0 – MUX3:0: Analog Channel Selection Bits: Estos bits seleccionarán el canal analógico que se usará en la presente conversión, de acuerdo a la tabla 2-5

Table 22-3. Input Channel Selections

MUX3:0	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1110	1.30V (V_{BG})
1111	0V (GND)

Tabla 2-5 Selección del canal analógico

ADCSRA – ADC Control and Status Register A

En este registro se hacen las demás configuraciones y también se tiene los bits de control de inicio/fin de conversión.

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Registro 2-2 ADCSRA

Bit 7 – ADEN: ADC Enable: Es la habilitación del ADC para su uso (1), caso contrario el ADC estará apagado (0)

Bit 6 – ADSC: ADC Start Conversion: Colocando un 1 en este bit se inicia la conversión analógica, ya sea en modo normal o en modo continuó, salvo que en el continuó no hará falta volverlo a colocar a 1 cuando se necesite otra conversión, lo que si se debe hacer en modo normal, ya que el conversor en este modo coloca a cero por hardware a este bit cuando la conversión AD se ha terminado.

Bit 5 – ADFR: ADC Free Running Select: Con 1 se habilita el modo continuó (free running), y el ADC muestreará y actualizará los registros de datos de la conversión continuamente, colocando un 0 el modo continuó se detiene.

Bit 4 – ADIF: ADC Interrupt Flag: Este bit se coloca a 1 cuando la conversión AD es completada y los registros de datos de la conversión son actualizados. Se ejecutará la rutina de interrupción si esta implementada y se tiene configurada las máscaras correspondientes para esta interrupción y el bit de

Interrupciones globales, y se pondrá a 0 por hardware al terminar la rutina de interrupción.

Bit 3 – ADIE: ADC Interrupt Enable: Es la máscara de habilitación para la interrupción por Conversión AD completada, se la habilita colocando un 1.

Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits: Estos bits determinan el divisor entre la frecuencia del Oscilador principal y la entrada de reloj del ADC, es según la tabla 1-6.

Table 22-4. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Tabla 2-6 Selección del Preescalador.

ADCL and ADCH – The ADC Data Register

Por último los registros de datos donde se almacena el resultado de 10bits de la conversión A/D, que son de dos formas según la justificación configurada en ADLAR.

ADLAR=0

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Figura 2-21:(a) Registro ADLAR

ADLAR=1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Figura 2-21:(b) Registro ADLAR

Cuando la conversión A/D ha sido completada, el resultado puede ser encontrado en éstos 2 registros, en el formato que ADLAR indique, primero se debe leer ADCL y posteriormente ADCH. Si la resolución es de 8 bits se puede usar ADLAR=1 y leer directamente ADCH como se menciona anteriormente.

2.12.2. TIEMPO DE CONVERSIÓN A/D

Por default, la circuitería de aproximaciones sucesivas requiere una frecuencia de reloj de entrada entre 50kHz y 200kHz para obtener la máxima resolución. Si una resolución más baja de 10 bits se requiere, la frecuencia de reloj de entrada al ADC puede ser mayor de 200kHz para obtener una razón de muestreo mayor.

El módulo ADC contiene un preescalador, el cual genera una frecuencia de reloj aceptable para el ADC de cualquier frecuencia del CPU arriba de 100kHz. El preescalador se activa por los bits ADPS en ADCSRA. El preescalador inicia su cuenta desde el momento en que el ADC se enciende colocando el bit a uno en ADEN de ADCSRA. El preescalador se mantiene corriendo mientras el ADEN este en uno, y se reinicia cuando ADEN este en bajo.

Una conversión normal toma 13 ciclos de reloj del ADC. La primera conversión después de que el ADC se enciende toma 25 ciclos de reloj del ADC.
El muestreo y retención actual toma lugar 1.5 ciclos de reloj del ADC después del inicio de una conversión normal y 13.5 ciclos de reloj del ADC después del comienzo de la primera conversión. Cuando una conversión se completa, el resultado se escribe en el registro de datos del ADC, y ADIF se activa. El software puede poner a uno ADSC de nuevo, y una nueva conversión será iniciada en la primera transición positiva del ciclo de reloj del ADC.

En modo libre, una nueva conversión será iniciada inmediatamente después de que la conversión se completa, mientras ADSC permanece en alto.

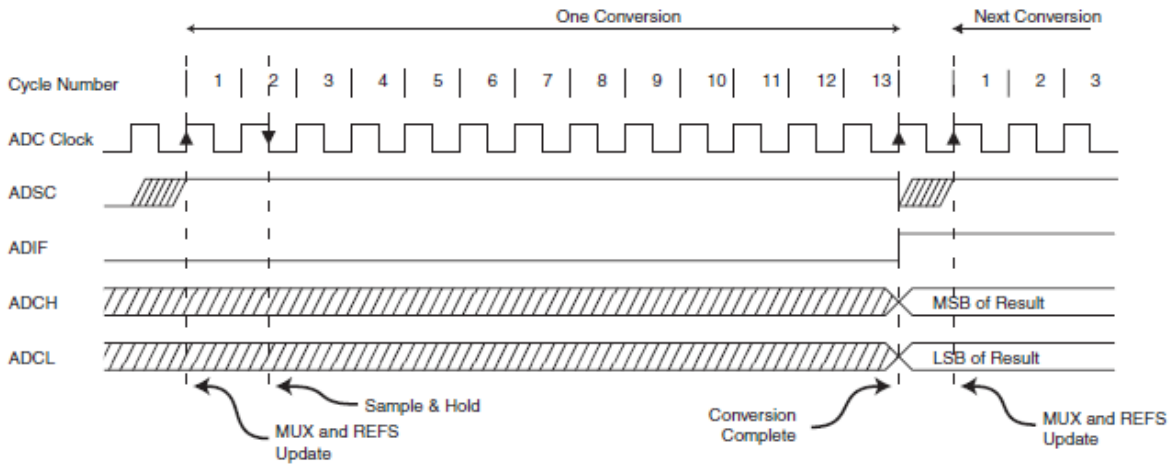

Figura 2-22 Diagrama de tiempos del ADC, Conversión Única.

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
Extended conversion	13.5	25
Normal conversions, single ended	1.5	13

Tabla 2-7 Tiempo de conversión del ADC

2.13. MODULO USART

Una de las formas más común y sencilla de comunicar cualquier dispositivo con una computadora es a través de su puerto serie, que es compatible con el denominado estándar RS232 *(o EIA232 Standard)*. En una computadora puede haber varios puertos seriales, normalmente denominados COM1, COM2, etc.

Los puertos serie son accesibles mediante conectores. La norma RS232 establece dos tipos de conectores, llamados DB-25 (de 25 pines) y DB-9 (de 9 pines), machos y hembras, como se puede ver en la figura 2-23. La norma RS232 se estableció para conectar una computadora con un módem, por lo que aparecen muchas patillas en los conectores DB-25 que en otro tipo de aplicaciones no se utilizan y en las que es más común utilizar el conector tipo DB-9.

Figura 2-23 Conectores DB-25 y DB-9

Cada uno de los pines del conector RS232 tiene una función especificada por la norma. Hay unos pines por los que se transmiten y reciben datos, y otros que controlan el establecimiento, flujo y cierre de la comunicación.
Para comunicarse con un microcontrolador es suficiente con tres líneas:

- Línea de transmisión (TxD), pin 3 *(Transmitted Data).*
- Línea de recepción (RxD), pin 2 *(Received Data).*
- Pin de tierra (SG), pin 5 *(Signal Ground).*

En este microcontrolador, la USART *(Universal Synchronous Asynchronous Receiver Transmitter)* es uno de los dos módulos seriales de entrada o salida. La USART puede ser configurada de modo que trabaje como un sistema asíncrono full-dúplex, para así poder comunicarse con otros dispositivos periféricos, como lo pueden ser las computadoras personales. Otra forma de configurar esta USART es de modo que funcione como un sistema síncrono full-dúplex, para poder comunicarse con otros dispositivos periféricos, como lo pueden ser los circuitos integrados que hagan la función de un convertidor A/D o D/A, memorias EEPROM, etc.

La USART puede ser configurada de los siguientes modos:

- *Asíncrono.*
- *Síncrono - Maestro.*
- *Síncrono - Esclavo.*

Para poder ser configurada de acuerdo a los modos de trabajo anteriores, se necesitan ver cuáles son los registros asociados a la USART.

2.13.1. REGISTROS DE CONTROL PARA EL MODULO USART

Antes de mencionar los registros de control del modulo USART se mencionaran los parámetros más importantes para poder realizar una transmisión serial en nuestro microcontrolador que son:

- Cantidad de bits de inicio.
- Cantidad de bits de parada.
- Bit de paridad.
- Tasa de Transmisión (Baud Rate)

***Bit de inicio*:** Es usado para indicar el inicio de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits.

***Bit de parada*:** Usado para indicar el fin de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits. Debido a la manera como se transfiere la información a través de las líneas de comunicación y que cada dispositivo tiene su propio reloj, es posible que los dos dispositivos no estén sincronizados. Por lo tanto, los bits de parada no sólo indican el fin de la transmisión sino además dan un margen de tolerancia para esa diferencia de los relojes.

***Bit de paridad*:** Es una forma sencilla de verificar si hay errores en la transmisión serial. Existen cuatro tipos de paridad: par, impar, marcada y espaciada. La opción de no usar paridad alguna también está disponible. Para paridad par e impar, el puerto serial fijará el bit de paridad (el último bit después de los bits de datos) a un valor para asegurarse que la transmisión tenga un número par o impar de bits en estado alto lógico. La paridad marcada y espaciada en realidad no verifican el estado de los bits de datos; simplemente fija el bit de paridad en estado lógico alto para la marcada, y en estado lógico bajo para la espaciada.

***Tasa de Transmisión (Baud Rate)*:** Indica el número de bits por segundo que se transfieren, y se mide en baudios (*bauds*). Por ejemplo, 300 baudios representan 300 bits por segundo. Cuando se hace referencia a los ciclos de reloj se está hablando de la velocidad de transmisión. Por ejemplo, si el protocolo hace una llamada a 4800 ciclos de reloj, entonces el reloj está corriendo a 4800 Hz, lo que significa que el puerto serial está muestreando las líneas de transmisión a 4800 Hz.

Los registros que intervienen para que nuestro microcontrolador pueda funcionar en el modo USART son:

- **UCSRA (USART Control and Status Register A)**
- **UCSRB (USART Control and Status Register B)**
- **UCSRC (USART Control and Status Register C)**

- **UDR (I/O Data Register)**
- **UBRRL-UBRRH (USART Baud Rate Register)**

USART Control and Status Register A – UCSRA

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Registro 2-3 UCSRA

BIT 7-RXC: USART Receive Complet.

Este registro es el que monitorea si ya se recibió y fue leído el dato; la bandera se pondrá a 1 si el dato fue recibido y no leído, y se pondrá a 0 cuando el dato ya fue leído.

BIT 6-TXC: USART Transmit Complet.

Este registro monitorea si la transmisión llega a su fin o no; la bandera se pondrá a 1 cuando la transmisión haiga llegado a su fin y se pondrá en 0 cuando esto no ocurra.

BIT 5-UDRE: USART Data Register Empty.

Esta bandera monitorea si el registro UDR esta vacio, cuando esta vacio y listo para ser escrito este bit se pone a 1.

BIT 4-FE: Frame Error.

Bandera de Error de Trama.

BIT 3-DOR: Data OverRun.

Bandera de monitoreo de desbordamiento.

BIT 2-PE: Parity Error.

Bandera de Error de paridad.

BIT 1-U2X: Double the USART transmission speed.

Este registro es para seleccionar la velocidad de transmisión (Baude Rate).

BIT 0-MPCM: Multi-processor Communication Mode.

Este registro es para activar la opción de Multi-processor Communication

Mode que tiene nuestro microcontrolador.

USART Control and Status Register B – UCSRB

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Registro 2-4 UCSRB

BIT 7-RXCIE: RX Complete Interrupt Enable.

Habilita o deshabilita interrupciones cuando la recepción de datos este completa; 1 habilita 0 deshabilita.

BIT 6-TXCIE: TX Complete Interrupt Enable.

Habilita o deshabilita interrupciones cuando la transmisión de datos este completa; 1 habilita 0 deshabilita.

BIT 5-UDRIE: USART Data Register Empty Interrupt Enable.

Este registro es para habilitar la interrupción cuando el registro UDRE esta vacio.

BIT 4-RXEN: Receiver Enable.

Habilita (poniendo un 1) o deshabilita (poniendo un 0) a RX

BIT 3-TXEN: Transmitter Enable.

Habilita (poniendo un 1) o deshabilita (poniendo un 0) a TX

BIT 2-UCSZ2: Character Size.

Selecciona el tamaño de la trama que se desea usar (este bit está relacionado con el registro UCSRC).

BIT 1-RXB8: Receive Data Bit 8.

Es el noveno bit cuando se recibe la trama de datos manejan serial frames y de ve ser leído antes de bajar los datos de UDR.

BIT 0-TXB8: Transmit Data Bit 8.

Es el noveno bit cuando se transmite la trama de datos manejan serial frames y de ve ser leído antes de bajar los datos de UDR.

USART Control and Status Register C – UCSRC

Bit	7	6	5	4	3	2	1	0	
	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	0	0	0	0	1	1	0	

Registro 2-5 UCSRC

BIT 7-URSEL: Register Select.

This bit selects between accessing the UCSRC or the UBRRH Register. It is read as one when reading UCSRC. The URSEL must be one when writing the UCSRC

BIT 6-UMSEL: USART Mode Select.

Selecciona el modo de transmisión síncrona o asíncrona.

Table 55. UMSEL Bit Settings

UMSEL	Mode
0	Asynchronous Operation
1	Synchronous Operation

Tabla 2-8 Selección de operación asíncrono-síncrono

BIT 5:4-UPM 1:0: Parity Mode

Selecciónanos el modo de paridad con la siguiente tabla 2-9

Table 56. UPM Bits Settings

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

Tabla 2-9 Selección de modo de paridad.

BIT 3-USBS: Stop Bit Select.

Selecciona el número de bits de paro de acuerdo a la siguiente tabla 2-10

Table 57. USBS Bit Settings

USBS	Stop Bit(s)
0	1-bit
1	2-bit

Tabla 2-10 Selección de modo de bits de paro.

BIT 2:1-UCSZ1:0: Character Size.

Seleccionamos el tamaño de la trama que deseamos manejar según la siguiente tabla

- Recordemos que en el registro UCSRB en su bit 2 se encuentra UCSZ2.

Table 58. UCSZ Bits Settings

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

Tabla 2-11 Selección del tamaño de trama.

BIT 0-UCPOL: Clock Polarity

Este bit solo se utiliza para el modo síncrono. Se escribe un 0 cuando el modo seleccionado es asíncrono.

USART I/O Data Register – UDR

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDR (Read)
	TXB[7:0]								UDR (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Registro 2-6 UDR

En este registro es donde se almacena los datos transmitidos y los recibidos.

USART Baud Rate Registers – UBRRL and UBRRHs

Bit	15	14	13	12	11	10	9	8	
	URSEL	–	–	–	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Registro 2-7 UBRRL-UBRRH

Bit 11-0-UBRR11:0: USART Baud Rate Register.

En estos bits (12) escribiremos la tasa de transmisión que usaremos para enviar la información, para poder saber que dato escribiremos tenemos que utilizar la tabla 1-12 donde calcularemos el dato a escribir en este registro.

Table 52. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps).

BAUD Baud rate (in bits per second, bps)

f_{OSC} System Oscillator clock frequency

UBRR Contents of the UBRRH and UBRRL Registers, (0 - 4095)

Tabla 2-11 Ecuaciones para el cálculo de la tasa de transmisión.

2.14. INTERRUPCIONES

Una de las características más importantes de los microcontroladores es que tienen la posibilidad de manejar interrupciones; las interrupciones se tratan de acontecimientos que hacen que el microcontrolador deje de lado lo que se encontraba realizando y atienda ese suceso, luego de terminar, regrese y continúe con la tarea que se estaba haciendo antes de este suceso.
Pero también se puede encontrar con que existen dos tipos de interrupciones posibles, en donde una es mediante una acción externa, es decir, por la acción de uno de sus pines, y la otra es interna, por ejemplo, cuando ocurre el desbordamiento de alguno de sus registros.

2.15. INTERRUPCIONES EN EL AVR ATmega8L

En el AVR tenemos diferentes fuentes de interrupciones, cada una teniendo su espacio de memoria de programa para el vector de interrupción, en esta dirección se indicará cual es la dirección de salto donde se encuentra el código que se debe ejecutar para esa interrupción. El contador de programa se carga con la dirección del vector de interrupción una vez que ésta sucede y ahí tendremos una instrucción de salto a la dirección de memoria donde está el código que atiende a la interrupción.

El microcontrolador tiene varias fuentes de interrupción, como son, por conversión A/D terminada, por escritura en la eeprom completada, por desborde de temporizadores, por recepción de datos en la USART, por transmisión completada de datos en la USART, interrupciones externas por pines del micro al detectar un flanco ya sea de bajada o subida, entre otras.

Podemos resaltar que en los micros AVR cada interrupción posee su propio vector de interrupción lo que hace más cómoda su ejecución y programación de la rutina correspondiente. Tabla 2-12

Interrupt Vectors in ATmega8

Table 18. Reset and Interrupt Vectors

Vector No.	Program Address(2)	Source	Interrupt Definition
1	0x000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, and Watchdog Reset
2	0x001	INT0	External Interrupt Request 0
3	0x002	INT1	External Interrupt Request 1
4	0x003	TIMER2 COMP	Timer/Counter2 Compare Match
5	0x004	TIMER2 OVF	Timer/Counter2 Overflow
6	0x005	TIMER1 CAPT	Timer/Counter1 Capture Event
7	0x006	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	0x007	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	0x008	TIMER1 OVF	Timer/Counter1 Overflow
10	0x009	TIMER0 OVF	Timer/Counter0 Overflow
11	0x00A	SPI, STC	Serial Transfer Complete
12	0x00B	USART, RXC	USART, Rx Complete
13	0x00C	USART, UDRE	USART Data Register Empty
14	0x00D	USART, TXC	USART, Tx Complete
15	0x00E	ADC	ADC Conversion Complete
16	0x00F	EE_RDY	EEPROM Ready
17	0x010	ANA_COMP	Analog Comparator
18	0x011	TWI	Two-wire Serial Interface
19	0x012	SPM_RDY	Store Program Memory Ready

Notes: 1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see "Boot Loader Support – Read-While-Write Self-Programming" on page 206.
2. When the IVSEL bit in GICR is set, Interrupt Vectors will be moved to the start of the boot Flash section. The address of each Interrupt Vector will then be the address in this table added to the start address of the boot Flash section.

Tabla 2-12 Vectores de Interrupciones de ATmega8

2.16. CONTROL DE UN LCD CON UN MICROCONTROLADOR

Antes de aparecer los módulos LCD (*Liquid Cristal Display*), los diseños electrónicos utilizaban los displays de siete segmentos para poder mostrar la información. Además de su gran limitación de poder mostrar los caracteres alfa numéricos y símbolos especiales, también consumían demasiada corriente y ocupaban demasiado espacio físico. Posteriormente aparecieron otros tipos de displays más complejos, que podían mostrar algunos caracteres y símbolos; pero tenían de igual manera mucho consumo de corriente y espacio físico desperdiciado. Finalmente aparecieron los módulos LCD, o pantallas de cristal líquido, el cual tiene la capacidad de mostrar cualquier carácter alfa numérico. Estos dispositivos ya vienen con su pantalla y toda la lógica de control pre-programada en la fábrica, y lo mejor de todo es que el consumo de corriente es mínimo y no se tendrán que organizar tablas especiales, como se hacía anteriormente con los displays de siete segmentos.

El LCD es actualmente el circuito más barato y confiable para mostrar datos en un proceso de monitoreo y control. Su interfaz con los controladores se realiza a través de un conector de 14 pines, cuya configuración es respetada por la mayoría de los fabricantes. A diferencia de los teclados, los fabricantes del LCD han estandarizado sus señales en un conector de 14 pines, así como sus comandos de control para el manejo del mismo.

En el LCD se pueden mostrar datos como la hora y la fecha, así como valores de variables tales como nivel, presión, gasto, temperatura, etc. El LCD puede también emplearse para programar parámetros internos del sistema, de acuerdo a su aplicación, ó para mostrar al usuario las opciones del sistema mientras lo opera. Las aplicaciones de los módulos LCD son infinitas, ya que podrán ser aplicados en la informática, comunicaciones, telefonía, instrumentación, robótica, automóviles, equipos industriales, etc.

El módulo LCD lleva integrado a sus circuitos una memoria ROM, conocida como
"generador de caracteres", que habrá de generar los patrones de la matriz de puntos (5 x 7 ó 7 x 9) que forman los caracteres en la pantalla. También tiene una RAM interna que almacena los caracteres y los exhibe en el módulo LCD.

2.16.1. DIVERSIDAD DE ALGUNOS MÓDULOS LCD

En la actualidad los módulos LCD tienen una gran variedad de versiones, que se clasifican en dos grupos. El primer grupo está referido a los módulos LCD de caracteres (solamente se podrán presentar caracteres y símbolos especiales en las líneas predefinidas en el módulo LCD) y el segundo grupo está referido a los módulos LCD matriciales (se podrán presentar caracteres, símbolos especiales y gráficos).Los módulos LCD varían su tamaño físico

dependiendo de la marca, por lo tanto en la actualidad no existe un tamaño estándar para los módulos LCD.
Los primeros módulos LCD tenían los caracteres de color negro y el fondo de la pantalla era de color verdoso claro. Posteriormente se crearon otros colores en donde los caracteres eran de color plata y así sucesivamente fueron variando los colores en el fondo y en los caracteres, incluyendo una luz posterior para los módulos LCD denominada Back Light, diseñada especialmente para mejorar la visualización de la pantalla, sobre todo en lugares muy oscuros.

2.16.2. ASPECTO FÍSICO

El LCD tiene un aspecto físico como el de la figura 2-24, en donde se puede observar que está constituido por un circuito impreso en el que están integrados los controladores del display (figura 2-25) y los pines para la conexión del display. Sobre el circuito impreso se encuentra el LCD en sí, rodeado por una estructura metálica que lo protege. En este LCD se pueden visualizar 2 líneas de 16 caracteres cada una, es decir, 32 caracteres.

Figura 2-24 LCD asignación de pines.

A pesar de que el display sólo puede visualizar 16 caracteres por línea, puede almacenar en total 40 por línea.

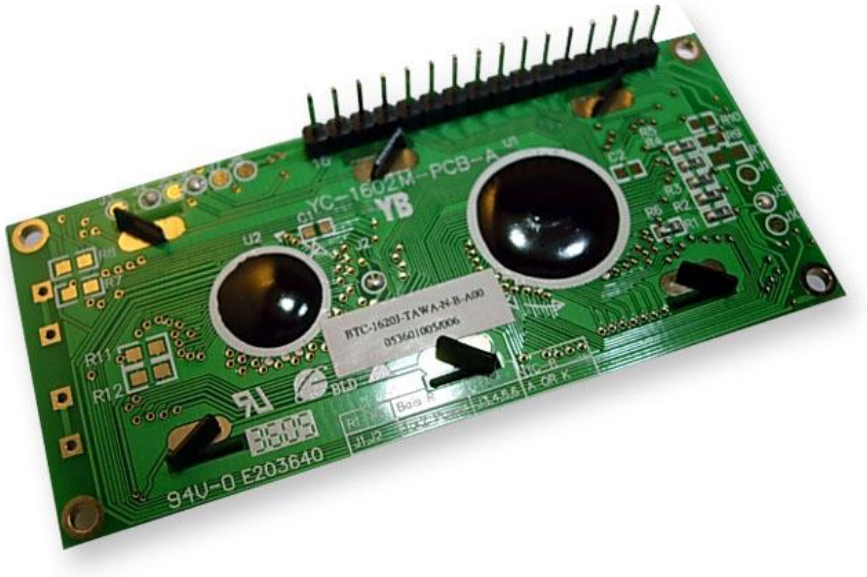

Figura 2-25 Controladores del LCD

2.16.3. LOS CARACTERES DEL LCD

El LCD dispone de una matriz de 5x8 puntos para representar cada carácter. En total se pueden representar 256 caracteres diferentes. 240 caracteres están grabados en LCD y representan las letras mayúsculas, minúsculas, signos de puntuación, etc. Existen 8 caracteres que pueden ser definidos por el usuario. En la figura 2-26 se observa gráficamente cómo es la matriz de representación de los caracteres, en donde se muestra dibujado el carácter A y un carácter definido por el usuario.

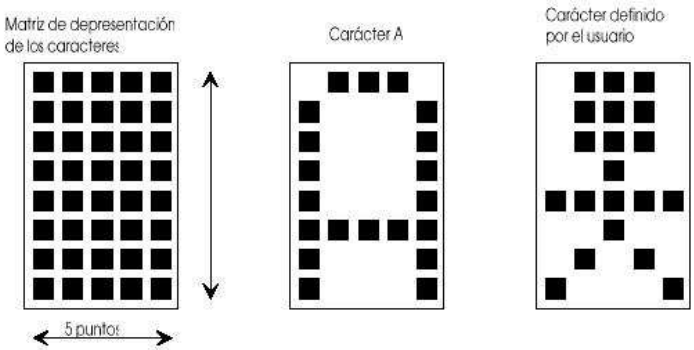

Figura 2-26 Matriz de representación de caracteres, representación del caracter A y de un caracter definido por el usuario.

En la tabla 2-13, se pueden apreciar los caracteres más importantes que es capaz de imprimir el display. Todos los códigos están en hexadecimal. No se han representado los caracteres correspondientes a los códigos desde el $80

hasta el $FF, que corresponden a símbolos extraños. Los códigos comprendidos entre el 0 y el 7 están reservados para que el usuario los defina.

Tabla 2-13 Código asociado a cada carácter imprimible por el display

Código	Car.	Código	Car.	Código	Car.	Código	Car.	Código	Car.	Código	Car.
$20	espacio	$30	0	$40		$50	P	$60	`	$70	p
$21	!	$31	1	$41	A	$51	Q	$61	a	$71	q
$22	"	$32	2	$42	B	$52	R	$62	b	$72	r
$23	#	$33	3	$43	C	$53	S	$63	c	$73	s
$24	$	$34	4	$44	D	$54	T	$64	d	$74	t
$25	%	$35	5	$45	E	$55	U	$65	e	$75	u
$26	&	$36	6	$46	F	$56	V	$66	f	$76	v
$27	'	$37	7	$47	G	$57	W	$67	g	$77	w
$28	(	$38	8	$48	H	$58	X	$68	h	$78	x
$29	)	$39	9	$49	I	$59	Y	$69	i	$79	y
$2A	*	$3A	:	$4A	J	$5A	Z	$6A	j	$7A	z
$2B	+	$3B	;	$4B	K	$5B	[	$6B	k	$7B	{
$2C	,	$3C	<	$4C	L	$5C	\	$6C	l	$7C	|
$2D	-	$3D	=	$4D	M	$5D	]	$6D	m	$7D	}
$2E	.	$3E	>	$4E	N	$5E	^	$6E	n	$7E	→
$2F	/	$3F	?	$4F	O	$5F	_	$6F	o	$7F	←

1.16.4. ASIGNACIÓN DE PINES

Los pines de conexión de un módulo LCD han sido estandarizados, por lo cual en la mayoría de ellos son exactamente iguales. Por otro lado, es de suma importancia localizar exactamente cuál es el pin número 1, ya que en algunos módulos se encuentra hacia la izquierda, y en otros módulos se encuentra a la derecha. En este caso se utilizó el modelo de display JHD162A, con la identificación de pines como lo muestra la figura 2-24, la cual nos indica que el pin número 1 está ubicado en la izquierda.
Y como todo pin cumple una función en específico; en la tabla 1-14 se puede ver la asignación de los pines.

Tabla 2-14 Asignación de pines del LCD

Pin	Simbología	Nivel	I/O	Función
1	VSS	-	-	0 V Tierra (GND).
2	VCC	-	-	+ 5 V DC.
3	Vee = Vcc	-	-	Ajuste del Contraste.
4	RS	0/1	I	0= Escribir en el módulo LCD. 1= Leer del módulo LCD
5	R/W	0/1	I	0= Entrada de una Instrucción. 1= Entrada de un dato.
6	E	1	I	Habilitación del módulo LCD
7	DB0	0/1	I/O	BUS DE DATO LINEA 1 (LSB).
8	DB1	0/1	I/O	BUS DE DATO LINEA 2
9	DB2	0/1	I/O	BUS DE DATO LINEA 3
10	DB3	0/1	I/O	BUS DE DATO LINEA 4
11	DB4	0/1	I/O	BUS DE DATO LINEA 5
12	DB5	0/1	I/O	BUS DE DATO LINEA 6
13	DB6	0/1	I/O	BUS DE DATO LINEA 7
14	DB7	0/1	I/O	BUS DE DATO LINEA 8 (MSB).
15	A	-	-	LED (+) Back Light
16	K	-	-	LED (-) Back Light.

2.16.5. INTERPRETACIÓN DEL SIGNIFICADO DE LOS PINES DEL LCD

El pin número 1 y 2: Están destinados para alimentar con los 5 volts que requiere el módulo para su funcionamiento.

El pin número 3: Es utilizado para ajustar el contraste de la pantalla, es decir, colocar los caracteres más oscuros o más claros para poderse observar mejor. La resistencia representada como “R” en la figura 2-27 es un potenciómetro variable que puede oscilar entre 10 kΩ y 20 kΩ indiferentemente.

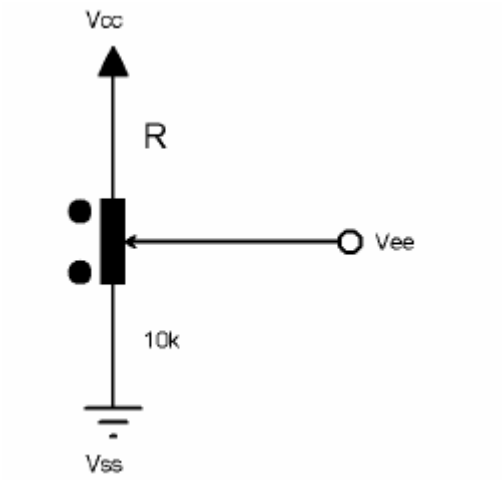

Figura 2-27 Control de contraste.

El pin número 4: Es denominado **"RS"** y trabaja paralelamente al Bus de datos del módulo LCD (Bus de datos son los Pines del 7 al 14). Este bus es utilizado de dos maneras, ya que se puede colocar un dato que representa una instrucción, o colocar un dato que tan sólo representa un símbolo o un carácter alfa numérico; pero para que el módulo LCD pueda entender la diferencia entre un dato o una instrucción, se utiliza el pin número 4 para tal fin. Si el pin **RS=0**, le dirá al módulo LCD que está presente en el bus de datos una instrucción, por el contrario, si el pin **RS=1**, le dirá al módulo LCD que está presente un símbolo o un carácter alfa numérico.

El pin número 5: Es denominado **"R/W"** y trabaja paralelamente al bus de datos del módulo LCD (el bus de datos son los pines del 7 al 14). También es utilizado de dos maneras, ya que se le puede decir al módulo LCD que escriba en pantalla el dato que está presente en el bus; por otro lado también se puede leer que dato está presente en el bus. Si el pin **R/W=0**, el módulo LCD escribe en pantalla el dato que está presente el bus. Pero si el pin **R/W=1**, significa que necesitamos leer el dato que está presente en el bus del módulo LCD.

El pin número 6: Es denominado **"E"**, que significa habilitación del módulo LCD, tiene una finalidad básica: conectar y desconectar el módulo. Esta desconexión no estará referida al voltaje que le suministra la corriente al módulo; la desconexión significa tan solo que se hará caso omiso a todo lo que esté presente en el bus de datos de dicho módulo LCD. En la mayoría de los circuitos electrónicos modernos que incluyan elementos electrónicos, como microcontroladores, memorias y módulos LCD, utilizan el mismo bus de datos, esto es para no tenerlo independientemente por cada elemento electrónico, esto implicaría que los circuitos electrónicos sean mucho más grandes por la cantidad de conexiones necesaria a cada uno de los elementos. Para los microcontroladores, memorias y módulos LCD, deberá existir en cada uno de ellos un pin de habilitación **"E"** que permita desconectar y conectar cuando sea necesario. Por ejemplo, si necesitamos

trabajar con la memoria RAM para obtener o escribir cierta información, será necesario que deshabilitemos el módulo LCD para que no presente basura en la pantalla, o se ejecuten instrucciones no deseadas.

Los pines desde el número 7 hasta el número 14: Son 8 líneas que se utilizan para colocar el dato que representa una instrucción para el módulo LCD, o un carácter alfa numérico. El bus de datos es de 8 bits de longitud, y el bit menos significativo está representado en el pin número 7 y el pin más significativo está representado en el pin número 14.

Los pines 15 y 16: Están destinados para suministrar la corriente al Back Light. Es importante conocer que no todos los módulos LCD disponen del Back Light, aunque tenga los pines de conexión en el circuito impreso.

2.16.6. INTERFAZ DEL DISPLAY CON EL MUNDO EXTERIOR

En la figura 2-28 aparecen las señales necesarias para el funcionamiento y control del display. Los datos se transmiten por un bus de datos de 8 bits de anchura; también el display ofrece la posibilidad de trabajar con este bus multiplexado en dos grupos de 4 bits.

Para el control del display son necesarios 3 bits: una señal de habilitación (E), una para indicar lectura/escritura (R/W) y otra para seleccionar uno de los dos registros internos (RS). Por ello el sistema de control del display necesitará utilizar 11 bits, pero en este caso se va a utilizar el bus multiplexado, esto quiere decir que, nada más se utilizaran 4 bits para el bus de datos y los 3 bits para el control, de modo que solamente habrá 7 bits para el control del display con el microcontrolador.

Utilizar el bus multiplexado de 4 bits es una opción muy útil para ahorrar bits en el sistema de control. De esta forma se ahorran bits en el controlador, pero se gana en complejidad a la hora de hacer el programa, al tener que hacer la multiplexación. Al utilizar un bus de 8 bits hacemos que el controlador sea más sencillo pero se necesitan muchos más bits.

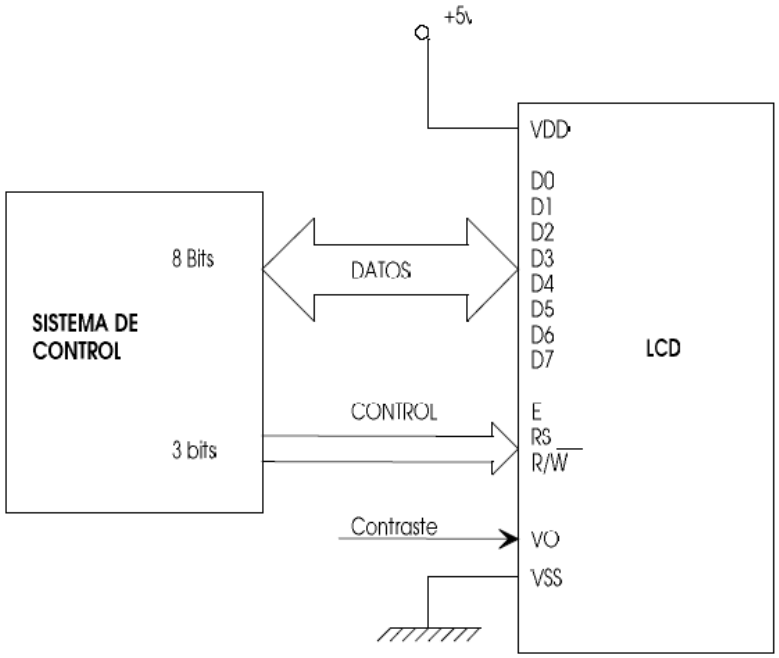

Figura 2-28 Interfaz del LCD con un sistema de control

En la figura 2-29 se observa cuando solamente se utiliza un bus de 4 bits, en donde solamente se utilizan los pines D4-D7 del display. En este tipo de conexión la transferencia de la información se realiza de la siguiente manera: primero los 4 bits más significativos y luego los 4 menos significativos.

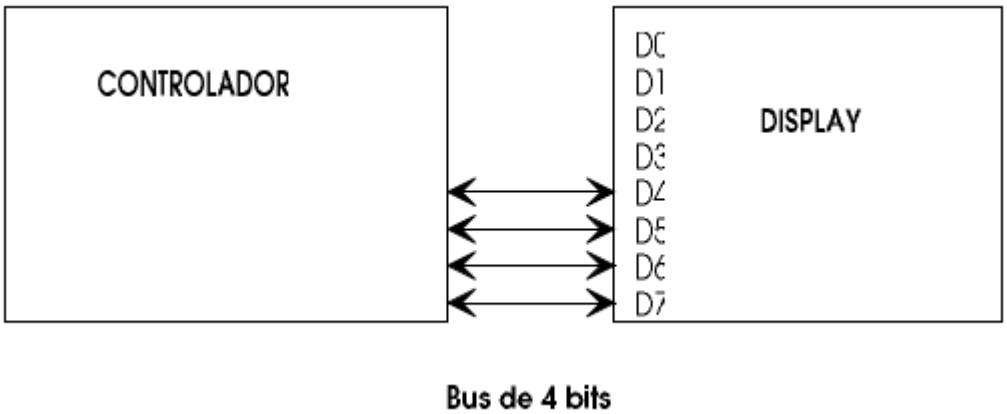

Figura 2-29 Conexión del LCD con un bus de 4 bits

2.16.7. CONEXIÓN DEL LCD AL MICROCONTROLADOR AVR ATmega8L

En la actualidad los microcontroladores son los elementos electrónicos de mayor utilidad, y en la figura 2-30 se describe de qué manera fue conectado el módulo LCD con el microcontrolador de la empresa Atmel modelo AVR ATmega8L.

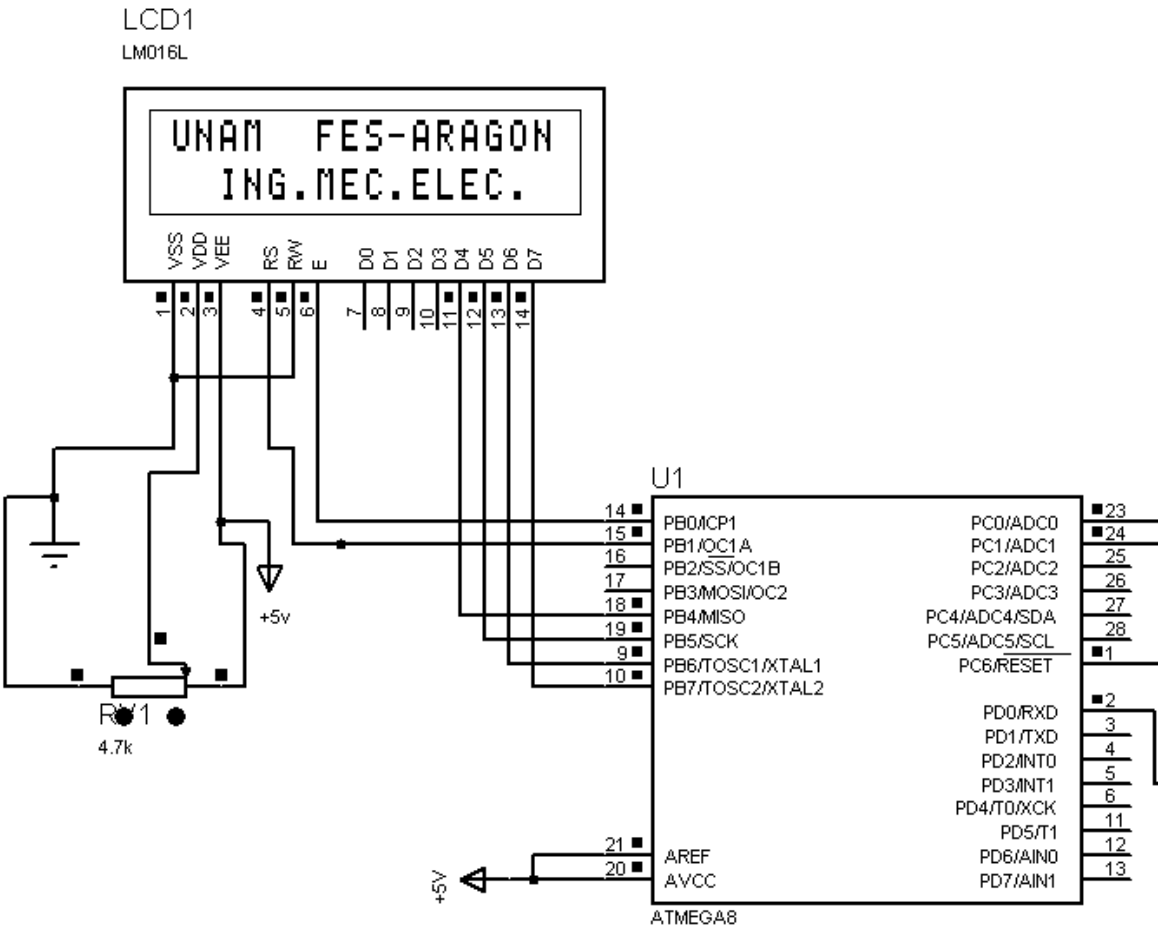

Figura 2-30 Conexión del LCD con el microcontrolador AVR ATmega8

CAPÍTULO 3: DISEÑO DE INTERFAZ

3.1. ¿QUÉ ES USB?

En un principio se tenía la interfaz serie y paralelo, pero era necesario unificar todos los conectores creando uno más sencillo y de mayores prestaciones. Así nació el USB (Universal Serial Bus) figura 3-1 con una velocidad de 12Mb/seg. y como su evolución, USB 2.0, apodado USB de alta velocidad, con velocidades en este momento de hasta 480 Mb/seg, es decir, 40 veces más rápido que las conexiones mediante cables USB 1.1. USB es una nueva arquitectura de bus o un nuevo tipo de bus desarrollado por un grupo de siete empresas (Compaq, Digital Equipment Corp, IBM PC Co., Intel, Microsoft, NEC y Northern Telecom)

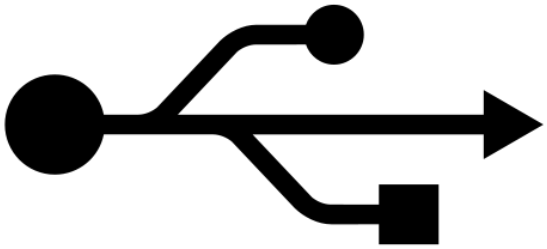

Figura 3-1 Símbolo USB

USB Universal Serial Bus es una interface plug&play entre la PC y ciertos dispositivos tales como teclados, mouse, scanner, impresoras, módems, placas de sonido, cámaras, sistemas de adquisición de datos y componentes de red, etc.

Este concepto se refiere a la cualidad de que con sólo conectar el dispositivo al servidor central, éste sea capaz de interpretar la información almacenada y reproducirla inmediatamente. Es decir, que el computador y el aparato hablen el mismo idioma y se entiendan entre sí. Además, este sistema permite conectar y desconectar los diferentes dispositivos sin necesidad de reiniciar el equipo.

Una característica importante es que permite a los dispositivos trabajar a velocidades mayores, en promedio a unos 12 Mbps, esto es más o menos de 3 a 5 veces más rápido que un dispositivo de puerto paralelo y de 20 a 40 veces más rápido que un dispositivo de puerto serial.

3.2. VELOCIDADES DE TRANSMICION

Los dispositivos USB se clasifican en cuatro tipos según su velocidad de transferencia de datos:

- **Baja velocidad (1.0):** Tasa de transferencia de hasta 1,5 Mbps (192 KB/s). Utilizado en su mayor parte por dispositivos de interfaz humana (*Human interface device*, en inglés) como los teclados, los ratones (mouse), etc.

- **Velocidad completa (1.1):** Tasa de transferencia de hasta 12 Mbps (1,5 MB/s) según este estándar, pero se dice en fuentes independientes que habría que realizar nuevamente las mediciones. Ésta fue la más rápida antes de la especificación USB 2.0, y muchos dispositivos fabricados en la actualidad trabajan a esta velocidad. Estos dispositivos dividen el ancho de banda de la conexión USB entre ellos, basados en un algoritmo de impedancias LIFO.

- **Alta velocidad (2.0):** Tasa de transferencia de hasta 480 Mbps (60 MB/s) pero por lo general de hasta 125Mbps (16MB/s). Está presente casi en el 99% de los PC actuales. El cable USB 2.0 dispone de cuatro líneas, un par para datos, una de corriente y una de toma de tierra.

- **Súper alta velocidad (3.0):** Tiene una tasa de transferencia de hasta 4.8 Gbps (600 MB/s). La velocidad del bus es diez veces más rápida que la del USB 2.0, debido a que han incluido 5 conectores extra, desechando el conector de fibra óptica propuesto inicialmente, y será compatible con los estándares anteriores. usa un cable de 9 hilos. En Octubre de 2009 la compañía taiwanesa ASUS lanzó la primera placa base que incluía puertos USB3, tras ella muchas otras le han seguido y se espera que en 2012 ya sea el estándar de facto.

3.3. CABLES Y CONECTORES

El cable de bus USB es de 4 hilos, y comprende líneas de señal (datos) y alimentación, con lo que las funciones pueden utilizar un único cable. Existen dos tipos de cable: apantallado y sin apantallar. En el primer caso el par de hilos de señal es trenzado; los de tierra y alimentación son rectos, y la cubierta de protección (pantalla) solo puede conectarse a tierra en el anfitrión. En el cable sin apantallar todos los hilos son rectos. Las conexiones a 15 Mbps y superiores exigen cable apantallado. figura 3-2

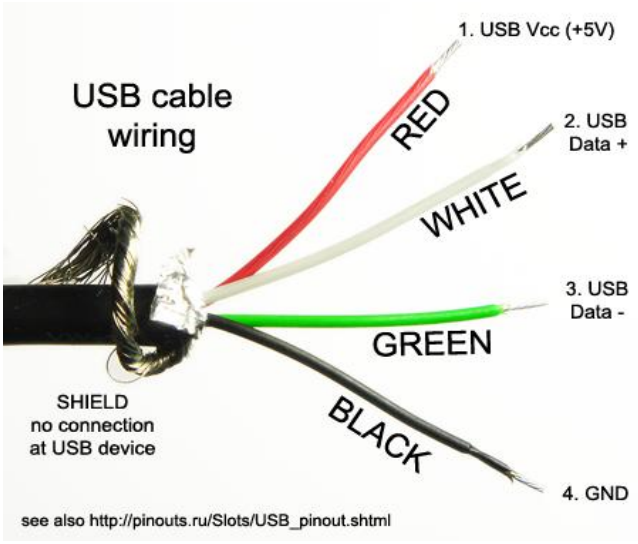

Figura 3-2 Cable de bus USB

Pin	Nombre	Color del cable	Descripción
1	VCC	Rojo	+5v
2	D−	Blanco	Data −
3	D+	Verde	Data +
4	GND	Negro	Tierra

Tabla 3-1 Asignación de pines del USB

Se utilizan diámetros estándar para los hilos de alimentación del bus. Para cada sección se autoriza una longitud máxima del segmento. En la tabla 3-2 se muestran estas distancias.

AWG	mm Ø	long. máx.
28	0.321	0.81 m
26	0.405	1.31 m
24	0.511	2.08 m
22	0.644	3.33 m
20	0.812	5.00 m

Tabla 3-2 Diámetro y longitud estándar de hilos USB

Se usan dos tipos de conectores, A y B. Ambos son polarizados (solo pueden insertarse en una posición) y utilizan sistemas de presión para sujetarse. Los de tipo A utilizan la hembra en el sistema anfitrión, y suelen usarse en dispositivos en los que la conexión es permanente (por ejemplo,

ratones y teclados). Los de tipo B utilizan la hembra en el dispositivo USB (función), y se utilizan en sistemas móviles (por ejemplo, cámaras fotográficas o altavoces). En general podemos afirmar que la hembra de los conectores A están en el lado del host (PC) o de los concentradores (hubs), mientras las de tipo B están del lado de los periféricos. Figura 3-3.

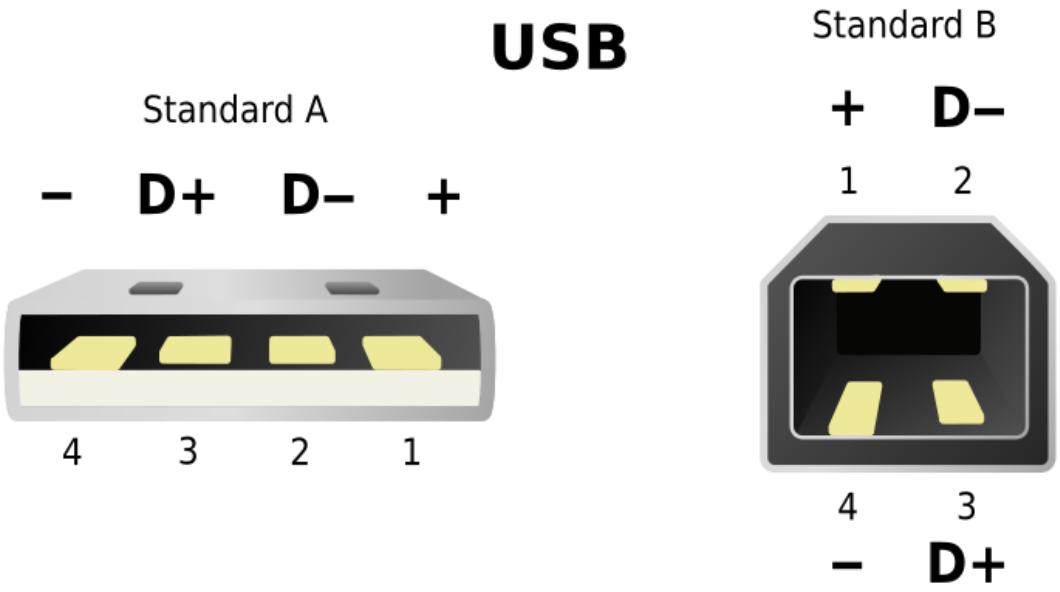

Figura 3-3 Tipos de conectores USB

3.4. TOPOLOGIA EN DISPOCITIBOS USB

Los dispositivos USB adoptan una topología de estrella y se organiza por niveles a partir de un controlador host instalado en la placa base, que actúa de interfaz entre el bus de ésta (generalmente a la interfaz PCI) y el primer dispositivo USB, el denominado concentrador raíz ("Root hub"), instalado también en la placa. El controlador de host es único; suele ser un chip Intel con una denominación como 82371AB/EB; 82801DB, etc. Dada la proliferación de este tipo de dispositivos, las placas modernas pueden disponer de varios concentradores raíz, cada uno con su propia salida (generalmente 2 conectores del tipo "A" por cada uno de ellos). Cada uno de estos concentradores se considera el origen de un bus (numerados sucesivamente a partir del 0), del que cuelgan los dispositivos en el orden en que son detectados por el Sistema.

El bus USB soporta intercambio simultáneo de datos entre un ordenador anfitrión y un amplio conjunto de periféricos. Todos los periféricos conectados comparten el ancho de banda del bus por medio de un protocolo de arbitraje basado en testigos ("Tokens"). El bus permite conexión y desconexión dinámica, es decir, que los periféricos se conecten, configuren, manipulen y desconecten mientras el sistema anfitrión y otros periféricos permanecen en funcionamiento.

En un bus USB existen dos tipos de elementos: Anfitrión ("host") y dispositivos; a su vez, los dispositivos pueden ser de dos tipos: concentradores y funciones.

- Los concentradores ("Hubs") son el centro de una estrella, y sirven para conectar con el sistema anfitrión, con otro hub o con una función. Cada hub puede conectar hasta 7 dispositivos, aunque lo normal es que sean de 4 salidas, y proporcionar 500 mA de energía de alimentación (hasta 2.5 W) a cada uno de ellos, ya que el cable de conexión tiene hilos de señal (datos) y de alimentación (5 V. CC ± 0.25 V).

- Una función es un dispositivo capaz de transmitir o recibir datos o información de control en un bus USB, suele conectarse como un dispositivo independiente enlazado por un cable de menos de 5 metros, a un puerto del hub o directamente al sistema anfitrión.

En el diagrama de capas de la figura 3-4 podemos ver cómo fluye la información entre las diferentes capas a nivel real y a nivel lógico.

En dicha figura está materializada la conexión entre el controlador anfitrión o host y un dispositivo o periférico. Este está constituido por hardware al final de un cable USB y realiza alguna función útil para el usuario.

El software cliente se ejecuta en el host y corresponde a un dispositivo USB; se suministra con el sistema operativo o con el dispositivo USB . El software del sistema USB, es el que soporta USB en un determinado sistema operativo y se suministra con el sistema operativo independientemente de los dispositivos USB o del software cliente.

El controlador host USB está constituido por el hardware y el software que permite a los dispositivos USB ser conectados al anfitrión. La conexión entre un host y un dispositivo requiere la interacción entre las capas.

- La capa de interfaz de bus USB proporciona la conexión física entre el host y el dispositivo.

- La capa de dispositivo USB es la que permite que el software del sistema USB realice operaciones genéricas USB con el dispositivo.

- La capa de función proporciona capacidades adicionales al host vía una adecuada capa de software cliente.

Las capas de función y dispositivos USB tienen cada una de ellas una visión de la comunicación lógica dentro de su nivel, aunque la comunicación entre

ellas se hace realmente por la capa de interfaz de bus USB.

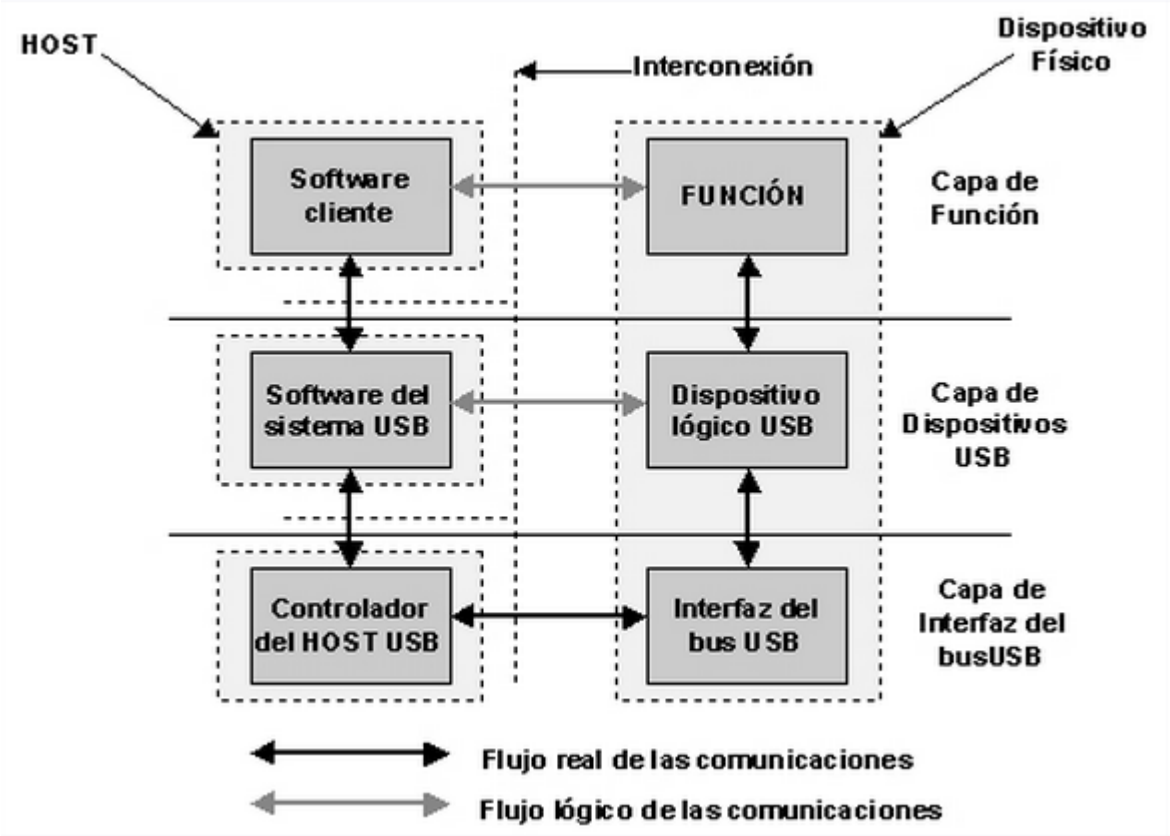

Figura 3-4 Capas del sistema de comunicaciones USB

3.5. FUNCIONAMIENTO

El bus serie USB es síncrono, y utiliza el algoritmo de codificación NRZI ("Non Return to Zero Inverted"). En este sistema existen dos voltajes opuestos; una tensión de referencia corresponde a un "1", pero no hay retorno a cero entre bits, de forma que una serie de unos corresponde a un voltaje uniforme; en cambio los ceros se marcan como cambios del nivel de tensión, de modo que una sucesión de ceros produce sucesivos cambios de tensión entre los conductores de señal.

A partir de las salidas proporcionadas por los concentradores raíz (generalmente conectores del tipo "A") y utilizando concentradores adicionales, pueden conectarse más dispositivos.

El protocolo de comunicación utilizado es de testigo, que guarda cierta similitud con el sistema Token-Ring de IBM. Puesto que todos los periféricos comparten el bus y pueden funcionar de forma simultánea, la información es enviada en paquetes; cada paquete contiene una cabecera que indica el periférico a que va dirigido. Existen cuatro tipos de paquetes distintos: Token; Datos; Handshake, y Especial; el máximo de datos por paquete es de 8; 16;

32 y 64 Bytes. Se utiliza un sistema de detección y corrección de errores bastante robusto tipo CRC ("Cyclical Redundancy Check").

El funcionamiento está centrado en el host, todas las transacciones se originan en él. Es el controlador host el que decide todas las acciones, incluyendo el número asignado a cada dispositivo (esta asignación es realizada automáticamente por el controlador "host" cada vez que se inicia el sistema o se añade, o elimina, un nuevo dispositivo en el bus), su ancho de banda, etc. Cuando se detecta un nuevo dispositivo es el host el encargado de cargar los drivers oportunos sin necesidad de intervención por el usuario. El sistema utiliza cuatro tipo de transacciones que resuelven todas las posibles situaciones de comunicación. Cada transacción utiliza un mínimo de tres paquetes, el primero es siempre un Token que avisa al dispositivo que puede iniciar la transmisión

- **Transferencia de control** ("Control transfer"): Ocurre cuando un dispositivo se conecta por primera vez. En este momento el controlador de host envía un paquete "Token" al periférico notificándole el número que le ha asignado.

- **Transferencia de pila de datos** ("Bulk data transfer"): Este proceso se utiliza para enviar gran cantidad de datos de una sola vez. Es útil para dispositivos que tienen que enviar gran cantidad de datos cada vez, como escáneres o máquinas de fotografía digital.

- **Transferencia por interrupción** ("Interrupt data transfer"): Este proceso se utiliza cuando se solicita enviar información por el bus en una sola dirección (de la función al host).

- **Transferencia de datos síncrona** ("Isochronous data transfer"): Este proceso se utiliza cuando es necesario enviar datos en tiempo real. Los datos son enviados con una cadencia precisa ajustada a un reloj, de modo que la transmisión es a velocidad constante

3.6. MODULO DE INTERFAZ USB

Para este trabajo se ha utilizado como interfaz de comunicación el dispositivo FT232BL de la empresa FTDI Chip que son especialistas en la conversión de periféricos a USB mediante la combinación USB-Serial (USB-RS232) y algunas de sus características son:

- **Transmisión Asíncrona Serial <=> USB**
- **Interfaz USART compatible 7/8 bits de datos, 1/2 bits de stop, inicio, paridad.**
- **Velocidad de transferencia de 300=>1M Baud (RS232)**
- **Convertidor integrado en USART para señales de 5V. y 3.3V.**

lógicas

- **Regulador integrado de 3.3V para USB I/O**
- **Voltaje de operación 4.35V-5.25V**
- **Compatible con USB 1.1 y USB 2.0**

Se pueden mencionar algunas áreas de aplicación que tiene el FT232BL

- **USB <=> RS232 convertidores**
- **USB <=> RS422 / RS485 convertidores**
- **Actualizar periféricos RS232 legado para USB**
- **Celulares y teléfonos inalámbricos cables USB de transferencia de datos e interfaces**
- **Interfaz de MCU basado en diseños de USB**
- **USB de audio y vídeo de bajo ancho de banda de transferencia de datos**
- **PDA <=> USB de transferencia de datos**
- **USB Lectores de tarjetas inteligentes**
- **Set Top Box (STB) PC - Interfaz USB**
- **Los módems USB de hardware**
- **Módems USB inalámbrico**
- **USB de instrumentación**
- **USB lectores de códigos de barras**

También se tienen los rangos máximos absolutos que se observan en la tabla 3-3

Tabla 3-3 Rangos Máximos

Parameter	Value	Units
Storage Temperature	–65°C to + 150°C	Degrees C
Floor Life (Out of Bag) at Factory Ambient (30°C/60% Relative Humidity)	192 Hours (Level 3 Compliant) **Note 2	
Ambient Temperature (Power Applied)	0°C to + 70°C	Degrees C
M.T.B.F. (at 35°C)	247484 Hours ≈ 28 Years	
VCC Supply Voltage	-0.5 to +6.00	V
D.C. Input Voltage - USBDP and USBDM	-0.5 to +3.8	V
D.C. Input Voltage - High Impedance Bidirectionals	-0.5 to +(Vcc +0.5)	V
D.C. Input Voltage - All other Inputs	-0.5 to +(Vcc +0.5)	V
DC Output Current – Outputs	24	mA
DC Output Current – Low Impedance Bidirectionals	24	mA
Power Dissipation (VCC = 5.25V)	500	mW
Electrostatic Discharge Voltage (Human Body Model) (I < 1uA)	+/- 3000	V
Latch Up Current (Vi = +/- 10V maximum, for 10 ms)	+/-200	mA

Se tienen las características eléctricas que son muy importantes a la hora del diseño. Estas características se observan en la tabla 3-4

Tabla 3-4 Características Eléctricas

Operating Voltage and Current

Parameter	Description	Min	Typ	Max	Units	Conditions
Vcc1	VCC Operating Supply Voltage	4.35	5.0	5.25	V	
Vcc2	VCCIO Operating Supply Voltage	3.0	-	5.25	V	
Icc1	Operating Supply Current	-	25	-	mA	Normal Operation
Icc2	Operating Supply Current	-	180	200	uA	USB Suspend **Note 3

UART IO Pin Characteristics (VCCIO = 5.0V)

Parameter	Description	Min	Typ	Max	Units	Conditions
Voh	Output Voltage High	3.2	4.1	4.9	V	I source = 2mA
Vol	Output Voltage Low	0.3	0.4	0.6	V	I sink = 2mA
Vin	Input Switching Threshold	1.3	1.6	1.9	V	**Note 4
VHys	Input Switching Hysteresis	50	55	60	mV	

UART IO Pin Characteristics (VCCIO = 3.0 - 3.6V)

Parameter	Description	Min	Typ	Max	Units	Conditions
Voh	Output Voltage High	2.2	2.7	3.2	V	I source = 1mA
Vol	Output Voltage Low	0.3	0.4	0.5	V	I sink = 2 mA
Vin	Input Switching Threshold	1.0	1.2	1.5	V	**Note 4
VHys	Input Switching Hysteresis	20	25	30	mV	

XTIN / XTOUT Pin Characteristics

Parameter	Description	Min	Typ	Max	Units	Conditions
Voh	Output Voltage High	4.0	-	5.0	V	Fosc = 6MHz
Vol	Output Voltage Low	0.1	-	1.0	V	Fosc = 6MHz
Vin	Input Switching Threshold	1.8	2.5	3.2	V	

RESET#, TEST, EECS, EESK, EEDATA Pin Characteristics

Parameter	Description	Min	Typ	Max	Units	Conditions
Voh	Output Voltage High	3.2	4.1	4.9	V	I source = 2mA
Vol	Output Voltage Low	0.3	0.4	0.6	V	I sink = 2 mA
Vin	Input Switching Threshold	1.3	1.6	1.9	V	**Note 5
VHys	Input Switching Hysteresis	50	55	60	mV	

RSTOUT Pin Characteristics

Parameter	Description	Min	Typ	Max	Units	Conditions
Voh	Output Voltage High	3.0	-	3.6	V	I source = 2mA
Vol	Output Voltage Low	0.3	-	0.6	V	I sink = 2mA

USB IO Pin Characteristics

Parameter	Description	Min	Typ	Max	Units	Conditions
UVoh	IO Pins Static Output (High)	2.8		3.6	V	RI = 1.5K to 3V3Out (D+) RI = 15K to GND (D-)
UVol	IO Pins Static Output (Low)	0		0.3	V	RI = 1.5K to 3V3Out (D+) RI = 15K to GND (D-)
UVse	Single Ended Rx Threshold	0.8		2.0	V	
UCom	Differential Common Mode	0.8		2.5	V	
UVDif	Differential Input Sensitivity	0.2			V	
UDrvZ	Driver Output Impedance	29		44	Ohm	**Note 5

Por otra parte, lo que corresponde a la asignación de pines, se observan en la figura 3-5 mientras que las dimensiones del tamaño se aprecian en la figura 3-6

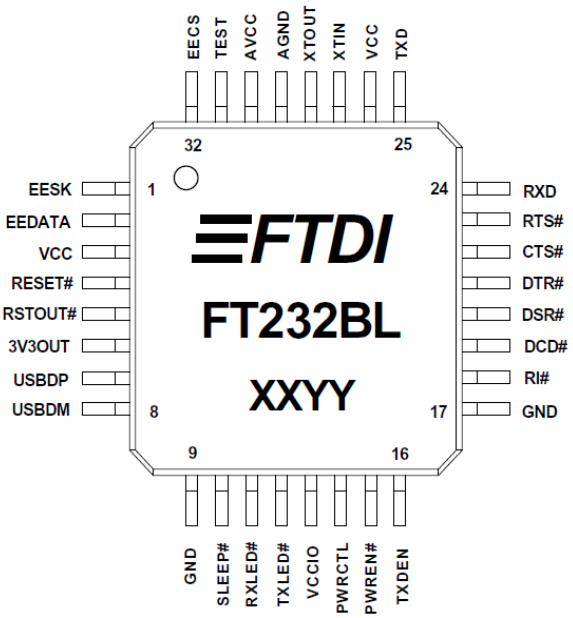

Figura 3-5 Asignación de pines.

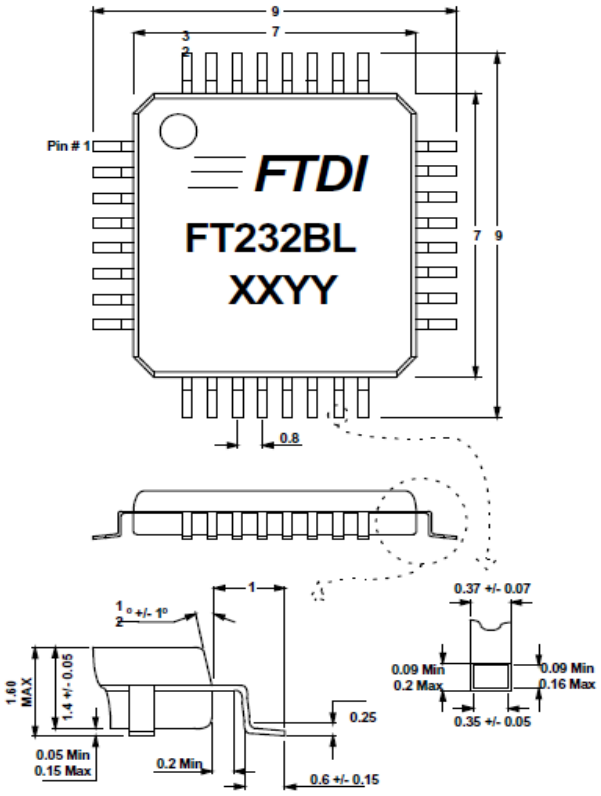

Figura 3-6 Dimensiones del encapsulado.

3.6.1. DIAGRAMA A BLOQUES DEL FT232BL

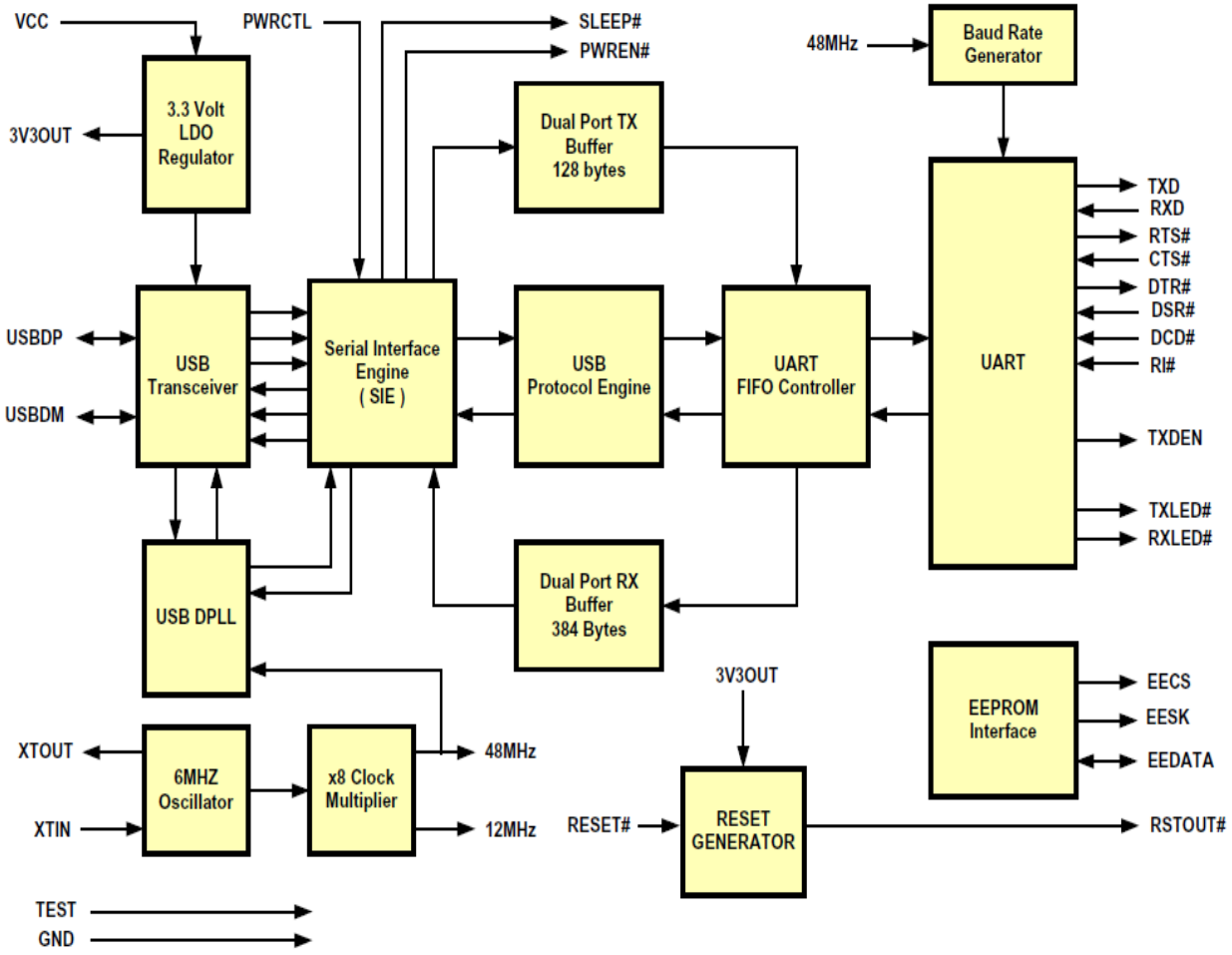

Figura 3-7 Diagrama a bloques FT232BL.

3.6.2. DESCRIPCION DE FUNCION DE LOS BLOQUES DEL FT232

- **Regulador LDO 3.3 Volt:** El regulador LDO 3.3V genera la tensión de 3,3 voltios de referencia para el bloque transmisor-receptor USB. Se requiere de un condensador externo que se conecta al pin 3V3OUT, también proporciona 3.3V de alimentación a RSTOUT #. La función principal de este bloque es para alimentar el receptor-transmisor USB y el generador reset sin necesidad de tener una fuente externa.

- **Transmisor-Receptor USB (USB TRANSCEIVER):** El bloque transmisor-receptor USB (Transcerver USB) ofrece la interfaz física para las velocidades de USB 1.1/ USB 2.0.

- **USB DPLL:** El bloque USB DPLL bloquea la entrada del código NRZI del USB y dispone de un reloj que recupera las señales de datos del bloque SIE.

- **6MHz Oscillator:** El bloque del oscilador de 6 MHz genera una frecuencia de trabajo de referencia de 6 MHz de reloj de entrada del bloque de reloj multiplicador x8 también el cual necesita un Reloj de un cristal de 6MHz externo ó un resonador cerámico.

- **x8 Clock Multiplier:** El multiplicador de reloj x8 toma la entrada de 6MHz del bloque oscilador y genera un reloj de referencia de 12MHz para el SIE USB Protocolo y el bloque de regulación UART FIFO. También genera un reloj de referencia de 48MHz para el DPPL USB y el bloque de velocidad de transmisión.

- **Interfaz Serie SIE:** El bloque Interfaz serie (SIE) realiza la conversión serie paralelo, paralelo serie de los datos a USB. De acuerdo con la especificación USB 2.0.

- **Protocolo USB:** El bloque de Protocolo USB gestiona el flujo de datos desde el extremo del dispositivo de control USB. Maneja el bajo nivel de protocolo USB las solicitudes generadas por el controlador de host USB y los comandos para controlar los parámetros funcionales de la UART.

- **Dual Port TX Buffer (128 bytes):** Los datos transmitidos del USB se almacena en el Dual Port TX Buffer eliminándolos de la memoria del registro de transmisión UART que controla el bloque UART FIFO.

- **Dual Port RX Buffer (384 bytes):** Los datos del registro de recepción de la UART se almacenan en el Dual Port RX Buffer antes de ser eliminados por la SIE en una solicitud de USB para los datos del dispositivo final.

- **Controlador UART FIFO:** El controlador UART FIFO se encarga de la transferencia de datos entre los RX/TX Buffers y la transmisión/recepción de registros de la UART.

- **UART:** La UART realiza la transmicion-recepcion asíncrona de 8/7 bits de conversión paralelo a serie y serie a paralelo de los datos sobre el protocolo RS232 (RS422 y RS485). Las señales de control de apoyo de la UART incluyen RTS, CTS, DSR, DTR, DCD y RI. La UART ofrece un transmisor de señal de habilitación de control (TXDEN) para ayudar en la interfaz con transmisores-receptores RS-485. La UART soporta las opciones RTS/CTS, DSR/DTR.

- **Baud Rate Generator:** La velocidad de transmisión del generador proporciona una entrada de reloj x16 a la UART del reloj de referencia de 48MHz y consta de un preescalador de 14 bits y 3 bits de registro que proporcionan la regulación fina de la velocidad de transmisión. Esto determina la velocidad en baudios de la UART que es programable de 183 baudios a 3 millones de baudios.

3.7. DIAGRAMA DEL MODULO DE INTERFAZ USB

A continuación se presenta el diagrama eléctrico el cual se uso para realizar el modulo de la interfaz para poder enviar los datos por USB el cual lo propone el fabricante del dispositivo figura 3-8.

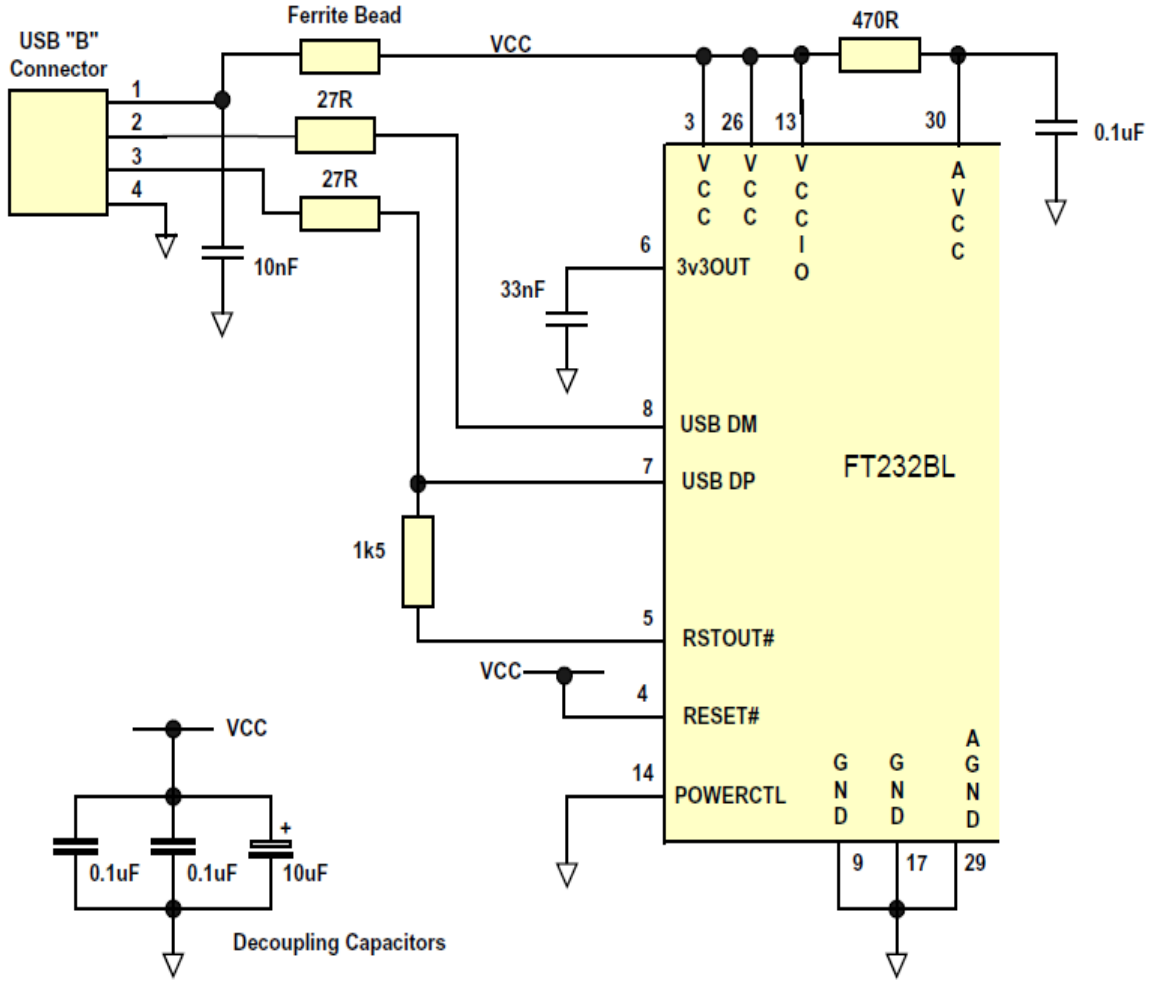

Figura 3-8 Diagrama de interfaz USB

3.8. ¿QUE ES UN INSTRUMENTO?

Un instrumento es un objeto fabricado (una herramienta, un aparato, etc.) que se
encarga de recoger una señal de determinada naturaleza y procesarla, para mostrar o registrar su valor haciendo uso de un sistema de representación electromagnético (instrumentos de aguja) electrónico (display), registro gráfico sobre el papel, señal acústica, óptica, etc. Pero se podría decir que es una herramienta que si, se observa y se controla, se puede utilizar para aprender y conocer mejor el universo que nos rodea. Incluso, esta definición puede ya pertenecer al pasado, dado que la incorporación de la computadora en los procesos de medida significa que los instrumentos clásicos que se utilizaban, hasta hace unos años, se ven sustituidos por el monitor de una computadora o por una unidad de almacenamiento de datos, un módem, etc. La posibilidad de interaccionar con una computadora, en el proceso de medida, ha permitido la creación de verdaderos equipos de instrumentación, basados en una arquitectura computacional auxiliar de un hardware adecuado.

El desarrollo tecnológico en todos los niveles que se ha producido en los últimos años, especialmente en el campo de la electrónica, hace que las empresas y los centros de investigación dispongan de instrumentos cada vez más competitivos. Esto se mantendrá en tanto que el instrumento sea fácil de utilizar, se integre sin complicaciones en un sistema de medida basado en computadora, y sea flexible, es decir, que se adapte fácilmente a las necesidades de cambios en la metodología de medida. De esta manera dispondremos de instrumentos y de sistemas de instrumentación especializados para cada campo, abiertos a diferentes configuraciones de medida

3.9. INSTRUMENTACIÓN VIRTUAL

Un instrumento virtual consiste de una computadora del tipo industrial, o una estación de trabajo, equipada con poderosos programas (software), hardware económico, tales como placas para insertar, y manejadores (drivers) que cumplen, en conjunto, las funciones de instrumentos tradicionales. Los instrumentos virtuales representan un apartamiento fundamental de los sistemas de instrumentación basados en el hardware a sistemas centrados en el software que aprovechan la potencia de cálculo, productividad, exhibición y capacidad de conexión de las populares computadoras de escritorio y estaciones de trabajo. Aunque la PC y la tecnología de circuitos integrados han experimentado avances significativos en las últimas dos décadas, es el software el que realmente provee la ventaja para construir sobre esta potente base de hardware para crear los instrumentos virtuales, proveyendo mejores maneras de innovar y de reducir los costos significativamente. Con los instrumentos virtuales, los ingenieros y científicos construyen sistemas de medición y automatización que se ajustan

exactamente a sus necesidades (definidos por el usuario) en lugar de estar limitados por los instrumentos tradicionales de funciones fijas (definidos por el fabricante).

La utilización de las computadoras, se ha hecho fundamental dentro de la infraestructura de cualquier disciplina tecnológica. Algunas ramas de la industria dependen de la ayuda de las computadoras, como las cadenas de producción, las comunicaciones, el transporte, los laboratorios de investigación y los sistemas de prueba y medida. El punto de inicio es el sistema físico que es de donde se recogen los datos, que también es el punto final, ya que sobre él es donde se actúa a partir de las órdenes de control. Dentro de este sistema se produce un proceso físico, el cual se puede definir como una combinación de operaciones que se ejecutan con el fin de efectuar alguna actuación o cambio sobre el sistema. El proceso físico se puede caracterizar por una serie de elementos de entrada y salida de materiales, energía e información. Los materiales y la energía son componentes básicos en todo proceso, mientras que la información es una parte indispensable que nos ayuda a controlar y desarrollar en las mejores condiciones las pautas de este proceso.

Ya que se tiene en cuenta que las computadoras son dispositivos excelentes para procesar información capturada del proceso físico, lo que se puede hacer es utilizar la computadora para controlar la adquisición, el procesado y el control de la actuación final sobre el proceso. En la figura 3-9 se observa el modelo de la función que desarrolla la computadora.

Figura 3-9 Función de la Computadora

Entonces, a partir de la información de entrada y de salida de la computadora, puede hacerse un intercambio de información hacia el proceso físico, como nos lo representa la figura 3-10.

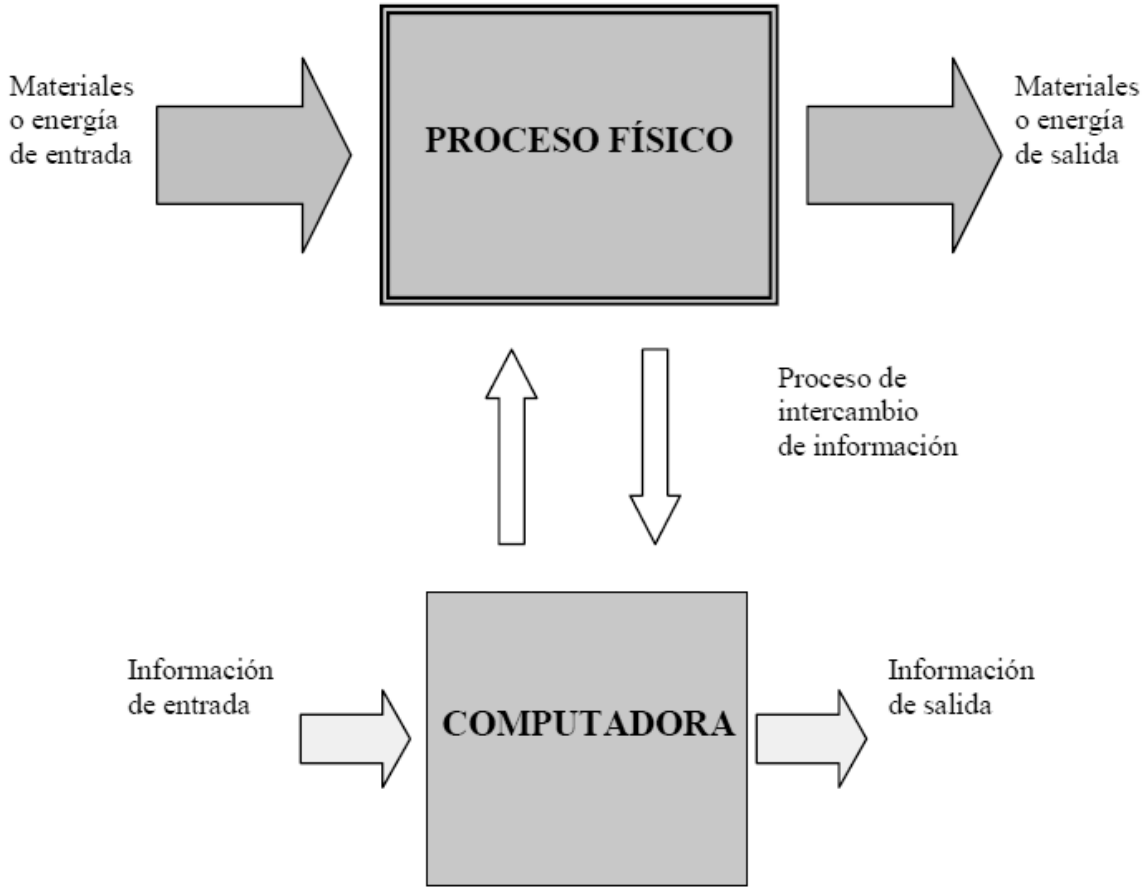

Figura 3-10 Utilización de la computadora en el control de un proceso.

La obtención de resultados óptimos a partir de un sistema de adquisición de datos basado en computadora, depende de cada uno de los elementos que se utilicen en el sistema. Estos elementos pueden ser: la computadora, los transductores, los actuadores, el acondicionamiento de la señal, la circuitería de adquisición de datos, la circuitería de análisis de los datos, el control y el software. Esto se visualiza en la figura 3-11, que es un sistema genérico en el cual se consideran una serie de elementos esenciales.

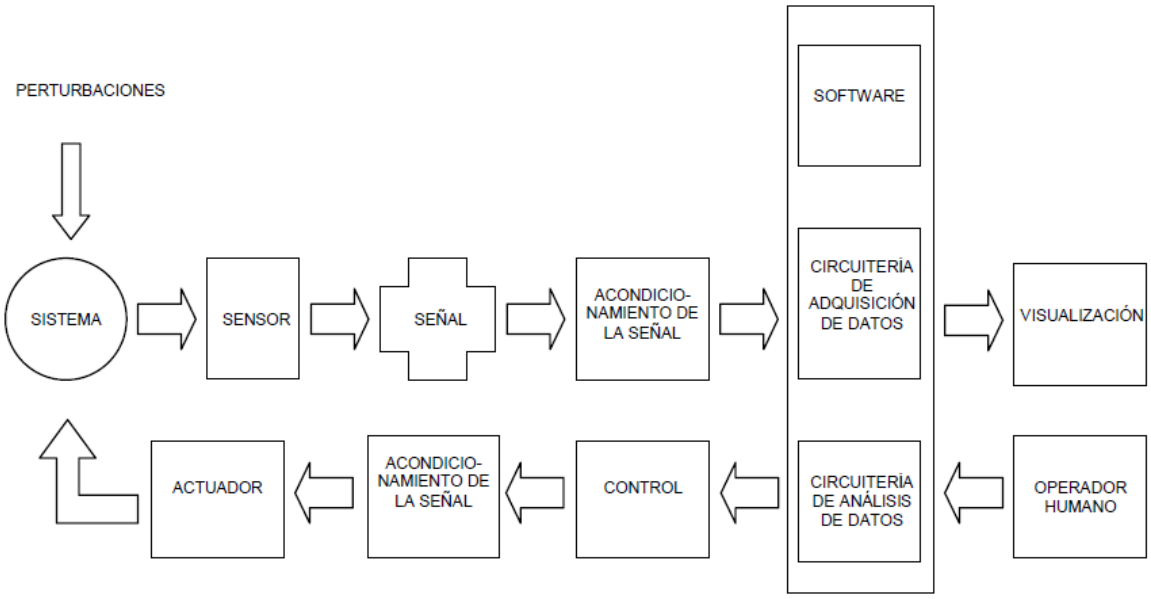

Figura 3-11 Sistema de adquisición de datos y de control.

Describiendo estos elementos, se puede decir que la computadora va a determinar la velocidad de proceso del sistema; tomando en cuenta que, en aplicaciones que requieran de un proceso en tiempo real de señales de alta frecuencia, se necesitará de una computadora potente, y para procesos más simples se pueden utilizar procesadores más lentos. El transductor es capaz de sensar el fenómeno físico y suministrar una señal eléctrica que puede ser aceptada por el sistema de adquisición. Un ejemplo de ello, un sensor de temperatura que transforma una temperatura en una señal eléctrica analógica que, convertida en digital mediante un convertidor analógico – digital (CAD), puede ser tratada por la computadora.

El acondicionamiento de la señal que surge del transductor, que en este caso es una señal eléctrica, tiene que ser tratada, convertida o escalada de forma que puede ser aceptada por el sistema de adquisición. Estas formas de acondicionamiento de la señal pueden ser la amplificación, la linealización y el aislamiento, que son las más comunes.

En la circuitería de adquisición de datos se puede o no incorporar una tarjeta que se ocupe de adquirir la señal analógica y que realice la conversión digital. Lo que refiere a la circuitería de análisis, se puede mencionar que la capacidad de proceso de las computadoras personales actuales se ha incrementado hasta el punto de tener potencia de cómputo suficiente para muchas aplicaciones de adquisición de datos y de análisis de resultados.

Por medio de los programas de aplicación (software) se logra un sistema completo de adquisición, análisis y presentación de resultados. De hecho, la circuitería de adquisición sin el software adecuado es inoperante, y lo mismo se podría decir de los programas sin la circuitería adecuada.

3.10. COMPONENTES DE UN SISTEMA BASADO EN INSTRUMENTACION VIRTUAL.

Los sistemas basados en la PC que permiten desarrollar aplicaciones con Instrumentación Virtual, tienen una estructura que se puede dividir en tres componentes fundamentales: Hardware de Adquisición de Datos, Acondicionamiento de la señal y la PC y software. Sin embargo, en cualquier aplicación con estos sistemas surgen otros elementos tales como: Transductores, Bloques Terminales, Cables conectores, etc. De esta forma el diagrama de bloques de una aplicación de Instrumentación se puede ver en la figura 3-12.

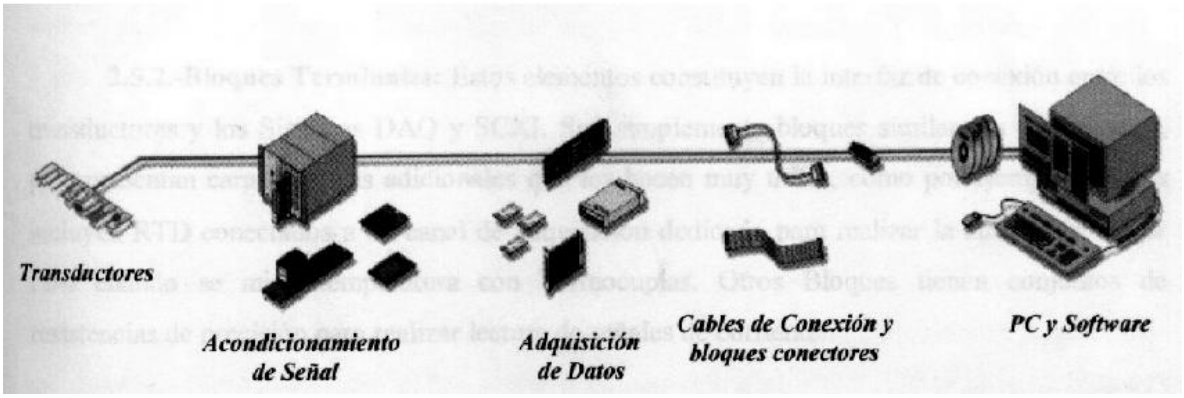

Figura 3-12 Diagrama de un sistema basado en Instrumentación virtual.

La función de cada uno de estos componentes es fundamental para el desarrollo de aplicaciones en Instrumentación Virtual, es por esto que se definirán brevemente las características de cada uno de estos.

3.10.1. TRANSDUCORES

Son los elementos que tienen la capacidad de transformar una señal en una de naturaleza distinta. Es así como se tienen transductores que pueden censar un fenómeno físico, entregando una señal eléctrica con la información necesaria para interpretarlo y otros que pueden actuar sobre un fenómeno determinado mediante la generación de señales. Las señales provenientes de ciertos transductores son ingresadas al sistema de adquisición de datos en forma de Voltaje ó Corriente, dependiendo esto último exclusivamente del transductor y de la tarjeta de Adquiridora.

Si tenemos transductores que se relacionan directamente con los actuadores (Generación de señal), las señales provenientes del Sistema de adquisición, ya sean de Voltaje o Corriente deben conectarse en forma adecuada a estos transductores de acuerdo a su principio de funcionamiento, teniendo en cuenta las limitaciones de las tarjetas de adquisición en la entrega de señales, especialmente limitaciones de corriente.

3.10.2. BLOQUES TERMINALES

Estos elementos constituyen la interfaz de conexión entre los transductores y los sistemas de adquisición y acondicionamiento de señal. Son simplemente bloques similares a las borneras pero presentan características adicionales que los hacen muy útiles, como por ejemplo algunos incluyen RTD conectados a un canal de adquisición dedicado para realizar la compensación cero cuando se mide temperatura con termocuplas. Otros bloques tienen conjuntos de resistencias de precisión para realizar lectura de señales de corriente.

3.10.3 HARDAWARE DE ACONDICIONAMIENTO DE SEÑAL

Las señales eléctricas generadas por los transductores deben ser acondicionadas para poder ser adquiridas por el Hardware. Los accesorios de acondicionamiento permiten Amplificar, Aislar y Filtrar para realizar mediciones mas exactas y además, Multiplexar y Excitar transductores como Strain Gauges y RTD. Además permiten aumentar el número de canales para aplicaciones que así lo requieren.

3.10.4. HARDWARE DE ADQUISICION DE DATOS

Estos dispositivos son fundamentales para implementar cualquier aplicación en instrumentación Virtual, ya que permiten relacionar en forma directa el concepto Virtual desarrollado en la computadora y en el concepto de instrumentación que se desarrolla en terreno, ahí donde están los sensores y actuadores, transformándose así en una interfaz absolutamente necesaria.

3.10.5. CABLES DE CONEXIÓN

La necesidad de conectar equipos entre si es evidente, mas a un cuando se trata de dispositivos de naturaleza distinta, por ejemplo PC y Hardware de adquisición. Es por esto que cada etapa de la aplicación deberá ser cableada con un tipo de cable conector apropiado y que cumpla las condiciones mínimas de compatibilidad para asegurar un correcto funcionamiento. Los cables conectores típicos son los que presentan aislación permitiendo tener mayor inmunidad al ruido y una conexión robusta y los cables planos que por lo general son de muy bajo costo, flexibles y menos robustos siendo de esta forma, más propenso al ruido.

3.10.6. COMPUTADORA

La computadora, ya sea una PC ó computadora portátil puede afectar drásticamente el desempeño de un sistema, principalmente porque puede potenciar o limitar las aplicaciones de Software y de Adquisición de datos. Hoy en día la tecnología basada en procesadores Pentium, AMD y

agregando el alto desempeño de la arquitectura de los buses PCI, USB, entregan una herramienta poderosa, sobre todo, en la velocidad. Además con la posibilidad de utilizar PCMCIA para PC portátiles, permiten desarrollar aplicaciones que entregan una mayor flexibilidad y movilidad. El desarrollo que ha tenido la transferencia de datos DMA en algunas arquitecturas de computadoras, permiten incrementar la tasa de transferencia y así la velocidad del sistema. De acuerdo a la arquitectura de la computadora y las características del Hardware se debe escoger el sistema operativo y el software de aplicación que entreguen mayores beneficios a la hora de establecer aplicaciones.

3.10.7. SOFTWARE

El Software transforma a la PC, Hardware de Adquisición y Acondicionamiento de señales en un sistema completo de adquisición de señales, análisis, procesamiento y visualización de datos, es decir, es el último y tal vez uno de los más importantes bloques para realizar una aplicación de Instrumentación Virtual.
A la hora de elegir el Software adecuado se debe considerar entre otras cosas:

- Compatibilidad con el Hardware de adquisición y Acondicionamiento.
- Funcionalidad expresada en Drivers para manejar un determinado hardware.
- Sistema operativo bajo el cual opera.
- Potencialidad y Flexibilidad.
- Dificultad y complejidad en la programación.

De acuerdo a las características antes mencionadas, el usuario podrá implementar y desarrollar sistemas a la medida de sus necesidades, entregándote una poderosa herramienta para realizar expansiones, modificaciones y generación de nuevos sistemas y aplicaciones en Instrumentación, Control, Monitoreo y Automatización de procesos Industriales.

3.11. EL SOFTWARE EN LA INSTRUMENTACIÓN VIRTUAL

En un instrumento virtual, el software es el componente más importante, ya que el
Ingeniero puede crear eficientemente sus propias aplicaciones con él, diseñando e integrando las rutinas que requiere un proceso en particular; también puede crear la interfaz de usuario que mejor satisfaga el objetivo de la aplicación. También puede definir cómo y cuándo la aplicación adquiere datos desde el dispositivo, cómo los procesa, manipula y almacena, y cómo se presentan los resultados al usuario. Si se cuenta con un software poderoso, a los instrumentos virtuales se les puede dotar con capacidades

de inteligencia y de toma de decisiones, de manera tal que se adapten cuando las señales medidas varíen inadvertidamente, o cuando se requiera mayor o menor potencia de procesamiento.

Una importante ventaja que provee el software es que se puede dividir en varios módulos. Esto se puede dar cuando, en un gran proyecto, los miembros del equipo de trabajo pueden abordar una tarea dividiéndola en unidades funcionales manejables, así, diseñando un instrumento virtual para solucionar cada una de estas tareas, y posteriormente reunirlas en un sistema completo, quedaría terminado el proyecto.

3.12. LABVIEW COMO HERRAMIENTA PARA CREAR INSTRUMENTOS VIRTUALES

LabVIEW es una parte integral de la instrumentación virtual, dado que provee un medio ambiente de desarrollo de aplicaciones que es fácil de utilizar y está diseñado específicamente para las necesidades de ingenieros y científicos. Por otra parte, LabVIEW ofrece poderosas características que facilitan la conexión a una gran variedad de hardware y otros software's. Una de las características más poderosas que LabVIEW ofrece a los ingenieros y científicos es un medio ambiente de programación, que es gráfico. Se pueden crear instrumentos virtuales y de igual manera crear las interfaces gráficas de usuario en la pantalla de la computadora con la cual se puede:

- Operar el programa de instrumentación.
- Controlar el hardware seleccionado.
- Analizar datos adquiridos.
- Visualizar los resultados.

Se puede personalizar a los instrumentos virtuales con perillas, botones, diales y gráficos, a fin de emular paneles de control de instrumentos tradicionales, crear paneles de ensayo personalizados, o representar visualmente el control y operación de procesos. La similitud existente entre los diagramas de flujo y los programas gráficos acorta la curva de aprendizaje, asociada con lenguajes tradicionales basados en texto.

Se puede determinar el comportamiento de los instrumentos virtuales conectando íconos entre sí para crear diagramas de bloques, que son notaciones de diseño naturales para ingenieros y científicos. Con el lenguaje gráfico, se desarrollan sistemas más rápidamente que con lenguajes de programación convencionales, mientras que se conserva la potencia y flexibilidad necesarias para crear una variedad de aplicaciones.

LabVIEW posee bibliotecas listas para ser utilizadas con el objeto de integrar instrumentos autónomos, equipos de adquisición de datos, productos para el control de movimiento y de visión, instrumentos GPIB/IEEE 488 y serie

RS-232, y PLC, entre otros, lo cual permite construir una solución completa de medición y automatización.

LabVIEW también tiene incorporadas las más importantes normas de instrumentación, tal como VISA, que es una norma que permite la operación de instrumentos GPIB, serie y VXI; PXI y software y hardware basados en la norma PXI *Sistems Alliance Compact* PCI; manejadores de instrumentos virtuales intercambiables IVI y VXI plug & play, que es un manejador para la norma que rige la instrumentación VXI. Un gran número de fabricantes de hardware y software desarrollan y mantienen centenares de bibliotecas de LabVIEW y manejadores de instrumentos, que nos ayudan a utilizar fácilmente sus productos con LabVIEW.
Este software ofrece maneras simples de incorporar programas en ActiveX, bibliotecas dinámicas (DLL) y bibliotecas compartidas de otras herramientas. Además se puede compartir código hecho en LabVIEW como una DLL, construir un programa ejecutable, o utilizar ActiveX. LabVIEW es un producto versátil, dado que se utiliza con una sola computadora equipada con este software para innumerables aplicaciones y propósitos. No sólo es versátil sino también extremadamente efectivo, desde el punto de vista del costo. La instrumentación virtual con LabVIEW demuestra ser económica, no sólo por los reducidos costos de desarrollo, sino también porque preserva la inversión del capital a lo largo de un extenso periodo. A medida que cambian sus necesidades, se pueden fácilmente modificar los sistemas, sin necesidad de adquirir nuevo equipamiento, y crear bibliotecas enteras de instrumentación a menor costo que el correspondiente a un solo instrumento comercial tradicional.

La mayoría de los sistemas computacionales utilizan alguna variante del sistema operativo Microsoft Windows. No obstante a ello, existen otras opciones que ofrecen claras ventajas para ciertos tipos de aplicaciones. El desarrollo de sistemas operativos de tiempo real y embebido continúa creciendo rápidamente en la mayoría de las industrias, a medida que la capacidad de cálculo es incorporada en paquetes más especializados y pequeños. Por ello, es importante minimizar las pérdidas resultantes del cambio hacia nuevas plataformas, y la elección del software correcto para dicho objetivo es un factor clave.

Estas preocupaciones las minimiza LabVIEW, ya que corre en versiones de Windows 95 y superiores, así como también sobre Mac OS, Sun Solares y Linux. Con LabVIEW también se puede compilar código que corra en sistemas operativos de tiempo real. Dada la importancia de los sistemas, National Instruments continúa poniendo a disposición versiones más antiguas de LabVIEW para los sistemas operativos Windows, Mac OS y Sun. Con esto se dice que LabVIEW es independiente de la plataforma seleccionada, ya que los instrumentos virtuales que se crean en una plataforma, se pueden transportar de manera transparente a cualquier otra plataforma LabVIEW, simplemente abriendo el instrumento virtual.

LabVIEW incluye un amplio conjunto de herramientas de visualización para presentar datos en la interfaz del usuario de la instrumentación virtual, tanto para gráficos continuos como también para visualización de gráficos 2D y 3D, en donde se puede reconfigurar de manera instantánea los atributos de la presentación de datos, tales como los colores, tamaño de fuentes, tipos de gráficos y más, así como también efectuar rotación, enfoque y desplazamiento dinámico en estos gráficos con el ratón.

3.13. LABVIEW

La palabra LabVIEW está formada por las iniciales de ***L**aboratory **V**irtual **I**nstrument **E**ngineering **W**orkbench*, el cual es un entorno gráfico para el desarrollo de aplicaciones en el campo de la instrumentación, desde la adquisición de datos hasta el control remoto de instrumentos. El entorno dispone de librerías matemáticas para el análisis de datos y de los *drivers* de control de varios instrumentos.

Figura 3-13 Icono de software LabVIEW

Los programas de LabVIEW se denominan instrumentos virtuales (VI, *Virtual Instrument*, en ingles), porque la apariencia de su interfaz con el usuario es la de un instrumento de laboratorio. Estos VI son equivalentes a las funciones de C o a los procedimientos de Pascal. Un VI consta de dos partes bien diferenciadas, el Panel Frontal (*Front Panel*) y el Diagrama de Bloques (*Block Diagram*). El panel frontal es la interfaz del programa con el usuario. En él están representadas todas las entradas y salidas del programa. Por analogía a un instrumento real, las entradas del panel frontal se llaman controles y las salidas, indicadores.

El diagrama de bloques es el código de programación escrito en lenguaje gráfico. Los distintos componentes del diagrama de bloques son los nodos del programa. Los componentes están conectados unos con otros. Estas interconexiones definen el flujo de datos en el diagrama de bloques. En la figura 3-14 se observa el panel frontal del Sistema de Monitoreo Para Acuario que se diseño.

Figura 3-14 Panel Frontal del sistema de Monitoreo para Acuario

En la figura 3-15 se puede ver el diagrama a bloques del sistema de monitoreo para acuario.

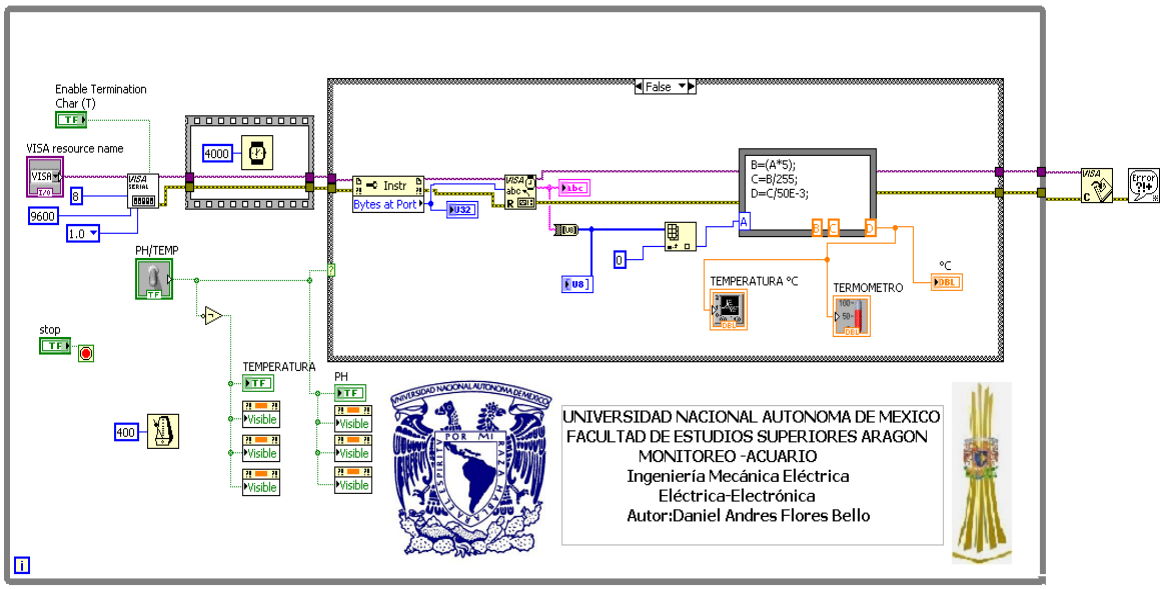

Figura 3-15 Diagrama a Bloques del Sistema de Monitoreo para Acuario

3.13.1. PROGRAMACION CON LABVIEW

Para empezar un programa en LabVIEW se debe seleccionar la opción **New VI** en la ventana que aparece cuando se abre el programa. Posterior a esta selección se crearán dos ventanas vacías, en donde una corresponde al panel frontal y la otra al diagrama a bloques.

El diseño del programa se suele empezar en el panel frontal, porque es donde se debe decidir cómo será la interfaz de usuario, es decir, qué entradas y salidas tendrá el programa.

Dentro del panel frontal se va a encontrar una paleta de herramientas (figura 3-16), que sirve para editar, modificar y depurar VI's.

Figura 3-16 Paleta de Herramientas

La paleta de controles (figura 3-17), que sirve para crear el panel frontal.

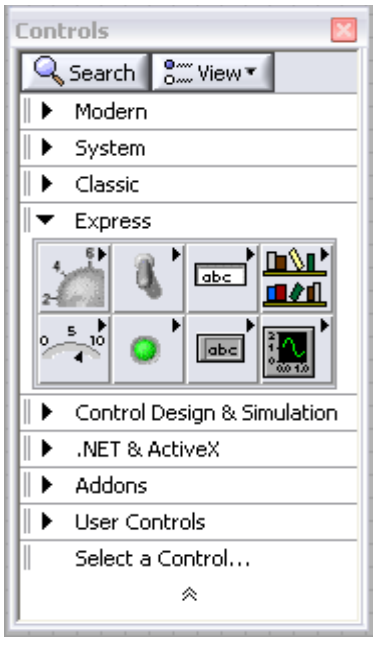

Figura 3-17 Paleta de Controles

Ahora, pasando a lo que es la ventana del diagrama a bloques, se encuentra una paleta de funciones (figura 3-18), la cual permite hacer la programación correspondiente al programa a realizar. Otra de las cosas que se observan en el diagrama a bloques son las referencias de lo que se haya puesto en el panel frontal, ya sea indicadores o controles.

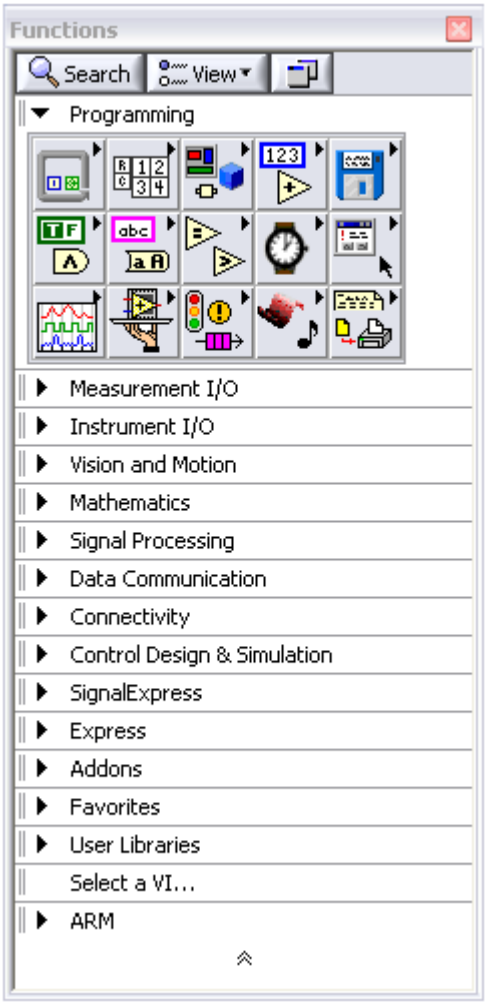

Figura 3-18 Paleta de Funciones

En el mismo diagrama de bloques se deben realizar las conexiones de los distintos elementos, que indiquen el flujo con que se van a realizar las distintas operaciones.

3.13.2. EJECUCIÓN DE UN PROGRAMA EN LABVIEW

Ya que se encuentra listo el programa, el siguiente paso es ejecutarlo (correr el programa); en LabVIEW hay varias formas de hacerlo, estas distintas opciones pueden encontrarse en la barra de tareas (figura 3-19) del diagrama a bloques.

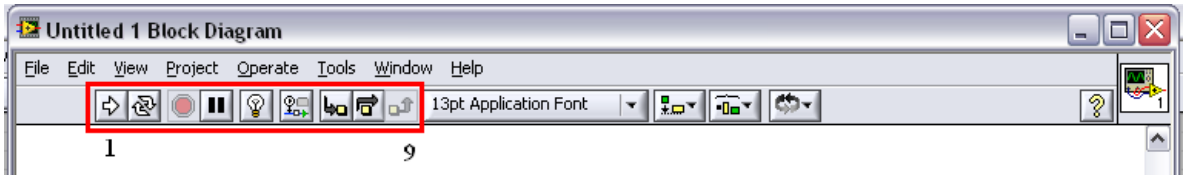

Figura 3-19 Barra De Tareas

A continuación, se describen las formas de correr un programa, de acuerdo a las opciones que se encuentran en la barra de tareas:

1. Esta opción hace que se ejecute el programa, a la vez LabVIEW compilará el programa, si es necesario.

2. Se hará la ejecución continua hasta que se apriete el botón de pausa o se finalice la ejecución.

3. Finaliza la ejecución.

4. Botón de pausa.

5. Este modo permite ver una animación de cómo se ejecuta el VI.

6. Permite colocar puntos de prueba para ver el flujo de los datos.

7. Ejecuta el primer paso de una estructura, o un subVI, y se para antes de ejecutar el siguiente paso.

8. Hace el inicio de ejecución paso a paso. Ejecuta una estructura, o un subVI, y se para en el siguiente nodo.

9. Finaliza la ejecución de una estructura, diagrama de bloques o VI, y se para.

Ahora, ya que se ha ejecutado un VI y existe algún error en él, aparecerá una ventana con una lista de los errores que se han encontrado. Si se quiere saber en donde se ha generado este error, se debe situar con el cursor sobre uno de los errores, apretando el botón find, con esto se indicará dónde está el error en el diagrama a bloques.

CAPÍTULO 4: AUTOMATIZACION DE ACUARIO

4.1. POTENCIAL DE HIDROGENO (pH)

El pH es una medida de la acidez o alcalinidad de una solución. El pH indica la concentración de iones hidronio [H_3O^+] presentes en determinadas sustancias. La sigla significa "potencial de hidrógeno". Este término fue acuñado por el químico danés Sørensen, quien lo definió como el logaritmo negativo de base 10 de la actividad de los iones hidrógeno. Esto es:

$$pH = - log [H +]$$

Por ejemplo, una concentración de [H_3O^+] = 1×10^{-7} M (0,0000001) es simplemente un pH de 7 ya que: pH = –log[10^{-7}] = 7

La escala de pH se establece en una recta numérica que va desde el 0 hasta el 14.El número 7 corresponde a las soluciones neutras. El sector izquierdo de la recta numérica indica acidez, que va aumentando en intensidad cuando más lejos se está del 7.Por ejemplo una solución que tiene el pH 1 es más ácida o más fuerte que aquella que tiene un pH 6. De la misma manera, hacia la derecha del 7 las soluciones son básicas y son más fuertes o más básicas cuanto más se alejan del 7. Por ejemplo, una base que tenga pH 14 es más fuerte que una que tenga pH 8.

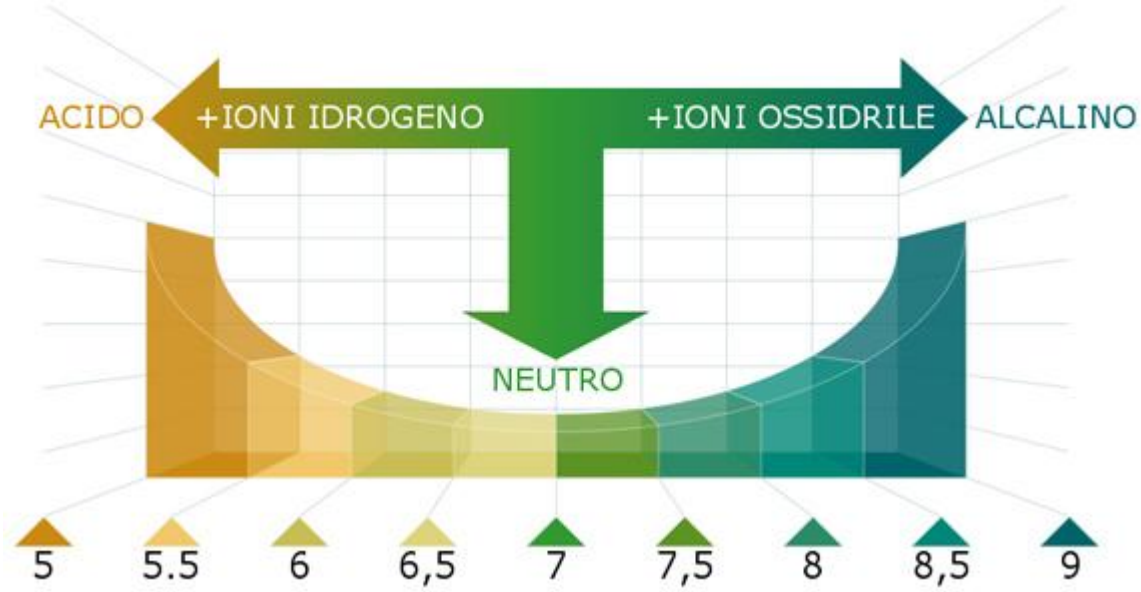

Figura 4-3 Escala de pH

4.2. TEMPERATURA

Dentro del campo de la medición de variables meteorológicas, los instrumentos clásicos utilizados pueden clasificarse en tres clases principales, basados en: la expansión térmica, el cambio de resistencia y las propiedades termoeléctricas de diversas sustancias, como una función de la temperatura. Los termómetros de mercurio y alcohol son ejemplos comunes de sensores de expansión térmica. Sin embargo, su valor es limitado en redes de monitoreo o remotas, debido a que no tienen la capacidad de registrar datos automatizados. Las estaciones meteorológicas tradicionales contienen termómetros de máxima, de mínima, termógrafos y psicrómetros.

Con lo anterior se encuentran algunas alternativas para sensores. Uno de ellos son los termopares, los cuales utilizan la tensión generada en la unión de dos metales en contacto térmico, debido a sus distintos comportamientos eléctricos. Los resistivos, que lo constituyen las RTD (*Resistance Temperature Detector*) o PT100, basadas en la dependencia de la resistividad de un conductor con la temperatura, están caracterizadas por un coeficiente de resistividad positivo PTC (*Positive Thermal Coefficient*), las NTC (*Negative Thermal Coefficient*), que se llaman termistores y están caracterizadas por un coeficiente de temperatura negativo. También se encuentran los semiconductores, que se basan en la variación de la conducción de una unión p-n polarizada directamente.

Un tipo común en los programas de medición meteorológica es el Detector de Temperatura por Resistencia (DTR). El DTR opera sobre la base de los cambios de resistencia de ciertos metales, principalmente el platino o el cobre, como una función de la temperatura. Estos dos metales son los más usados porque su resistencia muestra un aumento rigurosamente lineal con el incremento de la temperatura. Otro tipo de termómetro de cambio de resistencia es el termistor, hecho a partir de una mezcla de óxidos metálicos fusionados entre sí. Por lo general, el termistor arroja un cambio de resistencia con la temperatura mayor que el DTR. Como la relación entre la resistencia y la temperatura para un termistor no es lineal, estos sistemas generalmente están diseñados para usar una combinación de dos o más termistores y resistores fijos, que permitan obtener una respuesta casi lineal sobre un rango específico de temperatura.

La utilización de termopares exige requerimientos especiales para evitar corrientes de inducción, de fuentes cercanas de corriente alterna, que podrían ocasionar errores en la medición. Los termopares también son susceptibles al voltaje espurio causado por la humedad. Por estas razones, su uso es limitado en las mediciones rutinarias de campo.

4.3. DISEÑO DEL MODULO DE MEDICION Y TRANSMICION POR USB

Este módulo está encargado de hacer las mediciones del ambiente, al mismo tiempo que se despliegan en un LCD. Una vez que han sido visualizadas en el LCD, son transmitidas vía USB para que los visualice en un programa diseñado con instrumentación virtual. En la figura 4-1 se muestra la estructura general de la construcción del modulo de medición, el cual tiene como centro de proceso un microcontrolador.

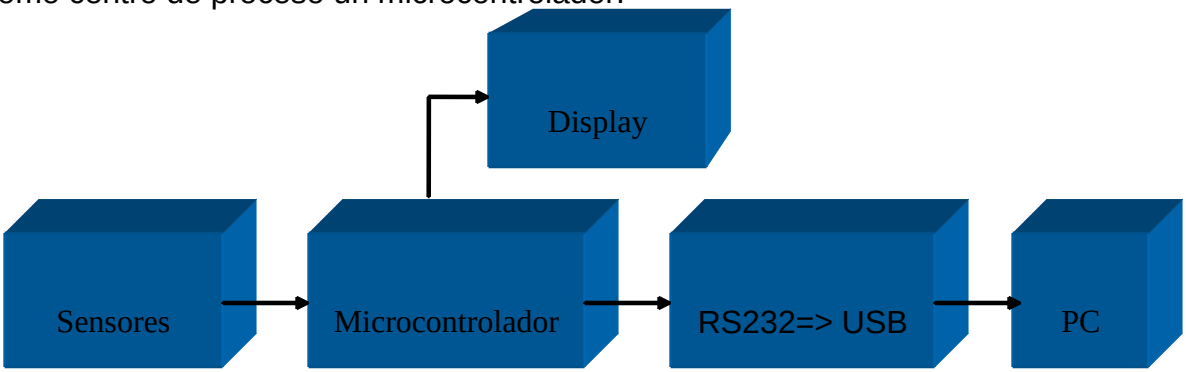

Figura 4-1 Modulo de medición.

4.3.1. SENSORES

Para empezar, se seleccionó a los sensores que se utilizarán para las mediciones de temperatura y pH. Estos son el LM35DZ de *National Semiconductor*, para la temperatura; el LM35DZ (figura 4-2) pertenece a una serie de sensores de temperatura, y que son circuitos integrados de precisión, en donde el voltaje de salida es proporcional a la temperatura en grados Celsius (centígrados). Algunas de sus características son:

- Nos proporciona la temperatura del sistema que se está midiendo en forma lineal, con una precisión de 10 mV / °C.
- Opera en un rango de 0°C a 100°C.
- Puede operar con una alimentación que va desde 4 a 30 volts.

Figura 4-2 LM35DZ

Ahora, para el pH se utiliza un sensor de pH estándar (sonda pH), es un sensor utilizado en el método electroquímico para medir el pH de una disolución. La determinación de pH consiste en medir el potencial que se desarrolla a través de una fina membrana de vidrio que separa dos soluciones con diferente concentración de protones. En consecuencia se conoce muy bien la sensibilidad y la selectividad de las membranas de vidrio delante el pH.

Una sonda para la medida de pH consiste en un par de electrodos, uno de calomel (mercurio, cloruro de mercurio) y otro de vidrio, sumergidos en la disolución de la que queremos medir el pH.

La varita de soporte del electrodo es de vidrio común y no es conductor, mientras que el bulbo sensible, que es el extremo sensible del electrodo, está formado por un vidrio polarizable (vidrio sensible de pH).

Se llena el bulbo con la solución de ácido clorhídrico 0.1N saturado con cloruro de plata. El voltaje en el interior del bulbo es constante, porque se mantiene su pH constante (pH 7) de manera que la diferencia de potencial solo depende del pH del medio externo.

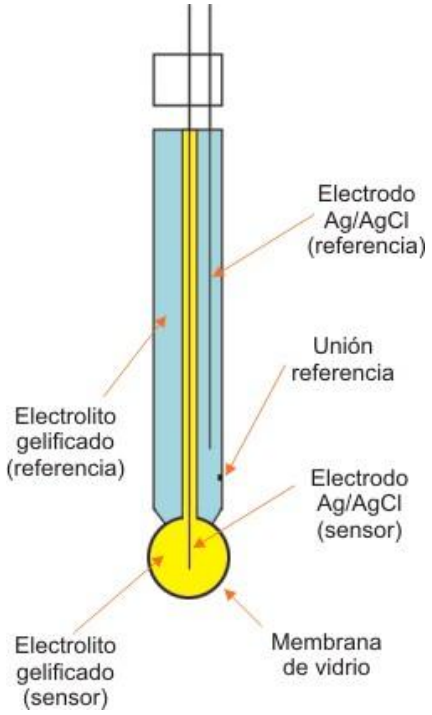

Figura 4-3 Esquema de sonda de pH

En teoría, una sonda de pH proporciona a su salida una tensión inversamente proporcional al pH del líquido, teniendo como cero el pH neutro. Es decir, para pH 7, la tensión de salida es 0V; cuando el pH es mayor que 7, la tensión de salida es negativa y cuando el pH es menor que 7 positiva, siendo los cambios de, aproximadamente, 60mV por grado de pH.

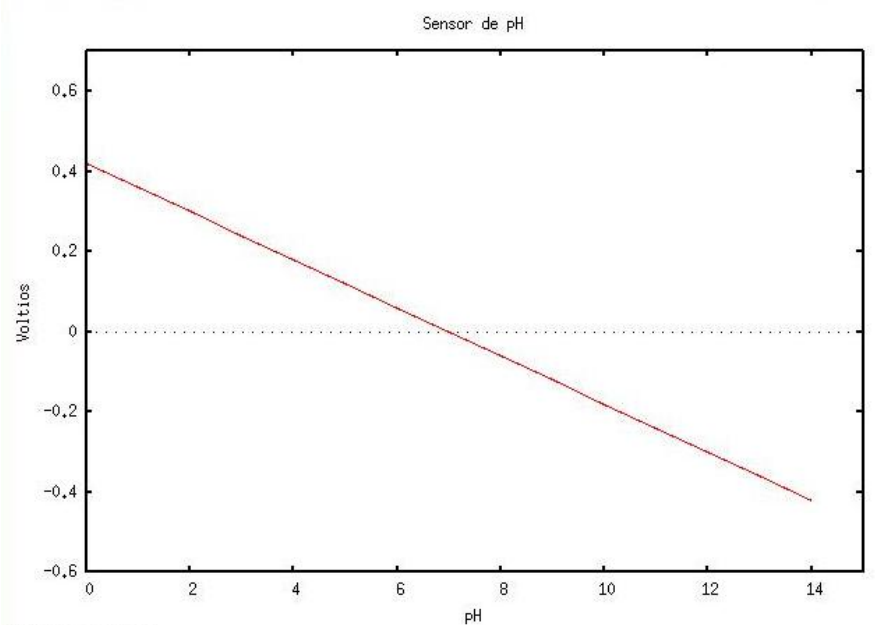

Figura 4-4 Grafica del comportamiento de la sonda de pH

4.3.2. PROGRAMANDO EL MICROCONTROLADOR

Habiendo escogido ya los sensores a utilizar, se procede a programar al microcontrolador, que en este caso es el AVR ATmega8L de Atmel *Corporation.* En este programa lo que se busca es hacer una conversión A/D para posteriormente hacer los cálculos correspondientes, dependiendo del sensor que se esté utilizando (el de temperatura o el de pH) en ese momento, para así desplegar el dato en el LCD y por ultimo enviar los datos al modulo de conversión RS232-USB para así poder recibirlos y visualizarlos en la PC. La selección del sensor se hace por medio de un switch. En la figura 4-5 se observa lo que corresponde al diagrama de flujo del programa principal (Anexo 1, Pág. 107)

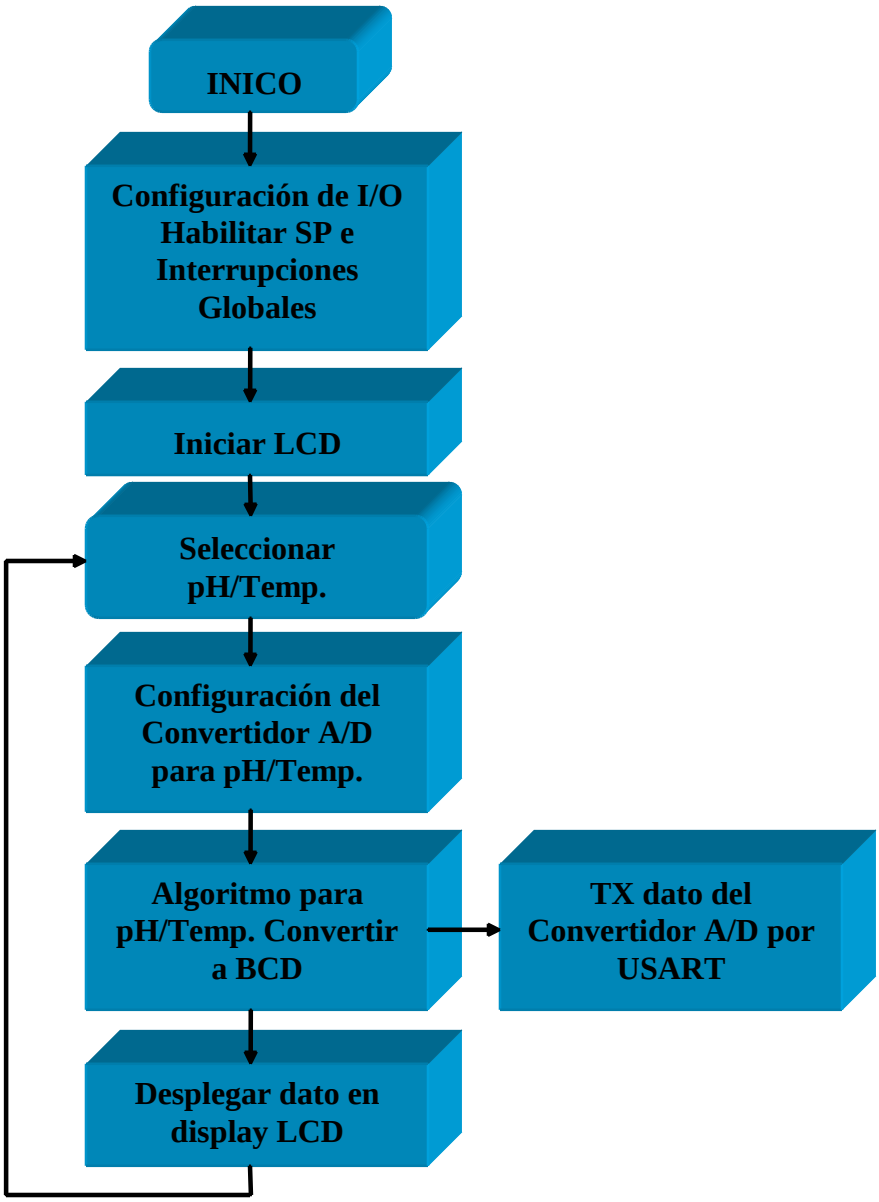

Figura 4-5 Programa principal

En el programa principal se hace la configuración del Puntero de Pila para la habilitación de llamado a Subrutinas así como la habilitación de Interrupciones y la configuración de los puertos de I/O, se hace la inicialización del LCD para que despliegue un mensaje al inicio de la operación del equipo, se configura un puerto para que lea la opción a medir ya sea el pH o la Temperatura, se configura el modulo del Convertidor A/D, se realizan los cálculos correspondientes (algoritmo) y la conversión a BCD para poder desplegar los datos en el LCD y también transmitir los datos del Convertidor A/D por el modulo USART del MC para poder visualizar el dato en la PC.

En la figura 4-6 se observa lo que corresponde al llamado de subrutina para la elección de medir pH.

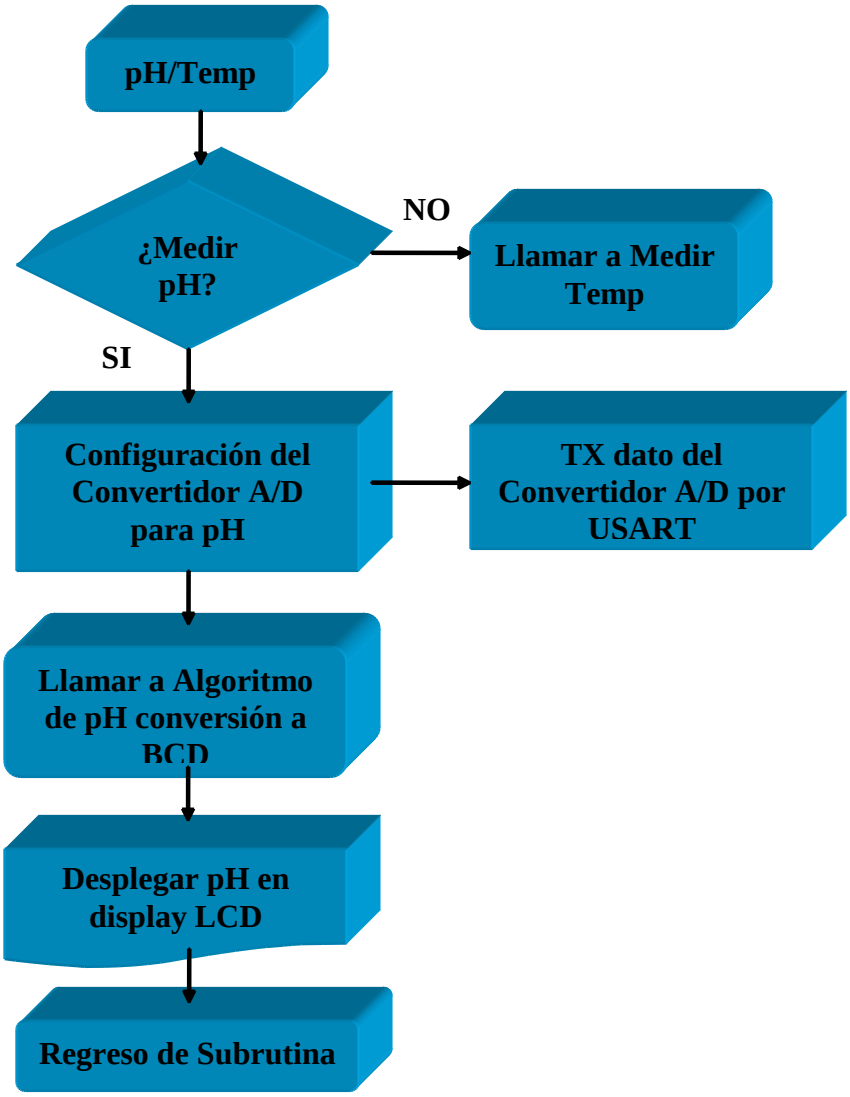

Figura 4-6 Subrutina pH

En las figuras 4-7 se muestran los diagramas a flujo para medir la Temperatura.

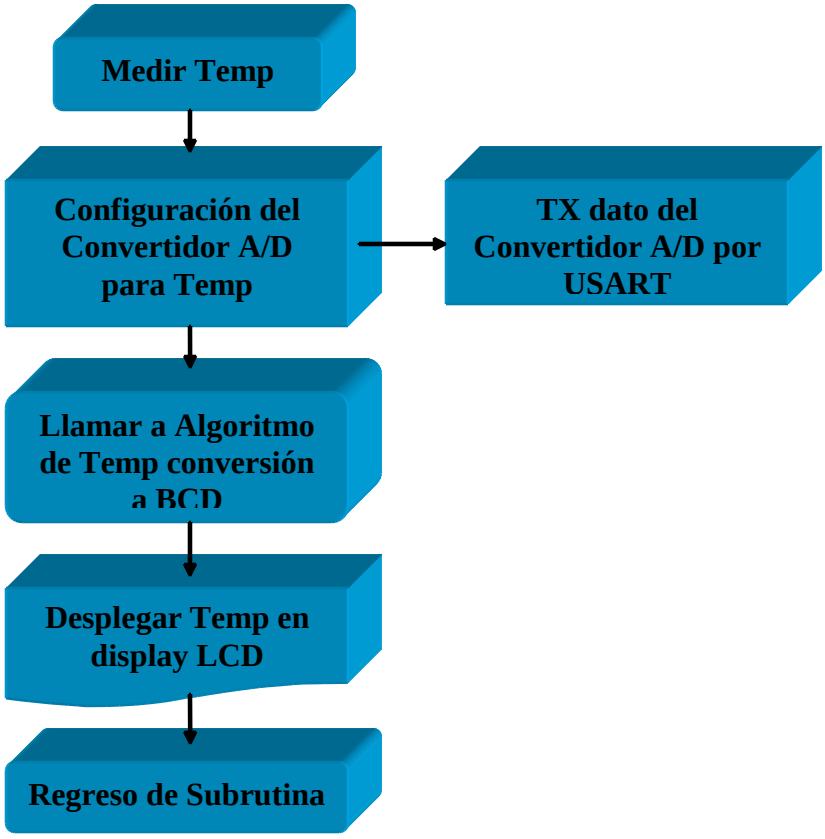

Figura 4-7 Subrutina Temperatura

4.3.3. ANALISIS Y DISEÑO DEL ALGORITMO

Primero se toman en cuenta varios conceptos:

- Funcionamiento del sensor de pH y Temperatura: LINEAL

Voltaje Maximo-Minimo de operación del sensor pH y Temperatura: 0.0-5.0v
Rango de la sonda pH: 0-10 pH
Rango del sensor de Temperatura: 0-100°C

- Funcionamiento del Convertidor A/D: LINEAL

Voltaje Maximo-Minimo de operación: 0.0-5.0v
Rango: 0-255, 8bits
Valor en volts por estado: 19.6 volts/estado

- Factor de conversión

Para obtener el factor de conversión es necesario dividir el valor máximo que puede medir el sensor ya sea en pH ó °C entre la cantidad de estados utilizados por el sensor que serán 255 igual que el Convertidor A/D

pH	Temperatura
10pH/255=0.039	**100°C/255=0.39**

Puesto que el Microcontrolador no puede operar números decimales debemos multiplicar a estos factores de conversión por algún numero que lo vuelva entero; para el caso del pH se elije al **1000=$3E8** y para la temperatura el **100=$64**.

pH	Temperatura
(0.039)(1000)=39=$27	**(0.39)(100)=39=$27**

- Algoritmo final para el Microcontrolador

Se multiplica el dato arrojado por el Convertidor A/D (ADR) por el factor de conversión ya sea de pH ó Temperatura.

pH	Temperatura
(ADR)($27)=ADR1000	**(ADR)($27)=ADR100**

A continuación es necesario eliminar el aumento de 1000 y 100 que fue otorgado al factor de conversión para volverse entero, para esto solo se divide entre 1000 y 100 en hexadecimal.

pH	Temperatura
ADR1000/$3E8=$pH	**ADR100/$64=$Temperatura**

Por último simplemente hay que dividir el valor del pH ó Temperatura obtenida entre 10 decimal ó $0A hexadecimal, esto se hace para convertir el valor de hexadecimal a decimal, el resultado de esta división nos da el digito de las unidades y el residuo de la división las centenas.

pH	Temperatura
$pH/$0A=pH	**$Temperatura/$0A=Temperatura °C**

Ejemplo: Si en la salida del sensor se mide un voltaje de 2.55v ¿Cuál será el valor de pH?

2.55v(255)/5v=130.05=$82
($82)($27)=$13CE
$13CE/$03E8=$5
$5/$0A=5pH

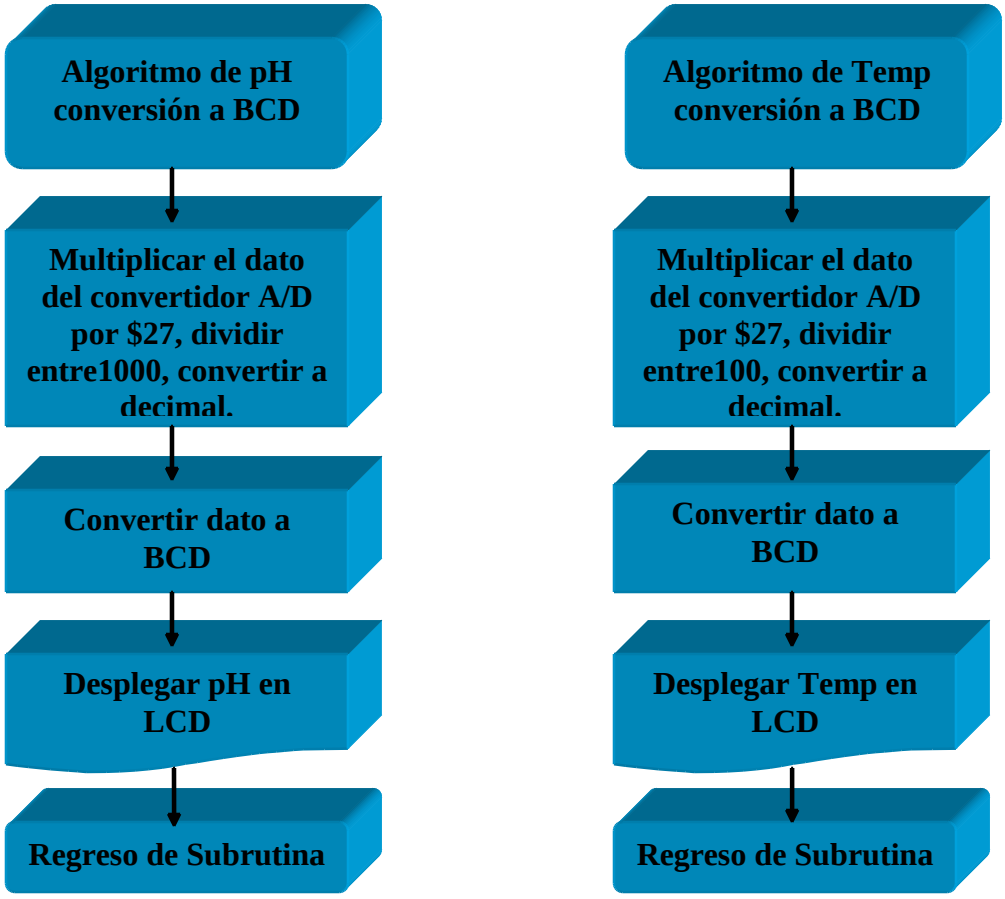

Figura 4-8 Subrutinas de algoritmo de pH y Temperatura

4.3.4. PROGRAMACION Y GRABADO DEL AVR

Todo este programa para el microcontrolador fue ensamblado con AVR Studio 4 y grabado en el microcontrolador en PonyProg, dado que estas herramientas son de fácil utilización.

AVR Studio 4 nos permite ensamblar, y aparte, cargar el programa en un simulador que trae incluido, para así poder comprobar el correcto funcionamiento de nuestro programa. Otra ventaja que tiene este software es que se puede descargar des de la página de ATMEL sin ningún costo.

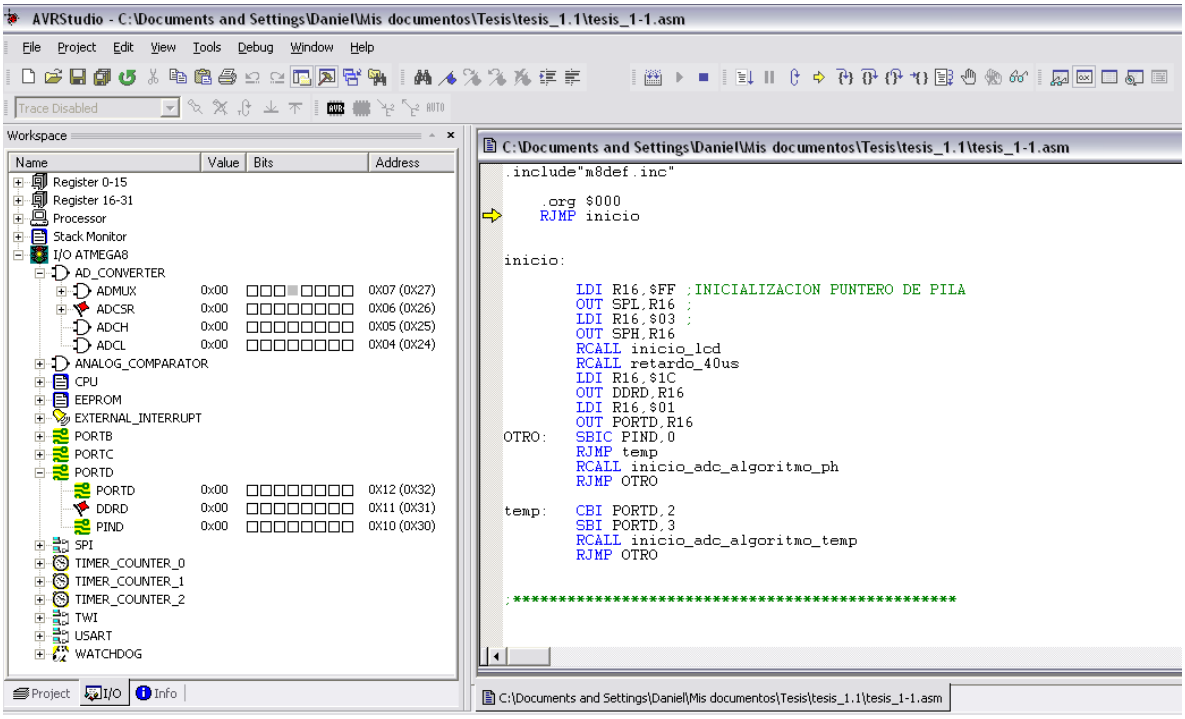

Figura 4-9 AVR Studio 4

Para poder utilizar PonyProg es necesario el programador de AVR's ya que este es solo el software. Este programador de AVR's fue hecho en un PCB en donde el diagrama a seguir fue el de la figura 4-11

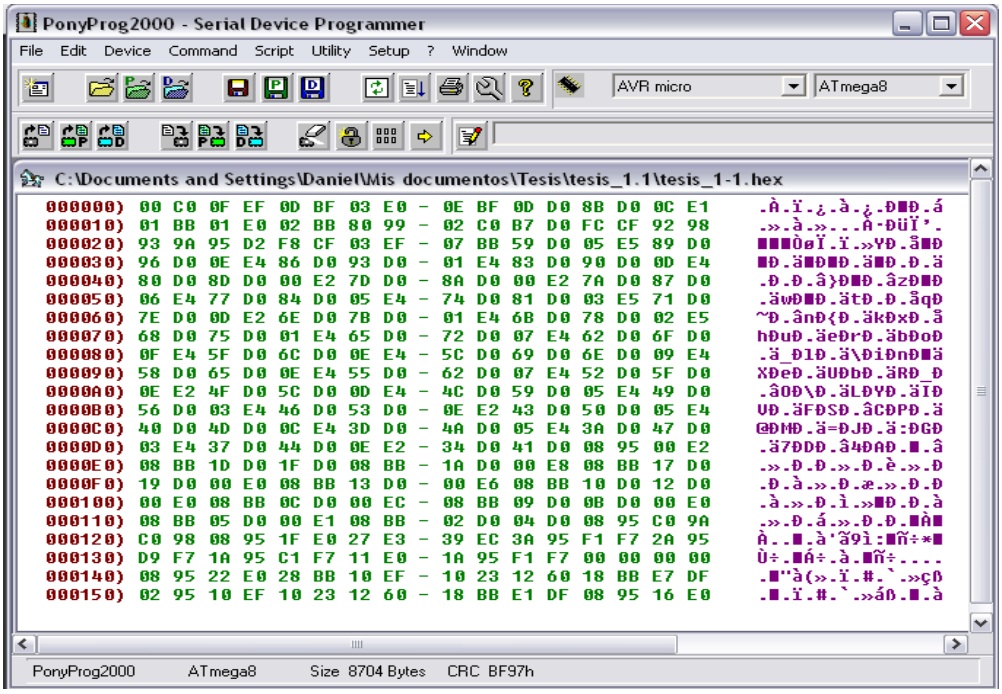

Figura 4-10 PonyProg

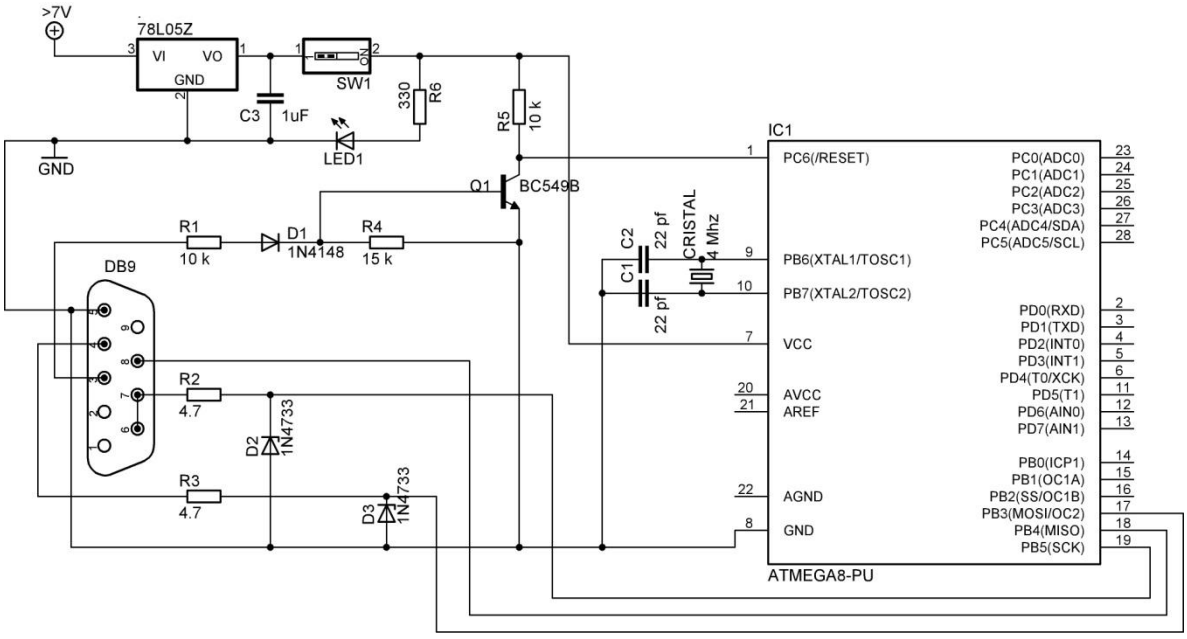

Figura 4-11 Diagrama esquemático del programador de AVR

Este diagrama esquemático fue llevado a otro programa para el diseño de las pistas, para la elaboración del circuito impreso, el cual es *Eagle 5.6.0 Professional* (figura 4-12).

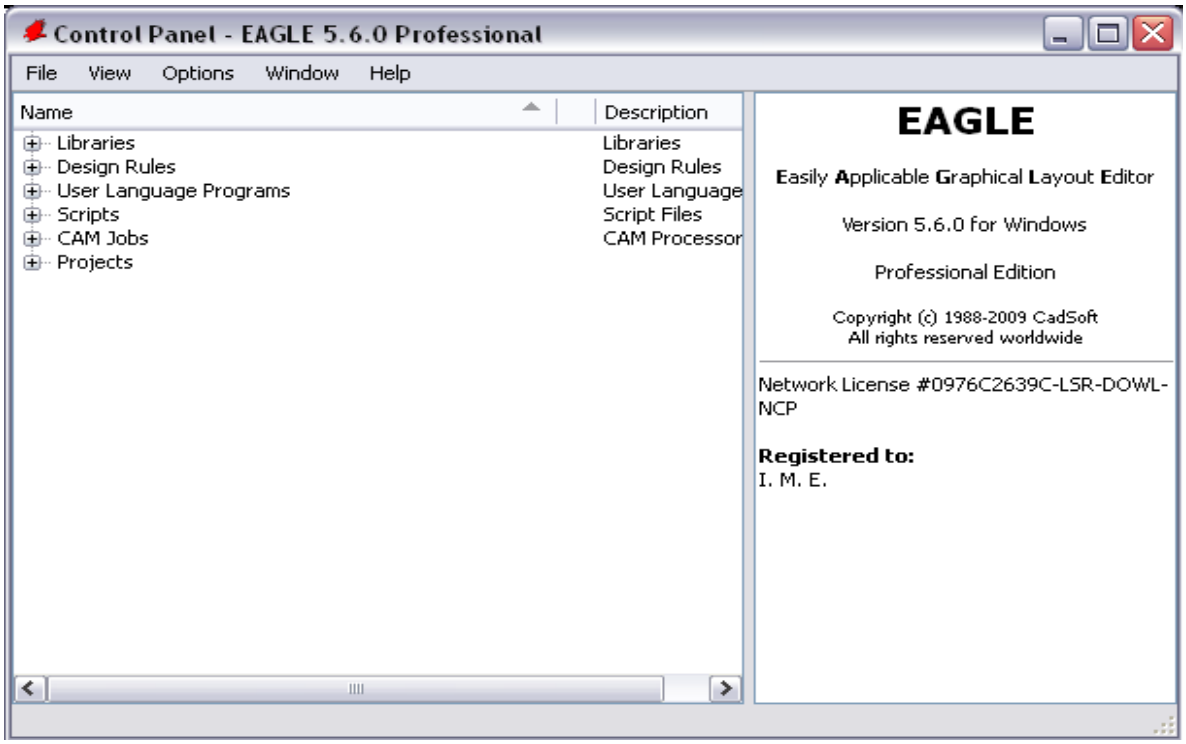

Figura 4-12 Panel de control de EAGLE

El resultado del diagrama esquemático con Eagle es el de la figura 4-13.

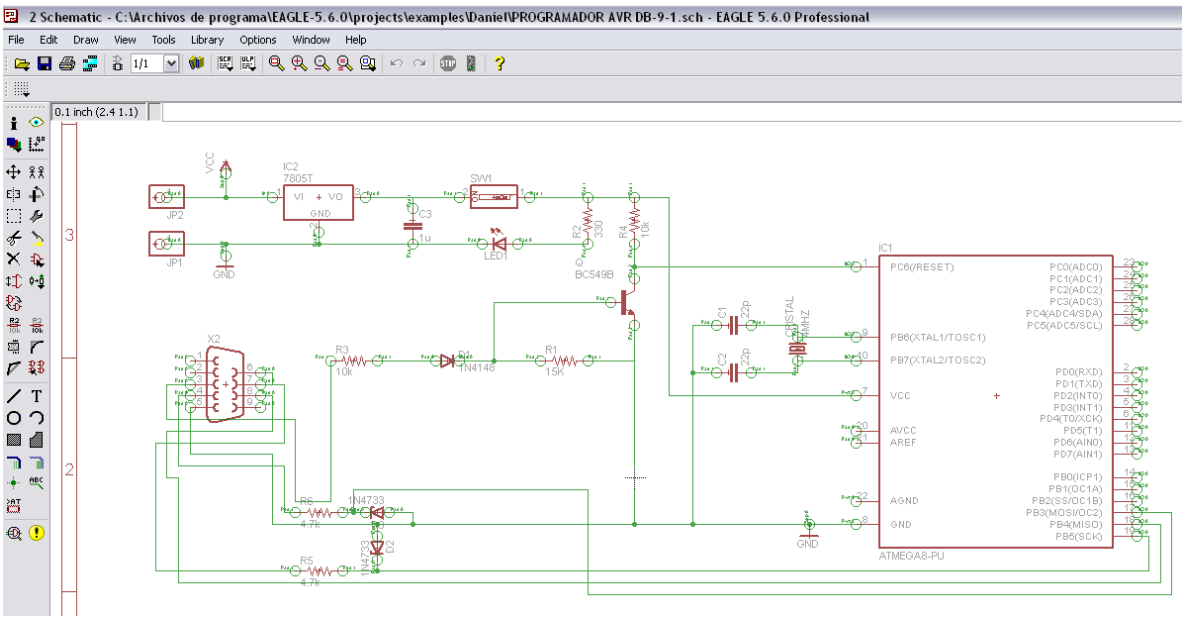

Figura 4-13 Diagrama esquemático del programador de AVR en EAGLE

Y por último, con la ayuda de Eagle, el diseño de las pistas para el circuito impreso fue el de la figura 4-14.

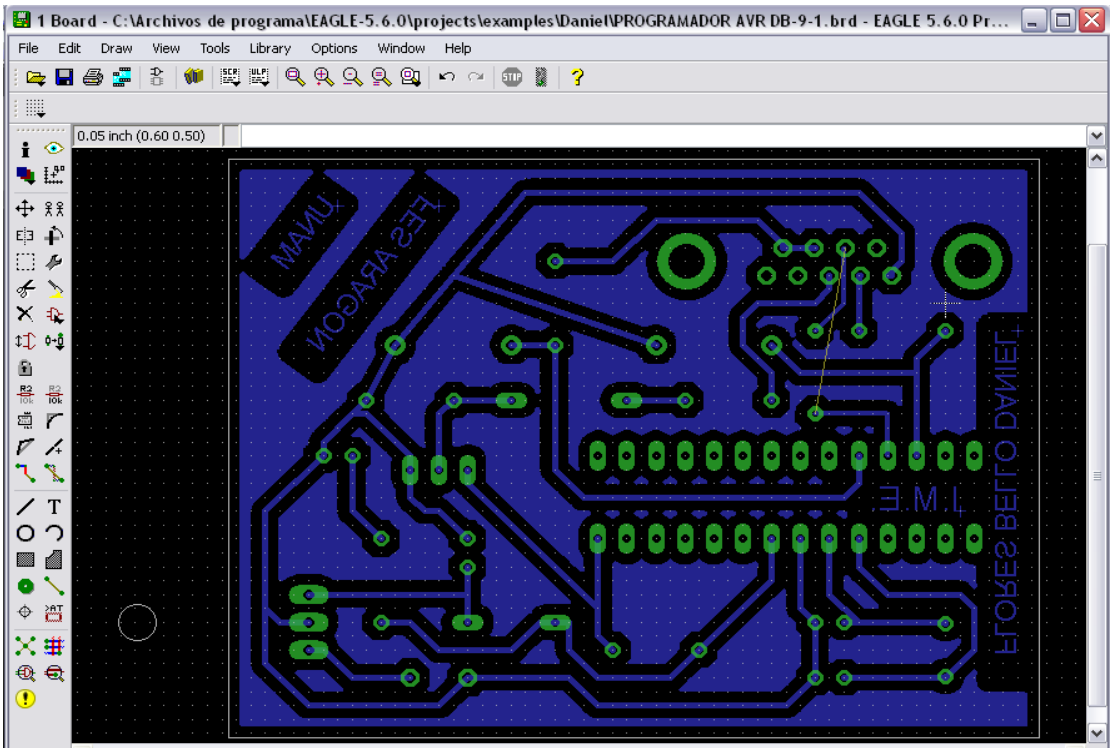

Figura 4-14 Diagrama del circuito impreso.

Y como finalización, de la placa de circuito impreso, de un grabador de AVR, se puede observar en la figura 4-15.

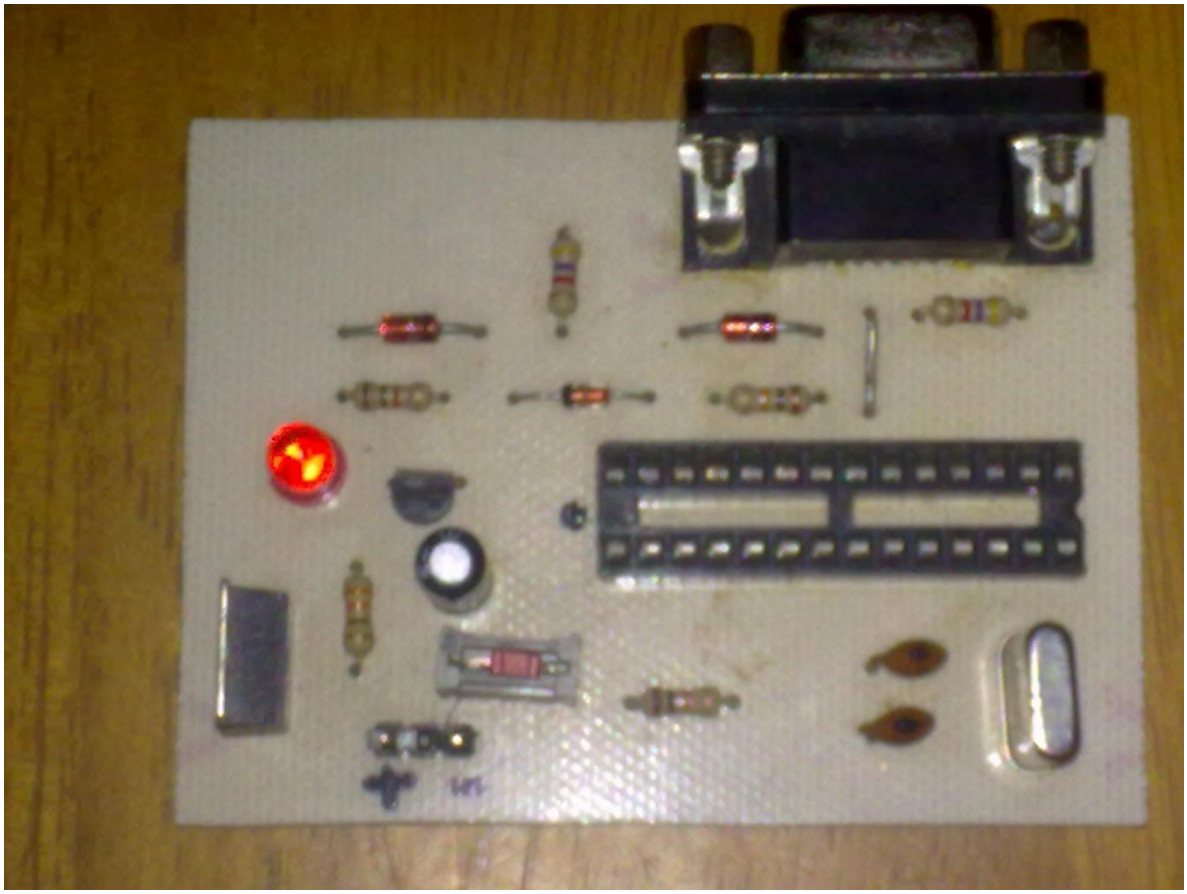

Figura 4-15 Programador de AVR

4.3.5 ELABORACION DEL MOULO DE MONITOREO PARA ACUARIO CON INTERFAZ USB

Habiendo ya seleccionado los sensores a utilizar, grabado el microcontrolador con el correspondiente programa, se procede a hacer el diagrama esquemático en Eagle, como lo muestra la figura 4-16

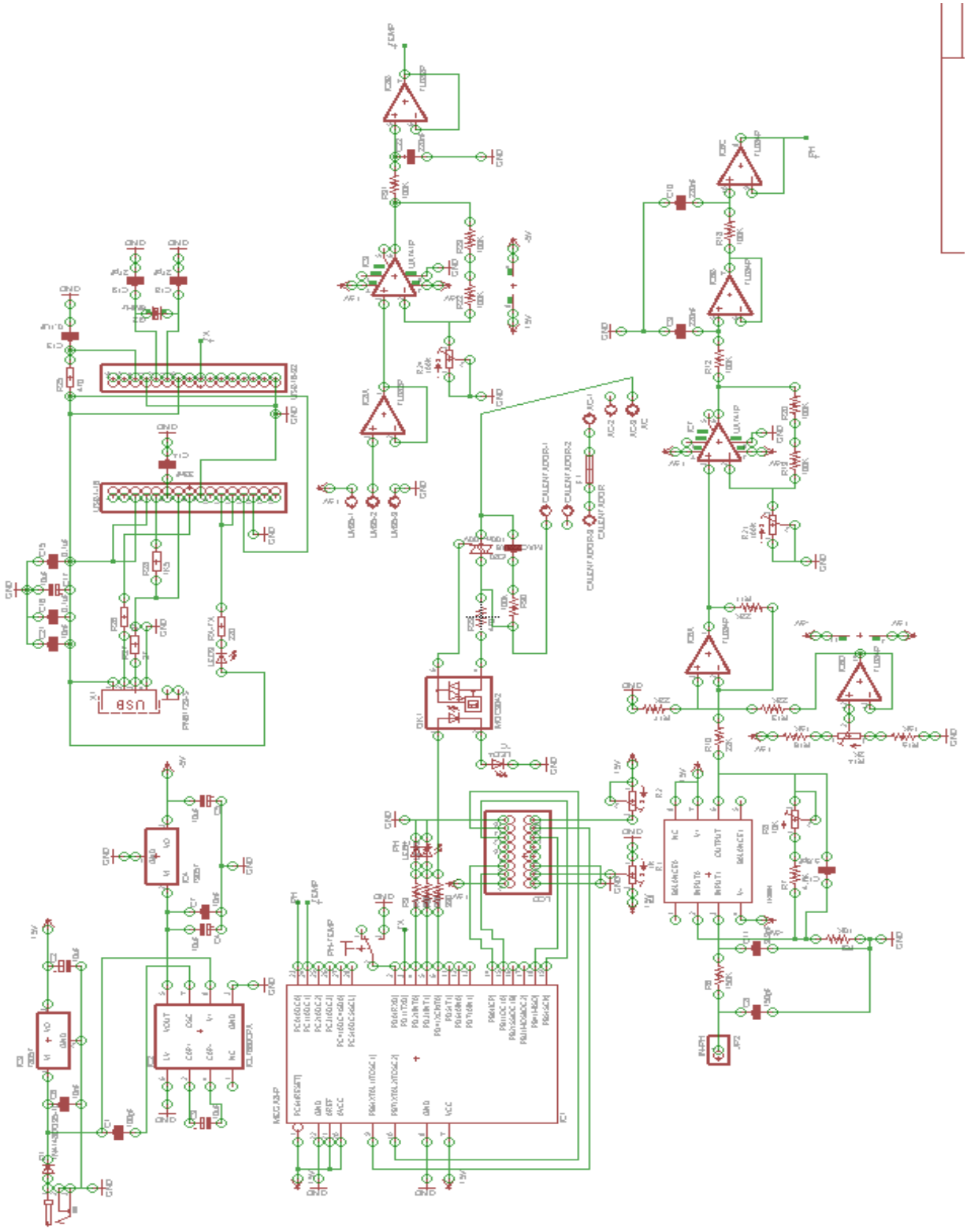

Figura 4-16 Esquemático del modulo de monitoreo con interfaz USB

Para los circuitos de acondicionamiento de las señales de la sonda de pH y sensor de temperatura ver Anexo 2, Pág. 158.

Posterior a esto, se diseñaron las pistas para realizar el circuito impreso (figura 4-17), con el mismo software.

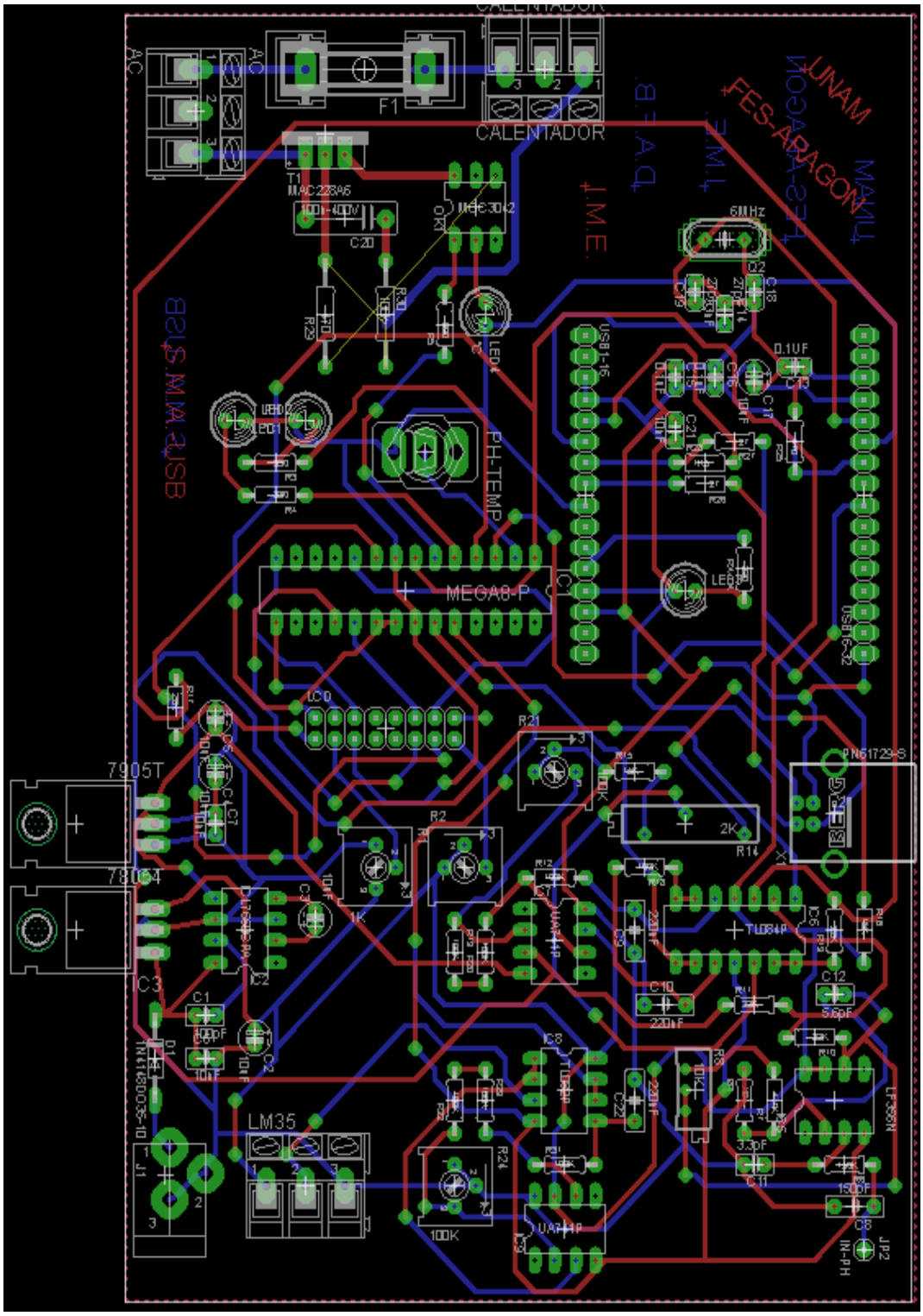

Figura 4-17 Diseño de PCB y componentes del circuito

Y por último, se observa la placa de circuito impreso, correspondiente al modulo de monitoreo con interfaz USB ya terminada en la figura 4-18.

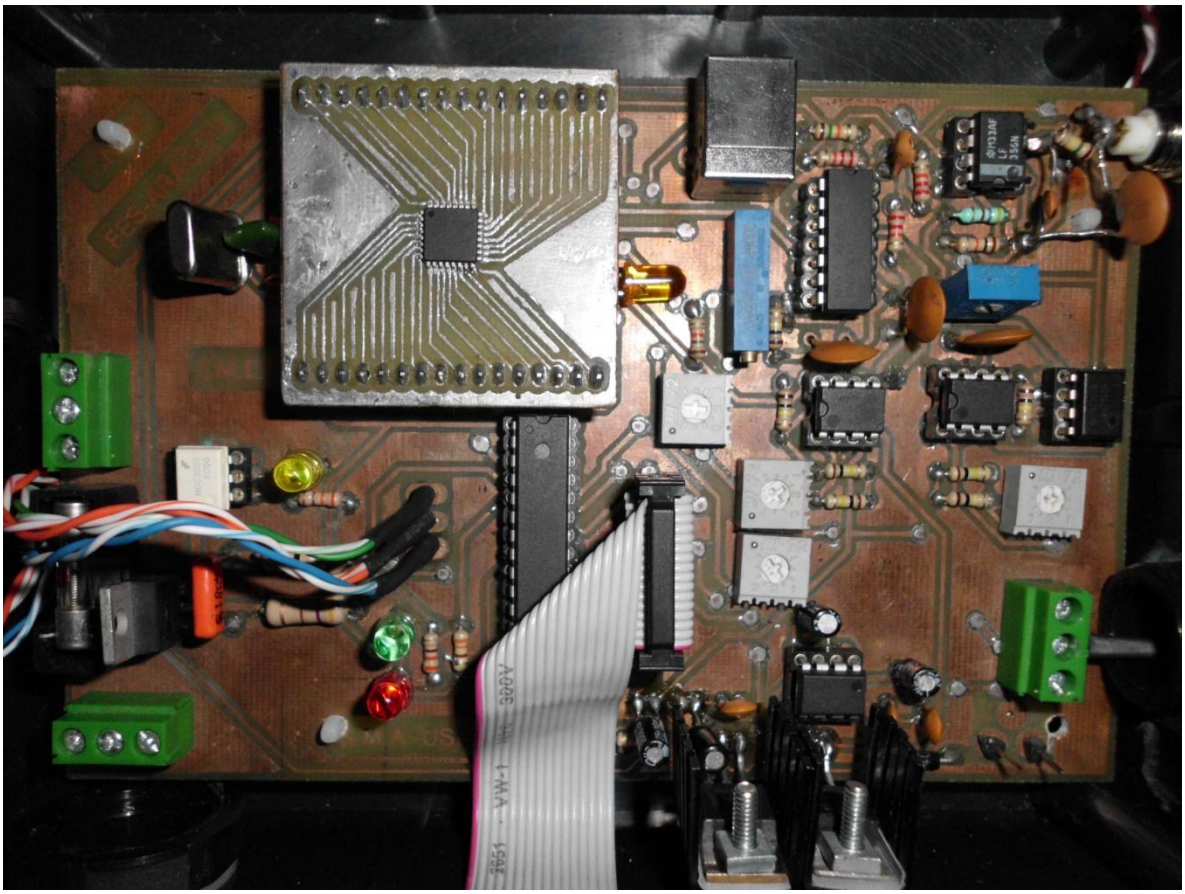

Figura 4-18 Modulo de monitoreo de acuario

4.3.6. MONITOREO DE DATOS EN LA PC

El monitoreo en la computadora se dará por medio de los datos provenientes de la interfaz USB, y gracias al programa ensamblado en LabVIEW. Este programa se presenta en la figura 4-19.

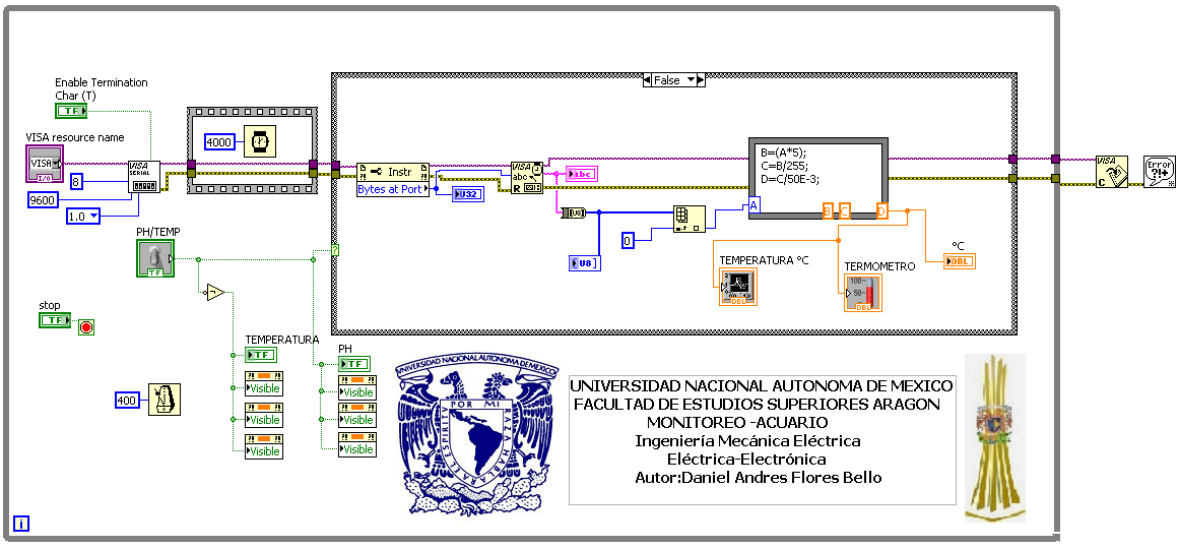

Figura 4-19 Programa del sistema de monitoreo de acuario

La forma en la cual se va a visualizar este programa en la PC, se observa en la figura 4-20, de este modo se estarán monitoreando, en tiempo real, las mediciones obtenidas por el microcontrolador.

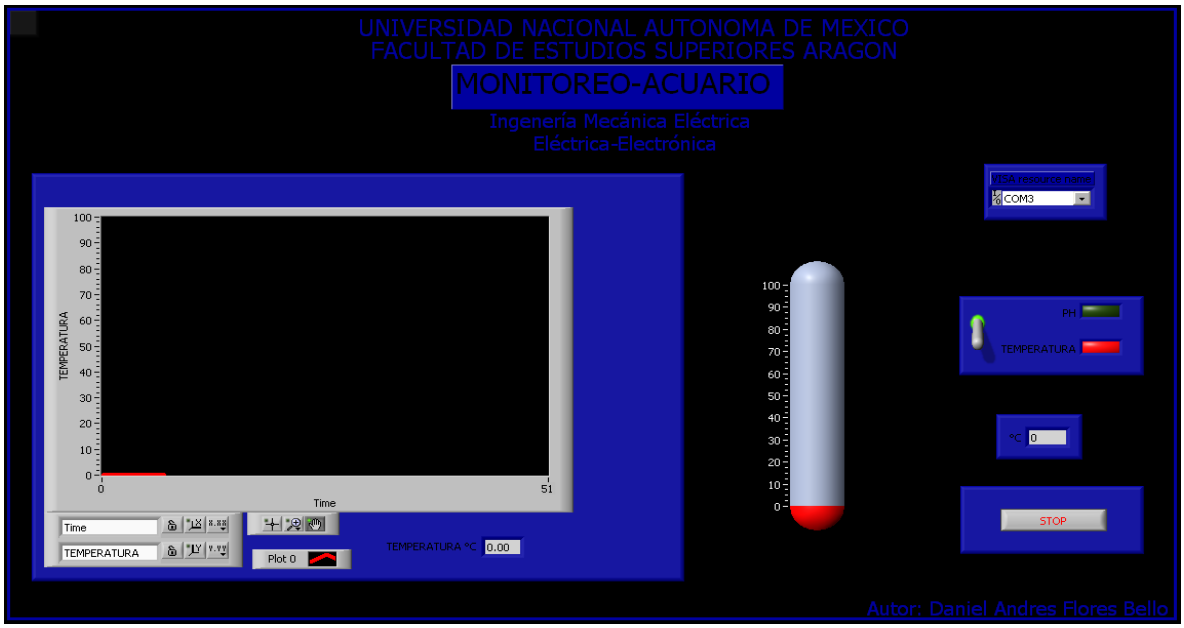

Figura 4-20 Panel frontal del sistema de monitoreo inalámbrico

CONCLUSIONES

Cuando se empezó hacer el diseño de este sistema, se tuvo que tomar en cuenta la selección de los componentes a utilizar, y que se adecuaran a él. En primera instancia tenemos a la selección de los transductores; en donde para sensar la temperatura se selecciono al LM35DZ, ya que es muy económico y pertenece a una serie de sensores de temperatura, que son circuitos integrados de precisión, por lo tanto nos proporciona la temperatura de forma lineal, además de que se comporta de forma muy normal. Ahora, con el sensor de pH, se selecciono una sonda estándar para medir el pH ya que estas sondas son fáciles de manejar y darles mantenimiento, también este tipo de sondas pueden emplearse en otros sistemas para la medición de pH que se encuentran en el mercado.

Al seleccionar el microcontrolador, se tomo en cuenta la cantidad de entradas y salidas que iba a tener el sistema así como también se toma en cuenta que es el que se domina mas, tanto en su manejo como en el ambiente de programación para el mismo, por ello se escogió el microcontrolador AVR Atmega8 de la compañía de Atmel.

Para controlar el LCD se tuvo que hacer en un principio con un bus de datos de 8 bits, debido a esto, se estaban utilizando 10 bits del microcontrolador para este objetivo, en realidad eran muchos, así que se enviaron los datos al LCD multiplexados, con esto se disminuyeron los bits para controlar el LCD con el microcontrolador a solamente 6.

Uno de los problemas que tomo algo de tiempo solucionar, fueron las operaciones internas que tenía que realizar el AVR para poder realizar el algoritmo y desplegar en el LCD el dato de las mediciones en forma decimal. A esto, se le dio solución con unas subrutinas de multiplicación y división para hacer operaciones de 8 y 16 bits, y para poder visualizarse en el LCD de forma decimal, estos datos se tuvieron que convertir a BCD (Decimal Codificado en Binario, en español) para posteriormente ser desplegados en el LCD, también este problema se resolvió con otra subrutina para realizar dicha conversión. Las subrutinas tanto de multiplicación, división y la de conversión a BCD se obtuvieron mediante las notas de aplicaciones para el AVR en la página de Atmel.

Otro detalle que llevo un poco de tiempo resolver fue la fuente de alimentación para el modulo, ya que se tiene que usar una fuente simétrica de +5v -5v, ya que para el acondicionamiento de las señales de los sensores se ocupan OP-AMP, el cual fue resuelto con el CI C7660 el cual convierte nuestro voltaje de entrada a un voltaje ± (+5v á ±5v). También se tuvo que diseñar una base para el modulo de conversión de RS232 a USB ya que el CI FT232BL que se encarga de esta conversión es de montaje superficial, la cual se realizo con el software EAGLE.

Debido a que hay diferentes plataformas de programación, se opto por LabVIEW, ya que maneja un ambiente gráfico tanto para programar como para la visualización del programa en ejecución y también por ser el software que se domina más.

En lo que corresponde a la elaboración de el PCB del modulo, se realizo primero por software en EAGLE y posteriormente se realizo de manera manual utilizando una placa de cobre, lija, cloruro férrico plancha; estos son algunos de los elementos y materiales más importantes para el desarrollo de un PCB.

Los resultados obtenidos con el sistema de monitoreo son los siguientes: cuando se enciende el modulo el display LCD despliega un mensaje, después detecta la posición del switch para medir ya sea el pH ó Temperatura y esta acción es indicada con unos Leds, verde para el pH y rojo para la temperatura imagen C-1.

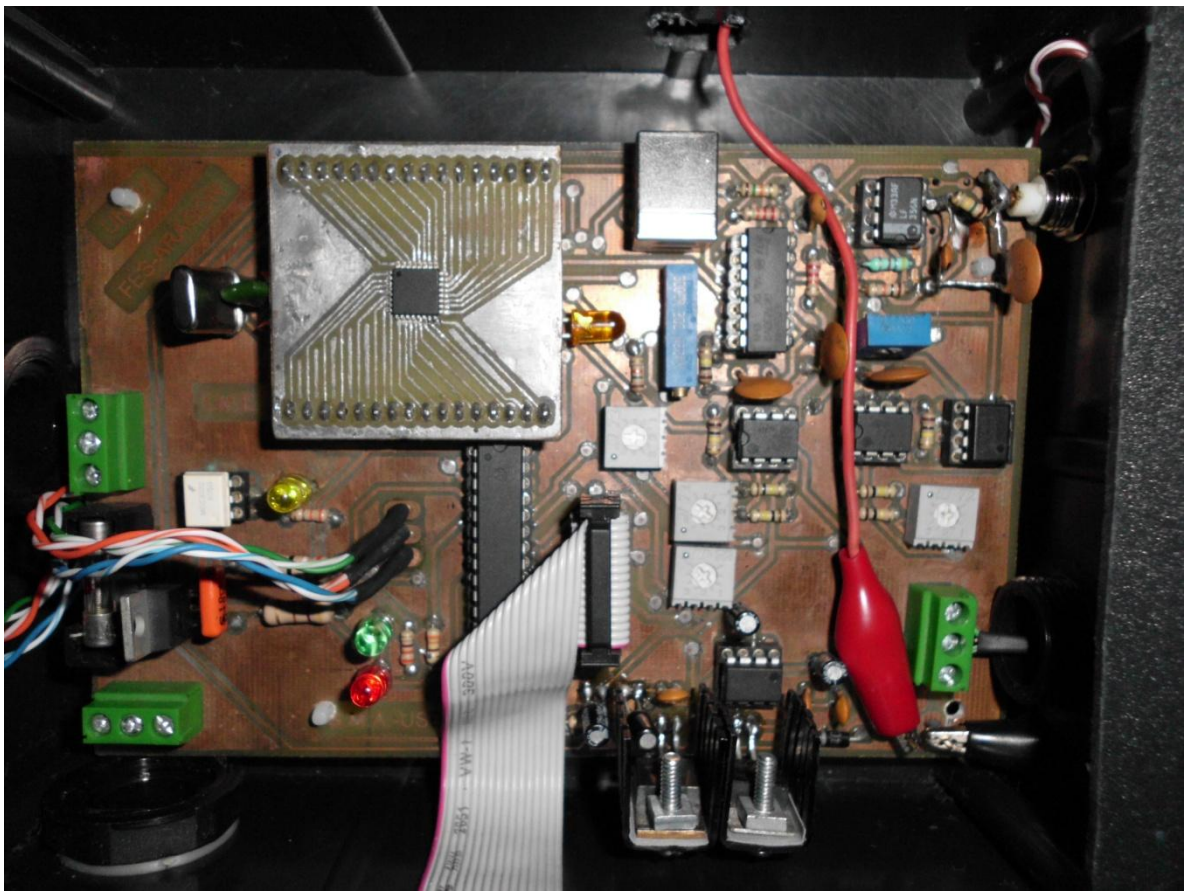

Figura C-1 Modulo de monitoreo de acuario

En la imagen C-2 se observa la ejecución del programa en LabVIEW para el monitoreo del acuario.

Figura C-2 Ejecución de LabVIEW y comparación con el modulo LCD

También se muestra como se realiza una visualización en la computadora donde en el programa se puede seleccionar entre los puertos COM que se detecten y seleccionar el que nos convenga, de igual forma se pueden visualizar el dato del parámetro a medir de forma decimal y en forma grafica.

Como consecuencia, este trabajo debe servir como modelo de aplicación de la tecnología en el proceso de crianza de peses y acondicionamiento para un acuario el cual requiere de el constante monitoreo no solo de estos parámetros si no de otros mas, el presente diseño queda evidentemente limitado para medir los parámetros de temperatura y pH aunque también al final del diseño se acoplo un pequeño sistema de control para el calentador del acuario el cual no fue diseñado con toda precisión

GLOSARIO

Acuariofilia.- Es la afición a la cría de peces y otros organismos acuáticos en acuario, bajo condiciones controladas.

Acuariología.- *Es la rama del conocimiento científico derivada de la acuariofilia que estudia todos los aspectos de creación y mantenimiento de especies y ecosistemas acuáticos de forma artificial y controlada, tanto en acuarios como en cualquier otro tipo de instalación. Es la acuariofilia llevada a un extremo científico bajo criterios reproducibles y controlables, y más allá de acuarios domésticos.*

Acuicultura.- Es el conjunto de actividades, técnicas y conocimientos de cultivo de especies acuáticas vegetales y animales.

ADC. La conversión analógica-digital (en español CAD) consiste en la transcripción de señales analógicas en señales digitales, con el propósito de facilitar su procesamiento (codificación, compresión, etc.) y hacer la señal resultante (la digital) más inmune al ruido y otras interferencias a las que son más sensibles las señales analógicas.

ALU. En computación, la Unidad Lógica Aritmética (ULA), o Arithmetic Logic Unit (ALU), es un circuito digital que calcula operaciones aritméticas (como suma, resta, multiplicación, etc.) y operaciones lógicas (como igual a, menor que, mayor que, etc.), entre dos números.
Muchos tipos de circuitos electrónicos necesitan realizar algún tipo de operación aritmética, así que incluso el circuito dentro de un reloj digital tendrá una ALU minúscula que se mantiene sumando 1 al tiempo actual, y se mantiene comprobando si debe activar el pitido del temporizador, etc.

Ampere. El amperio o ampere (símbolo A), es la unidad de intensidad de corriente eléctrica. Forma parte de las unidades básicas en el Sistema Internacional de Unidades y fue nombrado en honor de André-Marie Ampére. El amperio es la intensidad de una corriente constante que manteniéndose en dos conductores paralelos, rectilíneos, de longitud infinita, de sección circular despreciable y situados a una distancia de un metro uno de otro en el vacío, produciría una fuerza igual a 2×10-7 newton por metro de longitud.

Análogo. Es un tipo de señal generada por algún tipo de fenómeno electromagnético y que es representable por una función matemática continúa en la que es variable su amplitud y periodo (representando un dato de información) en función del tiempo. Algunas magnitudes físicas comúnmente portadoras de una señal de este tipo son eléctricas como la tensión y la potencia, pero también pueden ser hidráulicas como la presión, térmicas como la temperatura, mecánicas, etc. La magnitud también puede ser cualquier objeto medible como los beneficios o pérdidas de un negocio.

Asíncrono. Hace referencia al suceso que no tiene lugar en total correspondencia temporal con otro suceso. Por ejemplo, un motor asíncrono, que es a aquel cuya velocidad de rotación no corresponde con la frecuencia de corriente alterna que lo hace funcionar.

Baudio. En inglés *baud* es una unidad de medida, usada en telecomunicaciones, que representa el número de símbolos transmitidos por segundo en una red análoga.

Bit. Es el acrónimo de *Binary digit*. (dígito binario). Un bit es un dígito del sistema de numeración binario. Mientras que en el sistema de numeración decimal se usan diez dígitos, en el binario se usan sólo dos dígitos, el 0 y el 1. Un bit o dígito binario puede representar uno de esos dos valores, 0 ó 1.

CAD. Un conversor (o convertidor) analógico-digital (CAD), (ADC) es un dispositivo electrónico capaz de convertir una entrada analógica de voltaje en un valor binario, Se utiliza en equipos electrónicos como computadoras, grabadores de sonido y de vídeo, y equipos de telecomunicaciones. La señal analógica, que varía de forma continua en el tiempo, se conecta a la entrada del dispositivo y se somete a un muestreo a una velocidad fija, obteniéndose así una señal digital a la salida del mismo.

CPU. La unidad central de procesamiento o CPU (por el acrónimo en inglés de *central processing unit*), o simplemente el procesador o microprocesador, es el componente en una computadora, que interpreta las instrucciones y procesa los datos contenidos en los programas de la computadora.
DB – 25. *El conector DB25 (originalmente DE-25) es un conector analógico de 25 clavijas de la familia de conectores D-Subminiature (D-Sub o Sub-D).*

DB – 9. *El conector DB9 (originalmente DE-9) es un conector analógico de 9 clavijas de la familia de conectores D-Subminiature (D-Sub o Sub-D). El conector DB9 se utiliza principalmente para conexiones en serie, ya que permite una transmisión asíncrona de datos según lo establecido en la norma RS-232 (RS-232C).*

DLL. *Una biblioteca de enlace dinámico o más comúnmente llamada DLL (sigla en inglés de dinamic-link library) es el término con el que se refiere a los archivos con código ejecutable que se cargan bajo demanda de un programa por parte del sistema operativo. Esta denominación es exclusiva a los sistemas operativos Windows siendo ".dll" la extensión con la que se identifican estos ficheros, aunque el concepto existe en prácticamente todos los sistemas operativos modernos*

Driver. *Es un controlador de dispositivo, llamado normalmente controlador (en inglés, device driver) es un programa informático que permite al sistema operativo interactuar con un periférico, haciendo una abstracción del hardware y proporcionando una interfaz –posiblemente estandarizada- para*

usarlo. Se puede esquematizar como un manual de instrucciones que le indica al sistema operativo, cómo debe controlar y comunicarse con un dispositivo en particular. Por tanto, es una pieza esencial, sin la cual no se podría usar el hardware.

DSP. *Un procesador digital de señales o DSP (sigla en inglés de digital signal processor) es un sistema basado en un procesador o microprocesador que posee un juego de instrucciones, un hardware y un software optimizados para aplicaciones que requieran operaciones numéricas a muy alta velocidad.*

EEPROM. *También conocida como E²PROM, que son las siglas de Electrically-Erasable Programmable Read-Only Memory (ROM programable y borrable eléctricamente). Es un tipo de memoria ROM que puede ser programado, borrado y reprogramado eléctricamente, a diferencia de la EPROM que ha de borrarse mediante un aparato que emite rayos ultravioletas. Son memorias no volátiles.*

Electrodo. *Es una placa de membrana rugosa de metal un conductor utilizado para hacer contacto con una parte no metálica de un circuito, por ejemplo un semiconductor, un electrolito, el vacío (en una válvula termoiónica), un gas (en una lámpara de neón), etc. L palabra fue acuñada por el científico Michael Faraday y procede de las voces griegas elektron, que significa ámbar y de la que proviene la palabra electricidad; y hodos, que significa camino*

EPROM. *Son las siglas de Erasable Programmable Read-Only Memory (ROM programable borrable). Es un tipo de chip de memoria ROM no volátil inventado por el ingeniero Dov Frohman. Está formada por celdas de FAMOS (Floating Gate Avalanche-Injection Metal-Oxide Semiconductor) o "transistores de puerta flotante", cada uno de los cuales viene de fábrica sin carga, por lo que son leídos como 0 (por eso, una EPROM sin grabar se lee como 00 en todas sus celdas). Se programan mediante un dispositivo electrónico que proporciona voltajes superiores a los normalmente utilizados en los circuitos electrónicos. Las celdas que reciben carga se leen entonces como un 1.*

Hardware. *Corresponde a todas las partes físicas y tangibles de una computadora: sus componentes eléctricos, electrónicos, electromecánicos y mecánicos; sus cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado; contrariamente al soporte lógico e intangible que es llamado software.*

Hub.- *Dispositivo para compartir una red de datos o de puertos USB de un ordenador.*

Hz. *El hertzio, hercio o hertz (Símbolo Hz), es la unidad de frecuencia del*

Sistema Internacional de Unidades.

Kbps. *Un kilobit por segundo es una unidad de medida que se usa en telecomunicaciones e informática para calcular la velocidad de transferencia de información a través de una red. Su abreviatura y forma más corriente es kbps.*

LabVIEW. *Es una herramienta gráfica para pruebas, control y diseño mediante la programación. El lenguaje que usa se llama lenguaje G, donde la G simboliza que es lenguaje Gráfico.*

LCD. *Una pantalla de cristal líquido o LCD (acrónimo del inglés Liquid Crystal Display) es una pantalla delgada y plana formada por un número de píxeles en color o monocromos colocados delante de una fuente de luz o reflectora. A menudo se utiliza en dispositivos electrónicos de pilas, ya que utiliza cantidades muy pequeñas de energía eléctrica.*

Optoacoplador. *También llamado optoaislador o aislador acoplado ópticamente, es un dispositivo de emisión y recepción de luz que funciona como un interruptor excitado mediante la luz.*

PCB. *En electrónica, un circuito impreso o PCB (del inglés printed circuit board), es un medio para sostener mecánicamente y conectar eléctricamente componentes electrónicos, a través de rutas o pistas de material conductor, grabados en hojas de cobre laminadas sobre un sustrato no conductor.*

Plug & play. *Conocida también por su abreviatura PnP, es la tecnología que permite a un dispositivo informático ser conectado a una computadora sin tener que configurar (mediante jumpers o software específico (no controladores) proporcionado por el fabricante) ni proporcionar parámetros a sus controladores. Para que sea posible, el sistema operativo con el que funciona la computadora debe tener soporte para dicho dispositivo.*

pH.- *es una medida de la acidez o alcalinidad de una solución. El pH indica la concentración de iones hidronio [H_3O^+] presentes en determinadas sustancias*

RAM. *La memoria de acceso aleatorio (en inglés: random-access memory cuyo acrónimo es RAM) es la memoria desde donde el procesador recibe las instrucciones y guarda los resultados. Es el área de trabajo para la mayor parte del software de una computadora.*

ROM. *La memoria de sólo lectura (normalmente conocida por su acrónimo, Read Only Memory) es una clase de medio de almacenamiento utilizado en las computadoras y otros dispositivos electrónicos.*

RS – 232. *Es una interfaz que designa una norma para el intercambio serie*

de datos binarios

RTD. *Son sensores de temperatura resistivos. En ellos se aprovecha el efecto que tiene la temperatura en la conducción de los electrones para que, ante un aumento de temperatura, haya un aumento de la resistencia eléctrica que presentan.*

Regulador. *Es un dispositivo electrónico creado para obtener un valor de salida deseado en base al nivel de entrada, ya sea mecánico o eléctrico.*

Sensor. *Es un dispositivo capaz de medir magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas.*

Software. *Se refiere al equipamiento lógico o soporte lógico de una computadora digital, y comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de tareas específicas; en contraposición a los componentes físicos del sistema, llamados hardware.*

SRAM. *Static Random Access Memory (SRAM), o Memoria Estática de Acceso Aleatorio es un tipo de memoria basada en semiconductores que, a diferencia de la memoria DRAM, es capaz de mantener los datos (mientras esté alimentada) sin necesidad de circuito de refresco (no se descargan). Sin embargo, sí son memorias volátiles.*

Termistor. *Es un semiconductor que varía el valor de su resistencia eléctrica en función de la temperatura, su nombre proviene de Thermally sensitive resistor (Resistor sensible a la temperatura). Existen dos clases de termistores: NTC y PTC.*

Termopar. *Es un dispositivo formado por la unión de dos metales distintos que produce un voltaje (efecto Seebeck), que es función de la diferencia de temperatura entre uno de los extremos denominado "punto caliente" o unión caliente o de medida y el otro denominado "punto frío" o unión fría o de referencia.*

Termostato. *Es el componente de un sistema de control simple que abre o cierra un circuito eléctrico en función de la temperatura.*

Transductor. *La transducción de señal es el conjunto de procesos o etapas que ocurren de forma concatenada por el que una célula convierte una determinada señal o estímulo exterior, en otra señal o respuesta específica.*

TTL. *Son las siglas en inglés de transistor-transistor logic, es decir, "lógica transistor a transistor". Es una familia lógica o lo que es lo mismo, una tecnología de construcción de circuitos electrónicos digitales. En los componentes fabricados con tecnología TTL los elementos de entrada y*

salida del dispositivo son transistores bipolares.

USART. *Son las siglas de "Universal Synchronous Asynchronous Receiver-Transmitter" (en español, "Transmisor-Receptor Síncrono Asíncrono Universal"). Este controla los puertos y dispositivos serie. Existe un dispositivo electrónico encargado de generar la USART en cada puerto serie.*

USB.- *Son las siglas de Universal Serial Bus es un puerto que sirve para conectar periféricos a un ordenador o PC.*

VISA. *Es una norma de instrumentación.*

Volt. *El voltio o volt (símbolo V), es la unidad derivada del SI para el potencial eléctrico, fuerza electromotriz y el voltaje. Recibe su nombre en honor de Alessandro Volta.*

Bibliografía

Boylestad, Robert y Nashelsky, Louis. (2003). Electrónica: teoría de circuitos y dispositivos electrónicos. México: Pearson Educación

Arrignon, Jacques. (1984). Ecología y Piscicultura de aguas dulces. California: Mundi-Prensa

Métodos y artes de pesca y manejo de organismos acuícolas: agua dulce. Colegio Nacional de Educación Profesional Técnica

Atmel Corporation (2004). 8-bit AVRwith 8K Bytes In-System Programmable Flash, ATmega8 ATmega8L. Extraído el 25 de Mayo de 2008 desde http://www2.atmel.com/

National Instruments (2003). La instrumentación Virtual. Extraído el 20 de Mayo de 2011 desde http://www.ni.com/es/

Future Technology Devices International Chip (2005). FT232BL USB UART (USB - Serial) I.C. Extraído el 10 de Septiembre de 2010 desde http://www.ftdichip.com/

José Manuel García (s.f.). Medidor de pH para acuarios. Extraído el 28 de Diciembre de 2010 desde http://213.97.130.124/phm/phmeter.htm

ANEXO 1 CODIGO DE PROGRAMA DEL MICROCONTROLADOR

```
;*********************************************************************
****
;*          UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
*
;*            FACULTAD DE ESTUDIOS SUPERIORES ARAGON
*
;*                        ING.MECANICA ELECTRICA
*
;*TESIS: SISTEMA DE MONITOREO PARA ACUARIO CON INTERFAZ USB
*
;*                AUTOR: DANIEL ANDRES FLORES BELLO
*
;*                ASESOR: ING. ARTURO OCAMPO ALVAREZ
*
;*********************************************************************
****
.include"m8def.inc"
        .org $000
        RJMP inicio
inicio
                LDI R16,$FF  ;INICIALIZACION PUNTERO DE PILA
                OUT SPL,R16          ;
                LDI R16,$03 ;
                OUT SPH,R16
                RCALL inicio_lcd
                RCALL retardo_40us
                LDI R16,$1C
                OUT DDRD,R16
                LDI R16,$01
                OUT PORTD,R16
OTRO:           SBIC PIND,0
                RJMP temp
                RCALL inicio_adc_algoritmo_ph
                RJMP OTRO
```

```
temp:  CBI PORTD,2
               SBI PORTD,3
               RCALL inicio_adc_algoritmo_temp
               RJMP OTRO



;***********************************************
inicio_lcd:
;*********************************************************************
*******
               LDI R16,$F3  ;PUERTO B COMO SALIDA
               OUT DDRB,R16              ; PB7 PB6 PB5 PB4 PB3 PB2 PB1 PB0
;PUERTOS MCU
                                         ; DB7 DB6 DB5 DB4  X   X  RS
E  ;DISPLAY LCD
                                         ; R/W=GND
;*********************************************************************
*******
;*********************************************************************
*******               RCALL init_lcd        ;INICIALIZACION DEL LCD
;*********************************************************************
*******
;*********************************************************************
*******
;DESPLIEGA EL MENSAJE UNAM  FES-ARAGON
               LDI R16,'U'
               RCALL dato_lcd
               RCALL retardo_10us
               LDI R16,'N'
               RCALL dato_lcd
               rcall retardo_10us
               LDI R16,'A'
               RCALL dato_lcd
               RCALL retardo_10us
```

```
LDI R16,'M'
RCALL dato_lcd
RCALL retardo_10us
LDI R16,' '
RCALL dato_lcd
RCALL retardo_10us
LDI R16,' '
RCALL dato_lcd
RCALL retardo_10us
LDI R16,'F'
RCALL dato_lcd
RCALL retardo_10us
LDI R16,'E'
RCALL dato_lcd
RCALL retardo_10us
LDI R16,'S'
RCALL dato_lcd
RCALL retardo_10us
LDI R16,'-'
RCALL dato_lcd
RCALL retardo_10us
LDI R16,'A'
RCALL dato_lcd
RCALL retardo_10us
LDI R16,'R'
RCALL dato_lcd
RCALL retardo_10us
LDI R16,'A'
RCALL dato_lcd
RCALL retardo_10us
LDI R16,'G'
RCALL dato_lcd
RCALL retardo_10us
LDI R16,'O'
RCALL dato_lcd
RCALL retardo_10us
```

```
        LDI R16,'N'
        RCALL dato_lcd;
        RCALL retardo_10us

;DESPLIEGA EL MENSAJE ING.MEC.ELEC. EN LA SEGUNDA LINEA
        RCALL linea_2 ;CONFIGURACION PARA DESPLRGAR MENSAJE EN LA SEGUNDA LINEA

        LDI R16,'I'
        RCALL dato_lcd
        RCALL retardo_10us
        LDI R16,'N'
        RCALL dato_lcd
        RCALL retardo_10us
        LDI R16,'G'
        RCALL dato_lcd
        RCALL retardo_10us
        LDI R16,'.'
        RCALL dato_lcd
        RCALL retardo_10us
        LDI R16,'M'
        RCALL dato_lcd
        RCALL retardo_10us
        LDI R16,'E'
        RCALL dato_lcd
        RCALL retardo_10us
        LDI R16,'C'
        RCALL dato_lcd
        RCALL retardo_10us
        LDI R16,'.'
        RCALL dato_lcd
        RCALL retardo_10us
        LDI R16,'E'
        RCALL dato_lcd
        RCALL retardo_10us
        LDI R16,'L'
```

```
                RCALL dato_lcd
                RCALL retardo_10us
                LDI R16,'E'
                RCALL dato_lcd
                RCALL retardo_10us
                LDI R16,'C'
                RCALL dato_lcd
                RCALL retardo_10us
                LDI R16,'.'
                RCALL dato_lcd
                RCALL retardo_10us
                RET

;*********************************************************************
*
;*********************************************************************
init_lcd:
                LDI R16,$20 ;BUS DE DATOS A 4 BITS
                OUT PORTB,R16
                RCALL enable_lcd
                RCALL retardo_40us

                OUT PORTB,R16; MODO TRANSFERENCIA A 4 BITS $28
                RCALL enable_lcd
                LDI R16,$80
                OUT PORTB,R16
                RCALL enable_lcd
                RCALL retardo_40us

                LDI R16,$00 ;MODO FUNCIONAMIENTO: INCREMENTO
CONTADOR, DISPLAY QUIETO $06
                OUT PORTB,R16
                RCALL enable_lcd
                LDI R16,$60
                OUT PORTB,R16
                RCALL enable_lcd
```

```
                RCALL retardo_40us

                LDI R16,$00 ;ON/OFF: DISPLAY ON, CURSOR OFF, PARPADEO
OFF $0E
                OUT PORTB,R16
                RCALL enable_lcd
                LDI R16,$C0
                OUT PORTB,R16
                RCALL enable_lcd
                RCALL retardo_40us

                LDI R16,$00 ; DISPLAY CLEAR
                OUT PORTB,R16
                RCALL enable_lcd
                LDI R16,$10
                OUT PORTB,R16
                RCALL enable_lcd
                RCALL retardo_40us
                RET

;**********************************************************************
**********************
;**********************************************************************
**********************
enable_lcd:
                SBI PORTB,0 ;E=1
                CBI PORTB,0 ;E=0
                RET

;**********************************************************************
**********************
;**********************************************************************
**********************
retardo_40us:
;    delay loop generator
;     500000 cycles:
```

```
; ----------------------------
; delaying 499995 cycles:
        ldi  R17, $0F
WGLOOP01:  ldi  R18, $37
WGLOOP11:  ldi  R19, $C9
WGLOOP21:  dec  R19
        brne WGLOOP21
        dec  R18
        brne WGLOOP11
        dec  R17
        brne WGLOOP01
; ----------------------------
; delaying 3 cycles:
        ldi  R17, $01
WGLOOP31:  dec  R17
        brne WGLOOP31
; ----------------------------
; delaying 2 cycles:
        nop
        nop
; ============================
RET
;*****************************************************************************
*
;*****************************************************************************
*********************
dato_lcd:
                LDI R18,$02 ; RS=1
                OUT PORTB,R18
                LDI R17,$F0
                AND R17,R16
                ORI R17,$02
                OUT PORTB,R17
                RCALL enable_lcd
                SWAP R16
                LDI R17,$F0
```

```
                AND R17,R16
                ORI R17,$02
                OUT PORTB,R17
                RCALL enable_lcd
                RET

;*********************************************************************
********************
;*********************************************************************
**
retardo_10us:

; ============================
;    delay loop generator
;     20 cycles:
; ----------------------------
; delaying 18 cycles:
        ldi  R17, $06
WGLOOP000:  dec  R17
        brne WGLOOP000
; ----------------------------
; delaying 2 cycles:
        nop
        nop
RET
;*********************************************************************
;*********************************************************************
*******************************
linea_2:
                CBI PORTB,1
                LDI R16,$00 ;MODO FUNCIONAMIENTO: INCREMENTO
CONTADOR, DISPLAY QUIETO $06
                OUT PORTB,R16
                RCALL enable_lcd
                LDI R16,$60
                OUT PORTB,R16
```

```
		RCALL enable_lcd
		RCALL retardo_40us

		LDI R16,$C0 ; DIRECCION $40 DD RAM LINEA 2
		OUT PORTB,R16
		RCALL enable_lcd
		LDI R16,$20
		OUT PORTB,R16
		RCALL enable_lcd
		RCALL retardo_40us
		RET

;**********************************************************************
**********************************
;**********************************************************************
**********************************
inicio_adc_algoritmo_ph:
		SBI PORTD,2
		CBI PORTD,3
		LDI R16,$00 ;MODO FUNCIONAMIENTO: INCREMENTO
CONTADOR, DISPLAY QUIETO $06
		OUT PORTB,R16
		RCALL enable_lcd
		LDI R16,$60
		OUT PORTB,R16
		RCALL enable_lcd
		RCALL retardo_40us
		LDI R16,$C0 ; DIRECCION $42 DD RAM LINEA 2
		OUT PORTB,R16
		RCALL enable_lcd
		LDI R16,$20
		OUT PORTB,R16
		RCALL enable_lcd
		RCALL retardo_40us
;SEGUDA LINEA PH=0.00
		LDI R16,' '
```

```
RCALL dato_lcd
RCALL retardo_10us
LDI R16,' '
RCALL dato_lcd
RCALL retardo_10us
LDI R16,' '
RCALL dato_lcd
RCALL retardo_10us
LDI R16,'P'
RCALL dato_lcd
RCALL retardo_10us
LDI R16,'H'
RCALL dato_lcd
RCALL retardo_10us
LDI R16,'='
RCALL dato_lcd
RCALL retardo_10us
LDI R16,' '
RCALL dato_lcd
RCALL retardo_10us
LDI R16,'.'
RCALL dato_lcd
RCALL retardo_10us
LDI R16,' '
RCALL dato_lcd
RCALL retardo_10us
LDI R16,' '
RCALL dato_lcd
RCALL retardo_10us
LDI R16,' '
RCALL dato_lcd
RCALL retardo_10us
LDI R16,' '
RCALL dato_lcd
RCALL retardo_10us
LDI R16,' '
```

```
                RCALL dato_lcd
                RCALL retardo_10us

inicio_adc_ph:LDI R17,$20
                OUT ADMUX,R17
                LDI R18,$E1
                OUT ADCSRA,R18
espera:         SBIS ADCSRA,4
                RJMP espera
                IN R16,ADCH
                RCALL Tx
                RCALL algoritmo_ph
                ;R21=UNIDAD
                ;R17=1_DECIMA
                ;R16=2_DECIMA
                RCALL unidad
                RCALL decima_1
                RCALL decima_2
                SBIC PIND,0
                RET
                RJMP inicio_adc_ph
;*****************************************
Tx:             ldi r17,$02
                out ucsra,r17
                ldi r17,$08
                out ucsrb,r17
                ldi r17,$06
                out ucsrc,r17
                ldi r17,$0c
                ldi r18,$00
                out ubrrl,r17
                out ubrrh,r18
                clr r17
                clr r18
                rcall usart_transmit
                ret
```

```
usart_transmit:; Wait for empty transmit buffer
                    sbis UCSRA,UDRE
                    rjmp usart_transmit
                    ; Put data (r16) into buffer, sends the data
                    out UDR,r16
                    ret

;***********************************************************


;************************************************
algoritmo_ph:
;*********************************************************************
;1.- se multiplica el resultado (r16) por $0027
;***** Subroutine Register Variables

.def    mc16uL      =r16        ;multiplicand low byte
.def    mc16uH      =r17        ;multiplicand high byte
.def    mp16uL      =r18        ;multiplier low byte
.def    mp16uH      =r19        ;multiplier high byte
.def    m16u0 =r18          ;result byte 0 (LSB)
.def    m16u1 =r19          ;result byte 1
.def    m16u2 =r20          ;result byte 2
.def    m16u3 =r21          ;result byte 3 (MSB)
.def    mcnt16u     =r22        ;loop counter

;***** Code
                clr r18
                clr r17
                clr r19
                ldi mp16uL,$27

mpy16u:     clr     m16u3       ;clear 2 highest bytes of result
                clr     m16u2
                ldi     mcnt16u,16  ;init loop counter
```

```
                lsr     mp16uH
                ror     mp16uL

m16u_1:         brcc    noad8           ;if bit 0 of multiplier set
                add     m16u2,mc16uL        ;add multiplicand Low to byte 2 of res
                adc     m16u3,mc16uH        ;add multiplicand high to byte 3 of res
noad8: ror      m16u3           ;shift right result byte 3
                ror     m16u2           ;rotate right result byte 2
                ror     m16u1           ;rotate result byte 1 and multiplier High
                ror     m16u0           ;rotate result byte 0 and multiplier Low
                dec     mcnt16u             ;decrement loop counter
                brne m16u_1             ;if not done, loop more

;*************************************************************************
**
;*************************************************************************
**
;se divide el resultado (r19,r18) entre 1000=$03E8
;***** Subroutine Register Variables

.def    drem16uL=r14   ;remainder low byte
.def    drem16uH=r15   ;remainder high byte
.def    dres16uL=r18   ;result low byte
.def    dres16uH=r19   ;result high byte
.def    dd16uL      =r18   ;dividend low byte
.def    dd16uH      =r19   ;dividend high byte
.def    dv16uL      =r16   ;divisor low byte
.def    dv16uH      =r17   ;divisor high byte
.def    dcnt16u     =r20   ;loop counter

;***** Code
                clr r14
                clr r15
                clr r16
                clr r17
                clr r20
```

```
                ldi dv16uL,$E8
                ldi dv16uH,$03

div16u:         clr     drem16uL        ;clear remainder Low byte
                sub     drem16uH,drem16uH;clear remainder High byte and carry
                ldi     dcnt16u,17      ;init loop counter
d16u_1:         rol     dd16uL                  ;shift left dividend
                rol     dd16uH
                dec     dcnt16u                 ;decrement counter
                brne    d16u_2                  ;if done
                rjmp convercion                         ;

d16u_2:         rol     drem16uL        ;shift dividend into remainder
                rol     drem16uH
                sub     drem16uL,dv16uL     ;remainder = remainder - divisor
                sbc     drem16uH,dv16uH     ;
                brcc d16u_3             ;if result negative
                add     drem16uL,dv16uL     ;    restore remainder
                adc     drem16uH,dv16uH
                clc                     ;    clear carry to be shifted into result
                rjmp    d16u_1                  ;else
d16u_3:         sec                     ;    set carry to be shifted into result
                rjmp    d16u_1
;*****************************************************************
convercion:
;se hace la convercion de hexadecimal a decimal dividiendo entre 10=$0A
;primero de la unidad y luego la decima

;***** Subroutine Register Variables

.def    drem16uL=r21   ;remainder low byte
.def    drem16uH=r22   ;remainder high byte
.def    dres16uL=r18   ;result low byte
.def    dres16uH=r19   ;result high byte
.def    dd16uL         =r18   ;dividend low byte
```

```
.def    dd16uH      =r19   ;dividend high byte
.def    dv16uL      =r16   ;divisor low byte
.def    dv16uH      =r17   ;divisor high byte
.def    dcnt16u     =r20   ;loop counter

;***** Code
            clr r16
            clr r17
            clr r20

            ldi dv16uL,$0A
            ldi dv16uH,$00

div16u1:    clr     drem16uL      ;clear remainder Low byte
            sub     drem16uH,drem16uH;clear remainder High byte and
carry
            ldi     dcnt16u,17    ;init loop counter
d16u_11:    rol     dd16uL        ;shift left dividend
            rol     dd16uH
            dec     dcnt16u             ;decrement counter
            brne    d16u_21             ;if done
            rjmp convercion2                ;

d16u_21:    rol     drem16uL      ;shift dividend into remainder
            rol     drem16uH
            sub     drem16uL,dv16uL     ;remainder = remainder - divisor
            sbc     drem16uH,dv16uH     ;
            brcc d16u_31            ;if result negative
            add     drem16uL,dv16uL     ;    restore remainder
            adc     drem16uH,dv16uH
            clc                     ;    clear carry to be shifted into result
            rjmp    d16u_11             ;else
d16u_31:    sec                     ;    set carry to be shifted into result
            rjmp    d16u_11
;*****************************************************************
convercion2:
```

```
;***** Subroutine Register Variables

.def    drem16uL=r23   ;remainder low byte
.def    drem16uH=r24   ;remainder high byte
.def    dres16uL=r14   ;result low byte
.def    dres16uH=r15   ;result high byte
.def    dd16uL      =r14   ;dividend low byte
.def    dd16uH      =r15   ;dividend high byte
.def    dv16uL      =r16   ;divisor low byte
.def    dv16uH      =r17   ;divisor high byte
.def    dcnt16u     =r20   ;loop counter


;***** Code
            clr r18
            clr r19
            clr r23
            clr r24
            clr r20

div16u2:    clr     drem16uL      ;clear remainder Low byte
            sub     drem16uH,drem16uH;clear remainder High byte and
carry
            ldi     dcnt16u,17    ;init loop counter
d16u_12:    rol     dd16uL        ;shift left dividend
            rol     dd16uH
            dec     dcnt16u             ;decrement counter
            brne    d16u_22             ;if done
            clr r16

            rjmp VALORPH            ;

d16u_22:    rol     drem16uL      ;shift dividend into remainder
            rol     drem16uH
            sub     drem16uL,dv16uL     ;remainder = remainder - divisor
```

```
                    sbc     drem16uH,dv16uH    ;
                    brcc d16u_32           ;if result negative
                    add     drem16uL,dv16uL    ;    restore remainder
                    adc     drem16uH,dv16uH
                    clc                    ;    clear carry to be shifted into result
                    rjmp    d16u_12                ;else
d16u_32:        sec                    ;    set carry to be shifted into result
                    rjmp    d16u_12

VALORPH:    add r14,r23
                    RCALL conv_bcd
                    CLR R14
                    CLR R23
                    MOV R0,R21
                    MOV R1,R17
                    MOV R2,R16
                    RET
conv_bcd:
;*********************************************************************
*******
;*
;* "bin2BCD8" - 8-bit Binary to BCD conversion
;*
;* This subroutine converts an 8-bit number (fbin) to a 2-digit
;* BCD number (tBCDH:tBCDL).
;*
;* Number of words   :6 + return
;* Number of cycles   :5/50 (Min/Max) + return
;* Low registers used :None
;* High registers used  :2 (fbin/tBCDL,tBCDH)
;*
;* Included in the code are lines to add/replace for packed BCD output.
;*
;*********************************************************************
*****
;***** Subroutine Register Variables
```

```
.def    fbin    =r16            ;8-bit binary value
.def    tBCDL=r16               ;BCD result MSD
.def    tBCDH           =r17            ;BCD result LSD

;***** Code
                mov r16,r14

bin2bcd8:
        clr     tBCDH                   ;clear result MSD
bBCD8_1:  subi      fbin,10         ;input = input - 10
        brcs    bBCD8_2                 ;abort if carry set
        inc     tBCDH                   ;inc MSD
;-----------------------------------------------------------------------
;                               ;Replace the above line with this one
;                               ;for packed BCD output
;       subi    tBCDH,-$10      ;tBCDH = tBCDH + 10
;-----------------------------------------------------------------------
        rjmp    bBCD8_1                 ;loop again
bBCD8_2:subi        fbin,-10        ;compensate extra subtraction
;-----------------------------------------------------------------------
;                               ;Add this line for packed BCD output
;       add     fbin,tBCDH
;-----------------------------------------------------------------------
        ret
;*********************************************************************
;*********************************************************************
**********
unidad:
                        LDI R16,$00
                        CPSE R16,R0
                        RJMP otro00
                        RJMP cero

otro00:         LDI R16,$01
                        CPSE R16,R0
```

```
                RJMP otro1
                RJMP uno

otro1:      LDI R16,$02
                CPSE R16,R0
                RJMP otro2
                RJMP dos

otro2:      LDI R16,$03
                CPSE R16,R0
                RJMP otro3
                RJMP tres

otro3:      LDI R16,$04
                CPSE R16,R0
                RJMP otro4
                RJMP cuatro

otro4:      LDI R16,$05
                CPSE R16,R0
                RJMP otro5
                RJMP cinco

otro5:      LDI R16,$06
                CPSE R16,R0
                RJMP otro6
                RJMP seis

otro6:      LDI R16,$07
                CPSE R16,R0
                RJMP otro7
                RJMP siete

otro7:      LDI R16,$08
                CPSE R16,R0
                RJMP otro8
```

```
                    RJMP ocho

otro8:          LDI R16,$09
                    CPSE R16,R0
                    RET
                    RJMP nueve

cero:   RCALL imprime_unidad
                LDI R16,'0'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

uno:    RCALL imprime_unidad
                LDI R16,'1'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

dos:    RCALL imprime_unidad
                LDI R16,'2'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

tres:   RCALL imprime_unidad
                LDI R16,'3'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

cuatro: RCALL imprime_unidad
                LDI R16,'4'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET
```

```
cinco:  RCALL imprime_unidad
        LDI R16,'5'
        RCALL dato_lcd
        ;RCALL retardo_20us
        RET

seis:   RCALL imprime_unidad
        LDI R16,'6'
        RCALL dato_lcd
        ;RCALL retardo_20us
        RET

siete:  RCALL imprime_unidad
        LDI R16,'7'
        RCALL dato_lcd
        ;RCALL retardo_20us
        RET

ocho:   RCALL imprime_unidad
        LDI R16,'8'
        RCALL dato_lcd
        ;RCALL retardo_20us
        RET

nueve: RCALL imprime_unidad
        LDI R16,'9'
        RCALL dato_lcd
        ;RCALL retardo_20us
        RET

;************************************************
;************************************************
imprime_unidad:

        LDI R16,$00 ;MODO FUNCIONAMIENTO: INCREMENTO
```

```
CONTADOR, DISPLAY QUIETO $06
                OUT PORTB,R16
                RCALL enable_lcd
                LDI R16,$60
                OUT PORTB,R16
                RCALL enable_lcd
                RCALL retardo_40us

                LDI R16,$C0 ; DIRECCION $48 DD RAM LINEA 2
                OUT PORTB,R16
                RCALL enable_lcd
                LDI R16,$80
                OUT PORTB,R16
                RCALL enable_lcd
                RCALL retardo_40us
                RET
;************************************************
;************************************************
decima_1:
                        LDI R18,$00
                        CPSE R18,R1
                        RJMP otro_
                        RJMP cero1

otro_:          LDI R18,$01
                        CPSE R18,R1
                        RJMP otro1_1
                        RJMP uno1

otro1_1:                LDI R18,$02
                        CPSE R18,R1
                        RJMP otro2_1
                        RJMP dos1

otro2_1:                LDI R17,$03
                        CPSE R18,R1
```

```
                RJMP otro3_1
                RJMP tres1

otro3_1:        LDI R18,$04
                CPSE R18,R1
                RJMP otro4_1
                RJMP cuatro1

otro4_1:        LDI R18,$05
                CPSE R18,R1
                RJMP otro5_1
                RJMP cinco1

otro5_1:        LDI R18,$06
                CPSE R18,R1
                RJMP otro6_1
                RJMP seis1

otro6_1:        LDI R18,$07
                CPSE R18,R1
                RJMP otro7_1
                RJMP siete1

otro7_1:        LDI R18,$08
                CPSE R18,R1
                RJMP otro8_1
                RJMP ocho1

otro8_1:        LDI R18,$09
                CPSE R18,R1
                RET
                RJMP nueve1

cero1:  RCALL imprime_decima_1
            LDI R16,'0'
            RCALL dato_lcd
```

```
                ;RCALL retardo_20us
                RET

uno1:    RCALL imprime_decima_1
                LDI R16,'1'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

dos1:   RCALL imprime_decima_1
                LDI R16,'2'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

tres1:   RCALL imprime_decima_1
                LDI R16,'3'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

cuatro1:        RCALL imprime_decima_1
                LDI R16,'4'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

cinco1:RCALL imprime_decima_1
                LDI R16,'5'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

seis1:   RCALL imprime_decima_1
                LDI R16,'6'
                RCALL dato_lcd
```

```
                ;RCALL retardo_20us
                RET

siete1: RCALL imprime_decima_1
                LDI R16,'7'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

ocho1: RCALL imprime_decima_1
                LDI R16,'8'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

nueve1:         RCALL imprime_decima_1
                LDI R16,'9'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET
;*********************************************
;*****************************************
imprime_decima_1:

                LDI R16,$00 ;MODO FUNCIONAMIENTO: INCREMENTO
CONTADOR, DISPLAY QUIETO $06
                OUT PORTB,R16
                RCALL enable_lcd
                LDI R16,$60
                OUT PORTB,R16
                RCALL enable_lcd
                RCALL retardo_40us

                LDI R16,$C0 ; DIRECCION $50 DD RAM LINEA 2
                OUT PORTB,R16
                RCALL enable_lcd
```

```
                LDI R16,$A0
                OUT PORTB,R16
                RCALL enable_lcd
                RCALL retardo_40us
                RET

;***********************************************
;*************************************************************
decima_2:

                        LDI R17,$00
                        CPSE R17,R2
                        RJMP otro_2
                        RJMP cero2

otro_2:         LDI R17,$01
                        CPSE R17,R2
                        RJMP otro1_2
                        RJMP uno2

otro1_2:                LDI R17,$02
                        CPSE R17,R2
                        RJMP otro2_2
                        RJMP dos2

otro2_2:                LDI R17,$03
                        CPSE R17,R2
                        RJMP otro3_2
                        RJMP tres2

otro3_2:                LDI R17,$04
                        CPSE R17,R2
                        RJMP otro4_2
                        RJMP cuatro2

otro4_2:                LDI R17,$05
```

```
                CPSE R17,R2
                RJMP otro5_2
                RJMP cinco2

otro5_2:        LDI R17,$06
                CPSE R17,R2
                RJMP otro6_2
                RJMP seis2

otro6_2:        LDI R17,$07
                CPSE R17,R2
                RJMP otro7_2
                RJMP siete2

otro7_2:        LDI R17,$08
                CPSE R17,R2
                RJMP otro8_2
                RJMP ocho2

otro8_2:        LDI R17,$09
                CPSE R17,R2
                RET
                RJMP nueve2

cero2:  RCALL imprime_decima_2
        LDI R16,'0'
        RCALL dato_lcd
        ;RCALL retardo_20us
        RET

uno2:   RCALL imprime_decima_2
        LDI R16,'1'
        RCALL dato_lcd
        ;RCALL retardo_20us
        RET
```

```
dos2:   RCALL imprime_decima_2
        LDI R16,'2'
        RCALL dato_lcd
        ;RCALL retardo_20us
        RET

tres2:  RCALL imprime_decima_2
        LDI R16,'3'
        RCALL dato_lcd
        ;RCALL retardo_20us
        RET

cuatro2:    RCALL imprime_decima_2
        LDI R16,'4'
        RCALL dato_lcd
        ;RCALL retardo_20us
        RET
cinco2:RCALL imprime_decima_2
        LDI R16,'5'
        RCALL dato_lcd
        ;RCALL retardo_20us
        RET
seis2:  RCALL imprime_decima_2
        LDI R16,'6'
        RCALL dato_lcd
        ;RCALL retardo_20us
        RET

siete2: RCALL imprime_decima_2
        LDI R16,'7'
        RCALL dato_lcd
        ;RCALL retardo_20us
        RET

ocho2: RCALL imprime_decima_2
        LDI R16,'8'
```

```
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

nueve2:         RCALL imprime_decima_2
                LDI R16,'9'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

;*****************************************************

;*****************************************************
imprime_decima_2:

                LDI R16,$00 ;MODO FUNCIONAMIENTO: INCREMENTO
CONTADOR, DISPLAY QUIETO $06
                OUT PORTB,R16
                RCALL enable_lcd
                LDI R16,$60
                OUT PORTB,R16
                RCALL enable_lcd
                RCALL retardo_40us

                LDI R16,$C0 ; DIRECCION $ 51 RAM LINEA 2
                OUT PORTB,R16
                RCALL enable_lcd
                LDI R16,$B0
                OUT PORTB,R16
                RCALL enable_lcd
                RCALL retardo_40us
                RET
```

```
;*****************************************************
;*********************************************************************
inicio_adc_algoritmo_temp:

LDI R16,$00 ;MODO FUNCIONAMIENTO: INCREMENTO CONTADOR, DISPLAY QUIETO $06
            OUT PORTB,R16
            RCALL enable_lcd
            LDI R16,$60
            OUT PORTB,R16
            RCALL enable_lcd
            RCALL retardo_40us

            LDI R16,$C0 ; DIRECCION $42 DD RAM LINEA 2
            OUT PORTB,R16
            RCALL enable_lcd
            LDI R16,$20
            OUT PORTB,R16
            RCALL enable_lcd
            RCALL retardo_40us

;SEGUDA LINEA TEMP=00.00

            LDI R16,'T'
            RCALL dato_lcd
            RCALL retardo_10us
            LDI R16,'E'
            RCALL dato_lcd
            RCALL retardo_10us
            LDI R16,'M'
            RCALL dato_lcd
            RCALL retardo_10us
            LDI R16,'P'
            RCALL dato_lcd
            RCALL retardo_10us
```

```
                LDI R16,'='
                RCALL dato_lcd
                RCALL retardo_10us
                LDI R16,' '
                RCALL dato_lcd
                RCALL retardo_10us
                LDI R16,' '
                RCALL dato_lcd
                RCALL retardo_10us
                LDI R16,'.'
                RCALL dato_lcd
                RCALL retardo_10us
                LDI R16,' '
                RCALL dato_lcd
                RCALL retardo_10us
                LDI R16,' '
                RCALL dato_lcd
                RCALL retardo_10us
                LDI R16,'°'
                RCALL dato_lcd
                RCALL retardo_10us
                LDI R16,'C'
                RCALL dato_lcd
                RCALL retardo_10us
                LDI R16,' '
                RCALL dato_lcd
                RCALL retardo_10us

inicio_adc_temp: LDI R17,$21
                 OUT ADMUX,R17
                 LDI R18,$E1
                 OUT ADCSRA,R18
espera1:         SBIS ADCSRA,4
                 RJMP espera1
                 IN R16,ADCH
                 CPI R16,$3B
```

```
                    BRLO calentador
                    CBI PORTD,4
continua:   RCALL Tx
                    RCALL algoritmo_temp
                    ;r0=1_unidad
                    ;r1=2_unidad
                    ;r2=1_decima
                    ;r3=2_decima
                    RCALL unidad_1
                    RCALL unidad_2
                    RCALL decima_1_temp
                    RCALL decima_2_temp
                    SBIS PIND,0
                    RET
                    RJMP inicio_adc_temp

;*******************************************************
calentador: SBI PORTD,4
                    rjmp continua
;*******************************************************

;*************************************************************
algoritmo_temp:
;*********************************************************************
;1.- se multiplica el resultado (r16) por $0027

;***** Subroutine Register Variables

.def    mc16uL      =r16        ;multiplicand low byte
.def    mc16uH      =r17        ;multiplicand high byte
.def    mp16uL      =r18        ;multiplier low byte
.def    mp16uH      =r19        ;multiplier high byte
.def    m16u0 =r18          ;result byte 0 (LSB)
.def    m16u1 =r19          ;result byte 1
.def    m16u2 =r20          ;result byte 2
.def    m16u3 =r21          ;result byte 3 (MSB)
```

```
.def    mcnt16u     =r22            ;loop counter

;***** Code

            clr r18
            clr r17
            clr r19
            ldi mp16uL,$27

            clr     m16u3           ;clear 2 highest bytes of result
            clr     m16u2
            ldi     mcnt16u,16      ;init loop counter
            lsr     mp16uH
            ror     mp16uL

m16u_11:    brcc    noad81          ;if bit 0 of multiplier set
            add     m16u2,mc16uL        ;add multiplicand Low to byte 2 of res
            adc     m16u3,mc16uH        ;add multiplicand high to byte 3 of res
noad81:     ror     m16u3           ;shift right result byte 3
            ror     m16u2           ;rotate right result byte 2
            ror     m16u1           ;rotate result byte 1 and multiplier High
            ror     m16u0           ;rotate result byte 0 and multiplier Low
            dec     mcnt16u             ;decrement loop counter
            brne m16u_11            ;if not done, loop more

;*****************************************************************
**
;*****************************************************************
**
;se divide el resultado (r19,r18) entre 100=$0064
;***** Subroutine Register Variables

.def    drem16uL=r14   ;remainder low byte
.def    drem16uH=r15   ;remainder high byte
.def    dres16uL=r18   ;result low byte
.def    dres16uH=r19   ;result high byte
```

```
.def    dd16uL      =r18  ;dividend low byte
.def    dd16uH      =r19  ;dividend high byte
.def    dv16uL      =r16  ;divisor low byte
.def    dv16uH      =r17  ;divisor high byte
.def    dcnt16u     =r20  ;loop counter

;***** Code

            clr r14
            clr r15
            clr r16
            clr r17
            clr r20

            ldi dv16uL,$64
            ldi dv16uH,$00

            clr     drem16uL      ;clear remainder Low byte
            sub     drem16uH,drem16uH;clear remainder High byte and carry
            ldi     dcnt16u,17    ;init loop counter
d16u_111:rol dd16uL               ;shift left dividend
            rol     dd16uH
            dec     dcnt16u             ;decrement counter
            brne    d16u_211            ;if done
            rjmp valor_°C         ;

d16u_211:   rol     drem16uL      ;shift dividend into remainder
            rol     drem16uH
            sub     drem16uL,dv16uL     ;remainder = remainder - divisor
            sbc     drem16uH,dv16uH     ;
            brcc d16u_311               ;if result negative
            add     drem16uL,dv16uL     ;    restore remainder
            adc     drem16uH,dv16uH
            clc                   ;    clear carry to be shifted into result
            rjmp    d16u_111            ;else
d16u_311:   sec                   ;    set carry to be shifted into result
```

```
                rjmp    d16u_111

valor_°C:       ;R18=UNIDAD°C
                        ;R14=DECIMA°C
                        RCALL conv_bcd_2
                        ret
conv_bcd_2:
;*****************************************************************************
*******
;*
;* "bin2BCD8" - 8-bit Binary to BCD conversion
;*
;* This subroutine converts an 8-bit number (fbin) to a 2-digit
;* BCD number (tBCDH:tBCDL).
;*
;* Number of words   :6 + return
;* Number of cycles  :5/50 (Min/Max) + return
;* Low registers used :None
;* High registers used  :2 (fbin/tBCDL,tBCDH)
;*
;* Included in the code are lines to add/replace for packed BCD output.
;*
;*****************************************************************************
*******
;***** Subroutine Register Variables

.def    fbin    =r16            ;8-bit binary value
.def    tBCDL=r16               ;BCD result MSD
.def    tBCDH           =r17            ;BCD result LSD
;***** Code
                mov r16,r18
        clr     tBCDH                   ;clear result MSD
bBCD8_111:  subi    fbin,10         ;input = input - 10
        brcs    bBCD8_211           ;abort if carry set
        inc     tBCDH                   ;inc MSD
;-----------------------------------------------------------------------
```

```
;                               ;Replace the above line with this one
;                               ;for packed BCD output
;       subi    tBCDH,-$10      ;tBCDH = tBCDH + 10
;---------------------------------------------------------------------
        rjmp    bBCD8_111       ;loop again
bBCD8_211:subi      fbin,-10    ;compensate extra subtraction
;---------------------------------------------------------------------
;                               ;Add this line for packed BCD output
;       add     fbin,tBCDH
;---------------------------------------------------------------------

.def    fbin    =r18            ;8-bit binary value
.def    tBCDL=r18               ;BCD result MSD
.def    tBCDH       =r19        ;BCD result LSD

;***** Code
                mov r18,r14


        clr     tBCDH           ;clear result MSD
bBCD8_11:   subi    fbin,10     ;input = input - 10
        brcs    bBCD8_21        ;abort if carry set
        inc     tBCDH           ;inc MSD
;---------------------------------------------------------------------
;                               ;Replace the above line with this one
;                               ;for packed BCD output
;       subi    tBCDH,-$10      ;tBCDH = tBCDH + 10
;---------------------------------------------------------------------
        rjmp    bBCD8_11        ;loop again
bBCD8_21:subi       fbin,-10    ;compensate extra subtraction
;---------------------------------------------------------------------
;                               ;Add this line for packed BCD output
;       add     fbin,tBCDH
;---------------------------------------------------------------------
                ;r17=1_unidad
                ;r16=2_unidad
```

```
                ;r19=1_decima
                ;r18=2_decima

                MOV R0,R17
                MOV R1,R16
                MOV R2,R19
                MOV R3,R18

RET
;*****************************************
;*************************************************************
;*****************************************************************************
***********
unidad_1:
                        LDI R16,$00
                        CPSE R16,R0
                        RJMP otro000
                        RJMP cero11

otro000:        LDI R16,$01
                        CPSE R16,R0
                        RJMP otro111
                        RJMP uno11

otro111:        LDI R16,$02
                        CPSE R16,R0
                        RJMP otro211
                        RJMP dos11

otro211:        LDI R16,$03
                        CPSE R16,R0
                        RJMP otro311
                        RJMP tres11

otro311:        LDI R16,$04
                        CPSE R16,R0
```

```
                RJMP otro411
                RJMP cuatro11

otro411:        LDI R16,$05
                CPSE R16,R0
                RJMP otro511
                RJMP cinco11

otro511:        LDI R16,$06
                CPSE R16,R0
                RJMP otro611
                RJMP seis11

otro611:        LDI R16,$07
                CPSE R16,R0
                RJMP otro711
                RJMP siete11

otro711:        LDI R16,$08
                CPSE R16,R0
                RJMP otro811
                RJMP ocho11

otro811:        LDI R16,$09
                CPSE R16,R0
                RET
                RJMP nueve11

cero11:  RCALL imprime_unidad_1_temp
                LDI R16,'0'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

uno11:   RCALL imprime_unidad_1_temp
                LDI R16,'1'
```

```
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

dos11: RCALL imprime_unidad_1_temp
                LDI R16,'2'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

tres11: RCALL imprime_unidad_1_temp
                LDI R16,'3'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

cuatro11:       RCALL imprime_unidad_1_temp
                LDI R16,'4'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

cinco11:        RCALL imprime_unidad_1_temp
                LDI R16,'5'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

seis11: RCALL imprime_unidad_1_temp
                LDI R16,'6'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

siete11:        RCALL imprime_unidad_1_temp
                LDI R16,'7'
```

```
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

ocho11:         RCALL imprime_unidad_1_temp
                LDI R16,'8'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

nueve11:        RCALL imprime_unidad_1_temp
                LDI R16,'9'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

;************************************************
;************************************************

imprime_unidad_1_temp:

                LDI R16,$00 ;MODO FUNCIONAMIENTO: INCREMENTO
CONTADOR, DISPLAY QUIETO $06
                OUT PORTB,R16
                RCALL enable_lcd
                LDI R16,$60
                OUT PORTB,R16
                RCALL enable_lcd
                RCALL retardo_40us

                LDI R16,$C0 ; DIRECCION $47 DD RAM LINEA 2
                OUT PORTB,R16
                RCALL enable_lcd
                LDI R16,$70
                OUT PORTB,R16
                RCALL enable_lcd
```

```
                RCALL retardo_40us
                RET
;************************************************
;************************************************************
unidad_2:
                        LDI R16,$00
                        CPSE R16,R1
                        RJMP otro0002
                        RJMP cero112

otro0002:       LDI R16,$01
                        CPSE R16,R1
                        RJMP otro1112
                        RJMP uno112

otro1112:       LDI R16,$02
                        CPSE R16,R1
                        RJMP otro2112
                        RJMP dos112

otro2112:       LDI R16,$03
                        CPSE R16,R1
                        RJMP otro3112
                        RJMP tres112

otro3112:       LDI R16,$04
                        CPSE R16,R1
                        RJMP otro4112
                        RJMP cuatro112

otro4112:       LDI R16,$05
                        CPSE R16,R1
                        RJMP otro5112
                        RJMP cinco112

otro5112:       LDI R16,$06
```

```
                CPSE R16,R1
                RJMP otro6112
                RJMP seis112

otro6112:       LDI R16,$07
                CPSE R16,R1
                RJMP otro7112
                RJMP siete112

otro7112:       LDI R16,$08
                CPSE R16,R1
                RJMP otro8112
                RJMP ocho112

otro8112:       LDI R16,$09
                CPSE R16,R1
                RET
                RJMP nueve112

cero112:  RCALL imprime_unidad_2_temp
          LDI R16,'0'
          RCALL dato_lcd
          ;RCALL retardo_20us
          RET

uno112:   RCALL imprime_unidad_2_temp
          LDI R16,'1'
          RCALL dato_lcd
          ;RCALL retardo_20us
          RET

dos112:   RCALL imprime_unidad_2_temp
          LDI R16,'2'
          RCALL dato_lcd
          ;RCALL retardo_20us
          RET
```

```
tres112:    RCALL imprime_unidad_2_temp
            LDI R16,'3'
            RCALL dato_lcd
            ;RCALL retardo_20us
            RET

cuatro112:  RCALL imprime_unidad_2_temp
            LDI R16,'4'
            RCALL dato_lcd
            ;RCALL retardo_20us
            RET

cinco112:   RCALL imprime_unidad_2_temp
            LDI R16,'5'
            RCALL dato_lcd
            ;RCALL retardo_20us
            RET

seis112:    RCALL imprime_unidad_2_temp
            LDI R16,'6'
            RCALL dato_lcd
            ;RCALL retardo_20us
            RET

siete112:   RCALL imprime_unidad_2_temp
            LDI R16,'7'
            RCALL dato_lcd
            ;RCALL retardo_20us
            RET

ocho112:    RCALL imprime_unidad_2_temp
            LDI R16,'8'
            RCALL dato_lcd
            ;RCALL retardo_20us
            RET
```

```
nueve112:   RCALL imprime_unidad_2_temp
            LDI R16,'9'
            RCALL dato_lcd
            ;RCALL retardo_20us
            RET

;****************************************************
;****************************************************
imprime_unidad_2_temp:

            LDI R16,$00 ;MODO FUNCIONAMIENTO: INCREMENTO
CONTADOR, DISPLAY QUIETO $06
            OUT PORTB,R16
            RCALL enable_lcd
            LDI R16,$60
            OUT PORTB,R16
            RCALL enable_lcd
            RCALL retardo_40us

            LDI R16,$C0 ; DIRECCION $48 DD RAM LINEA 2
            OUT PORTB,R16
            RCALL enable_lcd
            LDI R16,$80
            OUT PORTB,R16
            RCALL enable_lcd
            RCALL retardo_40us
            RET
;************************************************************
;************************************************************
**********

decima_1_temp:
                  LDI R16,$00
                  CPSE R16,R2
                  RJMP otro0003
```

```
                RJMP cero113

otro0003:       LDI R16,$01
                CPSE R16,R2
                RJMP otro1113
                RJMP uno113

otro1113:       LDI R16,$02
                CPSE R16,R2
                RJMP otro2113
                RJMP dos113

otro2113:       LDI R16,$03
                CPSE R16,R2
                RJMP otro3113
                RJMP tres113

otro3113:       LDI R16,$04
                CPSE R16,R2
                RJMP otro4113
                RJMP cuatro113

otro4113:       LDI R16,$05
                CPSE R16,R2
                RJMP otro5113
                RJMP cinco113

otro5113:       LDI R16,$06
                CPSE R16,R2
                RJMP otro6113
                RJMP seis113

otro6113:       LDI R16,$07
                CPSE R16,R2
                RJMP otro7113
                RJMP siete113
```

```
otro7113:       LDI R16,$08
                CPSE R16,R2
                RJMP otro8113
                RJMP ocho113

otro8113:       LDI R16,$09
                CPSE R16,R2
                RET
                RJMP nueve113

cero113:  RCALL imprime_decima_1_temp
          LDI R16,'0'
          RCALL dato_lcd
          ;RCALL retardo_20us
          RET

uno113:   RCALL imprime_decima_1_temp
          LDI R16,'1'
          RCALL dato_lcd
          ;RCALL retardo_20us
          RET

dos113:   RCALL imprime_decima_1_temp
          LDI R16,'2'
          RCALL dato_lcd
          ;RCALL retardo_20us
          RET

tres113:  RCALL imprime_decima_1_temp
          LDI R16,'3'
          RCALL dato_lcd
          ;RCALL retardo_20us
          RET

cuatro113:  RCALL imprime_decima_1_temp
```

```
                LDI R16,'4'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

cinco113:       RCALL imprime_decima_1_temp
                LDI R16,'5'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

seis113:        RCALL imprime_decima_1_temp
                LDI R16,'6'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

siete113:       RCALL imprime_decima_1_temp
                LDI R16,'7'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

ocho113:        RCALL imprime_decima_1_temp
                LDI R16,'8'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

nueve113:       RCALL imprime_decima_1_temp
                LDI R16,'9'
                RCALL dato_lcd
                ;RCALL retardo_20us
                RET

;**************************************************
```

```
;*********************************************************
imprime_decima_1_temp:

            LDI R16,$00 ;MODO FUNCIONAMIENTO: INCREMENTO CONTADOR, DISPLAY QUIETO $06
            OUT PORTB,R16
            RCALL enable_lcd
            LDI R16,$60
            OUT PORTB,R16
            RCALL enable_lcd
            RCALL retardo_40us

            LDI R16,$C0 ; DIRECCION $50 DD RAM LINEA 2
            OUT PORTB,R16
            RCALL enable_lcd
            LDI R16,$A0
            OUT PORTB,R16
            RCALL enable_lcd
            RCALL retardo_40us
            RET
;*********************************************************
;*********************************************************
decima_2_temp:
                  LDI R16,$00
                  CPSE R16,R3
                  RJMP otro0004
                  RJMP cero114

otro0004:         LDI R16,$01
                  CPSE R16,R3
                  RJMP otro1114
                  RJMP uno114

otro1114:         LDI R16,$02
                  CPSE R16,R3
                  RJMP otro2114
```

```
                RJMP dos114

otro2114:       LDI R16,$03
                CPSE R16,R3
                RJMP otro3114
                RJMP tres114

otro3114:       LDI R16,$04
                CPSE R16,R3
                RJMP otro4114
                RJMP cuatro114

otro4114:       LDI R16,$05
                CPSE R16,R3
                RJMP otro5114
                RJMP cinco114

otro5114:       LDI R16,$06
                CPSE R16,R3
                RJMP otro6114
                RJMP seis114

otro6114:       LDI R16,$07
                CPSE R16,R3
                RJMP otro7114
                RJMP siete114

otro7114:       LDI R16,$08
                CPSE R16,R3
                RJMP otro8114
                RJMP ocho114

otro8114:       LDI R16,$09
                CPSE R16,R2
                RET
                RJMP nueve114
```

```
cero114:   RCALL imprime_decima_2_temp
           LDI R16,'0'
           RCALL dato_lcd
           ;RCALL retardo_20us
           RET

uno114:    RCALL imprime_decima_2_temp
           LDI R16,'1'
           RCALL dato_lcd
           ;RCALL retardo_20us
           RET

dos114:    RCALL imprime_decima_2_temp
           LDI R16,'2'
           RCALL dato_lcd
           ;RCALL retardo_20us
           RET

tres114:   RCALL imprime_decima_2_temp
           LDI R16,'3'
           RCALL dato_lcd
           ;RCALL retardo_20us
           RET

cuatro114: RCALL imprime_decima_2_temp
           LDI R16,'4'
           RCALL dato_lcd
           ;RCALL retardo_20us
           RET

cinco114:  RCALL imprime_decima_2_temp
           LDI R16,'5'
           RCALL dato_lcd
           ;RCALL retardo_20us
           RET
```

```
seis114:    RCALL imprime_decima_2_temp
            LDI R16,'6'
            RCALL dato_lcd
            ;RCALL retardo_20us
            RET

siete114:   RCALL imprime_decima_2_temp
            LDI R16,'7'
            RCALL dato_lcd
            ;RCALL retardo_20us
            RET

ocho114:    RCALL imprime_decima_2_temp
            LDI R16,'8'
            RCALL dato_lcd
            ;RCALL retardo_20us
            RET

nueve114:   RCALL imprime_decima_2_temp
            LDI R16,'9'
            RCALL dato_lcd
            ;RCALL retardo_20us
            RET

;*****************************************************
;*****************************************************
imprime_decima_2_temp:

            LDI R16,$00 ;MODO FUNCIONAMIENTO: INCREMENTO CONTADOR, DISPLAY QUIETO $06
            OUT PORTB,R16
            RCALL enable_lcd
            LDI R16,$60
            OUT PORTB,R16
            RCALL enable_lcd
```

```
                RCALL retardo_40us

                LDI R16,$C0 ; DIRECCION $50 DD RAM LINEA 2
                OUT PORTB,R16
                RCALL enable_lcd
                LDI R16,$B0
                OUT PORTB,R16
                RCALL enable_lcd
                RCALL retardo_40us
                RET
;*****************************
```

ANEXO 2 DIAGRAMAS DE CIRCUITOS DE ACONDICIONAMIENTO DE SAÑALES

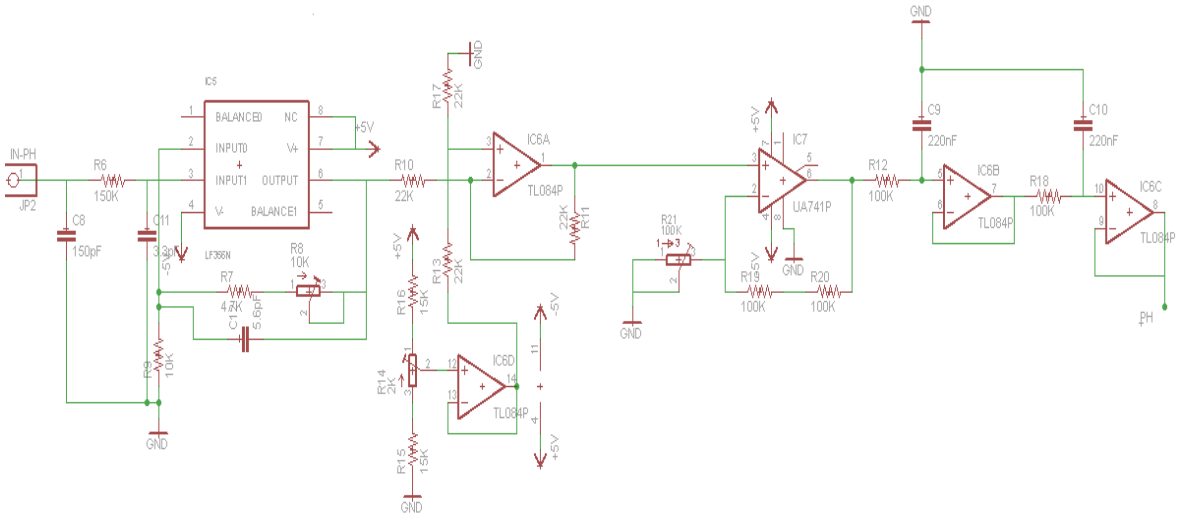

Figura A2-1 Circuito de acondicionamiento de la sonda de pH

El diagrama que se utilizó para el acondicionamiento de la señal de la sonda de pH fue el diseñado por José Manuel García. En cual a la salida del mismo da una tensión de 0V á 999mV que corresponde a los niveles de pH de 0 á 9.99, por lo que a la salida se opto por incorporar un amplificador no inversor con ganancia de 5 para poder tener la salida de tensión de 0V-5V para los niveles de 0-9.9 de pH.

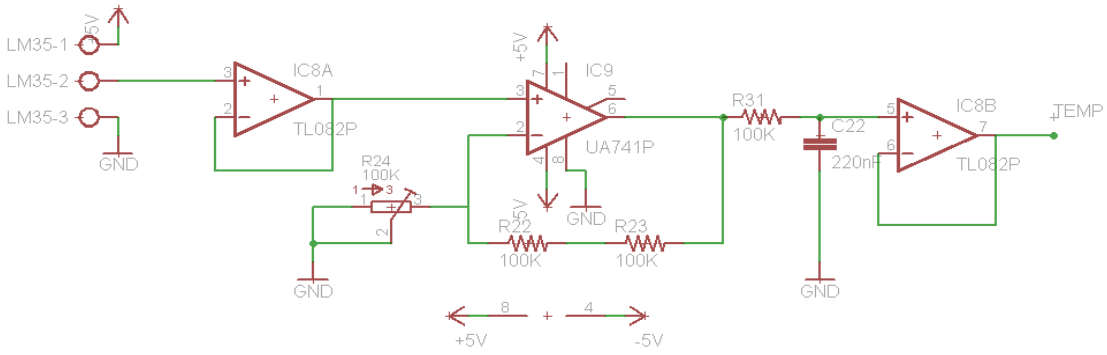

Figura A2-2 Circuito de acondicionamiento del LM35DZ

En este caso con el circuito de acondicionamiento del sensor de temperatura (LM35DZ), como ya se mencionó funciona de manera lineal es decir: cada 10mV por grado °C solo se opto por de igual manera incorporar un amplificador no inversor de ganancia de 5 para obtener de igual forma a la salida una tensión de 0V-5V para los niveles de 0-100°C, y se coloco un seguidor de voltaje para que la tensión debida a la resistencia eléctrica del cable sea mínima al tener una impedancia muy alta y un filtro paso bajo con una frecuencia de corte de aproximadamente 10Hz para eliminar el ruido debido a interferencias.