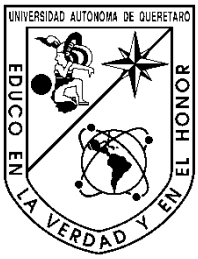

Universidad Autónoma de Querétaro

Facultad de ingeniería

Licenciatura en Ingeniera en Automatización

"Automatización de un prototipo deshidratador de laboratorio para frutas y hortalizas"

Que como parte de los requisitos para obtener el grado de

Ingeniero en Automatización (SIM08)

Presenta:

Tapia Cisneros Juan Carlos

Dirigido por:

Dr. Genaro Martín Soto Zarazúa

Dr. Genaro Martín Soto Zarazúa	_____
Presidente	Firma
Dr. Manuel Toledano Ayala	_____
Secretario	Firma
Dr. Gonzalo Macías Bobadilla	_____
Vocal	Firma
Dr. Yunny Meas Vong	_____
Suplente	Firma

Dr. Aurelio Domínguez González	_____
Director de la Facultad	Firma

Centro Universitario Querétaro, Qro, México "07 de marzo de 2013"

RESUMEN

La presente tesis documenta la instrumentación y control del proceso de deshidratado controlando el flujo y temperatura del aire, así como el registro de las variables involucradas en el proceso de deshidratación. La instrumentación incluye una celda de carga, un medidor de flujo másico y sensores de temperatura y humedad. Así como etapas de potencia para el control de la velocidad del viento y temperatura. La estrategia de control implementado es un control Proporcional Integral Derivativo discreto que utiliza la información trasmitida utilizando en protocolo ZigBee para recuperar la información proveniente de los sensores. Como sistema alterno para el registro de parámetros se utiliza una memoria microSD la cual registra toda la información relevante, el sistema principal para el registro es mediante software que muestra en tiempo real la información mediante gráficos y tablas. Dicha interfaz permite realizar la configuración de los parámetros de control. Final mente se documenta la implementación del protocolo USB-HID para el adaptador USB-ZigBee.

Palabras Clave: Deshidratador de hortalizas, Instrumentación, Control, Interfaz grafica de usuario para deshidratador.

AGRADECIMIENTOS.

- A Dios por la oportunidad.
- A mis padres por su constante apoyo.
- A mis hermanas por su comprensión.
- A mis amigos por sus consejos.
- A mis profesores por los conocimientos compartidos.

ÍNDICE DE CONTENIDO

INDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1:

I INTRODUCCIÓN

El proceso de deshidratado es esencial para la preservación de los productos en la agricultura. Los productos alimenticios, especialmente las frutas y vegetales requieren aire caliente en un rango de 45 – 60 grados centígrados para un deshidratado seguro. El deshidratado bajo condiciones controladas de temperatura y humedad permiten disminuir el contenido de humedad de forma razonablemente rápida sin comprometer la calidad de los productos (Sharma, 1994).

Los deshidratadores solares requieren una gran cantidad de superficie de captación solar para procesar cantidades industriales de producto, un sistema hibrido que involucre quemadores de gas, resistencias eléctricas y colectores solares son la solución ideal para un diseño que este enfocado al procesamiento a gran escala. El uso de sistemas forzados de deshidratación mejoran significativamente la calidad de los productos finales esto debido a que el tiempo de deshidratación se reduce significativamente, y con ello se reduce el tiempo de exposición al medio ambiente al mismo tiempo que se reduce la actividad microbiana. Lo que se traduce en el aumento de la calidad de los productos.

La presente tesis trata de la instrumentación de un deshidratador que será empleado dentro de un laboratorio para realizar diversas pruebas que permitan controlar las variables de temperatura y velocidad de aire para posteriormente diseñar en base a los resultados de laboratorio, un deshidratador industrial hibrido. La instrumentación y control expuesto en la tesis está enfocado a un dispositivo que será usado como herramienta de investigación, es por ello que se requieren gran precisión de los dispositivos de medición y registro de todas las variables monitoreadas para su futuro análisis.

I.1 Justificación

En el mercado existen una gran cantidad de deshidratadores tanto industriales como caseros, la mayoría de ellos cuentan con controles básicos o inexistentes. El proceso de deshidratación requiere mantener un constante flujo de aire caliente, la temperatura debe ser la suficiente para generar la deshidratación sin causar un efecto de cocción en los alimentos, lo que genera la necesidad de la instrumentación y automatización del proceso de deshidratación.

Algunos de los deshidratadores requieren de suministro eléctrico para aumentar la temperatura del aire que fluirá sobre las charolas, otros cuentan con quemadores de gas, y los denominados ecológicos requieren de energía solar para su funcionamiento. Es común encontrar deshidratadores híbridos que unen lo mejor de las tecnologías, siendo la combinación más usual los deshidratadores solares en unión con quemadores de gas, este tipo de dispositivo permite utilizar la energía solar durante el día y los quemadores en horas en las que la radiación no es suficiente.

Entre las formas más habituales para el diseño del deshidratador se encuentran los deshidratadores verticales, los cuales permiten aumentar la cantidad de charolas en relación a la superficie requerida para su instalación, por otro lado, existen los diseños tipo túnel de viento, dicho diseño es ideal para su uso dentro de un laboratorio y facilita el estudio de las condiciones correctas para optimizar el proceso de deshidratación.

I.2 Planteamiento del problema

Al contar con sobre producción, el agricultor de temporal acostumbra desechar parte de la cosecha, lo que representa pérdidas debido al costo de la producción, fuera de temporada el producto aumenta su precio debido a la escases. El empleo del proceso de deshidratación permite el almacenamiento del producto así como su fácil manejo al disminuir su peso y volumen. Siendo estos factores relevantes en la logística que involucra la producción de frutas y hortalizas.

Para resolver los problemas logísticos en el trasporte y almacenamiento de frutas y hortalizas se pretende instrumentar y automatizar un deshidratador prototipo de deshidratador que permitirá realizar los estudios de laboratorio necesarios para optimizar el proceso de

deshidratación. Los resultados de la etapa de experimentación fundamentarán el desarrollo de un deshidratador industrial.

I.3 Hipótesis y objetivos

I.3.1 Hipótesis general

La implementación de un sistema de control retroalimentado basado en un micro controlador permitirá monitorear y controlar el proceso de deshidratación.

I.3.2 Objetivo general

Instrumentar y automatizar un prototipo de deshidratador para frutas y hortalizas que será usado en los laboratorios del Centro de Investigación y Desarrollo Tecnológico en Electroquímica (CIDETEQ) para la realización de pruebas que contribuirán en el desarrollo de un deshidratador industrial.

I.3.3 Objetivos específicos

- Instrumentación de medición acorde a los requerimientos definidos por los diseñadores del CIDETEQ, que serán utilizados para el monitoreo y control del clima en el túnel deshidratador.
- Acondicionamiento de señales para manejar un controlador único que responda simultáneamente a las distintas señales provenientes de cada sensor.
- Diseño y desarrollo del software de Interfaz Maquina Usuario para la recolección, almacenamiento y visualización de información en línea.

CAPÍTULO 2:

II REVISIÓN DE LITERATURA

II.1 Proceso de deshidratación

El termino *deshidratación* se refiere a remover la humedad de un material con el objetivo primario de reducir la actividad microbiana y la degradación (Ratti, 2001). Durante la deshidratación se producen dos procesos de forma simultánea:

- Trasferencia de la energía del medio circundante para evaporar la humedad superficial.
- Trasferencia de la humedad interna a la superficie del sólido para la subsecuente evaporación debido al primer proceso.

La transferencia de energía del medio al sólido puede ocurrir como resultado de la convección, conducción o radiación y en algunos casos como resultado de la combinación de estos efectos (Mujumdar, 2006).

El método de calentamiento por convección es posiblemente el modo más común para realizar el deshidratado en partículas y en pastas solidas. El calor es suministrado mediante aire caliente que es soplado sobre la superficie del sólido. Los deshidratadores que emplean este método son llamados secadores directos.

El método por conducción son empleados en procesos que involucran productos finos o muy húmedos. El calor es suministrado por superficies calientes que contienen o trasportan el sólido. La humedad evaporada es trasportada utilizando corrientes de gas. En comparación a los deshidratadores directos, los deshidratadores por condición o indirectos tienen una mayor eficiencia.

Los deshidratadores por radiación permiten el calentamiento desde el interior del sólido, reduciendo su resistencia a la trasferencia térmica y con ello disminuyendo la energía necesaria para el secado. El principal inconveniente es su alto costo de operación (Mujumdar, 2006).

II.1.1 Generalidades del proceso de deshidratación

El proceso de deshidratación se impulsó en gran medida durante el período de las guerras mundiales, en donde se requería el manejo de grandes cantidades de alimento pero debido a cuestiones de logística este alimento debía ser de fácil manejo, poco volumen y gran contenido nutritivo, por lo que el proceso de deshidratación para su futura rehidratación en el lugar de consumo resulto la solución ideal.

Desde la antigüedad se ha empleado el proceso de deshidratación para conservar frutos para su futuro consumo, el método de deshidratación utilizado era el denominado natural que consiste en exponer el alimento a los rayos solares, este proceso generaba alimentos de baja calidad debido a que no se controlaba la temperatura y el tiempo empleado en el proceso favorecía la descomposición o actividad microbiana. En la actualidad los procesos de deshidratación forzados redujeron el tiempo requerido para la deshidratación de días a horas, y con ello se mejoro la calidad del producto deshidratado.

Durante el proceso de deshidratación se presentan cambios significativos en las propiedades físicas del alimento. Estas propiedades dependen de varios factores como son el pre tratamiento, el contenido de humedad, el método utilizado, y las condiciones de deshidratación. (Krokida et al., 2000). Las propiedades físicas básicas a considerar son:

- Densidad aparente: Para materiales porosos o en polvo se define como la masa de una muestra dividida por su volumen aparente.
- Densidad de partículas: Es la densidad excluyendo todos los poros y es determinada por el radio de la muestra y su volumen real excluyendo los poros.
- Porosidad: La porosidad es la fracción del volumen vacío en una muestra y esta es usualmente estimada a partir de la densidad aparente y real del material. (Boukouvalas et al., 2006; Krokida y Philippopoulos, 2005).

La calidad de los productos alimenticios es realmente importante y lamentablemente el proceso de deshidratación afecta significativamente la calidad de los mismos. En la actualidad las técnicas para deshidratación se están desarrollando para no solo retirar la humedad del alimento con la finalidad de disminuir la actividad microbiana y así aumentar su vida útil, se

trabaja en el desarrollo de procesos que permitan aumentar la calidad del producto conservando las ventajas del proceso de deshidratación como son el aumento de vida útil, disminución de volumen y peso que facilita el trasporte y manejo del producto. Algunas de las características que se pretende mejorar son: aroma, sabor, color, textura y el valor nutricional. Sin embargo, la calidad de la comida puede ser afectada por otros parámetros ajenos como son: el pH, la composición de la comida, los pre tratamientos y la presencia de sal, aceites o solventes (Mujumar,1997), lo que impide el desarrollo de un producto que satisfaga completamente las características de calidad.

II.1.2 Clasificación

La clasificación de deshidratadores se basa en los siguientes criterios:

- Tipo de operación: Pasivo o continuo.
- Presión de operación: vacio, presión atmosférica o alta presión
- Modo de trasferencia de energía: conducción, convección, radiación, por radio frecuencia o una combinación de lo anterior.
- Estado del producto antes de la deshidratación: estacionario, en movimiento, agitado, fluido o atomizado
- Tiempo requerido: poco <1min, medio 1-60min, largo >60min (Araya-Farias et al., 2009).

Los deshidratadores pueden ser agrupados de acuerdo a la fuente de energía:

- Solar: Los deshidratadores soleares de uso industrial requieren de una zona de recolección de radiación solar en la que se calienta aire que es impulsado sobre el sólido a deshidratar.
- Eléctrico: Este tipo de deshidratadores utilizan resistencias eléctricas para calentar el aire que es impulsado sobre el sólido a deshidratar.
- De Gas: Los deshidratadores de gas emplean quemadores para calentar el aire que es impulsado sobre el sólido a deshidratar.

Una tercera clasificación puede ser vista en la Figura II.1, la cual clasifica los deshidratadores en base al método de trasferencia de calor utilizado, la Figura II.1 incluye ejemplos de deshidratadores que emplean cada una de los distintos métodos de calentamiento.

Figura II.1 Calcificación de los deshidratadores de acuerdo al método de trasferencia de calor

Los diseñadores y en general la industria especializada en el área de productos deshidratados prefieren los métodos convencionales debido a la simplicidad de construcción. Es por esto que los deshidratadores híbridos que utilizan quemadores de gas y colección solar son la tendencia dentro de la industria especializada en la deshidratación de alimentos.

II.1.3 Funcionamiento

El funcionamiento de los distintos tipos de deshidratadores varía únicamente en la fuente de calor empleada para su funcionamiento, a continuación se describe de forma general el funcionamiento de los deshidratadores.

Los deshidratadores solares requieren colectores solares para calentar el aire, dependiendo del tamaño y configuración del deshidratador puede ser necesario el empleo de un

soplador (Figura II.2 a) para impulsar el aire desde el punto de colección hasta el deshidratador vertical, en la mayoría de la configuraciones no es necesario el uso del soplador (Figura II.2 b) únicamente se utiliza el flujo termodinámico natural para dirigir el flujo del aire.

Figura II.2 a) Deshidratador solar con soplador b) Deshidratador sin soplador

El uso de sopladores es común en los deshidratadores industriales, en los que se pretende mantener una temperatura elevada en relación al volumen del aire desplazado.

Los deshidratadores de gas (Figura II.3) emplean quemadores u hornillas para calentar el aire que es impulsado por ventiladores axiales. Para este tipo de deshidratadores se debe controlar la temperatura y las emisiones de gases.

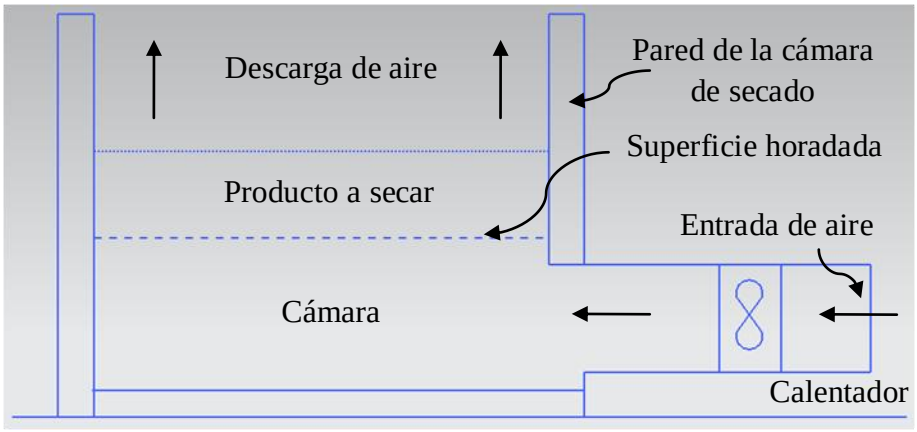

Figura II.3 Deshidratador de gas

Los deshidratadores eléctricos funcionan de forma similar al deshidratador de gas, la principal ventaja es la no emisión de gases teniendo como principal desventaja es el gran consumo energético.

II.1.4 Variables a monitorear

Las variables que deben ser controladas para una correcta deshidratación son:

- Temperatura
- Humedad
- Variación en el peso del alimento
- Velocidad del viento
- Tiempo de proceso

Cada una de las variables anteriores influyen notablemente en el proceso de deshidratación por lo que deberán ser monitoreadas y controladas durante el proceso de deshidratación.

a) Temperatura

Para deshidratar el alimento de forma gradual se debe alcanzar una temperatura de 40 a 70 grados centígrados, para lo cual se requiere un quemador de gas o una resistencia eléctrica que caliente el aire que circulara dentro del deshidratador.

El uso de calentadores solares es recomendado en caso de disponer de un área de colección de radiación amplia. Para igualar la potencia de un calentador comercial eléctrico de 15kw se requeriría una superficie de 15 metros cuadrados, suponiendo un cien por ciento de eficiencia y operación durante la hora de máxima incidencia.

b) Humedad

El rango de temperaturas y presiones usadas en la deshidratación se comportan como gases ideales (Barbosa-Canovas and Vega-Marcado, 1996; Karel and Lund, 2003a; Vega-Mercado et al., 2001).

A continuación se formalizan los conceptos relacionados con el cálculo de la humedad atmosférica:

Presión de vapor: La cantidad de vapor presente en la atmósfera se puede expresar por la presión que ejerce el vapor, *e*, independientemente de los otros gases. La presión total de la

atmósfera es la suma de la presión que ejerce el aire seco más la presión ejercida por el vapor de agua, e (según la ley de Dalton) y la cantidad máxima de vapor que puede presentarse depende de la temperatura ambiente. Cuanto mayor sea la temperatura, más vapor puede contener el aire.

Cuando el aire está saturado de vapor de agua, la presión parcial del vapor de agua, e_s, depende sólo de la temperatura de acuerdo a la ecuación de Clausius-Clapeyron.

$$e_s = 6.11 \cdot 10^{[7.5 \cdot T/(T+237.3)]} \quad (1)$$

donde T se entrega en [ºC] y el resultado , e_s, es en [hPa].

Humedad absoluta: ρv [g/m3], es la densidad de vapor de agua contenido en el aire a una temperatura y presión determinados (masa/volumen):

$$\rho_v = \frac{e}{R_v T} \quad (2)$$

donde *e* esta en [hPa] y T esta en [ºK], Rv = 461 [J/(kg ºK)].

Razón de mezcla: La razón de mezcla, *r* [g/Kg], se define como la razón entre la masa de vapor de agua, ρ_v, y la masa de aire seco, ρ_d:

$$r = \frac{\rho_v}{\rho_d} = 622 \frac{e}{p-e} \quad (3)$$

donde p es la presión atmosférica (medida en hPa).

Humedad específica: La humedad específica, *q* [g/Kg], de una muestra de aire húmedo, representa la cantidad de vapor de agua, ρ_v, contenida en la masa de aire húmedo, $\rho_v + \rho_d$:

$$q = \frac{\rho_v}{\rho_v+\rho_d} = \frac{0.622}{p-0.378e} \quad (4)$$

donde p es la presión de atmosférica en hPa.

Humedad relativa: Es una medida relativa de la cantidad de humedad que el aire húmedo puede contener a una temperatura dada (Araya-Farias et al., 2009). Indica que tan cerca está el aire de la saturación. Se mide en porcentaje entre 0 y 100, donde el 0% significa aire seco y 100% aire saturado:

$$HR = \frac{r}{r_s} \cong \frac{e}{e_s} \cong \frac{q}{q_s} \quad (5)$$

Para realizar el cálculo de la presión de vapor y la razón de mezcla de vapor se puede recurrir al denominado método hidrométrico el cual requiere aplicar la ecuación 1 para calcular la presión de vapor de agua en condiciones de saturación y la ecuación 3 para calcular la razón de mezcla de saturación. Finalmente, empleando la ecuación 5, podemos obtener cualquier término relacionado con la humedad conociendo la humedad relativa, la temperatura de aire y la presión atmosférica.

c) Peso del alimento

El peso del alimento está estrechamente relacionado con la cantidad de agua contenida, para realizar un correcto análisis del alimento se debe considerar la variación del peso durante el proceso de deshidratación.

La Tabla II.1 muestra el contenido de agua de algunos frutos y hortalizas que son candidatos a ser deshidratados debido a su gran contenido de agua.

Tabla II.1 Características de los productos a deshidratar.

Característica	***Jitomate***	***Pepino***	***Zanahoria***
Agua	94%	96%	89%
Hidratos de carbono	3%	2%	7%
Proteína	1%	0.7%	0.9%
Lipidos	0.3%	-	0.2%
Potasio	258mg/100g	140mg/100g	280mg/100g
Sodio	3 mg/100g	8mg/100g	75mg/100g
Calcio	10mg/100g	17mg/100g	41mg/100g
Hierro	0.6mg/100g	0.3mg/100g	0.7mg/100g
Fosforo	24 mg/100g	22mg/100g	34mg/100g
Vitamina C	26 mg/100g	-	6mg/100g
Vitamina A	207 mg/100g	-	-
Tiamina	0.06mg/100g	0.03mg/100g	-
Riboflavina	0.04mg/100g	0.03mg/100g	-
Niacina	28ug/100g	-	-
Grasas	-	0.2%	-

Retinol	-	2mg/100g	1.3mg/100g
Acido ascórbico	-	1mg/100g	-
Acido fólico	-	16ug/100g	-

d) Velocidad del viento

Durante el deshidratado se requiere un flujo constante de aire caliente, para determinar la correcta relación entre la velocidad del viento y temperatura adecuada debemos ser capaces de controlar finamente estas dos variables. Las cuales resultan ser las más importantes para el proceso de deshidratación.

e) Tiempo de proceso

Un sistema automático de deshidratado debe ser capaz de determinar el tiempo requerido para procesar el alimento en base a información básica como puede ser el tipo de fruto a procesar. Para ello se utilizará la información de la celda de carga y la relación entre la humedad de entrada y la de salida del túnel además de modelos preestablecidos basados en experimentación previa.

II.2 Marco teórico

II.2.1 Instrumentación del deshidratador

Se presentan los instrumentos de medición seleccionados para cada una de las variables consideradas, además, se incluyen la etapa de acondicionamiento de señales acordes al instrumento.

(i) Sensor de Humedad y Temperatura: SHT11

El sensor SHT11 (Figura II.4) integra, en una pequeña tarjeta (4.93x7.47mm) de montaje superficial, los elementos para el procesamiento de señales y la salida digital calibrada de fábrica referente a la temperatura y humedad relativa. Utiliza un único sensor capacitivo para medir la humedad relativa, este tipo de sensores detectan los cambios en el dieléctrico que para la aplicación es aire y de acuerdo a la presencia de humedad varia su capacitancia. Además, cuanta con un sensor band-gap para la temperatura, este sensor basa su funcionamiento en la variación de la corriente que pasa por un diodo, la variación de temperatura climática está directamente relacionada con la corriente y voltaje en el diodo. Las señales analógicas de estos sensores son digitalizadas por medio de un ADC de 14bits para finalmente ser enviadas por una interfaz serial integrada en el circuito, con esto se obtiene una sensor de respuesta rápida, con una señal de alta calidad e inmune al ruido.

Figura II.4 Sensor SHT11

(ii) Celda de carga: Modelo D

Una celda de carga analógica está constituida por un bloque de metal, que puede ser de aluminio, hierro o acero inoxidable comúnmente, este, en muchos de los casos tiene un hueco en el centro para facilitar la flexión, misma que es utilizada por un juego de extensómetros

adheridos a la superficie del bloque de metal en las partes más débiles para convertir dicha flexión en una variación de su impedancia.

Estos extensómetros, son conectados en forma de puente de puente de Wheatstone de una forma equilibrada de tal manera que a la salida existan 0 volts, independientemente de el voltaje de excitación con que se cuente.

Una celda de carga debe de contar con un punto de apoyo y un punto de carga, una vez fija en su punto de apoyo y aplicándole una carga esta se deforma, al igual que los extensómetros, los cuales cambian su impedancia y permiten el paso de corriente eléctrica variando la diferencia de potencial proporcionalmente al incremento de la carga y la deformación de la celda.

El modelo D (Donut Shaped Load Cell) o celda de carga tipo dona (ver Figura II.5) es ideal para aplicaciones en la que se requiere pasar la carga directamente a través de la celda. El rango de operación puede variar desde 150 gr a 10 toneladas con una resolución infinita, además, cuentan con compensación de temperatura de 15 ^{o}C a 71^{o}C, siendo su temperatura de operación -54 ^{o}C a 121 ^{o}C.

Figura II.5 Celda de carga Honeywell Modelo D

Debido a que la salida es del orden de 2mV/V es necesario una etapa de acondicionamiento de la señal, la cual consiste en un amplificador des instrumentación INA122 Figura II.6.

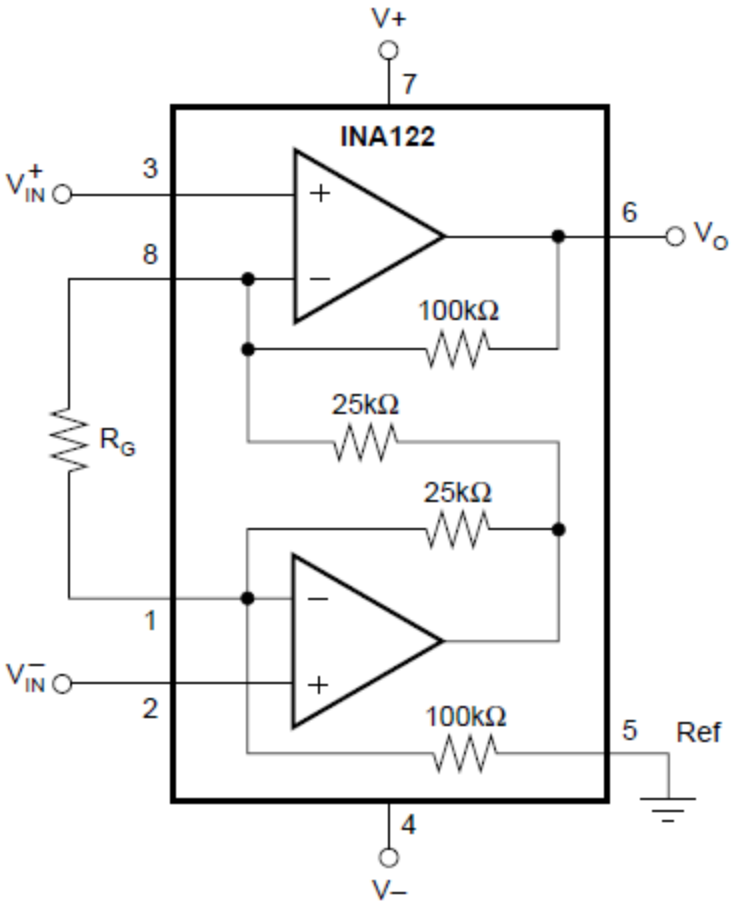

Figura II.6 Amplificador de instrumentación INA122.

$$V_o = (\mathrm{V_{in}^{+}} - \mathrm{V_{in}^{-}})\mathrm{G} \quad (6)$$

$$G = 5 + \frac{200\mathrm{k}}{\mathrm{R_G}} \quad (7)$$

El amplificador INA122 tiene como característica el no requerir alimentación negativa, lo que simplifica su implementación en los circuitos electrónicos.

(iii) Medidor de Flujo: SS 20.260

Se basa en la elevación de la temperatura del fluido en su paso por un cuerpo caliente. Consta de una fuente de alimentación de precisión, que proporciona un calor constante al punto medio del tubo por el cual circula el caudal. En puntos equidistantes de la fuente de calor se encuentran sondas de resistencias para medir temperatura. Con el fluido en reposos la

temperatura es idéntica en las dos sondas, al circular el fluido se entrega una cantidad de calor hacia el segundo elemento y se presenta una diferencia de temperatura, la cual es proporcional a la masa que circula a través del tubo. El sistema está conectado a un puente de medición, que determina la diferencia de temperaturas.

El medidor de velocidad de flujo para aire y gases de la marca SCHMIDT SS 20.260 (Figura II.7), es un sensor que cumple los más altos estándares internacionales, algunas de sus aplicaciones son en el área de seguridad, como monitoreador del aire administrado en los quemadores industriales, y en la detección del la calidad del flujo en procesos de deshidratado.

Figura II.7 Medidor de Flujo SS 20.260 marca Schmidt

El sensor SS 20.260 es colocado en paralelo con el flujo para garantizar que afecte directamente el elemento sensor. Además, permite el monitoreo de temperatura. Las señales de salida son lineales en el estándar 4 a 20mA.

Debido a que la señal es del orden 4 a 20mA, es necesario utilizar una resistencia de 250ohms para genera 5v a 20mA.

II.2.2 Estrategia de control

(i) Sistema de control discreto

Un sistema de control discreto es aquel que incluye un computador digital en el bucle de control para realizar un procesamiento de señal.

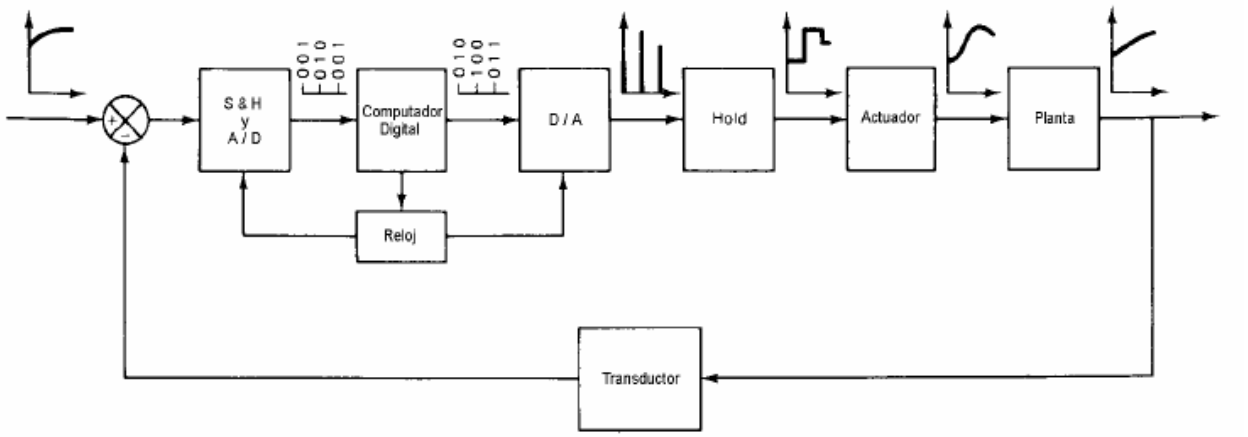

Figura II.8 Lazo de control PID digital.

Como se muestra en la Figura II.8 la salida es continua y realimentada a través de un transductor que convierte la señal de salida en señal eléctrica.

Las etapas del procesamiento se realizan dentro de los siguientes componentes:

- Etapa de codificación (S&H y A/D): Es un circuito de muestreo con periodo figo T para muestrear la señal, la digitaliza mediante un proceso de cuantificación y la mantiene (Figura II.9).

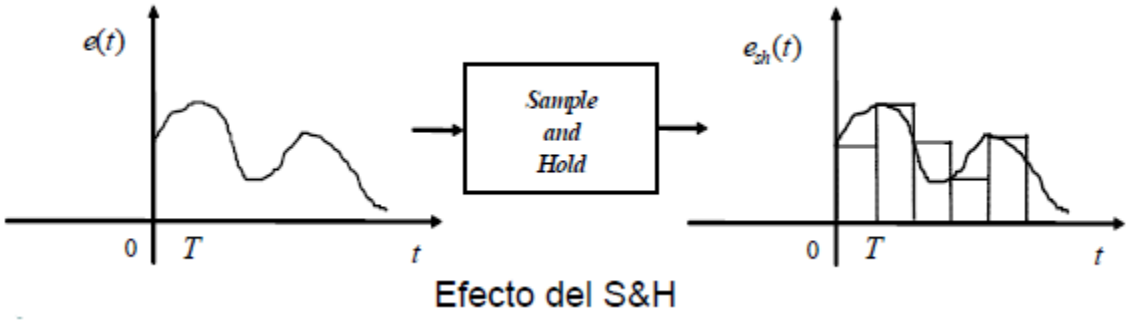

Figura II.9 Etapa de codificación.

- El computador digital (Figura II.10) procesa la secuencia de valores de entrada digital a través de algoritmo y produce una salida digital, según establezca la ley de control.

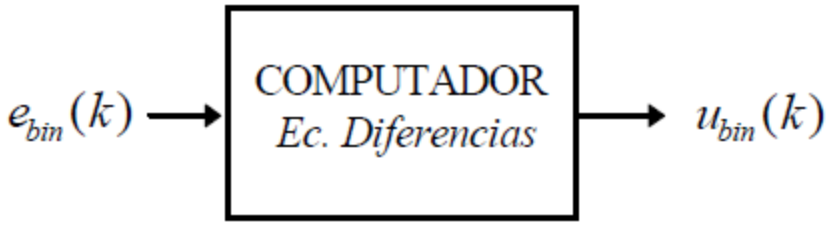

Figura II.10 Etapa de cómputo.

- Convertidor digital/analógico D/A: es un circuito de reconstrucción (Hold) que convierte el valor digital en una señal continua.
- Circuito Hold: Retiene el valor de salida un periodo T.
- Reloj: Mantiene sincronizado el proceso, para ello se define el periodo de muestreo T.

(ii) Controlador PID discreto

El control PID (Figura II.11) es la suma de tres términos: Proporcional al error, Integral del error y la Derivada del error.

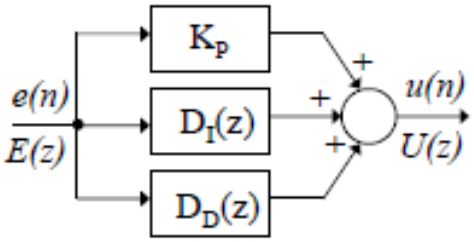

Figura II.11 Controlador PID.

$$U(z) = K_p E(z) + D_I(z)E(z) + D_D(z)E(z) \quad (8)$$

Donde:

K_p: es una constante proporcional

$D_I(z)$: es la integral del error

$D_D(z)$: es la derivada del error

(iii) Implementación del control PID discreto

Para la implementación del control PID discreto se utilizan principalmente dos técnicas: la aproximación rectangular y la aproximación trapezoidal.

La aproximación rectangular se realiza en el dominio analógico y a continuación se transfiere al dominio discreto, gracias a lo anterior su implementación es fácil y de resultados satisfactorios.

Los modelos matemáticos para la implementación del control PID discreto por el método de aproximación rectangular (Figura II.12) se muestran a continuación.

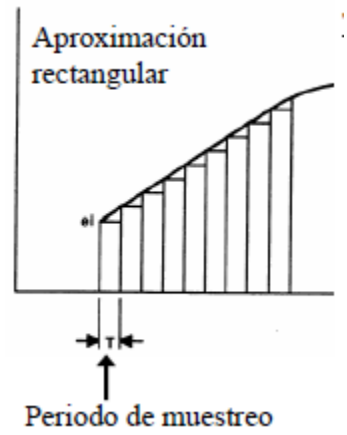

Figura II.12 Aproximación rectangular.

Termino proporcional

$$K_p e(t) = K_p e(n) \tag{9}$$

Termino integral

$$K_i \int e(t) = K_i T \sum_i e_i \tag{10}$$

Termino derivativo

Si T es suficientemente pequeño se puede aproximar por:

$$K_d \frac{e(t)}{dt} = K_d \frac{e(n) - e(n-1)}{T} \tag{11}$$

Si se conoce $e(n-1)$ se puede obtener una mejor aproximación de la derivada:

$$K_d \frac{e(t)}{dt} = K_d \frac{e(n)-e(n-1)}{T} \quad (12)$$

Algoritmo de posición

$$u(n) = k_p e(n) + K_i T \sum_i e_i + K_d[e(n) - e(n-1)]/T \quad (13)$$

Raramente utilizado debido a que puede generar una señal de control igual a cero.

Algoritmo de velocidad

$$\Delta u(n) = u(n) - u(n-2) \quad (14)$$

Al calcular únicamente el incremento en la señal de control no es posible generar una salida cero en caso de falla electrónica.

$$u(n) = u(n-2) + K_1 e(n) + K_2 e(n-1) + K_3 e(n-2) \quad (15)$$

$$K_1 = K_p + K_d/T + K_i T \quad (16)$$

$$K_2 = K_i T + 2K_d/T \quad (17)$$

$$K_3 = K_d/T - K_p \quad (18)$$

II.2.3 Actuadores

(i) Calentador de gas

Los calentadores o quemadores requieren un suministro constante de gas, comúnmente gas propano GLP o natural, son de respuesta rápida o instantánea y requieren instalaciones especiales para conducir los gases de combustión.

(ii) Calentador eléctrico

Los calentadores eléctricos utilizan una resistencia que es calentada al hacer pasar por ella una corriente eléctrica alta. Su respuesta es lenta en comparación a los calentadores de gas pero no requieren instalaciones especiales para manipular el aire calentado, sin embargo debido a la potencia requerida se recomienda utilizar instalaciones eléctricas con capacidad para cargas resistivas del orden de kilowatts.

Las unidades de medida utilizadas en la industria para indicar la capacidad del calentado son los BTU (unidad térmica británica) que es la energía que requiere una libra de agua para

aumentar un grado Fahrenheit. En el sistema internacional se emplean el Julio donde un BTU equivale a mil Julios. Para el sistema técnico se emplean las calorías que es la energía necesaria para elevar un gramo de agua un grado Celsius, la equivalencia Julio Caloria es: una caloría equivale a 4.186.

(iii) Ventilador axial

Un ventilador axial, comúnmente eléctrico, es aquel que es colocado en línea con el proceso, su diseño permite la máxima eficiencia con el mínimo volumen requerido. Se encuentran disponibles en distintos tamaños, siendo esto el factor para la elección del tipo de motor que será utilizado.

II.2.4 Tecnologías de comunicación

(i) Protocolo ZigBee

El protocolo ZigBee usa el estándar 802.15.4 como base y adiciona las funciones de ruteo y manejo de red. Fue diseñado para satisfacer las necesidades comerciales e industriales de aplicaciones que requerían una baja tasa de trasferencia de información. El protocolo ZigBee agrega la posibilidad de trabajar en red, utilizando los módulos como repetidores, lo que amplía su alcance.

En la Tabla II.2 se comparan las distintas tecnologías de comunicación disponibles.

Tabla II.2Tecnologias de comunicación.

	ZigBee y 802.15.4	*GSM/GPRS CDMA*	*802.11*	*Bluetooth*
Aplicación	Monitoreo y control	Voz y datos en áreas amplias	Internet de alta velocidad	Conexión entre dispositivos
Duración de la Batería	Años	Una semana	Una semana	Una semana
Ancho de banda	250Kbps	2Mbps	54Mbps	720Kbps

Alcance típico	100m	Algunos Kilómetros	50-100m	10-100m
Ventajas	Costo y bajo consumo energético	Infraestructura existente	Velocidad y ubicuidad	Conveniencia

(ii) Modulo XBee

Los módulos XBee (Figura II.13) están diseñados bajo el estándar IEEE 802.15.4. Este estándar soporta redes inalámbricas de bajo costo y bajo consumo. Pese a su bajo consumo de energía, la transferencia de datos entre dispositivos es confiable.

Figura II.13 Modulo ZigBee

(iii) Formas de comunicación

La comunicación puede ser de forma trasparente o en modo API (Application Programming Interface). Por defecto, los módulos se encuentran configurados para trasmitir en forma trasparente, esto es, apenas se reciben los datos en el pin DI son enviados por RF, y viceversa, los datos que son recibidos por RF son trasmitidos por el puerto DO. En el caso del modo API, los datos entrantes y salientes son contenidos en tramas que definen los eventos o operaciones de cada modulo.

(iv) Topología de Redes

Las siguientes topologías son soportadas por los módulos XBee/XBee-PRO.

a) Nodo a nodo (Peer-to-Peer)

Por defecto, los módulos xBee están configurados para comunicarse nodo a nodo sin la necesidad de depender de una relación maestro/esclavo definida. Para ello se configura su

propiedad End Device (CE=0) y se deshabilita la asociación con los módulos (A1=0). Además de configurar los parámetros ID y CH idénticos para todos los dispositivos dentro de la red.

b) NonBeacon (w/Coordinator)

Para habilitar esta configuración basta con configurar la propiedad Coordinator Enable (CE=1). Esta configuración es útil cuando se planea enviar información desde uno o más módulos a un modulo (coordinador) central. Para evitar posible interferencia entre redes se debe definir una PAN (Personal Area Network), la PAN ID debe ser única y configurada en todos los dispositivos.

c) Unicast Mode

Por defecto, el modulo RF opera bajo este modo. Al recibir un dato el receptor envía una ACK (acknowledgement), si el transmisor no recibe el ACK re envía el paquete tres veces o hasta recibir el ACK. Esta comunicación se caracteriza por cruzar los valores MY (Source Address) y DL (Destination Address Low) entre los dispositivos para una configuración de 16bits y SH (Source Address High), SL (Source Address Low) y DH (Destination Address High), DL para una configuración de 32bits configurando MY=0xFFFE.

d) Broadcast Mode

Cualquier receptor dentro del alcance del trasmisor puede recibir un paquete siempre que se configure su dirección de destino como broadcast. Esta configuración deshabilita el envío de ACKs.

Para configurar el modulo para este tipo de transmisión se debe colocar DL=0x0000FFFF y DH=0x00000000

(v) Esquemático

Dentro de la documentación del modulo xBee, el fabricante proporciona los siguientes esquemas (Figura II.14), y descripción de pines (Tabla II.3) para la correcta operación del modulo.

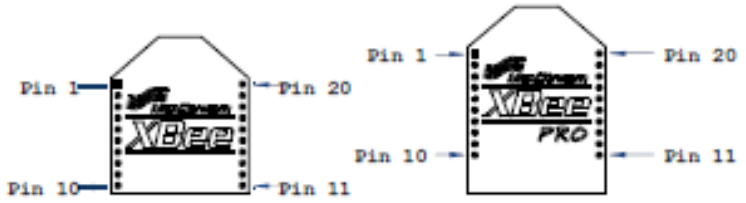

Figura II.14 Esquema de Pines XBee/XBee pro

Tabla II.3 Asignación de pines para el XBee/XBee-Pro.

#Pin	***Nombre***	***Dirección***	***Descripción***
1	VCC	-	Alimentación de voltaje (3.3v)
2	DOUT	Salida	Dato de salida UART
3	DIN/CONFIG’	Entrada	Dato de entrada UART
4	DO8*	Salida	Salida digital #8
5	RESET’	Entrada	Re-Set del modulo (El pulso debe ser al menos de 200ns)
6	PWM0/RSSI	Salida	Salida #0 de PWM / Indicador de la fuerza de la señal RX
7	PWM1	Salida	Salida #1 de PWM
8	[reservado]	-	No conectar
9	DRT/SLEEP_RQ/DI8	Entrada	Pin de control para dormir o señal de entrada #8
10	GND	-	Tierra
11	AD4/DIO4	Bidireccional	Entrada analógica #4 o Entrada/Salida digital #4
12	CTS’/DIO7	Bidireccional	Pin para borrar el buffer de envío o Entrada/Salida digital #7
13	ON/SLEEP’	Salida	Indicador del estado del modulo
14	VREF	Entrada	Voltaje de referencia para los AD
15	Associate/AD5/DIO5	Bidireccional	Indicador de asociación, Entrada analógica #5 o Entrada/Salida digital #6
16	RST’/AD6/DIO6	Bidireccional	Pin para solicitar el envío, Entrada analógica #5 o Entrada/Salida digital #6
17	AD3/DIO3	Bidireccional	Entrada analógica #3 o Entrada/Salida digital #3

18	AD2/DIO2	Bidireccional	Entrada analógica #2 o Entrada/Salida digital #2
19	AD1/DIO1	Bidireccional	Entrada analógica #1 o Entrada/Salida digital #1
20	AD0/DIO0	Bidireccional	Entrada analógica #0 o Entrada/Salida digital #0

II.2.5 Almacenamiento de datos

Las memorias Secure Digital (SD) son tarjetas multimedia de tamaño reducido (11x15mm), se definen como dispositivos de almacenamiento flash removible, diseñadas específicamente para aplicaciones en la que se busca tamaño compacto, bajo consumo y bajo costo.

Algunas de las características básicas de las memorias SD se mencionan en la Tabla II.4.

Tabla II.4 Características de las memorias SD.

Características
Hasta 32Mbytes de almacenamiento
Compatible con el protocolo MultiMediaCard
Soporta SPI
Velocidad variable de reloj de 0 a 20Mhz

(i) Descripción física

Las dimensiones de las tarjetas MicroSD se describen en la Figura II.15.

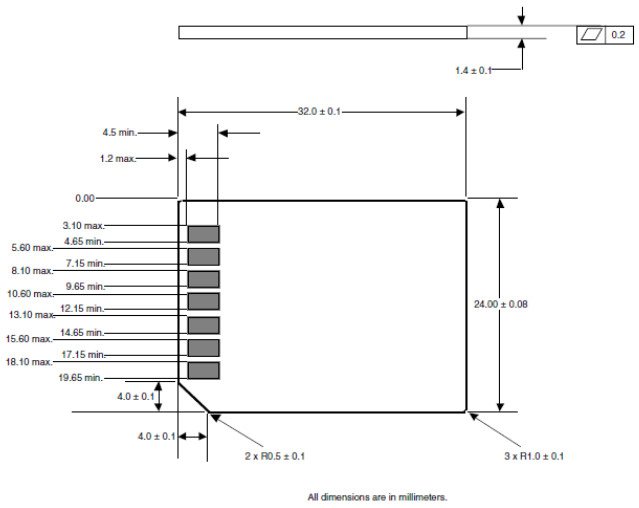

Figura II.15 Esquema y dimensiones de la tarjeta MicroSD.

Las Memorias cuentan con siete contactos expuestos por una de sus caras. El anfitrión se conecta utilizando un conector como se muestra en la Figura II.16.

Figura II.16 Zócalo MicroSD

En la Tabla II.5 se muestra la descripción de cada pin o contacto expuesto.

Tabla II.5 Descripción de pines.

Pin #	***Nombre***	***Type***	***Descripción***
1	CS	Entrada	Selector de chip (Activo en bajo)
2	DataIn	Entrada	Salida de comando y datos del anfitrión a la tarjeta

3	VSS1	Alimentación	Alimentación negativa
4	VDD	Alimentación	Alimentación positiva
5	CLK	Entrada	Reloj
6	VSS2	Alimentación	Alimentación negativa
7	DataOut	Salida	Salida de datos y estado de la tarjeta al anfitrión

Para la topología SPI bus se requieren cuatro señales:

- CS: Señal del anfitrión a la tarjeta, indica la selección del chip.
- CLK: Señal del anfitrión a la tarjeta como señal de reloj.
- DataIn: Señal del anfitrión a la tarjeta. Permite el envío de información.
- DataOut: Señal de la tarjeta al anfitrión. Permite el envío de información.

El diagrama de la Figura II.17 muestra el circuito mínimo para comunicar y operar la MicroSD.

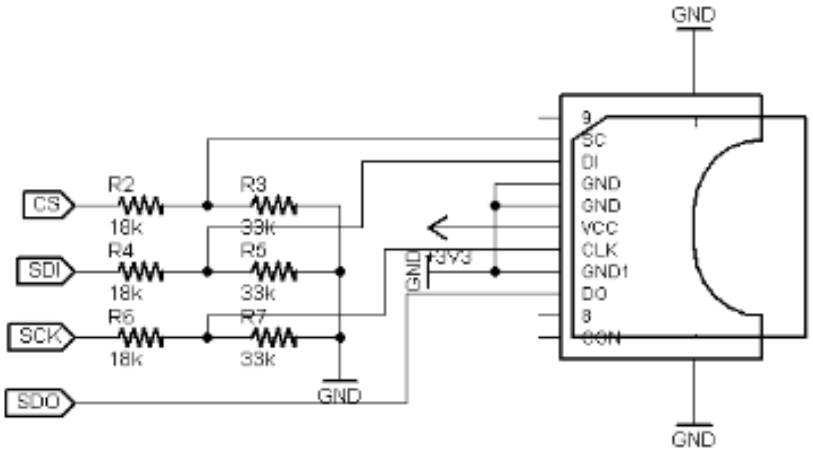

Figura II.17 Diagrama eléctrico para la tarjeta MicroSD.

(ii) Funciones mínimas para su implementación (firmware)

Dentro de la librería "mmcsd.c" disponible en el paquete "Proteus 7 Professional" se encuentran definidas las funciones para implementar una memoria SD en un Micro controlador (18F4550).

a) Función: mmcsd_init()

Envía un tren de pulsos para reactivar a la memoria SD. Si se logra iniciar la SD, la función responde con un 0 (falso), de otra forma responde con un 1(verdadero) para indicar un error de comunicación.

b) Funcion: mmcsd_write_byte(dirección, dato)

Escribe un dato en la dirección indicada, utilizando un buffer de escritura. Se recomienda utilizar mmcsd_flush_buffer() para limpiar el buffer.

CAPÍTULO 3:

III METODOLOGÍA

III.1 Metodología

El prototipo sobre el que se realiza la instrumentación y control se describe en la Figura III.1. Las dimensiones aproximadas del prototipo son 50x50x250cm, el diseño es en forma de túnel de viento debido a que este diseño facilita la caracterización de la relación de velocidad de viento contra calor. En un diseño tipo torre, la caracterización requeriría considerar las variaciones de la temperatura en base a las distintas corrientes generadas internamente.

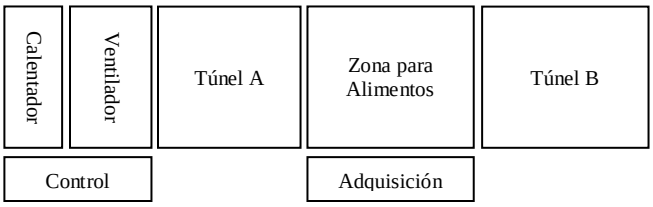

Figura III.1 Diagrama general del prototipo de deshidratador.

El prototipo requiere una etapa de calentamiento, un ventilador y sensores dentro de los túneles A, B y en la zona de alimentos. La etapa de adquisición de datos se encuentra localizada por debajo de la zona de alimentos, dicha etapa trasmite la información de forma inalámbrica a la etapa de control que se encuentra localizada sobre el calentador, el cual cuenta con resistencias eléctricas y un ventilador axial, el control interpretara la información para realizar el control sobre los dos actuadores contemplados. El desarrollo del proyecto se dividió en siete etapas:

- Selección de instrumentos de medición y actuadores.
- Selección de la estrategia de control.
- Algoritmo de control.
- Almacenamiento de datos.
- Diseño de la tarjetas para monitoreo y control.
- Diseño del software de Interfaz Máquina Usuario.

- Pruebas al sistema.

A continuación se desglosa cada uno de las etapas del proyecto.

III.1.1 Selección de instrumentos de medición y actuadores.

Para la elección de instrumentos se considero las condiciones a las que se someterían durante su uso. Los instrumentos deben operar en un ambiente no corrosivo a presión ambiente con variaciones térmicas dentro del rango de 40 a 70 grados centígrados, procurando una velocidad de respuesta alta sin comprometer su precisión.

III.1.2 Selección de la estrategia de control.

Debido a que se desconoce el modelo real de la planta, el control debe ser capaz de adaptarse a cualquier cambio en la dinámica del sistema. Por ello se propone utilizar un control PID discreto con la posibilidad de modificar sus constantes desde el panel principal o desde software.

El control responde a la información reportada desde la etapa de adquisición formada por seis sensores de temperatura y humedad, un sensor de peso y un sensor de flujo, dicha tarjeta de adquisición promedia las seis temperaturas y humedades, la información es trasmitida en una trama ordenada separada por comas para su correcta interpretación, toda la informa es registrada por la microSD al mismo tiempo que se utiliza en el algoritmo de control que genera la señal PWM que es utilizada por los Triacs para activar o desactivar el calentador y el ventilador respectivamente.

III.1.3 Etapas de control.

El prototipo es capaz de operar sin utilizar un CPU, además cuneta con la posibilidad de operar multitarea para evitar retrasos al realizar las operaciones de escritura, censando y control. Se propuso el desarrollo de un sistema modular para realizar las operaciones de adquisición de datos y control de forma simultánea.

III.1.4 Almacenamiento de datos.

El dispositivo primario para realizar el registro de información es el CPU, si la computadora se encuentra conectada registrar la información y despliega los gráficos correspondientes. Alternativamente se cuenta con la información registrada en la microSD la cual almacenara los datos para eventualmente ser transmitidos al CPU para su registro.

III.1.5 Diseño de las tarjetas para monitoreo y control.

Se diseñaron las dos tarjetas requeridas por el algoritmo de control para realizar las actividades de monitoreo y control. La tarjeta de monitoreo realiza el tratamiento necesario a las señales, así como el empaquetado y trasmisión de información. Por su parte la tarjeta de control interpreta la información proveniente de la tarjeta de adquisición de datos y generar las señales de control para los actuadores.

III.1.6 Diseño del software de Interfaz Máquina Usuario

El software tiene la capacidad de almacenar información y visualizar en tiempo real la información procedente de los sensores.

La interfaz incluye gráficos y tablas con la información actualizada de acuerdo a la tasa de muestreo definida por el usuario. Los parámetros tales como setpoint, tasa de muestreo, nombre y dirección de los archivos de registro, entre otros, deben ser configurables desde dicha interfaz.

III.1.7 Pruebas al sistema.

Las pruebas se realizaron dentro del laboratorio de instrumentación y control de la Universidad Autónoma de Querétaro. Dichas pruebas consisten en la obtención del parámetro de incertidumbre en el caso de los sensores de temperatura, peso, y velocidad del viento. Adicionalmente se probo la respuesta del sistema de control a una serie de datos enviados desde el CPU, dichos datos validaran la respuesta del sistema para una planta experimental.

Para el sensor de peso se propuso utilizar pesas calibradas como patrón para validar los datos registrados por el micro controlador. Se utilizarían tres pesas distintas en cinco corridas dejando 30 segundos para la estabilización de la señal, los datos resultantes se registrarían en tablas y se les daría tratamiento estadístico para determinar la incertidumbre. Previo a las pruebas se realizaría una validación para determinar la correcta operación del amplificador utilizado, para ello se recurriría a una fuente de voltaje y a un osciloscopio el cual permitiría comparar las señales de entrada y salida para determinar la correcta amplificación de la señal.

La etapa de control de temperatura, velocidad y validación del sensor de flujo se realizaría utilizando un secador de pelo comercial, del cual se extraerían los componentes principales tales como: resistencia eléctrica y motor. Las pruebas permitirían probar la respuesta del sistema de control y la correcta operación de todas las partes involucradas en el prototipo real.

CAPÍTULO 4:

IV RESULTADOS Y DISCUSIÓN

IV.1 Selección de instrumentos de medición y actuadores.

Como ya se menciono en el capítulo 3: Metodología en el apartado Instrumentación, los sensores seleccionados son:

(i) Medidor de Flujo: SS 20.260

Para medición de flujo se propuso el sensor SS20.260 de la marca Schmidt Technology que opera en un rango de 0.2 a 50m/s con una precisión de +-5% teniendo una salida de 4 a 20mA con comportamiento lineal. El sensor permite obtener el flujo del aire en un túnel abierto, esto es, el sensor puede ser instalado en la salida a medio ambiente.

(ii) Celda de carga: Modelo D

La celda de carga seleccionada fue el Modelo D marca Honeywell con capacidad máxima de 10lb, el diseño cuenta con Thru-Hole lo que permite "colgar" una canastilla que contenga las charolas con material a deshidratar.

(iii) Sensores temperatura y humedad: SHT11

El sensor SHT11 fue seleccionado debido a su fácil manejo y gran precisión bajo las condiciones a las que se somete dentro del proceso.

(iv) Calentador: 1RKT3

Como calentador se propuso utilizar el modelo 1RKT3 de la marca Dayton, el cual genera un flujo de 790cfm a 51216 BTU/Hr con un consumo de 15kW de una toma de 240v.

IV.2 Selección de la estrategia de control.

Se implemento un control PID analógico para controlar la activación del ventilador axial y el calentador eléctrico. Ambas salidas se controlan de forma independiente utilizando

para ello una etapa de potencia formada por Triacs con numero de parte BTA06600 los cuales se encuentran acoplados ópticamente con la tarjeta de control mediante el MOC3010.

IV.3 **Etapas de control.**

En el diagrama de la Figura IV.1 se muestran las distintas tarjetas y sus respectivos periféricos.

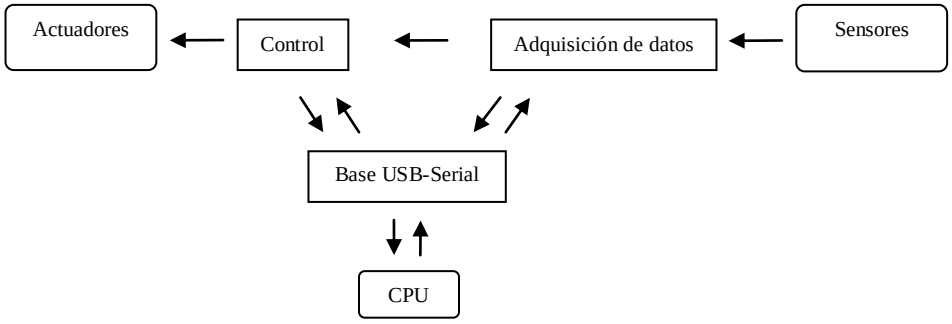

Figura IV.1 Diagrama de las etapas de control.

La etapa de adquisición de datos procesa seis sensores SHT11 utilizando comunicación SPI, tres entradas analógicas, dos para el sensor de flujo y una para la celda de carga, los datos son procesados y transmitidos en una única trama que contiene la información y los promedios de humedad y temperatura. La trama trasmitida es descrita en la Tabla IV.1, los distintos tipos de datos son separados por comas, lo que simplifica la posterior interpretación de la información contenida en la trama.

Tabla IV.1 Descripción de la trama envidad por la tarjeta de adquisición de datos.

#Dato	***Tipo de carácter***	***Carácter o formato***	***Descripción***
1	Char	$	Indica el inicio de la trama
2	Float	0.2	Promedio de las temperaturas reportadas por los seis sensores SHT11
3-8	Float	0.2	Temperaturas reportadas por los sensores SHT11
9-10	Long unsigned	Lu	Valor de los ADCs 1 y 2 dedicados al sensor de flujo

11	Long unsigned	Lu	Valor del ADC 3 dedicado a la celda de carga
12	Flotante	0.2	Promedio de las humedades relativas reportadas por los seis sensores SHT11
13-18	Flotante	0.2	Temperaturas reportadas por los sensores SHT11
9	Char	\r\n	Indica el fin de la trama

La velocidad de muestreo es configurable desde el CPU, por default se manejara un segundo como tiempo mínimo entre muestreos.

La etapa de control interpreta la información proveniente de la etapa de adquisición, los datos son almacenados en la microSD que funciona como respaldo de información. Los parámetros para realizar el control son modificables desde el teclado matricial y desde el CPU.

La tarjeta USB-Serial es la encargada de transmitir las configuraciones realizadas desde el CPU, además recibe la información desde los módulos de control y adquisición de datos.

IV.4 **Almacenamiento de datos.**

El software de monitoreo genera un archivo en la dirección definida por el usuario, dicho archivo con formato separado por comas (.csv) contiene toda la información trasmitida por parte de la tarjeta de control.

La tarjeta de control registra la información en la microSD durante todo el experimento, la información contiene todo lo reportado desde la tarjeta de adquisición, dicha información permanece en la memoria SD hasta que utilizando la opción B disponible desde la pantalla principal en la tarjeta de control, inicie la trasmisión de toda la información contenida en la memoria microSD. De esta forma se puede apagar la computadora y dejar grabando los datos en la memoria SD para al finalizar el experimento descargar todos los registros al software.

La memoria microSD puede contener hasta 32GB de información, lo que equivale a 53333 registros de 60 bytes, lo que permite un registro de 14 horas registrando una trama de 60 bytes cada segundo.

IV.5 **Diseño de tarjetas para monitoreo y control**

Las tarjetas se diseñaron utilizando el software Proteus 7 Professional. Los PCB's (Printed Circuit Board) se maquinaron utilizando un equipo ProtoMat. A continuación se explica detalladamente cada una de las tarjetas.

IV.5.1 Tarjeta para monitoreo

La tarjeta fue diseñada utilizando componentes electrónicos comunes de la clase DIP, lo que facilitara su adquisición. El micro controlador seleccionado es el PIC18f4550, dicho micro controlador cuenta con suficientes salidas/entradas para leer dos señales analógicas: temperatura y flujo proveniente del sensor Schmidt technology - SS 20.260 y una señal acondicionada proveniente de la celda de carga. Además, cuenta con entradas independientes para los seis sensores SHT11.

La información recuperada de los sensores es transmitida utilizado el siguiente formato: "$,%0.2f,%0.2f,%0.2f,%0.2f,%0.2f,%0.2f,%0.2f,%Lu,%Lu,%Lu,",AD1,AD2,AD3,%0.2f,%0.2f, %0.2f,%0.2f,%0.2f,%0.2f,%0.2f,\r\n". La trama contiene la información de cada uno de los sensores y un promedio de temperatura y humedad la descripción completa de la trama puede ser consultada en la Tabla IV.1. El intervalo entre reportes y la cantidad de muestras es configurable desde la interfaz de usuario.

Como parte del acondicionamiento de la señal proveniente de la celda de carga se coloco un amplificador operacional INA122. Para el acondicionamiento de la señal de 4-20mA generada en el medidor de flujo, se utilizo una resistencia de 240ohm para generar el voltaje que es leído ADC del PIC.

(i) Descripción general

La función de la tarjeta de monitoreo es adquirir la información de los ocho sensores conectados, codificar y transmitirla. Para lograrlo se colocaron seis conectores para sensores SHT11, una entrada para sensor de 0-10v y una para 4 a 20mA. Los componentes son visibles en la siguiente Figura IV.2 y descritos a continuación.

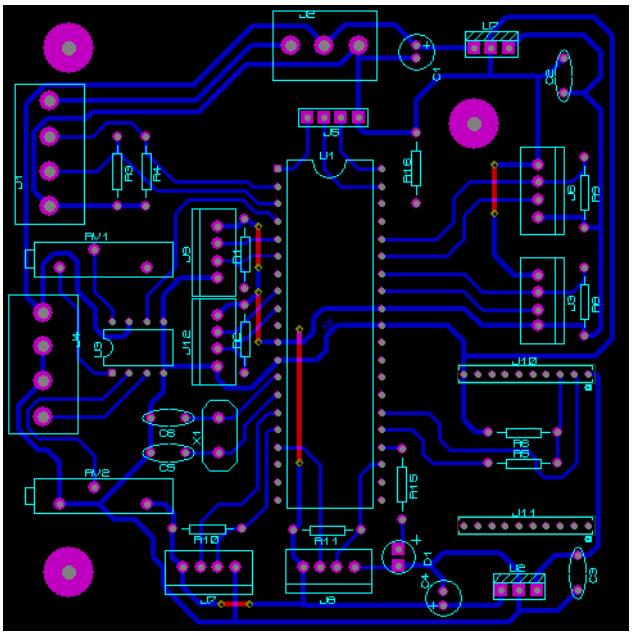

Figura IV.2 Tarjeta para monitoreo

Los conectores J3, J6, J7, J8, J9 y J12 son los conectores para el sensor SHT11, los cuatro pines son VDD, GND, SCK y DATA. Como parte del circuito minimo para la operación del sensor SHT11 se requiere una resistencia de 10Kohm como pull up en el pin DATA. Los conectores usados cuentan con poka-yoke que evita la posible conexión incorrecta de los sensores.

La Figura IV.3 muestra las placas que se utilizaron como conectores para el sensor, el diseño es mínimo, con ello se pretende ocupar el menor volumen para no alterar las condiciones del experimento al introducir una sonda de gran tamaño.

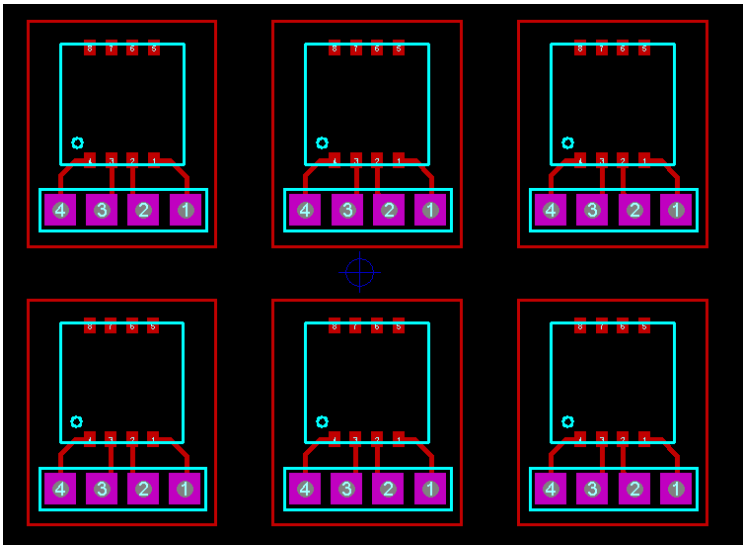

Figura IV.3 Tarjeta de montaje para los sensores SHT11

El conector J1 está configurado para leer el sensor de flujo SS 20,260, dicho sensor requiere alimentación de 24v, y entrega dos señales de 4 a 20mA para ello se coloco una resistencia a la salida para generar voltaje que es leído por los analógicos cero y uno del micro controlador.

El conector J4 está configurado para leer la celda de carga, la cual requiere +10v y dos líneas GND, la salida (2mV/V) puede generar hasta 24mV por lo que se requirió un amplificador operacional (U6) para generar un voltaje que pudiera ser leído por el micro controlador.

El Trimpot RV2 genera un voltaje en la terminal negativa del amplificador para lograr amplificar únicamente la diferencia de peso sin considerar el peso constante de la canastilla. Dicho de otra forma, se coloca en la terminal negativa un voltaje que corresponde al peso de la canastilla sin alimento, la terminal positiva es alimentada con el voltaje proporcional al peso de la canastilla mas el peso del alimento, con este arreglo se logra eliminar la constante de peso de la canastilla y únicamente amplificar el peso del alimento, con ello se logra una mayor resolución en las mediciones.

El conector J10 y J11 forma el zócalo para el XBee, se requieren un divisor de voltaje para pasar de 5v a los 3.3 que soporta el xBee en el pin DIN.

Los componentes U2 y U7 son reguladores de voltaje de 3.3 y 5v respectivamente, cada uno cuenta con capacitores para rectificar y mantener el valor de los voltajes.

(ii) Diagrama eléctrico

La Figura IV.5 muestra el diagrama eléctrico de la tarjeta de adquisición y transmisión.

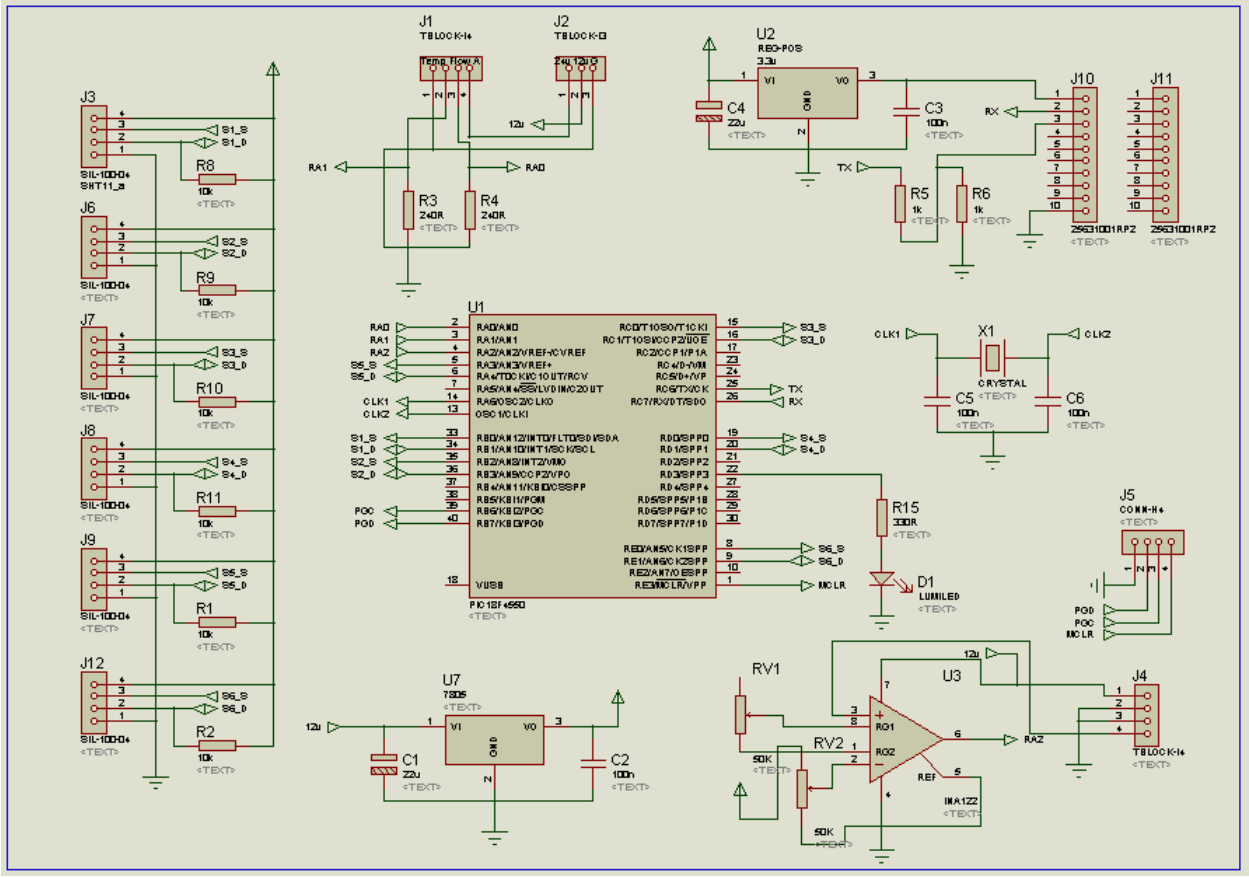

Figura IV.4 Diagrama eléctrico de la tarjeta de adquisición y transmisión.

(iii) Diagramas de flujo

La Figura IV.5 muestra el diagrama de flujo del firmware para la tarjeta de monitoreo. En él se menciona la variable "tiempo" que es declarada de forma global para poder ser accedida desde cualquier punto en el programa, esta variable se utiliza para controlar la velocidad de muestreo, la velocidad mínima de muestreo quedo definida a una muestra por segundo. Esto debido al tiempo requerido para leer los distintos sensores y entradas analógicas.

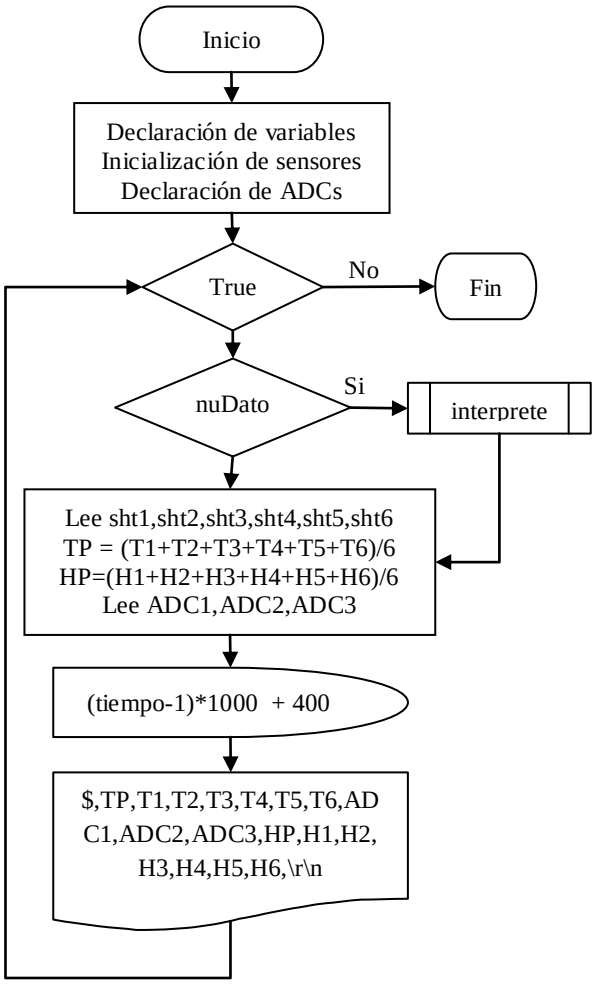

Figura IV.5 Diagrama de flujo para el firmware de la tarjeta de monitoreo.

La Figura IV.6 muestra el diagrama de flujo para la subrutina "interprete". Las variables tiempo (tiempo de muestreo), bLCD (habilita la LCD) y bSR (habilita la transmisión de información detallada) pueden ser modificado al recibir por el puerto serial la trama "&,a,b,c\r" en donde el símbolo & indica el inicio de la instrucción y sirve como identificador de comando, el "a" es el tiempo en segundos entre reportes, "b" y "c" son variables booleanas que habilitan la LCD y el envío de información detallada a través del puerto serial.

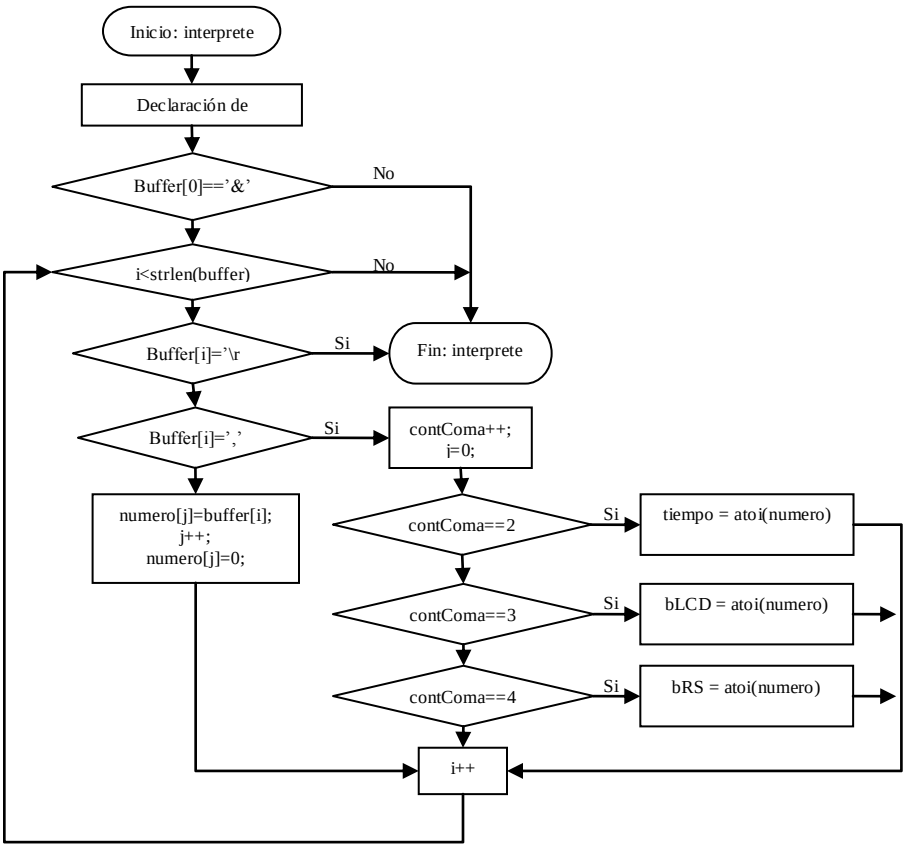

Figura IV.6 Diagrama de flujo para la subrutina “interprete”.

El código puede ser consultado en el apéndice A.

(iv) Memoria fotográfica

La Figura IV.7 muestra en una vista superior la tarjeta de adquisición y transmisión de datos, en ella se aprecian los conectores para los distintos sensores y las conexiones mediante clemas para los cables de alimentación y a sensores de flujo y carga.

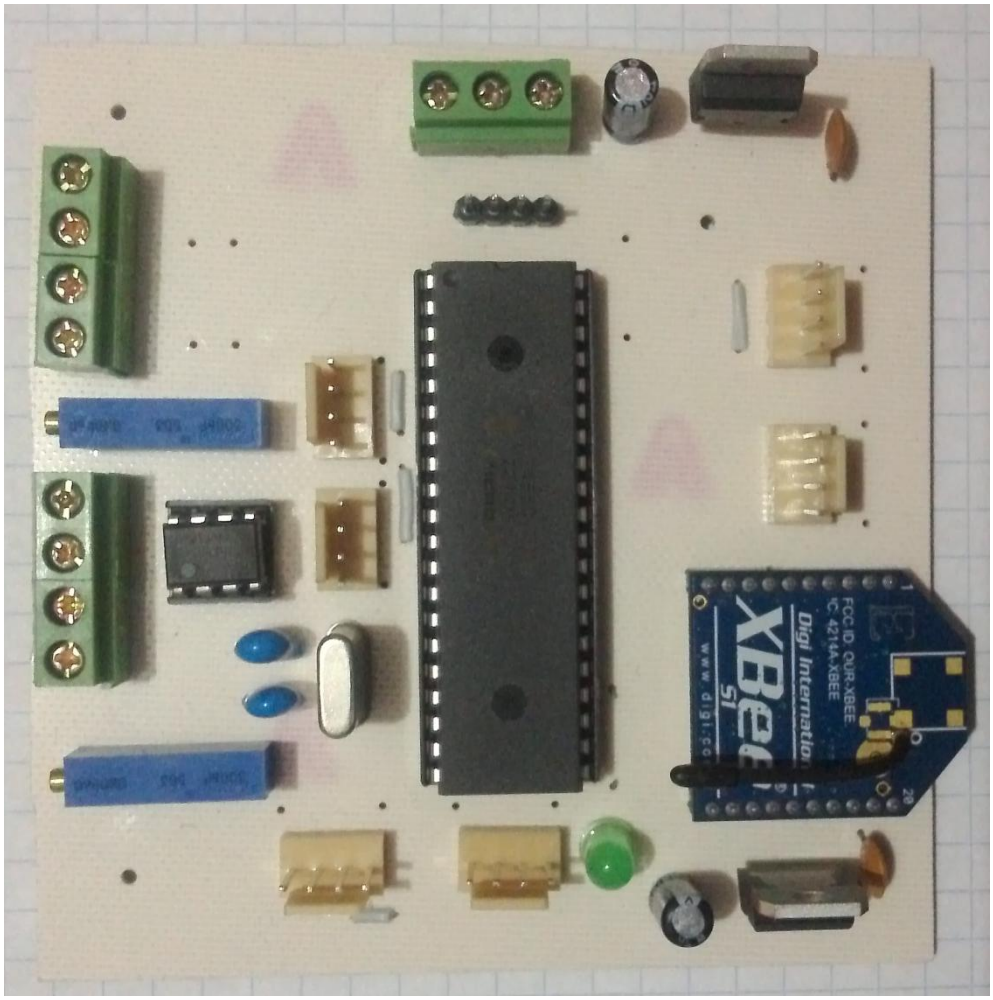

Figura IV.7 Vista superior de la tarjeta de adquisición y transmisión de datos.

La Figura IV.8 muestra uno de los seis sensores SHT11, la tarjeta mide 10x15mm, en ella esta soldado el sensor SHT11 y los cuatro pines macho a 90 grados.

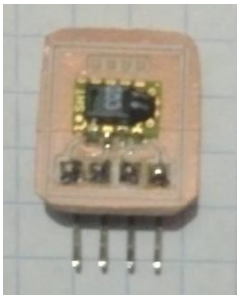

Figura IV.8 Vista superior de la tarjeta para el sensor SHT11.

IV.5.2 Tarjeta para control

La tarjeta de control y registro cuenta con etapas de potencia genéricas con capacidad para modular el ancho de pulso de la señal de control. Además, cuenta con una memoria microSD para el registro de las variables climáticas reportadas por la tarjeta de adquisición de datos. Incluye una pantalla LCD para visualizar las distintas variables de forma inmediata desde el panel de control, para ayudar en la navegación entre pantallas se cuenta con un teclado matricial con el que se facilita la interacción hombre maquina.

(i) Descripción general de componentes

La principal función de la tarjeta de control es el correr el algoritmo para realizar el PID analógico, además, cuenta con registro de eventos y capacidad para cambiar los paramentaros y activar actuadores desde un teclado matricial. Los componentes principales se pueden observar en la Figura IV.9 y son descritos en la Tabla IV.2.

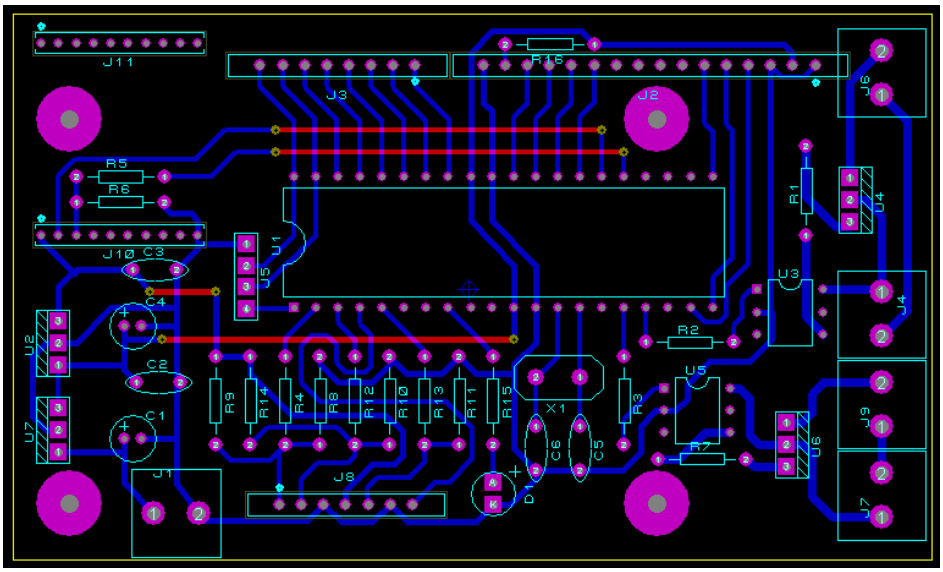

Figura IV.9 Tarjeta de control

Tabla IV.2 Descripción de componentes principales de la tarjeta de control.

Nombre	*Descripción*	*Notas*
J1	Clema	Alimentación 10v GND
J2	Tira de pines macho	Conector para LCD

J3	Tira de pines macho	Conector teclado 4x4
J4	Clema	Salida de control PWM 220v CA
J5	Tira de pines macho	Conector ICSP
J6	Clema	Entrada 220v CA
J7	Clema	Salida de control PWM 220v CA
J8	Tira de pines	Conector para MicroSD
J9	Clema	Entrada 220v CA
J10 y J11	Tira de pines milimétricos hembra	Zócalo para XBee
U1	Micro controlador	PIC18F4550
U2	Regulador de voltaje	Regulador 3.3v
U3	Opto acoplador	MOC3010
U4	Triac	BTA06600
U5	Opto acoplador	MOC3010
U6	Triac	BTA06600
U7	Regulador de voltaje	Regulador 5v

Para alojar a la memoria microSD se requirió el diseño de una tarjeta (Figura IV.10) la cual únicamente cuenta con el espacio para el zócalo y la tira de pines macho.

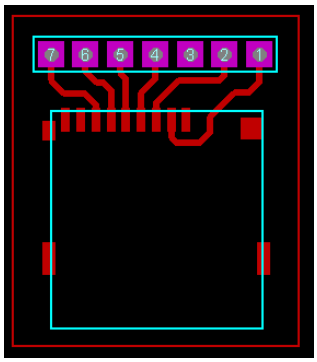

Figura IV.10 Tarjeta para la memoria microSD

(ii) Diagrama eléctrico

La Figura IV.11 muestra el diagrama eléctrico de la tarjeta de control.

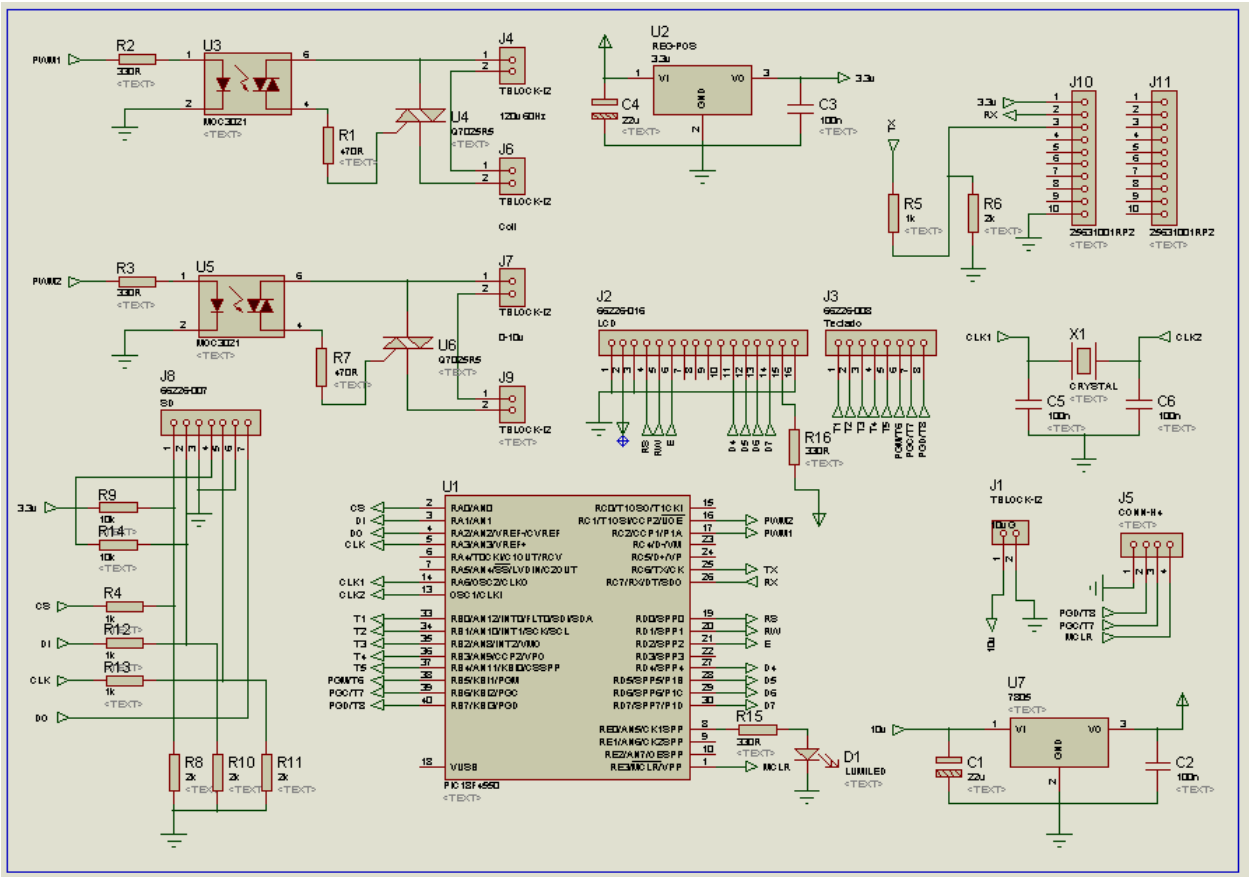

Figura IV.11 Diagrama eléctrico de la tarjeta de control.

(iii) Diagramas de flujo

La Figura IV.12 muestra el diagrama de flujo para el firmware de la tarjeta de control. En dicho diagrama se mencionan las subrutinas que se utilizan para configurar parámetros tales como el setpoint y las constantes involucradas en el cálculo del PID.

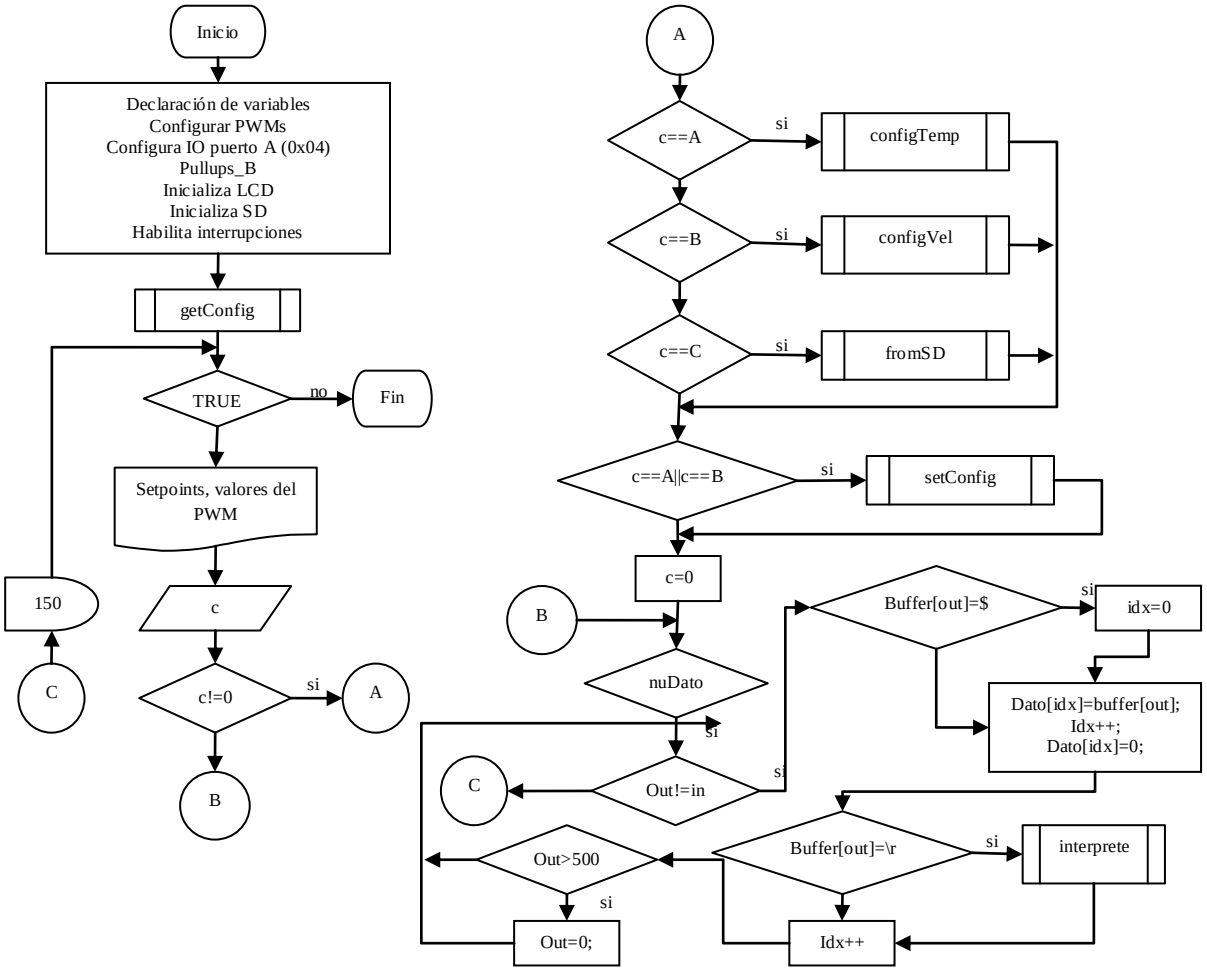

Figura IV.12 Diagrama de flujo para el firmware de la tarjeta de control.

La Figura IV.13 muestra en detalle cómo se realiza el proceso de interpretación de los datos trasmitidos por la tarjeta de monitoreo. Dicha tarjeta envía la trama con símbolo inicial '$', este símbolo es requerido para indicarle al programa que tipo de interpretación debe ser aplicada. Los datos provenientes en la trama con inicial '$' son interpretado y utilizados para computar el nuevo valor del PWM que es la señal de control resultante del control PID discreto.

La trama recibida con el carácter inicial '#' es enviada desde el software, la trama contiene la información referente a los setpoints y a los valores requeridos por el control PID para su computo. Esto dota al programa de la posibilidad de ser programado desde software y directamente desde hardware utilizando el teclado y la LCD que se encuentran incorporados a la tarjeta de control.

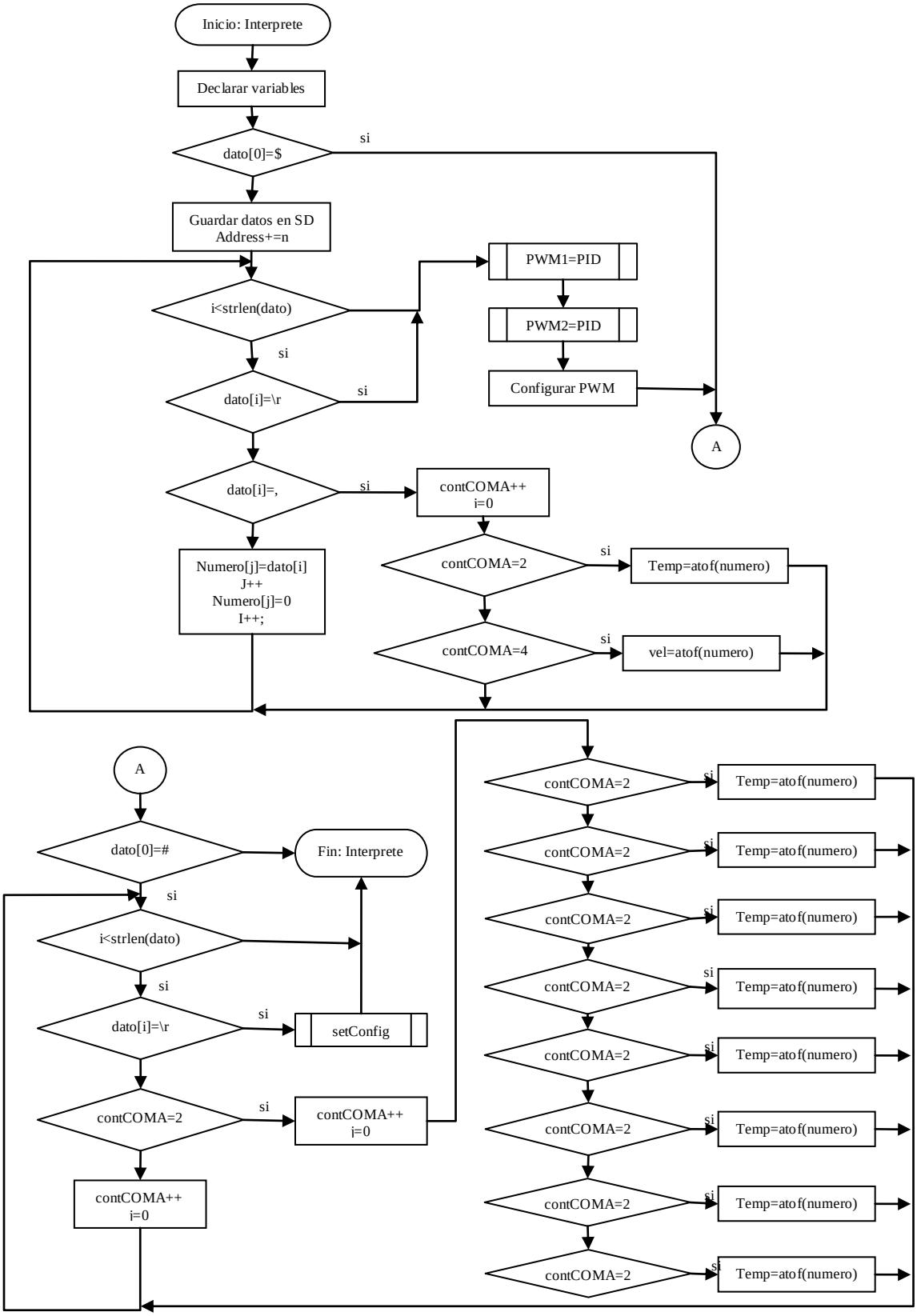

Figura IV.13 Diagrama de flujo para el subproceso "interprete".

El código completo puede ser consultado en el apéndice B.

Para facilitar la visualización de los valores de control y hacer la configuración de parámetros de forma fácil accesible se doto al dispositivo de control con una LCD de 4x20 caracteres y un teclado matricial. Al momento de ser alimentado, la tarjeta de control lee las configuraciones almacenadas en la memoria eeprom, con ello se inicializa las variables de control.

Al terminar la inicialización de la microSD se muestra la pantalla principal (Figura IV.14), en la cual se muestra el valor del SetPoint y el valor de la señal de control (PWM).

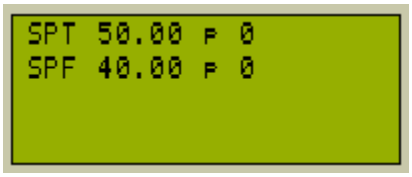

Figura IV.14 Pantalla principal, en espera de información.

El teclado matricial cuenta con las teclas A, B, C, D y E. Cada letra corresponde a una acción distinta, siendo la tecla "A", la opción para configurar los parámetros de setpoint y constantes para el control PID relacionado con la Temperatura, la tecla "B" para la configuración de parámetros relacionados con el Flujo y C para leer y trasmitir la información contenida en la microSD. La Figura IV.15 muestra el dialogo mostrado al entrar en la opción "A" o "B", en ambas opciones se solicita capturar los valores de SetPoint, el de las constantes Ke, Kd y Ki, para determinar la variable que se está modificando se coloca el símbolo "<" en el renglón que se está modificando. Para cambiar el renglón se utilizan las teclas "B" para avanzar al renglón superior y "A" para avanzar al renglón inferior.

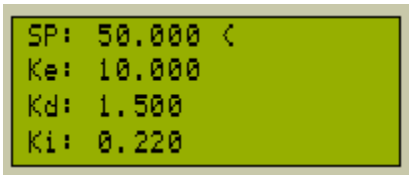

Figura IV.15 Pantalla de configuración.

Las variables pueden ser flotantes o enteras, para introducir el carácter punto se utiliza la tecla "*", para terminar y guardar los cambios se utiliza la tecla "#", si no se quiere guardar los cambios o salir se utiliza la tecla "D".

(iv) Memoria fotográfica

En la Figura IV.16 se muestra en una vista superior la tarjeta de control.

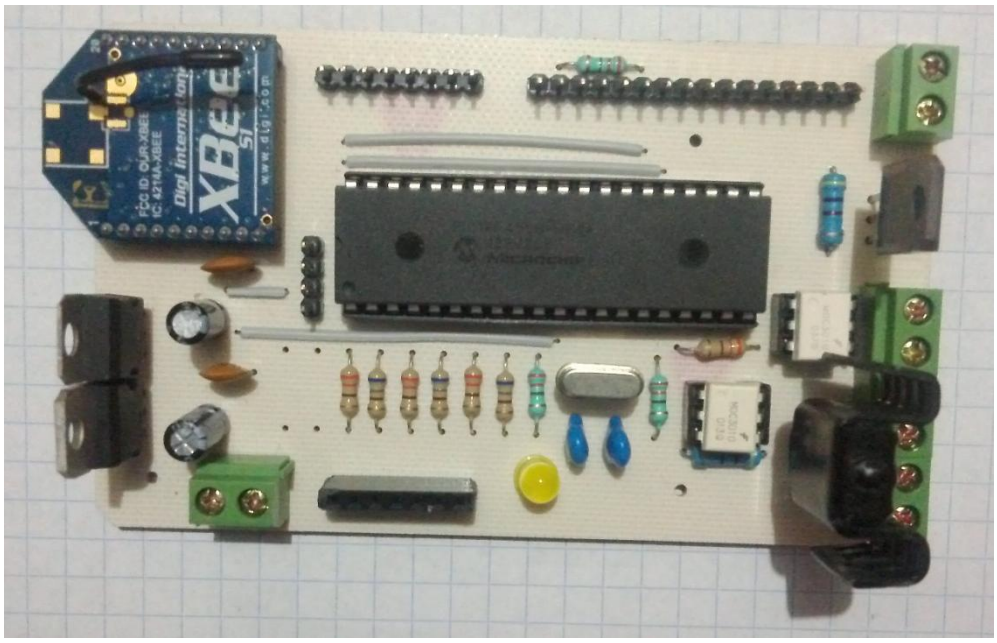

Figura IV.16 Vista superior de la tarjeta de control.

La Figura IV.17 muestra la tarjeta utilizada para alojar el zócalo para la memoria microSD.

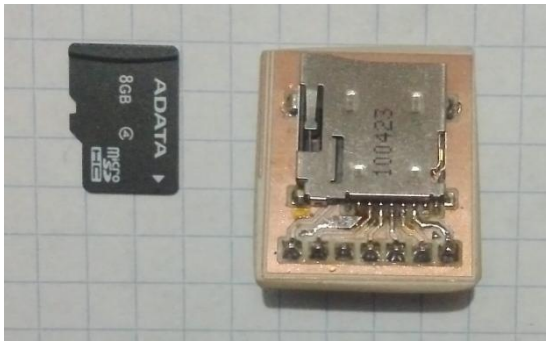

Figura IV.17 Vista superior de la tarjeta para la memoria microSD.

Las Figura IV.18 muestra la LCD de 4x20 caracteres utilizada durante las pruebas, se utilizo cable plano para fabricar los conectores, los cuales serán reemplazados cuando se tenga definida la longitud requerida para su instalación sobre el deshidratador.

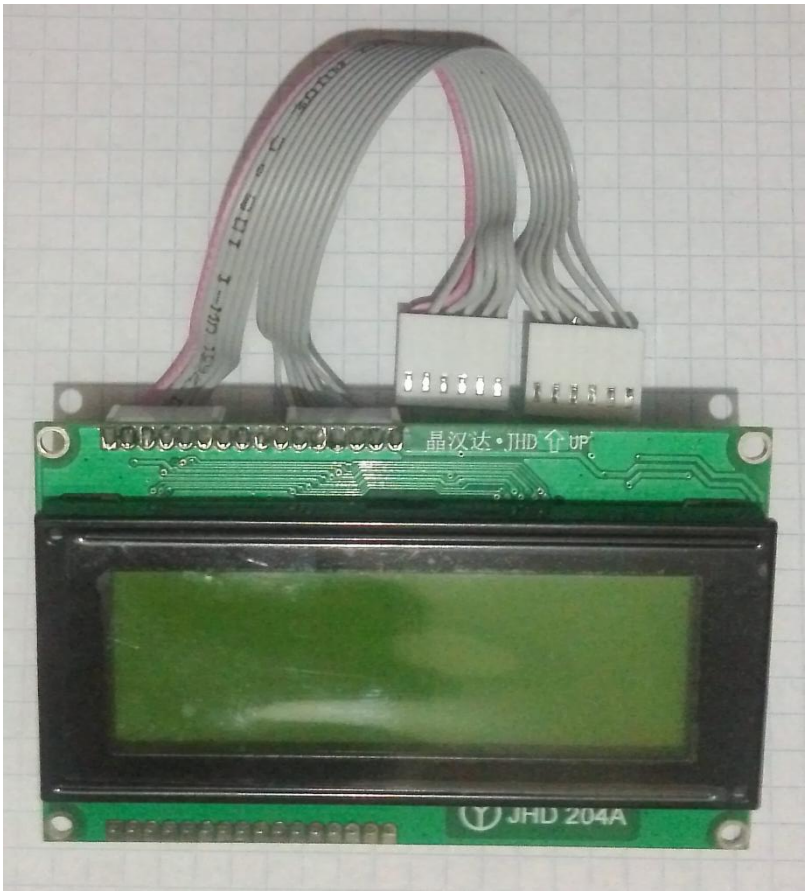

Figura IV.18 LCD de 4x20.

IV.6 **Diseño del software de Interfaz Maquina Usuario**

La interfaz fue desarrollada en Visual Estudio Express 2012, implementando comunicación USB-HID para la recolección de información directamente de la tarjeta de acondicionamiento de señales. El software cuenta con campos para definir el tiempo de muestreo, Set Point y la cantidad de muestras, además permite la visualización de la información en varias graficas y tablas.

IV.6.1 Tarjeta USB serial

La tarjeta USB serial (Figura IV.19) fue desarrollada en la Universidad Autónoma de Querétaro por Omar Alejandro Zamudio Ramírez, dicha tarjeta cuenta con un XBee, PIC y conector USB. No incluye el Firmware por lo que se desarrollo el código del apéndice C que permite la comunicación USB-Serial.

Figura IV.19 Tarjeta USB-Serial a XBee.

IV.6.2 Implementación de la librería

Para implementar la librería AtUsbHid.dll se desarrollo la clase USB.cs para facilitar su uso, dicha clase contiene todos los métodos necesarios para la comunicación USB-HID. El código completo puede ser consultado en el apéndice D.

IV.6.3 Interfaz de usuario

La interfaz de Usuario (Figura IV.20) cuenta con tres campos principales:

- Configuración: Permite definir el directorio de destino de la información e iniciar la comunicación entre la base USB y el software.
- Salida: Permite definir los valores de Set Point, tiempos de muestreo y numero de muestras.
- Área para gráficos y tablas: Facilitan la visualización de información.

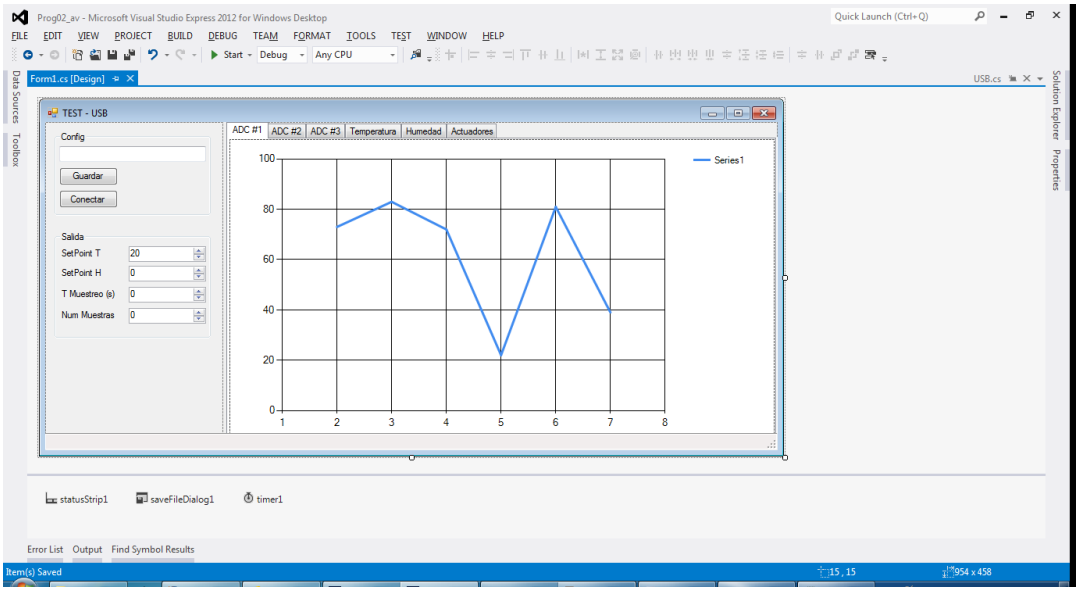

Figura IV.20 Interfaz de usuario

El código de la clase principal puede ser consultado en el apéndice E.

IV.7 **Resultados de la pruebas al sistema completo**

Debido a que no se cuentan con los sensores y el prototipo del deshidratador, no es posible realizar las pruebas en campo. Las pruebas se realizaron para el control y para la adquisición de forma separada.

Para el control se utilizo un banco de tramas que fueron enviadas mediante un trasmisor xbee conectado a xbee conectado a una computadora que ejecutaba una HyperTerminal, al enviar las tramas mostradas en la mostradas en la Tabla IV.3. Los resultados obtenidos fueron introducidos en una hoja de cálculo creada en Microsoft Excel 2007, la

Tabla IV.4 muestra los valores de PWM (señales de control) y los esperados para cada valor en la trama.

Tabla IV.3 Banco de tramas para la prueba del algoritmo de control

Tramas
$,25,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\r\n
$,26,0,0,0,0,0,0,0.1,0,0,0,0,0,0,0,0,0\r\n
$,29,0,0,0,0,0,0,0.5,0,0,0,0,0,0,0,0,0\r\n
$,31,0,0,0,0,0,0,0.9,0,0,0,0,0,0,0,0,0\r\n
$,34,0,0,0,0,0,0,1.2,0,0,0,0,0,0,0,0,0\r\n
$,36,0,0,0,0,0,0,1.6,0,0,0,0,0,0,0,0,0\r\n
$,38,0,0,0,0,0,0,1.9,0,0,0,0,0,0,0,0,0\r\n
$,41,0,0,0,0,0,0,2.3,0,0,0,0,0,0,0,0,0\r\n
$,45,0,0,0,0,0,0,2.5,0,0,0,0,0,0,0,0,0\r\n
$,48,0,0,0,0,0,0,2.8,0,0,0,0,0,0,0,0,0\r\n
$,48,0,0,0,0,0,0,2.7,0,0,0,0,0,0,0,0,0\r\n
$,47,0,0,0,0,0,0,2.4,0,0,0,0,0,0,0,0,0\r\n
$,46,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0\r\n
$,43,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0\r\n
$,42,0,0,0,0,0,0,2.1,0,0,0,0,0,0,0,0,0\r\n
$,40,0,0,0,0,0,0,2.4,0,0,0,0,0,0,0,0,0\r\n
$,38,0,0,0,0,0,0,2.6,0,0,0,0,0,0,0,0,0\r\n
$,37,0,0,0,0,0,0,2.4,0,0,0,0,0,0,0,0,0\r\n
$,36,0,0,0,0,0,0,2.1,0,0,0,0,0,0,0,0,0\r\n
$,37,0,0,0,0,0,0,2.2,0,0,0,0,0,0,0,0,0\r\n
$,38,0,0,0,0,0,0,2.3,0,0,0,0,0,0,0,0,0\r\n

$,40,0,0,0,0,0,0,2.2,0,0,0,0,0,0,0,0,0\r\n

$,42,0,0,0,0,0,0,2.3,0,0,0,0,0,0,0,0,0\r\n

$,43,0,0,0,0,0,0,2.3,0,0,0,0,0,0,0,0,0\r\n

$,42,0,0,0,0,0,0,2.3,0,0,0,0,0,0,0,0,0\r\n

$,40,0,0,0,0,0,0,2.2,0,0,0,0,0,0,0,0,0\r\n

$,39,0,0,0,0,0,0,2.4,0,0,0,0,0,0,0,0,0\r\n

$,39,0,0,0,0,0,0,2.3,0,0,0,0,0,0,0,0,0\r\n

$,40,0,0,0,0,0,0,2.3,0,0,0,0,0,0,0,0,0\r\n

$,40,0,0,0,0,0,0,2.3,0,0,0,0,0,0,0,0,0\r\n

Tabla IV.4 Tabla de resultados, valores de control calculados vs esperados.

# Trama	*Temperatura*	*Flujo*	*PWM #1 (PIC)*	*PWM #1*	*PWM #2 (PIC)*	*PWM #2*
1	25	0	16	16.65	2	2.76
2	26	0.1	16	16.89	2	2.64
3	29	0.5	14	14.97	2	2.39
4	31	0.9	13	13.88	2	2.13
5	34	1.2	11	11.47	1	1.95
6	36	1.6	9	9.88	1	1.61
7	38	1.9	8	8.08	1	1.36
8	41	2.3	4	4.97	0	0.95
9	45	2.5	0	0.46	0	0.75
10	48	2.8	0	0	0	0.39
11	48	2.7	0	0	0	0.49
12	47	2.4	0	0	0	0.8
13	46	2	0	0	1	1.24
14	43	2	0	0	1	1.23
15	42	2.1	0	0.11	1	1.14
16	40	2.4	2	2.12	0	0.81
17	38	2.6	4	4.32	0	0.59
18	37	2.4	5	5.61	0	0.82
19	36	2.1	7	7.01	1	1.15
20	37	2.2	6	6.29	1	1.02
21	38	2.3	5	5.49	0	0.92
22	40	2.2	3	3.48	1	1.05
23	42	2.3	1	1.28	0	0.93

24	43	2.3	0	0	0	0.94
25	42	2.3	0	0.81	0	0.94
26	40	2.2	2	2.82	1	1.06
27	39	2.4	3	3.91	0	0.82
28	39	2.3	4	4	0	0.95
29	40	2.3	2	2.99	0	0.94
30	40	2.3	3	3	0	0.94

En la tabla anterior se aprecia la similitud entre la señal calculada y la señal esperada, la principal diferencia se genera al momento de realizar el redondeo necesario debido a que el valor de PWM es un valor entero entre 0 y 255.

BIBLIOGRAFÍA

Araya-Farias, M y Ratti, C, Dehydration of Foods: General Concepts by Taylor & Francis Group, LLC, 2009.

Barbosa-Canovas, G. and Vega-Marcado, H., Eds., Fundamentals of air–water mixtures and ideal dryers, in Dehydration of Foods, Chapman & Hall, New York, 1996, pp. 9–27.

Boukouvalas, Ch.J., Krokida, M.K., Maroulis, Z.B., and Marinos-Kouris, D., Density and porosity: Literature data compilation for foodstuffs, Int. J. Food Prop., 9, 715-746, 2006.

Karel, M. and Lund, D.B., Dehydration, in Physical Principles of Food Preservation, 2nd ed., Marcel Dekker, New York, 2003a, pp. 378–460.

Krokida, M.K y Maurolis, Z.B., The effect of drying method on viscoelastic behaviour of dehydrated fruits and vegetables, Int. J. Food Sci. Technol., 35, 391-400, 2000.

Krokida, M.K. y Philippopoulos, C., rehydration of dehydrated foods, Dry. Technol., 23, 799-830, 2005.

Maruane, C. y Garreaud, R., Determinación de Humedad en la Atmosfera, DGF – U de chile, 2006

Mujumdar, A.S., Drying fundamentals, in industrial Drying of Foods, Baker, C.G.J., Ed., Blackie Academic and Professional, London, 1997, pp. 7-30.

Mujumdar, A.S., Principles, Classification, and Selection of Dryers by Taylor & Francis Group, LLC, 2006.

Ratti, C., Hot air and freeze-drying of high value foods: A review, *J. Food Eng.*, 49, 311–389, 2001.

Sharma VK, Colangelo A, Spagna G. Experimental investigation of different solar dryers suitable for fruit and vegetable drying. Department Enerzia, Divisione Ingegneria Sperimentale, Italy, Drying 1994;94;879-86.

Vega-Mercado, H., Gongora-Nieto, M.M., and Barbosa-Canovas, G., Advances in dehydration of foods, J. Food Eng., 49, 271–289, 2001.

APÉNDICES

A. Código del micro controlador dedicado al monitoreo

```
#include <18F4550.h>
#fuses HS,NOWDT,NOMCLR
#DEVICE ADC = 10
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6,rcv=PIN_C7)

int1 nuDato=0,bS=1,bLCD=0,bRS=0;
int tiempo=1,in;
int buffer[100];

#include "lcd.c"
#include "sht11_multi.c"
#include <stdlib.h>

#INT_RDA
void serial_isr()
{
  int8 x;
  x = fgetc();
  if(x=='&')in=0;
  buffer[in]=x;
  if(in+1>=100)
    in=0;
  else
    in++;
  buffer[in]=0;
  if(x=='\r')nuDato=1;
}

void interprete();

void main()
{
  float tempProm = 0.0, Temp1,Temp2,Temp3,Temp4,Temp5,Temp6;
  float humProm = 0.0, Hum1,Hum2,Hum3,Hum4,Hum5,Hum6;
  int16 AD1,AD2,AD3;

  lcd_init();
  delay_ms(50);
  sht_init(1);
  sht_init(2);
  sht_init(3);
  sht_init(4);
  sht_init(5);
  sht_init(6);
  setup_adc_ports(AN0_TO_AN2);
  setup_adc(ADC_CLOCK_INTERNAL);

  printf(lcd_putc,"\fTrabajando...\n");
  printf("\nTrabajando...\r\n");
  delay_ms(1000);
```

```
while(true)
{
  output_toggle(PIN_C5);

  if(nuDato)interprete();

  tempProm=0.0,Temp1=0.0,Temp2=0.0,Temp3=0.0,Temp4=0.0,Temp5=0.0,Temp6=0.0;
  humProm=0.0, Hum1=0.0,Hum2=0.0,Hum3=0.0,Hum4=0.0,Hum5=0.0,Hum6=0.0;
  sht_rd(Temp1,Hum1,1);
  if(bLCD)printf(lcd_putc,"\f1 T=%1.2f\n  H=%1.2f",Temp1,Hum1);
  if(bRS)printf("T=%1.2f\n  H=%1.2f ",Temp1,Hum1);
  delay_ms(100);

  sht_rd(Temp2,Hum2,2);
  if(bLCD)printf(lcd_putc,"\f2 T=%1.2f\n  H=%1.2f",Temp2,Hum2);
  if(bRS)printf("T=%1.2f\n  H=%1.2f ",Temp2,Hum2);
  delay_ms(100);

  sht_rd(Temp3,Hum3,3);
  if(bLCD)printf(lcd_putc,"\f3 T=%1.2f\n  H=%1.2f",Temp3,Hum3);
  if(bRS)printf("T=%1.2f\n  H=%1.2f ",Temp3,Hum3);
  delay_ms(100);

  sht_rd(Temp4,Hum4,4);
  if(bLCD)printf(lcd_putc,"\f4 T=%1.2f\n  H=%1.2f",Temp4,Hum4);
  if(bRS)printf("T=%1.2f\n  H=%1.2f ",Temp4,Hum4);
  delay_ms(100);

  sht_rd(Temp5,Hum5,5);
  if(bLCD)printf(lcd_putc,"\f4 T=%1.2f\n  H=%1.2f",Temp5,Hum5);
  if(bRS)printf("T=%1.2f\n  H=%1.2f ",Temp5,Hum5);
  delay_ms(100);

  sht_rd(Temp6,Hum6,6);
  if(bLCD)printf(lcd_putc,"\f4 T=%1.2f\n  H=%1.2f",Temp6,Hum6);
  if(bRS)printf("T=%1.2f\n  H=%1.2f ",Temp6,Hum6);
  delay_ms(100);

  tempProm = Temp1+Temp2+Temp3+Temp4+Temp5+Temp6;
  tempProm = (float)(tempProm/6.0);

  humProm = Hum1+Hum2+Hum3+Hum4+Hum5+Hum6;
  humProm = (float)(humProm/6.0);

  set_adc_channel(0);
  delay_us( 50 );
  AD1 = read_adc();
  if(bLCD)printf(lcd_putc,"\f1 A/D = %Lu", AD1);
  if(bRS)printf("A/D = %Lu ", AD1);

  set_adc_channel(1);
  delay_us( 50 );
  AD2 = read_adc();
  if(bLCD)printf(lcd_putc,"\f2 A/D = %Lu", AD2);
  if(bRS)printf("A/D = %Lu ", AD2);
```

```
    set_adc_channel(2);
    delay_us( 50 );
    AD3 = read_adc();
    if(bLCD)printf(lcd_putc,"\f3 A/D = %Lu", AD3);
    if(bRS)printf("A/D = %Lu\r\n", AD3);

    delay_ms(400);
    delay_ms((tiempo-1)*1000);

    printf("$,");
    printf("%0.2f,",tempProm);
    printf("%0.2f,%0.2f,%0.2f,%0.2f,%0.2f,%0.2f,",Temp1,Temp2,Temp3,Temp4,Temp5,Temp6);
    printf("%Lu,%Lu,%Lu,",AD1,AD2,AD3);
    printf("%0.2f,",humProm);
    printf("%0.2f,%0.2f,%0.2f,%0.2f,%0.2f,%0.2f,",Hum1,Hum2,Hum3,Hum4,Hum5,Hum6);
    printf("\r\n");
  }
}

void interprete()
{
  char numero[8];
  int contCOMA=0,i=0,j=0;

  if(buffer[0]=='&')
  {
    for(i=0;i<strlen(buffer);i++)
    {
      if(buffer[i]=='\r')break;
      if(buffer[i]==',')
      {
        contCOMA++;
        j=0;

        if(contCOMA==2)
          tiempo = atoi(numero);
        if(contCOMA==3)
          bLCD = atoi(numero);
        if(contCOMA==4)
          bRS = atoi(numero);
      }
      else
      {
        numero[j]=buffer[i];
        j++;
        numero[j]=0;
      }
    }
  }
}
```

B. Código del micro controlador dedicado al control y registro.

```
#include <18F4550.H>
#use delay(clock=4M, crystal)
#fuses XT,NOWDT,NOPROTECT,NOMCLR
#use rs232(baud=9600, xmit=PIN_C6,rcv=PIN_C7)

#use fast_io(a)
#define MMCSD_PIN_SCL     PIN_A3 //o
#define MMCSD_PIN_SDI     PIN_A2 //i
#define MMCSD_PIN_SDO     PIN_A1 //o
#define MMCSD_PIN_SELECT  PIN_A0 //o

#include "kbd_lib.c"
#include "lcd.c"
#include "mmcsd.c"
#include <stdlib.h>

int1 nuDato = 0;
int16 in=0,out=0;
int32 address = 0;
char buffer[500];
char datos[125];  //116+1
float sp_temp, sp_vel;
float error_temp_T0=0,integral_temp_T0=0,error_vel_T0=0,integral_vel_T0=0;
float a_t=0,b_t=0,c_t=0,a_f=0,b_f=0,c_f=0;
unsigned int pwm1=0,pwm2=0;

#INT_RDA
void serial_isr()
{
  int8 x;
  x = fgetc();
  buffer[in]=x;
  in++;
  if(in==500)in=0;
  if(x=='\r')nuDato=1;
}

void interprete();
void fromSD();
void configTEMP(float tSP,float fa,float fb,float fc);
void configVEL(float tSP,float fa,float fb,float fc);
void getConfig();
void setConfig();
int PID(float rT, float yT, float eT0, float iT0, float a, float b, float c, unsigned int max, unsigned int min, int1 t);

void main()
{
  int t=0,SD_res=1;
  int idx=0;
```

```
char c;

SETUP_TIMER_2(T2_DIV_BY_16,255,1);
setup_ccp1(CCP_PWM);
setup_ccp2(CCP_PWM);
set_pwm1_duty(0);
set_pwm2_duty(0);

  #asm
  nop
  #endasm

for(t=0;t<5;t++)
{
  output_toggle(PIN_E0);
  delay_ms(250);
}

printf("COM OK...");

set_tris_a(0x00000100);
port_b_pullups(TRUE);
lcd_init();

printf(lcd_putc,"\fLCD OK...\n");

kbd_init();

while(address<20)
{
  SD_res = mmcsd_init();
  if(SD_res==0)
    address++;

  printf(lcd_putc,"\fIniciando SD\n>>> %lu%c (%i)",address*5,37,SD_res);

  for(t=0;t<25;t++)
  {
    spi_xfer(mmcsd_spi, 0xFF);
  }
}
address=0;
printf(lcd_putc,"\fSD OK...\n");
printf("SD OK...\n\r");

enable_interrupts(INT_RDA);
enable_interrupts(GLOBAL);

//leer eeprom
getConfig();

while(true)
{
  printf(lcd_putc,"\fSPT %0.2f p %u\nSPF %0.2f p %u",sp_temp,pwm1,sp_vel,pwm2);
  //printf("SPT %0.2f p %u\nSPF %0.2f p %u",sp_temp,pwm2,sp_vel,pwm1);
  output_toggle(PIN_E0);
```

```
    c=kbd_getc();
    if(c!=0)
    {
      if(c=='A')
        configTEMP(SP_temp,a_t,b_t,c_t);
      if(c=='B')
        configVel(SP_vel,a_f,b_f,c_f);
      if(c=='C')
        fromSD();
      //if(c=='D')

      if(c=='A'||c=='B')
        setConfig();
      c=0;
    }

    if(nuDato==1)
    {
      while(out!=in)
      {
        if(buffer[out]=='$')
          idx=0;
        datos[idx]=buffer[out];
        idx++;
        datos[idx]=0;
        if(buffer[out]=='\r')
          interprete();
        out++;
        if(out==500)out=0;
      }
      nuDato=0;
    }

    delay_ms(150);
  }
}

void interprete()
{
  /*
    vector datos
    $,TT,T1,T2,T3,T4,T5,T6,VV,...
    TT:Temp promedio
    VV:Vel fan
  */

  //printf(">>> %s",datos);

  char numero[8];
  int contCOMA=0,i=0,j=0,n;
  float temp, vel;


  if(datos[0]=='$')
  {
```

```
  //guarda los datos
  while(i<10)
  {
    n = strlen(datos);
    j=0;
    mmcsd_write_data(address,n,datos);

    j=0;
    while(j<10)
    {
      mmcsd_flush_buffer();
      j++;
    }
    i++;
  }
  //address += 512;
  address += n;

  for(i=0;i<250;i++)
  {
    if(datos[i]=='\r')break;
    if(datos[i]==',')
    {
      contCOMA++;
      j=0;

      if(contCOMA==2)
        temp = atof(numero);
      if(contCOMA==9)
        vel = atof(numero);
    }
    else
    {
      numero[j]=datos[i];
      j++;
      numero[j]=0;
    }
  }

  pwm1=PID(sp_temp,temp,error_temp_T0,integral_temp_T0,1,0.1,0.01,254,0,1);
  pwm2=PID(sp_vel,vel,error_vel_T0,integral_vel_T0,1,0.1,0.01,254,0,0);

  //PWM
  set_pwm1_duty(pwm1);
  set_pwm2_duty(pwm2);
  #asm
  nop
  #endasm
}
if(datos[0]=='#')
{
  for(i=0;i<250;i++)
  {
    if(datos[i]=='\r')break;
    if(datos[i]==',')
    {
```

```
          contCOMA++;
          j=0;

          if(contCOMA==2)
            sp_temp = atof(numero);
          if(contCOMA==3)
            a_t = atof(numero);
          if(contCOMA==4)
            b_t = atof(numero);
          if(contCOMA==5)
            c_t = atof(numero);
          if(contCOMA==6)
            sp_vel = atof(numero);
          if(contCOMA==7)
            a_f = atof(numero);
          if(contCOMA==8)
            b_f = atof(numero);
          if(contCOMA==9)
            c_f = atof(numero);
        }
        else
        {
          numero[j]=datos[i];
          j++;
          numero[j]=0;
        }
      }
      setConfig();
    }
}

unsigned int PID(float rT, float yT, float eT0, float iT0, float a, float b, float c, unsigned int max, unsigned int min, int1 t)
{
  float eT = 0, iT = 0, dT = 0;
  signed int16 uT=0;
  eT = rT - yT;          //error
  iT = b * eT + iT0;     //integral
  dT = c * (eT - eT0);   //derivativo
  uT = a * eT + iT + dT;
  if(uT>max)
    uT=max;
  if(uT<min)
    uT=min;

  if(t)
  {
    error_temp_T0=eT;
    integral_temp_T0=iT;

    printf("=========================================\r\n");
    printf("*****        TEMPERATURA        *****\r\n");
    printf("Error: rT - yT = %0.2f = %0.2f - %0.2f\r\n",eT,rT,yT);
    printf("Integral: b * eT + iT0 = %0.2f = %0.2f * %0.2f + %0.2f\r\n",iT,b,eT,iT0);
    printf("Derivativo: c * (eT - eT0) = %0.2f = %0.2f * (%0.2f - %0.2f)\r\n",dT,c,eT,eT0);
    printf("Control: a * eT + iT + dT = %ld = %0.2f * %0.2f + %0.2f+ %0.2f\r\n",uT,a,eT,iT,dT);
    printf("=========================================\r\n\r\n");
```

```
  }
  else
  {
    error_vel_T0=eT;
    integral_vel_T0=iT;

    printf("=========================================\r\n");
    printf("*****          Viento          *****\r\n");
    printf("Error: rT - yT = %0.2f = %0.2f - %0.2f\r\n",eT,rT,yT);
    printf("Integral: b * eT + iT0 = %0.2f = %0.2f * %0.2f + %0.2f\r\n",iT,b,eT,iT0);
    printf("Derivativo: c * (eT - eT0) = %0.2f = %0.2f * (%0.2f - %0.2f)\r\n",dT,c,eT,eT0);
    printf("Control: a * eT + iT + dT = %ld = %0.2f * %0.2f + %0.2f+ %0.2f\r\n",uT,a,eT,iT,dT);
    printf("=========================================\r\n\r\n");

  }

  return uT;
}

//lector por sectores
void fromSD()
{
  int32 jL = 0;
  int res = 1,i=0;
  char p[512];
  for(jL=0;jL<=address;jL+=512)
  {
    printf(lcd_putc,"\f %lu / %lu",jL,address);
    printf("%lu :: ",jL);
    res = mmcsd_read_block(jL, 512, &p);
    i=0;
    while(i<10)
    {
      res = mmcsd_read_block(jL, 512, &p);
      if(res==0)
      {
        printf("%s",p);
        break;
      }
      i++;
    }
    printf("\n\r");
  }
}

void configTEMP(float tSP,float fa,float fb,float fc)
{
  char numero[8];
  int linea=0;
  numero = "0";
  int i=0;
  char c=0;

  while(c!='#'&&c!='D')
  {
```

```
    c=kbd_getc();

    switch(c)
    {
      case '#': //aceptar
        sp_temp = tSP;
        a_t=fa;
        b_t=fb;
        c_t=fc;
        break;
      case 'A'://NA
        break;
      case 'B':
        numero="0.0";
        i=0;
        Linea--;
        break;
      case 'C':
        numero="0.0";
        i=0;
        Linea++;
        break;
      case 'D'://cancelar
        break;
      case '*'://punto
        numero[i]='.';
        i++;
        numero[i]=0;
        break;
      default:
        if(c!=0)
        {
          numero[i]=c;
          i++;
          numero[i]=0;
          if(Linea==0)
            tSP = atof(numero);
          if(Linea==1)
            fa = atof(numero);
          if(Linea==2)
            fb = atof(numero);
          if(Linea==3)
            fc = atof(numero);
        }
        break;
    }
    lcd_gotoxy(1,1);printf(lcd_putc,"\fSP: %0.3f",tSP);if(Linea==0)printf(lcd_putc," <");
    lcd_gotoxy(1,2);printf(lcd_putc,"Ke: %0.3f",fa);;if(Linea==1)printf(lcd_putc," <");
    lcd_gotoxy(1,3);printf(lcd_putc,"Kd: %0.3f",fb);;if(Linea==2)printf(lcd_putc," <");
    lcd_gotoxy(1,4);printf(lcd_putc,"Ki: %0.3f",fc);;if(Linea==3)printf(lcd_putc," <");
    delay_ms(100);
  }
}

void configVel(float tSP,float fa,float fb,float fc)
{
```

```
char numero[8];
int linea=0;
numero = "0";
int i=0;
char c=0;

while(c!='#'&&c!='D')
{
  c=kbd_getc();

  switch(c)
  {
    case '#': //aceptar
      sp_vel = tSP;
      a_f=fa;
      b_f=fb;
      c_f=fc;
      break;
    case 'A'://NA
      break;
    case 'B':
      numero="0.0";
      i=0;
      Linea--;
      break;
    case 'C':
      numero="0.0";
      i=0;
      Linea++;
      break;
    case 'D'://cancelar
      break;
    case '*'://punto
      numero[i]='.';
      i++;
      numero[i]=0;
      break;
    default:
      if(c!=0)
      {
        numero[i]=c;
        i++;
        numero[i]=0;
        if(Linea==0)
          tSP = atof(numero);
        if(Linea==1)
          fa = atof(numero);
        if(Linea==2)
          fb = atof(numero);
        if(Linea==3)
          fc = atof(numero);
      }
      break;
  }
  lcd_gotoxy(1,1);printf(lcd_putc,"\fSP: %0.3f",tSP);if(Linea==0)printf(lcd_putc," <");
  lcd_gotoxy(1,2);printf(lcd_putc,"Ke: %0.3f",fa);;if(Linea==1)printf(lcd_putc," <");
```

```
    lcd_gotoxy(1,3);printf(lcd_putc,"Kd: %0.3f",fb);;if(Linea==2)printf(lcd_putc," <");
    lcd_gotoxy(1,4);printf(lcd_putc,"Ki: %0.3f",fc);;if(Linea==3)printf(lcd_putc," <");
    delay_ms(100);
  }
}

void getConfig()
{
  int a,b;

  /* TEMPERATURA */
  //SETPOINT
  a = read_eeprom(0);
  delay_ms(500);
  b = read_eeprom(1);
  sp_temp = a + b * 0.01;

  //Ke
  a = read_eeprom(2);
  delay_ms(500);
  b = read_eeprom(3);
  a_t = a + b * 0.01;

  //Kd
  a = read_eeprom(4);
  delay_ms(500);
  b = read_eeprom(5);
  b_t = a + b * 0.01;

  //Ki
  a = read_eeprom(6);
  delay_ms(500);
  b = read_eeprom(7);
  c_t = a + b * 0.01;

  /* FLUJO */
  //SETPOINT
  delay_ms(500);
  a = read_eeprom(8);
  delay_ms(500);
  b = read_eeprom(9);
  sp_vel = a + b * 0.01;

  //Ke
  delay_ms(500);
  a = read_eeprom(10);
  delay_ms(500);
  b = read_eeprom(11);
  a_f = a + b * 0.01;

  //Kd
  delay_ms(500);
  a = read_eeprom(12);
  delay_ms(500);
  b = read_eeprom(13);
  b_f = a + b * 0.01;
```

```
  //Ki
  delay_ms(500);
  a = read_eeprom(14);
  delay_ms(500);
  b = read_eeprom(15);
  c_f = a + b * 0.01;
}

void setConfig()
{
  int a,b;

  /* TEMPERTURA */
  //SETPOINT
  a = (int)sp_temp;
  b = (sp_temp - a) * 100;
  write_eeprom(0,a);
  delay_ms(500);
  write_eeprom(1,b);

  //Ke
  a = (int)a_t;
  b = (a_t - a) * 100;
  write_eeprom(2,a);
  delay_ms(500);
  write_eeprom(3,b);

  //Kd
  a = (int)b_t;
  b = (b_t - a) * 100;
  write_eeprom(4,a);
  delay_ms(500);
  write_eeprom(5,b);

  //Ki
  a = (int)c_t;
  b = (c_t - a) * 100;
  write_eeprom(6,a);
  delay_ms(500);
  write_eeprom(7,b);

  /* Flujo */
  //SETPOINT
  a = (int)sp_vel;
  b = (sp_vel - a) * 100;
  write_eeprom(8,a);
  delay_ms(500);
  write_eeprom(9,b);

  //Ke
  a = (int)a_f;
  b = (a_f - a) * 100;
  write_eeprom(10,a);
  delay_ms(500);
  write_eeprom(11,b);
```

```
  //Kd
  a = (int)b_f;
  b = (b_f - a) * 100;
  write_eeprom(12,a);
  delay_ms(500);
  write_eeprom(13,b);

  //Ki
  a = (int)c_f;
  b = (c_f - a) * 100;
  write_eeprom(14,a);
  delay_ms(500);
  write_eeprom(15,b);
}
```

C. Código del micro controlador dedicado a la comunicación USB-serial.

```
#include <18F2455.h>
#fuses NOMCLR,HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN,NOPBADEN
#use delay(clock=48000000)
#use rs232(baud=19200, xmit=PIN_C6, rcv=PIN_C7, bits=8, parity=N)
//#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, bits=8, parity=N)

#define USB_CONFIG_HID_TX_SIZE   64
#define USB_CONFIG_HID_RX_SIZE   64

#define USB_CONFIG_VID 0x0462

#include <pic18_usb.h>
#include <usb_desc_hid.h>
#include <usb.c>

#INCLUDE <STDLIB.H>

char datoIN[USB_CONFIG_HID_RX_SIZE];

char dato[USB_CONFIG_HID_TX_SIZE];
char pila[800];
int16 in=0;
int16 out=0;

#INT_RDA
void serial_rda()
{
   output_toggle(PIN_B5);
   pila[in]=fgetc();
   in++;
   if(in==800)
      in=0;
}

void main()
{
   int b=0;
   int i=0,k=1;
   usb_init();
   output_low(PIN_B5);

   enable_interrupts(INT_RDA);
   enable_interrupts(GLOBAL);

   while(true)
   {
      usb_task();
      if (usb_enumerated()&&usb_attached())
      {
         if (usb_kbhit(1))
         {
            usb_get_packet(1,datoIN,USB_CONFIG_HID_RX_SIZE);
            printf("%s",datoIN);
            for(i=0;i<USB_CONFIG_HID_RX_SIZE-1;i++){datoIN[i]=0;}
         }
```

```
         /*if(kbhit())
         {
            pila[in]=fgetc();
            in++;
            if(in==800)
               in=0;
         }*/


         if(b==1 && !kbhit())
         {
             usb_put_packet(1,dato,USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGGLE);
             k=1;
             b=0;
         }

         if(!kbhit())
         {
            while((in!=out || k!=1) && !kbhit())
            {
               if(in!=out)
               {
                  dato[0]=0;
                  dato[k]=pila[out];
                  if(k+1<USB_CONFIG_HID_TX_SIZE) dato[k+1]=0;
                  k++;
                  out++;
                  if(out==800) out=0;

                  if(k==USB_CONFIG_HID_TX_SIZE ||  dato[k-1]=='\n' ||  dato[k-1]=='\r')
                  {
                     if(!kbhit())
                     {
                        usb_put_packet(1,dato,USB_CONFIG_HID_TX_SIZE,USB_DTS_TOGGLE);
                        k=1;
                     }
                     else
                        b=1;
                  }
               }
            }
         }

      }
   }
}
```

D. Clase: USB.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.InteropServices;

namespace Prog02_av
{
    class USB
    {
        [DllImport("AtUsbHid.dll", EntryPoint = "findHidDevice")]
        static extern bool findHidDevice(uint VendorID, uint ProductID);
        public bool FindHidDevice(uint VendorID, uint ProductID)
        {
            return findHidDevice(VendorID, ProductID);
        }

        [DllImport("AtUsbHid.dll", EntryPoint = "closeDevice")]
        static extern void closeDevice();
        public void CloseDevice()
        {
            closeDevice();
        }

        [DllImport("AtUsbHid.dll", EntryPoint = "writeData")]
        static extern bool writeData(byte[] buffer);
        public bool WriteData(byte[] buffer)
        {
            return writeData(buffer);
        }

        [DllImport("AtUsbHid.dll", EntryPoint = "readData")]
        static extern bool readData(byte[] buffer);
        public bool ReadData(byte[] buffer)
        {
            return readData(buffer);
        }
        [DllImport("AtUsbHid.dll", EntryPoint = "setFeature")]
        static extern bool setFeature(byte[] buffer);
        public bool SetFeature(byte[] buffer)
        {
            return setFeature(buffer);
        }
        [DllImport("AtUsbHid.dll", EntryPoint = "getFeatureReportLength")]
        static extern int getFeatureReportLength();
        public int GetFeatureReportLength()
        {
            return getFeatureReportLength();
        }
        [DllImport("AtUsbHid.dll", EntryPoint = "getOutputReportLength")]
        static extern int getOutputReportLength();
        public int GetOutputReportLength()
        {
            return getOutputReportLength();
        }
        [DllImport("AtUsbHid.dll", EntryPoint = "getInputReportLength")]
        static extern int getInputReportLength();
```

```
        public int GetInputReportLength()
        {
            return getInputReportLength();
        }
    }
}
```

E. Clase principal

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

namespace Prog02_av
{
   public partial class Form1 : Form
   {
      USB comUSB;
      int longitudEntrada;
      int longitudSalida;
      Thread tTX;
      bool bCerrar = false;
      Queue<string> cola = new Queue<string>();

      public Form1()
      {
         InitializeComponent();
      }

      private void dataGridView1_Click(object sender, EventArgs e)
      {
         chart1.Visible = true;
      }

      private void chart1_Click(object sender, EventArgs e)
      {
         chart1.Visible = false;
      }

      private void dataGridView2_Click(object sender, EventArgs e)
      {
         chart2.Visible = true;
      }

      private void chart2_Click(object sender, EventArgs e)
      {
         chart2.Visible = false;
      }

      private void dataGridView3_Click(object sender, EventArgs e)
      {
         chart3.Visible = true;
```

```
}

private void chart3_Click(object sender, EventArgs e)
{
   chart3.Visible = false;
}

private void button1_Click(object sender, EventArgs e)
{
   DialogResult res = saveFileDialog1.ShowDialog();
   if (res.ToString() == "OK")
      textBox1.Text = saveFileDialog1.FileName;
}

private void button2_Click(object sender, EventArgs e)
{
   if (button2.Text.Equals("Conectar"))
   {
      button2.Text = "Desconectar";
      conectar();
      tTX = new Thread(new ThreadStart(TX));
      tTX.Start();

      groupBox2.Enabled = true;
   }
   else
   {
      button2.Text = "Conectar";
      groupBox2.Enabled = false;
   }
}

private void button3_Click(object sender, EventArgs e)
{
   string orden;

   //PC to AData
   orden = "&,";
   orden += numericUpDown3.Value + ",";
   if (checkBox1.CheckState == CheckState.Checked)
      orden += "1" + ",";
   else
      orden += "0" + ",";
   if (checkBox2.CheckState == CheckState.Checked)
      orden += "1" + ",";
   else
      orden += "0" + ",";
   orden += "\n";

   cola.Enqueue(orden);

   //PC to Control
   orden = "#,";
   orden += numericUpDown1.Value + ",";
   orden += textBox2.Text + ",";
   orden += textBox3.Text + ",";
```

```
    orden += textBox4.Text + ",";
    orden += numericUpDown2.Value + ",";
    orden += textBox5.Text + ",";
    orden += textBox6.Text + ",";
    orden += textBox7.Text + ",";
    orden += "\n";

    cola.Enqueue(orden);
}

private void TX()
{
    while (!bCerrar)
    {
        if (cola.Count > 0 && longitudSalida != 0)
        {
            string orden = cola.Dequeue();
            byte[] buffer_out = new byte[longitudSalida];
            int k = 0;
            for (int i = 0; i < orden.Length; i++)
            {
                buffer_out[k] = Convert.ToByte(orden[i]);
                if (k == longitudSalida - 1)
                {
                    comUSB.WriteData(buffer_out);
                    for (int j = 0; j < longitudSalida; j++)buffer_out[j] = 0;
                    k = 0;
                }
                else
                    k++;
            }
            if (k != 0)
            {
                comUSB.WriteData(buffer_out);
            }

            Thread.Sleep(100);
        }
        else
            Thread.Sleep(1000);
    }
}

private void interprete(string trama)
{
    //trama: "$,40,30,1,0,\n"
    string[] datos = trama.Split(',');

    chart1.Series[0].Points.AddY(datos[1]);
    chart2.Series[0].Points.AddY(datos[2]);
    chart3.Series[0].Points.AddY(datos[3]);
    chart3.Series[1].Points.AddY(datos[4]);

    string fecha_hora = DateTime.Now.ToShortDateString() +
                        " " +
                        DateTime.Now.ToShortTimeString();
```

```
        object[] t1 = { fecha_hora , datos[1] };
        object[] t2 = { fecha_hora, datos[2] };
        object[] t3 = { fecha_hora, datos[3], datos[4] };

        dataGridView1.Rows.Add(t1);
        dataGridView2.Rows.Add(t2);
        dataGridView3.Rows.Add(t3);
    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        byte[] buffer_in;
        buffer_in = new byte[longitudEntrada];
        try
        {
            if (comUSB.ReadData(buffer_in))
            {
                for (int i = 1; i < buffer_in.Length; i++)
                {
                    if (buffer_in[i] != 0 && buffer_in[i] != '\r' && buffer_in[i] != '\n')
                    {
                        if ((char)buffer_in[i] == '$')
                            statusStrip1.Text = "" + (char)buffer_in[i];
                        else
                            statusStrip1.Text += (char)buffer_in[i];
                    }
                    else
                    {
                        interprete(statusStrip1.Text);
                        break;
                    }
                }
            }
        }
        catch { }
    }

    private void conectar()
    {
        comUSB = new USB();
        bool result = comUSB.FindHidDevice(0x0462, 0x0020);
        if (result)
        {

            longitudEntrada = comUSB.GetInputReportLength();
            longitudSalida = comUSB.GetOutputReportLength();

            timer1.Enabled = true;
        }
    }
  }
}
```