SUBSECRETARÍA DE EDUCACIÓN SUPERIOR
DIRECCIÓN GENERAL DE EDUCACIÓN SUPERIOR TECNOLÓGICA
INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ

SECRETARÍA DE EDUCACIÓN PÚBLICA

SEP

INSTITUTO TECNOLÓGICO DE TUXTLA GUTIERREZ

Sistema de adquisición de datos para invernadero

Reporte Final

Integrantes:

David Enrique Matías Hernández

Francisco Arturo Montesinos González

Asesor:

Ing. José Ángel Zepeda Hernández

Tuxtla Gutiérrez Chiapas, Enero del 2012.

ÍNDICE

Pág.

CAPITULO I

Introducción---------- 5

Justificación---------- 6

Objetivos: General y específicos---------- 7

Caracterización del área en que participó---------- 8

Problemas a resolver, priorizándolos---------- 9

Alcances y limitaciones---------- 10

CAPITULO II

Fundamento teórico---------- 11

CAPITULO III

Procedimiento y descripción de las actividades realizadas---------- 32

Resultados, planos, gráficas, prototipos y programas---------- 42

CAPÍTULO IV

Conclusiones y recomendaciones---------- 56

Referencias bibliográficas y virtuales---------- 57

Anexo A---------- 58

Anexo B---------- 66

Anexo C---------- 90

Anexo D---------- 95

ÍNDICE DE FIGURAS

Pág.
Figura 1.- Organigrama de la empresa en la que se participó---------- 8
Figura 2.- Estructura básica de un sistema de adquisición de datos---------- 12
Figura 3.- Organización centralizada del SAD---------- 13
Figura 4.- Organización descentralizada del SAD---------- 14
Figura 5.- Organización completamente descentralizada del SAD---------- 15
Figura 6.- Esquema del amplificador operacional ideal---------- 15
Figura 7.- Esquema del convertidor de corriente a voltaje---------- 16
Figura 8.- Esquema del circuito para sumar una desviación de voltaje------- 17
Figura 9.- Esquema del circuito sumador inversor diseñado para satisfacer la ecuación de una línea recta $y = mx + b$---------- 17
Figura 10.- Cuantificación de una señal analógica---------- 19
Figura 11.- Pines del microcontrolador PIC18F4550---------- 20
Figura 12.- Diagrama a bloques del microcontrolador PIC18F4550---------- 20
Figura 13.- Rango de tensiones de conversión del ADC---------- 22
Figura 14.- Transmisión de datos utilizando módems---------- 23
Figura 15.- Esquema de un LCD típico---------- 24
Figura 16.- Coordinador PAN con múltiples nodos---------- 25
Figura 17.- Conexiones mínimas requeridas para el Xbee---------- 26
Figura 18.- Estructura básica de un programa en CCS C---------- 27
Figura 19.- Pantalla de bienvenida de VS 2010---------- 27
Figura 20.- Arquitectura de la plataforma .NET framework---------- 30
Figura 21.- Diseño del sistema de adquisición de datos---------- 32
Figura 22.- Tarjeta SAD---------- 33
Figura 23.- CAS del sensor de temperatura---------- 35
Figura 24.- CAS del sensor de humedad relativa---------- 37
Figura 25.- contenedor de agua del sistema---------- 39
Figura 26.- CAS del sensor de presión diferencial---------- 40
Figura 27.- Esquema del circuito de la tarjeta SAD---------- 42
Figura 28.- PCB de la tarjeta CAS del Sensor de Temperatura---------- 43
Figura 29.- PCB de la tarjeta CAS del sensor de Humedad Relativa---------- 43
Figura 30.- PCB de la tarjeta CAS del sensor de PH---------- 43
Figura 31.- PCB de la tarjeta CAS del sensor de luminosidad---------- 43
Figura 32.- PCB de la tarjeta CAS del sensor de presión Diferencial para determinar nivel de agua---------- 43
Figura 33.- PCB de la tarjeta de adquisición de datos SAD---------- 44

ÍNDICE DE FIGURAS (cont.)

Pág.
Figura 34.- Diagrama de flujo del programa del PIC18F4550---------- 45
Figura 35.- Algoritmo de la base de datos---------- 47
Figura 36.- Controles de la forma principal---------- 48
Figura 37.- Controles de la forma de gráficas---------- 49
Figura 38.- Controles de la forma de configuración---------- 49
Figura 39.- Controles de la forma de nivel de agua---------- 50
Figura 40.- Fotografía de la tarjeta SAD---------- 51
Figura 41.- Tarjeta SAD activada---------- 51
Figura 42.- Sensores: temperatura, humedad relativa, luminosidad y presión diferencial---------- 51
Figura 43.- Radio transmisor XBEE Serie 1 Pro y tarjeta exploradora / programadora para XBEE dongle---------- 52
Figura 44.- Configuración de la aplicación HERVA V2.1---------- 52
Figura 45.- Visualización y almacenamiento de los datos en HERVA V2.1-- 53
Figura 46.- Gráfica de temperatura---------- 53
Figura 47.- Gráfica de humedad relativa---------- 54
Figura 48.- Almacenamiento de tabla de datos---------- 54
Figura 49.- HERVA V2.1 realizando un almacenamiento de datos---------- 55

ÍNDICE DE TABLAS

Pág.
Tabla 1.- Características del microcontrolador PIC18F4550---------- 19
Tabla 2.- Puertos del microcontrolador PIC18F4550---------- 21
Tabla 3.- Selección del canal de conversión del ADC---------- 21
Tabla 4.- Configuración del registro ADCON0---------- 22
Tabla 5.- Datos obtenidos en una tabla de HERVA V2.1---------- 55

INTRODUCCIÓN

El siguiente documento describe el s**istema de adquisición de datos para invernadero HERVA**, construido durante el periodo correspondiente a la Residencia Profesional de Agosto-Diciembre del 2011 en el Instituto Tecnológico de Tuxtla Gutiérrez.

Los sistemas de adquisición de datos surgen de la necesidad de monitorear algún proceso en donde se presenten variables que brindan información significativa durante el tiempo de actividad que conlleva tal proceso, muchas veces el problema de estos sistemas es que son costosos y en algunos casos, podrían no ajustarse a los requerimientos necesarios.

El sistema de adquisición de datos para invernadero **HERVA** es una herramienta muy útil para el registro de datos de forma rápida, precisa y de bajo costo, cuya finalidad es el monitoreo de las variables principales englobadas en los invernaderos de hoy en día para fines didácticos, científicos y hasta de comercialización. **HERVA** se compone de una tarjeta de adquisición **SAD** y una aplicación especial **HERVA V2.1** para las opciones de configuración, registro y almacenamiento de los datos desde distancias de hasta 300 m en interiores y 1.6 Km al aire libre.

La tarjeta de adquisición **SAD** se encuentra seccionada por 6 módulos: la tarjeta principal de control y visualización y 5 tarjetas de acondicionamiento de señal **CAS**, cuyas variables a registrar son: **temperatura, humedad relativa, ph, luminosidad y nivel de agua** para ofrecer un mejor confort al usuario al poder remover alguna tarjeta **CAS** sin afectar el funcionamiento de los demás módulos.

JUSTIFICACIÓN

Se decidió el diseño y construcción de un sistema de adquisición de datos para invernadero, debido a que es un proyecto que tiene un impacto social y económico muy fuerte, se sabe que existen muchos invernaderos que no cuentan con ningún sistema que mantenga las condiciones más favorables para sus cultivos, si esto se logrará, sería posible mejorar las condiciones de los cultivos y habrían menos perdidas y muchas más ganancias, aprovechando así todas las ventajas que un invernadero inteligente puede otorgar. Quizás uno de los inconvenientes con estos sistemas sea su precio, normalmente son costosos y no mucha gente esta dispuesta a pagar su precio sin alguna garantía.

El sistema de adquisición de datos es el primer paso de un sistema inteligente para invernaderos, logrando el monitoreo de forma eficaz y eficiente de las variables que se desean controlar y mantener estables en el invernadero, a un precio razonablemente menor que un sistema comercial, consiguiendo además un impacto tecnológico dentro de estos sistemas ya que su aplicación no solo recae en invernaderos sino en muchas otras aplicaciones donde se requiera el monitoreo de variables físicas, como temperatura, humedad relativa, PH, luminosidad, presión diferencial, etc. Con fines tanto de control, como en investigaciones, gracias a que todo el monitoreo se almacena en la base de datos.

El impacto ambiental de este proyecto es contribuir al desarrollo de flora en cultivos controlados, contribuyéndo a la construcción de más y mejores invernaderos.

OBJETIVOS

Objetivo general:

El sistema de adquisición de datos para invernadero, realizará el monitoreo de distintas variables físicas comunicándose inalámbricamente entre la tarjeta de adquisición de datos y la aplicación de la base de datos en una PC para el almacenamiento de los datos en tablas y gráficas que faciliten su análisis.

Objetivos Específicos:

- Diseño, programación y construcción de la tarjeta principal de control y visualización.

- Diseño y construcción de las 5 tarjetas de acondicionamiento de señal **CAS**.

- Diseño y programación de aplicación especial **HERVA V2.1** para las opciones de configuración, registro y almacenamiento de los datos.

CARACTERIZACIÓN DEL ÁREA EN QUE PARTICIPÓ.

Área Principal: Ingeniería Electrónica.

Áreas de conocimiento aplicadas:

- Electrónica analógica
- Electrónica digital
- Programación de microcontroladores
- Programación orientada a objetos y a eventos
- Mediciones eléctricas
- Instrumentación electrónica

El proyecto se llevó a cabo en el departamento de ingeniería electrónica, en el área de laboratorios del edificio "I" del Instituto Tecnológico de Tuxtla Gutiérrez, cuyo organigrama es el mostrado en la figura 1.

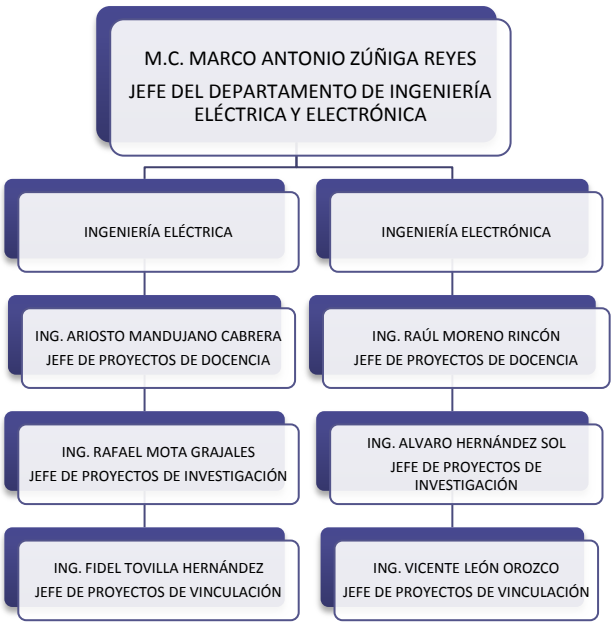


Figura 1.- Organigrama de la empresa en la que se participo

El proyecto "sistema de adquisición de datos para invernadero", surgió de la necesidad de realizar mediciones y monitorear variables como: temperatura, humedad relativa, ph, etc. en el invernadero ubicado en la División de Estudios de Posgrado, edificio Z, del Instituto Tecnológico de Tuxtla Gutiérrez. Hasta ahora se logró la construcción del prototipo, pero no se han realizado mediciones con el invernadero, ya que se contempla a futuro, además de añadirle la parte de control necesaria al sistema.

PROBLEMAS A RESOLVER

- Estandarización y acondicionamiento de las señales de los sensores dentro del rango de 0-5 V desde una distancia de 9 m de longitud.

- Envío de los datos, desde la tarjeta de adquisición SAD hacia la PC.

- Construcción de un programa visual que sea capaz de graficar y almacenar los datos provenientes de la tarjeta de adquisición SAD, además de ser amigable con el usuario.

- Lograr que el usuario realice un manejo fácil del sistema.

ALCANCES Y LIMITACIONES

El sistema de adquisición de datos para invernadero tiene un alto impacto en áreas donde se requiera monitorear variables como temperatura, humedad relativa, ph, etc. Gracias a que el sistema es capaz de mostrar las medidas de las variables tanto en forma local, como de forma remota en una base de datos, lo hace un sistema integral muy cómodo y fácil de manejar, que sin duda soluciona el problema de registro de datos a aquellos que se dedican a la investigación o que simplemente intentan mantener un control analítico sobre un proceso. Sus características hacen de él, no sólo un sistema de adquisición de datos de fines didácticos sino también de comercialización.

Por el momento, no se tiene un control sobre los procesos, a pesar de contar con entradas y salidas de propósito general, debido a que es necesario conocer el tipo de control que se desea aplicar sobre los procesos y conocer las características de los dispositivos con los que desea llevar a cabo la acción de control para manejar las etapas de potencia adecuadas. Por el momento el sistema se encuentra abierto para futuras modificaciones.

FUNDAMENTO TEÓRICO.

UNIDAD DE ADQUISICIÓN DE DATOS

Los sistemas de adquisición de datos o registradores de datos, recogen datos de sensores para su posterior procesamiento y análisis. Estos sistemas se utilizan en circunstancias en las que los sensores han recogido grandes cantidades de datos del entorno del sistema y no es necesario procesar los datos recopilados en tiempo real. Los sistemas de adquisición de datos se usan normalmente en experimentos científicos y sistemas de control de procesos en los que los procesos físicos, tales como una reacción química, ocurren muy rápidamente (Summerville. 2005).

Por ejemplo en la industria alimenticia la regulación de temperatura es un proceso muy común, en otras industrias se regula la presión de algún fluido, o en una estación meteorológica se registra el comportamiento en el tiempo de la velocidad del viento, su dirección, de la temperatura, etc. Al mismo tiempo que se hace más grande la tendencia de medición (adquisición) y/o manipulación de procesos físicos, también cada vez es necesario el uso de procesadores digitales, que son los que se encargan de recibir datos de los sensores, enviar los datos de manera local o a distancia hacia una estación de control o CPU, y hasta ser la encargada del mando de alguna acción de control. Sin embargo, la mayoría de los transductores o sensores disponibles para la medición de variables físicas únicamente ofrecen una salida analógica, ya sea en forma de tensión o corriente, que a al mismo tiempo sea proporcional a la variable que se mide. Por tanto para llevar esta información a una base de datos es necesario interpretar todas estas variables que están en el dominio analógico y que por consiguiente tiene variaciones continuas en su señal, hacia un dominio digital que es el usado por los procesadores y en el cual la señal será representada por valores numéricos o discretos (Calleja Gjumlich, 1998).

Los sistemas de adquisición de datos pueden tomar entradas desde un gran número de fuentes, realizar ciertas funciones matemáticas en las entradas y proceder al almacenamiento de los datos en una memoria de estado sólido o un sistema magnético de disco o cinta. Un registrador de datos consiste esencialmente en un multiplexor, un elemento de muestreo y retención, un convertidor analógico/ digital y algún sistema de registro o manipulación y registro de salida, como se muestra en la figura 2, las señales de entrada (después de los adecuados acondicionamientos de señal) se alimentan al multiplexor, este selecciona una señal, y el elemento de muestreo y retención toma una muestra de la señal y la mantiene tanto tiempo como necesite la conversión analógico/digital para realizar la transformación sin errores debido a fluctuaciones de entrada. La salida de la unidad es por tanto, una señal digital. El multiplexor puede conmutar cada señal de entrada y, por tanto, obtener la salida digital de cada una de ellas. Estas salidas alimentan, a menudo, a un ordenador que puede no sólo almacenar los datos sino que también los puede procesar.

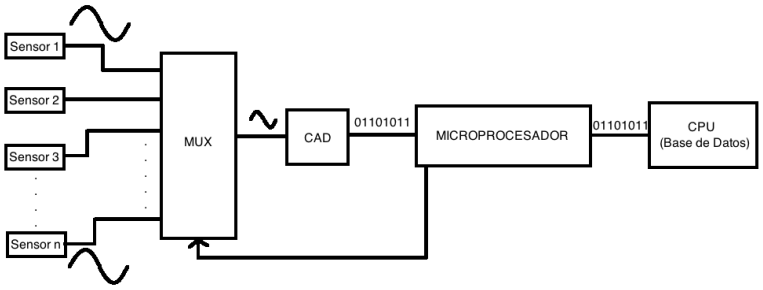


Figura 2.- Estructura básica de un sistema de adquisición de datos

Típicamente un sistema de adquisición de datos puede tener de 20 a 100 entradas, aunque conectándolo a un ordenador se pueden manejar muchas más. Puede tener un tiempo de muestreo y conversión de unos 10 µs y una rapidez de respuesta de 0.5 V/s. la *rapidez de respuesta* es la máxima velocidad de cambio de la tensión de entrada que puede seguirse. La precisión es, típicamente de ±0.005% de la entrada a fondo de escala. El término *diafonía* se utiliza para escribir la interferencia que puede suceder con la entrada muestreada como resultado de la existencia de otras entradas de la señal (Bolton. 1995).

UNIDAD DE ADQUISICIÓN DE DATOS

Es muy común que una unidad de adquisición de datos no funcione de manera adecuada, a pesar de que los circuitos integrados con que se construyo son de la mejor calidad. Al examinarla, no se encuentra algún dispositivo defectuoso; no obstante, en la salida digital se obtienen códigos que varían impredeciblemente, aún para las señales de entradas constantes (Calleja Gjumlich, 1998).

Esto suele ocurrir cuando se desprecian los aspectos asociados con la construcción de la unidad: la organización que deberá tener, los elementos pasivos que se utilicen, la construcción física del impreso, el ambiente en el que debe operar, las trayectorias que siguen los conductores con los que se aplican las entradas al circuito; etc. Todos estos aspectos tienen en común el ser parte integral de la unidad de adquisición de datos, de manera que es imposible tener un comportamiento correcto si no se les considera en las etapas de planeación y diseño.

En lo que respecta a la organización de la unidad de adquisición de datos, se tienen los siguientes tipos:

ORGANIZACIÓN CENTRALIZADA

Existen comercialmente unidades de adquisición de datos de propósito múltiple, que pueden funcionar conjuntamente con una computadora personal; por lo general su estructura es como la ilustrada en la figura 3.

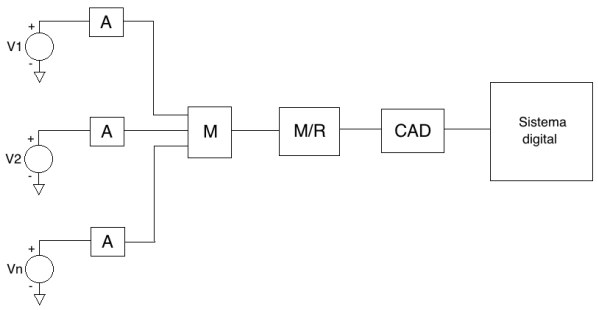


Figura 3.- Organización centralizada del SAD.

De donde “A” corresponde al acondicionamiento de la señal, lo que incluye tanto la función de amplificación como de filtrado; el bloque “M” corresponde al multiplexor; M/R es el circuito de muestreo y retención y CAD es el convertidor analógico-digital. El razonamiento que justifica esta organización es principalmente económico, ya que usualmente el convertidor es el componente más caro de la unidad. Por tanto en aplicaciones en donde se maneja un número reducido de señales y que además tienen frecuencias bajas, la organización centralizada es adecuada.

Por tanto la ecuación (1) sirve para calcular la frecuencia de conversión en esta configuración.

$$f_{CC} = \frac{1}{n(t_{CM} + t_{AMR} + t_{CONV})} \qquad Ec.(1)$$

Donde:

- f_{CC} es la frecuencia máxima de cuantificación.
- t_{CM} es el tiempo de conmutación del multiplexor.
- t_{AMR} es el tiempo de adquisición en el circuito de retención y muestreo.
- t_{CONV} el tiempo de conversión del convertidor A-D.
- n el número de canales del multiplexor.

Una vez que se ha tomado una muestra en el circuito de muestreo y retención, es posible adelantar la conmutación del multiplexor, a fin que la señal a su salida ya este estable cuando termine la conversión; cuando se sigue un procedimiento de este tipo puede alcanzarse la frecuencia máxima de cuantificación por canal, esto mediante el uso de la ecuación (2).

$$t_{CCM} = \frac{1}{n(t_{AMR} + t_{CONV})} \qquad Ec.(2)$$

ORGANIZACIÓN PARCIALMENTE DESCENTRALIZADA

Una opción para aumentar la frecuencia de conversión sin encarecer excesivamente la unidad, consiste en el empleo de la organización parcialmente descentralizada como se muestra en la figura 4.

A cada señal analógica se le asocia un circuito de muestreo y retención definido por la ecuación 3.

$$f_{CC} = \frac{1}{t_{AMR} + n(t_{CONV} + t_{CM})} \qquad Ec.(3)$$

Por lo general se cumple que el tiempo de conmutación del multiplexor es muy inferior a los otros lapsos por lo que puede plantearse la en simplificación de la ecuación 4.

$$f_{CC} \approx \frac{1}{t_{AMR} + nt_{CONV}} \qquad Ec.(4)$$

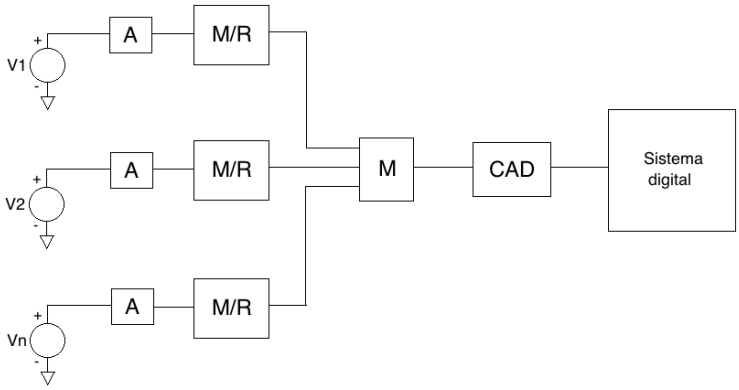


Figura 4.- Organización descentralizada del SAD.

ORGANIZACIÓN COMPLETAMENTE DESCENTRALIZADA

Esta arquitectura mostrada en la figura 5, incluye un convertidor por cada canal analógico de entrada; esto permite obtener frecuencias de conversión muy altas, y se obtiene la ecuación 5.

$$f_{CC} = \frac{1}{t_{AMR} + t_{CONV}} \qquad Ec.(5)$$

En ocasiones es muy factible hacer uso de una versión hibrida de las distintas organizaciones, en especial cuando se tienen señales cuyas frecuencias difieren mucho entre sí.

Desde el punto de vista de la unidad propiamente dicha, es conveniente seguir el flujo natural de las señales, es decir, empezando con las fuentes de señal, continuando con las etapas de acondicionamiento, de multiplexor, de muestreo y retención, de conversión, para terminar con el sistema digital.

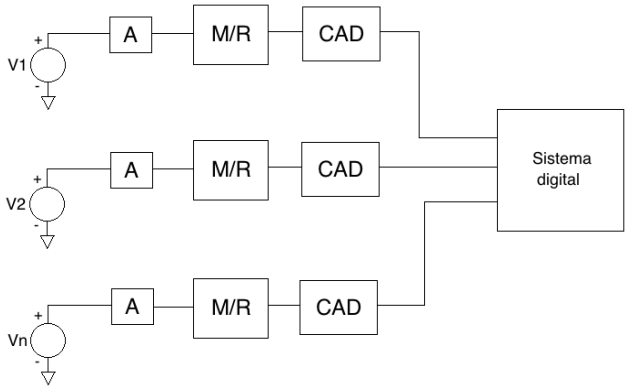


Figura 5.- Organización completamente descentralizada del SAD.

ACONDICIONAMIENTO DE LA SEÑAL

Como bien sabemos todos los sensores Analógicos proporcionan una señal en forma de tensión o corriente proporcional a la variable que se mide, aunque hay que tomar en cuenta que en su gran mayoría, estas señales son relativamente pequeñas, en la escala de los mV o los mA por lo tanto es necesario antes que nada, una etapa que amplifique esta señal. La participación de los Amplificadores Operaciones en esta etapa es primordial, ya bien sea por el hecho de que se desea amplificar la señal o también se requiera el uso de algún filtro que la señal de salida no contenga ruido a otras frecuencias que afecten el funcionamiento general del sistema de Adquisición, en caso de usar algún Amplificador Operacional en configuración de seguidor como Buffer, serviría para eliminar efectos de carga, lo cual hace más estable la salida con menos variaciones y además al acoplamiento de impedancias entre los dispositivos en la entrada y salida del mismo amplificador; Por tanto los amplificadores operacionales pueden servir para muchas cosas a la vez (Calleja Gjumlich, 1998).

Para la parte de amplificación de la señal el Amplificador Operacional es ideal ya que su función es amplificar la señal dependiendo su ganancia la cual puede ser regulada por medio de resistencias para entender esto se tiene la siguiente configuración.

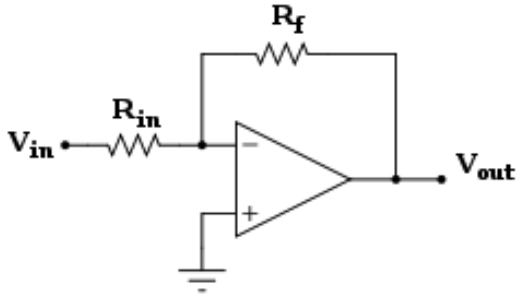


Figura 6.- Esquema del amplificador operacional ideal

Corresponde a un Amplificador Operacional en forma de Inversor, su ganancia se obtiene con las resistencias R_f y R_{in}, el producto de la división se muestra en la ecuación (6) y el voltaje en la salida se muestra en la ecuación 7.

$$\Delta_V = \frac{R_f}{R_{in}} \qquad Ec.(6)$$

$$V_{OUT} = V_{IN} \cdot \Delta_V \qquad Ec.(7)$$

En un amplificador operacional ideal se tiende a pensar en una ganancia infinita, lo cual en el términos ideales es posible pero en la realidad, la ganancia máxima posible va a depender de ciertas características de cada amplificador operacional, ya que para empezar un amplificador operacional no resulta una salida de voltaje mayor al voltaje con el cual el Amplificador se alimenta, si se intenta superar esto la señal de salida está saturada, esto no quiere decir que no sirva de nada, de hecho hay aplicaciones en donde esta característica puede servir por ejemplo, en circuitos comparadores.

Así como se tiene esta configuración de Amplificador Operacional Inversor existen otras configuraciones, por mencionar algunas: Comparador, No inversor, Seguidor, Sumador, Restador, Derivador, Integrador, etc.

Hasta ahora se ha visto cómo Amplificar la señal de voltaje, pero en cuanto a corriente se tiene que hacer uso de un convertidor de Corriente a voltaje, y de ahí mismo obtener la relación de estas dos, para poder seguir manteniendo la proporción con la variable a medir. Teniendo esto en voltaje, ya se procede de la misma manera. La configuración de dicho conversión se ilustra en la figura 7.

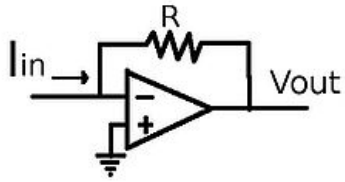


Figura 7.- Esquema del convertidor de corriente a voltaje

El convertidor de corriente a voltaje, se conoce también como Amplificador de trans-impedancia, llegada a este una corriente (I_{in}), la transforma en un voltaje proporcional a esta, con una impedancia de entrada muy baja, ya que está diseñado para trabajar con una fuente de corriente. Con el resistor R como factor de proporcionalidad, la relación resultante entre la corriente de entrada y el voltaje de salida es $V_{out} = -R*I_{in}$, esta es una forma de convertir la corriente a tensión, aunque existen otros métodos, usando algo que se llama “Transformador de corriente”, cuyo devanado primario se inserta en serie con el circuito eléctrico, en este caso la salida de algún sensor que proporcione corriente, con esta configuración en el devanado secundario se obtiene una reproducción escalada de corriente de primario; esta corriente de secundario puede ahora convertirse a tensión por medio de una resistencia.

Se podría usar un resistencia para convertir la corriente en tensión la única desventaja es que no ofrece aislamiento.

El circuito de la figura 8 se puede considerar como un medio para diseñar un circuito acondicionador de señal (CAS), útil en una aplicación de microcontrolador y que se comporta de acuerdo con la ecuación de una línea recta, $y = mx + b$. Esta ecuación se presenta con frecuencia cuando se diseñan los CAS. Compare la ecuación $y = mx + b$ con el circuito de la figura 9 y corresponde al voltaje de salida, V_o, x es el voltaje de la señal de entrada, E_i, m corresponde a la ganancia del circuito, R_f/R_i, y b es R_s/R_2 veces E_{cd} (Coughlin, 1999).

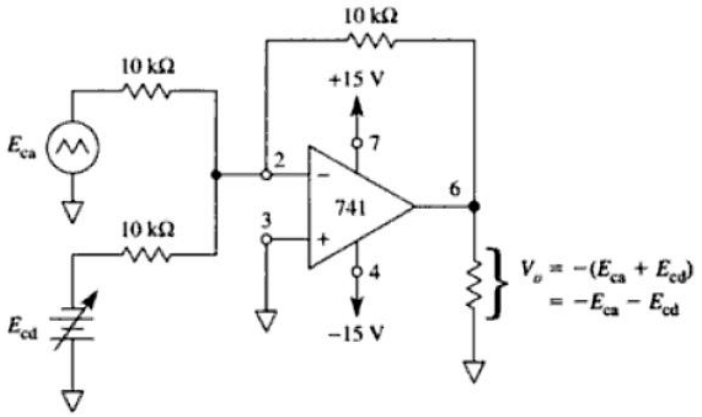


Figura 8.- Esquema del circuito para sumar una desviación de voltaje

Por lo tanto, si en la aplicación que le interesa a usted se utiliza un sensor que produce una señal de entrada, medida respecto a la tierra, y que hay que amplificar y desviar, entonces lo que se puede utilizar es una CAS similar al de la figura 9. (*Nota:* en caso de algunos sensores se genera una salida diferencial, por lo que para estos dispositivos se necesita un CAS capaz de medir voltajes diferenciales). Para diseñar una unidad de CAS es necesario obtener la ecuación del circuito. Esta ecuación se obtiene tomando en cuenta lo que usted *recibe*, las condiciones de salida del sensor, y después transformando lo anterior en lo que usted *desea*, es decir, las condiciones de entada del convertidor A/D del microcontrolador.

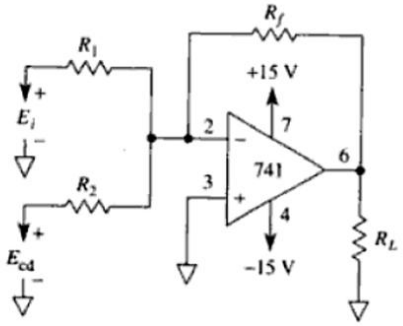


Figura 9.- Esquema del circuito sumador inversor diseñado para satisfacer la ecuación de una línea recta *y = mx + b*

CANALIZACIÓN DE LA SEÑAL

En un sistema de adquisición la parte que tiene que ver con canalización tiene que ver con arreglos de interruptores conectados con sus terminales de salida hacia un nodo en común, mejor como conocido como Multiplexor, este es muy útil en aplicaciones en donde tenemos varias señales y deben canalizarse secuencialmente hacia una sola etapa de procesamiento. Además de los interruptores, un multiplexor debe incluir un bloque de lógica que sirve para poder seleccionar y activar un solo interruptor a la vez, este bloque se representa con algún código digital, con esto se permite reducir el número de entradas de mando en multiplexores con un número elevado de canales. El hecho de que los interruptores que conforman al multiplexor compartan un nodo en común introduce fuentes de error considerables cuando se manejan señales de bajo nivel (Calleja Gjumlich, 1998).

La operación dinámica que rige el funcionamiento de un multiplexor a su vez especifica que en cuanto a la apertura de un canal y el cierre de otro canal son tareas que se accionan simultáneamente por el circuito digital de control del multiplexor, por tanto si el tiempo en el que se abre y se cierra un interruptor fueran iguales o muy cercanos, podría haber la posibilidad de generar un corto circuito entre las fuentes conectadas a los interruptores involucrados, en el caso de un sistema de adquisición de datos, que las señales de sensores se cruzasen entre sí, dando como resultado una señal desconocida. Para evitar que suceda esto, los multiplexores se construyen de manera tal que, al conmutarse de un canal cualquiera a otro, la apertura del interruptor previamente encendido anteceda al cierre del nuevo interruptor, este comportamiento se denomina, “romper antes de cerrar”(“Break-Before-Make”).

MUESTREO DE SEÑALES

En la operación de sistemas basados en la toma de muestras de señales analógicas deben considerarse dos aspectos básicos relacionados con la frecuencia de la señal que se cuantifica: la frecuencia a la cual se realiza esta cuantificación, y la velocidad de conversión del dispositivo que se use (Calleja Gjumlich, 1998).

El primer aspecto está relacionado con el teorema del muestreo, que especifica cuántas veces por ciclo es necesario cuantificar o muestrear una señal, de manera que sea posible reconstruirla a partir de las muestras.

Cuando se muestrea una señal analógica, la información que se puede extraer de las muestras depende de la relación que existe entre la frecuencia de la señal analógica y la frecuencia de muestreo, es decir, del espaciamiento entre las muestras. Por este mismo motivo es preferente que nuestra señal de muestreo tenga una frecuencia más elevada que la frecuencia de la señal analógica, ya que

de lo contrario se corre el riesgo de mal interpretar los resultados, ya sea con la aparición de algún “alias” o muestras que a simple vista no nos dicen nada acerca de la variable a medir.

Para cuantificar una señal analógica es necesario muestrearla, lo que equivale a capturar valores instantáneos de la señal a intervalos regulares, para hacer esto, se toma la señal de entrada analógica V_i y se le multiplica por un tren de pulsos, obteniendo así una salida V_o, que son los valores instantáneos que la señal analógica fue tomando en cada pulso, tal y como se muestra en la figura 10.

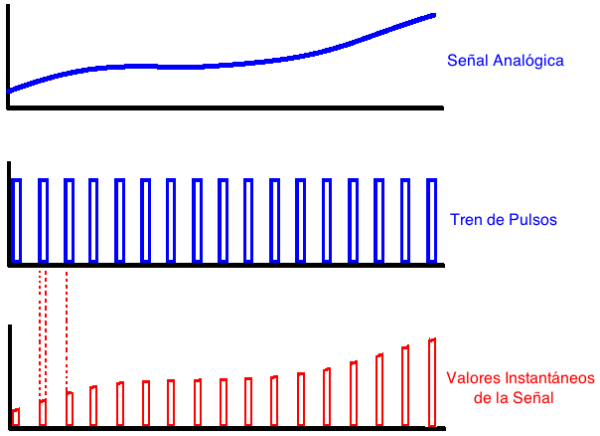


Figura 10.- Cuantificación de una señal analógica.

MICROCONTROLADOR PIC18F4550

Éste microcontrolador de 8 bits, cuenta con gran variedad de número de pines y prestaciones medias/altas y sus características fundamentales se muestran en la tabla 1 (Microchip, 2006).

Carácterísticas	PIC18F4550
Frecuencia de operación	Hasta 48MHz
Memoria de Programa (bytes)	32 768
Memoria RAM de datos (bytes)	2 048
Memoria EEPROM de datos (bytes)	256
Interrupciones	20
Líneas de E/S	35
Temporizadores	4
Módulos de Comparación/Captura/PWM (CCP)	1
Módulos de Comparación/Captura/PWM mejorado (ECCP)	1
Canales de Comunicación Serie	MSSP, EUSART
Canal USB	1
Puerto Paralelo de Transmisión de Datos (SPP)	1
Canales de Conversión A/D de 10 bits	13 Canales
Comparadores analógicos	2
Juego de instrucciones	75 (83 ext.)
Encapsulado utilizado	PDIP 40 pines

Tabla 1.- Características del microcontrolador PIC18F4550

Los pines del microcontrolador PIC18F4550 se pueden observar en la figura 11.

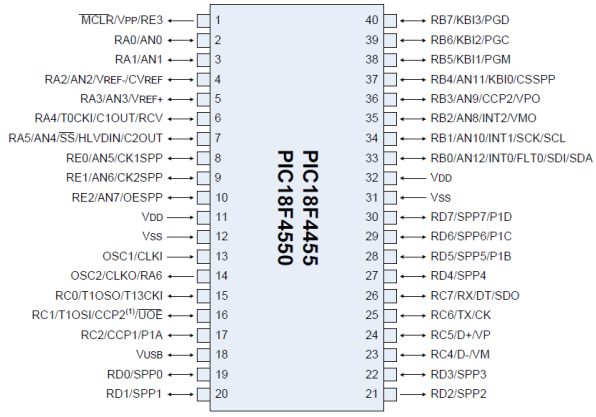


Figura 11.- Pines del microcontrolador PIC18F4550

El diagrama a bloques del microcontrolador PIC18F4550 es el mostrado en la figura 12.

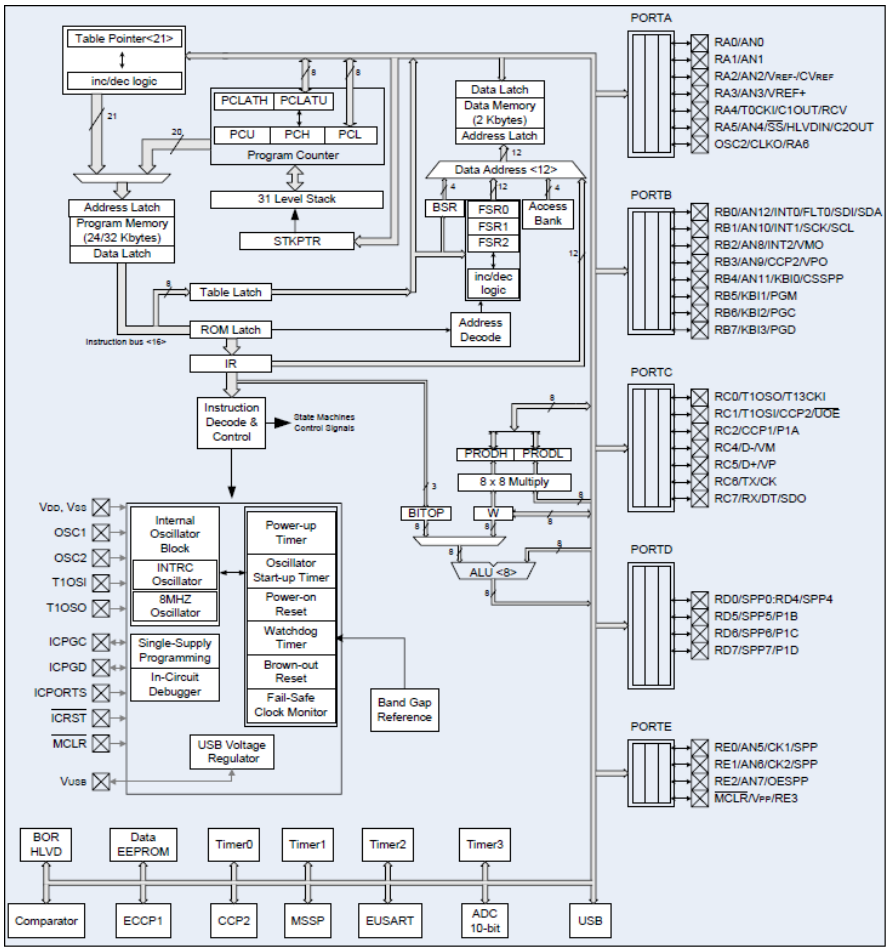


Figura 12.- Diagrama a bloques del microcontrolador PIC18F4550

El µC PIC18F4550 dispone 5 puertos de Entrada/Salida (E/S) que incluyen un total de 35 líneas digitales de E/S, mostrados en la tabla 2.

Puerto	Líneas de Entrada/Salida
PORTA	7 Líneas de Entrada/Salida
PORTB	8 Líneas de Entrada/Salida
PORTC	6 Líneas de Entrada/Salida + 2 Líneas de entrada
PORTD	8 Líneas de Entrada/Salida
PORTE	3 Líneas de Entrada/Salida + 1 Línea de Entrada

Tabla 2.- Puertos del microcontrolador PIC18F4550

Todas las líneas digitales de E/S disponen de al menos una función alternativa asociada a alguna circuitería específica del µC. cuando una línea trabaja en el modo alternativo no puede ser utilizada como línea digital de E/S estándar.

El convertidor Analógico-Digital del microcontrolador PIC18F4550 consta de las siguientes características:

- 10 bits de resolución.
- 13 Canales multiplexados.
- Señal de reloj de conversión configurable.
- Tiempo de adquisición programable (0 a $20T_{AD}$).
- Posibilidad de establecer el rango de tensiones de conversión mediante tensiones de referencia externas.

Para que uno de los 13 canales pueda ser seleccionado, previamente debe haber sido configurado como entrada analógica mediante los bits PCFG3...PCFG0 del registro ADCON1 (A: analógico /D: digital), la selección del canal de conversión se muestra en la tabla 3.

PCFG3..PCFG0	AN12	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
0000	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

Tabla 3.- Selección del canal de conversión del ADC.

Una vez configurado como línea de entrada analógica, un canal puede ser seleccionado mediante los bits CHS3..CHS0 del registro ADCON0, como se muestra en la tabla 4.

CHS3	CHS2	CHS1	CHS0	CANAL SELECCIONADO
0	0	0	0	CANAL AN0 (RA0)
0	0	0	1	CANAL AN1 (RA1)
0	0	1	0	CANAL AN2 (RA2)
0	0	1	1	CANAL AN3 (RA3)
0	1	0	0	CANAL AN4 (RA5)
0	1	0	1	CANAL AN5 (RE0)
0	1	1	0	CANAL AN6 (RE1)
0	1	1	1	CANAL AN7 (RE2)
1	0	0	0	CANAL AN8 (RB2)
1	0	0	1	CANAL AN9 (RB3)
1	0	1	0	CANAL AN10 (RB1)
1	0	1	1	CANAL AN11 (RB4)
1	1	0	0	CANAL AN12 (RB0)
1	1	0	1	No implementado
1	1	1	0	No implementado
1	1	1	1	No implementado

Tabla 4.- Configuración del registro ADCON0.

Por defecto el rango de tensiones de conversión del convertidor A/D del PIC18F4550 es de 0V a 5V., sin embargo, en ocasiones puede resultar interesante modificar este rango para aumentar la resolución de la conversión acercando las tensiones de referencia máxima y mínima $_{VREF+}$ y V_{REF-} a los límites de variación de la señal que se desea digitalizar, como se ilustra en la figura 13.

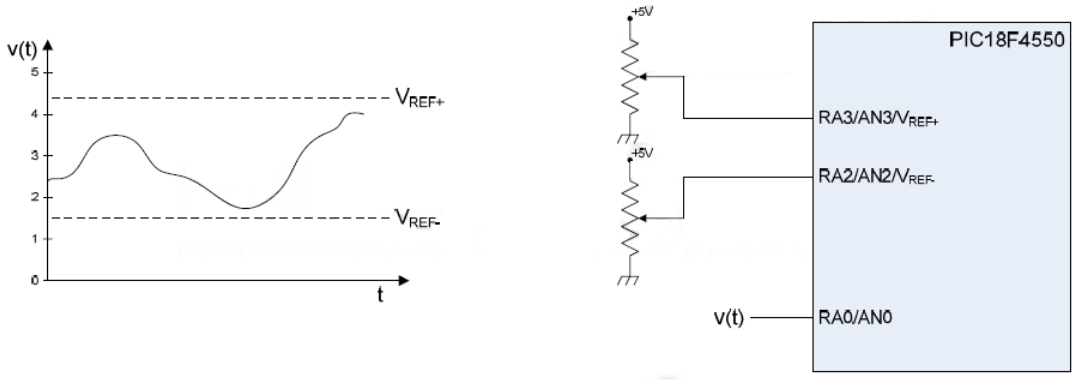


Figura 13.- Rango de tensiones de conversión del ADC.

Esto se puede conseguir configurando las líneas RA2/AN2/V_{REF-} y RA3/AN3/V_{REF+} como tensiones de referencia de convertidor A/D (poniendo a '1' los bits VCFG1 y VCFG0 del registro ADCON1). De esta forma el rango de tensiones de conversión vendrá determinado por las tensiones que se conecten en dichas líneas.

Las señales de un transductor tienen que ser transmitidas al punto de medida. Esta transmisión puede ser una pequeña distancia o a una distancia donde a través de una interfaz RS232C al receptor de datos, es decir, el lugar donde se almacenen los datos, como se muestra en la figura 14 (Bolton. 1995).

Figura 14.- Transmisión de datos utilizando módems.

El microcontrolador PIC18F4550 dispone del módulo de comunicación serie síncrono asíncrono universal (USART), para la transmisión o recepción de datos en serie. Esta operación puede dividirse en dos categorías síncrona o asíncrona. La transmisión síncrona utiliza una señal de reloj y una línea de datos, mientras que en la transmisión asíncrona no se envía la señal de reloj, por lo que el emisor y el receptor deben tener relojes con la misma frecuencia y fase. Cuando la distancia entre el emisor y el receptor es pequeña se suele utilizar la transmisión síncrona, mientras que para instancias mayores se utilizará la transmisión asíncrona (García Breijo. 2008).

El *USART* puede transmitir o recibir datos serie. Puede transferir tramas de datos de 8 o 9 bits por transmisión y detectar errores de transmisión. También puede generar interrupciones cuando se produce una recepción de datos o cuando la transmisión ha sido completada.

El microcontrolador PIC18F4550, tiene las siguientes características de su canal de comunicación serie EUSART:

- Modo de trabajo
    - Modo asíncrono de 8 bits
    - Modo asíncrono de 9 bits
    - Modo síncrono Maestro
    - Modo síncrono Esclavo
- Auto-activación por detección de dato recibido
- Detección automática de velocidad de comunicación (baudrate).
- Transmisión y detección de carácter de BREAK (bus LIN).

Para configurar de las líneas TX y RX para el modo asíncrono, las líneas RC6/TX y RC7/RX deben configurarse adecuadamente para que puedan funcionar como las líneas de transmisión y recepción respectivamente.

- Poner a ‘1’ el bit SPEN (RCSTA).
- Poner a ‘1’ el bit 7 del resgistro TRISC (línea RC7/RX configurada como entrada).
- Poner a ‘0’ el bit 6 del registro TRISC (línea RC6/TX configurada como salida).

Si se recibe un dato por el canal EUSART se pone a ‘1’ el flag RCIF (bit de interrupción). Si el bit de habilitación RCIE (bit de habilitación) esta a ‘1’ y las interrupciones estan habilitadas a nivel global se genera una interrupción y el µC pasa a ejecutar el código situado a partir de la posición 0008H (según el nivel de prioridad establecido).

LCD

Se acostumbran a utilizar *LCD* del tipo *HD44780*, con un número de líneas variable y un numero de caracteres por línea también variable, como la que se muestra en la figura 15 (por ejemplo 2 X16 se trabaja con dos líneas de 16 caracteres cada una).

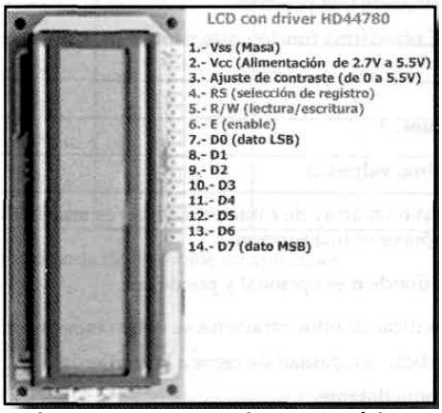


Figura 15- Esquema de un LCD típico.

El bus de datos es de 8 bits, aunque también existe la posibilidad de trabajar con 4 bits (con un menor número de caracteres).

RADIOS XBEE

Zigbee es un protocolo de comunicaciones, inalámbrica basado en el estándar de comunicaciones para redes inalámbricas IEEE 802.15.4 creado por Zigbee Alliance, una organización teóricamente sin ánimo de lucro, de más de 200 grandes empresas, muchas de ellas fabricantes de semiconductores (Oyarce. 2008).

Zigbee permite que dispositivos electrónicos de bajo consumo puedan realizar sus comunicaciones inalámbricas. Es especialmente útil para redes de sensores en entornos industriales, médicos y, sobre todo, domóticos.

Las comunicaciones Zigbee se realizan en la banda libre de 2.4GHz. A diferencia de bluetooth, este protocolo no utiliza FHSS (Frecuency hooping), sino que realiza las comunicaciones a través de una única frecuencia, es decir, de un canal. Normalmente puede escogerse un canal de entre 16 posibles. El alcance depende de la potencia de transmisión del dispositivo así como también del tipo de antenas utilizadas (cerámicas, dipolos, etc.). Una red Zigbee la pueden formar, teóricamente hasta 65635 equipos.

Entre las necesidades que satisface el módulo se encuentran

- Bajo costo
- Ultra bajo consumo de potencia
- Uso de bandas de radio libres y sin necesidad de licencias
- Instalación barata y simple.
- Redes flexibles y extensibles.

Cada módulo Zigbee, al igual que ocurre con las direcciones MAC de los dispositivos Ethernet, tiene una dirección única. En el caso de los módulos Zigbee cada uno de ellos tiene una dirección única de 64 bits que viene grabada de fábrica. Por otro lado, la red Zigbee, utiliza para sus algoritmos de ruteo direcciones de 16 bits. Cada vez que un dispositivo se asocia a una red Zigbee, el Coordinador al cual se asocia le asigna una dirección única en toda la red de 16 bits.

Éstos módulos Xbee pueden ser ajustados para usarse en redes de configuración punto a punto, punto a multipunto o peer to peer. Un ejemplo se muestra en la figura 16, donde se muestra una conexión multipunto, con un coordinador, conectado a varios nodos.

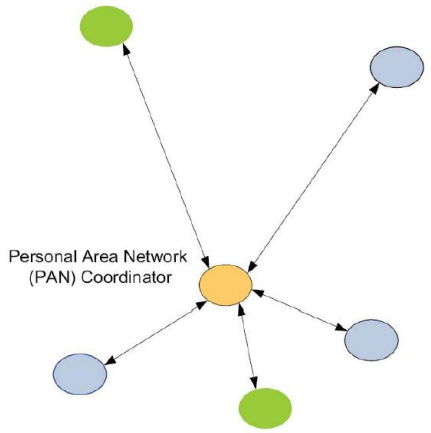


Figura 16.- Coordinador PAN con múltiples nodos.

El circuito básico para el Xbee la mostrada en la figura 17.

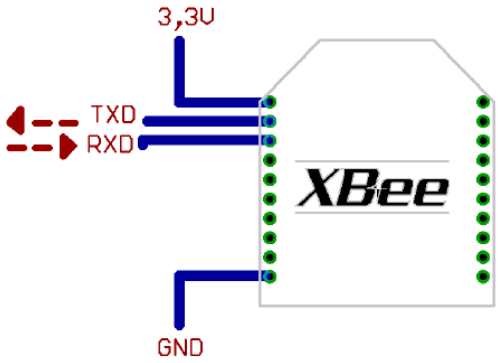


Figura 17.- Conexiones mínimas requeridas para el Xbee.

El módulo requiere una alimentación desde 2.8 a 3.4 V, la conexión a tierra y las líneas de transmisión de datos por medio del *UART* (*TXD* y *RXD*) para comunicarse con un microcontrolador, o directamente a un puerto serial utilizando algún módulo convertidor adecuado para los niveles de voltaje.

COMPILADOR CCS C

El *compilador c* de *ccs* ha sido desarrollado específicamente para PIC MCU, obteniendo la máxima optimización del compilador con estos dispositivos. Dispone de una amplia librería de funciones predefinidas, comandos de preprocesador y ejemplos, Además, suministra los controladores (*drivers*) para diversos dispositivos como LCD, convertidores AD, relojes en tiempo real, EEPROM serie, etc., (García Breijo. 2008).

Un compilador convierte el lenguaje de alto nivel a instrucciones de código máquina; un *cross-compiler* es un compilador que funciona en un procesador (normalmente en un PC) diferente al procesador objeto. El compilador *ccs c* es un *cross-compiler*. Los programas son editados y compilados a instrucciones máquina en el entorno de trabajo del PC, el código máquina puede ser cargado del PC al sistema PIC mediante el ICD2 (o cualquier programador) y puede ser depurado desde el entorno de trabajo del PC.

El *CCS C* es *C* estándar y, además de las directivas estándar (*#include*, etc.), suministra unas directivas específicas para PIC (*#device*, etc.); además incluye funciones específicas (*bit_set()*, etc.). Se suministra con un editor que permite controlar la sintaxis del programa.

Para escribir un programa en *C* con el *CCS C* se deben tener en cuenta una serie de elementos básicos de su estructura como se muestra en la figura 18.

- Directivas de preprocesador: controlan la conversión del programa a código máquina por parte del compilador.

- Programas o funciones: conjunto de instrucciones. Puede haber uno o varios; en cualquier caso siempre debe haber uno definido como principal mediante la inclusión de la llama *main()*.
- Instrucciones: indican cómo se debe comportar el PIC en todo momento.
- Comentarios: permiten describir lo que significa cada línea del programa.

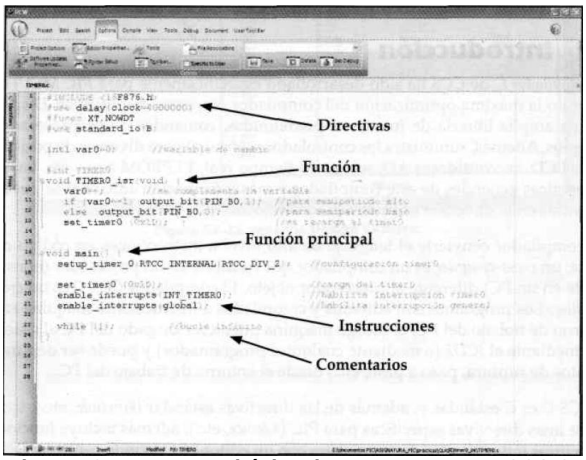


Figura 18.- Estructura básica de un programa en CCS C.

IDE VISUAL STUDIO 2010

La gama de productos de Visual Studio comparte un único entorno de desarrollo integrado (IDE), como se aprecia en la figura 19, que se compone de varios elementos: la barra de menús, la barra de herramientas Estándar, varias ventanas de herramientas que se acoplan u ocultan automáticamente a la izquierda, en la parte inferior y a la derecha, así como en el espacio del editor. Las ventanas de herramientas, menús y barras de herramientas disponibles dependen del tipo de proyecto o archivo en el que esté trabajando (Ruíz. 2010).

Figura 19.- Pantalla de bienvenida de VS 2010.

LENGUAJE C#

C# es un lenguaje orientado a objetos elegante y con seguridad de tipos que permite a los desarrolladores compilar diversas aplicaciones sólidas y seguras que se ejecutan en .NET Framework.Se puede utilizar C# para crear aplicaciones cliente de Windows tradicionales, servicios Web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de base de datos, y mucho, mucho más. Visual C# 2010 proporciona un editor de código avanzado, cómodos diseñadores de interfaz de usuario, depurador integrado y numerosas herramientas más para facilitar el desarrollo de aplicaciones basadas en la versión 4.0 del lenguaje C# y la versión 4 de .NET Framework.

La sintaxis de C# es muy expresiva, pero también es sencilla y fácil de aprender. La sintaxis de C# basada en signos de llave podrá ser reconocida inmediatamente por cualquier persona familiarizada con C, C++ o Java. Los desarrolladores que conocen cualquiera de estos lenguajes pueden empezar a trabajar de forma productiva en C# en un plazo muy breve. La sintaxis de C# simplifica muchas de las complejidades de C++ y proporciona características eficaces tales como tipos de valor que admiten valores NULL, enumeraciones, delegados, expresiones lambda y acceso directo a memoria, que no se encuentran en Java. C# admite métodos y tipos genéricos, que proporcionan mayor rendimiento y seguridad de tipos, e iteraciones, que permiten a los implementadores de clases de colección definir comportamientos de iteración personalizados que el código cliente puede utilizar fácilmente. Las expresiones Language-Integrated Query (LINQ) convierten la consulta en una construcción de lenguaje de primera clase.

Como lenguaje orientado a objetos, C# admite los conceptos de encapsulación, herencia y polimorfismo. Todas las variables y métodos, incluido el método Main que es el punto de entrada de la aplicación, se encapsulan dentro de definiciones de clase. Una clase puede heredar directamente de una clase primaria, pero puede implementar cualquier número de interfaces. Los métodos que reemplazan a los métodos virtuales en una clase primaria requieren la palabra clave override como medio para evitar redefiniciones accidentales. En C#, una struct es como una clase sencilla; es un tipo asignado en la pila que puede implementar interfaces pero que no admite la herencia.

Además de estos principios básicos orientados a objetos, C# facilita el desarrollo de componentes de software a través de varias construcciones de lenguaje innovadoras, entre las que se incluyen las siguientes:

- Firmas de métodos encapsulados denominadas *delegados*, que habilitan notificaciones de eventos con seguridad de tipos.

- Propiedades, que actúan como descriptores de acceso para variables miembro privadas.

- Atributos, que proporcionan metadatos declarativos sobre tipos en tiempo de ejecución.

- Comentarios en línea de documentación XML.

- Language-Integrated Query (LINQ) que proporciona funciones de consulta integradas en una gran variedad de orígenes de datos.

Si necesita interactuar con otro software de Windows, como objetos COM o archivos DLL nativos de Win32, podrá hacerlo en C# mediante un proceso denominado "interoperabilidad". La interoperabilidad habilita los programas de C# para que puedan realizar prácticamente las mismas tareas que una aplicación C++ nativa. C# admite incluso el uso de punteros y el concepto de código "no seguro" en los casos en que el acceso directo a la memoria es totalmente crítico.

El proceso de compilación de C# es simple en comparación con el de C y C++, y es más flexible que en Java. No hay archivos de encabezado independientes, ni se requiere que los métodos y los tipos se declaren en un orden determinado. Un archivo de código fuente de C# puede definir cualquier número de clases, structs, interfaces y eventos.

A manera de resumen, C# cubre las siguientes características:

- C# es un lenguaje moderno y orientado a objetos, con una sintaxis muy similar a la de C++ y Java. Combina la alta productividad de Visual Basic con el poder y la flexibilidad de C++.

- La misma aplicación que se ejecuta bajo Windows podría funcionar en un dispositivo móvil de tipo PDA con C#.NET no nos atamos a ninguna plataforma en particular.

- Se puede crear una gran variedad de aplicaciones en C#, aplicaciones de consola, aplicaciones para Windows con ventanas y controles, aplicaciones para la Web, etc.

- C# gestiona automáticamente la memoria, y de este modo evita los problemas de programación tan típicos en lenguajes como C o C++.

- Mediante la plataforma .NET desde la cual se ejecuta es posible interactuar con otros componentes realizados en otros lenguajes .NET de manera muy sencilla.

- También es posible interactuar con componentes no gestionados fuera de la plataforma .NET. Por ello, puede ser integrado con facilidad en sistemas ya creados.

- Desde C# podremos acceder a una librería de clases muy completa y muy bien diseñada, que nos permitirá disminuir en gran medida los tiempos de desarrollo.

ARQUITECTURA DE LA PLATAFORMA .NET FRAMEWORK

Los programas de C# se ejecutan en .NET Framework como se muestra en la figura 20, un componente que forma parte de Windows y que incluye un sistema de ejecución virtual denominado Common Language Runtime (CLR) y un conjunto unificado de bibliotecas de clases.CLR es la implementación comercial de Microsoft de CLI (Common Language Infrastructure), un estándar internacional que constituye la base para crear entornos de ejecución y desarrollo en los que los lenguajes y las bibliotecas trabajan juntos sin ningún problema.

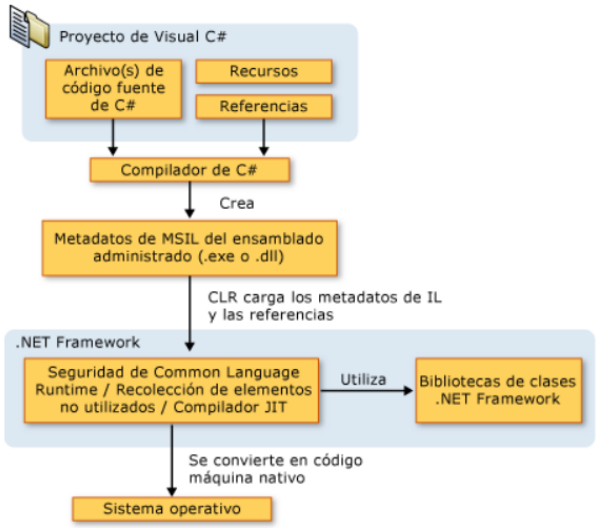


Figura 20.- Arquitectura de la plataforma .NET framework.

El código fuente escrito en C# se compila en un lenguaje intermedio (IL) conforme con la especificación CLI. El código de lenguaje intermedio y recursos tales como mapas de bits y cadenas se almacenan en disco en un archivo ejecutable denominado ensamblado, cuya extensión es .exe o .dll generalmente. Un ensamblado contiene un manifiesto que proporciona información sobre los tipos, la versión, la referencia cultural y los requisitos de seguridad del ensamblado.

Cuando se ejecuta un programa de C#, el ensamblado se carga en CLR, con lo que se pueden realizar diversas acciones en función de la información del manifiesto. A continuación, si se cumplen los requisitos de seguridad, CLR realiza una compilación Just In Time (JIT) para convertir el código de lenguaje intermedio en instrucciones máquina nativas.CLR también proporciona otros servicios

relacionados con la recolección automática de elementos no utilizados, el control de excepciones y la administración de recursos. El código ejecutado por CLR se denomina algunas veces "código administrado", en contraposición al "código no administrado" que se compila en lenguaje máquina nativo destinado a un sistema específico. En el diagrama siguiente se muestran las relaciones en tiempo de compilación y tiempo de ejecución de los archivos de código fuente de C#, las bibliotecas de clases de .NET Framework, los ensamblados y CLR.

La interoperabilidad del lenguaje es una característica clave de .NET Framework. Como el código de lenguaje intermedio generado por el compilador de C# cumple la especificación de tipos común (CTS), este código generado en C# puede interactuar con el código generado en las versiones .NET de Visual Basic, Visual C++ o cualquiera de los más de 20 lenguajes conformes a CTS. Un único ensamblado puede contener varios módulos escritos en diferentes lenguajes .NET, y los tipos admiten referencias entre sí como si estuvieran escritos en el mismo lenguaje.

Además de los servicios en tiempo de ejecución, .NET Framework también incluye una amplia biblioteca de más de 4.000 clases organizadas en espacios de nombres que proporcionan una gran variedad de funciones útiles para la entrada y salida de archivos, la manipulación de cadenas, el análisis XML, los controles de los formularios Windows Forms y muchas tareas más. La aplicación de C# típica utiliza continuamente la biblioteca de clases de .NET Framework para el tratamiento de las tareas comunes de "infraestructura".

PROCEDIMIENTO Y DESCRIPCIÓN DE LAS ACTIVIDADES REALIZADAS.

1. Acopio de Información

Las primeras dos semanas fueron dedicadas al acopio de información general acerca de sistemas de adquisición de datos, tal como: características, estructuras e instrumentación requerida para realizar el diseño del sistema, además de averiguar las necesidades claves de los invernaderos actuales, también se investigó teoría acerca del lenguaje Visual C# 2010, para poder crear la aplicación de registro de datos.

2. Diseño del sistema de adquisición de datos

El siguiente paso fue definir los componentes del sistema, para ello se determinaron los sensores en la intervención de los procesos clave en un invernadero, en cuyo caso se estableció el siguiente diseño de la placa de adquisición llamada tarjeta **SAD** mostrada en la figura 21.

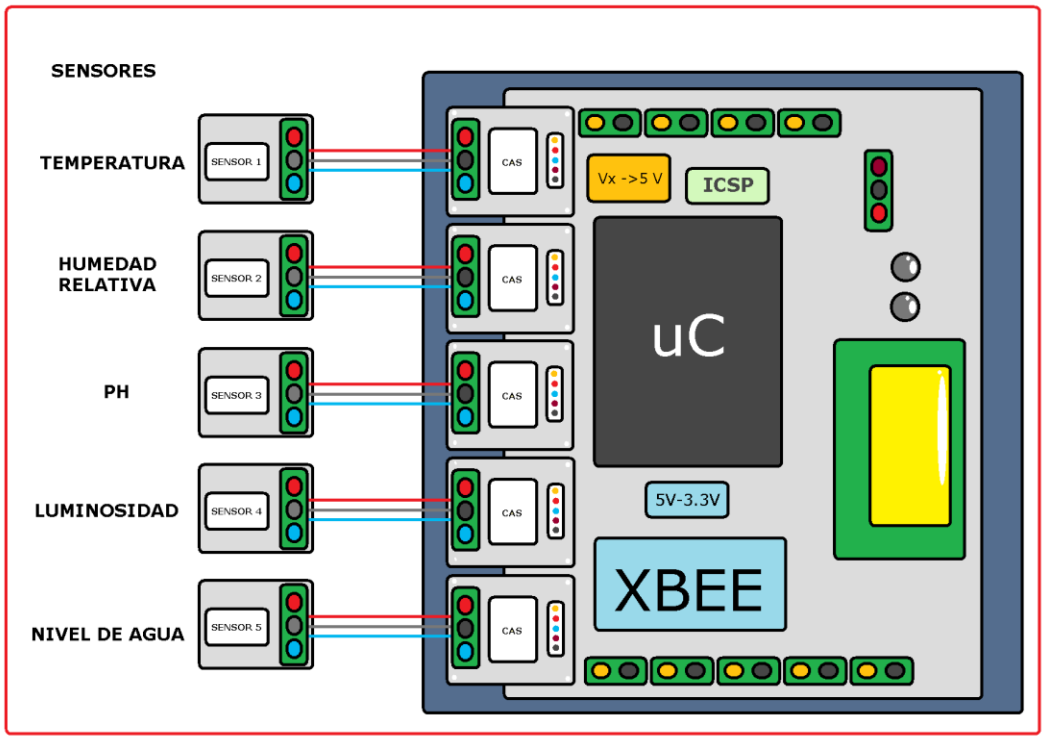


Figura 21.- Diseño del sistema de adquisición de datos.

El diseño de la tarjeta **SAD**, corresponde a un sistema de adquisición de datos con organización centralizada, únicamente se tienen por separado las etapas de acondicionamiento para cada sensor llamadas **CAS**, lo cual da la posibilidad de removerlas sin afectar el funcionamiento de las demás. Los componentes de la tarjeta **SAD** se muestran en la figura 22.

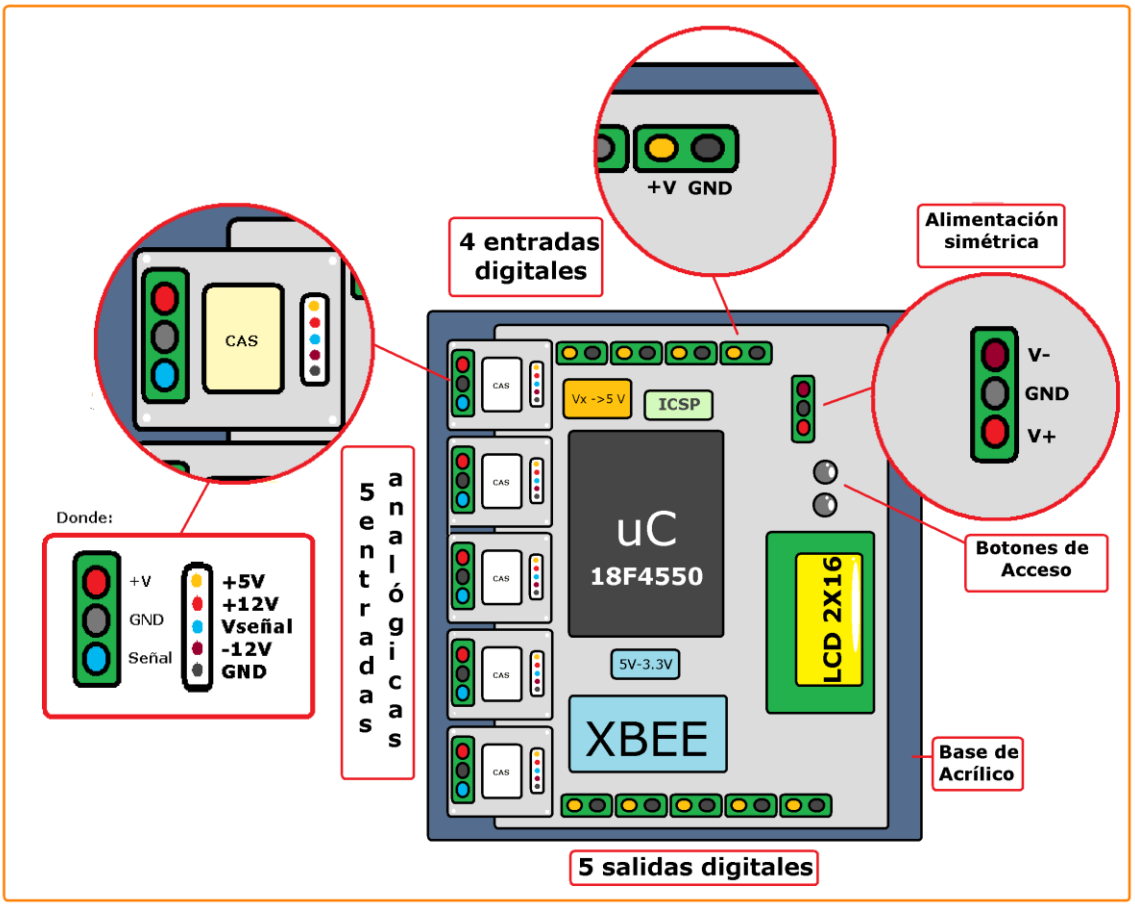


Figura 22.- Tarjeta SAD

La tarjeta **SAD** se alimenta simétricamente con un voltaje de ±15V debido a algunos sensores (en el caso del sensor de presión diferencial para registrar nivel de agua) y tarjetas CAS, además ese voltaje es reducido a +5V por medio de un IC 7805 y a +3.3V por medio de un LM317T (necesario para alimentar los radios XBEE V1).

Cuenta con 2 botones: uno para habilitar o deshabilitar el sistema y el otro para conmutar la visualización de los 5 sensores en el LCD.

Las conexiones de las tarjetas **CAS** con la tarjeta **SAD** son a través de headers para removerlas cómodamente, en caso de ser necesario.

El recuadro nombrado **ICSP** corresponde a la programación “In-Circuit Serial Programming”, para poder programar al microcontrolador PIC18F4550 sin tener que removerlo de la tarjeta **SAD**.

Los sensores del sistema son los siguientes:

- Temperatura – **LM35**
- Humedad relativa – **HIH-4030**
- PH – **WQ-201**
- Luminosidad – **TEMT6000**
- Presión diferencial – **MPX2010DP** (utilizado para medir nivel de agua)

Diseño de circuitos acondicionadores de señal.

Los diferentes sensores entregan distintos valores y formas de señal, por lo cual se tiene la necesidad de estandarizarlas a valores de 0-5 V con diferentes configuraciones de circuitos acondicionadores de señal, en cuyo caso serán las siguientes tarjetas **CAS**:

Tarjeta CAS de temperatura con sensor LM35.

El LM35 brinda una señal de 10mV/°C, por tanto si se desea limitar la temperatura de 0°C-100°C, los rangos de voltaje serian:

Rangos de voltaje $\begin{cases} \text{Para } 0°C \rightarrow 0V \\ \text{Para } 100°C \rightarrow 1V \end{cases}$

Como es necesario estandarizar ese rango de voltaje hacia un rango de 0-5V, se sugiere una ganancia de 5 usando amplificadores operaciones. En un amplificador esta ganancia se logra fácilmente con un arreglo de resistencias en una configuración de inversor, para calcular el voltaje de salida, se tiene la ecuación (8).

$$V_O = V_f\left(\frac{R_f}{R_1}\right) \qquad Ec.(8)$$

Para encontrar la R_1, sustituímos en la ecuación 8 el voltaje de salida y el voltaje de entrada (5V y 1V respectivamente), se propone una R_f de por ejemplo 100KΩ.

$$5 = \left(\frac{100k\Omega}{R_1}\right) \rightarrow R_1 = \frac{100k\Omega}{5} = 20k\Omega$$

Por tanto para obtener la ganancia necesaria con un rango de 0-5V, es necesaria una R_i = 20KΩ y un circuito inversor con ganancia unitaria (R_i = R_f = 10kΩ) para obtener una señal con magnitud positiva.

Por lo tanto el circuito propuesto CAS del sensor de temperatura es el mostrado en la figura 23.

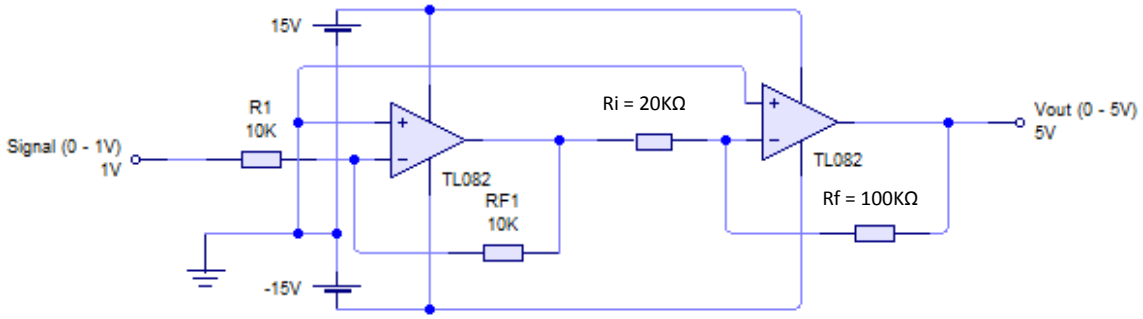


Figura 23.- CAS del sensor de temperatura.

Tarjeta CAS de humedad relativa con sensor HIH-4030

El sensor HIH-4030 arroja un voltaje 30.680mV/%RH, presentando un offset de 0.958mV a 0% de humedad relativa, por tanto se puede calcular qué voltaje ofrece a un 100% de humedad relativa utilizando la ecuación de la recta mostrada en la ecuación 9.

$$mx + b = y \qquad Ec.(9)$$

considerando que la pendiente de la recta (*m*) en este caso es de 30.680mV, a una humedad relativa (*x*) del 100% y con un offset (*b*) de 0.958V.

$$(30.680mV)(100) + 0.958V = 4.026V$$

Ahora para lograr transformar un rango de voltaje de 0.958-4.026V a un rango estándar de 0-5V, haciendo uso la ecuación de la pendiente como se muestra en la ecuación 10.

$$m = \frac{V_{f2} - V_{i2}}{V_{f1} - V_{i1}} \qquad Ec.(10)$$

Donde:

- M: Pendiente de la recta.
- V_{f1}: Voltaje máximo que entrega el sensor (4.026V).
- V_{i1}: Voltaje mínimo que entrega el sensor (0.958V).
- V_{f2}: Voltaje máximo deseado (5V).
- V_{i2}: Voltaje mínimo deseado (0V).

Se hace la sustitución de los valores obtenidos en la ecuación (10).

$$m = \frac{5 - 0}{4.026 - 0.958} = \frac{5}{3.068} = 1.6297$$

A partir del uso de la ecuación (9) se logra encontrar la desviación (*b*), ahora tomando en consideración el voltaje mínimo deseado (0V), la pendiente *m* (1.62) y el voltaje mínimo que entrega el sensor (0.958V).

$$(1.62)(0.958) + b = 0$$

Despejando de la ecuación anterior la desviación, se obtiene.

$$b = -1.5519 \ldots$$

Por lo tanto, la ecuación del voltaje de salida del CAS se muestra en la ecuación (11).

$$V_O = (1.62)(V_S) - 1.5519 \qquad Ec.(11)$$

Donde V_S es el voltaje que entrega el sensor, y una vez obtenida la ecuación del CAS, expresada de la forma *y=mx+b*, ahora se desea un circuito en el que la ganancia de 1.62 y la desviación de -1.5519 se definan de manera independiente. La solución es un amplificador operacional como el que se muestra en la figura 24; un amplificador inversor con ganancia de -1 seguido por un sumador inversor. La ecuación general del voltaje de salida del sumador es la que se muestra en la ecuación (12).

$$V_O = -\left(\frac{R_f}{R_1}\right)(V_S) - \left(\frac{R_f}{R_2}\right)E_{cd} \qquad Ec.(12)$$

Con base en la correspondencia de los coeficientes de V_S en la ecuación (11) y en la ecuación (12) se obtiene:

$$\frac{R_f}{R_1} = 1.62$$

Se elige una R_f = 10KΩ, para obtener el valor de R_1.

$$R_1 = \frac{10k\Omega}{1.62} = 6.17k\Omega$$

Correlacionando los términos correspondientes a la desviación de cd de la ecuación (11) y los de la ecuación (12), se obtiene:

$$-\frac{R_f}{R_2}E_{cd} = -1.5519$$

Suponiendo que se conecta E_{cd} a la fuente de +15V y resolviendo para R2.

$$R_2 = \frac{E_{cd} \cdot R_f}{1.5519} = \frac{15V \cdot 10k\Omega}{1.5519}$$

$$R_2 = 96.77k\Omega$$

Por tanto el diseño del circuito **CAS** del sensor de humedad relativa es el mostrado en la figura 24.

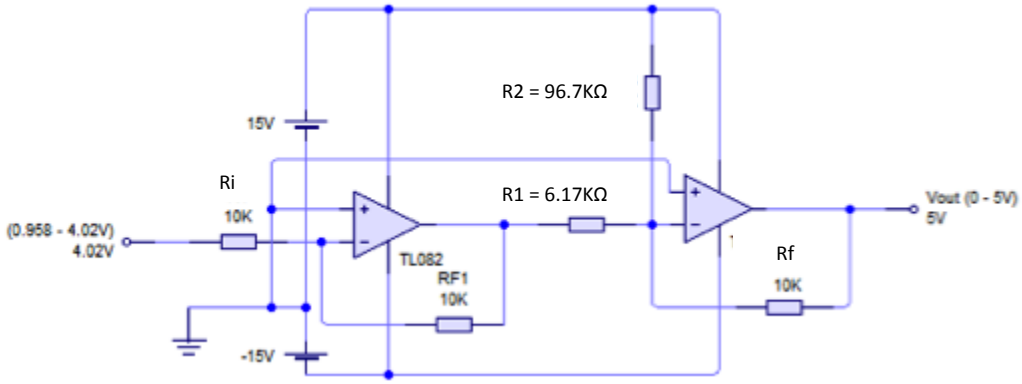


Figura 24.- CAS del sensor de humedad relativa.

Tarjeta CAS de ph con sensor WQ-201

El sensor WQ-201 entrega una corriente en un rango de 4 – 20 mA que equivale a un ph de 0 a 14 unidades, en ese caso se recurre a un convertidor de corriente a voltaje comercial, el **RCV420** capaz de convertir una corriente de entrada de 4-20 mA, a su proporcional en voltaje en un rango de 0-5V, la conexión típica solo sugiere conectar unos capacitores de 1µF aterrizados en su alimentación positiva y negativa (ver hoja de datos en anexos).

Tarjeta CAS de luminosidad con sensor TEMT6000

El sensor TEMT6000 entrega un voltaje proporcional a la fuente de alimentación respecto al número de luxes que logra captar, por tanto no hace falta una etapa de amplificación, ya que es posible que entregue una señal de salida en el rango de 0-5V, si es alimentado con 5V.

Tarjeta CAS de nivel de agua con sensor MPX2010DP

El sensor MPX2010DP es un sensor de presión diferencial, que entrega un voltaje de 0 hasta 25 mV, que representa un rango de presión de 0 – 1.45 Psi (0 – 10 KPa), esto con el propósito de medir el nivel de agua aplicando el concepto de presión hidrostática.

Un fluido pesa y ejerce presión sobre las paredes sobre el fondo del recipiente que lo contiene y sobre la superficie de cualquier objeto sumergido en él. Esta presión, llamada **presión hidrostática**, provoca, en fluidos en reposo, una fuerza perpendicular a las paredes del recipiente o a la superficie del objeto sumergido sin importar la orientación que adopten las caras. Si el líquido fluyera, las fuerzas resultantes de las presiones ya no serían necesariamente perpendiculares a las superficies. Esta presión depende de la densidad del líquido en cuestión y de la altura a la que esté sumergido el cuerpo y se calcula mediante la expresión de la ecuación (13).

$$P = \rho g h \qquad Ec.(13)$$

Donde, usando unidades del SI:

- P: Presión hidrostática (en pascales).
- **ρ**: Densidad del líquido (en kilogramos sobre metro cúbico).
- g: Aceleración de la gravedad (en metros sobre segundo al cuadrado).
- h: Altura del fluido (en metros). Un líquido en equilibrio ejerce fuerzas perpendiculares sobre cualquier superficie sumergida en su interior.

Debido a que el sensor MPX2010DP entrega una señal proporcional a la presión (0-10KPa) detectada y la densidad del agua es de 1000 Kg/m^3, por lo tanto es fácil calcular la altura despejándola de la fórmula de presión hidrostática:

Como

$$1Pa = 1\frac{N}{s^2} = \frac{Kg}{m \cdot s^2}$$

Despejando *h* de la ecuación (13).

$$h = \frac{P}{\rho g} = \frac{p_x\left(\frac{Kg}{m \cdot s^2}\right)}{\left(1000\frac{Kg}{m^3}\right)\left(9.8\frac{m}{s^2}\right)} = \frac{P_x}{9800}(en\ metros)$$

Donde Px es la presión obtenida del sensor MPX2010DP quedando la altura en metros, ahora es posible calcular la cantidad de agua en un contenedor de forma cilíndrica (ver figura 25) con la expresión de su volumen, ahora sólo es necesario saber la longitud de su radio y tener en cuenta la conversión de volumen de m^3 a Litros (1Litro = 1x10^{-3} m^3).

Para calcular el volumen del contenedor de forma cilíndrica se debe tomar en cuenta la ecuación (14).

$$V = \pi r^2 h \qquad Ec.(14)$$

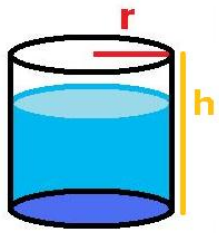


Figura 25.- Contenedor de agua del sistema.

Como ya se tiene el cálculo de la altura (*h*), sólo basta con saber el radio del cilindro y luego realizar la conversión a litros.

$$Litros\ de\ agua = (V)\left(\frac{1Litro}{1x10^{-3}m^3}\right)$$

Donde V es el volumen del contenedor de forma cilíndrica en m^3, consiguiendo la cantidad de agua sin problemas.

Ahora bien, para la aplicación del sensor de presión MPX2010DP es necesario utilizar un amplificador de instrumentación debido a que entrega una diferencia de voltaje muy pequeña, esto se logra con la implementación de un amplificador de instrumentación **INA126** cuya ganancia es ajustada de acuerdo a la estandarización aplicando ecuación (10), de 0-5V como señal de salida.

$$\frac{5-0}{0.025-0} = 200$$

Como la ganancia del INA126 se rige únicamente por una resistencia externa debido a que sus resistencias internas se encuentran ajustadas perfectamente para dar un valor de amplificación absoluto.

La ganancia de este amplificador se ajusta con la ecuación (15).

$$G = 5 + \frac{80k\Omega}{R_G} \qquad Ec.(15)$$

Por lo tanto, si se requiere una ganancia de 200, se resuelve la ecuación (15) para R_G.

$$R_G = \frac{80k\Omega}{200-5} = 410.25\Omega$$

El circuto CAS del sensor de presión diferencial se muestra en la figura 26.

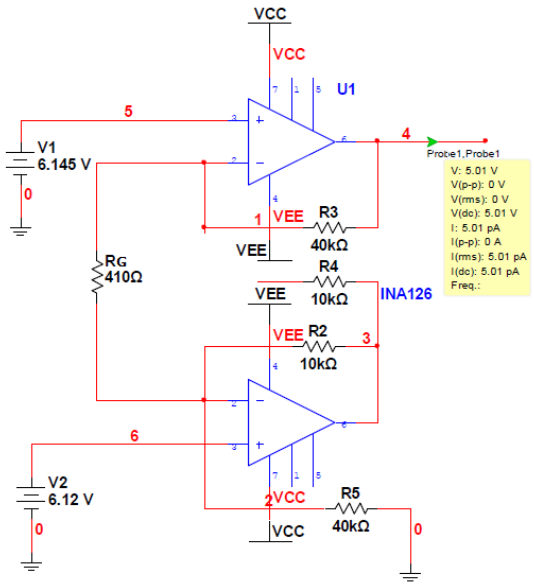


Figura 26.- CAS del sensor de presión diferencial.

3.- Programación de la tarjeta de adquisición de datos SAD

El controlador de la tarjeta de adquisición de datos es un microcontrolador PIC **18F4550** debido a su fácil adquisición en el mercado local, su fácil y rápida programación y con periféricos diversos. Para poder programarlo en lenguaje C se recurrió a la utilización del software **PIC-C CCS V4.106**.

La programación comprende los siguientes aspectos:

- Botón de habilitación o des-habilitación de la tarjeta **SAD**, y al mismo tiempo vía remota desde la computadora.

- Botón visualización de sensores, donde se puede conmutar la visualización (en el LCD) de las variables.

- Selección de sensores vía remota desde la computadora por medio de la utilización de la comunicación RS232.

- Habilitación temporizada del backlight del LCD, con respuesta de encendido mediante cualquier pulsación de los botones del sistema, el backlight se mantiene habilitado para la visualización de las variables hasta por 2 minutos con el fin de contribuir al ahorro energético.

4.- Construcción del sistema de adquisición de datos.

El diseño de la tarjeta **SAD** y las tarjetas de acondicionamiento **CAS** se realizó con la aplicación PCB WIZARD y su construcción fue mediante una fresadora de PCBs

5.- Programación de la base de datos.

La base de datos se programó se realizó con el software VISUAL C# 2010 versión EXPRESS, compatible con la plataforma Windows y la FRAMEWORK V4, cuya característica principal es la de configuración general del sistema y el registro de datos.

6.- Pruebas y ajustes del sistema.

Se realizaron pequeños ajustes a algunas de las placas de acondicionamiento de señal (tarjetas CAS), concluyendo que, como precaución, siempre es necesario tener un seguidor de voltaje para cada sensor, por el momento estas modificaciones son necesarias sólo para los siguientes sensores:

- HIH-4030 (Sensor humedad).
- TEMP6000 (Sensor de luz).
- MPX2010DP (Sensor de presión).

Con lo anterior el sistema funciona correctamente de acuerdo con los requerimientos planteados.

RESULTADOS, PLANOS, GRAFICAS, PROTOTIPOS Y PROGRAMAS

En la figura 27 se muestra el esquema del sistema de adquisición de datos para invernadero HERVA.

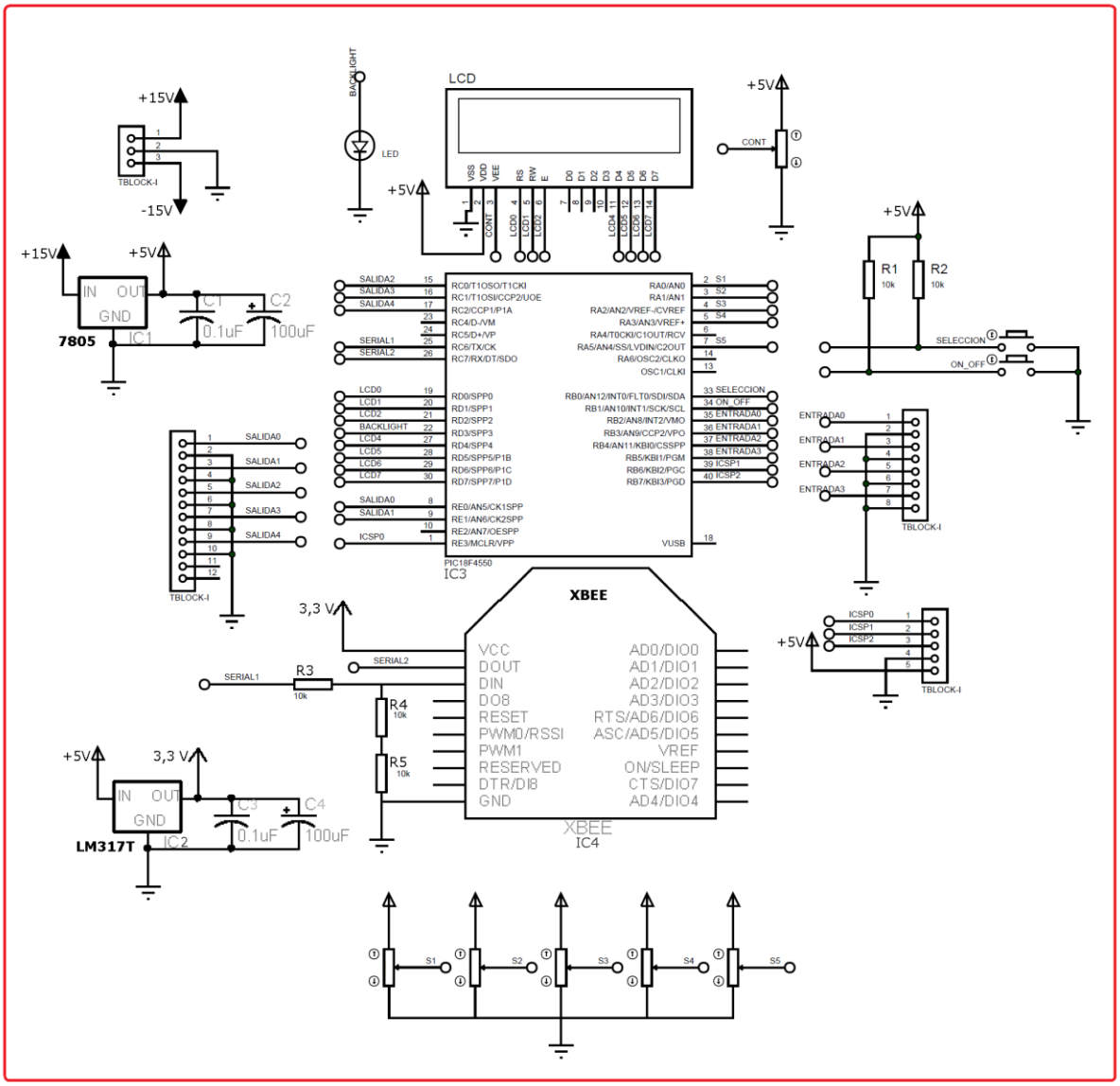


Figura 27.- Esquema del circuito de la tarjeta SAD.

De las figuras 28-32, se muestran los rutados de las tarjetas de acondicionamiento CAS de cada sensor.

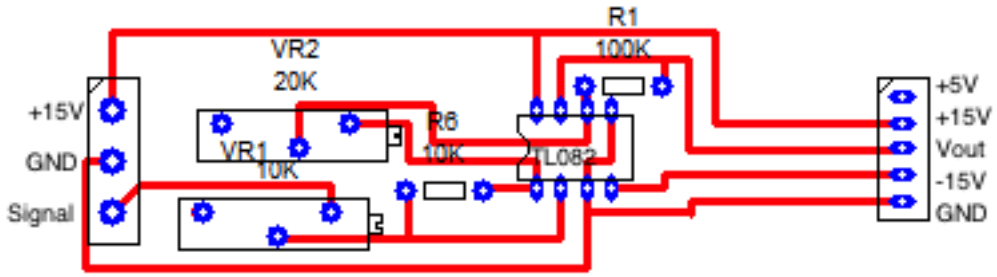


Figura 28.- PCB de la tarjeta CAS de sensor de temperatura.

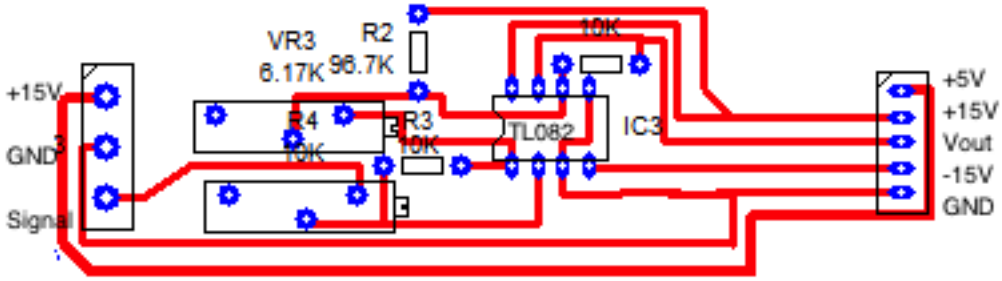


Figura 29.- PCB de la tarjeta CAS sensor de humedad relativa.

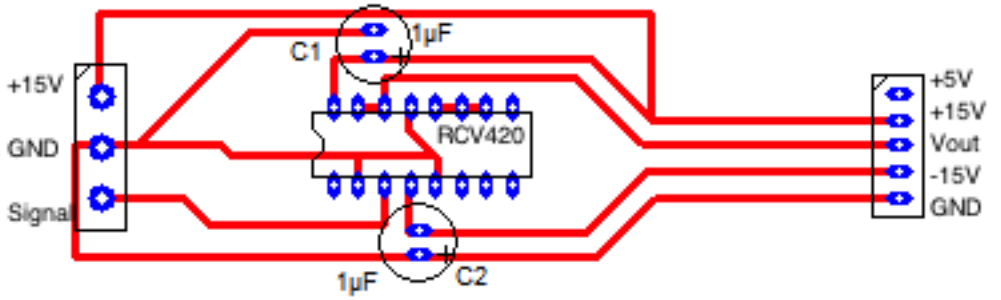


Figura 30.- PCB de la tarjeta CAS sensor de ph.

Figura 31.- PCB de la tarjeta CAS sensor de luminosidad.

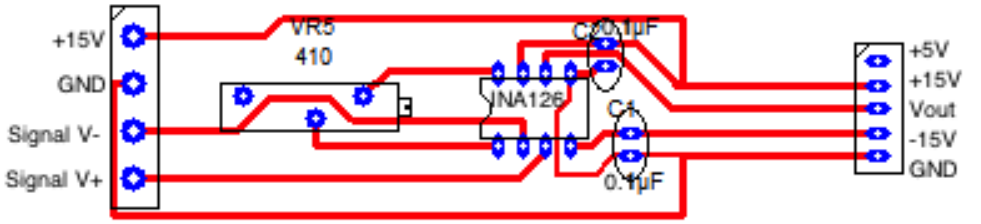


Figura 32.- PCB de la tarjeta CAS sensor de presión diferencial para determinar nivel de agua.

La figura 33 muestra el rutado de la placa de visualización y control de la tarjeta SAD.

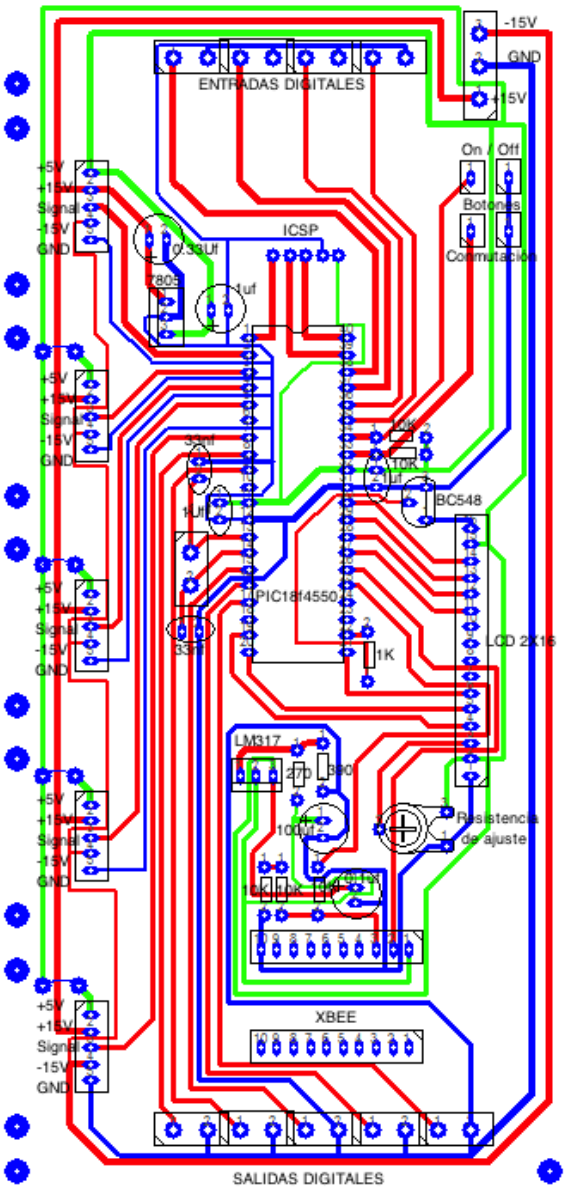


Figura 33.- PCB de la tarjeta de adquisición de datos SAD.

Programación del microcontrolador PIC18F4550. (Anexo A)

La tarjeta **SAD** contiene un microcontrolador PIC18F4550 que realiza las funciones de tomar datos, envío y recepción de información vinculado con la aplicación **HERVA V2.1**, cuyo diagrama de flujo se muestra en la figura 34.

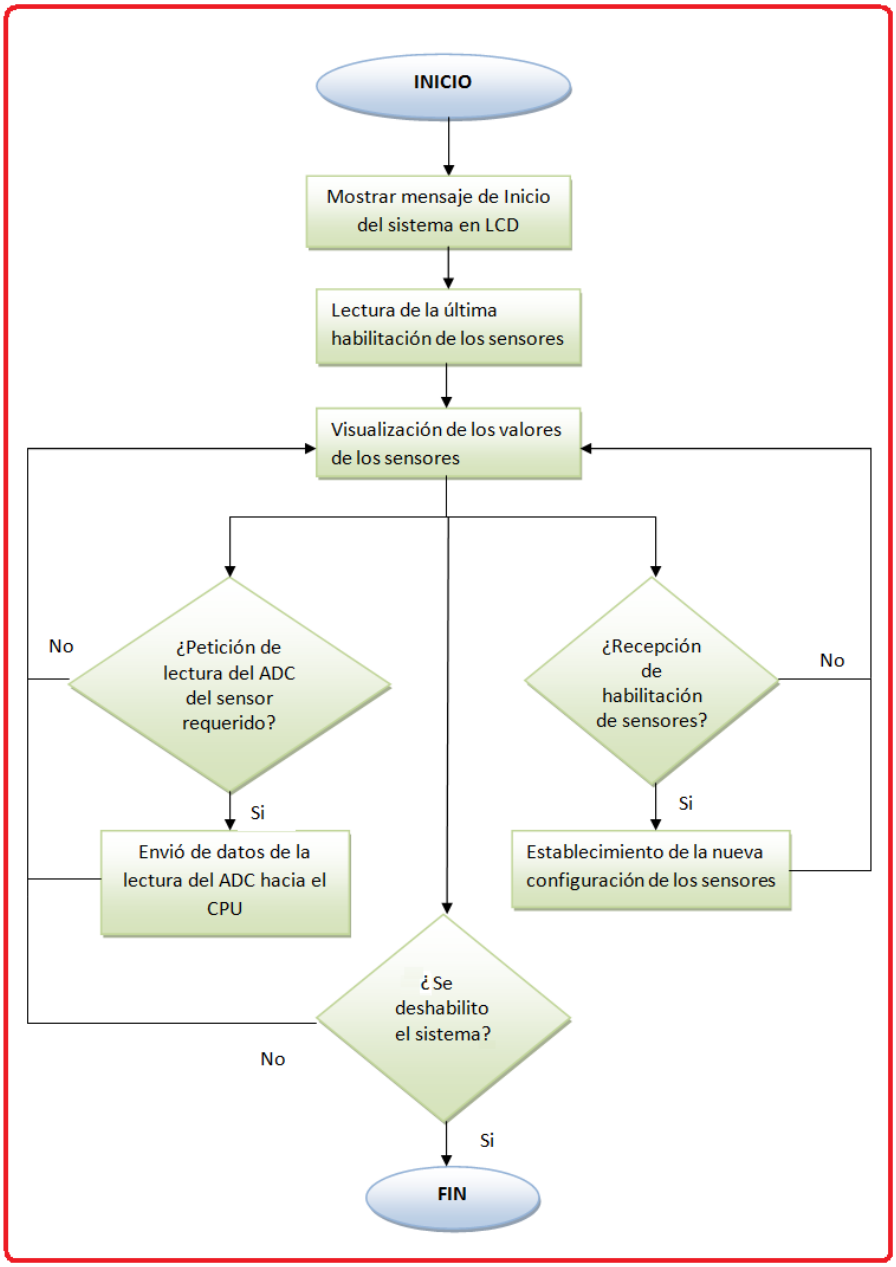


Figura 34.- Diagrama de flujo del programa del PIC18F4550.

Inicialmente el microcontrolador muestra un mensaje de bienvenida saliendo del modo sleep, posteriormente realiza la lectura de la última habilitación de selección de los sensores, luego se ejecuta la visualización individual de los valores de los sensores y finalmente se accede a tres condiciones: “Petición de la

lectura del ADC del sensor requerido", donde se realiza el envio del dato de la lectura del ADC hacia la computadora, la segunda condición "recepción de la habilitación de sensores", mediante el cual se recibe la nueva configuración de los sensores que se desean habilitar y "deshabilitación del sistema", por medio del botón de deshabilitación donde el microcontrolador entra nuevamente en modo sleep.

Programación de la base de datos. (Anexo B)

La base de datos se programó se realizó con el software VISUAL C# 2010 versión EXPRESS, mediante el diagrama de flujo se encuentra ilustrado en la figura 35.

Primeramente, es posible seleccionar 3 opciones, las cuales son: (1) "nuevo proyecto", (2) "abrir gráfica existente", donde se puede acceder hacia alguna gráfica previamente almacenada y la opción (3) "¿cómo usar?", ésta última abre un tutorial en PDF sobre el manejo del sistema.

Cuando se selecciona la opción (1) "nuevo proyecto", se debe definir el puerto COM con que se establecerá la comunicación, luego seleccionar los sensores, definir el tiempo de muestreo e intervalo de muestreo, posteriormente se da comienzo hacia el proceso de registro, en caso de haber elegido el sensor de nivel, se tendrá que definir el radio del contenedor para poder efectuar el cálculo del volumen en litros del mismo.

Una vez que el proceso de registro de los datos haya iniciado puede ser pausado o reanudado en cualquier instante, de no hacerlo el registro seguirá hasta alcanzar el tiempo de muestreo, cuando se llega a este instante es posible realizar el análisis de resultados, ya sea optando por mostrar las gráficas de los valores arojados por los sensores correspondientes o guardar las tablas con los datos obtenidos durante el lapso de tiempo definido.

En caso de requerir de un nuevo registro, es posible iniciar un nuevo proyecto cada vez que el tiempo de muestreo haya concluido o la aplicación no se encuentre registrando algúna variable.

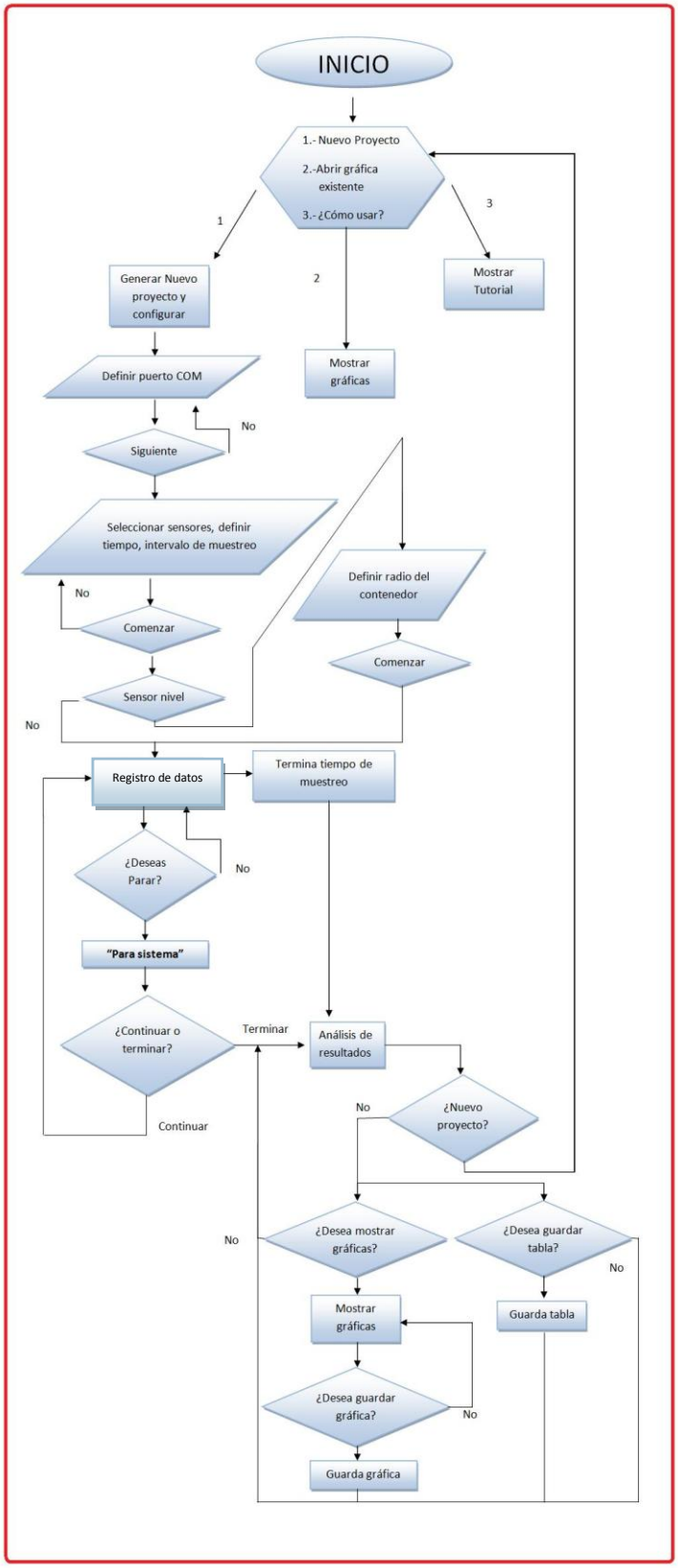


Figura 35.- Algoritmo de la base de datos.

La aplicación del sistema HERVA V2.1 comprende de diversas formas para su uso, para programarlos se recurrió a la utilización de las diferentes controles y herramientas del entorno de desarrollo integrado visual C# 2010.

La forma principal de HERVA V2.1 contiene los controles mostrados en la figura 36.

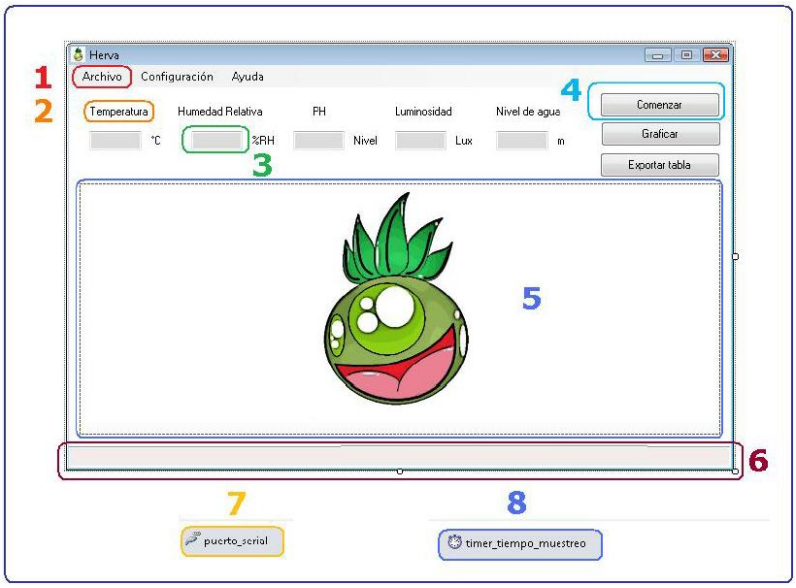


Figura 36.- Controles de la forma principal.

1. **MenuStrip.** Muestra los comandos y las opciones de la aplicación agrupados por funcionalidad.

2. **Label.** Proporciona información de tiempo de ejecución o texto descriptivo para un control.

3. **TextBox.** Permite al usuario especificar texto, así como funciones de edición de varias líneas y máscaras de caracteres para contraseña.

4. **Button.** Desencadena un evento cuando un usuario hace clic sobre él.

5. **PictureBox.** Muestra una imagen. Y **DataGridView:** Muestra filas y columnas de datos en una cuadrícula que se puede personalizar.

6. **StatusStrip.** Muestra información para el usuario acerca del objeto que se está viendo, los componentes del mismo o su funcionamiento.

7. **SerialPort.** Representa un recurso del puerto serie.

8. **Timer.** Componente que desencadena un evento a intervalos definidos por el usuario.

La forma de gráficas de HERVA V2.1 contiene los controles mostrados en la figura 37.

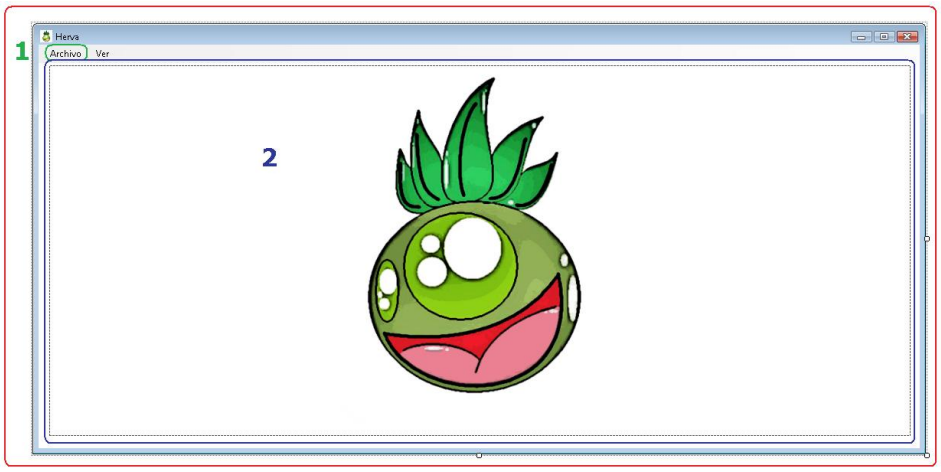


Figura 37.- Controles de la forma de gráficas.

1. **MenuStrip:** Muestra los comandos y las opciones de la aplicación agrupados por funcionalidad.

2. **PictureBox:** Muestra una imagen. Y **Chat:** Control de Windows Form para gráficos.

La forma de configuración de HERVA V2.1 contiene los controles mostrados en la figura 38.

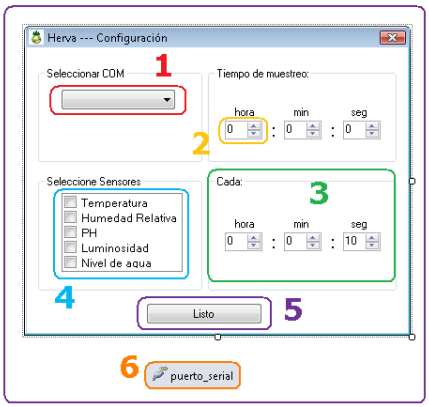


Figura 38.- Controles de la forma de configuración.

1. **ComboBox:** Muestra un cuadro de texto editable con una lista desplegable de los valores permitidos.

2. **NumericUpDown:** Muestra un único valor numérico que el usuario puede aumentar o disminuir haciendo clic en los botones de arriba o abajo del control.

3. **GroupBox:** Muestra un marco alrededor de un grupo de controles con un título opcional

4. **CheckedListBox:** Muestra una lista de elementos con una casilla a la izquierda de cada elemento.

5. **Button:** Desencadena un evento cuando un usuario hace clic sobre él.

6. **SerialPort:** Representa un recurso del puerto serie.

La forma de nivel de agua de HERVA V2.1 contiene los controles mostrados en la figura 39.

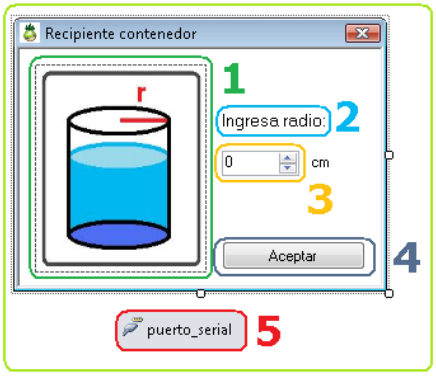


Figura 39.- Controles de la forma de nivel de agua.

1. **PictureBox:** Muestra una imagen.

2. **Label:** Proporciona información de tiempo de ejecución o texto descriptivo para un control.

3. **NumericUpDown:** Muestra un único valor numérico que el usuario puede aumentar o disminuir haciendo clic en los botones de arriba o abajo del control.

4. **Button:** Desencadena un evento cuando un usuario hace clic sobre él.

5. **SerialPort:** Representa un recurso del puerto serie.

La tarjeta SAD se muestra en la figura 40, su fincionamiento fué probado en las canchas de basquetbol del Instituto Tecnológico de Tuxtla Gutiérrez bajo las condiciones de temperatura ambiental de 25°C, humedad relativa ambiental de 60%, luminocidad de 980 luxes, ph de agua potable de 7.3 y nivel de agua en un recipiente de plástico con una sección transversal de 2827.4 cm^2 con 3 Litros de agua.

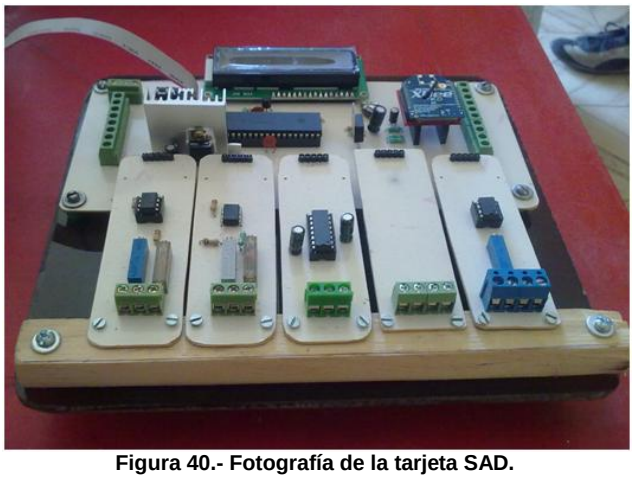


Figura 40.- Fotografía de la tarjeta SAD.

En la figura 41 se la visualización individual de la variable “humedad relativa”, y se muestran 4 de los sensores utilizados para la adquisición.

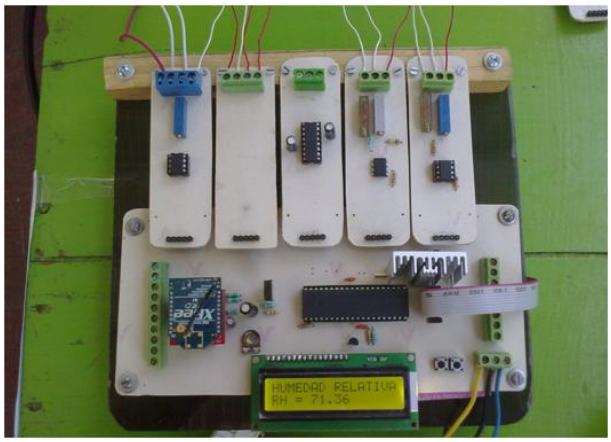


Figura 41.- Tarjeta SAD activada.

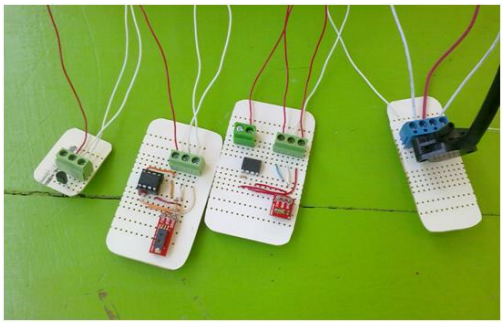


Figura 42.- Sensores: temperatura, humedad relativa, luminosidad y presión diferencial.

La figura 43 muestra el tarjeta exploradora/programadora de Xbee dongle habilitado para la transmisión y recepción de datos entre la tarjeta SAD y la aplicación especial HERVA 2.1 ejecutándose en la computadora.

Figura 43.- Radio transmisor XBEE Serie 1 Pro y tarjeta exploradora / programadora para XBEE dongle.

Para comenzar a utilizar la aplicación HERVA V2.1 inicialmente se debe dirigir al menú “configuración/opciones” (A) y selecciones el puerto COM con el que el dispositivo dongle del Xbee se nombró (B) y posteriormente seleccione los sensores cuya conexión existe en la tarjeta SAD, luego inserte el tiempo de muestreo (duración total de registro) y el intervalo requerido para tomar las muestras (C) una vez establecidos todos los parámetros, pulsar el botón “aceptar”, como se ve en la figura 44.

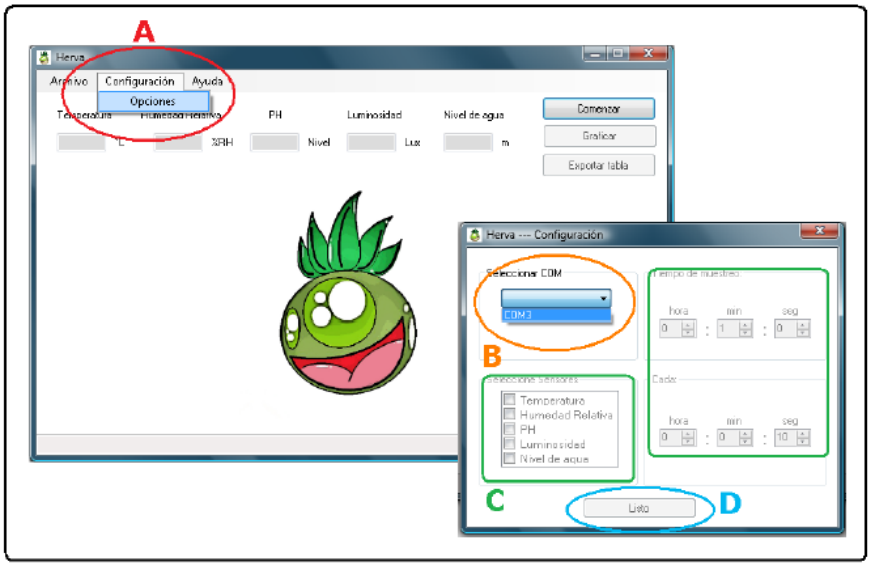


Figura 44.- Configuración de la aplicación HERVA V2.1

Una vez terminado el proceso de muestreo en el tiempo establecido, es posible obtener las gráficas correspondientes a los sensores seleccionados pulsando el botón “graficar” (E), para cambiar la vualización entre gráficas basta con dirigirse al menú “ver/variable a visualizar” (F) y almacenar la gráfica que se visualiza en ese momento en el fichero definido por el usuario, por medio de la opción del menú “archivo/guardar“(G), como se muestra en la figura 45.

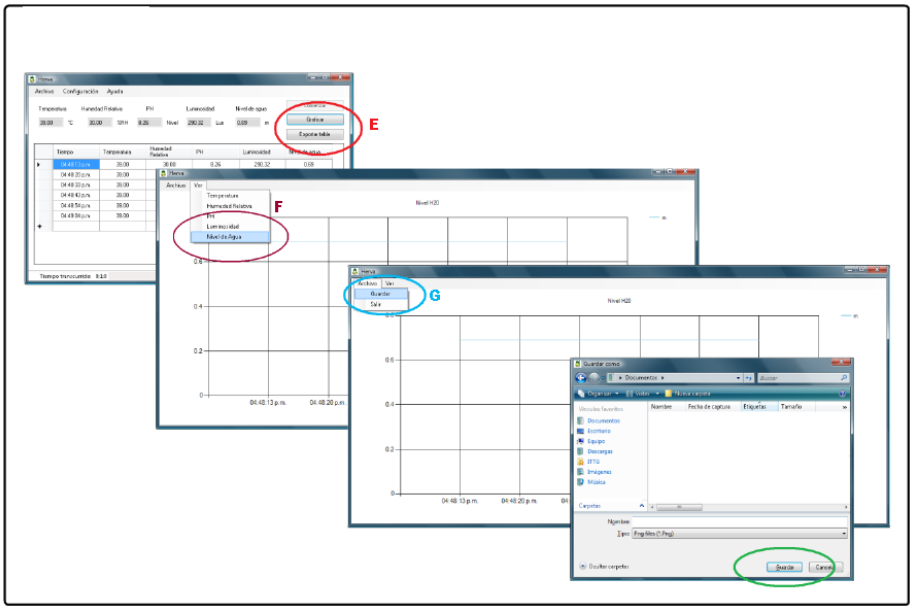


Figura 45.- Visualización y almacenamiento de los datos en HERVA V2.1

En la figura 46 y 47 se exponen dos gráficas de los datos de temperatura y humedad relativa del ambiente respectivamente.

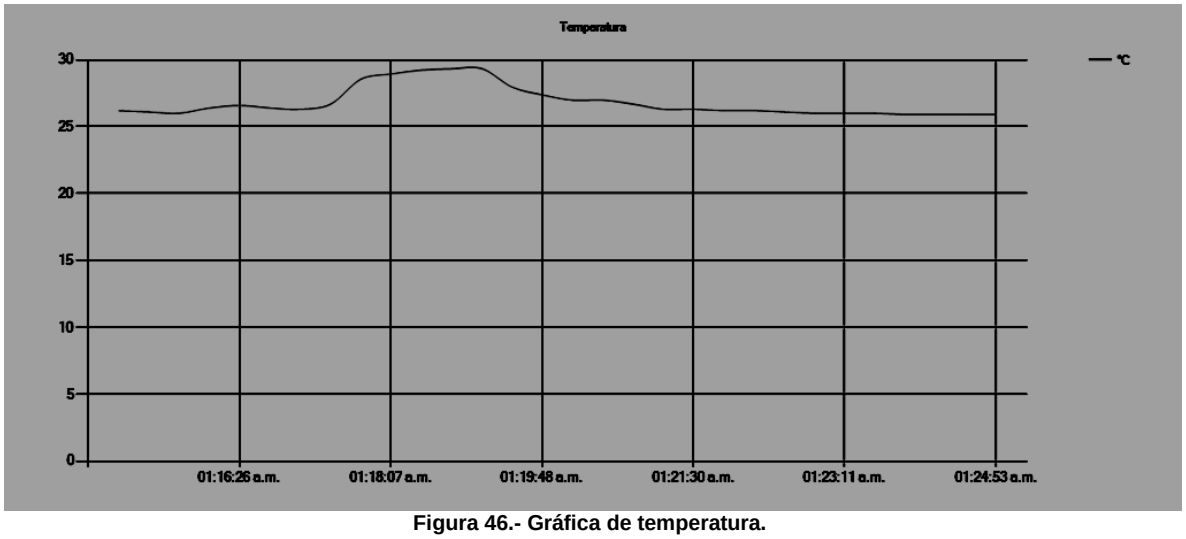


Figura 46.- Gráfica de temperatura.

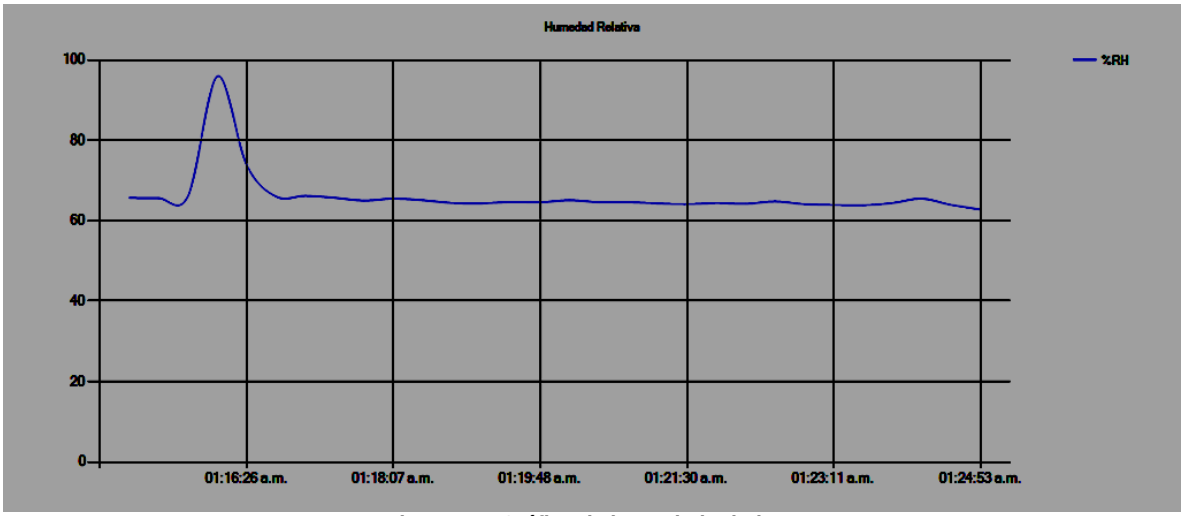


Figura 47.- Gráfica de humedad relativa.

Es posible exportar la tabla de datos pulsando el botón “exportar tabla” (H) y finalmente presionar el botón “guardar” (I) para almacenarla en el fichero elegido, como se muestra en la figura 48.

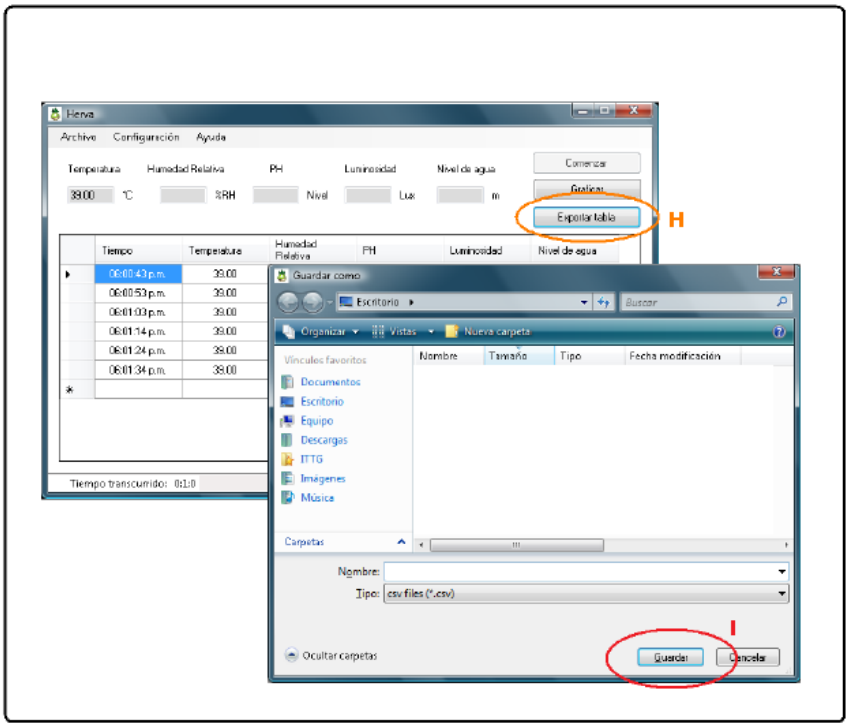


Figura 48.- Almacenamiento de tabla de datos.

La apariencia de la tabla de datos almacenados es la ilustrada en la tabla 5.

Tiempo	Temperatura	Humedad Relativa	PH	Luminosidad	Nivel de agua
02:33:14 p.m.	28.64	64.22	7.29	537.63	2.06
02:33:24 p.m.	28.64	65.68	7.28	544.47	2.06
02:33:34 p.m.	27.37	63.34	7.30	542.52	2.06
02:33:45 p.m.	28.64	65.49	7.29	561.09	2.06
02:33:55 p.m.	26.68	65.49	7.24	524.92	2.06

Tabla 5.- Datos obtenidos en una tabla de HERVA V2.1

La figura 49 muestra a la aplicación HERVA 2.1 ejecutándose en una PC, registrándo los datos de los sensores de temperatura, humedad relativa y luminosidad del medio ambiente.

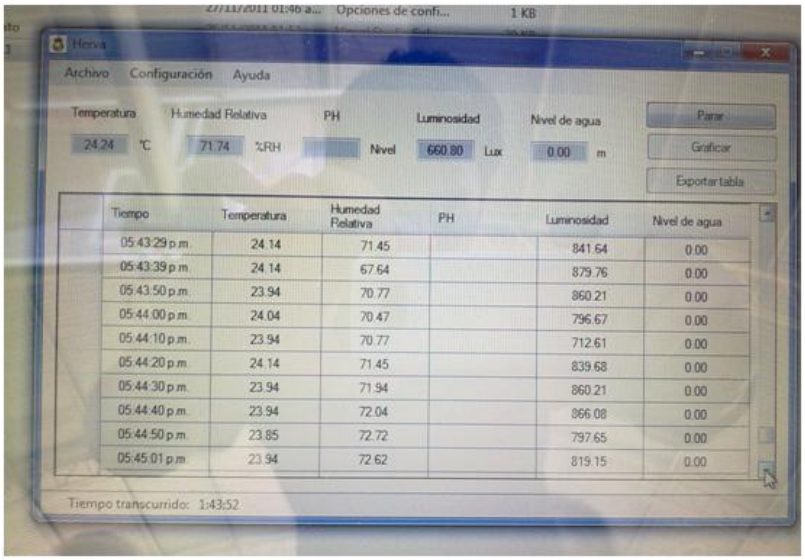


Figura 49.- HERVA V2.1 realizando un almacenamiento de datos.

CONCLUSIONES Y RECOMENDACIONES

Podemos concluir que el sistema de adquisición de datos para invernadero **HERVA** (tarjeta **SAD** y **HERVA V2.1**) realiza el registro de datos provenientes de los sensores de temperatura, humedad relativa, ph, luminosidad y presión diferencial utilizado para medir el nivel de agua de manera rápida, eficaz y de bajo costo, gracias a la programación con el modo **ICSP** es posible reprogramar la tarjeta **SAD** para futuras aplicaciones, además que el envío de datos se realiza cómodamente de forma inalámbrica, evitando el menor cableado posible. La adaptación de las tarjetas **CAS** es diversa para distintas clases de sensores gracias a que son desmontables, haciendo del sistema no solamente optimo para invernaderos sino también es posible usarlo en otras aplicaciones en donde se desee el monitoreo de variables.

La tarjeta de adquisición de datos **SAD** por el momento no tiene la capacidad de control, pero cuenta con 2 módulos PWMs y 7 E/S de propósito general, esto con la finalidad de que a futuro se pueda controlar algún proceso con las etapas de potencia necesarias que más se acomoden a nuestras necesidades.

Se recomienda realizar las conexiones correctamente, para evitar dañar componentes del circuito, colocar la tarjeta de adquisición **SAD** en un lugar seguro y que no se exponga a factores ambientales peligrosos, como lluvia, rocíos, etc. que puedan dañar el circuito, cuando se desee desmontar alguna de las tarjetas **CAS**, realizarlo con mucho cuidado, para no dañarlas.

REFERENCIAS BIBLIOGRÁFICAS Y VIRTUALES

Bibliografía

1. Bolton, William. Mediciones y pruebas eléctricas y electrónicas. España. Marcombo. 1995.
2. Calleja Gjumlich, Hugo, “Circuitos Electrónicos para la adquisición de datos”, SEP, 1ra edición, 1998.
3. Ceballos, Francisco Javier. “Enciclopedia Microsoft Visual C#”. Alfaomega. 3ra edición. México. 2010.
4. Corrales, Santiago. Electrónica práctica con microcontroladores PIC. Ecuador. Imprenta gráfica. 2006.
5. Coughlin, Robert F. “Amplificadores operacionales y circuitos integrados lineales”. Pearson. 5ta edición. 1999. México. 544 págs.
6. García, Eduardo. “Compilador C CCS y Simulador PROTEUS - Para Microcontroladores PIC”. Alfaomega. 1ra edición.2008. México.276 págs.
7. Oyarce, Andrés. Guía del Usuario Xbee Series 1. MCI electronics. 2008
8. Ruiz, Diego. Csharp, la guía total del programador. México. Editorial USERS. 2010
9. Sumerville, Ian. Ingeniería del software. España. Editorial Addison-Wesley. 2005.

Referencias virtuales:

1. http://msdn.microsoft.com/es-es/library/67ef8sbd.aspx Consultada en septiembre del 2011.

2. http://msdn.microsoft.com/es-es/vcsharp/default.aspx Consultada en noviembre del 2011.

3. http://www.pedrov.info/ Consultada en noviembre del 2011.

4. http://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf Hoja de datos microcontrolador PIC18F4550. MICROCHIP. 2006. Consultada en octubre del 2011.

ANEXO A

Programación del microcontrolador PIC18F4550.

```
//Directivas
#include <18f4550.h>
#fuses HS, NOWDT, NOMCLR, INTRC_IO, PLL1, CPUDIV1, NOPUT, NOVREGEN, NOLVP, PROTECT
#device adc = 10                          //ADC de 10 bits
#use delay( CLOCK = 8 MHz )               //Cristal interno de 8 MHz
#use RS232( UART1, BAUD = 9600 )          //Configuramos y habilitamos la UART del PIC
#include "<lcd.c>                         //Cargamos librería de LCD
#include <stdlib.h>                       //Librería necesaria para la conversión de un string a int
#include <internal_eeprom.c>              //Librería usada en la utilización de la memoria eeprom interna
#include <math.h>                         //Librería de funciones y constantes matemáticas
#priority rda                             //Prioridad a la interrupción recepción de dato disponible

//Salida para el control del backlight del LCD
#define BACKLIGHT  PIN_D3

//Variables globales de uso general
float aux_sensor;
int8  aux_canal = 0;
int8  aux_on_off = 0;
long  aux_timer = 0;
unsigned aux_selecciona_luz = 0;

//Variables utilizadas en la comunicación serial RS232
int8 recibe_seleccion = 0;
char cadena[20];
int8 radio = 0;
float radio_m = 0;
char aux_nivel[20];
 int1 aux_nivel_bool = 0;
int8 memoria;

//Variables utilizadas para el control del LCD de 2X16
int8 x = 0;
long j = 0;
float aux_sensor1 = 0;
float aux_sensor2 = 0;

//Funciones de tiempo
void tiempo()
{
   Delay_ms(200);
}

void tiempo_boton()
{
   Delay_ms(500);
}

//Programa principal
void main() {

   //Deshabilitación de periféricos que no se utilizan
   //setup_spi(SPI_SS_DISABLED);
   setup_psp( PSP_DISABLED );
   setup_wdt( WDT_OFF );
   setup_timer_1( T1_DISABLED );
   setup_timer_2( T2_DISABLED , 0, 1);
   setup_timer_3( T3_DISABLED );
   setup_ccp1( CCP_OFF );
   setup_ccp2( CCP_OFF );
   setup_comparator( NC_NC_NC_NC );
   setup_vref( FALSE );
```

```
   //Inicializamos LCD
   lcd_init();

   //Configuramos entradas analógicas
   setup_adc_ports( AN0_TO_AN4 | VSS_VDD);
   setup_adc( ADC_CLOCK_INTERNAL );

   //Configuramos temporizador timer0 para el control del LCD
   setup_timer_0 ( T0_INTERNAL | T0_DIV_256 );

   //Habilitamos interrupciones
   enable_interrupts( GLOBAL );
   enable_interrupts( INT_EXT1_H2L );
   enable_interrupts( INT_RDA );

//Etiqueta de inicio (loop)
INICIO:

   set_timer0( 0 );                          //Reset a timer0
   aux_timer = 0;
   output_low( BACKLIGHT );                  //Apagamos backlight

   sleep();                                  //Ponemos a dormir al PIC

//Si se presenta alguna interrupción del tipo RDA o EXT1 el PIC se despierta
   output_high( BACKLIGHT );

//Saludo inicial
      for( x = 0; x < 5; x++ ){
         printf( lcd_putc, "\f(oo,) Hello!\nSistema [ On ]" );
         tiempo();
         tiempo();
         printf( lcd_putc, "\f(^^,) Hello!\nSistema [ On ]" );
         tiempo();
         tiempo();
      }


//Habilitamos interrupciones del timer0 y la externa EXT
   enable_interrupts( INT_TIMER0 );
   enable_interrupts( INT_EXT_H2L );

   aux_selecciona_luz = 2;
   memoria = read_eeprom( 0x00 );

//Obtenemos la información de la última variable que se visualizó
//en el LCD para comenzar desde ahí.
  if( memoria )
  {
     write_eeprom( 0x00, 0x00 );
     write_eeprom( 0x01, 0x00 );
     memoria = 0;
     goto PRIMERO;
  }

  else
  {

        aux_canal = read_eeprom ( 0x01 );

//Segunda etiqueta para comenzar la visualización de
//las variables en el LCD
PRIMERO:

//Limpiamos pantalla
        printf( lcd_putc,"\f" );

//Configuramos canal del ADC
        set_adc_channel( aux_canal );
```

```
      switch( aux_canal )
      {
        case 0:
            printf( lcd_putc,"TEMPERATURA     " );
            break;

        case 1:
            printf( lcd_putc,"HUMEDAD RELATIVA" );
            break;

        case 2:
            printf( lcd_putc,"PH              " );
            break;

        case 3:
            printf( lcd_putc,"LUMINOSIDAD     " );
            break;

        case 4:
            printf( lcd_putc,"NIVEL DE AGUA   " );
            break;

        default:
            printf( lcd_putc,"TEMPERATURA     " );
            break;
      }
  }

   aux_on_off = 1;

//Ciclo infinito para la visualización del estado de las
//variable seleccionadas
   while( TRUE )
   {

//Condición de deshabilitación de la tarjeta
      if( aux_on_off == 2 )
      {
        output_high( BACKLIGHT );
        aux_on_off = 0;
        printf( lcd_putc,"\f" );
        disable_interrupts( INT_TIMER0 );
        disable_interrupts( INT_EXT_H2L );

         for( x = 0; x < 5; x++ )
         {
            printf( lcd_putc, "\f(o_o)    Bye...\nSistema [Sleep]" );
            tiempo();
            printf( lcd_putc, "\f(-_-)z   Bye...\nSistema [Sleep]" );
            tiempo();
            printf( lcd_putc, "\f(-_-)zz  Bye...\nSistema [Sleep]" );
            tiempo();
            printf( lcd_putc, "\f(-_-)zzZ Bye...\nSistema [Sleep]" );
            tiempo();
         }

        printf( lcd_putc,"\f" );
        goto INICIO;
   }
```

```
//Bucle de estabilidad para la visualización de las variables
//del sistema por medio del uso de promedio de las mediciones
        if( j < 1000 ){
           switch ( aux_canal )
           {
             case 0: aux_sensor1 = read_adc()/10.23; break;
             case 1: aux_sensor1 = read_adc()/10.23; break;
             case 2: aux_sensor1 = read_adc()*14.0/1023; break;
             case 3: aux_sensor1 = read_adc()/1.023; break;
             case 4:
             if(aux_nivel_bool == 0)
               aux_sensor1 = read_adc()/0.1023/9800;
             else
               aux_sensor1 = (read_adc()/0.1023/9800)* pi * radio_m * 1000;
             break;
             default: aux_sensor1 = read_adc()/10.23; break;
           }
             aux_sensor2 = aux_sensor2 + aux_sensor1;
             j++;
        }
        else{

           aux_sensor2 = aux_sensor2 / (j-1);

           lcd_gotoxy(1,2);

          //Formato de visualización de las variables del sistema
           switch ( aux_canal )
           {
             case 0: printf(lcd_putc,"Grados C = %3.2f  ", aux_sensor2); break;
             case 1: printf(lcd_putc,"RH = %3.2f        ", aux_sensor2); break;
             case 2: printf(lcd_putc,"Nivel = %3.2f     ", aux_sensor2); break;
             case 3: printf(lcd_putc,"Luxes = %3.2f     ", aux_sensor2); break;
             case 4:
             if(aux_nivel_bool == 0)
               printf(lcd_putc,"m = %3.2f         ", aux_sensor2);
             else
               printf(lcd_putc,"L = %5.2f         ", aux_sensor2);

               break;
             default: printf(lcd_putc,"Grados C = %3.2f ", aux_sensor2); break;

           }

           aux_sensor2 = 0;
           j = 0;
        }
  }
}

//Función de habilitación o deshabilitación de la tarjeta SAD
#INT_EXT1
void despierta()
{
  aux_on_off++;
  if( aux_on_off > 2 ) aux_on_off = 1;
  tiempo_boton();
}

//Función de conmutación de la visualización de las variables del sistema
#INT_EXT
void seleccionar_sensor()
{
  set_timer0(0);
  aux_timer = 0;

  aux_selecciona_luz += 1;
```

```
//Condición para encender el backlight del LCD de forma eficiente
  if( aux_selecciona_luz ==1 )
  {

    enable_interrupts( INT_TIMER0 );
    output_high( BACKLIGHT );
  }

  if(aux_selecciona_luz > 1)
  {

    aux_canal++;

//Función de selección de sensores
//Switch de limitación de valores para la conmutación eficiente de visualización
//en la LCD
  switch( recibe_seleccion )
  {
    case  1:
          if( aux_canal > 0 ) aux_canal = 0;                    //T
          break;
    case  2:
          if( aux_canal < 1 || aux_canal > 1 ) aux_canal = 1; //RH
          break;
    case  3:
          if( aux_canal > 1 ) aux_canal = 0;                    //T RH
          break;
    case  4:
          if( aux_canal < 2 || aux_canal > 2 ) aux_canal = 2; //PH
          break;
    case  5:
          if( aux_canal < 1 || aux_canal > 2 ) aux_canal = 0; //PH T
          if( aux_canal == 1 ) aux_canal = 2;
          break;
    case  6:
          if( aux_canal < 1 || aux_canal > 2 ) aux_canal = 1; //PH RH
          break;
    case  7:
          if( aux_canal > 2 ) aux_canal = 0;                    //PH RH T
          break;
    case  8:
          if( aux_canal < 3 || aux_canal > 3 ) aux_canal = 3; //L
          break;
    case  9:
          if( aux_canal > 3 ) aux_canal = 0;                    //L T
          if( aux_canal > 0 ) aux_canal = 3;
          break;
    case 10:
          if( aux_canal < 1 || aux_canal > 3) aux_canal = 1;  //L H
          if( aux_canal == 2) aux_canal = 3;
          break;
    case 11:
          if( aux_canal > 3) aux_canal = 0;                     //L H T
          if( aux_canal == 2) aux_canal = 3;
          break;
    case 12:
          if( aux_canal < 2 || aux_canal > 3) aux_canal = 2; //L PH
          break;
    case 13:
          if( aux_canal > 3) aux_canal = 0;                     //L PH T
          if( aux_canal == 1 ) aux_canal = 2;
          break;
    case 14:
          if( aux_canal < 1 || aux_canal > 3) aux_canal = 1; //L PH T
          break;
    case 15:
          if( aux_canal > 3) aux_canal = 0;                     //L PH RH T
          break;
```

```
    case 16:
            if( aux_canal < 4 || aux_canal > 4) aux_canal = 4; //N
            break;
    case 17:
            if( aux_canal > 4 ) aux_canal = 0;                      //N T
            if( aux_canal > 0 ) aux_canal = 4;
            break;
    case 18:
            if( aux_canal < 1 || aux_canal > 4) aux_canal = 1;  //N RH
            if( aux_canal > 1 ) aux_canal = 4;
            break;
    case 19:
            if( aux_canal > 4) aux_canal = 0;                       // N RH T
            if( aux_canal > 1 ) aux_canal = 4;
            break;
    case 20:
            if( aux_canal < 2 || aux_canal > 4) aux_canal = 2; //N PH
            if( aux_canal == 3 ) aux_canal = 4;
            break;
    case 21:
            if( aux_canal > 4) aux_canal = 0;                       //N PH T
            if( aux_canal == 1 ) aux_canal = 2;
            if( aux_canal == 3 ) aux_canal = 4;
            break;
    case 22:
            if( aux_canal < 1 || aux_canal > 4) aux_canal = 1;  // N PH RH
            if( aux_canal == 3 ) aux_canal = 4;
            break;
    case 23:
            if( aux_canal > 4) aux_canal = 0;                       // N PH RH T
            if( aux_canal == 3 ) aux_canal = 4;
            break;
    case 24:
            if( aux_canal < 3 || aux_canal > 4) aux_canal = 3; // N L
            break;
    case 25:
            if( aux_canal > 4) aux_canal = 0;                       // N L T
            if( aux_canal == 1 ) aux_canal = 3;
            break;
    case 26:
            if( aux_canal < 1 || aux_canal > 4) aux_canal = 1;  // N L RH
            if( aux_canal == 2 ) aux_canal = 3;
            break;
    case 27:
            if( aux_canal > 4) aux_canal = 0;                       // N L RH T
            if( aux_canal == 2 ) aux_canal = 3;
            break;
    case 28:
            if( aux_canal < 2 || aux_canal > 4) aux_canal = 2;  // N L PH
            break;
    case 29:
            if( aux_canal > 4) aux_canal = 0;                       // N L PH T
            if( aux_canal == 1 ) aux_canal = 2;
            break;
    case 30:
            if( aux_canal < 1 || aux_canal > 4) aux_canal = 1;  // N L PH RH
            break;
    case 31:
            if( aux_canal > 4 ) aux_canal = 0;                      // Todos los sensores habilitados
            break;
    default:
            if( aux_canal > 4 ) aux_canal = 0;                      // Todos los sensores habilitados
            break;
}
    set_adc_channel( aux_canal );
    delay_us(10);

    lcd_gotoxy(1,1);

    write_eeprom( 0x01, aux_canal );
```

```
      switch( aux_canal )
      {
        case 0:
            printf( lcd_putc,"TEMPERATURA     " );
            break;

        case 1:
            printf( lcd_putc,"HUMEDAD RELATIVA" );
            break;

        case 2:
            printf( lcd_putc,"PH              " );
            break;

        case 3:
            printf( lcd_putc,"LUMINOSIDAD     " );
            break;

        case 4:
            printf( lcd_putc,"NIVEL DE AGUA   " );
            break;
      }
    }

    tiempo_boton();
}

//Función de control del backlight del LCD, para
//dejarlo encendido sólo por 2 minutos y medio
#INT_TIMER0
void lcd_backlight()
{

    clear_interrupt( INT_TIMER0 );
    set_timer0( 0 );

    aux_timer += 1;

    if( aux_timer == 20 )
    {
      aux_timer = 0;
      output_low( BACKLIGHT );
      aux_selecciona_luz = 0;
      disable_interrupts( INT_TIMER0 );
    }
}

//Función de recepción de datos por la UART del PIC
#INT_RDA
void recibe()
{
    //Recibimos el datos para luego compararlo
    int8 recibe_dato = getc();

    switch( recibe_dato )
    {
      //Enviamos el valor de temperatura hacia HERVA V2.1
      case '0':
            set_adc_channel( 0 );
            delay_us(10);
            aux_sensor = read_adc()/10.23;
            printf( "%3.2f\r\n", aux_sensor );
            set_adc_channel( aux_canal );
            break;
```

```
    //Enviamos el valor de humedad relativa hacia HERVA V2.1
    case '1':
            set_adc_channel( 1 );
            delay_us(10);
            aux_sensor = read_adc()/10.23;
            printf( "%3.2f\r\n", aux_sensor );
            set_adc_channel( aux_canal );
            break;

    //Enviamos el valor de PH hacia HERVA V2.1
    case '2':
            set_adc_channel( 2 );
            delay_us(10);
            aux_sensor = read_adc()*14.0/1023;
            printf( "%3.2f\r\n", aux_sensor );
            set_adc_channel( aux_canal );
            break;

    //Enviamos el valor de luminosidad hacia HERVA V2.1
    case '3':
            set_adc_channel( 3 );
            delay_us(10);
            aux_sensor = read_adc()/1.023;
            printf( "%3.2f\r\n", aux_sensor );
            set_adc_channel( aux_canal );
            break;

    //Enviamos el valor de Nivel de agua hacia HERVA V2.1
    case '4':
            set_adc_channel( 4 );
            delay_us(10);
            aux_sensor = (read_adc()/0.1023/9800)* pi * radio_m * 1000; //Litros
            printf( "%f\r\n", aux_sensor );
            set_adc_channel( aux_canal );
            break;

    //Obtenemos la habilitación de los sensores
    case 'h':
            gets(cadena);
            recibe_seleccion = atoi(cadena);
            printf( "h\r\n");
            break;

    //Obtenemos el radio del recipiente contenedor de agua
    case 'e':
            gets(aux_nivel);
            radio = atoi(aux_nivel);
            radio_m = radio/100.0; // convertimos centímetros a metros...
            radio_m = pow(radio_m,2); //elevamos radio al cuadrado...
            aux_nivel_bool = 1;
            printf( "h\r\e");
            break;
    default:
            break;
  }
}
```

ANEXO B

Programación de la base de datos de HERVA V2.1

```
//Espacios de nombres generados por default, necesarios para organizar las múltiples
//clases de visual C#
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

//Espacios de nombres para establecer comunicación 232 entre HERVA y la tarjeta
//SAD
using System.IO.Ports;

//Espacios de nombres necesario para exportar tabla
using System.Collections;

namespace WindowsFormsApplication1
{
    public partial class Forma_principal : Form
    {

        //Variables usadas para la generación de las graficas de las variables
        public static string[] xValor;
        public static double[] yValor_temperatura;
        public static double[] yValor_humedad;
        public static double[] yValor_ph;
        public static double[] yValor_luminosidad;
        public static double[] yValor_nivel;

        //Variables usadas para establecer la comunicación 232 entre HERVA y la
        //tarjeta SAD
        string RxString;

        //Variables usadas para el control de la visualización de la tabla
        public static int aux_canal = 0;
        bool aux_primera_vez = false;
        bool aux_muestra_tabla = false;
        bool aux_toma_dato = false;
        Int32 fila = 0;

        //Variables usadas para el desbordamiento del temporizador
        int hr1;
        int min1;
        int seg1;
        int hr2;
        int min2;
        int seg2;

        //Función de inicialización de la forma principal
        public Forma_principal()
        {
            InitializeComponent();
        }
```

```
//Función del evento click del menú nuevo
private void menu_archivo_nuevo_Click(object sender, EventArgs e)
{
    if (habilita_sensores.aux_listo == true)
    {
        toolStripStatusLabel_cronometro.Text = "";
        boton_comenzar.Enabled = true;
        boton_graficar.Enabled = false;
        boton_guardar.Enabled = false;
        text_temperatura.Clear();
        text_humedad.Clear();
        text_PH.Clear();
        text_luminosidad.Clear();
        text_nivel.Clear();
        seg1 = 0;
        seg2 = 0;
        min1 = 0;
        min2 = 0;
        hr1 = 0;
        hr2 = 0;
        aux_canal = 0;
        fila = 0;
        DataGridView_tabla.Rows.Clear();
        menu_config_opcion.Enabled = true;
        habilita_sensores.aux_listo = false;
        if (aux_muestra_tabla == false)
        {
            aux_muestra_tabla = true;
            DataGridView_tabla.Visible = false;
            pictureBox_planta.Visible = true;
        }
        if (puerto_serial.IsOpen)
        {
            try
            {
                puerto_serial.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
        MessageBox.Show("Nuevo proyecto listo!");
    }
    else
        MessageBox.Show("Debe configurar \nel sistema...");
}

//Función del evento click del menú Abrir
private void menu_archivo_abrir_Click(object sender, EventArgs e)
{
    OpenFileDialog ff = new OpenFileDialog();

    ff.Filter = "Png files (*.Png)|*.Png|All files (*.*)|*.*";

    ff.RestoreDirectory = true;

        if (ff.ShowDialog() == DialogResult.OK)
            if (ff.FileName != "")
            {
                Process.Start(ff.FileName);
            }
}
```

```
//Función del evento click del menú salir
private void menu_archivo_salir_Click(object sender, EventArgs e)
{
    if (puerto_serial.IsOpen) puerto_serial.Close();    //Cerramos puerto COM
    timer_tiempo_muestreo.Enabled = false;              //Deshabilitamos el temporizador
    Close();                                            //Cerramos forma principal
}

//Función del evento click del menú opciones
private void menu_config_opcion_Click(object sender, EventArgs e)
{
    habilita_sensores.aux_temperatura = false;
    habilita_sensores.aux_humedad = false;
    habilita_sensores.aux_PH = false;
    habilita_sensores.aux_luminosidad = false;
    habilita_sensores.aux_nivel = false;

    habilita_sensores tercer_form = new habilita_sensores();
    tercer_form.Show();
}

//Función del evento click del menú acerca de
private void menu_ayuda_acerca_de_Click(object sender, EventArgs e)
{
//Muestra un cuadro de diálogo con información general de HERVA
    MessageBox.Show("Herva Ver.X\nDavid Enrique Matías…
    Hernández\nFrancisco Arturo Montesinos Gonzalez…
    \nenriquematias@live.com.mx\nblack_piano@hotmail.com\nITTG 2011");
}

//Función del evento click del menú ver ayuda
private void menu_ayuda_ver_Click(object sender, EventArgs e)
{
    Process.Start("TUTORIAL.doc");
}

//Función del evento click del botón comenzar
private void boton_comenzar_Click(object sender, EventArgs e)
{
//Condición para comenzar la temporización de muestreo
    if (habilita_sensores.aux_listo == true)
    {
        //Cargamos el nombre del puerto COM seleccionado
        if (puerto_serial.IsOpen) puerto_serial.Close();
        else puerto_serial.PortName = habilita_sensores.aux_com;

        if (timer_tiempo_muestreo.Enabled == true)
        {
            timer_tiempo_muestreo.Enabled = false;


            if (puerto_serial.IsOpen)                   //Cerramos puerto serial
            {
                try
                {
                    puerto_serial.Close();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }
```

```
            //Condición de pausa o continuación de la temporización
            if (aux_primera_vez == false) boton_comenzar.Text = "Comenzar";
            else boton_comenzar.Text = "Continuar";

            boton_graficar.Enabled = true;
            boton_guardar.Enabled = true;
            menu_archivo_nuevo.Enabled = true;
        }
        else
        {
            timer_tiempo_muestreo.Enabled = true;
            boton_comenzar.Text = "Parar";
            aux_primera_vez = true;
            aux_muestra_tabla = true;
            menu_archivo_nuevo.Enabled = false;
            menu_config_opcion.Enabled = false;

            //Switch para optimizar el el muestreo de los datos por primera vez
            //con todas las combinaciones posibles de los sensores
            switch (habilita_sensores.aux_habilita_sensores)
            {
                case 1 :
                    aux_canal = 0;
                    break;
                case 2:
                    aux_canal = 1;
                    break;
                case 3:
                    aux_canal = 0;
                    break;
                case 4:
                    aux_canal = 2;
                    break;
                case 5:
                    aux_canal = 0;
                    break;
                case 6:
                    aux_canal = 1;
                    break;
                case 7:
                    aux_canal = 0;
                    break;
                case 8:
                    aux_canal = 3;
                    break;
                case 9:
                    aux_canal = 0;
                    break;
                case 10:
                    aux_canal = 1;
                    break;
                case 11:
                    aux_canal = 0;
                    break;
                case 12:
                    aux_canal = 2;
                    break;
                case 13:
                    aux_canal = 0;
                    break;
                case 14:
                    aux_canal = 1;
                    break;
                case 15:
                    aux_canal = 0;
                    break;
                case 16:
                    aux_canal = 4;
                    break;
```

```
                    case 17:
                        aux_canal = 0;
                        break;
                    case 18:
                        aux_canal = 1;
                        break;
                    case 19:
                        aux_canal = 0;
                        break;
                    case 20:
                        aux_canal = 2;
                        break;
                    case 21:
                        aux_canal = 0;
                        break;
                    case 22:
                        aux_canal = 1;
                        break;
                    case 23:
                        aux_canal = 0;
                        break;
                    case 24:
                        aux_canal = 3;
                        break;
                    case 25:
                        aux_canal = 0;
                        break;
                    case 26:
                        aux_canal = 1;
                        break;
                    case 27:
                        aux_canal = 0;
                        break;
                    case 28:
                        aux_canal = 2;
                        break;
                    case 29:
                        aux_canal = 0;
                        break;
                    case 30:
                        aux_canal = 1;
                        break;
                    case 31:
                        aux_canal = 0;
                        break;
                }

                //Abrimos puerto COM para el envío y recepción de los datos
                if(!puerto_serial.IsOpen)
                {
                    try
                    {
                        puerto_serial.Open();
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show(ex.Message);
                    }
                }
            }
        }
        else
        {
            //En caso de haber ejecutado por primera vez a HERVA tendrá que ir a
            //Configuración->Opciones de lo contrario aparecerá el siguiente cuadro
            //de diálogo:
            MessageBox.Show("Debe configurar \nel sistema...");
        }
    }
```

```
//Función de la visualización, registro de los datos
private void obtiene_dato(object sender, EventArgs e)
{

    switch (aux_canal)
    {
        case 0:

            //Limpiamos el TextBox de temperatura, e insertamos el nuevo valor obtenido
            text_temperatura.Clear();
            text_temperatura.Text = RxString;

            if (habilita_sensores.aux_habilita_sensores == 1)
            {
                aux_toma_dato = false;

                //Agregamos filas con los nuevos datos recibidos
                DataGridView_tabla.Rows.Add( DateTime.Now.ToLongTimeString(),…
                text_temperatura.Text );
                fila += 1;

                //Habilitamos la visualización de la tabla
                if (aux_muestra_tabla == true)
                {
                    aux_muestra_tabla = false;

                    DataGridView_tabla.Visible = true;
                    pictureBox_planta.Visible = false;
                }
            }

            break;

        case 1:
            text_humedad.Clear();
            text_humedad.Text = RxString;

            if (habilita_sensores.aux_habilita_sensores == 2 || …
            habilita_sensores.aux_habilita_sensores == 3)
            {
                aux_toma_dato = false;

                DataGridView_tabla.Rows.Add( DateTime.Now.ToLongTimeString(),…
                text_temperatura.Text );
                fila += 1;

                if (aux_muestra_tabla == true)
                {
                    aux_muestra_tabla = false;

                    DataGridView_tabla.Visible = true;
                    pictureBox_planta.Visible = false;
                }
            }

            break;

        case 2:
            text_PH.Clear();
            text_PH.Text = RxString;

            if (habilita_sensores.aux_habilita_sensores >= 4 && …
            habilita_sensores.aux_habilita_sensores <= 7)
            {
                aux_toma_dato = false;

                DataGridView_tabla.Rows.Add(DateTime.Now.ToLongTimeString(),…
                text_temperatura.Text, text_humedad.Text, text_PH.Text);
```

```
                fila += 1;

                if (aux_muestra_tabla == true)
                {
                    aux_muestra_tabla = false;

                    DataGridView_tabla.Visible = true;
                    pictureBox_planta.Visible = false;
                }

            }

            break;

        case 3:
            text_luminosidad.Clear();
            text_luminosidad.Text = RxString;

            if (habilita_sensores.aux_habilita_sensores >= 8 && …
            habilita_sensores.aux_habilita_sensores <= 15)
            {
                aux_toma_dato = false;

                DataGridView_tabla.Rows.Add(DateTime.Now.ToLongTimeString(), …
                text_temperatura.Text, text_humedad.Text, text_PH.Text, …
                text_luminosidad.Text );
                fila += 1;

                if (aux_muestra_tabla == true)
                {
                    aux_muestra_tabla = false;

                    DataGridView_tabla.Visible = true;
                    pictureBox_planta.Visible = false;
                }
            }
            break;

        case 4:
            text_nivel.Clear();
            text_nivel.Text = RxString;

            if (habilita_sensores.aux_habilita_sensores >= 16 && …
            habilita_sensores.aux_habilita_sensores <=31)
            {
                aux_toma_dato = false;

                DataGridView_tabla.Rows.Add(DateTime.Now.ToLongTimeString(), …
                text_temperatura.Text, text_humedad.Text, text_PH.Text, …
                text_luminosidad.Text, text_nivel.Text);
                fila += 1;

                if (aux_muestra_tabla == true)
                {
                    aux_muestra_tabla = false;

                    DataGridView_tabla.Visible = true;
                    pictureBox_planta.Visible = false;
                }
            }
            break;
    }
```

```
//Incrementamos la variable de conmutación de variables
aux_canal += 1;
//Control del incremento de la variable de conmutación
switch (habilita_sensores.aux_habilita_sensores)
{
    case 1:
        if (aux_canal > 0) aux_canal = 0;                   // T
        break;
    case 2:
        if (aux_canal < 1 || aux_canal > 1) aux_canal = 1;  // RH
        break;
    case 3:
        if (aux_canal > 1) aux_canal = 0;                   // T RH
        break;
    case 4:
        if (aux_canal < 2 || aux_canal > 2) aux_canal = 2;  // PH
        break;
    case 5:
        if (aux_canal < 1 || aux_canal > 2) aux_canal = 0;  // PH T
        if (aux_canal == 1) aux_canal = 2;
        break;
    case 6:
        if (aux_canal < 1 || aux_canal > 2) aux_canal = 1;  // PH RH
        break;
    case 7:
        if (aux_canal > 2) aux_canal = 0;                   // PH RH T
        break;
    case 8:
        if (aux_canal < 3 || aux_canal > 3) aux_canal = 3;  // L
        break;
    case 9:
        if (aux_canal > 3) aux_canal = 0;                   // L T
        if (aux_canal > 0) aux_canal = 3;
        break;
    case 10:
        if (aux_canal < 1 || aux_canal > 3) aux_canal = 1;  // L H
        if (aux_canal == 2) aux_canal = 3;
        break;
    case 11:
        if (aux_canal > 3) aux_canal = 0;                   // L H T
        if (aux_canal == 2) aux_canal = 3;
        break;
    case 12:
        if (aux_canal < 2 || aux_canal > 3) aux_canal = 2;  // L PH
        break;
    case 13:
        if (aux_canal > 3) aux_canal = 0;                   // L PH T
        if (aux_canal == 1) aux_canal = 2;
        break;
    case 14:
        if (aux_canal < 1 || aux_canal > 3) aux_canal = 1;  // L PH T
        break;
    case 15:
        if (aux_canal > 3) aux_canal = 0;                   // L PH RH T
        break;
    case 16:
        if (aux_canal < 4 || aux_canal > 4) aux_canal = 4;  // N
        break;
    case 17:
        if (aux_canal > 4) aux_canal = 0;                   // N T
        if (aux_canal > 0) aux_canal = 4;
        break;
    case 18:
        if (aux_canal < 1 || aux_canal > 4) aux_canal = 1;  // N RH
        if (aux_canal > 1) aux_canal = 4;
        break;
    case 19:
        if (aux_canal > 4) aux_canal = 0;                   // N RH T
        if (aux_canal > 1) aux_canal = 4;
```

```
                break;
            case 20:
                if (aux_canal < 2 || aux_canal > 4) aux_canal = 2;   // N PH
                if (aux_canal == 3) aux_canal = 4;
                break;
            case 21:
                if (aux_canal > 4) aux_canal = 0;                    // N PH T
                if (aux_canal == 1) aux_canal = 2;
                if (aux_canal == 3) aux_canal = 4;
                break;
            case 22:
                if (aux_canal < 1 || aux_canal > 4) aux_canal = 1;   // N PH RH
                if (aux_canal == 3) aux_canal = 4;
                break;
            case 23:
                if (aux_canal > 4) aux_canal = 0;                    // N PH RH T
                if (aux_canal == 3) aux_canal = 4;
                break;
            case 24:
                if (aux_canal < 3 || aux_canal > 4) aux_canal = 3;   // N L
                break;
            case 25:
                if (aux_canal > 4) aux_canal = 0;                    // N L T
                if (aux_canal == 1) aux_canal = 3;
                break;
            case 26:
                if (aux_canal < 1 || aux_canal > 4) aux_canal = 1;   // N L RH
                if (aux_canal == 2) aux_canal = 3;
                break;
            case 27:
                if (aux_canal > 4) aux_canal = 0;                    // N L RH T
                if (aux_canal == 2) aux_canal = 3;
                break;
            case 28:
                if (aux_canal < 2 || aux_canal > 4) aux_canal = 2;   // N L PH
                break;
            case 29:
                if (aux_canal > 4) aux_canal = 0;                    // N L PH T
                if (aux_canal == 1) aux_canal = 2;
                break;
            case 30:
                if (aux_canal < 1 || aux_canal > 4) aux_canal = 1;   // N L PH RH
                break;
            case 31:
                if (aux_canal > 4) aux_canal = 0;                    //TODOS HABILITADOS
                break;
        }
    }

    //Función del evento de recepción de datos del control SerialPort
    private void puerto_serial_DataReceived(object sender, SerialDataReceivedEventArgs e)
    {

        if (timer_tiempo_muestreo.Enabled == true)
            RxString = puerto_serial.ReadLine();
        puerto_serial.DiscardInBuffer();

        this.Invoke(new EventHandler(obtiene_dato));
    }

    //Función del evento click del botón graficar
    private void boton_graficar_Click(object sender, EventArgs e)
    {

        int aux_cuenta_filas = DataGridView_tabla.Rows.Count - 1;    //Cuenta numero de filas
        int j;
        xValor = new string[aux_cuenta_filas];
        for (int i = 0; i < aux_cuenta_filas; i++ )
        {
```

```
            //Almacenamiento de los datos de las filas totales de la primer columna
            xValor[i] = DataGridView_tabla[0, i].Value.ToString();
        }

        if (habilita_sensores.aux_temperatura == true)
        {
            yValor_temperatura = new double[aux_cuenta_filas];
            for (j = 0; j < aux_cuenta_filas; j++)
            {
                yValor_temperatura[j] = Convert.ToDouble(DataGridView_tabla[1, …
                j].Value.ToString());
            }
        }

        if (habilita_sensores.aux_humedad == true)
        {
            yValor_humedad = new double[aux_cuenta_filas];
            for (j = 0; j < aux_cuenta_filas; j++)
            {
                yValor_humedad[j] = Convert.ToDouble(DataGridView_tabla[2, …
                j].Value.ToString());
            }
        }

        if (habilita_sensores.aux_PH == true)
        {
            yValor_ph = new double[aux_cuenta_filas];
            for (j = 0; j < aux_cuenta_filas; j++)
            {
                yValor_ph[j] = Convert.ToDouble(DataGridView_tabla[3, j].Value.ToString());
            }
        }

        if (habilita_sensores.aux_luminosidad == true)
        {
            yValor_luminosidad = new double[aux_cuenta_filas];
            for (j = 0; j < aux_cuenta_filas; j++)
            {
                yValor_luminosidad[j] = Convert.ToDouble(DataGridView_tabla[4, …
                j].Value.ToString());
            }
        }

        if (habilita_sensores.aux_nivel == true)
        {
            yValor_nivel = new double[aux_cuenta_filas];
            for (j = 0; j < aux_cuenta_filas; j++)
            {
                yValor_nivel[j] = Convert.ToDouble(DataGridView_tabla[5, …
                j].Value.ToString());
            }
        }

        forma_grafica segundo_form = new forma_grafica();
        segundo_form.Show();
    }

    //Función del evento clic del Botón guardar
    private void boton_guardar_Click(object sender, EventArgs e)
    {
        try
        {

            //Variables para la exportación de la tabla de datos
            ArrayList titulos = new ArrayList();
            DataTable datosTabla = new DataTable();
```

```
            //Creamos un objeto SaveFileDialog para preguntar al usuario una
            //ubicación para guardar el archivo
            SaveFileDialog ff = new SaveFileDialog();

            //Tipos de formato de almacenamiento de dato
            ff.Filter = "csv files (*.csv)|*.csv|html files (*.html)|*.html|doc files …
            (*.doc)|*.doc|xls files (*.xls)|*.xls|All files (*.*)|*.*";

            ff.RestoreDirectory = true;

            if (ff.ShowDialog() == DialogResult.OK)
            {
                if (ff.FileName != "")
                {

                    OtrosFormatos OF = new OtrosFormatos(ff.FileName.ToString());

                    //Obtenemos los títulos del DataGridView y creamos las columnas
                    //de la tabla
                    foreach (DataGridViewColumn item in DataGridView_tabla.Columns)
                    {
                        titulos.Add(item.HeaderText);
                        datosTabla.Columns.Add();
                    }

                    //Se crean los renglones de la tabla
                    foreach (DataGridViewRow item in DataGridView_tabla.Rows)
                    {
                        DataRow rowx = datosTabla.NewRow();
                        datosTabla.Rows.Add(rowx);
                    }

                    //Se pasan los datos del dataGridView a la tabla
                    foreach (DataGridViewColumn item in DataGridView_tabla.Columns)
                    {
                        foreach (DataGridViewRow itemx in DataGridView_tabla.Rows)
                        {
                            datosTabla.Rows[itemx.Index][item.Index] = …
                            DataGridView_tabla[item.Index, itemx.Index].Value;
                        }
                    }

                    //Se exportan los datos hacia la clase OtrosFormatos
                    if(ff.FilterIndex == 1)OF.ExportCSV(titulos, datosTabla);
                    if(ff.FilterIndex > 1) OF.Export(titulos, datosTabla);
                    Process.Start(OF.xpath);
                    MessageBox.Show("Tabla almacenada");
                }
            }

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
```

```
//Función del evento de desbordamiento del temporizador para la toma de datos y
//comparación del tiempo de muestreo
private void timer_tiempo_muestreo_Tick_1(object sender, EventArgs e)
{
    if (Convert.ToInt32(habilita_sensores.hora1) == hr1)
    {
        if (Convert.ToInt32(habilita_sensores.min1) == min1)
        {
            if (Convert.ToInt32(habilita_sensores.seg1) == seg1)
            {
                aux_toma_dato = false;
                timer_tiempo_muestreo.Enabled = false;

                boton_comenzar.Text = "Comenzar";
                boton_comenzar.Enabled = false;
                boton_graficar.Enabled = true;
                boton_guardar.Enabled = true;

                menu_archivo_nuevo.Enabled = true;

                habilita_sensores.aux_listo = true;
                MessageBox.Show("Muestreo realizado");
            }
        }
    }

    //Visualización del tiempo transcurrido en el control StatusStrip
    toolStripStatusLabel_cronometro.Text = "       Tiempo transcurrido:   " + …
    hr1.ToString() + ":" + min1.ToString() + ":" + seg1.ToString();

    if (Convert.ToInt32(habilita_sensores.hora2) == hr2)
    {
        if (Convert.ToInt32(habilita_sensores.min2) == min2)
        {
            if (Convert.ToInt32(habilita_sensores.seg2) == seg2)
            {
                seg2 = 0;
                min2 = 0;
                hr2 = 0;

                aux_toma_dato = true;
            }
        }
    }


    //Switch de petición de los datos cada segundo por variable
    if (aux_toma_dato == true)
    {
        switch (aux_canal)
        {
            case 0:
                text_temperatura.Clear();
                puerto_serial.Write("0");
                break;

            case 1:
                text_humedad.Clear();
                puerto_serial.Write("1");
                break;

            case 2:
                text_PH.Clear();
                puerto_serial.Write("2");
                break;
```

```
                case 3:
                    text_luminosidad.Clear();
                    puerto_serial.Write("3");
                    break;

                case 4:
                    text_nivel.Clear();
                    puerto_serial.Write("4");
                    break;
            }
        }

        //Incremento de las variables del temporizador cada segundo
        seg1 += 1;
        seg2 += 1;

        if (seg1 == 60)
        {
            seg1 = 0;
            min1 += 1;
        }

        if (min1 == 60)
        {
            min1 = 0;
            hr1 += 1;
        }

        if (seg2 == 60)
        {
            seg2 = 0;
            min2 += 1;
        }

        if (min2 == 60)
        {
            min2 = 0;
            hr2 += 1;
        }
    }

    private void DataGridView_tabla_RowsAdded(object sender, DataGridViewRowsAddedEventArgs e)
    {
        if (fila > 0 && (DataGridView_tabla[0, fila].Value.ToString() == …
        DataGridView_tabla[0, fila - 1].Value.ToString()))
        {
            DataGridView_tabla.Rows.RemoveAt(fila);
            fila-=1;
        }
    }

    //Función del evento para cerrar la forma principal
    private void Forma_principal_FormClosing(object sender, FormClosingEventArgs e)
    {
        if (puerto_serial.IsOpen) puerto_serial.Close();
        timer_tiempo_muestreo.Enabled = false;
    }

}
}
```

Programación de la forma de gráficas de HERVA V2.1.

```
//Espacios de nombres generados por default, necesarios para organizar las múltiples
//clases de visual C#

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

//Espacios de nombres para los métodos y propiedades para el manejo del control Chart
using System.Windows.Forms.DataVisualization.Charting;

//Espacios de nombres para realizar la exportación de datos
using System.IO;

namespace WindowsFormsApplication1
{
    public partial class forma_grafica : Form
    {
        int aux_muestra_tabla = 0;

        //Función de inicialización de la forma de gráficas
        public forma_grafica()
        {

            InitializeComponent();
        }

        //Función del evento de carga de la forma de gráficas
        private void forma_grafica_Load(object sender, EventArgs e)
        {

            //Condición de habilitación para dibujar la gráfica
            if (habilita_sensores.aux_temperatura == true)
            {
               chart_grafica_temperatura.Series["°C"].Points.DataBindXY(Forma_principal.xValor,…
            ...Forma_principal.yValor_temperatura);
                temperaturaToolStripMenuItem.Visible = true;
            }

            if (habilita_sensores.aux_humedad == true)
            {
                chart_grafica_humedad.Series["%RH"].Points.DataBindXY(Forma_principal.xValor,…
                Forma_principal.yValor_humedad);
                humedadRelativaToolStripMenuItem.Visible = true;
            }

            if (habilita_sensores.aux_PH == true)
            {
                chart_grafica_PH.Series["PH"].Points.DataBindXY(Forma_principal.xValor,…
                Forma_principal.yValor_ph);
                pHToolStripMenuItem.Visible = true;
            }

            if (habilita_sensores.aux_luminosidad == true)
            {
              Chart_grafica_luminosidad.Series["Lux"].Points.DataBindXY(Forma_principal.xValor,…
              Forma_principal.yValor_luminosidad);
              luminosidadToolStripMenuItem.Visible = true;
            }
```

```
        if (habilita_sensores.aux_nivel == true)
        {
            chart_grafica_nivel.Series["Litros"].Points.DataBindXY(Forma_principal.xValor,…
            Forma_principal.yValor_nivel);
            nivelDeAguaToolStripMenuItem.Visible = true;
        }
    }

    //Función del evento click del menú Salir
    private void salirToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Close();
    }

    //Función del evento click del menú Guardar
    private void guardarToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Stream myStream;
        SaveFileDialog ff = new SaveFileDialog();

        //Generamos objeto SaveFileDialog para almacenamiento de las gráficas
        ff.Filter = "Png files (*.Png)|*.Png|All files (*.*)|*.*";
        ff.FilterIndex = 1;
        ff.RestoreDirectory = true;

        if (ff.ShowDialog() == DialogResult.OK)
        {
            if ((myStream = ff.OpenFile()) != null)
            {
                using (myStream)
                {
                    switch (aux_muestra_tabla)
                    {
                        case 1:
                                chart_grafica_temperatura.SaveImage(myStream, …
                                System.Drawing.Imaging.ImageFormat.Png);
                                break;
                        case 2:
                                chart_grafica_humedad.SaveImage(myStream, …
                                System.Drawing.Imaging.ImageFormat.Png);
                                break;
                        case 3:
                                chart_grafica_PH.SaveImage(myStream, …
                                System.Drawing.Imaging.ImageFormat.Png);
                                break;
                        case 4:
                                chart_grafica_luminosidad.SaveImage(myStream, …
                                System.Drawing.Imaging.ImageFormat.Png);
                                break;
                        case 5:
                                chart_grafica_nivel.SaveImage(myStream, …
                                System.Drawing.Imaging.ImageFormat.Png);
                                break;
                    }
                }
            }
        }
    }
```

```
        //Función del evento click del menú mostrar gráfica de temperatura
        private void temperaturaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            chart_grafica_temperatura.Visible = true;
            chart_grafica_humedad.Visible = false;
            chart_grafica_PH.Visible = false;
            chart_grafica_luminosidad.Visible = false;
            chart_grafica_nivel.Visible = false;

            pictureBox_planta.Visible = false;

            aux_muestra_tabla = 1;
        }
        //Función del evento click del menú mostrar gráfica de humedad relativa
        private void humedadRelativaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            chart_grafica_temperatura.Visible = false;
            chart_grafica_humedad.Visible = true;
            chart_grafica_PH.Visible = false;
            chart_grafica_luminosidad.Visible = false;
            chart_grafica_nivel.Visible = false;

            pictureBox_planta.Visible = false;

            aux_muestra_tabla = 2;
        }
        //Función del evento click del menú mostrar gráfica de PH
        private void pHToolStripMenuItem_Click(object sender, EventArgs e)
        {
            chart_grafica_temperatura.Visible = false;
            chart_grafica_humedad.Visible = false;
            chart_grafica_PH.Visible = true;
            chart_grafica_luminosidad.Visible = false;
            chart_grafica_nivel.Visible = false;

            pictureBox_planta.Visible = false;

            aux_muestra_tabla = 3;
        }
        //Función del evento click del menú mostrar gráfica de luminosidad
        private void luminosidadToolStripMenuItem_Click(object sender, EventArgs e)
        {
            chart_grafica_temperatura.Visible = false;
            chart_grafica_humedad.Visible = false;
            chart_grafica_PH.Visible = false;
            chart_grafica_luminosidad.Visible = true;
            chart_grafica_nivel.Visible = false;

            pictureBox_planta.Visible = false;

            aux_muestra_tabla = 4;
        }

        //Función del evento click del menú mostrar gráfica de nivel de agua
        private void nivelDeAguaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            chart_grafica_temperatura.Visible = false;
            chart_grafica_humedad.Visible = false;
            chart_grafica_PH.Visible = false;
            chart_grafica_luminosidad.Visible = false;
            chart_grafica_nivel.Visible = true;

            pictureBox_planta.Visible = false;

            aux_muestra_tabla = 5;
        }
    }
}
```

Programación de la forma de configuración de HERVA V2.1

```
//Espacios de nombres generados por default, necesarios para organizar las múltiples
//clases de visual C#
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Management;

//Espacios de nombres para establecer comunicación 232 entre HERVA y la tarjeta
//SAD
using System.IO.Ports;

namespace WindowsFormsApplication1
{
    public partial class habilita_sensores : Form
    {

        //Variables usadas para la habilitación de los sensores y el establecimiento de tiempo
        //de muestreo
        public static decimal hora1;
        public static decimal hora2;
        public static decimal min1;
        public static decimal min2;
        public static decimal seg1;
        public static decimal seg2;

        public static bool aux_listo = false;
        public static bool aux_sensores = false;

        public static string aux_com = "";

        public static bool aux_temperatura = false;
        public static bool aux_humedad = false;
        public static bool aux_PH = false;
        public static bool aux_luminosidad = false;
        public static bool aux_nivel = false;

        public static int aux_habilita_sensores = 0;

        string cadena = string.Empty;

        //Función de inicialización de la forma de configuración
        public habilita_sensores()
        {
            InitializeComponent();
        }

        //Función del evento de carga la forma de configuración
        private void habilita_sensores_Load(object sender, EventArgs e)
        {
            //Obtenemos nombre de los puertos COM disponibles
            string[] puertos = SerialPort.GetPortNames();
            foreach (string port in puertos)
            {
                comboBox_serial.Items.Add(port);
            }
        }
```

```
//Función del evento del despliegue de la lista del control comboBox
private void comboBox_serial_SelectionChangeCommitted(object sender, EventArgs e)
{
    //Obtenemos nombre del puerto COM para enviar la selección de sensores a la tarjeta
    //SAD
    try
    {
        aux_com = comboBox_serial.Items[comboBox_serial.SelectedIndex].ToString();
        puerto_serial.PortName = aux_com;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }

    groupBox_tiempo_muestreo1.Enabled = true;
    groupBox_sensores.Enabled = true;

    //Abrimos puerto serial
    try
    {
        puerto_serial.Open();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }

    groupBox_tiempo_muestreo2.Enabled = true;
    button_listo.Enabled = true;
}

//Función del evento clic del botón listo.
private void button_listo_Click(object sender, EventArgs e)
{
    if (aux_com == "") MessageBox.Show("Debe seleccionar \nun puerto COM");
    else
    {

         //Obtenemos nombres de los sensores habilitados
        if (checkedListBox_seleccion.CheckedItems.Count != 0)
        {
            string aux_sensor_nombre = "";

            for (int x = 0; x <= checkedListBox_seleccion.CheckedItems.Count - 1; x++)
            {
                aux_sensor_nombre = checkedListBox_seleccion.CheckedItems[x].ToString();

                if (aux_sensor_nombre == "Temperatura") aux_temperatura = true;
                if (aux_sensor_nombre == "Humedad Relativa") aux_humedad = true;
                if (aux_sensor_nombre == "PH") aux_PH = true;
                if (aux_sensor_nombre == "Luminosidad") aux_luminosidad = true;
                if (aux_sensor_nombre == "Nivel de agua") aux_nivel = true;
            }
```

```
//Preparamos la habilitación de los sensores con banderas de todas las
//combinaciones posibles
if (!aux_nivel && !aux_luminosidad && !aux_PH && !aux_humedad && …
aux_temperatura) aux_habilita_sensores = 1;
if (!aux_nivel && !aux_luminosidad && !aux_PH &&  aux_humedad && …
!aux_temperatura) aux_habilita_sensores = 2;
if (!aux_nivel && !aux_luminosidad && !aux_PH &&  aux_humedad && …
aux_temperatura) aux_habilita_sensores = 3;
if (!aux_nivel && !aux_luminosidad &&  aux_PH && !aux_humedad && …
!aux_temperatura) aux_habilita_sensores = 4;
if (!aux_nivel && !aux_luminosidad &&  aux_PH && !aux_humedad && …
aux_temperatura) aux_habilita_sensores = 5;
if (!aux_nivel && !aux_luminosidad &&  aux_PH &&  aux_humedad && …
!aux_temperatura) aux_habilita_sensores = 6;
if (!aux_nivel && !aux_luminosidad &&  aux_PH &&  aux_humedad && …
aux_temperatura) aux_habilita_sensores = 7;
if (!aux_nivel &&  aux_luminosidad && !aux_PH && !aux_humedad && …
!aux_temperatura) aux_habilita_sensores = 8;
if (!aux_nivel &&  aux_luminosidad && !aux_PH && !aux_humedad && …
aux_temperatura) aux_habilita_sensores = 9;
if (!aux_nivel &&  aux_luminosidad && !aux_PH &&  aux_humedad && …
!aux_temperatura) aux_habilita_sensores = 10;
if (!aux_nivel &&  aux_luminosidad && !aux_PH &&  aux_humedad && …
aux_temperatura) aux_habilita_sensores = 11;
if (!aux_nivel &&  aux_luminosidad &&  aux_PH && !aux_humedad && …
!aux_temperatura) aux_habilita_sensores = 12;
if (!aux_nivel &&  aux_luminosidad &&  aux_PH && !aux_humedad && …
aux_temperatura) aux_habilita_sensores = 13;
if (!aux_nivel &&  aux_luminosidad &&  aux_PH &&  aux_humedad && …
!aux_temperatura) aux_habilita_sensores = 14;
if (!aux_nivel &&  aux_luminosidad &&  aux_PH &&  aux_humedad && …
aux_temperatura) aux_habilita_sensores = 15;
if ( aux_nivel && !aux_luminosidad && !aux_PH && !aux_humedad && …
!aux_temperatura) aux_habilita_sensores = 16;
if ( aux_nivel && !aux_luminosidad && !aux_PH && !aux_humedad && …
aux_temperatura) aux_habilita_sensores = 17;
if ( aux_nivel && !aux_luminosidad && !aux_PH &&  aux_humedad && …
!aux_temperatura) aux_habilita_sensores = 18;
if ( aux_nivel && !aux_luminosidad && !aux_PH &&  aux_humedad && …
aux_temperatura) aux_habilita_sensores = 19;
if ( aux_nivel && !aux_luminosidad &&  aux_PH && !aux_humedad && …
!aux_temperatura) aux_habilita_sensores = 20;
if ( aux_nivel && !aux_luminosidad &&  aux_PH && !aux_humedad && …
aux_temperatura) aux_habilita_sensores = 21;
if ( aux_nivel && !aux_luminosidad &&  aux_PH &&  aux_humedad && …
!aux_temperatura) aux_habilita_sensores = 22;
if ( aux_nivel && !aux_luminosidad &&  aux_PH &&  aux_humedad && …
aux_temperatura) aux_habilita_sensores = 23;
if ( aux_nivel &&  aux_luminosidad && !aux_PH && !aux_humedad && …
!aux_temperatura) aux_habilita_sensores = 24;
if ( aux_nivel &&  aux_luminosidad && !aux_PH && !aux_humedad && …
aux_temperatura) aux_habilita_sensores = 25;
if ( aux_nivel &&  aux_luminosidad && !aux_PH &&  aux_humedad && …
!aux_temperatura) aux_habilita_sensores = 26;
if ( aux_nivel &&  aux_luminosidad && !aux_PH &&  aux_humedad && …
aux_temperatura) aux_habilita_sensores = 27;
if ( aux_nivel &&  aux_luminosidad &&  aux_PH && !aux_humedad && …
!aux_temperatura) aux_habilita_sensores = 28;
if ( aux_nivel &&  aux_luminosidad &&  aux_PH && !aux_humedad && …
aux_temperatura) aux_habilita_sensores = 29;
if ( aux_nivel &&  aux_luminosidad &&  aux_PH &&  aux_humedad && …
!aux_temperatura) aux_habilita_sensores = 30;
if ( aux_nivel &&  aux_luminosidad &&  aux_PH &&  aux_humedad && …
aux_temperatura) aux_habilita_sensores = 31;

aux_sensores = true;
```

```
//Obtenemos el valor del tiempo total e intervalo de muestreo
hora1 = numericUpDown_hora1.Value;
hora2 = numericUpDown_hora2.Value;

min1 = numericUpDown_min1.Value;
min2 = numericUpDown_min2.Value;

seg1 = numericUpDown_seg1.Value;
seg2 = numericUpDown_seg2.Value;

aux_listo = true;

//Condición de protección del tiempo total e intervalo de muestreo
if ( seg1 <= seg2 && min1 == 0 && hora1 == 0 )
    MessageBox.Show("Debe incrementar el \ntiempo de muestreo...");
else
{
    if (aux_nivel == true)
    {

        //En caso de seleccionar Nivel de agua aparecerá la forma de Nivel
        if (Forma_Nivel.aux_listo_nivel == false)
        {
            try
            {
                puerto_serial.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

            Forma_Nivel cuarto_form = new Forma_Nivel();
            cuarto_form.Show();
        }
        else
        {
            try
            {
                puerto_serial.Open();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

            puerto_serial.Write("h");
            puerto_serial.WriteLine(aux_habilita_sensores.ToString() + "\n\r");

            Forma_Nivel.aux_listo_nivel = false;

            try
            {
                puerto_serial.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

            habilita_sensores.ActiveForm.Close();
        }
    }
    else
    {
```

```
                        //Enviamos la información de habilitación a la tarjeta SAD
                        puerto_serial.Write("h");
                        puerto_serial.WriteLine(aux_habilita_sensores.ToString() + "\n\r");

                        try                             //Cerramos puerto COM
                        {
                            puerto_serial.Close();
                        }
                        catch (Exception ex)
                        {
                            MessageBox.Show(ex.Message);
                        }

                        habilita_sensores.ActiveForm.Close();
                    }
                }
            }
            else
            {
                MessageBox.Show("Debe seleccionar\nal menos un sensor");
            }
        }
    }
}
```

Programación de la forma de nivel de agua de HERVA V2.1

```
//Espacios de nombres generados por default, necesarios para organizar las múltiples
//clases de visual C#
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Forma_Nivel : Form
    {
        public static bool aux_listo_nivel = false;

        //Función de inicialización de la forma de Nivel de agua
        public Forma_Nivel()
        {
            InitializeComponent();
        }

        //Función del evento clic del botón aceptar
        private void button_aceptar_nivel_Click(object sender, EventArgs e)
        {
            if (puerto_serial.IsOpen) puerto_serial.Close();
            else puerto_serial.PortName = habilita_sensores.aux_com;

            try                                         //Abrimos puerto COM
            {
                puerto_serial.Open();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

            if (numericUpDown_cm.Value == 0) MessageBox.Show("Debe ingresar el radio…
            \nde la sección transversal\ndel contenedor de agua");
            else
            {

                //Enviamos la información del radio de la sección transversal del contenedor
                //hacia la tarjeta SAD
                puerto_serial.Write("e");
                puerto_serial.WriteLine(numericUpDown_cm.Value.ToString() + "\n\r");
                aux_listo_nivel = true;
                try                                     //Cerramos puerto COM
                {
                    puerto_serial.Close();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
                Forma_Nivel.ActiveForm.Close();
            }
        }
    }
}
```

Programación de la clase otros formatos de HERVA V2.1

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Collections;
using System.Data;

namespace WindowsFormsApplication1
{
    class OtrosFormatos
    {
        StreamWriter w;
        string ruta;
        public string xpath { get { return ruta; } set { value = ruta; } }

        /// <summary>
        /// Constructor que establece el path del archivo
        /// </summary>
        /// <param name="path"></param>
        public OtrosFormatos(string path)
        {
            ruta = @path;
        }

        /// <summary>
        /// Exporta datos a un archivo
        /// </summary>
        /// <param name="titulos"></param>
        /// <param name="datos"></param>
        public void Export(ArrayList titulos, DataTable datos)
        {
            try
            {
                FileStream fs = new FileStream(ruta, FileMode.Create, FileAccess.ReadWrite);
                w = new StreamWriter(fs);
                string comillas = char.ConvertFromUtf32(34);
                StringBuilder html = new StringBuilder();
                html.Append(@"<!DOCTYPE html PUBLIC" + comillas + "-//W3C//DTD XHTML 1.0 …
                Transitional//EN" + comillas + " " + comillas + …
                "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" + comillas + ">");
                html.Append(@"<html xmlns=" + comillas + "http://www.w3.org/1999/xhtml" + …
                comillas + ">");
                html.Append(@"<head>");
                html.Append(@"<meta http-equiv=" + comillas + "Content-Type" + comillas + …
                "content=" + comillas + "text/html; charset=utf-8" + comillas + " />");
                html.Append(@"<title>Untitled Document</title>");
                html.Append(@"</head>");
                html.Append(@"<body>");

                //Generando encabezados del archivo
                html.Append(@"<table WIDTH=730 CELLSPACING=0 CELLPADDING=10 border=8 …
                BORDERCOLOR=" + comillas + "#333366" + comillas + " bgcolor=" + comillas + …
                "#FFFFFF" + comillas + ">");
                html.Append(@"<tr> <b>");
                foreach (object item in titulos)
                {
                    html.Append(@"<th>" + item.ToString() + "</th>");
                }
                html.Append(@"</b> </tr>");
```

```
                //Generando datos del archivo
                for (int i = 0; i < datos.Rows.Count; i++)
                {
                    html.Append(@"<tr>");
                    for (int j = 0; j < datos.Columns.Count; j++)
                    {
                        html.Append(@"<td>" + datos.Rows[i][j].ToString() + "</td>");
                    }
                    html.Append(@"</tr>");
                }

                html.Append(@"</body>");
                html.Append(@"</html>");

                w.Write(html.ToString());
                w.Close();
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }
        /// <summary>
        /// Genera un archivo CSV
        /// </summary>
        /// <param name="titulos"></param>
        /// <param name="datos"></param>
        public void ExportCSV(ArrayList titulos, DataTable datos)
        {
            try
            {
                FileStream fs = new FileStream(ruta, FileMode.Create, FileAccess.ReadWrite);
                w = new StreamWriter(fs);
                string comillas = char.ConvertFromUtf32(34);
                StringBuilder CSV = new StringBuilder();
                //Encabezados
                for (int i = 0; i < titulos.Count; i++)
                {
                    if (i != (titulos.Count - 1))
                        CSV.Append(comillas + titulos[i].ToString() + comillas + ",");
                    else
                        CSV.Append(comillas + titulos[i].ToString() + comillas + …
                        Environment.NewLine);
                }
                //datos
                for (int i = 0; i < datos.Rows.Count; i++)
                {
                    for (int j = 0; j < datos.Columns.Count; j++)
                    {
                        if (j != (titulos.Count - 1))
                            CSV.Append(comillas + datos.Rows[i][j].ToString() + comillas + ",");
                        else
                            CSV.Append(comillas + datos.Rows[i][j].ToString() + comillas + …
                            Environment.NewLine);
                    }
                }
                w.Write(CSV.ToString());
                w.Close();
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }
    }
}
```

ANEXO C

Tutorial de HERVA

Bienvenido a **HERVA V2.1**

HERVA V2.1 es una aplicación que permite el registro de datos remotamente desde la tarjeta **SAD** hacia el ordenador visualizándolos en una tabla por intervalos de tiempo, ambos definidos por el usuario con el propósito de graficar los datos para una interpretación más eficiente.

Nota: Antes de comenzar a utilizar HERVA V2.1 por primera vez, usted debe descargar e instalar los componentes de Microsoft .NET Framework 4 necesarios para la ejecución en la arquitectura de máquina y el sistema operativo de destino desde el siguiente vínculo:

http://www.microsoft.com/downloads/es-es/details.aspx?FamilyID=9cfb2d51-5ff4-4491-b0e5-b386f32c0992

Lo que usted debe tener en cuenta cada vez que utilice HERVA V2.1 es lo siguiente:

Coloque la tarjeta **SAD** en un lugar seguro y seco (ver **figura 1**).

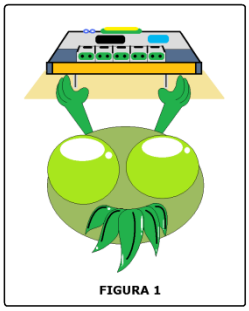


FIGURA 1

Conecte los sensores de sólo en el orden correcto (ver **figura 2**); mediante el uso de desatornilladores podrá realizar la conexión de los sensores hacia la tarjeta **SAD** con el cable de 9 m de longitud (ver **figura 3**).

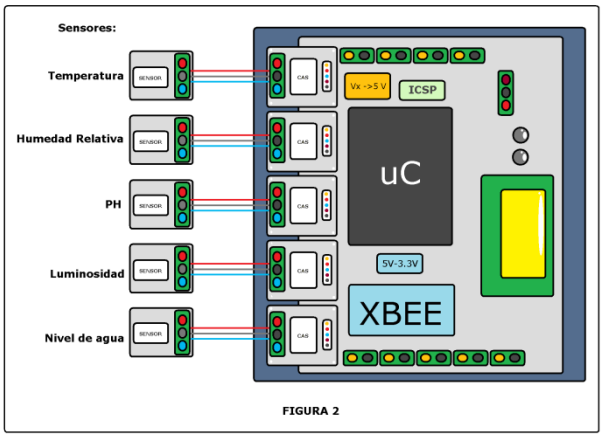


FIGURA 2

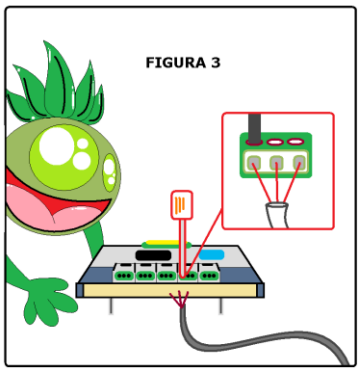


FIGURA 3

Alimente la tarjeta **SAD**, con los niveles de voltaje permitidos (ver **figura 4**).

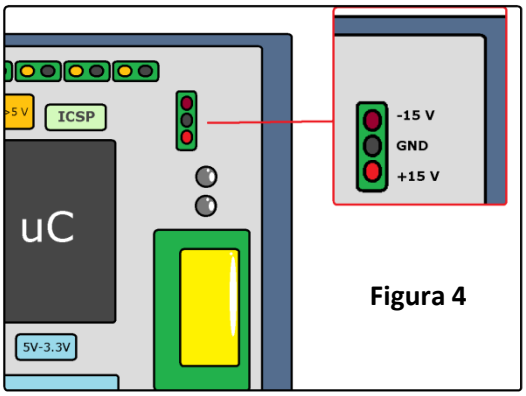
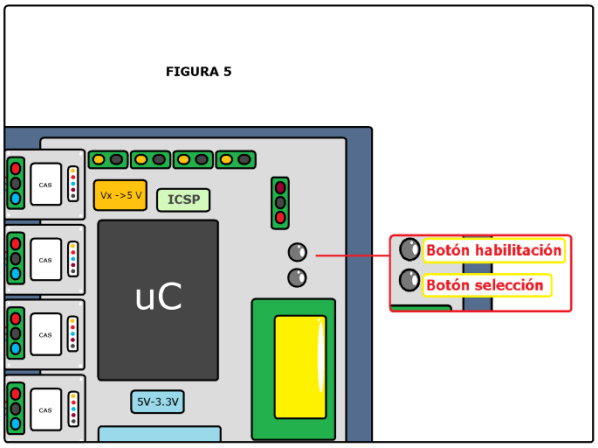


Figura 4

Active el sistema pulsando el botón de habilitación ubicado a un costado del LCD (usted puede cambiar la visualización de las variables usando el botón de selección) (ver **figura 5**).

FIGURA 5



Encienda su PC e inserte el módulo **XBee dongle** en algún puerto USB de su computadora (ver **figura 6**), espere a Windows lo detecte y le asigne un número de puerto COM virtual

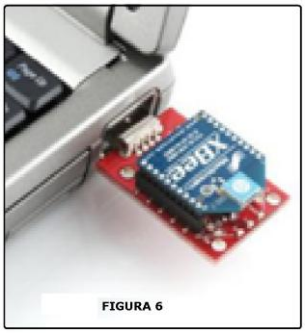


FIGURA 6

Diríjase al menú **Configuración->Opciones** (A) y seleccione el puerto COM con el que el **XBee dongle** se nombró (B) y posteriormente seleccione los sensores cuya conexión existe en la tarjeta SAD, luego inserte el tiempo de muestreo (duración total del registro) y el intervalo requerido para tomar las muestras (periodo de la muestra) (C) (ver **figura 7**), una vez establecidos todos los parámetros pulse el botón **Aceptar** (D).

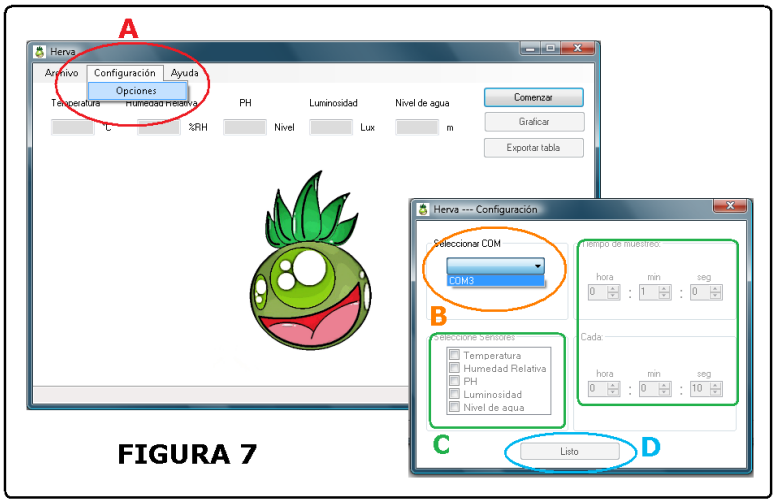


FIGURA 7

NOTA: En caso de haber seleccionado el sensor de nivel de agua y dar **Aceptar**, aparecerá una ventana (ver **figura 7.1**), donde tendrá que ingresar la distancia del radio del recipiente, luego haga clic en **Aceptar** y luego una vez más la ventana de configuración.

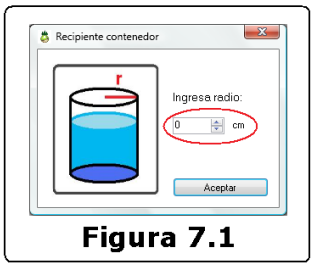
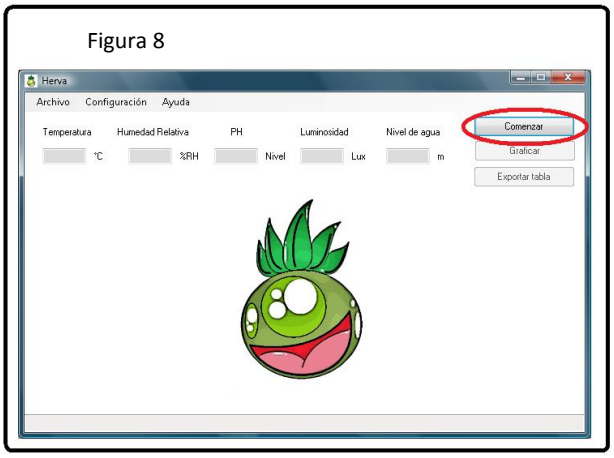


Figura 7.1

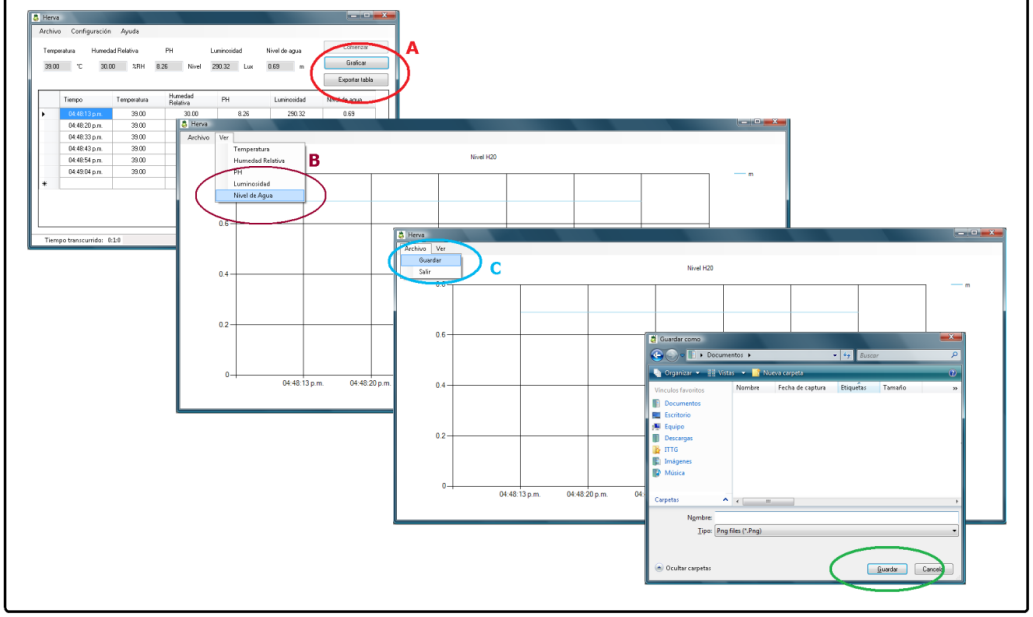
Haga clic en el botón comenzar para iniciar el proceso de muestreo (ver **figura 8**).

Figura 8



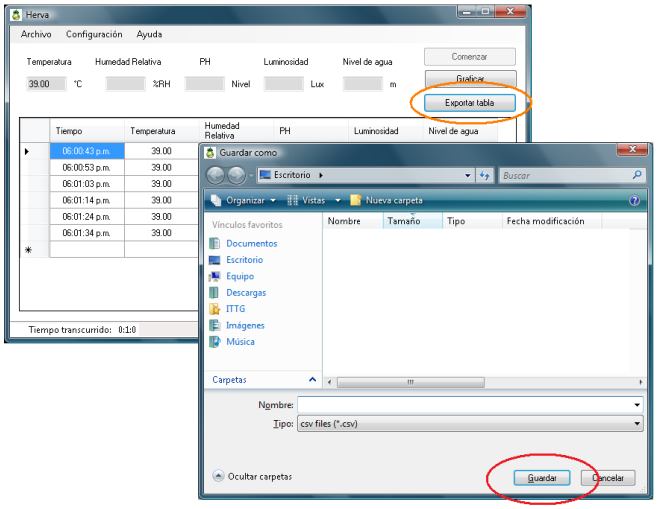
Una vez terminado el proceso de muestreo en el tiempo establecido, podrá graficar los datos de la tabla pulsando el botón **Graficar** (A), por dónde aparecerá una ventana cuya grafica se visualiza de forma manual dirigiéndose al menú **Ver->Variable a visualizar** (B) (**ver figura 9**), usted podrá almacenar la gráfica que se visualiza en ese instante dirigiéndose a **Archivo->guardar** (C).

Figura 9



Nota: Las gráficas son almacenadas en formato *.**PNG**

Figura 10

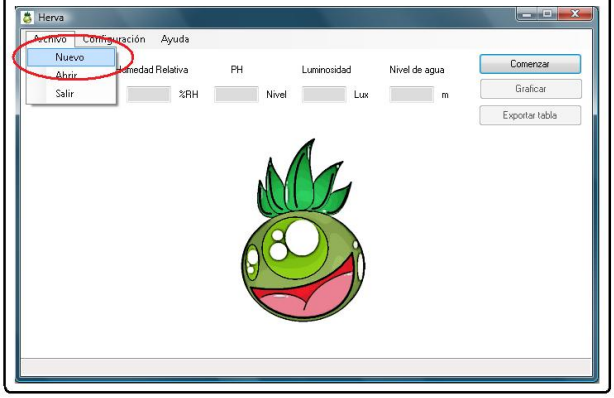


Es posible que necesite el exportar los datos de la tabla para insertar los mismos en su trabajo de investigación, para ello deberá pulsar el botón **Exportar tabla** (ver **figura 10**).

Nota: La tabla podrá ser almacenarla en un archivo con las siguientes extensiones: *.csv, *.doc, *.html, *.xls.

Usted podrá repetir un nuevo proceso de muestreo dirigiéndose al menú **Archivo->nuevo** (ver **figura 11**), y regresando al paso **6**.

Figura 11



Las gráficas almacenadas en la PC con anterioridad pueden ser visualizadas accediendo al menú **Archivo-> Abrir** o con cualquier visor de imágenes.

Para acceder a la ayuda de **HERVA V2.1** diríjase al menú **Ayuda->Soporte técnico**.

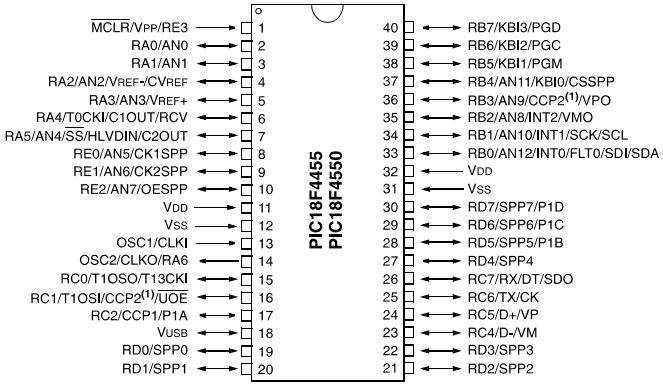
Nota: Por cualquier problema que pueda surgir durante la ejecución de la aplicación, ciérrelo e inicialícelo de nuevo, vuelva a reconfigurarlo y comience un nuevo proceso de muestreo.

ANEXO D

Hojas de datos

Microcontrolador PIC18F4550

CONFIGURACIÓN DE LOS PINES:



CARACTERÍSTICAS

Features	PIC18F2455	PIC18F2550	PIC18F4455	PIC18F4550
Operating Frequency	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz
Program Memory (Bytes)	24576	32768	24576	32768
Program Memory (Instructions)	12288	16384	12288	16384
Data Memory (Bytes)	2048	2048	2048	2048
Data EEPROM Memory (Bytes)	256	256	256	256
Interrupt Sources	19	19	20	20
I/O Ports	Ports A, B, C, (E)	Ports A, B, C, (E)	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	1	1
Enhanced Capture/ Compare/PWM Modules	0	0	1	1
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Universal Serial Bus (USB) Module	1	1	1	1
Streaming Parallel Port (SPP)	No	No	Yes	Yes
10-Bit Analog-to-Digital Module	10 Input Channels	10 Input Channels	13 Input Channels	13 Input Channels
Comparators	2	2	2	2
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT
Programmable Low-Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	28-pin PDIP 28-pin SOIC	28-pin PDIP 28-pin SOIC	40-pin PDIP 44-pin QFN 44-pin TQFP	40-pin PDIP 44-pin QFN 44-pin TQFP

PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
MCLR/VPP/RE3	1	18	18			Master Clear (input) or programming voltage (input).
MCLR				I	ST	Master Clear (Reset) input. This pin is an active-low Reset to the device.
VPP				P		Programming voltage input.
RE3				I	ST	Digital input.
OSC1/CLKI	13	32	30			Oscillator crystal or external clock input.
OSC1				I	Analog	Oscillator crystal input or external clock source input.
CLKI				I	Analog	External clock source input. Always associated with pin function OSC1. (See OSC2/CLKO pin.)
OSC2/CLKO/RA6	14	33	31			Oscillator crystal or clock output.
OSC2				O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO				O	—	In RC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6				I/O	TTL	General purpose I/O pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX Configuration bit is set.

3: These pins are No Connect unless the ICPRT Configuration bit is set. For NC/ICPORTS, the pin is No Connect unless ICPRT is set and the DEBUG Configuration bit is cleared.

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTA is a bidirectional I/O port.
RA0/AN0	2	19	19			
RA0				I/O	TTL	Digital I/O.
AN0				I	Analog	Analog input 0.
RA1/AN1	3	20	20			
RA1				I/O	TTL	Digital I/O.
AN1				I	Analog	Analog input 1.
RA2/AN2/VREF-/CVREF	4	21	21			
RA2				I/O	TTL	Digital I/O.
AN2				I	Analog	Analog input 2.
VREF-				I	Analog	A/D reference voltage (low) input.
CVREF				O	Analog	Analog comparator reference output.
RA3/AN3/VREF+	5	22	22			
RA3				I/O	TTL	Digital I/O.
AN3				I	Analog	Analog input 3.
VREF+				I	Analog	A/D reference voltage (high) input.
RA4/T0CKI/C1OUT/RCV	6	23	23			
RA4				I/O	ST	Digital I/O.
T0CKI				I	ST	Timer0 external clock input.
C1OUT				O	—	Comparator 1 output.
RCV				I	TTL	External USB transceiver RCV input.
RA5/AN4/SS/HLVDIN/C2OUT	7	24	24			
RA5				I/O	TTL	Digital I/O.
AN4				I	Analog	Analog input 4.
SS				I	TTL	SPI slave select input.
HLVDIN				I	Analog	High/Low-Voltage Detect input.
C2OUT				O	—	Comparator 2 output.
RA6	—	—	—	—	—	See the OSC2/CLKO/RA6 pin.

t

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/AN12/INT0/ FLT0/SDI/SDA	33	9	8			
RB0				I/O	TTL	Digital I/O.
AN12				I	Analog	Analog input 12.
INT0				I	ST	External interrupt 0.
FLT0				I	ST	Enhanced PWM Fault input (ECCP1 module).
SDI				I	ST	SPI data in.
SDA				I/O	ST	I^2C™ data I/O.
RB1/AN10/INT1/SCK/ SCL	34	10	9			
RB1				I/O	TTL	Digital I/O.
AN10				I	Analog	Analog input 10.
INT1				I	ST	External interrupt 1.
SCK				I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL				I/O	ST	Synchronous serial clock input/output for I^2C mode.
RB2/AN8/INT2/VMO	35	11	10			
RB2				I/O	TTL	Digital I/O.
AN8				I	Analog	Analog input 8.
INT2				I	ST	External interrupt 2.
VMO				O	—	External USB transceiver VMO output.
RB3/AN9/CCP2/VPO	36	12	11			
RB3				I/O	TTL	Digital I/O.
AN9				I	Analog	Analog input 9.
CCP2(1)				I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
VPO				O	—	External USB transceiver VPO output.
RB4/AN11/KBI0/CSSPP	37	14	14			
RB4				I/O	TTL	Digital I/O.
AN11				I	Analog	Analog input 11.
KBI0				I	TTL	Interrupt-on-change pin.
CSSPP				O	—	SPP chip select control output.
RB5/KBI1/PGM	38	15	15			
RB5				I/O	TTL	Digital I/O.
KBI1				I	TTL	Interrupt-on-change pin.
PGM				I/O	ST	Low-Voltage ICSP™ Programming enable pin.
RB6/KBI2/PGC	39	16	16			
RB6				I/O	TTL	Digital I/O.
KBI2				I	TTL	Interrupt-on-change pin.
PGC				I/O	ST	In-Circuit Debugger and ICSP programming clock pin.
RB7/KBI3/PGD	40	17	17			
RB7				I/O	TTL	Digital I/O.
KBI3				I	TTL	Interrupt-on-change pin.
PGD				I/O	ST	In-Circuit Debugger and ICSP programming data pin.

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTC is a bidirectional I/O port.
RC0/T1OSO/T13CKI	15	34	32			
RC0				I/O	ST	Digital I/O.
T1OSO				O	—	Timer1 oscillator output.
T13CKI				I	ST	Timer1/Timer3 external clock input.
RC1/T1OSI/CCP2/ $\overline{UOE}$	16	35	35			
RC1				I/O	ST	Digital I/O.
T1OSI				I	CMOS	Timer1 oscillator input.
CCP2(2)				I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
$\overline{UOE}$				O	—	External USB transceiver $\overline{OE}$ output.
RC2/CCP1/P1A	17	36	36			
RC2				I/O	ST	Digital I/O.
CCP1				I/O	ST	Capture 1 input/Compare 1 output/PWM 1 output.
P1A				O	TTL	Enhanced CCP1 PWM output, channel A.
RC4/D-/VM	23	42	42			
RC4				I	TTL	Digital input.
D-				I/O	—	USB differential minus line (input/output).
VM				I	TTL	External USB transceiver VM input.
RC5/D+/VP	24	43	43			
RC5				I	TTL	Digital input.
D+				I/O	—	USB differential plus line (input/output).
VP				I	TTL	External USB transceiver VP input.
RC6/TX/CK	25	44	44			
RC6				I/O	ST	Digital I/O.
TX				O	—	EUSART asynchronous transmit.
CK				I/O	ST	EUSART synchronous clock (see RX/DT).
RC7/RX/DT/SDO	26	1	1			
RC7				I/O	ST	Digital I/O.
RX				I	ST	EUSART asynchronous receive.
DT				I/O	ST	EUSART synchronous data (see TX/CK).
SDO				O	—	SPI data out.

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTD is a bidirectional I/O port or a Streaming Parallel Port (SPP). These pins have TTL input buffers when the SPP module is enabled.
RD0/SPP0	19	38	38			
RD0				I/O	ST	Digital I/O.
SPP0				I/O	TTL	Streaming Parallel Port data.
RD1/SPP1	20	39	39			
RD1				I/O	ST	Digital I/O.
SPP1				I/O	TTL	Streaming Parallel Port data.
RD2/SPP2	21	40	40			
RD2				I/O	ST	Digital I/O.
SPP2				I/O	TTL	Streaming Parallel Port data.
RD3/SPP3	22	41	41			
RD3				I/O	ST	Digital I/O.
SPP3				I/O	TTL	Streaming Parallel Port data.
RD4/SPP4	27	2	2			
RD4				I/O	ST	Digital I/O.
SPP4				I/O	TTL	Streaming Parallel Port data.
RD5/SPP5/P1B	28	3	3			
RD5				I/O	ST	Digital I/O.
SPP5				I/O	TTL	Streaming Parallel Port data.
P1B				O	—	Enhanced CCP1 PWM output, channel B.
RD6/SPP6/P1C	29	4	4			
RD6				I/O	ST	Digital I/O.
SPP6				I/O	TTL	Streaming Parallel Port data.
P1C				O	—	Enhanced CCP1 PWM output, channel C.
RD7/SPP7/P1D	30	5	5			
RD7				I/O	ST	Digital I/O.
SPP7				I/O	TTL	Streaming Parallel Port data.
P1D				O	—	Enhanced CCP1 PWM output, channel D.

Amplificador operacional de entrada JFET Dual de amplio ancho de banda
MARCA: ST Modelo: TL082

DESCRIPCIÓN GENERAL

Estos dispositivos son de bajo costo, alta velocidad, amplificadores operacionales de doble entrada JFET con un voltaje de offset en entrada internamente recortado (Tecnología BI-FET IITM). Estos requieren bajo corriente de alimentación, todavía mantiene un largo producto de ganancia de ancho de banda y una rápida velocidad de giro. Además, un voltaje de entrada JFET bien adaptado a dispositivos, proporcionando muy poca polarización de entrada y corrientes de offset. Los pines del TL082 son compatibles con el estándar LM1558 permitiendo a diseñadores el poder inmediatamente aumentar el rendimiento total de los actuales LM1558 y la mayoría de los diseñadores del LM358.

Estos integrados pueden ser usados en aplicaciones como tales como integradores de alta velocidad, convertidores Digital/Analógico rápidos, circuitos de muestro y retención y muchas otros circuitos que requieran bajo voltaje de offset en entrada, baja corriente de polarización, alta impedancia de entrada, alta velocidad de giro y amplio ancho de banda. Estos dispositivos además muestran poco ruido y voltaje de deriva en offset.

CARACTERÍSTICAS

- Voltaje de offset internamente recortado 15 mV
- Baja corriente de polarización de entrada 50 pA
- Bajo ruido de voltaje en entrada 16 nV/Hz
- Bajo ruido de corriente en entrada 0.01 pA/Hz
- Amplia Ganancia de ancho de banda 4 MHz
- Alta Velocidad de giro 13 V/µs
- Corriente de alimentación baja 3.6 mA
- Distorsión armónica total baja A_V=10, <0.02%
- Ruido de esquina 1/f baja 50 Hz
- Rápido tiempo de asentamiento hasta 0.01% 2 µs

CONFIGURACIÓN

CONEXION TIPICA

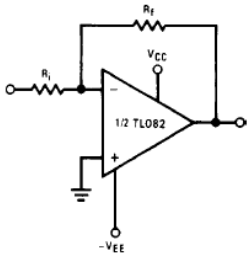
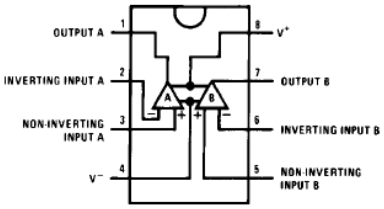


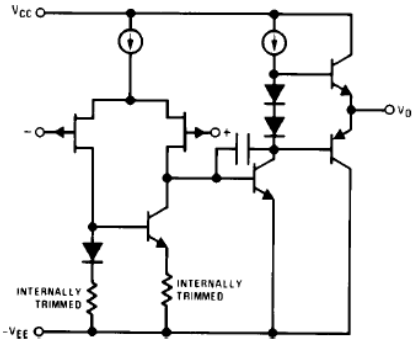
DIAGRAMA DE CONEXION

DIP/SO Package (Top View)



TL/H/8357–3

ESQUEMATICO SIMPLIFICADO



RANGOS ABSOLUTOS MÁXIMOS

- Voltaje de Alimentación ±18V
- Voltaje de entrada diferencial ±30 V
- Rango de Voltaje de entrada ±15 V
- Duración de corto circuito de salida Continua
- Rango de Temperatura almacenada -65°C a +150°C
- Temperatura de conducción (soldado, 10 segundos) 260°C

CARACTERÍSTICAS ELÉCTRICAS EN CD

Symbol	Parameter	Conditions	TL082C			Units
			Min	Typ	Max	
V_{OS}	Input Offset Voltage	$R_S = 10\ k\Omega$, $T_A = 25°C$ Over Temperature		5	15 20	mV mV
$\Delta V_{OS}/\Delta T$	Average TC of Input Offset Voltage	$R_S = 10\ k\Omega$		10		$\mu V/°C$
I_{OS}	Input Offset Current	$T_j = 25°C$, (Notes 4, 5) $T_j \leq 70°C$		25	200 4	pA nA
I_B	Input Bias Current	$T_j = 25°C$, (Notes 4, 5) $T_j \leq 70°C$		50	400 8	pA nA
R_{IN}	Input Resistance	$T_j = 25°C$		10^{12}		Ω
A_{VOL}	Large Signal Voltage Gain	$V_S = \pm 15V$, $T_A = 25°C$ $V_O = \pm 10V$, $R_L = 2\ k\Omega$ Over Temperature	25 15	100		V/mV V/mV
V_O	Output Voltage Swing	$V_S = \pm 15V$, $R_L = 10\ k\Omega$	±12	±13.5		V
V_{CM}	Input Common-Mode Voltage Range	$V_S = \pm 15V$	±11	+15 −12		V V
CMRR	Common-Mode Rejection Ratio	$R_S \leq 10\ k\Omega$	70	100		dB
PSRR	Supply Voltage Rejection Ratio	(Note 6)	70	100		dB
I_S	Supply Current			3.6	5.6	mA

CARACTERÍSTICAS ELÉCTRICAS EN AC

Symbol	Parameter	Conditions	TL082C			Units
			Min	Typ	Max	
	Amplifier to Amplifier Coupling	$T_A = 25°C$, f = 1Hz-20 kHz (Input Referred)		−120		dB
SR	Slew Rate	$V_S = \pm 15V$, $T_A = 25°C$	8	13		V/μs
GBW	Gain Bandwidth Product	$V_S = \pm 15V$, $T_A = 25°C$		4		MHz
e_n	Equivalent Input Noise Voltage	$T_A = 25°C$, $R_S = 100\Omega$, f = 1000 Hz		25		nV/√Hz
i_n	Equivalent Input Noise Current	$T_j = 25°C$, f = 1000 Hz		0.01		pA/√Hz

Amplificador de Instrumentación
Marca: Burr-Brown Modelo: INA126

Características

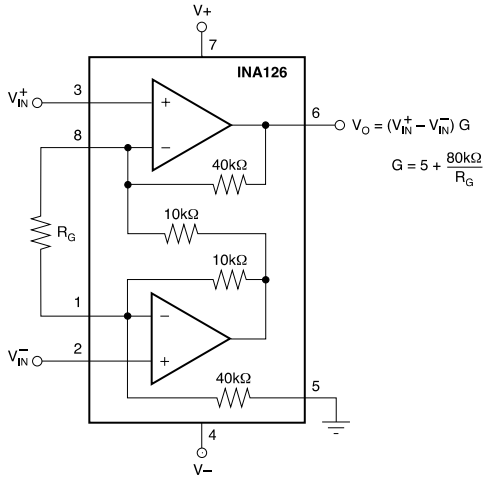
- Baja corriente inactiva :175µA/chan
- Amplio rango de alimentación: ±1.35V a ±18V
- Bajo voltaje de offset: 250 µV máximo.
- Bajo offset de deriva: 3µV/°C máximo.
- Bajo ruido: 35nV/Hz
- Baja entrada de corriente de polarización: 25 nA máximo.
- 8 pines DIP, SO-8, MOSOP-8 de montaje superficial, Dual: 16 pines DIP, so-16, SSOP-16.

Descripción

El INA126 y el INA2126 son amplificadores de instrumentación de precisión para la medición precisa, bajo ruido en la adquisición de señales diferenciales. Su diseño de dos OPAMS provee excelente desempeño con baja corriente inactiva. Esto, combinado con un amplio rango de voltaje de operación, hace de él ideal para instrumentación portable y sistemas de adquisición de datos.

Su ganancia puede ser puesta desde 5V/V hasta 10000V/V con una simple resistencia externa. El circuito de entrada recortado con laser provee bajo voltaje de offset, bajo voltaje de deriva y un excelente rechazo en modo común.

Esta especificado para trabajar desde -40°C hasta +85°C, temperaturas de rango industrial.

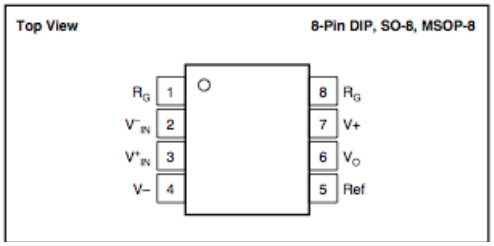


Rangos Absolutos Máximos

Voltaje de fuente de alimentación, V+ a V-	36V
Voltaje de entrada	(V-)-0.7 a (V+)+0.7V
Corriente de entrada	10mA
Corto circuito de salida	continuo
Temperatura de operación	-55°C a +125°C
Temperatura de almacenamiento	-55°C a +125°C
Soldadura, 10 s	300°C

Nota: superar estos rangos puede causar daños permanentes. El voltaje de entrada es limitado por diodos internos conectados a la fuente de alimentación.

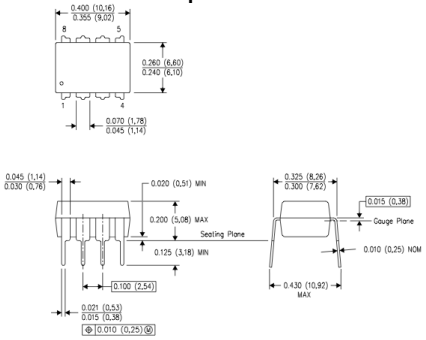
Configuración de los Pines



Sensibilidad a descargas eléctricas.

Este circuito puede ser dañado por descargas eléctricas, Texas Instruments recomienda que todos los circuitos integrados sean manejados con las precauciones apropiadas. Hacer caso omiso al apropiado manejo y procedimientos de instalación puede causar daño.

Daño por descargas eléctricas puede ir desde una simple degradación en el desempeño hasta un daño completo en el dispositivo. Circuitos integrados de Precisión pueden ser más susceptibles a sufrir daños debido a que muy pequeños cambios en los parámetros pueden causar que el dispositivo sea puesto a condiciones que no cumplen con sus especificaciones.



Características Eléctricas.

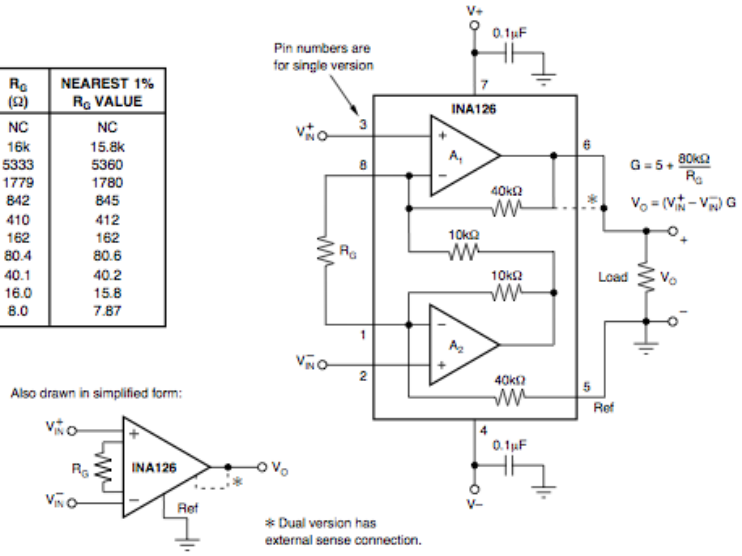
A T_A = +25°C, V_S = ±15V, R_L = 25kΩ

PARAMETER	CONDITIONS	INA126P, U, E INA2126P, U, E MIN	TYP	MAX	INA126PA, UA, EA INA2126PA, UA, EA MIN	TYP	MAX	UNITS
INPUT								
Offset Voltage, RTI			±100	±250		±150	±500	µV
vs Temperature			±0.5	±3		*	±5	µV/°C
vs Power Supply (PSRR)	V_S = ±1.35V to ±18V		5	15		*	50	µV/V
Input Impedance			10^9 || 4			*		Ω || pF
Safe Input Voltage	R_S = 0	(V–)–0.5		(V+)+0.5	*		*	V
	R_S = 1kΩ	(V–)–10		(V+)+10	*		*	V
Common-Mode Voltage Range	V_O = 0V	±11.25	±11.5		*	*		V
Channel Separation (dual)	G = 5, dc		130					dB
Common-Mode Rejection	R_S = 0, V_{CM} = ±11.25V	83	94		74	90		dB
INA2126U (dual SO-16)		80	94					dB
INPUT BIAS CURRENT			–10	–25		*	–50	nA
vs Temperature			±30			*		pA/°C
Offset Current			±0.5	±2		*	±5	nA
vs Temperature			±10			*		pA/°C
GAIN			G = 5 to 10k			*		V/V
Gain Equation			G = 5 + 80kΩ/R_G			*		V/V
Gain Error	V_O = ±14V, G = 5		±0.02	±0.1		*	±0.18	%
vs Temperature	G = 5		±2	±10		*	*	ppm/°C
Gain Error	V_O = ±12V, G = 100		±0.2	±0.5		*	±1	%
vs Temperature	G = 100		±25	±100		*	*	ppm/°C
Nonlinearity	G = 100, V_O = ±14V		±0.002	±0.012		*	*	%
NOISE								
Voltage Noise, f = 1kHz			35			*		nV/√Hz
f = 100Hz			35			*		nV/√Hz
f = 10Hz			45			*		nV/√Hz
f_B = 0.1Hz to 10Hz			0.7			*		μV_{PP}
Current Noise, f = 1kHz			60			*		fA/√Hz
f_B = 0.1Hz to 10Hz			2			*		pA_{PP}
OUTPUT								
Voltage, Positive	R_L = 25kΩ	(V+)–0.9	(V+)–0.75		*	*		V
Negative	R_L = 25kΩ	(V–)+0.95	(V–)+0.8		*	*		V
Short-Circuit Current	Short-Circuit to Ground		+10/–5			*		mA
Capacitive Load Drive			1000			*		pF
FREQUENCY RESPONSE								
Bandwidth, –3dB	G = 5		200			*		kHz
	G = 100		9			*		kHz
	G = 500		1.8			*		kHz
Slew Rate	V_O = ±10V, G = 5		0.4			*		V/µs
Settling Time, 0.01%	10V Step, G = 5		30			*		µs
	10V Step, G = 100		160			*		µs
	10V Step, G = 500		1500			*		µs
Overload Recovery	50% Input Overload		4			*		µs
POWER SUPPLY								
Voltage Range		±1.35	±15	±18	*	*	*	V
Current (per channel)	I_O = 0		±175	±200		*	*	µA
TEMPERATURE RANGE								
Specification Range		–40		+85	*		*	°C
Operation Range		–55		+125	*		*	°C
Storage Range		–55		+125	*		*	°C
Thermal Resistance, θ_{JA}								
8-Pin DIP			100			*		°C/W
SO-8 Surface-Mount			150			*		°C/W
MSOP-8 Surface-Mount			200			*		°C/W
16-Pin DIP (dual)			80			*		°C/W
SO-16 (dual)			100			*		°C/W
SSOP-16 (dual)			100			*		°C/W

Conexión Típica

DESIRED GAIN (V/V)	R_G (Ω)	NEAREST 1% R_G VALUE
5	NC	NC
10	16k	15.8k
20	5333	5360
50	1779	1780
100	842	845
200	410	412
500	162	162
1000	80.4	80.6
2000	40.1	40.2
5000	16.0	15.8
10000	8.0	7.87

NC: No Connection.



En la figura se muestra la conexión básica requerida para la operación del INA126. Para aplicaciones en donde hay ruido o alta impedancia, las entradas de alimentación requieren capacitores de desacoplo cerca del dispositivo como se muestra.

La salida es referida a la salida de referencia (Ref) terminal que normalmente es aterrizada. Esto de hacer una conexión baja impedancia para asegurar un buen rechazo de modo común. Una resistencia de 8Ω en serie con el pin de Referencia esto hará que el dispositivo degrade aproximadamente 80 dB CMR.

Ajustar Ganancia

La ganancia es ajustada conectando una resistencia externa, R_G, como se muestra:

$$G = 5 + \frac{80K\Omega}{R_G}$$

El término 80 KΩ en la ecuación viene de la resistencia interna de filamento metálico que son cortados con laser para valores de precisión absoluta. La precisión y el coeficiente de temperatura de estas resistencias estan incluidas en la precisión de la ganancia y especificaciones de deriva.

La estabilidad y la temperatura de deriva de la resistencia externa de ajuste de ganancia, R_G, también afecta la ganancia. La contribución de R_G para la precisión en la ganancia y la deriva pueden desprenderse directamente de la ganancia.

Temperatura
Marca: ST Sensor de Temperatura Modelo: LM35

Descripción
La serie LM35 es un circuito integrado de precisión sensor de temperatura, su voltaje de salida es linealmente proporcional a °C de temperatura. El LM35 tiene una ventaja por encima de otros sensores de temperatura lineal calibrados en °K, ya que no es necesario hacer la sustracción de voltaje de salida para la obtención de la temperatura en escala de °C. El LM35 no requiere de una calibración externa o recorte para proporcionar una precisión típica de ±1/4 °C a temperatura de cuarto y +- ¾ de °C sobre su rango total de temperatura comprendido desde -55 hasta +150 °C. El bajo costo está asegurado. La Impedancia de salida del LM35, es lineal, y la calibración inherentemente precisa hace que la interconexión de circuitos de lectura o control sea especialmente sencilla. Este puede ser usado con fuentes de poder simples, o fuentes simétricas.

Características

- Calibrado directamente en grados Celsius
- Factor de escala lineal de +10.0 mV/°C
- 0.5 °C de precisión asegurada (a +25°C)
- Probado para pleno -55 hasta +150 °C rango.
- Adecuado para aplicaciones remotas.
- Opera desde 4 hasta 30 Volts
- Menos de 60 Micro Amperes de corriente de drenaje
- Poco auto-calentamiento, 0.08 °C sin aire
- No linealidad solo +- ¼°C típico.
- Baja impedancia de salida, 0.1 Ω para 1 mV carga.

Rangos máximos absolutos

- Voltaje de alimentación +35V a -0.2V
- Salida de voltaje +6V a -1.0V
- Corriente de salida 10 mA

Temperatura almacenada

TO-46 paquete	-60 °C a +180°C
TO-92 paquete	-60 °C a +150 °C
SO-8 paquete	-65 ºC a +150 °C
TO-202 paquete	-65 °C a +150 °C

Temperatura de conducción

- TO-46 paquete (soldadura, 10 segundos) 300°C
- TO-92 paquete (soldadura, 10 segundos) 260°C
- TO-202 paquete (soldadura, 10 segundos) +230°C

Especificación de Rango de temperatura de operación: T_{min} a T_{max}

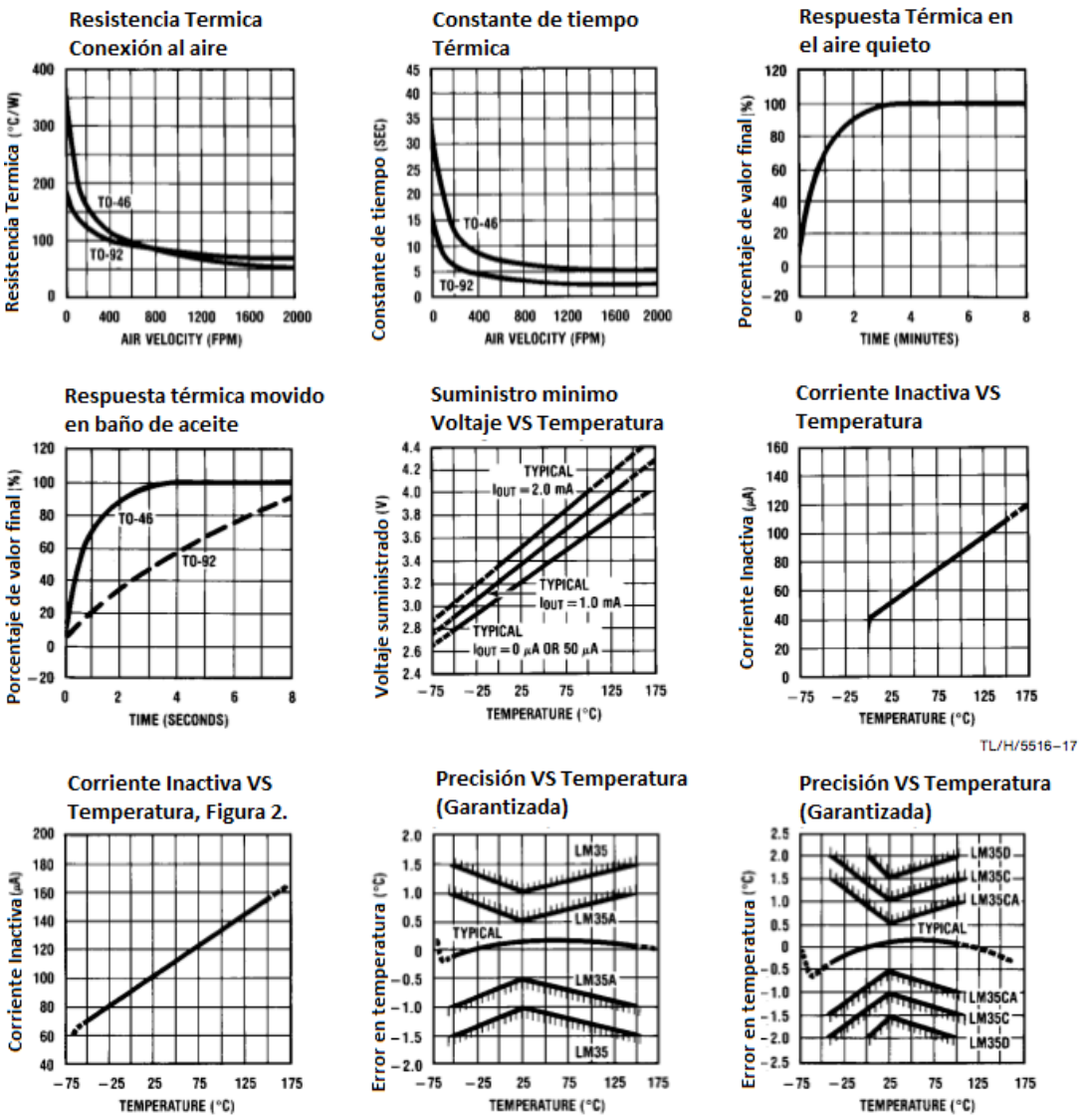
- LM35, LM35A -55 °C a +150 °C
- LM35C, LM35CA -40 °C a +110 °C
- LM35D 0°C a 100 °C

Características Eléctricas

Parametro	Condiciones	LM35A			LM35CA			Unidades (Max)
		Tipico	Limite Probado (Nota 4)	Limite de Diseño (Nota 5)	Tipico	Limite Probado (Nota 4)	Limite de Diseño (Nota 5)	
Precisión (Nota 7)	$T_A = +25°C$ $T_A = -10°C$ $T_A = T_{MAX}$ $T_A = T_{MIN}$	±0.2 ±0.3 ±0.4 ±0.4	±0.5 ±1.0 ±1.0		±0.2 ±0.3 ±0.4 ±0.4	±0.5 ±1.0 	 ±1.0 ±1.5	°C °C °C °C
No linearidad (Nota 8)	$T_{MIN} \le T_A \le T_{MAX}$	**±0.18**		**±0.35**	**±0.15**		**±0.3**	°C
Ganancia del sensor (pendiente media)	$T_{MIN} \le T_A \le T_{MAX}$	**+10.0**	**+9.9, +10.1**		**+10.0**		**+9.9, +10.1**	mV/°C
Carga de regulación (Nota 3) $0 \le I_L \le 1$ mA	$T_A = +25°C$ $T_{MIN} \le T_A \le T_{MAX}$	±0.4 **±0.5**	±1.0 	 **±3.0**	±0.4 **±0.5**	±1.0 	 **±3.0**	mV/mA mV/mA
Linea de Regulación (Nota 3)	$T_A = +25°C$ $4V \le V_S \le 30V$	±0.01 **±0.02**	±0.05 	 **±0.1**	±0.01 **±0.02**	±0.05 	 **±0.1**	mV/V mV/V
Corriente Inactiva (Nota 9)	$V_S = +5V, +25°C$ $V_S = +5V$ $V_S = +30V, +25°C$ $V_S = +30V$	56 **105** 56.2 **105.5**	67 68 	 **131** **133**	56 **91** 56.2 **91.5**	67 68 	 **114** **116**	µA µA µA µA
Cambio de Corriente Inactiva (Nota 3)	$4V \le V_S \le 30V, +25°C$ $4V \le V_S \le 30V$	0.2 **0.5**	1.0 	 **2.0**	0.2 **0.5**	1.0 	 **2.0**	µA µA
Coeficiente de Temperatura de Corriente Inactiva		**+0.39**		**+0.5**	**+0.39**		**+0.5**	µA/°C
Temperatura minima para medir Precisión	In circuit of *Figure 1*, $I_L = 0$	+1.5		+2.0	+1.5		+2.0	°C
Estabilidad a largo Plazo	$T_J = T_{MAX}$, for 1000 Horas	±0.08			±0.08			°C

Parametro	Condiciones	LM35			LM35C, LM35D			Unidades (Max)
		Tipico	Limite Probado (Nota 4)	Limite de Diseño (Nota 5)	Tipico	Limite Probado (Nota 4)	Limite de Diseño (Nota 5)	
Precisión LM35, LM35C (Nota 7)	$T_A = +25°C$ $T_A = -10°C$ $T_A = T_{MAX}$ $T_A = T_{MIN}$	±0.4 ±0.5 ±0.8 ±0.8	±1.0 ±1.5 	 ±1.5	±0.4 ±0.5 ±0.8 ±0.8	±1.0 	 ±1.5 ±1.5 ±2.0	°C °C °C °C
Precisión LM35D (Nota 7)	$T_A = +25°C$ $T_A = T_{MAX}$ $T_A = T_{MIN}$				±0.6 ±0.9 ±0.9	±1.5 	 ±2.0 ±2.0	°C °C °C
No linearidad (Nota 8)	$T_{MIN} \le T_A \le T_{MAX}$	**±0.3**		**±0.5**	**±0.2**		**±0.5**	°C
Ganancia del sensor (pendiente media)	$T_{MIN} \le T_A \le T_{MAX}$	**+10.0**	**+9.8, +10.2**		**+10.0**		**+9.8, +10.2**	mV/°C
Carga de regulación (Nota 3) $0 \le I_L \le 1$ mA	$T_A = +25°C$ $T_{MIN} \le T_A \le T_{MAX}$	±0.4 **±0.5**	±2.0 	 **±5.0**	±0.4 **±0.5**	±2.0 	 **±5.0**	mV/mA mV/mA
Linea de Regulación (Nota 3)	$T_A = +25°C$ $4V \le V_S \le 30V$	±0.01 **±0.02**	±0.1 	 **±0.2**	±0.01 **±0.02**	±0.1 	 **±0.2**	mV/V mV/V
Corriente Inactiva (Nota 9)	$V_S = +5V, +25°C$ $V_S = +5V$ $V_S = +30V, +25°C$ $V_S = +30V$	56 **105** 56.2 **105.5**	80 82 	 **158** **161**	56 **91** 56.2 **91.5**	80 82 	 **138** **141**	µA µA µA µA
Cambio de Corriente Inactiva (Nota 3)	$4V \le V_S \le 30V, +25°C$ $4V \le V_S \le 30V$	0.2 **0.5**	2.0 	 **3.0**	0.2 **0.5**	2.0 	 **3.0**	µA µA
Coeficiente de Temperatura de Corriente Inactiva		**+0.39**		**+0.7**	**+0.39**		**+0.7**	µA/°C
Temperatura minima para medir Precisión	In circuit of *Figure 1*, $I_L = 0$	+1.5		+2.0	+1.5		+2.0	°C
Estabilidad a largo Plazo	$T_J = T_{MAX}$, for 1000 Horas	±0.08			±0.08			°C

Características Típicas de desempeño



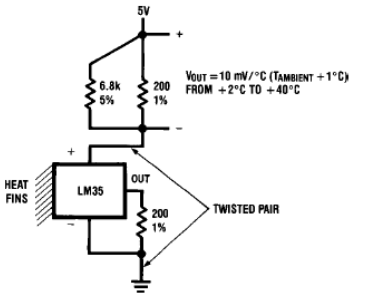
Aplicaciones

El LM35 puede ser aplicado de mismo modo que otros circuitos integrados sensores de temperatura. Este puede ser pegado a una superficie y su temperatura variara aproximadamente 0.01 °C con respecto a la temperatura de la superficie. Esto supone que la temperatura del aire ambiente, es casi la misma que la temperatura de la superficie, si la temperatura del aire fuera mucho mayor o menor que la temperatura de la superficie, la temperatura del LM35 sería un intermedio entre la temperatura de la superficie y la temperatura del aire.

Cargas capacitivas: como la mayoría de los circuitos micro-energizado, el LM35 tiene la capacidad de controlar grandes cargas capacitivas. El LM35 es capaz de controlar 50 pF sin precauciones especiales. Si las grandes cargas son anticipadas, es fácil el aislar o desacoplar la carga con una resistencia.

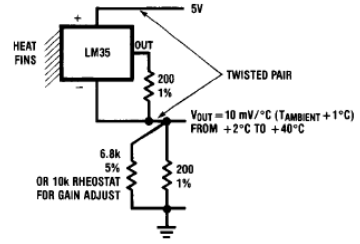
Cuando al LM35 se le aplica una resistencia de carga de 200 Ω, es relativamente inmune a la capacitancia del alambrado, debido a que la capacitancia forma una desviación de la tierra a la entrada, no en la salida. Sin embargo, como con cualquier circuito lineal conectado ha cableado en un medio ambiente hostil, su

rendimiento puede ser afectado negativamente por intensas fuentes electromagnéticas tal como relés, radio transmisores, motores, etc.



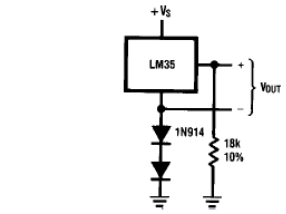
TL/H/5516–5

FIGURA 5. Sensor de temperatura remoto de dos lineas (Sensor Aterrizado)



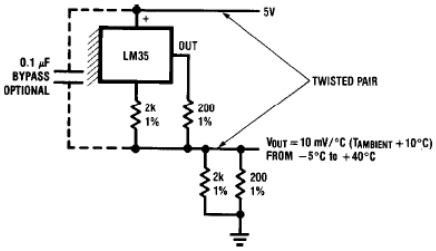
TL/H/5516–6

FIGURA 6.



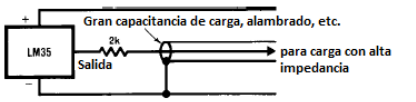
TL/H/5516–7

FIGURA 7. Sensor de temperatura, fuente unica −55° a +150°C



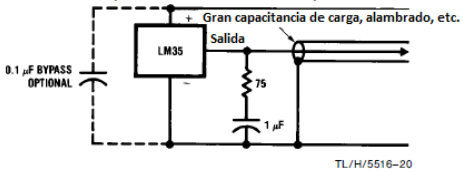
TL/H/5516–8

FIGURA 8. Sensor de temperatura remoto de dos lineas (salida con referencia a tierra)



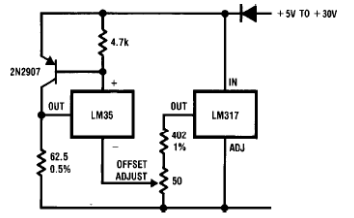
TL/H/5516–19

FIGURE 3. LM35 con desacoplamiento de carga capacitiva



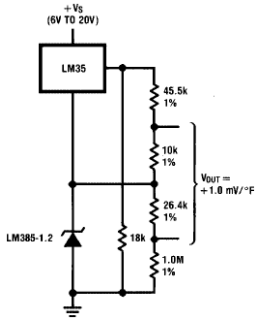
TL/H/5516–20

FIGURE 4. LM35 con Apagador R-C



TL/H/5516–9

FIGURA 9. Fuente de corriente 4-20 mA (0 a +100ºC)



TL/H/5516–10

FIGURA 10. Termometro farenheit

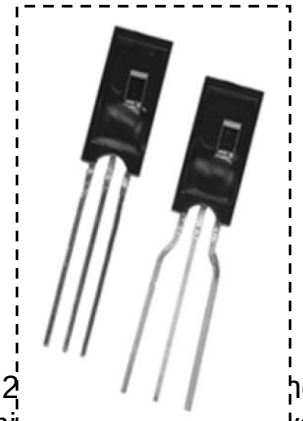
Humedad relativa
Marca: Noneywell Sensor de %RH Modelo: Serie HIH-4030

Descripción:
Los sensores de la serie HIH-4030 son diseñados específicamente para un alto volumen de usuarios de OEM (Original Equipment Manufacturer, Fabricante de Equipamiento Original).
La entrada directa a un controlador u otro dispositivo es posible gracias a su voltaje de salida lineal, con un consumo de corriente típico de sólo 200 µA, la serie

HIH-4000 es a menudo ideal para el bajo consumo o sistemas operados por baterías.
Su intercambiabilidad reduce o elimina costos de calibración en la producción OEM, ya que los datos de calibración estan disponibles en su hoja de datos.
La serie HIH-4030 ofrece instrumentación de calidad en su rendimiento de detección de RH (Relative Humidity, Humedad Relativa) a precio competitivo, encapsulado SIP (Single In-line Package). El sensor de RH consta de un polímero capacitivo termoestable como elemento de detección con un chip de acondicionamiento de señales integrado. La construcción de múltiples capas del elemento del sensor proporciona una excelente resistencia contra humectante, polvo, suciedad, aceites y sustancias químicas ambientales comunes.

Características

- Encapsulado de plástico moldeada termoestable
- Voltaje de salida lineal VS % Humedad Relativa (%RH)
- Diseño de baja potencia
- Alta precisión
- Tiempo de respuesta rápido
- Rendimiento estable
- Químicamente resistente



Aplicaciones

- Equipos de refrigeración
- Equipos de climatización
- Equipo médico
- Secado
- Metrología
- Sistemas alimentados por baterías

Tabla 1.- especificaciones de rendimiento (A los 5 Vdc a 25 °C [77 °F] y 30 %RH [humedad relativa] a menos que se indique lo contrario. Las especificaciones del rendimiento de %RH incluyen pruebas de los errores del sistema de medición (±0.5 % típico))

Parámetro	Mínimo	Típico	Máximo	Unidades
Intercambiabilidad (Mejor ajuste de línea)	---	---	---	---
0% a 60%	-5	---	5	%RH
60% a 100%	-8	---	8	%RH
Intercambiabilidad (Curva de 2° orden)	---	±3.5	---	%RH
Precisión1 (Mejor ajuste de línea)	---	±3.5	---	%RH
Precisión (Curva de 2° orden)	---	±2.5	---	%RH
Histéresis	---	3	---	%RH
Repetibilidad	---	±0.5	---	%RH
Tiempo de establecimiento	---	---	70	ms
Tiempo de respuesta (1/e en aire de movimiento lento)	---	15	---	s
Estabilidad2 (@ 50 %RH)	---	±1.2 (por año)	---	%RH
Estabilidad3 (@ 50 %RH)	---	±0.5 (por año)	---	%RH
Voltaje de la fuente	4	---	5.8	Vdc
Corriente de la fuente	---	---	500	µA
Voltaje de salida (Ajuste de 1° orden)	$V_{SALIDA} = V_{FUENTE} * (0.0062 * (\text{sensor RH}) + 0.16)$			
Voltaje de salida (Ajuste de curva de 2° orden)	$V_{SALIDA} = 0.00003*(\text{sensor RH})2 + 0.0281*(\text{sensor RH}) + 0.820$, típico @ 25 °C			
Compensación de temperatura	VSALIDA = (0.0305 + 0.000044*T – 0.0000011*T2)*(sensor RH) + (0.9237 –			

	0.0041*T + 0.000040*T2), Temperatura en °C			
Temperatura de operación	-40[-40]	Ver figura 1	85[185]	°C[°F]
Humedad de operación	0	Ver figura 2	100	%RH
Temperatura de almacenamiento	-40[-40]	---	125[257]	°C[°F]
Humedad de almacenamiento	Ver figura 2			%RH
Notas: 1 Exclusivo para HIH – 4000 – 003 2 Especificaciones incluyen pruebas fuera de la zona de operación recomendada 3 Especificaciones incluyen pruebas para sólo la zona de operación recomendada				

Consideraciones:

- No exponga al sensor a los ambientes de condensación, si lo hace hará que la salida del sensor indique 0 %RH
- El sensor es sensible a la luz, para un mayor rendimiento, proteja al sensor de luz brillante
- El sensor es sensible a la estática, proteja la conexión del sensor a un máximo de 15 kV
- La salida del sensor es radiométrica al la fuente de alimentación

Datos de calibración de fábrica

Los sensores HIH-4000 pueden ser ordenados con una calibración y datos impresos (Tabla 2)

Tabla 2.- Ejemplo de datos impresos

Modelo	HIH-4000-001
Canal	92
Oblea	030996M
MRP	337313
Valores calculados a 5 V V_{SALIDA} @ 0 %RH V_{SALIDA} @ 75.3 %RH	 0.958 V 3.268 V
Salida lineal para 2 %RH, precisión @25 °C Zero offset Pendiente de la recta RH	 0.958 V 30.680 mV/%RH (V_{SALIDA} – Zero offset)/Pendiente de la recta (V_{SALIDA}-0.958)/0.0307
Respuesta radiométrica para 0 % a 100 %RH V_{SALIDA}	 V_{FUENTE} (0.1915 a 0.8130)

FIGURA 1.- CONDICIONES RECOMENDADAS DE OPERACIÓN

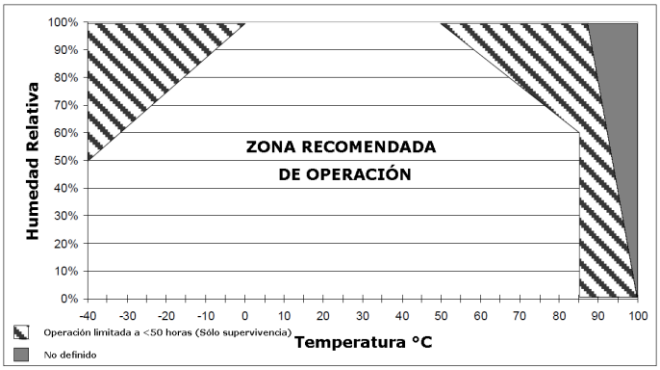


FIGURA 2.-AMBIENTE DE ALMACENAMIENTO

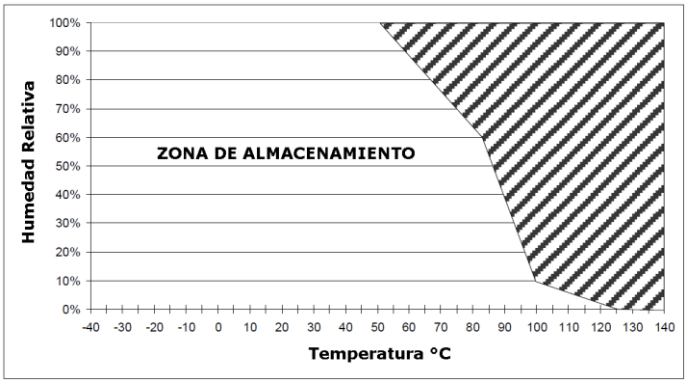


Figura 3.- DIMENSIONES EN mm/[in]

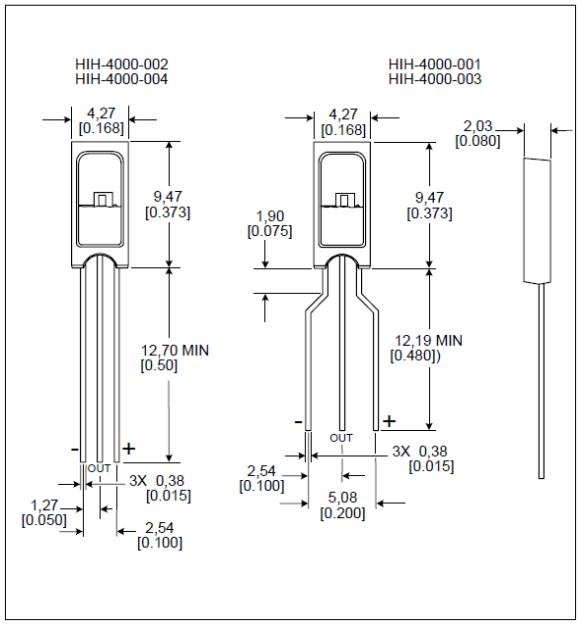


FIGURA 4.- MEJOR AJUSTE DE LÍNEA

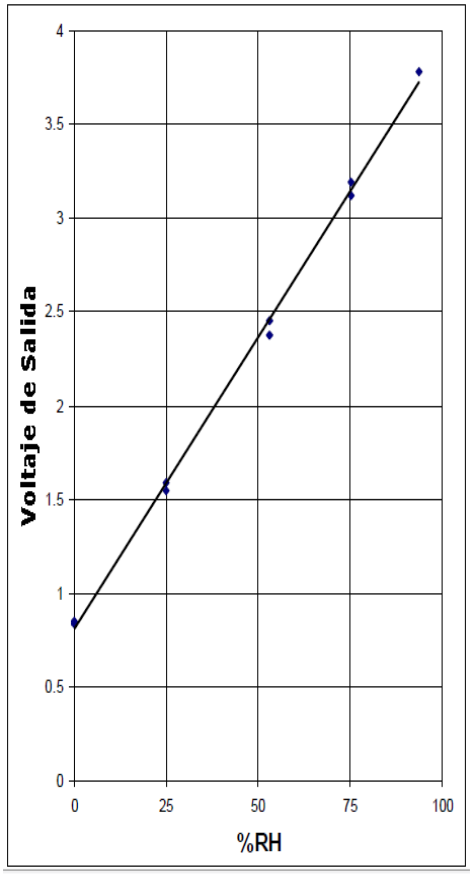
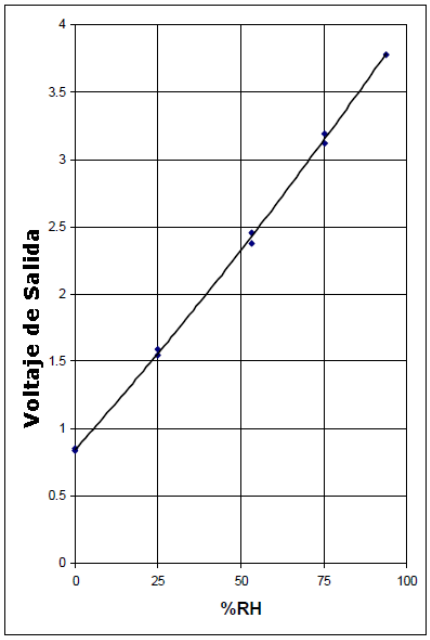


FIGURA 5.- Curva típica de 2° orden



PH
Marca: Global Water Sensor de PH Modelo: WQ201

Descripción:
El Sensor de PH WQ201 de GLOBAL WATER, es un dispositivo de medición de PH de agua resistente y fiable.
El transmisor de PH se monta en 25’ de cable de grado marino, con longitudes de hasta 500’ disponibles bajo petición.
La salida del sensor es de 4 – 20 mA con una configuración de tres hilos, el cable rojo es para la tensión positiva, el cable blanco para la tensión de salida y el negro es para tierra. La electrónica de la WQ201 es completamente encapsulado en epoxy de grado marino dentro de una carcasa de acero inoxidable. La unidad también utiliza un escudo extraíble y un sensor de PH reemplazable para su fácil mantenimiento.
Al igual que todos las salidas de 4 – 20 mA de los sensores de Global Water, se puede añadir capacidades de grabación y control al WQ201 con el registrador de datos (Datalogger) GL500 y el controlador PC300. El GL500 se conecta a la salida de 4 – 20 mA del sensor de PH para la grabación de datos y el controlador PC300 se conecta a la salida del sensor para el control de bombas o alarmas.
Características:

- Mediciones de PH sumergibles
- Sistema electrónico completamente encapsulado
- Salida: 4 – 20 mA
- Cable de grado marino con liberación de tensión
- Carcasa de acero inoxidable
- Elemento de PH reemplazable

Aplicaciones:

- Agricultura
- Monitoreo de lagos
- Análisis de referencia
- Supervisión de mitigación
- Otras aplicaciones ambientales

Especificaciones:

Salida	4 – 20 mA
Rango	0 a 14 pH
Precisión	2% de escala total
Presión máxima	40 psi
Voltaje de operación	10 a 30 Vdc
Consumo de corriente	5.5 mA más la salida del sensor
Temperatura de operación	-5 a +55 °C
Tiempo de calentamiento	3 segundos mínimo
Tamaño de la sonda	En aguas abiertas: 1¼" día. x 10 long (3.2 cm día x 25 cm long)
Peso	1 lb (454 g)

Luminosidad
Marca: Vishay Semiconductors Sensor de luz Modelo: TEMT6000

Descripción
TEMT6000 es un fototransistor NPN epitaxial de silicón, puesto en un molde transparente miniatura para montura superficial en tarjetas de circuitos impresos. El sensor es sensible al espectro visible.

Características
- Adaptado a la respuesta del ojo humano
- Amplio ángulo de media sensibilidad
- Empaquetado estilo SMD (montaje superficial) puesto en tecnología PCB
- Adecuado para IR soldadura por reflujo
- Componente libre de plomo

Aplicaciones
- Sensor de luz ambiental para pantalla retro-iluminada con regulación en:
- Teléfonos móviles
- Computadoras Laptop
- PDAs
- Cámaras
- Cuadros de mando

Rangos Máximos Absolutos

Parámetro	Condición de prueba	Símbolo	Valor	Unidad
Voltaje colector-emisor		VCEO	6	V
Voltaje emisor-colector		VECO	1.5	V
Corriente de colector		IC	20	mA
Potencia total de disipación	Tamb ≤ 55 °C	Ptot	100	mW
Temperatura de unión		Tj	100	°C
Rango de temperatura de operación		Tamb	-40 a +85	°C
Rango de temperatura almacenada		Tstg	-40 a +85	°C
Temperatura de soldadura	t ≤ 3 s	Tsd	260	°C
Resistencia térmica Unión/Ambiente		RthJA	450	K/W

Tamb = 25 °C a menos que se especifique lo contrario.

Características Básicas

Parámetro	Condición de prueba	Símbolo	mínimo	Típico	Máximo	Unidad
Voltaje colector emisor de ruptura	IC=0.1 mA	VCEO	6			V
Corriente oscura de colector	VCE=5V, E=0	ICEO		3	50	nA
Capacitancia de colector emisor	VCE=0V, f=1 MHz, E=0	CCEO		16		pF
Corriente de luz de colector	EV=20 lx, Luz estándar A	Ica	3.5	10	16	µA
	EV=100 lx, Luz estándar A	Ica		50		µA
Angulo de media sensibilidad		Φ		±60		Grados
Amplitud de onda de sensibilidad pico		λp		570		nm
Rango espectral de ancho de banda		Λ0.1		360 a 970		nm
Voltaje de saturación de colector emisor	EV=20 lx, 0.45 µA	VCEsat		0.1		V

Tamb = 25 °C a menos que se especifique lo contrario.

Características Típicas (T_{amb} = 25 °C, a menos que se especifique lo contrario)

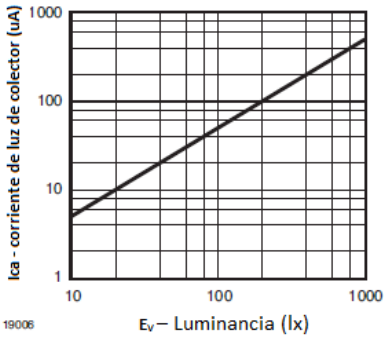


Figura 1. Corriente de luz de colector vs. Liminancia

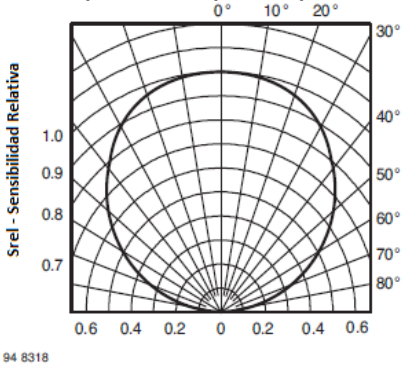


Figura 3. Sensibilidad Relativa radiante vs. Desplazamiento angular

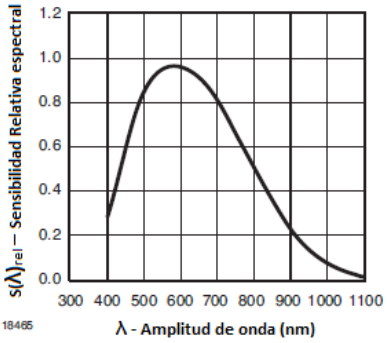
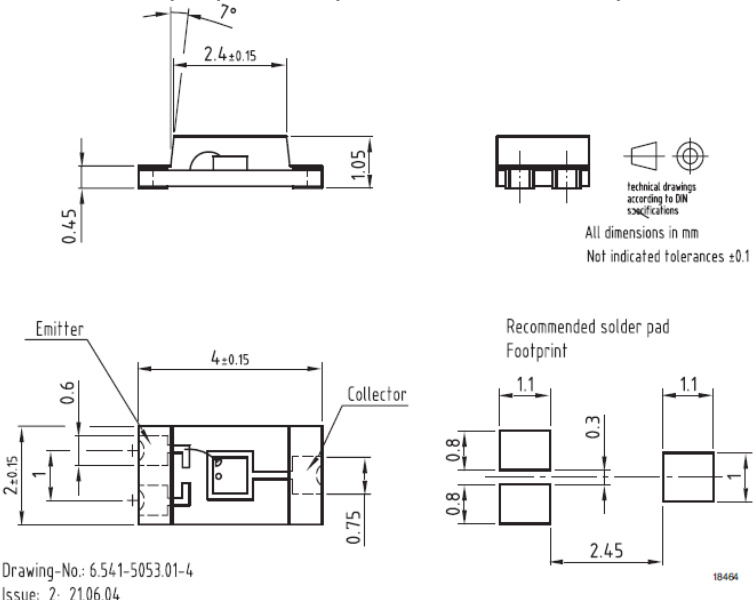


Figura 2. Capacidad de respuesta Relativa Espectral vs. Amplitud de onda

Empaquetado (dimensiones en nm)



Presión
Marca: Motorola Sensor de Presión diferencial MPX2010D

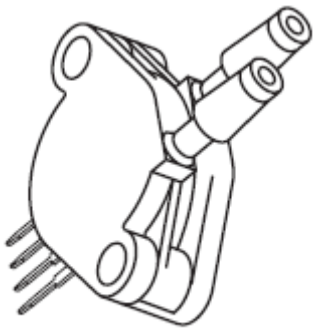
Descripción.
Los MPX2010/MPX2010G, es la serie de sensores de presión de silicio piezoeléctrico que proveen una buena precisión y una salida de voltaje lineal – directamente proporcional a la presión aplicada. Estos sensores albergan un solo chip de silicio monolítico – una capa fina de resistencias integradas en cada chip. El sensor es cortado con láser para su precisión, calibración y compensación de temperatura.

Características:

- Temperatura compensada por encima de 0°C a +85°C.
- Proporcional a la tensión de alimentación.
- Opción diferencial y manométrica.

Aplicaciones:

- Diagnóstico respiratorio.
- Control de movimiento de aire.
- Controladores.
- Presión de conmutación.



MPX2010DP

Número de pin			
1	Gnd	3	V_S
2	$+V_{out}$	4	$-V_{out}$

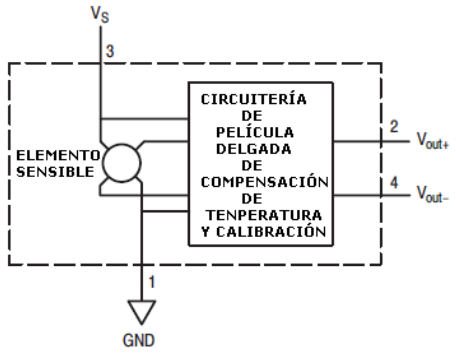


Figura 1.- Esquema del sensor de presión de temperatura compensada y calibrado

La siguiente figura diagrama a bloques del circuito interno del chip de presión independiente del sensor.

Voltaje de salida VS Presión diferencial aplicada.
El voltaje de salida del sensor de presión diferencial incrementa con la presión aplicada al lado de presión (P1) en relación al lado de vacio (P2). Similarmente, el voltaje de salida incrementa tanto como se incrementa el vacio en el lado de vacio (P2) relativo al lado de presión (P1).
Rangos máximos (NOTA)

Parámetros	Símbolo	Valor	Unidades
Presión Máxima (P1 > P2)	Pmax	75	KPa
Temperatura de almacenamiento	Tstg	-40 a +125	°C
Temperatura de operación	TA	-40 a +125	°C

NOTA: La exposición más allá de los límites puede causar daño permanente o degradación del dispositivo.

CARACTERÍSTICAS DE OPERACIÓN (V_S = 10 Vcd, T_A = 25 °C a menos que se indique lo contrario, P_1>P_2)

Características	Símbolo	Min.	Típico	Max.	Unidades
Rango de presión(1)	POP	0	---	10	KPa
Voltaje de alimentación(2)	VS	---	10	16	Vcd
Corriente de alimentación	IO	---	6.0	---	mAcd
Lapso de escala completa (3)	VFSS	24	25	26	mV
Compensación (Offset) (4)	VOFF	-1.0	---	1.0	mV
Sensibilidad	ΔV/ΔP	---	2.5	---	mV/kPa
Linealidad (5)	---	-1.0	---	-1.0	%VFSS
Histéresis de presión(5) (0 a 10 kPa)	---	---	±0.1	---	%VFSS
histéresis de Temperatura(5) (-40°C a +125 °C)	---	---	±0.5	---	%VFSS
Efecto de temperatura en el lapso de escala completa (5)	TCVFSS	-1.0	---	1.0	%VFSS
Efecto de temperatura en el Offset(5)	TCVOFF	-1.0	---	1.0	mV
Impedancia de entrada	ZIN	1000	---	2550	Ω
Impedancia de salida	ZOUT	1400	---	3000	Ω
Tiempo de respuesta (6) (10% a 90%)	tR	---	1.0	---	ms
Calentamiento	---	---	20	---	ms
Estabilidad de Offset (7)	---	---	±0.5	---	%VFSS

NOTAS:
1.0 kPa igual a 0.145 psi.
Este dispositivo es radiométrico dentro del rango de excitación. El uso del dispositivo por encima del rango especificado de excitación puede inducir error adicional debido al calentamiento espontáneo del dispositivo.
El Lapso de escala completa (VFSS) está definido como la diferencia algebraica entre la tensión de salida a la presión nominal completa y la tensión de salida a la presión nominal mínima.
Corrimiento (Offset VFSS) esta definido como la tensión de salida a la presión nominal mínima.
Precisión (error presupuesto) consiste en lo siguiente:
Linealidad: desviación de salida de una relación de línea recta con la presión, utilizando el método de punto final, en el rango de presión especificado.
La histéresis de temperatura: la desviación de salida a cualquier temperatura dentro del rango de temperatura de funcionamiento, después de que la temperatura se realiza un ciclo hacia y desde los puntos de la temperatura mínima o máxima de operación, con la diferencia de presión cero aplicada.
La histéresis de presión: la desviación de salida a cualquier presión dentro del rango especificado, cuando la presión se realiza un ciclo y de la presión mínima o máxima, a 25 ° C.
TcSpan: desviación de salida a la presión nominal máxima en el rango de temperatura de 0 a 85 ° C, en comparación con 25 ° C.
TcOffset: desviación de salida con una presión nominal mínima aplicada, en el rango de temperatura de 0 a 85 ° C, en comparación con 25 ° C.
El tiempo de respuesta está definido como el tiempo para el cambio de incremental en la salida desde 10% hasta 90% de su valor final cuando se somete a un cambio en la presión especificada.
La estabilidad de compensación es la desviación del producto de salida cuando se cometen a 1000 horas de presión pulsada, los ciclos de temperatura con la prueba de sesgo.

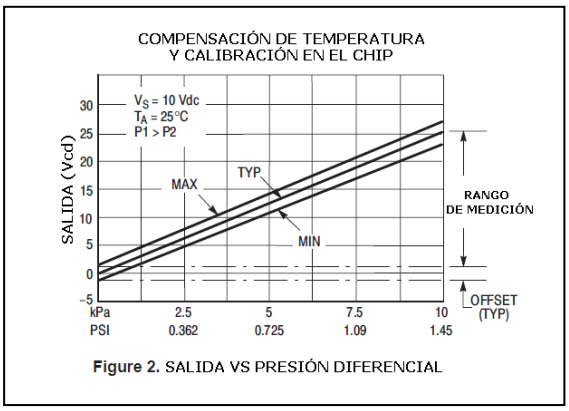


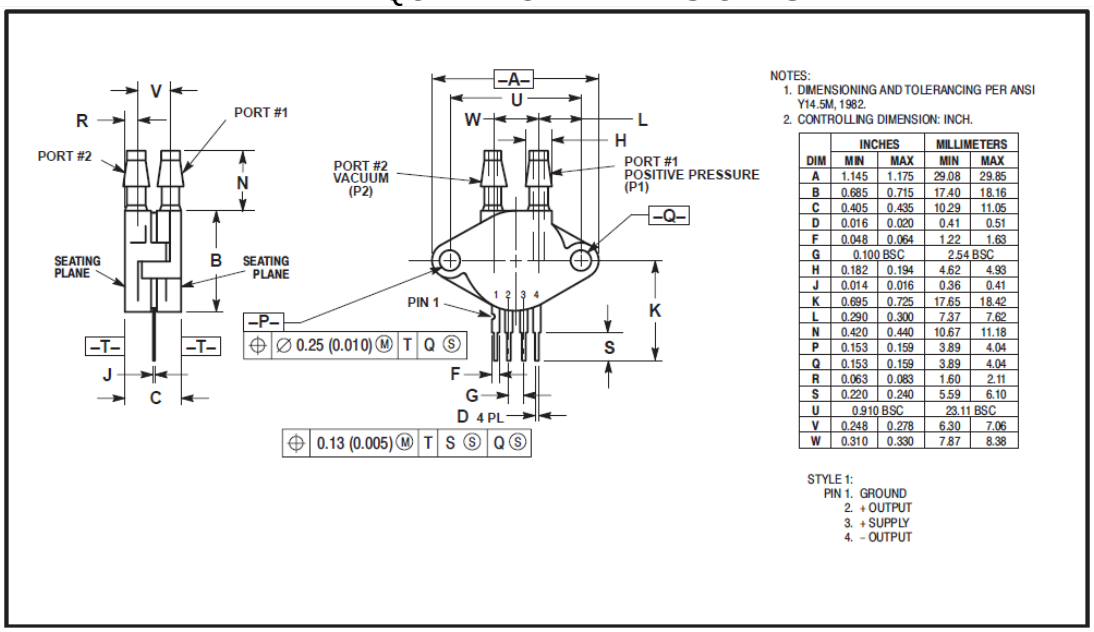
Figure 2. SALIDA VS PRESIÓN DIFERENCIAL

La figura 2 muestra las características de salida del MPX2010DP a 25°C. La salida es directamente proporcional a presión diferencial y es esencialmente una línea recta.

El efecto de temperatura en escala completa de medición y la compensación es muy pequeña y son mostrados en las características de operación.

Este rendimiento sobre la temperatura es logrado por tener ambas

EMPAQUETADO Y DIMENSIONES



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	1.145	1.175	29.08	29.85
B	0.685	0.715	17.40	18.16
C	0.405	0.435	10.29	11.05
D	0.016	0.020	0.41	0.51
F	0.048	0.064	1.22	1.63
G	0.100 BSC		2.54 BSC	
H	0.182	0.194	4.62	4.93
J	0.014	0.016	0.36	0.41
K	0.695	0.725	17.65	18.42
L	0.290	0.300	7.37	7.62
N	0.420	0.440	10.67	11.18
P	0.153	0.159	3.89	4.04
Q	0.153	0.159	3.89	4.04
R	0.063	0.083	1.60	2.11
S	0.220	0.240	5.59	6.10
U	0.910 BSC		23.11 BSC	
V	0.248	0.278	6.30	7.06
W	0.310	0.330	7.87	8.38

STYLE 1:
PIN 1. GROUND
2. + OUTPUT
3. + SUPPLY
4. – OUTPUT