Universidad Autónoma de Baja California
Campus Ensenada
Facultad de Ingeniería

ESTACIONES METEOROLÓGICAS Y DE MONITOREO DE TEMPERATURAS PARA LOS TELESCOPIOS DEL OBSERVATORIO DE SAN PEDRO MÁRTIR.

Reporte de Servicio Social Profesional
que como opción de titulación presenta:

Juan Francisco Martínez Osuna.

Ensenada, B. C., a Mayo de 2005

Índice

1. Introducción.

El Observatorio Astronómico Nacional (OAN) en San Pedro Mártir (SPM) Baja California ha demostrado ser uno de los mejores sitios astronómicos en nuestro planeta, ésto debido a su excelente seeing y a su gran cantidad de noches despejadas (Cruz González et at. 2005). La Universidad Nacional Autónoma de México (UNAM) inició la construccion de este observatorio en 1967 y para 1971 ya tenia instalados los telescopios de 0.84 y 1.5m de diámetro. El OAN-SPM además cuenta actualmente con un telescopio de 2.1m el cual, por su variada instrumentación, está al nivel de los mejores telescopios de su clase en el mundo (López y Gutiérrez 2003). En el presente SPM está siendo evaluado como posible sitio para telescopios de mucho mayor tamaño como es el caso del telescopio gigante óptico/infrarrojo de 30 metros (http://www.tmt.org/) que, utilizando óptica adaptativa, ayudará a entender procesos como la formación de las galaxias distantes y otros fenómenos del universo que hasta ahora no han podido ser atacados debido a las limitaciones de los telescopios actuales. Otro proyecto considerado para SPM es el Large Synoptic Survey Telescope (http://www.lsst.org/) ideado para localizar supernovas, estallidos gama y asteroides próximos a la tierra, así como para estudiar la energía oscura misteriosa que acelera la expansión del universo. El LSST tendrá un espejo primario de 8.4m de diámetro y proveerá imágenes digitales de objetos astronómicos muy débiles barriendo todo el cielo visible cada tres noches con lo cual se generará una especie de película que pondrá en evidencia a los objetos que cambian de brillo o se mueven muy rápidamente.

La importancia de hacer estudios del clima en SPM reside principalmente en evaluar el retorno que se obtendrá de las grandes inversiones que se haran en el sitio. Para esta evaluación es indispensable conocer los parámetros más importantes del lugar como son la cantidad de noches despejadas, el seeing, el clima y los cambios físicos que ocurren en su atmósfera.

1.1. Estación meteorológica.

Una estación meteorológica es un equipo dedicado a medir y registrar regularmente diversas variables meteorológicas. Las bases de datos generados por estos equipos se utilizan tanto para la elaboración de predicciones meteorológicas a partir de modelos numéricos así como para estudios climáticos.

En todos los observatorios astronómicos se requiere de estaciones meteorológicas funcionando de la manera más automática y continua posible. Aparte de los parámetros meteorológicos es de particular importancia la determinación de la diferencia de temperatura entre el espejo primario y el ambiente así como entre el exterior e interior del edificio los cuales afectan de manera directa la calidad de las imágenes obtenidas con los telescopios y por lo tanto la eficiencia de los mismos (Michel et al. 2001).

Algunos ejemplos de los parámetros físicos que se pueden medir en un sitio son la temperatura, la humedad relativa, la presión atmosférica, la precipitación pluvial, la velocidad del viento y la insolación. Nuestras estaciones son capaces de medir todos ellos.

Ya que los altos niveles de humedad ponen en riesgo los equipos ópticos y electrónicos, es conveniente llevar un monitoreo continuo para también poder generar señales de alarma y así evitar posibles deterioros en la superficie de los espejos y cortos en el equipo electrónico.

2. Antecedentes.

Por muchos años la información meteorológica de SPM fue recabada, de manera parcial, por los asistentes nocturnos quienes leían (al principio y final de cada noche) sensores analógicos de temperatura, presión atmosférica y humedad relativa instalados en el piso de telescopio del 2.1m. Debido a la necesidad de tener un registro más fidedigno y representativo del clima en SPM, a finales de 1998 se instalaron dos estaciones meteorológicas rudimentarias en los telescopios de 0.84 y 1.5m. Estas estaciones consistían en hasta ocho sensores conectados a una PC por medio de una tarjeta de adquisición de datos comercial. Estas estaciones permitieron recabar información meteorológica en SPM por más de 4 años (Michel, Hiriart y Chapela 2003).

En 2003 se instaló una nueva estación meteorológica en el telescopio de 0.84m (Chapela 2004), la cuál consistía en una PC conectada a varios módulos basados en el microcontrolador AT89C52

de Atmel y un convertidor analógico/digital (ADC). Ésto permitió seguir una filosofía de diseño modular que proporcionaba protección a la PC así como a los otros módulos ya que en caso de que, por ejemplo, un módulo fuera destruido por una descarga eléctrica solo se tenía que reemplazar el modulo dañado sin que esto afectara la operación de los demás módulos. Éste diseño también permitió reducir al mínimo el cableado y, lo que es más importante, aumentar el número de sensores empleados. La implementación de esta estación implicó también una mayor facilidad en su mantenimiento. En la Figura 1 se muestra un diagrama a bloques de esta estación.

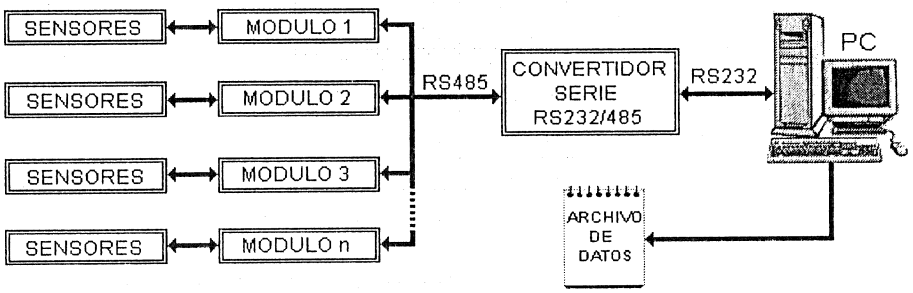

Figura 1. Diagrama a bloques de la estación meteorológica del 0.84m.

3. Objetivo.

El objetivo del presente trabajo fue el implementar nuevas estaciones meteorológicas, para todos los telescopios del OAN-SPM, que permitieran medir parámetros como velocidad del viento, temperatura, humedad relativa, presión barométrica, insolación y precipitación pluvial además de permitir el monitoreo de las temperaturas en los diferentes niveles del edificio y en la óptica de los telescopios para a su vez proponer maneras de mejorar el seeing de cúpula y eventualmente implementar los sistemas de alarma para cuando el equipo esté en riesgo debido a condiciones meteorológicas adversas.

4. Desarrollo.

4.1. Microcontroladores.

Un microcontrolador es un circuito integrado de alta escala de integración que cuenta por lo general con los siguientes elementos:

- Una unidad Central de Procesamiento (CPU).
- Memoria RAM para contener los datos.
- Memoria para el programa tipo ROM/ EPROM/ EEPROM.
- Líneas de E/S para comunicación.
- Control de periféricos (generador de reloj, temporizadores, puertos serie y paralelo).

Los productos que incorporan microcontroladores tienen las siguientes ventajas:

- El ser programables permite su optimización en periodos de tiempo relativamente cortos.
- Aumento de sus prestaciones ya que el mayor control sobre un determinado elemento representa una mejora considerable en el mismo.
- Aumento de la fiabilidad al reemplazar un elevado número de elementos por un microcontrolador disminuyendo los riesgo de averías y reduciendo el número de ajustes necesarios.
- Reducción del tamaño del producto acabado ya que la integración del microcontrolador en un chip disminuye el volumen, la mano de obra y los stocks.
- Mayor versatilidad porque las características de control están programadas y su modificación solo requiere de cambios en el programa de instrucciones.

4.1.1. ADuC832.

El ADuC832 de Analog Devices (http://www.analog.com) es un microcontrolador de 8 bits que tiene integrado, entre otras cosas, un convertidor analógico/digital (ADC) de 8 canales y con 12 bits de resolución y 2 convertidores digital/analógico (DAC) también con 12 bits de resolución; todo esto en un solo circuito integrado de 52 patas. Un diagrama a bloques de este dispositivo puede verse en la Figura 2.

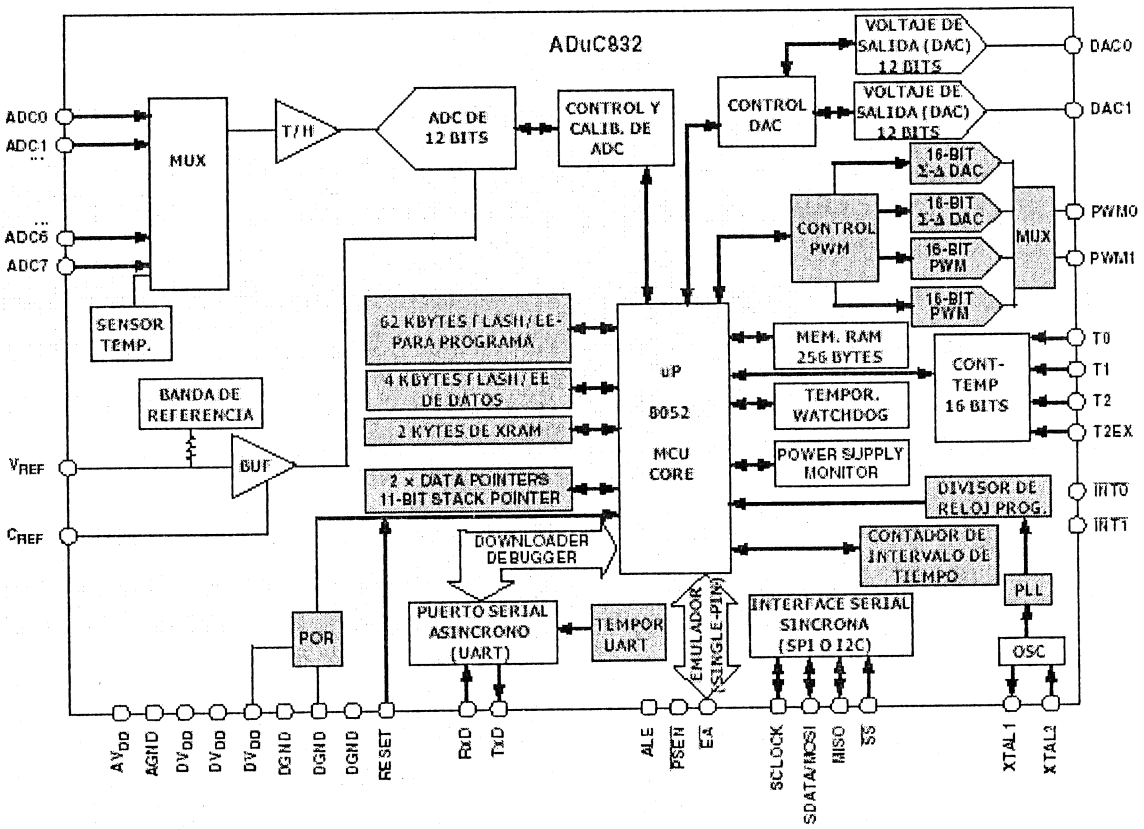

Figura 2. Diagrama a bloques del microcontrolador ADuC832.

El ADuC832 cuenta con 62kB de memoria de programa tipo Flash/EE no-volátil, para correr el código del usuario, esta memoria tiene las ventajas de una EEPROM de reprogramación flexible combinadas con la densidad y eficiencia de espacio de las EPROM. Cuenta además con 4kB de memoria de datos también tipo Flash/EE la cual es accesible al usuario por medio de un grupo de registros de control. Cuenta también con 256B de RAM de propósito general y 2kB de RAM extendida interna.

Este dispositivo requiere para su operación de un cristal de 32 kHz como entrada a un lazo de fijación de fase (PLL) que genera una frecuencia de reloj de hasta 16.77 MHz. El núcleo del microcontrolador es de la familia del 8052 y por lo tanto utiliza el juego de instrucciones del mismo. Cuenta con cuatro puertos bidireccionales de 8 bits cada uno, tres contadores-temporizadores de 16 bits, y puertos especialmente destinados a las comunicaciones que pueden ser de los tipos: ducto de Circuitos Inter-Integrados (I^2C), Interfaz Serial de Periféricos (SPI) y transmisor/receptor asíncrono universal (UART).

El bloque de conversión A/D incorpora un convertidor rápido de 8 canales,consistente en un multiplexor, un sistema de muestreo y retención y un ADC con 12 bits de resolución. Este ADC consiste en un convertidor convencional de aproximaciones sucesivas que acepta un rango de voltaje de entrada de 0V a V_{REF}, donde V_{REF} puede ser interno con una referencia calibrada de fabrica de 2.5V ò externo con una referencia que puede estar entre 1V y el voltaje de alimentación.

El ADuC832 tiene coeficientes de calibración programados de fábrica que son automáticamente cargados al ADC al alimentarlo, asegurando su óptimo rendimiento, asimismo, el núcleo del ADC contiene registros de compensación (offset) interna así como de calibración de ganancia la cual que puede ser ajustada por medio de hardware para minimizar errores del sistema.

La utilización de este microcontrolador nos facilita la construcción de nuestro módulo, pues al tener integrado un ADC y un DAC se ahorraron esfuerzos en el diseño y las pruebas de laboratorio.

4.2. Comunicación serie.

Por medio del puerto serie (UART) es posible cargar el programa al microcontrolador y llevar a cabo la comunicación entre los módulos de adquisición y la PC apegandose a los estándares RS-232/485.

Una de las ventajas del puerto serie es que requiere de únicamente un solo cable para transmitir los 8 bits lo ciual reduce de manera apreciable la cantidad de cables utilizados. También permite que los cables que se emplean para la comunicación puedan ser mucho más largos. Estas ventajas minimizan los costos de cableado, el espacio de cableado, el número de controladores de ducto y, principalmente, permite la transmisión de los datos a distancias mayores.

Los puertos serie, son bi-direccionales, lo que permite a cada dispositivo recibir y transmitir datos al mismo tiempo, teniendo así una comunicación del tipo llamado full-duplex (Figura 3).

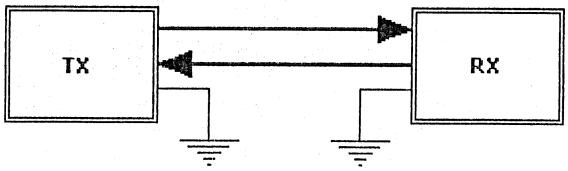

Figura 3. Transmisión serie tipo full-duplex.

4.2.1. Estándares RS-232 y RS-485.

La resistencia y capacitancia de los cables largos actúan como filtros que suavizan las transiciones agudas de las señales digitales. Este efecto es más evidente cuando se requieren detectar pulsos de corta duración (altas velocidades de transmisión) donde se observa una relación inversa entre la longitud del cable y la velocidad máxima de transmisión de datos.

El estándar RS-232 define 20kbps como su velocidad máxima y 15m la longitud máxima del cable de transmisión. Esta interfaz es útil para la comunicación punto a punto a velocidades bajas, pero no para ser utilizado en sistemas de alta velocidad a largas distancias.

En cambio, el estándar RS-485 es utilizado para comunicaciones multipunto con la mayoría de los sistemas empleando una arquitectura maestro/esclavo donde el esclavo tiene una dirección única y el maestro periódicamente monitorea a todos los esclavos conectados. Este estándar permite tener, en el mismo ducto, hasta 32 controladores y 32 receptores con distancias de transmisión de hasta 100m para velocidades de hasta 1Mbps cuando se utiliza cable del tipo par-trenzado. Para distancias inferiores a 15m se puede aumentar la velocidad de transmisión hasta 10Mbps. Las velocidades de transmisión más altas empleadas para distancias muy grandes, ~1.2km, están entre 19200 y 38400 baudios. Si se utiliza menor velocidad de transmisión estas interfaces podrán tolerar cables aún más largos. De hecho existen transmisores RS-485 con alcance de hasta 11km a velocidades de 1200 baudios. La Figura 4 muestra la relación entre distancia de transmisión (longitud del cable) y la velocidad de transmisión.

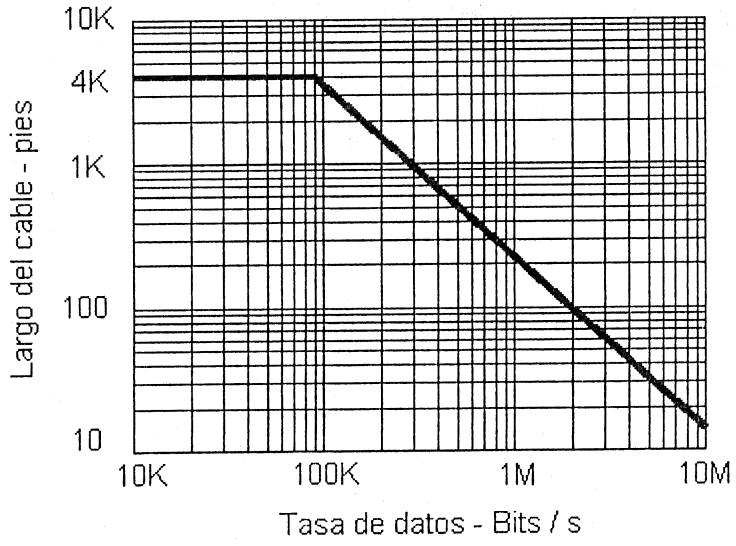

Figura 4. Relación distancia-velocidad de transmisión para el RS-485.

La diferencia principal entre los estándares RS-232 y el RS-485 consiste en que en el primero las señales son representadas por niveles de voltaje con respecto a tierra, mientras que en el segundo cada señal utiliza un par de cables por los que se lleva a cabo la transmisión de señales de manera diferencial. La Figura 5 muestra la estructura de un ducto RS485 half-duplex, observamos que ambos transmisor y receptor están conectados al mismo par de cables, teniendo la línea digital TE como línea de control de flujo la cual puede ser controlada ya sea por software o por circuitería.

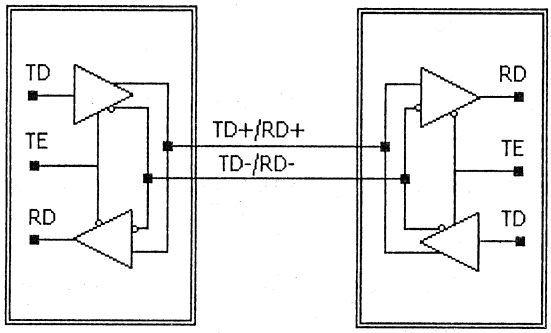

Figura 5. Transmisión half-duplex en un ducto diferencial RS-485.

4.2.2. MAX232

El dispositivo MAX232 (http://www.ti.com/) es un receptor/controlador dual que cambia los niveles de voltaje de una fuente de 5V (TTL ó CMOS) a los niveles de voltaje definidos por el estándar RS-232. Cada receptor convierte los niveles de entrada de EIA-232 a niveles de 5V mientras que cada controlador convierte los niveles de entrada TTL/CMOS a niveles EIA-232. El circuito típico de operación se muestra en la Figura 6.

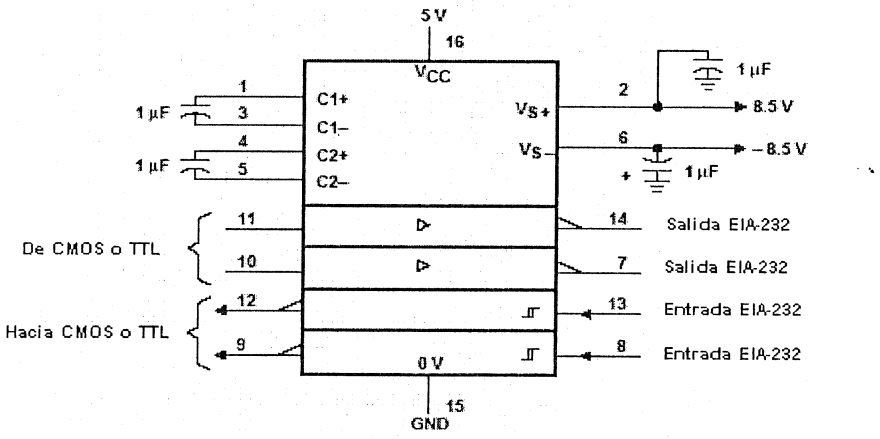

Figura 6. Aplicación típica del receptor/controlador MAX232.

4.2.3. SN75176

El dispositivo SN75176 (http://www.ti.com/) es un transmisor/receptor de ducto diferencial diseñado para comunicación bidireccional de datos en líneas de ductos de transmisión multipunto.

Este dispositivo combina un línea controladora diferencial con tercer estado y una línea receptora de entrada diferencial las cuales operan con una sola fuente de alimentación de 5V. El controlador y receptor tienen habilitaciones activo-alto y activo-bajo (DE y $\overline{RE}$ respectivamente), que se pueden conectar externamente para funcionar como control de dirección. Las salidas diferenciales de control y las entradas diferenciales del receptor están conectadas internamente para formar puertos de ducto de entrada/salida diferencial, para ofrecer un mínimo de carga al ducto cuando el controlador está deshabilitado o cuando no hay voltaje de alimentación.

Este controlador está diseñado para soportar cargas de hasta 60mA, también impone limites de voltaje (entre –10V y 15V) así como impedancia mínima de entrada de 12kΩ. Tiene una sensibilidad de entrada de ±200mA y una histéresis típica de 50mV. La Figura 7 muestra el diagrama lógico del transmisor / receptor SN75176.

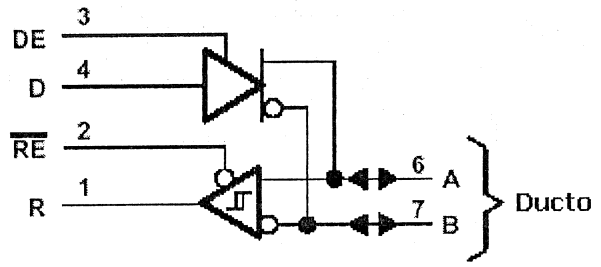

Figura 7. Diagrama del transmisor/receptor SN75176.

La Figura 8 presenta un diagrama esquemático de la etapa de comunicación serie. Observamos que tanto el circuito integrado MAX232 como el SN75176 están conectados a las patas RX y TX del microcontrolador por medio del puente JP2 con el cual se selecciona el tipo de comunicación que se deseé (RS-232 ó RS-485), además se tiene un conector DB9 para realizar la comunicación entre módulos (RS-485), la comunicación módulo-PC (RS-232) y la alimentación del módulo (de 9 a 30V).

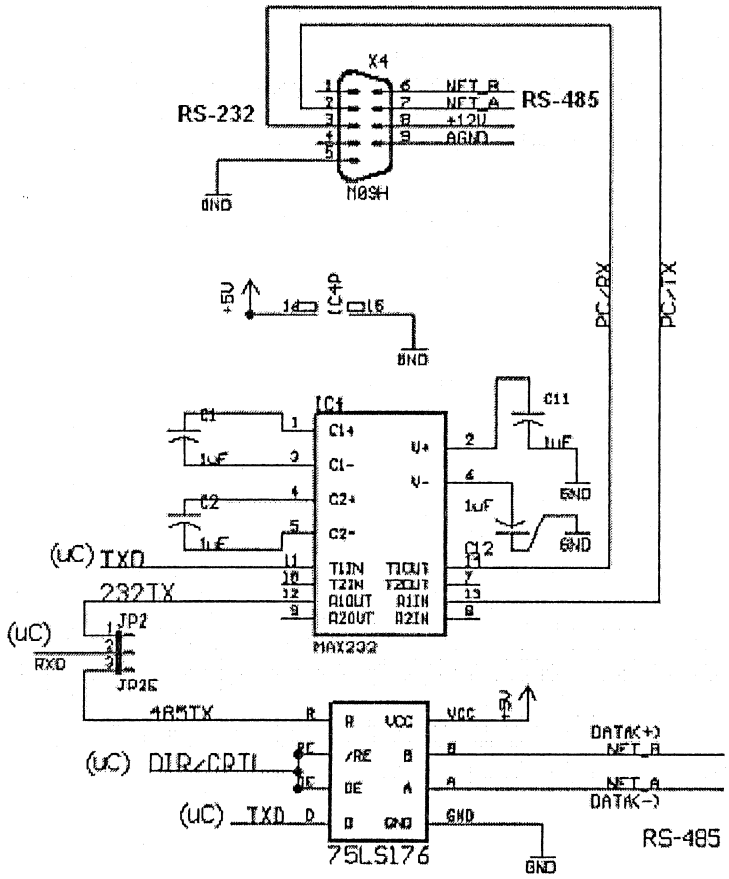

Figura 8. Diagrama de conexión del RS-232/RS-485

5. Diseño electrónico.

5.1. Módulo de adquisición de datos.

El módulo de adquisición de datos está basado en el diseño de la estación meteorológica de Chapela (2004) en el cual los módulos están conectados en paralelo al puerto serie de la computadora, lo cual proporciona las siguientes ventajas:

- Simplificación del diseño. Se diseñó solo un módulo y se agregan tantos como requiera la aplicación.
- Rapidez en reparación ya que solo hay que reemplazar el módulo dañado.
- Bajo costo ya que es más barato construir copias de circuitos idénticos que hacer una serie de circuitos diferentes.
- Protección al sistema. Todos los módulos son independientes entre sí, por lo que cualquier problema no dañará al sistema completo, sino que solo al módulo afectado.

Los elementos y etapas principales son (véase la Figura 9):

- Microcontrolador.
- Etapa de transmisión/recepción serie.
- Etapa de acondicionamiento de las señales.

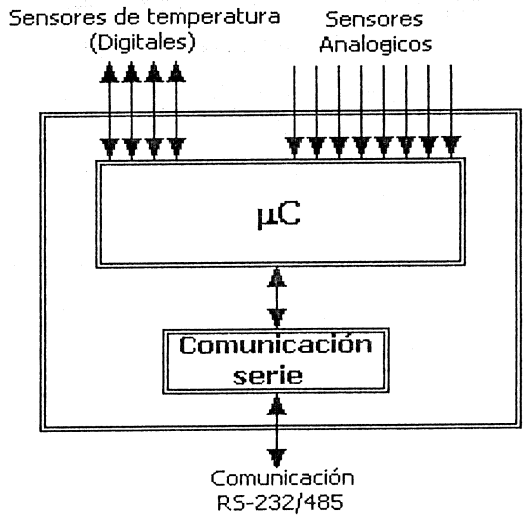

Figura 9. Diagrama a bloques del módulo de adquisición.

5.2. Diagrama electrónico.

El diseño del diagrama electrónico así como el de la tarjeta impresa se realizó con la ayuda del paquete EAGLE (http://eagle-systems.net/), la Figura 10 presenta el diagrama esquemático del módulo.

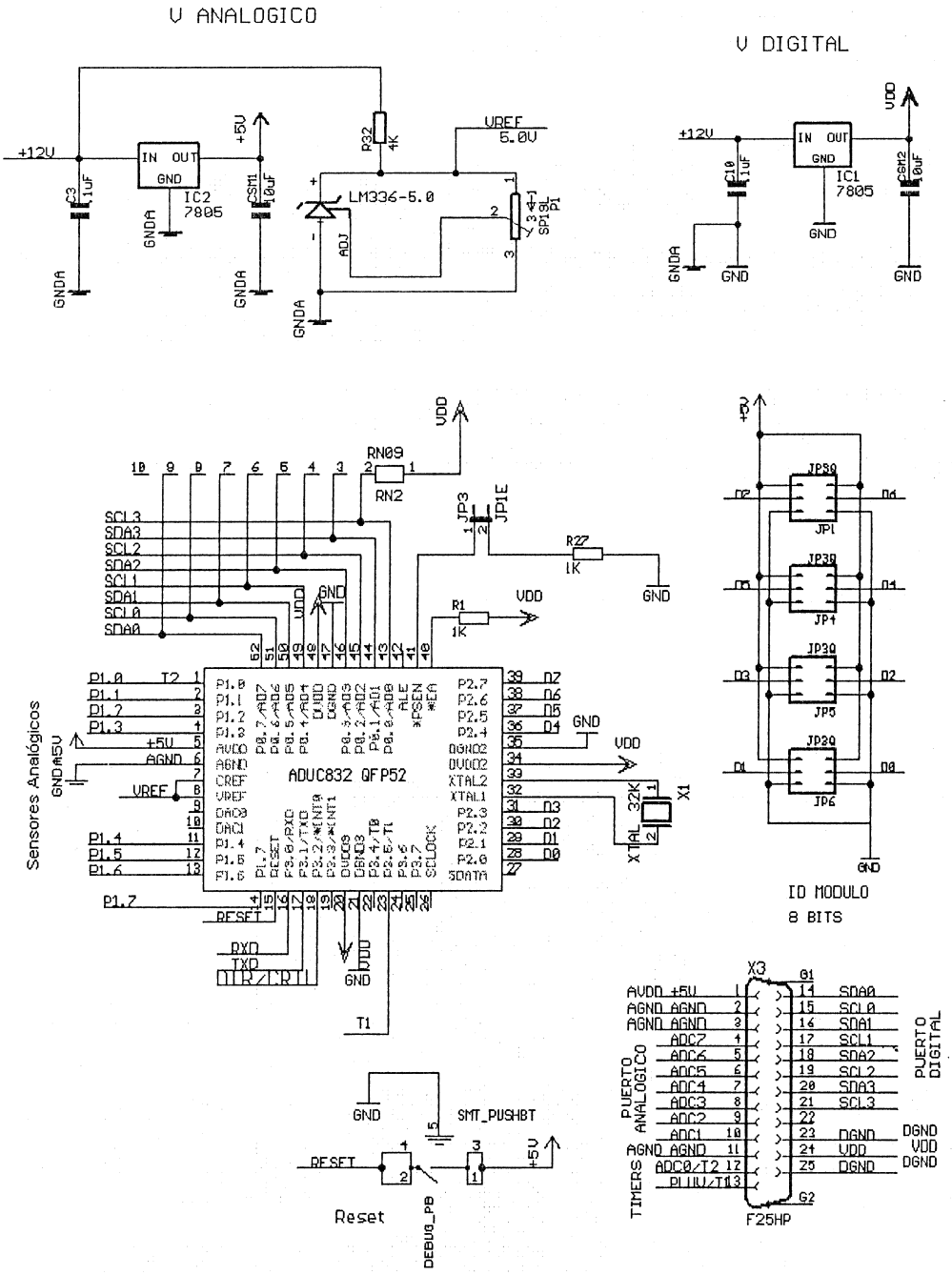

Figura 10. Diagrama esquemático del módulo.

Algunas etapas de nuestro diagrama ya fueron mencionadas, por lo que no aparecen en ésta figura. Se observa que los voltajes de alimentación son proporcionados por los reguladores de voltaje 7805 (IC1 e IC2), teniendo la alimentación digital y analógica independientes entre si. También se cuenta con una referencia de voltaje de 5V LM336-5.0 para proporcionar el voltaje de referencia del convertidor A/D.

La componente principal es el circuito integrado ADuC832 donde el puerto 1 está configurado como convertidor A/D. Los voltaje en las líneas AVDD y AGND (analógicos) son proporcionados por el regulador IC2 mientras que los voltajes de las líneas VDD y GND (digitales) son suministrados por el regulador IC1.

La línea RESET va conectada al push button que se encarga de reiniciar al microcontrolador y dejarlo en condición de cargarle el programa (modo "download") siempre y cuando se tenga el puente JP3 conectado a la línea $\overline{\text{PSEN}}$. La comunicación serie se da a través de las líneas RXD (P3.0) y TXD (P3.1) mientras que la línea P3.2 está destinada a controlar la línea de dirección del transmisor/receptor SN75176 (controla la dirección de transmisión/recepción de la comunicación serie RS-485). La línea P3.5 está configurada como contador de pulsos y es utilizada básicamente para recibir los pulsos generados por el pluviómetro y el anemómetro.

Como se comentó anteriormente, el microcontrolador requiere de un cristal de 32.768kHz el cual va conectado a las líneas XTAL1 y XTAL2. El puerto 2 está conectado a los puentes JP1, JP4, JP5 y JP6 con el propósito de proporcionar la identificación o nombre del módulo teniendo la línea P2.0 como el bit menos significativo.

La línea $\overline{\text{EA}}$ esta conectada a VDD a través de una resistencia de 1kΩ, con ésto, al encender o reiniciar al dispositivo se accesan los 62kB de memoria de programa FLASH/EE interna. El puerto 0 se utiliza para la comunicación con los sensores digitales de temperatura ya que por medio de programación generamos 4 ductos de comunicación I^2C. En la Figura 11 se muestra la tarjeta impresa del módulo.

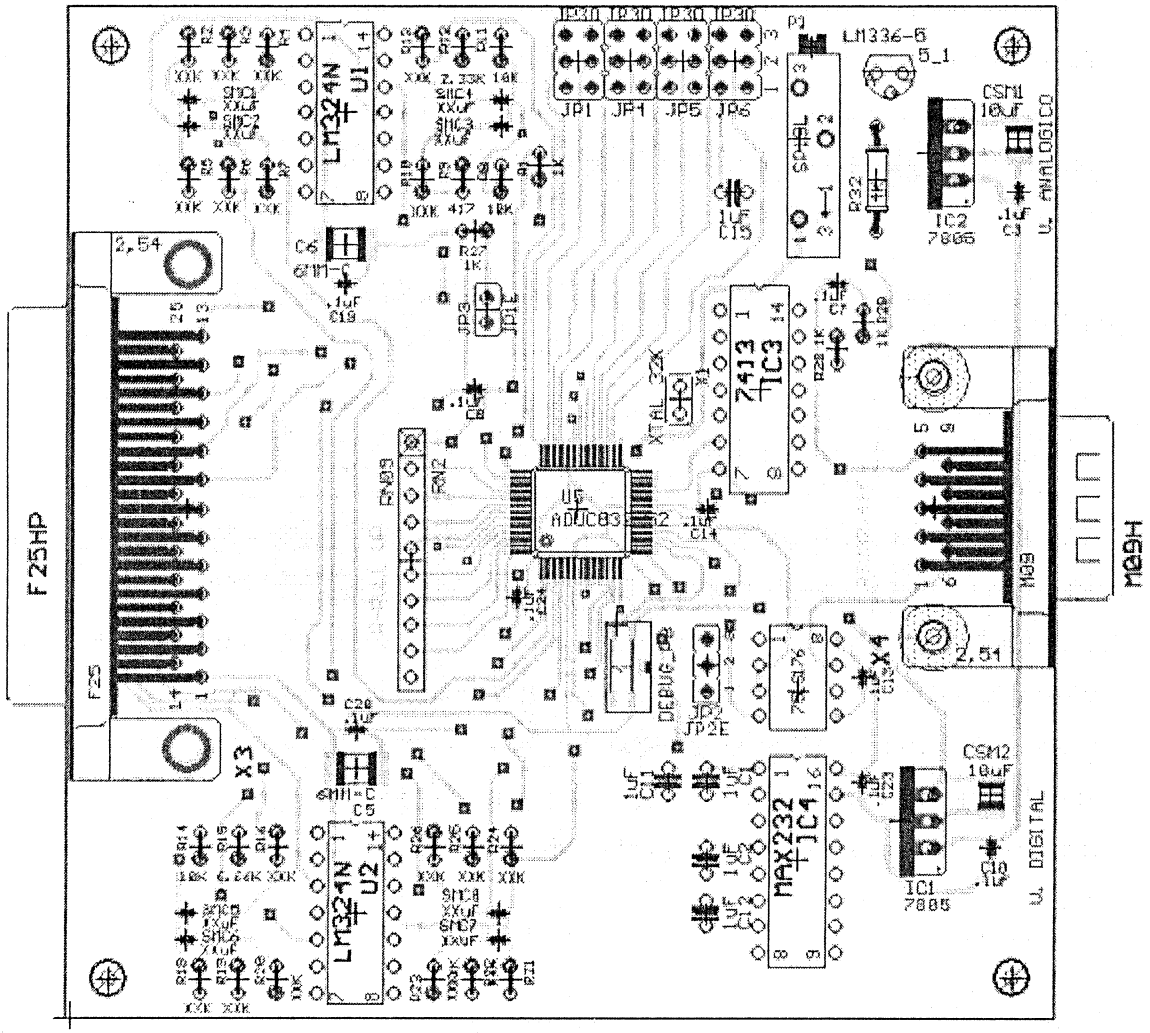

Figura 11. Tarjeta impresa del módulo.

6. Sensores.

6.1. Temperatura.

El sensor de temperatura utilizado es el LM92 de National Semiconductor (http://www.national.com/) véase la Figura 12. Este es un sensor de salida digital, con una resolución de 12 bits + signo, el cual se comunica por medio del estándar I^2C. Su voltaje de alimentación debe estar entre 2.7V y 5.5 V y su rango de temperaturas de operación es de -55 a 150°C. Su resolución es de 0.0625°C.

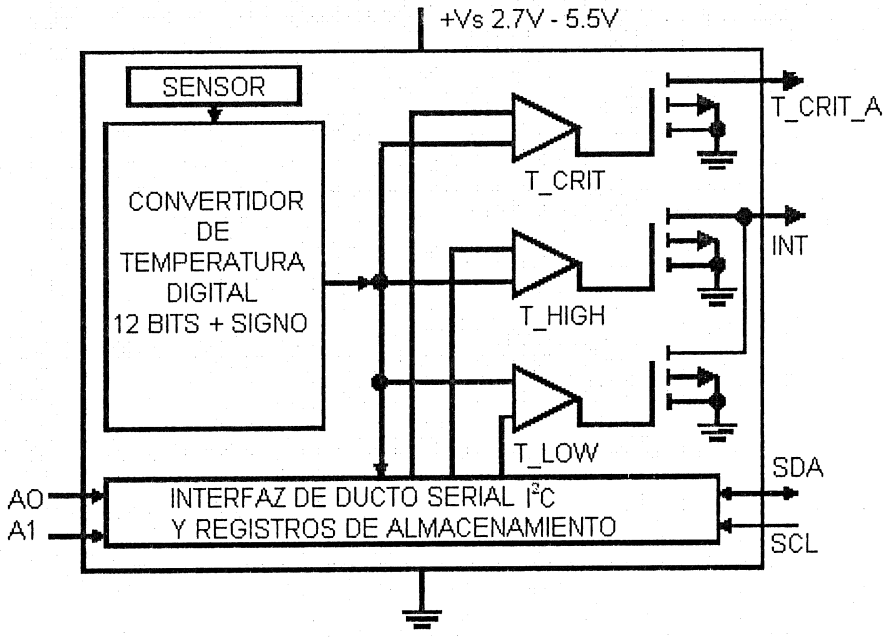

Figura 12. Diagrama a bloques del sensor LM92.

Con las entradas A0 y A1 se define la dirección, o identificación, de cada sensor por lo que es posible tener hasta cuatro sensores compartiendo los mismos cables. Mediante puentes conectados a tierra ó +Vs se definen cada una de las cuatro posibles direcciones (00,01,10 y 11).

Las salidas digitales INT y T_CRIT_A pueden ser utilizadas para controlar sistemas. La línea INT produce una señal que se activa cuando la temperatura se sale de dos limites programados por el usuario mientras que T_CRIT_A se activa cuando la temperatura excede un limite también programado por el usuario. Estas salidas no son empleadas en nuestra aplicación.

La línea de entrada SCL proporciona la sincronía para la transmisión y recepción de datos los cuales fluyen a través de la línea bidireccional SDA. La comunicación está apegada al protocolo de comunicación definido por el estándar I^2C.

Los datos de temperatura están representados por 13 bits en formato complemento a dos. En la Tabla 1 se muestran algunos ejemplos de esta relación.

VALOR BINARIO	TEMPERATURA (°C)
0 0001 1001 0000	+25
0 0000 0000 0001	+0.0625
0 0000 0000 0000	0
1 1111 1111 1111	-0.0625
1 1110 0111 0000	-25

Tabla 1. Ejemplos del formato de datos de temperatura del LM92.

Para nuestra aplicación, el rango de temperaturas que nos interesa medir va de -20 a 30°C aproximadamente. En la Tabla 2 se muestran los valores de precisión del sensor a diferentes temperaturas, así como algunas especificaciones ya mencionadas según su hoja de datos.

PARAMETRO	VALOR
Voltaje de alimentación	2.7V a 5.5V
Corriente de alimentación	350μA
Linealidad	±0.5 °C
Resolución	.0625°C/ADU
Rango de operación	-50 a 150 °C

Tabla 2. Especificaciones del sensor de temperatura LM92.

El LM92 opera como esclavo en el ducto serial, así la línea SCL es una entrada (no genera ningún reloj, sino que siempre es recibido del dispositivo maestro) y la línea SDA es una línea de datos bidireccional. En la Figura 13 se presenta un diagrama de tiempos para el funcionamiento de este sensor en el modo de lectura (el único empleado en esta aplicación). Siguiendo el diagrama se observa que primero se debe tener una condición de inicio, la cual siempre es generada por el maestro del ducto. Esta condición sucede sólo si se realiza una transición de alto a bajo en la línea SDA mientras la línea SCL se encuentre en estado alto. Una vez realizada la condición se dice que el ducto está ocupado o en funcionamiento.

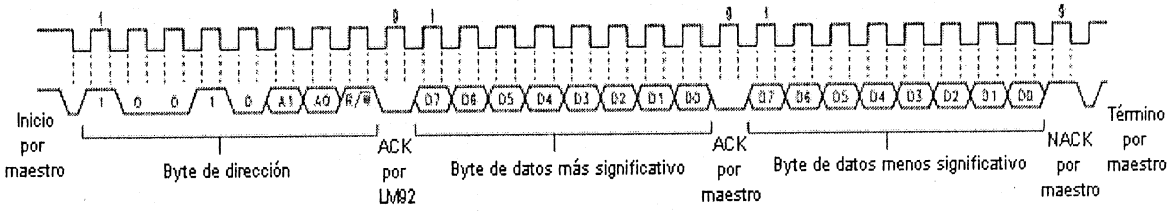

Figura 13. Diagrama de tiempos para lectura del sensor LM92 .

Después de ésto el maestro envía un byte (con el bit más significativo por delante) por la línea SDA junto con 8 pulsos de reloj, los primeros 7 bits son de dirección, de los cuales los 5 bits más significativos están pre-programados en un registro interno del LM92, siendo estos (10010) y los 2 bits menos significativos de la dirección son definidos por las entrada A1-A0. El bit 8 (menos significativo) es un bit de reconocimiento de lectura/escritura ($R/\overline{W}$) el cual determina la dirección del mensaje, que para esta aplicación deberá ser de lectura. El siguiente bit es de reconocimiento (ACK) generado por el LM92 reflejado en un nivel lógico bajo en la línea SDA señalando que recibió la dirección.

Siguiendo lo anterior tenemos ya el dato de temperatura guardado en los registros del LM92, el sensor transmite primero el byte más significativo, seguido de un bit de reconocimiento (ACK) generado por el maestro del ducto reflejado en un nivel lógico bajo en la línea SDA señalando que recibió el byte satisfactoriamente. Posteriormente el LM92 envía el byte menos significativo siendo los últimos tres bits de este las banderas de alarma CRIT, HIGH y LOW, un bit de nivel lógico alto deberá ser generado por el maestro (NACK) para aprobar satisfactoriamente la transferencia de los datos, el cual va seguido de la condición de término que siempre es generada por el maestro del ducto. Ésta condición sucede sólo si se realiza una transición de bajo a alto en la línea SDA

mientras la línea SCL se encuentre en estado alto. Una vez realizada esta condición se dice que el ducto está liberado o en reposo.

Es indispensable mencionar que un dato es válido solo si la línea SDA está estable durante un periodo de reloj alto, esto es, una transición en la línea SDA sólo puede ocurrir cuando la señal de reloj en la entrada SCL sea bajo.

En la Figura 14 se muestra la aplicación típica del sensor LM92. El uso del condensador de desacoplo es indispensable para cables largos pero si el sensor está cerca del módulo entonces se puede prescindir de él.

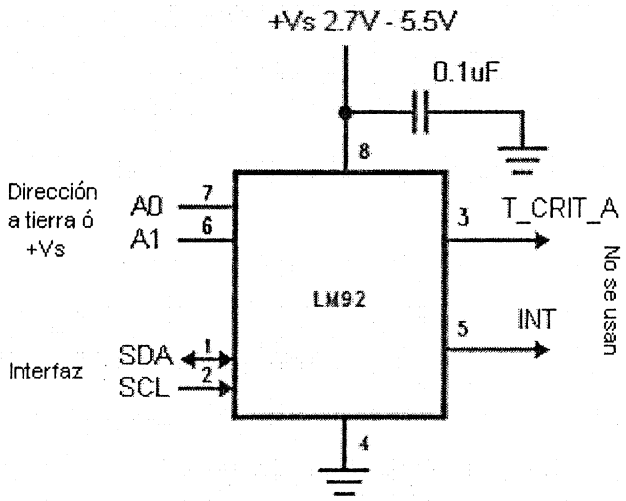

Figura 14. Aplicación típica del LM92.

6.1.1. Calibración e instalación.

Antes de la instalación de los termometros fue necesaria su calibración para lo cual se habilitaron dos módulos que nos permitieron calibrar hasta 32 sensores a la vez. Como referencia para la calibración se emplearon termómetros comerciales calibrados de fabrica. Se pusieron los sensores y las puntas de los termómetros en el interior de una hielera lo más cercanos posible entre sí. La hielera se cubrió con una cobija para evitar la entrada de corrientes de aire por las ranuras. Con la hielera apagada y con ayuda de un calentón, se calentó el aire dentro de la hielera a una temperatura de aproximadamente 30°C. Después se ejecutó un programa de adquisición de datos

que generaba los promedios de temperatura de los sensores cada minuto y se encendió la hielera dejándola enfriar hasta aproximadamente -20°C, el programa tomó datos por tiempo de alrededor de 2 horas. Procesando estos datos se obtuvieron valores de calibración para todos los termometros, estos valores se escriben en un archivo que es leído por el programa de la PC de manera que se corrija el valor que entregue el sensor.

Los termómetros utilizados para la calibración de los sensores fueron:

- Un termómetro de mercurio de laboratorio.
- Un termómetro OMEGA modelo DP460 con termopar, que tiene una precisión de ±0.5°C.
- Un transductor RTD (sensor de temperatura resistivo) Honeywell modelo HEL-707, consistente en una película delgada de platino con salida de 100Ω a 0°C. Este sensor tiene una precisión de ±0.3°C en el rango de temperatura de -75 a 540°C.

6.2. Humedad relativa.

La humedad relativa es la relación entre la cantidad de vapor de agua contenido en el aire y la máxima cantidad que podría contener. Cuanto mayor es la temperatura, mayor es la capacidad del aire para absorber vapor de agua, por lo que esta capacidad depende directamente de la temperatura del aire. La humedad relativa se expresa en porcentaje, así, cuando la humedad relativa llega al 100% significa que el aire está saturado de agua.

El sensor de humedad relativa utilizado es el HIH-3610-003 de Honeywell (http://www.honeywell.com/) el cual produce una voltaje lineal de salida que es directamente proporcional a la humedad relativa. Este sensor cuenta con un capacitor plano con una segunda capa de polímero para protegerlo de polvo, suciedad, etc. En la Tabla 3 se muestran algunas de sus especificaciones.

Parámetro	Min.	Típico	Máx.
Voltaje de alimentación (V)	4	5	5.8
Corriente de alimentación (μA)	-	200	-
Rango de humedad (%)	0	-	100
Temperatura de operación (°C)	-40	-	85
Precisión (%)	-	±2 (a 25°C)	-

Tabla 3. Especificaciones del sensor de humedad HIH-3610.

Cada sensor viene de fabrica con valores de calibración propios. Los valores de calibración de los sensores utilizados se muestran en la Tabla 4.

Sensor	89	404	386	268
Vsal @ 0% RH (V)	0.866	0.885	0.851	0.884
Vsal @75.3% RH (V)	3.269	3.235	3.182	3.244

Tabla 4. Valores de calibración de los sensores de humedad.

En la Figura 15a se muestra la construcción de este tipo de sensores mientras que en la Figura 15b se ve la dependencia que tienen con la temperatura. Segun el fabricante esta dependencia no deseada puede ser corregida con la ecuacion $RH = RH_0/(1.0546-0.00216T)$ donde RH_0 es la salida del sensor y T es la temperatura en grados centigrados.

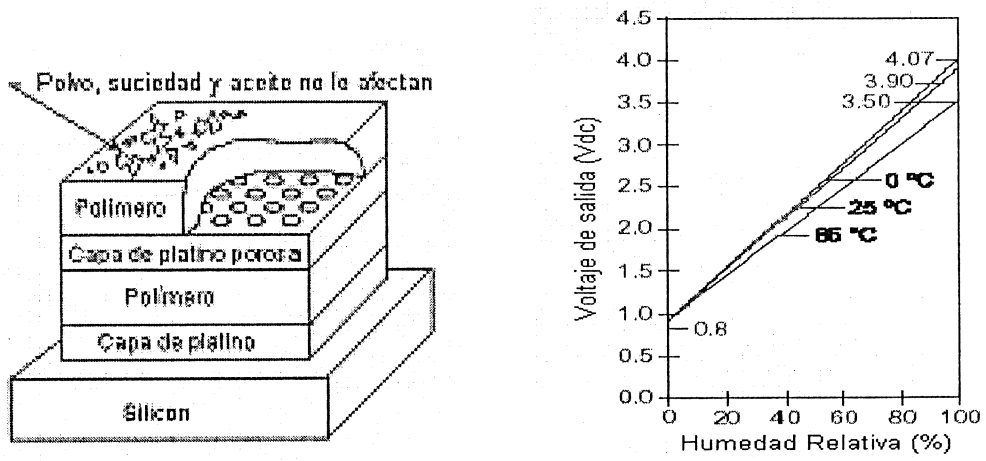

Figura 15. a) composicion del sensor de humedad. b) su dependencia con la temperatura.

6.3. Presión atmosférica.

Se define como la fuerza por unidad de superficie ejercida por el peso de la atmósfera (aire). Al nivel del mar, la presión atmosférica es de alrededor de 101.3kPa (equivalente a 760mmHg), mientras que a una altura de aproximadamente 5500 metros esta presión se reduce a la mitad.

El sensor de presión atmosférica utilizado es el MPX4115A de Motorola (http://www.motorola.com/). Este sensor proporciona un voltaje de salida compensado por temperatura y está compuesto por un transductor piezo-resistivo monolítico de silicio que permite una señal de salida precisa que es proporcional a la presión aplicada. Algunas de sus especificaciones se muestran en la Tabla 5.

Parámetro	Min.	Típico	Máx.
Rango de operación (kPa)	15	-	115
Voltaje de alimentación (V)	4.85	5.1	5.35
Corriente de alimentación (mA)	-	7	10
Voltaje de salida (V)	0.2	-	4.8
Tiempo de respuesta (ms)	-	1.0	-
Temperatura de operación (°C)	-40	-	125

Tabla 5. Especificaciones del sensor de presión MPX4115A

La Figura 16a muestra la construcción del sensor de presión atmosférica, mientras que la Figura 16b muestra la curva de respuesta del sensor, voltaje de salida contra presión.

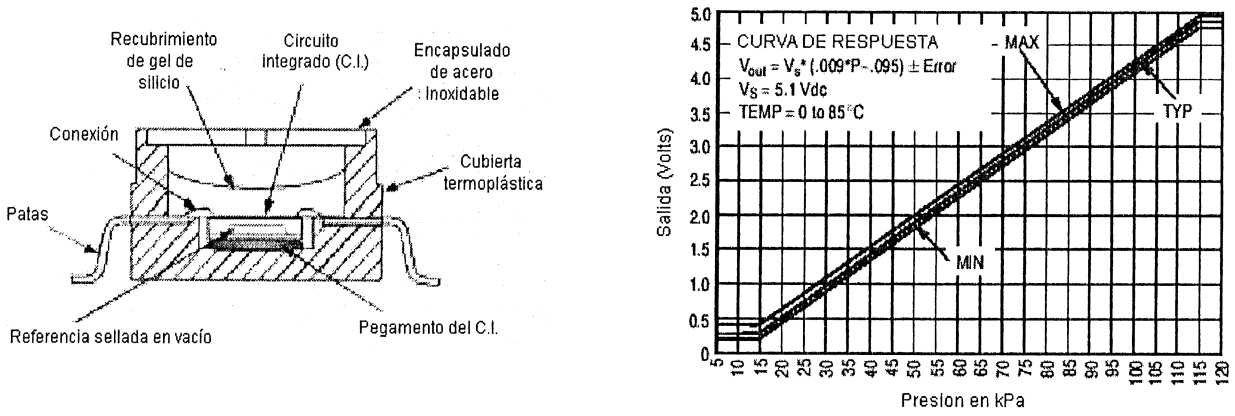

Figura 16. a) construcción del sensor de presion y b) curva de respuesta.

6.4. Pluviómetro.

La precipitación pluvial es cualquier tipo de agua que cae sobre la superficie de la Tierra. Las diferentes formas de precipitación incluyen lluvia, neblina, nieve, agua nieve, y granizo.

La Figura 17 muestra la composición interna del sensor de precipitación pluvial Rain Collector 7852M de Davis Instruments (http://www.davisnet.com/). En este sensor, la lluvia entra a un cono colector para posteriormente pasar a través de un pequeño filtro (malla) y llenar una de las cámaras del contenedor. Cuando se colecta una cantidad de agua correspondiente a una altura de 0.2mm, se inclina mecánicamente el contenedor de manera que esta agua es vertida. Cuando el contenedor se inclina, se activa un interruptor magnético cuyo contacto genera un pulso de voltaje que será recibido por el contador del microcontrolador. Al inclinarse se pone automáticamente la otra cámara del contenedor en posición de colectar.

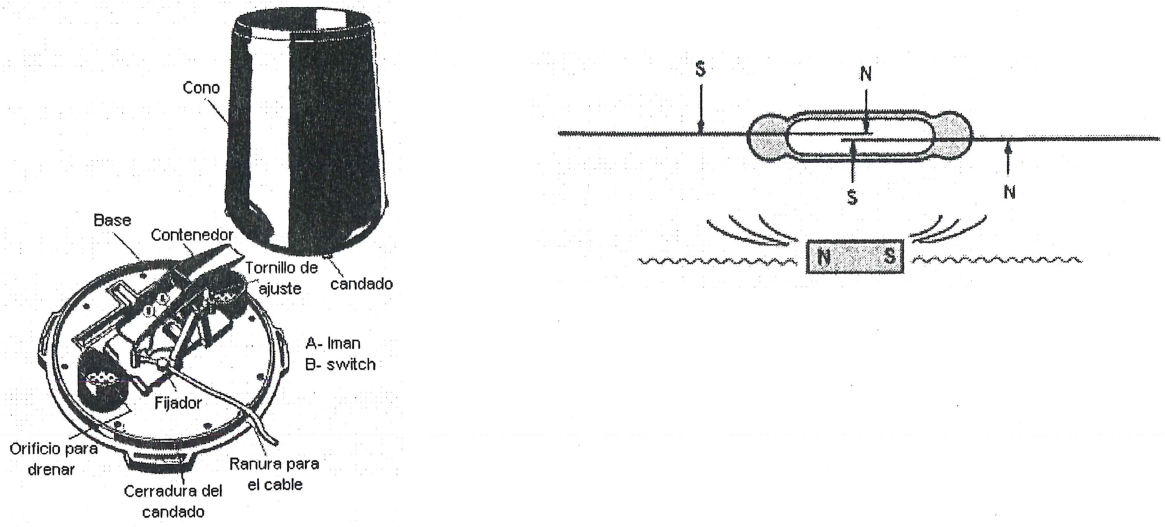

Figura 17. a) pluviometro y b) detalle del interruptor magnetico.

Para el mejor funcionamiento del contador, se integró el circuito antirebote mostrado en la Figura 18. Este circuito se basa en un schmitt trigger que evita que los transitorios parásitos sean registrados.

Al cerrar el interruptor del pluviómetro, el capacitor se descargará a través de R2 hasta 0.9V y la salida del schmitt trigger pasará a nivel alto. Al abrir el interruptor el capacitor se volverá a cargar y cuando alcance 1.7V, la salida pasará a nivel bajo, así los rebotes del interruptor mecánico no tendrán efecto, porque serán absorbidos por el capacitor mientras este se carga o se descarga.

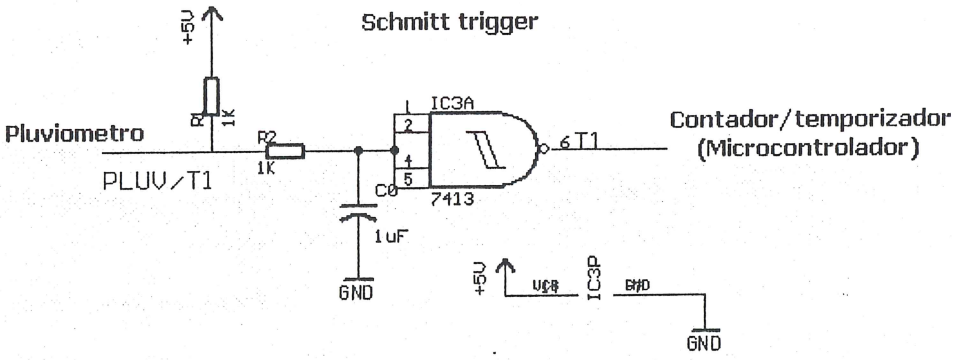

Figura 18. Circuito antirebote para pluviómetro.

6.5. Radiación solar.

Se conoce por radiación solar al conjunto de radiaciones electromagnéticas que son emitidas por el sol. Estas van desde los rayos gama hasta las ondas de radio. La unidad práctica que describe la radiación solar que llega a la tierra es la irradiancia y está medida en watts por metro cuadrado.

El sensor de radiación solar que utilizamos es el modelo 7821 de Davis Instruments (http://www.davisnet.com/). Este sensor consiste en un fotodiodo de silicio de respuesta espectral ancha, el cual convierte la radiación incidente en un voltaje. Este sensor mide la radiación global, por lo que toma en cuenta tanto la componente directa así como la componente difusa de la radiación solar.

La Tabla 6 muestra algunas especificaciones de este sensor.

Parámetro	Valor
Voltaje de alimentación	5V, ±10%: 3mA típico
Voltaje de salida	0V a 3V. (1.67 mV por W/m^2)
Precisión	±5%
Resolución	1 W/m^2

Tabla 6. Especificaciones del sensor de radiación solar 7821.

7. Programación

7.1. Programa del microcontrolador

El programa del microcontrolador se encarga de muestrear los sensores (digitales y analógicos), procesar los datos muestreados y enviar los resultados a la computadora. En el caso de los sensores digitales de temperatura, el resultado entregado de cada sensor es el promedio de la temperatura en grados centígrados así como su desviación estándar, para los sensores analógicos, se entregan los promedios de los voltajes medidos junto con sus desviaciones estándar. En el apéndice A se podrá encontrar un listado de este programa.

El promedio está dado por

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} ,$$

La desviación estándar está dada por la aproximación

$$\sigma = \sqrt{\overline{x^2} - (\bar{x})^2} ,$$

donde $\overline{x^2}$ es el promedio de los cuadrados y está dado por

$$\overline{x^2} = \frac{\sum_{i=1}^{n} x_i^2}{n} ,$$

Por lo que para calcular estos valores solo es necesario que el microcontrolador lleve las sumatorias de las mediciones, las sumatorias de las mediciones al cuadrado y el número de lecturas.

Un diagrama de flujos de este programa puede verse en la Fig.19. Al iniciar el programa se configuran los puertos de entrada/salida y serie del microcontrolador, los contadores-temporizadores, las interrupciones y el convertidor A/D. Ya que se realizó la configuración se lee la identificación (nombre) del módulo, se inicializa el contador T1 y las variables. Posteriormente se inicia el muestreo de los sensores primero el contador, después los sensores digitales (temperatura) y por último los sensores analógicos. Se van haciendo las sumatorias del valor de cada sensor leído así como la sumatoria del mismo valor al cuadrado hasta que se produce una interrupción por el puerto serie. Al realizarse la interrupción, el microcontrolador recibirá su identificación (nombre) ,

cuando esto sucede, guarda el valor del contador y lo reinicializa, después calcula el promedio y la desviación estándar para cada sensor (analógicos y digitales) y envía los resultados a la PC junto con el número de muestras y los pulsos del contador. Una vez terminada la transmisión de los datos se reinicializan los registros de las sumatorias y de las sumatorias de cuadrados y se repite el ciclo.

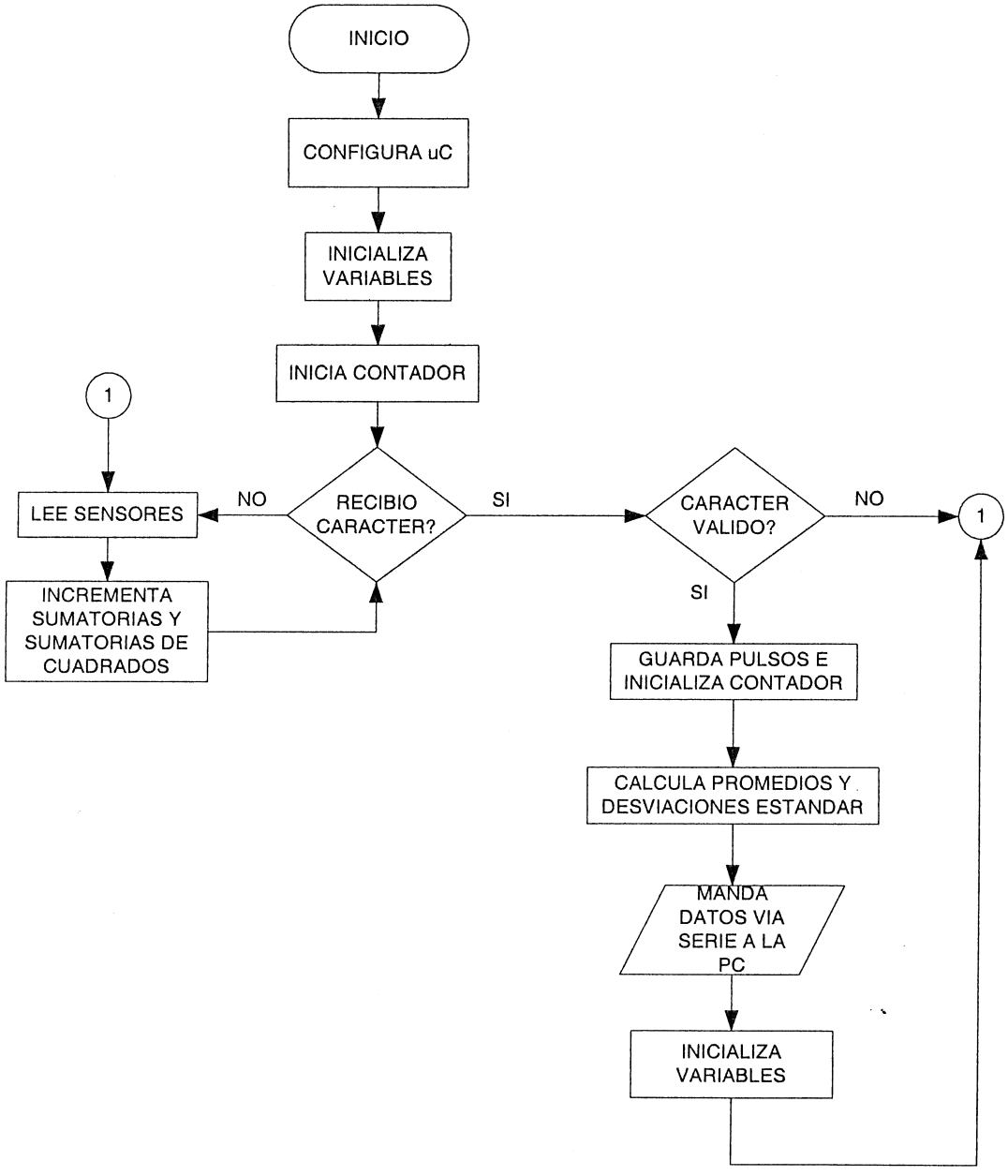

Figura 19. Diagrama de flujo del programa del microcontrolador.

7.2. Programa de la PC

Este programa fue desarrollado en lenguaje de programación C bajo el sistema operativo Linux y se ejecuta en la PC, se encarga de muestrear todos los módulos cada minuto y de generar un archivo de datos diario para cada modulo. En el apéndice B se puede encontrar un listado de este programa.

Como se muestra en el diagrama de flujo de la figura 24, al iniciar el programa se configura el puerto serie de la PC para una velocidad de 9600 baudios y se abre la comunicación con el puerto, posteriormente abre un archivo de configuración que contiene el nombre del telescopio así como la identificación de los módulos conectados. Despues de esto, abre el archivo de datos de calibración de los sensores y posteriormente inicializa los módulos con el fin de eliminar datos erróneos. Monitoreando el reloj interno de la PC, espera a que sea el inicio de un minuto (segundo 0), cuando esto sucede, envía el comando de identificación de los módulos para muestrarlos y despues escribe los resultados procesados en el archivo diario de datos y espera a que pase otro minuto para repetir el ciclo.

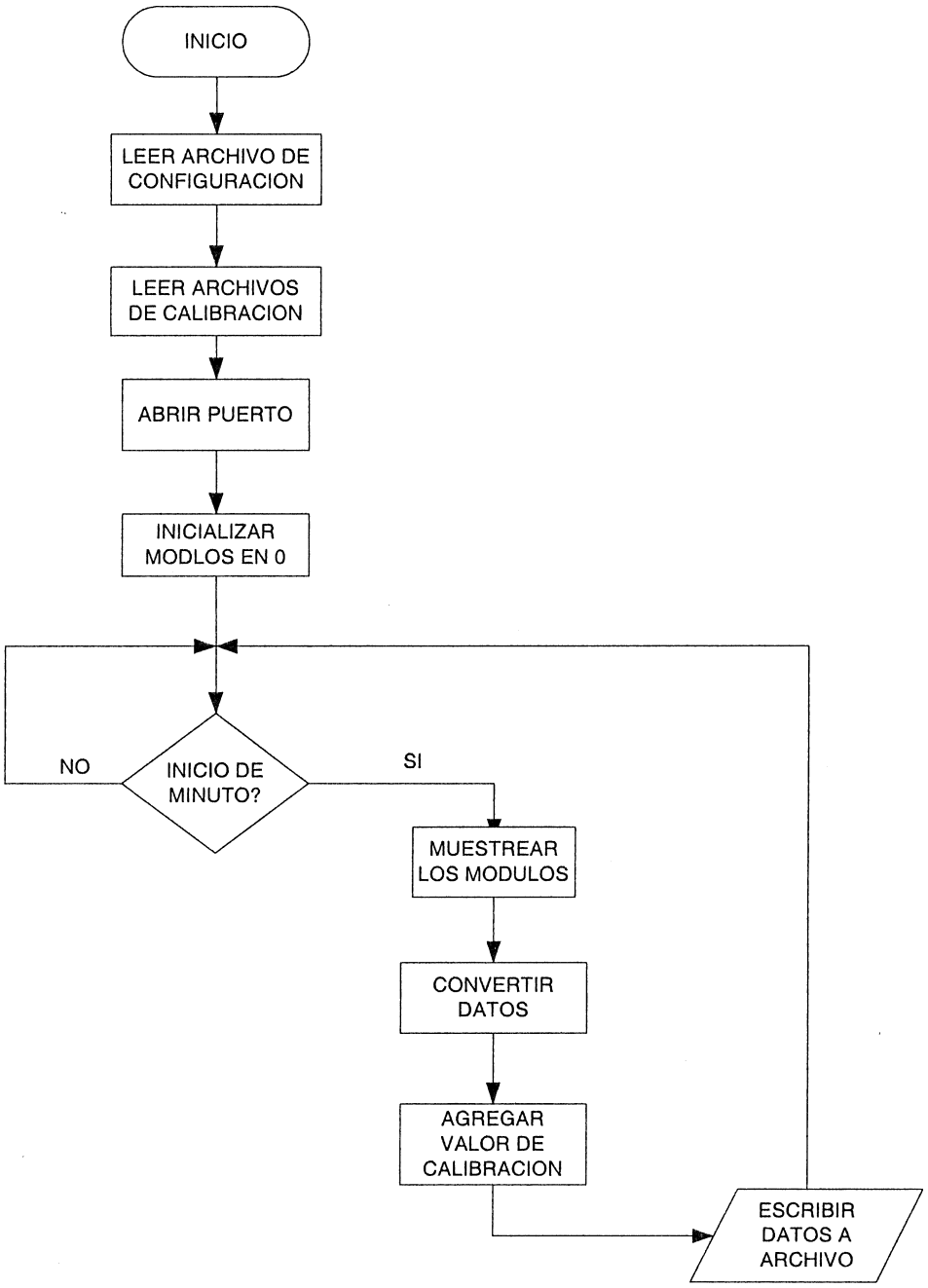

Figura 20. Diagrama de flujo del programa de la PC.

8. Resultados y Conclusiones.

Se han diseñado y construido nuevas estaciónes meteorológicas y de monitoreo de temperaturas para todos los telescopios del observatorio en SPM. Dichas estaciones constan de módulos de adquisición de datos basados en el microcontrolador ADuC832 de arquitectura 8052. Los módulos de cada estación están conectados en paralelo al puerto serie de una computadora que opera bajo el ambiente Linux. Las estaciones han estado operando de manera continua desde finales de noviembre del 2004, un ejemplo de los resultados entregados se puede observar en la Figura 21.

El presente trabajo me ha permitido enfrentar el reto de realizar un proyecto y aprender a atacar y superar los problemas que se presentan durante la realización del mismo. Además me permitió poner en práctica los conocimientos adquiridos en mi carrera y sobre todo me permitió obtener más conocimientos y desenvolverme en la práctica.

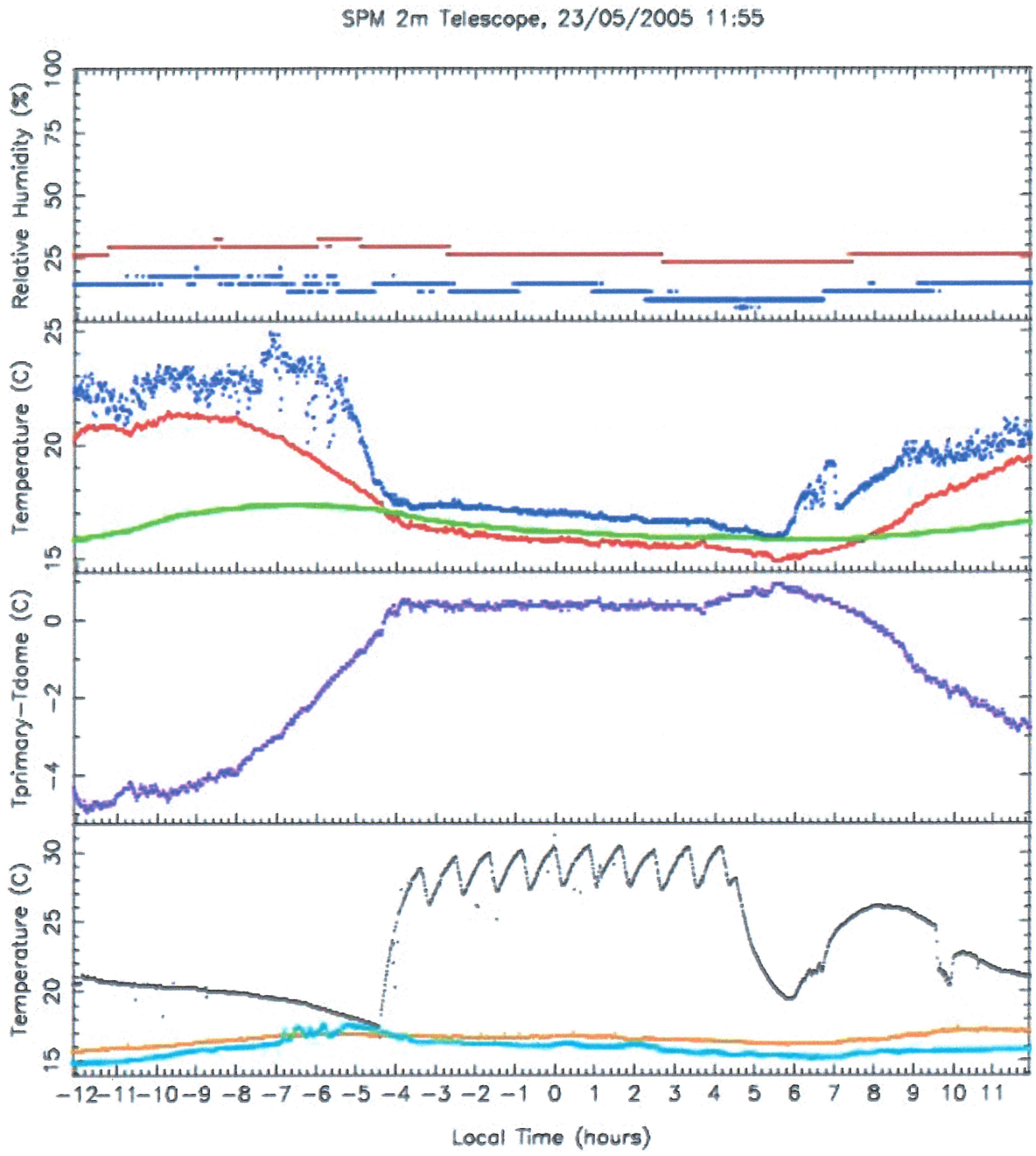

Figura 21. Ejemplo de algunas mediciones tomadas durante un día con la estación meteorológica del 2m. En la gráfica superior se presentan la humedad relativa dentro (rojo) y fuera (azul) del edificio. En la segunda gráfica se muestran las temperaturas del espejo primario (verde), del aire en la cúpula (rojo) y del aire en el exterior, nótese la caiía de temperatura de aproximadamente 5°C durante la noche. La tercera gráfica presenta la diferencia de temperatura entre el espejo primario y el aire en la cupula la cual afecta de manera directa la calidad de imagen del telescopio. En la última gráfica se presenta la temperatura en la planta baja del edificio (azul claro), en el primer piso (naranja) y en el baño (gris) donde claramente se ve el efecto del calefactor.

Referencias.

Chapela A. 2004. *"Estación Meteorológica para el Telescopio de 84cm de San Pedro Mártir"*. Tesis de licenciatura. UABC.

Cruz González et. al. 2004. SPIE. Vol. 5382, pp. 634-642.

Michel R. Bohigas J., Arroyo E. Zazueta S. 2001. *"The meteorological stations of the 1.5 and 0.84m Telescopes of the OAN: Description and results"*, Rev. Méx. AA, Vol. 37, pp. 165-171.

Michel R., Echeverría J., Costero R., Harris O., Magallón J. y Escalante K. 2003a. *"Seeing measurements at San Pedro Mártir observatory using the DIMM method"*, Rev. Méx. AA Vol. 39, pp. 291-301.

Michel R., Hiriart D. y Chapela A. 2003b. *"Four years of meteorological measurements at San Pedro Mártir observatory"*. Rev. Méx. AA Vol. 19, pp. 19-102.

Apendice A.
Programa del Microcontrolador.

```
#include <ADuC832.h>
#include <stdio.h>
#include <math.h>
#include "sio.h"
#include "funciones.h"

extern float sqrt  (float val);

sbit P0_7 = 0x87;
sbit P0_6 = 0x86;
sbit P0_5 = 0x85;
sbit P0_4 = 0x84;    // Configuración del puerto 0 para sensores de temperatura.
sbit P0_3 = 0x83;
sbit P0_2=  0x82;
sbit P0_1 = 0x81;
sbit P0_0 = 0x80;
sbit  LED = 0xB4;                  // Dirección por default del LED (mapa: p3.4).
sbit P3_5 = 0xB5;                              // Dirección de T1 (mapa P3.5).
sbit PCTL = 0xB2;                    // Dirección de Dir/Ctrl de 485 (mapa P3.2).
int N;
unsigned char h,i,j,bus;
unsigned int x;
xdata unsigned int a,b,d,maximo,minimo;
xdata float dig_temp,adc_volts,volts;
bdata int i2c_data;                            // Declaración de variable de signo.
bdata unsigned char i2c_addr=0x48;   // Palabra default de dirección del sensor.
unsigned char bdata CADBUSY;                 // Variable CADBUSY (genera 16 bits).
sbit BUSY = CADBUSY ^ 7;     // BUSY ADC (para direccionar el bit 7 de ADCCON3).
xdata float sum[8],sum_2[8];
xdata float temp0[4],temp0_2[4];
xdata float temp1[4],temp1_2[4];
xdata float temp2[4],temp2_2[4];
xdata float temp3[4],temp3_2[4];
char ID;
unsigned char comando[6],index=0;
sbit signo =i2c_data^15 ;                                          // Por rotación.

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-PRINCIPAL-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void main(void)
{

    CFG831=0x81;             // Registro que permite acceso a la memoria externa.
    com_initialize();               // Inicializa I/O serial usando el TIMER 3.
    com_baudrate();                                              // 9600 baudios.
    T1 = 1;                            // Habilita TIMER 1 para no usar P3.5.
//configura ADC'S.
    ADCCON1 = 0xEC;
    ADCCON2 = 0x80;                                                  // ADCH 0.

    EX0 = 0;                                 // Deshabilita la interrupción INT0.
    IT0 = 1;                                    // INT0 interrupt edge triggered.
    ET0 = 0;                                       // Deshabilita el timer 0 INT.
    EA  = 1;                                 // Habilita interrupciones (P3.2=1).

// Configura I2C.
    SPICON=0x0F;
    I2CM=1;

// Inicializa el bus I2C.
    for (bus=0; bus<=3; bus++) {manda_data(1); manda_clock(1);}
    inicializa_variables();                              // Inicializa variables.


    LED^ =0x0;
    PCTL = 0;
    lee_ID();                              // Lee jumpers de dirección del módulo.
    inicia_conteo();

    for(;;)                                                  // Ciclo FOR infinito.
    {
        if(com_rbuflen()>=1)
        {
            revisa_serie();                    // Revisa caracteres en el buffer.
        }
        cuenta_pulsos();                              // Pulsos en el contador T1.
        temperaturas();                  // Sensores de temperatura (digitales).
        sumatorias();//                                  // Sensores analógicos.
        N++;                                              // Número de muestras.
        retardo(6100);
    }
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void inicializa_variables(void)        // Inicializar todas las variables en 0.
{
    char i;

    N=1;
    a=0;
    adc_volts=0;
    for(j=0;j<4;j++) temp0[j]=0;
    for(j=0;j<4;j++) temp0_2[j]=0;
    for(j=0;j<4;j++) temp1[j]=0;
    for(j=0;j<4;j++) temp1_2[j]=0;
    for(j=0;j<4;j++) temp2[j]=0;
    for(j=0;j<4;j++) temp2_2[j]=0;
    for(j=0;j<4;j++) temp3[j]=0;
    for(j=0;j<4;j++) temp3_2[j]=0;
    for(i=0;i<8;i++) sum[i]=0;
    for(i=0;i<8;i++) sum_2[i]=0;
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void lee_ID(void)                              // Leer la identificación del módulo.
{
    ID=(P2&0x07);                                        //ID del micro (comando [3]).
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void inicia_conteo (void)              // Configurar el contador/temporizador T1.
{
    TMOD = 0x5B;
    TCON = 0x40;
    b=0;
    TH1=0;
    TL1=0;
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
```

```
void cuenta_pulsos(void)              // Pulsos en el contador/temporizador T1.
{
    b=(unsigned int)TH1&0x0F;
    b=b<<8;
    b|=(unsigned int)TL1;
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void retardo(unsigned int pause)                                    // Retardo.
{
    for(x=0;x<=pause;x++);
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void revisa_serie()           // Revisar si recibió si recibió un mando válido.
{
    int k,i,z=0;
    char p;

    while (z==0)
    {
         k=_getkey();
         if (index>0)
         {
             if (index>=5)
             {
                  comando[index]=k;
                  for(i=0; i<=5; i++) index=0;
                  if ((comando[3]-48)== (unsigned char)ID) p=1;
                  else p=0;
                  if ( (comando[1]=='2') && /
                     (comando[2]=='M') && /
                       ((comando[3]-48)==(unsigned char)ID) && /
                     (comando[5]==';'))
                  {
                      ejecuta();
                      z=1;
                  }
             }
             else
             {                                                      // Intermedios.
                  comando[index]=k;
                  index++;
             }
         }
         else
        {                                                           // Inicio.
             if (k==':')
             {
                  comando[index]=k;
                  index++;
             }
         }
    }
}


//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void ejecuta(void)                                  // Ejecutar comando válido.
{
    switch(comando[4])
    {
         case 'V':
         {
             N--;
          cuenta_pulsos();
             TH1=0;
             TL1=0;
             manda_datos();                                         // Manda via serie.
             inicializa_variables();                               // Inicializa variable.
             break;
         }
         case 'I':
         {
             printf("\nSoy el modulo:%c",ID+48);
             printf(" Version V1.0\n");
             break;
         }
    }
    PCTL = 0;
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void manda_datos()                                          // Mandar los datos del uC.
{
    char h;

    for (h=0; h<8; h++) sum[h]= (sum[h]/N);
    for (h=0; h<8; h++) sum_2[h]= sqrt ((sum_2[h]/N)-((sum[h])*(sum[h])));
    for (h=0; h<4; h++) temp0[h]= (temp0[h]/N);
    for (h=0; h<4; h++) temp1[h]= (temp1[h]/N);
    for (h=0; h<4; h++) temp2[h]= (temp2[h]/N);
    for (h=0; h<4; h++) temp3[h]= (temp3[h]/N);
    for (h=0; h<4; h++) temp0_2[h]=sqrt((temp0_2[h]/N)-((temp0[h])*(temp0[h])));
    for (h=0; h<4; h++) temp1_2[h]=sqrt((temp1_2[h]/N)-((temp1[h])*(temp1[h])));
    for (h=0; h<4; h++) temp2_2[h]=sqrt((temp2_2[h]/N)-((temp2[h])*(temp2[h])));
    for (h=0; h<4; h++) temp3_2[h]=sqrt((temp3_2[h]/N)-((temp3[h])*(temp3[h])));

    retardo(1000);
    printf(" %d %d %1.1f %1.1f ",N,b,sum[0],sum[1]);
    printf("%1.1f %1.1f %1.1f ",sum[2],sum[3],sum[4]);
    printf("%1.1f %1.1f %1.1f ",sum[5],sum[6],sum[7]);
    printf("%1.1f %1.1f %1.1f ",temp0[0],temp0[1],temp0[2]);
    printf("%1.1f %1.1f %1.1f ",temp0[3],temp1[0],temp1[1]);
    printf("%1.1f %1.1f %1.1f ",temp1[2],temp1[3],temp2[0]);
    printf("%1.1f %1.1f %1.1f ",temp2[1],temp2[2],temp2[3]);
    printf("%1.1f %1.1f %1.1f ",temp3[0],temp3[1],temp3[2]);
    printf("%1.1f %4.3f %4.3f ",temp3[3],sum_2[0],sum_2[1]);
    printf("%4.3f %4.3f %4.3f ",sum_2[2],sum_2[3],sum_2[4]);
    printf("%4.3f %4.3f %4.3f ",sum_2[5],sum_2[6],sum_2[7]);
    printf("%4.3f %4.3f %4.3f ",temp0_2[0],temp0_2[1],temp0_2[2]);
    printf("%4.3f %4.3f %4.3f ",temp0_2[3],temp1_2[0],temp1_2[1]);
    printf("%4.3f %4.3f %4.3f ",temp1_2[2],temp1_2[3],temp2_2[0]);
    printf("%4.3f %4.3f %4.3f ",temp2_2[1],temp2_2[2],temp2_2[3]);
    printf("%4.3f %4.3f %4.3f ",temp3_2[0],temp3_2[1],temp3_2[2]);
    printf("%4.3f \n",temp3_2[3]);
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
                  // Esperar hasta que se reciba un caracter por el puerto serie.
char _getkey (void)
{
    int k;

    do
    {
```

```
        k = com_getchar ();
    }
    while (k == -1);
    return ((unsigned char) k);
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
                                        // Poner caracteres en el puerto serie.
char putchar (char c)
{
    volatile unsigned int i;

    while (com_putchar (c) != 0)
    {
        for (i=0; i<1000; i++)
        {
            / no hacer nada /
        }
    }
    return (c);
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
            // sumatorias y sumatorias de cuadrados de los sensores analógicos.
void sumatorias(void)
{
    char i;

    for (i=0; i<8; i++)
    {
        adc_volts=lee_sensor_analogico(i);
        sum[i]=adc_volts+sum[i];
        sum_2[i]+=adc_volts*adc_volts;
    }
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
float lee_sensor_analogico(char sensor)                 // Leer los  sensores
analógicos.
{
    ADCCON2= 0x80|sensor;
    BUSY=0xFF;
    SCONV=1;                                        // Inicia una conversión.
    while(BUSY)                                     // Esperar a que termine.
    {
        CADBUSY=ADCCON3;     // El bit 7 (BUSY) en 0 al termino de la conversión.
    }
    a=(unsigned int)ADCDATAH&0x0F;
    a=a<<8;
    a|=(unsigned int)ADCDATAL;
    volts=a*0.00122100122;                              // Vref externo = 5V.
    return(volts);
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
    // Calcular sumatorias y sumatorias de cuadrados de los sensores digitales.
void temperaturas(void)
{
    char j=0;

    bus=0;
    for (j=0; j<4; j++)
    {
        dig_temp=lee_sensor_digital(j);
        temp0[j]=dig_temp+temp0[j];
        temp0_2[j]+=dig_temp*dig_temp;
    }

    bus=1;
    for (j=0; j<4; j++)
    {
        dig_temp=lee_sensor_digital(j);
        temp1[j]=dig_temp+temp1[j];
        temp1_2[j]+=dig_temp*dig_temp;
    }
    bus=2;
    for (j=0; j<4; j++)
    {
        dig_temp=lee_sensor_digital(j);
        temp2[j]=dig_temp+temp2[j];
        temp2_2[j]+=dig_temp*dig_temp;
    }
    bus=3;
    for (j=0; j<4; j++)
    {
        dig_temp=lee_sensor_digital(j);
        temp3[j]=dig_temp+temp3[j];
        temp3_2[j]+=dig_temp*dig_temp;
    }
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
float lee_sensor_digital(unsigned char sensor)   // Leer los sensores digitales.
{
    unsigned char ack,i,x=0,k,addr;
    int d=0;
    float f;
    unsigned char dd=0;

    addr=i2c_addr|sensor;
// START
    i2c_startp2();
    d=addr;                                             // Escibe dirección.
    i=7;
    manda_clock(0);
    do
    {
        i--;
        x=(addr>>i)&0x01;
        manda_data(x);
        manda_clock(1);
        retardo(dd);
        manda_clock(0);
    } while(i);

// Mandar READ/Write BYTE
    manda_data(1);
    manda_clock(1);
    retardo(dd);
    manda_clock(0);

// Leer ACK
    manda_clock(1);
    ack=lee_data();
    if (ack==0)
    {
```

```
        manda_clock(0);
        i2c_data=0;                                          // Limpiar variable.

// Lectura de los 8 bits mas significativos
        manda_data(1);
        for(i=15;i>=8;i--)
        {
            manda_clock(1);
            retardo(dd);
            k=lee_data();
            manda_clock(0);
            i2c_data=((unsigned int)k<<i)|i2c_data;
            d=((unsigned int)k<<i);
            i2c_data=d|i2c_data;
        }
// MASTER ACK
        manda_data(0);
        manda_clock(1);
        retardo(dd);
        manda_clock(0);

// Lectura de los 8 bits menos significativos
        manda_data(1);
        for(i=0,x=7;i<=7;i++,x--)
        {
            manda_clock(1);
            retardo(dd);
            k=lee_data();
            manda_clock(0);
            if (x==0)
                 i2c_data=((unsigned int)k)|i2c_data;
            else
                 i2c_data=((unsigned int)k<<x)|i2c_data;
        }
//MASTER NACK
        manda_data(1);
        manda_clock(1);
        retardo(dd);
        manda_clock(0);
//STOP
        i2c_stopp2();
        d=i2c_data>>3;
        if (signo==1)
        {
            d=(~d+1)*-1;
        }
        f=(d*0.0625);
        return(f);
    }
    else
      return(-55);    // Mandar un valor erroneo de temperatura muy baja(-55C).
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void i2c_startp2(void)                  // Función de inicio del protocolo I2C.
{
    manda_data(0);
    retardo(1);
    manda_clock(0);
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void i2c_stopp2(void)                   // Función de inicio del protocolo I2C.
{
    manda_data(0);
    retardo(1);
    manda_clock(1);
    retardo(1);
    manda_data(1);
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void manda_data(char p)                 // Poner un dato en la línea SDA.
{
    switch(bus)
    {
         case 0:{P0_7=p&0x01; break;}
         case 1:{P0_5=p&0x01; break;}
         case 2:{P0_3=p&0x01; break;}
         case 3:{P0_1=p&0x01; break;}
    }
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void manda_clock(char p)                // Poner un dato en la línea SCL.
{
    switch(bus)
    {
         case 0: {P0_6=p&0x01; break;}
         case 1: {P0_4=p&0x01; break;}
         case 2: {P0_2=p&0x01; break;}
         case 3: {P0_0=p&0x01; break;}
    }
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
unsigned char lee_data(void)             // Leer un dato en la línea SDA.

{
    unsigned char p;

    switch(bus)
    {
         case 0:{p=P0_7; break;}
         case 1:{p=P0_5; break;}
         case 2:{p=P0_3; break;}
         case 3:{p=P0_1; break;}
    }
    return(p&0x01);
}
```

Apendice B.
Programa de la PC.

```
#include <sys/types.h>
#include <stdio.h>
#include <sys/time.h>
#include <sys/ioctl.h>
#include <unistd.h>
#include <string.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <time.h>
#include <stdlib.h>
#include <math.h>


#define BAUDRATE B9600
#define MaxBuf 500

struct termios oldtio, newtio;
struct tm *hora_fecha;

time_t reloj;

char  Buffer[MaxBuf][8], Linea[MaxBuf], Nombre[MaxBuf];
char  COM[30], Telescopio[10], error[81];
char  car, path[] = "/home/tlaloc/estacion/";
float y[48], x[48], a[24][4][8];
int   i, j, k, puerto, readerr, Mediciones, Cuentas;
int   horas, minutos, segundos, min_ini, min_fin, seg_ini, seg_fin;
int   NumModulos, Modulo[8];

FILE *Archivo;

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void LeeConfiguracion()           // Lee la configuracion del hardware.
{
    sprintf(Nombre,"%sprograma/estacion.cfg",path);

    Archivo=fopen(Nombre,"r");

    if(Archivo == NULL)
    {
        printf("%s\n", Nombre);
        perror(error);
        exit(-1);
    }

    fscanf(Archivo, "%s %s \n", Telescopio, Linea);
    printf("%s %s\n", Telescopio, Linea);
    fscanf(Archivo, "%s %s \n", COM, Linea);
    printf("%s %s\n", COM, Linea);
    fscanf(Archivo, "%i %s\n", &NumModulos, Linea);
    printf("%d %s\n",NumModulos, Linea);
    for(i=0; i<NumModulos; i++)
    {
        fscanf(Archivo, "%i %s \n", &Modulo[i], Linea);
        printf("%d %s\n", Modulo[i], Linea);
    }
    fclose(Archivo);
    printf("--- %s leido.\n\n", Nombre);
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void LeeCalibraciones()                    // Lee archivos calibracion.
{
    for(i=0;i<NumModulos;i++)
    {
        sprintf(Nombre, "%sprograma/Offsets%d.dat", path, Modulo[i]);

        Archivo=fopen(Nombre,"r");
        if(Archivo == NULL)
        {
            printf("%s\n", Nombre);
            perror(error);
            exit(-1);
        }
        for(j=0; j<24; j++)
        {   fscanf(Archivo, "%f %f %f %f %s \n", &a[j][0][i],
                    &a[j][1][i], &a[j][2][i], &a[j][3][i], Linea);

        printf("%f %f %f %f %s \n", a[j][0][i], a[j][1][i],
               a[j][2][i], a[j][3][i], Linea);
        }
        fclose (Archivo);
        printf("--- %s leido.\n\n", Nombre);
    }
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void AbrePuerto()                    // Inicializa el puerto serie.
{
// r/w sin esperar a que llegue algo.
    puerto = open(COM, O_RDWR | O_NOCTTY);
    if (puerto<0) { perror(COM); exit (-1); }

// Guardar condiciones de puerto actuales.
    tcgetattr (puerto, &oldtio);
    bzero (&newtio, sizeof(newtio));

// Baudrate=9600, CS8=8n1(8 bits, sin paridad y 1 stop bit)
// CLOCAL=conexionn local, sin control de modem
// CREAD=habilitar caract. recepcion.
    newtio.c_cflag = BAUDRATE | CS8 | CLOCAL | CREAD;

// IGNPAR= Ignorar bytes con error de paridad.
    newtio.c_iflag = IGNPAR;

// Salida sin formato.
    newtio.c_oflag = 0;
    newtio.c_lflag = 0;
    newtio.c_cc[VTIME] = 0x00;
```

```
    newtio.c_cc[VMIN] = 0x00;
    tcflush (puerto, TCIFLUSH);
    tcsetattr (puerto, TCSANOW, &newtio);

    printf("--- Puerto %s abierto. \n\n", COM);
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
int SoloDigitos(char h)
{
    if(h=='0'||h=='1'||h=='2'||h=='3'||h=='4'||h=='5'||h=='6'||
       h=='7'||h=='8'||h=='9'||h=='.'||h=='-'||h==' '||h=='\n') return (1);
    else return (0);
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void LeeModulos()
{
    int rflag, resp;

    for(i=0; i<NumModulos; i++)
    {
        sprintf(Linea,":2M%dV;",Modulo[i]);
        write(puerto, &Linea, 6);

        time(&reloj);
        hora_fecha = localtime(&reloj);
        seg_ini = hora_fecha->tm_sec;

        j=0;
        Buffer[100][i] = '\0';
        do
        {
            rflag = read(puerto, &car, 1);
            resp = SoloDigitos(car);
            if(rflag>0 && resp>0) Buffer[j++][i] = car;
            time(&reloj);
            hora_fecha = localtime(&reloj);
            seg_fin = hora_fecha->tm_sec;
        } while (car != '\n' && j < (MaxBuf-2) && (seg_fin-seg_ini) < 3);
        usleep(30000);          // Necesario, De otra manera se atora.
    }
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void EsperaUnMinuto()
{

    time (&reloj);
    hora_fecha = localtime (&reloj);
    min_ini = hora_fecha->tm_min;
    sleep(50);

    do
    {
        usleep(100000);
        time (&reloj);
        hora_fecha = localtime (&reloj);
        min_fin = hora_fecha->tm_min;
    } while (min_fin == min_ini);
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void Convierte_a_Numeros()          // Extrae valores en el buffer.
{
int DatosConvertidos;

    DatosConvertidos = sscanf (Linea,
            "%d %d \
             %f %f %f %f %f %f %f %f \
             %f %f %f %f %f %f %f %f \
             %f %f %f %f %f %f %f %f \
             %f %f %f %f %f %f %f %f \
             %f %f %f %f %f %f %f %f \
             %f %f %f %f %f %f %f %f ",
            &Mediciones,&Cuentas,
            &x[ 0],&x[ 1],&x[ 2],&x[ 3],&x[ 4],&x[ 5],&x[ 6],&x[ 7],
            &x[ 8],&x[ 9],&x[10],&x[11],&x[12],&x[13],&x[14],&x[15],
            &x[16],&x[17],&x[18],&x[19],&x[20],&x[21],&x[22],&x[23],
            &x[24],&x[25],&x[26],&x[27],&x[28],&x[29],&x[30],&x[31],
            &x[32],&x[33],&x[34],&x[35],&x[36],&x[37],&x[38],&x[39],
            &x[40],&x[41],&x[42],&x[43],&x[44],&x[45],&x[46],&x[47]);

    if(DatosConvertidos !=50 )
       {
          Mediciones = 0; Cuentas=0;
           for(i=0;i<48;i++) x[i] = 0.0;
       }
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void Calibra()                          // Convierte a unidades reales.
{
    for(j=0;j<24;j++) y[j] = a[j][0][k] + a[j][1][k]*x[j] +
                  a[j][2][k]*x[j]*x[j] + a[j][3][k]*x[j]*x[j]*x[j];

    for(j=24;j<48;j++) y[j] = a[j-24][1][k]*x[j];
}

//*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
void ArchivaDatos()
{
    sprintf(Nombre, "%sdatos/%d%02d%02d%d_%s.dat",
            path,
            hora_fecha->tm_year+1900,
            hora_fecha->tm_mon+1,
            hora_fecha->tm_mday,
            Modulo[k],
            Telescopio);

    Archivo=fopen(Nombre, "a");
// Si NO logra abrirlo, tratar varias veces.
    if (Archivo == NULL) { perror(error); usleep(500000);
Archivo=fopen(Nombre, "a");}
    if (Archivo == NULL) { perror(error); usleep(500000);
Archivo=fopen(Nombre, "a");}
// Si logro abrirlo, escribir datos, si NO continuar.
    if (Archivo != NULL)
```

```
    {
//Hora, mediciones y cuentas
    fprintf(Archivo, "%02d %02d %d %d ",horas,minutos,Mediciones,Cuentas);
//Analogicos
    fprintf(Archivo, "%1.3f %1.3f %1.3f %1.3f %1.3f %1.3f %1.3f %1.3f ",
            y[ 0], y[ 1], y[ 2], y[ 3], y[ 4], y[ 5], y[ 6], y[ 7]);
//Termometros
    fprintf(Archivo, "%1.3f %1.3f %1.3f %1.3f %1.3f %1.3f %1.3f %1.3f ",
            y[ 8], y[ 9], y[10], y[11], y[12], y[13], y[14], y[15]);
    fprintf(Archivo, "%1.3f %1.3f %1.3f %1.3f %1.3f %1.3f %1.3f %1.3f ",
            y[16], y[17], y[18], y[19], y[20], y[21], y[22], y[23]);
//Sigmas Analogicos
    fprintf(Archivo, "%1.3f %1.3f %1.3f %1.3f %1.3f %1.3f %1.3f %1.3f ",
            y[24], y[25], y[26], y[27], y[28], y[29], y[30], y[31]);
//Sigmas Termometros
    fprintf(Archivo, "%1.3f %1.3f %1.3f %1.3f %1.3f %1.3f %1.3f %1.3f ",
            y[32], y[33], y[34], y[35], y[36], y[37], y[38], y[39]);
    fprintf(Archivo, "%1.3f %1.3f %1.3f %1.3f %1.3f %1.3f %1.3f %1.3f \n",
            y[40], y[41], y[42], y[43], y[44], y[45], y[46], y[47]);
    fflush(Archivo);
    fclose(Archivo);
    }
}

//-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*- PRINCIPAL -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
int main (void)
{
    printf("\nEstacion meteorologica SPM V3.0 (mayo 2005)\n");

    LeeConfiguracion();
    LeeCalibraciones();
    AbrePuerto();
    LeeModulos();                                          // Resetearlos.
    printf("\n Corriendo ...\n\n");

    do                        // Ciclo infinito de adquisicion de datos.
    {
         EsperaUnMinuto();
         horas   = hora_fecha->tm_hour;
         minutos = hora_fecha->tm_min;
        segundos = hora_fecha->tm_sec;

        LeeModulos();
        for (k=0; k<NumModulos; k++)
        {
            if (Buffer[100][k] != '\0')
            {
                for (j=0; j<MaxBuf; j++) Linea[j] = Buffer[j][k];
                Convierte_a_Numeros();
                Calibra();
                ArchivaDatos();
                readerr=system("/home/tlaloc/estacion/programa/plot2m");
            }
        }
    } while(1);
    return 0;
```