UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

Diseño e Integración de una estación remota para fines de telemetría

T E S I S

QUE PARA OPTAR POR EL GRADO DE :

M A E S T R O E N I N G E N I E R Í A

Á R E A : E L É C T R I C A
O P C I Ó N : I N S T R U M E N T A C I Ó N

P R E S E N T A :

MARIO ALBERTO MENDOZA BÁRCENAS

TUTOR Y DIRECTOR DE TESIS: M. en I. LAURO SANTIAGO CRUZ.

Ciudad Universitaria, D.F. 2008

Jurado asignado

Presidente: Dr. Esau Vicente Vivas

Secretario: Dr. Felipe Orduña Bustamante

Vocal: M.I. Lauro Santiago Cruz

1° suplente: M.I. Antonio Salva Calleja

2° suplente: M.I. Antonio Pérez López

LUGAR DONDE SE REALIZÓ LA TESIS:

Instituto de Ingeniería, U.N.A.M.

TUTOR DE TESIS:

M.I. Lauro Santiago Cruz

_____

Firma

Dedicatorias

Mi trabajo de tesis lo dedico a ti: Dios
por haberme ayudado a conseguir esta meta,por
darme claridad en mis ideas
cuando más lo necesitaba,
por estar siempre junto a mi...

A mis padres: Rufina Bárcenas y Mario Mendoza,
por todo su gran apoyo, por estar conmigo en todos
los momentos, por creer en mi, por su comprensión y amor.
A ustedes les debo todo lo que soy... GRACIAS.

A mis hermanos: Marianita y Carlos.
Gracias por estar conmigo y apoyarme siempre.

A mi abuelita Concepción Mendoza,
por tu gran, gran apoyo, amor y comprensión.
Te quiero mucho...

In memoriam:
A mi abuelita Trinidad Morales Ramírez

Desde donde estés, a tí te debo haber llegado
hasta aquí. Gracias por tus consejos y enseñanzas.Sie
mpre los llevo y llevaré presentes...

In memoriam:
A mi abuelito Roberto Alejo González Enriquez

Desde donde estés, te dedico este trabajo.
Gracias por todos los excelentes momentos que
pasamos juntos, por tus consejos y enseñanzas.
¡Siempre te recordaré!

Gracias a mi familia: mis tíos Gretell y Luis, Laura y Felipe, Salvadora y Javier, a mis primos, Lulú, Aline y Aldo y a mi sobrinita "Chispa".

Gracias a todos ustedes por su apoyo, por su cariño y por estar siempre a mi lado. Sin rencores.

Un verdadero amigo es alguien que te conoce tal como eres, comprende dónde has estado, te acompaña en tus logros y tus fracasos, celebra tus alegrías, comparte tu dolor y jamás te juzga por tus errores...

Un amigo verdadero es alguien que cree en ti aunque tu hayas dejado de creer en ti mismo...

Gracias Ana Laura Padilla Ortiz...

Gracias a mi tutor y director de tesis: M.I. Lauro Santiago Cruz.

Gracias jefe por su paciencia, apoyo, confianza y amistad .

Gracias al M.I. Israel Nava Bravo, por su enorme apoyo en la etapa inicial de este proyecto.

Gracias por su gran apoyo y amistad a los compañeros del laboratorio de Instrumentación: Angélica, Anuar y David.

Gracias al Dr. Esaú Vicente, Dr. Felipe Orduña, M.I. Antonio Pérez y

M.I. Antonio Salva, por sus comentarios, los cuales enriquecieron este trabajo.

Agradezco a la Dirección General de Estudios de Posgrado (D.G.E.P.) de la U.N.A.M. por la beca otorgada durante mis estudios de maestría.

Agradezco al Ing. Javier Vázquez Hernández, por su enorme y desinteresado apoyo y asesoría en gran parte de este proyecto.

Agradezco al CONALEP, Plantel Xochimilco, y en especial a mi jefe, Ing. Raúl Flores Pablo, por su gigantesco apoyo y comprensión.

Agradezco enormemente el apoyo de:

Ing. Francisco Flores Mejía

Ing. Juan Olalde Fuentes

Lic. Moisés Espinoza

Ing. José Luis Carrasco Martínez

De la Red de Estaciones Meteorológicas Automáticas del Servicio Meteorológico Nacional, por su asesoría sobre estaciones meteorológicas y sensores.

Eternamente agradecido con la UNAM, por haberme permitido una vez más, ser parte de su comunidad y brindarme los medios para seguir creciendo como profesionista y como persona.

Capítulo 1

ANTECEDENTES

En el presente capítulo se plantean conceptos generales sobre sistemas de telemetría, se describe la problemática a resolver y se plantea la propuesta de solución a la misma.

Capítulo 2

DESCRIPCIÓN GENERAL DE LA ESTACIÓN REMOTA

En este capítulo se presenta una descripción general de cada uno de los componentes que conformarán a la estación, como son: el microcontrolador, la unidad de memoria, unidad de alimentación de energía y la unidad de comunicación de datos, así como las interfaces de comunicación asociadas.

Capítulo 3

DISEÑO E INTEGRACIÓN DEL HARDWARE DEL SISTEMA

En este capítulo se presentan los detalles de cada uno de los elementos que integran el hardware de la estación remota, así como la justificación de su incorporación dentro de la simbiosis tecnológica del equipo desarrollado.

Capítulo 4

DISEÑO E INTEGRACIÓN DEL SOFTWARE DEL SISTEMA

En este capítulo se presentan los detalles de cada uno de los elementos que componen el software de la estación remota y de la estación base.

Capítulo 5

INTEGRACIÓN Y PRUEBAS AL SISTEMA

En este capítulo se presenta la integración de los elementos de hardware, considerando los circuitos impresos de las tarjetas principal y auxiliar. También se presenta una actualización de los ajustes, técnicos y conceptuales, realizados al sistema y una evaluación del equipo desarrollado.

Capítulo 6

RESULTADOS Y CONCLUSIONES

En este capítulo se presentan los resultados y consideraciones obtenidas del proyecto completo, una vez que se ha finalizado el desarrollo de la estación remota para fines de telemetría. En la última parte de este capítulo se mencionan algunas recomendaciones que pueden ser tomadas en cuenta, con objeto de mejorar la operación del sistema desarrollado.


DISEÑO E INTEGRACIÓN
DE UNA ESTACIÓN REMOTA
PARA FINES DE TELEMETRIA

Apéndices


DISEÑO E INTEGRACIÓN
DE UNA ESTACIÓN REMOTA
PARA FINES DE TELEMETRIA

Glosario

Bibliografía

Índice temático

Índice de figuras

CAPÍTULO 1

CAPÍTULO 2

CAPÍTULO 3

Índice de tablas

PRÓLOGO

El constante monitoreo de fenómenos tanto físicos como químicos es muy importante, ya que nos permite conocer su comportamiento actual y con base en estas observaciones estimar su comportamiento futuro. Muchos de estos fenómenos ocurren en lugares muy lejanos y/o de difícil acceso, bajo condiciones en las que la recopilación de la información obtenida de las observaciones para su análisis en sitio se vuelve complicada. Muchos de estos eventos tienen que ver con fenómenos naturales, tales como terremotos, lluvias, huracanes, nevadas, etcétera, sobre los cuales el hombre no tiene control y que pueden ocurrir en cualquier momento y, otros tantos, que son causados sistemáticamente por la mano del hombre; sin embargo, independientemente del origen de los eventos, en muchos de ellos, se requiere una vigilancia intensa para evitar, en la medida de lo posible, que se salgan de control, y causen daños en personas o en materiales.

Dentro de la naturaleza, el clima ha sido una de los factores determinantes en el desarrollo de las sociedades humanas. Desde que el hombre dejó de ser nómada para convertirse en sedentario, el factor clima ha sido vigilado y considerado para el florecimiento de culturas que se caracterizaron por su desarrollo en torno a los cultivos agrícolas y crianza de animales. En un principio, culturas tan esplendorosas como la egipcia y la babilonia, tomaban como base fenómenos como el cambio en el nivel de los ríos o el aspecto del cielo para realizar sus pronósticos con respecto al clima. Con el advenimiento de la ciencia y el desarrollo de los primeros instrumentos de medición, como el termómetro de mercurio por Galileo en 1607, o el barómetro en 1643 por Torricelli, los pronósticos respecto al clima, comenzaron a tomar un matiz científico.

Con el avance de la ciencia, el desarrollo de nueva instrumentación, el nacimiento de la electrónica y de las telecomunicaciones, la rama de la geofísica, llamada meteorología, comenzó a adquirir un mayor número de herramientas, las cuales permitirían a los especialistas en el tema una mayor comprensión de los fenómenos climatológicos, además de una mayor monitoreo de los mismos, a partir de la construcción de estaciones meteorológicas automáticas, operadas a partir de elementos electrónicos y microprocesados. Con la inclusión de tales herramientas tecnológicas, la meteorología se ha convertido en una gran herramienta científica que permite al hombre un mayor conocimiento de su

entorno, lo cual redunda en un mejor aprovechamiento de zonas de desarrollo agrícola, comercial y ganadero a nivel mundial, además de proporcionar pronósticos del clima cada vez más certeros, los cuales pueden prevenir la pérdida de vidas humanas ante eventos potencialmente catastróficos como los huracanes, ciclones, lluvias fuertes, entre otros.

En este contexto, la empresa mexicana "IROSA", dedicada inicialmente a la comercialización, distribución y también al desarrollo de equipo e instrumentación relacionada con los campos de la meteorología e hidrometeorología, instrumentación industrial y convencional y topografía, requirió al Instituto de Ingeniería (I.I.) de la Universidad Nacional Autónoma de México (U.N.A.M.), el diseño y construcción de una estación remota automática de diseño propio, bajo costo, con opción de comunicación de datos por medios tanto alámbricos como inalámbricos de corto y de largo alcance, y de fabricación mexicana, que en un futuro permita servir como opción comercial en el mercado nacional. Como respuesta al requerimiento mencionado, en este trabajo de tesis se describe el diseño e integración de una estación remota para fines de telemetría, con aplicación en el área meteorológica; aunque el enfoque global del proyecto es una aplicación genérica que permita, a través de diferentes medios de comunicación, enlazar uno o más puntos con una estación central.

Así mismo, el concepto integral bajo el que se concibe a la estación remota, plantea no sólo el desarrollo a nivel hardware, sino también, a nivel software de un sistema completo de adquisición y envío de datos.

INTRODUCCIÓN

En el presente trabajo de tesis se describe el diseño y desarrollo de una estación remota, con aplicación a la meteorología, que tiene como características principales ser un equipo de operación e instalación sencilla, mantenimiento mínimo y concebido a partir de elementos de fácil adquisición en el mercado mexicano, costo razonable y de alto grado de integración. El esquema general que plantea el proyecto es el diseño y construcción de una estación remota para fines de telemetría, la cual estará conformada por un dispositivo central de gestión y procesamiento de datos, implementado sobre la plataforma de un microcontrolador RISC de 8 bits, de altas prestaciones y de bajo costo, así como por 4 subsistemas periféricos: de alimentación de energía por red convencional y por batería, ésta última recargada por medio de celdas solares; almacenamiento externo de datos en memoria EEPROM, que permitirá guardar aproximadamente 4MB de información; reloj en tiempo real para la programación de eventos de adquisición de datos y referencia para los datos adquiridos y protocolos de comunicación de datos asociados de media y alta velocidad.

La finalidad que tiene el desarrollo del proyecto que describiré en este trabajo, es el diseño y construcción de un prototipo de adquisición, procesamiento y transmisión de datos, que permita la medición a distancia de variables, físicas y/o químicas, el cual pueda ser utilizado en diversos ámbitos de investigación, con sólo algunas adaptaciones.

El trabajo escrito de la tesis está estructurado de la siguiente manera: 6 capítulos, la bibliografía los apéndices y el glosario de términos, cuyas características principales son:

En el capítulo 1 se hace una referencia general a todos los tópicos estrechamente vinculados con el equipo desarrollado, sobre todo en lo relacionado con tecnologías de comunicación e intercambio de datos.

En el capítulo 2 se realiza una descripción detallada de cada una de las unidades que componen al *hardware* del sistema, dentro de un esquema general.

En el capítulo 3 se presenta el proceso, pormenorizado, asociado con la selección y configuración de cada uno de los componentes del *hardware* que integra la estación remota.

En el capítulo 4 se presenta el desarrollo e integración del software, tanto para la estación remota como para la estación base.

En el capítulo 5 se describe la vinculación de los elementos del hardware de la estación remota, a través del diseño y desarrollo de las tarjetas de circuito impreso, para, con base en éstas, concebir un esquema terminado del prototipo y proceder a la realización de pruebas de evaluación del mismo.

En el capítulo 6 se presentan los resultados y conclusiones obtenidos del proyecto que lleven a la generación de observaciones y correcciones que tomar en cuenta para versiones futuras del desarrollo de este proyecto, junto con una serie de recomendaciones para futuros desarrollos o mejoras sobre la base del equipo presentado.

En la sección de bibliografía se listan las diferentes fuentes (libros, manuales, páginas de Internet, etc.) consultados durante el desarrollo de este trabajo de tesis.

Adicionalmente se presenta una sección de apéndices, en la cual se podrán consultar extractos de los códigos fuente utilizados y un glosario de términos, en el cual se concentran las definiciones de términos que a lo largo del trabajo se indican en itálicas.

1.1. Introducción a los Sistemas de Telemetría

La Comisión Federal de Comunicaciones de los Estados Unidos (FCC) define a la *telemetría* como *"el uso de las telecomunicaciones para el monitoreo automático o la grabación de mediciones a distancia desde un instrumento de medición remoto".* Se puede extender una parte de esta definición para decir que, los sistemas de telemetría son los medios por los cuales los datos son reunidos en una ubicación "remota" y luego transportados hacia un usuario en una ubicación "base". A la información que se transporta se le refiere como "Telemetría". El proceso por el cual los datos son reunidos es llamado "telemedición".

Los procesos de medición pueden tener una actividad complementaria de control, por medio de comandos que son enviados hacia el sistema de la ubicación remota. La FCC define además "comando" como: "*el uso de telecomunicaciones para la transmisión de señales para iniciar, modificar ó finalizar funciones del equipo a distancia".* La actividad de comando puede ser utilizada para configurar ó realizar mediciones y determinar las funciones realizadas.

Muchas personas han visto el valor de los sistemas de telemetría en ejemplos tan monumentales como las misiones espaciales. Por citar algún ejemplo, las impresionantes fotografías de los anillos de Saturno, obtenidas a través de sistemas telemétricos diseñados para ser flexibles a través de ambientes muy peligrosos por la nave *Voyager 2* en 1981 (figura 1.2); las cuales fueron obtenidas a través de sistemas de comando y telemetría diseñados para ser flexibles a través de ambientes muy peligrosos.

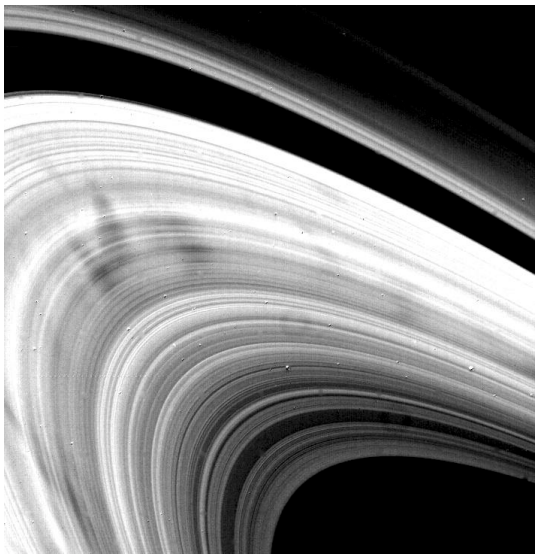


Fig. 1.1. Fotografías obtenidas por la sonda Voyager 2.

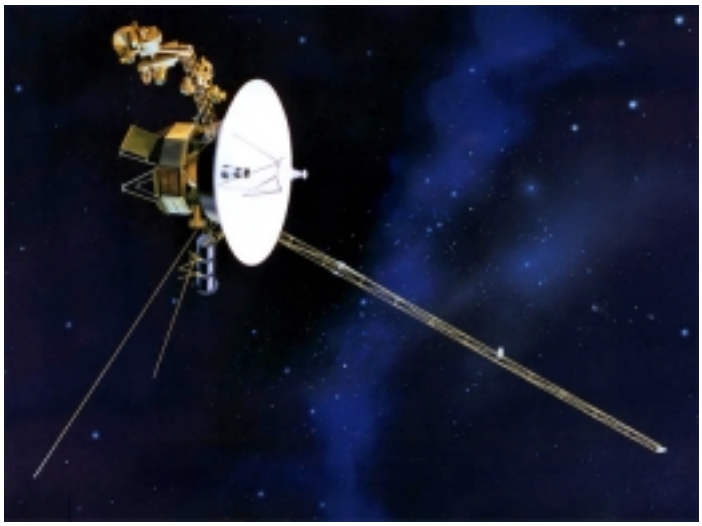


Figura 1.2. Sonda espacial Voyager 2.

Más sutil es la influencia de los telecomandos y los sistemas telemétricos en el control en tiempo real de los satélites internacionales de comunicaciones que permiten comunicaciones de voz y datos todos los días, ejemplos de ello se muestran en la figura 1.3.

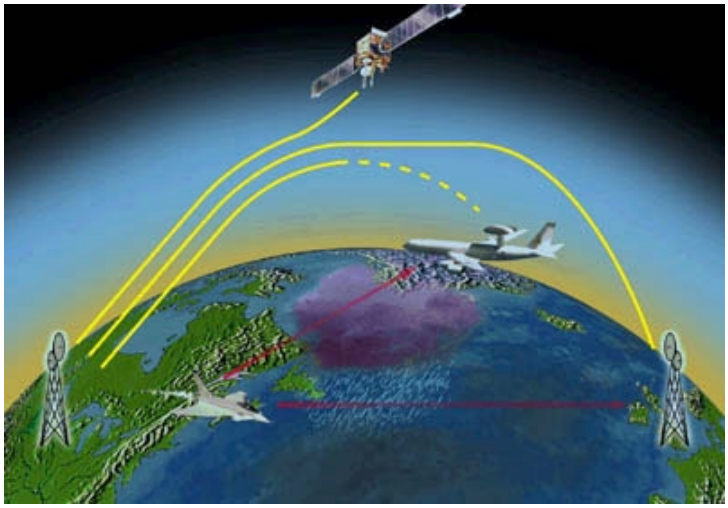


Figura 1.3. Transmisión de señal vía satélite y mediante rebote ionosférico.

A un nivel más terrenal, se encuentran los sistemas de telemetría que monitorean los signos vitales de una persona enferma en un hospital, retransmitiendo los datos hacia una estación central de monitoreo, como se muestra en la figura 1.4.

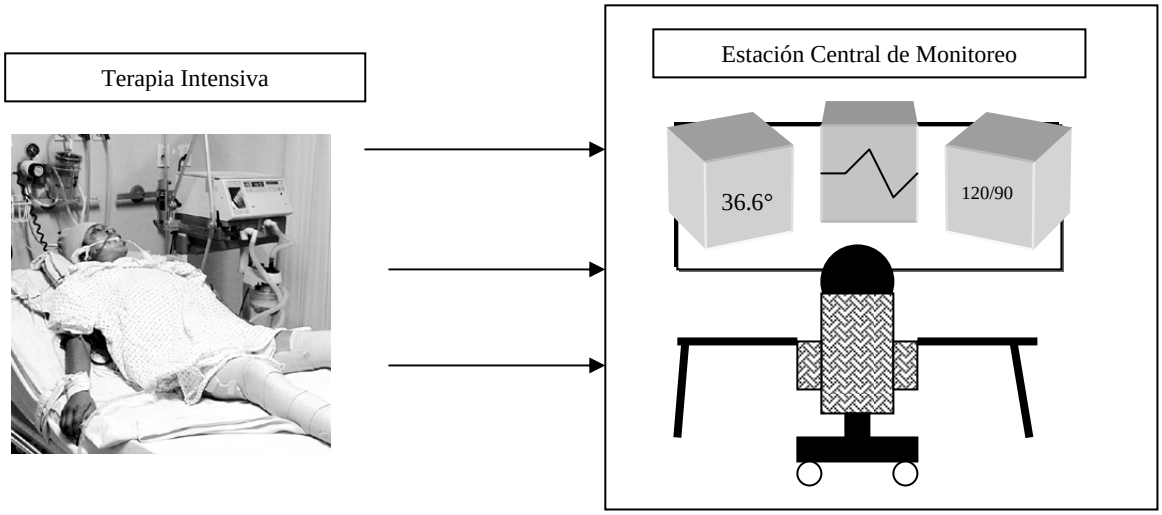


Figura 1.4. Monitoreo de signos vitales en enfermos.

La Red Digital de Servicios Integrados (ISDN), la cual tiene su primer referente en la recomendación I.120 de 1984 del entonces denominado Comité Consultivo Internacional de Teléfonos y Telégrafos (CCITT), hoy conocida como Unión Internacional de Telecomunicaciones (ITU), y con mucho mayor auge en la década de los 90's, es considerada hoy, por algunos especialistas como una tecnología relativamente antigua no así obsoleta; sólo desplazada por los actuales servicios de Internet de banda ancha. En la figura 1.5 se muestra un esquema de la ISDN, que posee dos canales de telemetría asociados, que le permiten tener el control sobre lo que parece ser un estándar de servicios telefónicos.

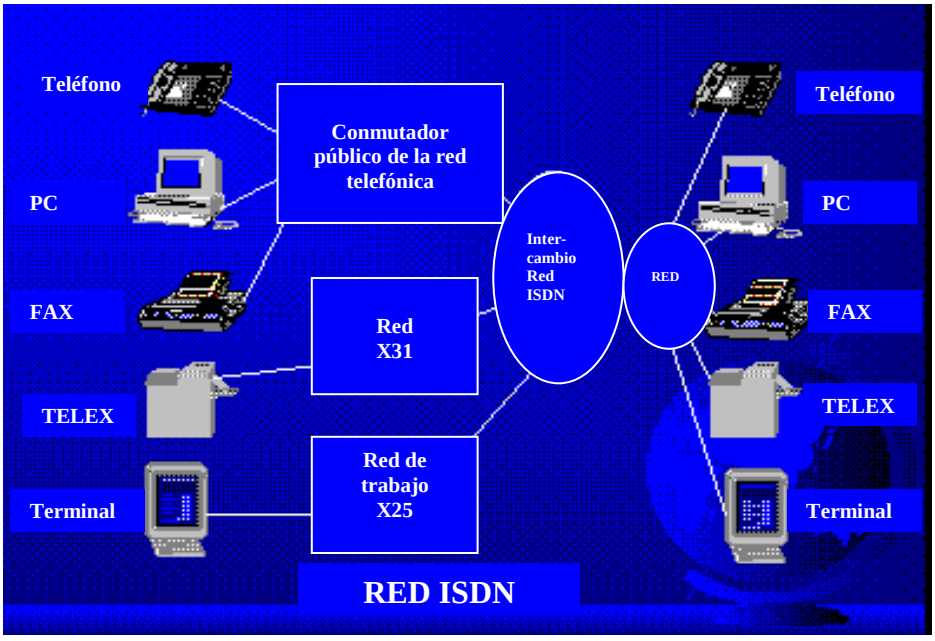


Figura 1.5. Elementos de una red ISDN.

De lo comentado anteriormente podemos concluir que los sistemas de telemetría pueden ser encontrados en muchas facetas de la vida diaria.

Un sistema telemétrico genérico se muestra en la figura 1.6. Este diagrama de bloques tiene mucho en común con un sistema generalizado de comunicaciones. Esta concordancia no debe sorprender, debido a que un sistema de telemetría es realmente un sistema de comunicaciones especializado. La estructura generalizada del sistema mostrada en la figura puede ser usada en la discusión de los componentes de un sistema de telemetría.

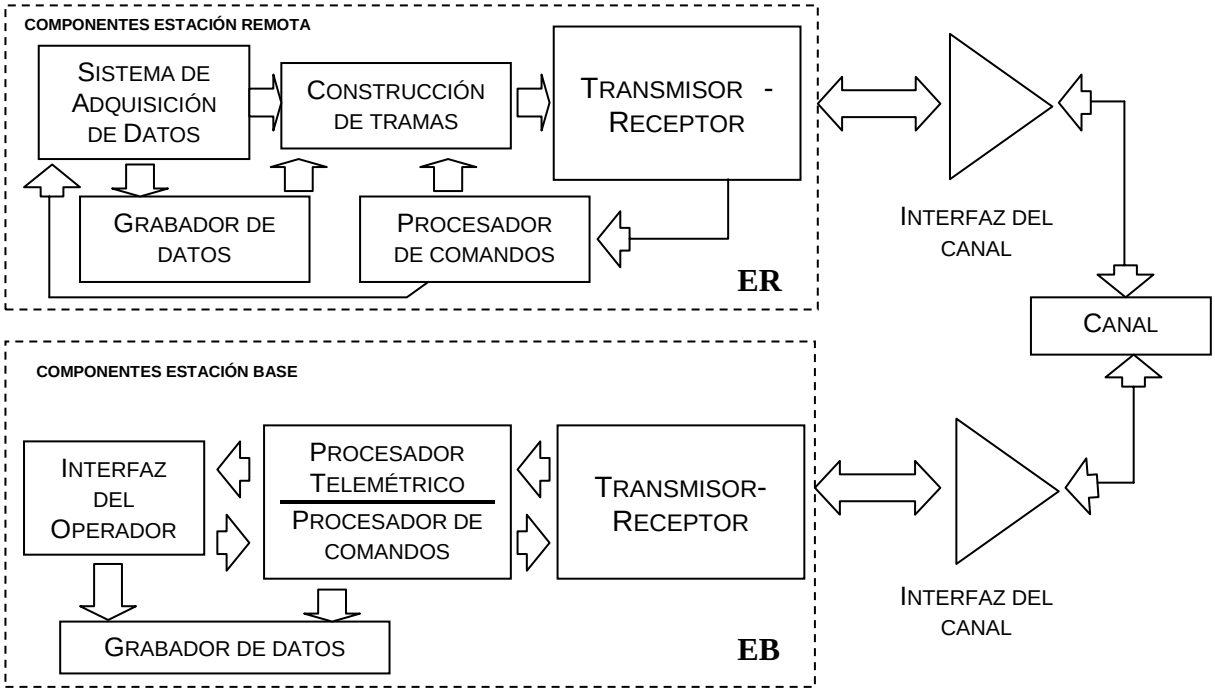


Figura 1.6. Un sistema generalizado de telemetría y comando.

Los componentes del sistema de telemetría y comando están segmentados dentro de dos sistemas mayores: el segmento ESTACIÓN BASE (EB) y el segmento ESTACIÓN REMOTA (ER). Esta es justamente una designación para indicar la relativa interrelación entre los componentes. Esto no significa que la ER siempre debe ser equipo llevado a un lugar lejano, mientras que la EB deberá ser una compleja instalación (aunque a veces, puede darse el caso). La ER puede ser un estante que contenga instrumentación en una parte de una habitación, con la EB siendo una computadora de escritorio en otra parte de la misma habitación, siendo el canal de comunicación un cable coaxial enlazando ambos lados.

El segmento EB se ocupa sobre todo de las operaciones con la ER y de las funciones de procesamiento de datos. La EB tiene como sus mayores subsistemas la interfaz del operador, el procesamiento de datos, el hardware para transmisión de datos y la interfaz hacia el canal de transmisión. Las funciones generales de los subsistemas más importantes mostrados en la figura 1.6 son las siguientes:

ER

1. ***Sistema de adquisición de datos.*** Un Sistema de Adquisición de datos no es más que un equipo electrónico cuya función es el control o simplemente el registro de una ó varias variables de un proceso cualquiera, de manera general puede estar compuesto por elementos como: sensores, amplificadores, multiplexores, convertidores A/D y D/A, microprocesadores, filtros, actuadores, etcétera.

EB

2. ***Interfaz del operador.*** Despliega los datos y el operador interactúa con las entradas del sistema.
3. **Grabador de datos.** Dispositivo para grabar los datos recibidos así como cualquier entrada del operador. Frecuentemente hay dos tipos de grabadores: uno que permite grabar los datos "en bruto" antes de procesarlos y otros que permiten grabar los datos que ya han sido procesados.
4. **Procesador telemétrico/Procesador de comando.** Procesa las entradas "en bruto" dentro de un formato entendible para el operador o para la estación remota.
5. **Transmisor/receptor.** Dispositivo que sirve de interfaz con el canal de transmisión y recepción; puede ser una antena, fibra óptica, cable coaxial, etcétera.

Cabe comentar hasta este punto, que dado el gran avance que ha tenido la microelectrónica hasta la fecha, varios bloques tanto del módulo de la ER como de la EB ya se pueden encontrar dentro de la estructura, como subsistemas periféricos, de microprocesadores de altas prestaciones.

El **canal** es el medio de transmisión entre la ER y la EB. El canal puede ser caracterizado por como afecta a las señales transmitidas. Algunos parámetros característicos del canal de transmisión son:

1. Ruido característico (Ruido blanco ó ruido en banda angosta).

2. Características de degradación de señal (rutas múltiples ó efectos atmosféricos)
3. Interferencias. Se conoce como interferencias o *jamming* a aquellas señales externas al sistema de comunicación y que por lo tanto enmascara a los objetivos de interés. Un ejemplo tiene que ver con la reciente proliferación de los sistemas WiFi que operan en la banda C (alrededor de los 5,66 GHz) y que se han convertido en un verdadero problema para radares meteorológicos, que precisamente sufren interferencias.
4. Pérdidas de potencia en la señal debido a diversos factores climáticos, como las lluvias, el viento, etcétera.

El segmento ER se encarga sobre todo de la adquisición de datos de los sensores que miden las características del ambiente y las características operativas de la propia estación remota. Usualmente, sólo se pueden encontrar presentes funciones limitadas de procesamiento de datos en la estación remota. El segmento ER tiene como sus mayores subsistemas a los siguientes: sistema sensor, el acondicionamiento electrónico de la señal, la colección de datos de banda base, el hardware de procesamiento de señal, el hardware para la transmisión de datos y la interfaz hacia el canal de transmisión. Las principales funciones de algunos de estos subsistemas son:

1. **Sensores.** Son varios tipos de dispositivos que producen señales electrónicas de salida, las cuales pueden ser proporcionales a la medición deseada.
2. **Procesador de comandos.** Es un intérprete cuya función es la de recibir, procesar y ejecutar los comandos enviados desde la EB.
3. **Constructor de tramas.** Dispositivo que permite construir grupos de datos que constituyen la información enviada desde la ER hacia la EB a través de un flujo serial de datos telemétricos.
4. **Transmisor/receptor.** Dispositivo que transforma o codifica los mensajes para su transmisión/recepción adecuada. El medio de transmisión, por su naturaleza física, es posible que modifique o degrade la señal en su trayecto desde el Transmisor hacia el Receptor. Por ello, el receptor ha de tener un mecanismo de decodificación capaz de recuperar el mensaje dentro de ciertos límites de degradación de la señal.

Algunos de estos módulos, como ya se había mencionado, en la actualidad los podemos encontrar reunidos en la arquitectura interna de un microprocesador. Por lo que el grado de integración de los sistemas actuales tiende a ser muy alto.

1.1.1. Tipos y fuentes de datos

En general, existen tres tipos principales de datos que pueden ser transmitidos: voz, video y datos. El último es casi exclusivamente información de computadoras. Aunque las señales de voz y video son analógicas en su naturaleza, la tecnología actual digitaliza virtualmente cualquier señal de la fuente. Una vez que este proceso ha ocurrido, cualquier señal, independientemente del tipo de información representada, puede ser considerada como dato. Como ejemplo se puede mencionar que las señales de video las podemos encontrar en: la difusión de señales de televisión (TV), la TV por cable, el satélite y los sistemas de vigilancia. Históricamente y aún actualmente, muchos sistemas de transmisión continúan manejando señales de video en forma analógica, debido a la tecnología que para su procesamiento se ha desarrollado. La operación analógica continúa en algunos casos debido a que proporciona una adecuada calidad a un precio muy atractivo, por ejemplo en sistemas de seguridad y monitoreo.

Aunque se han desarrollado diferentes tipos de redes, se ha incrementado la presión para el establecimiento de normas para la transmisión de datos, que regulen la operación de las mismas a tasas ó velocidades establecidas. Las velocidades comúnmente utilizadas son 1200, 2400, 4800, 9600 y 19200 *bps* ó múltiplos y submúltiplos de 64 kbps. En la actualidad, un mayor intervalo de velocidades de 2, 8, 34, 140 o 565 Mbps es muy utilizado en los vínculos de comunicaciones de transporte largo "*long-haul*" de las redes WAN por correo, operadores telefónicos y de telecomunicaciones. Algunos ejemplos de velocidades o tasas de transmisión de datos se muestran en la tabla no.1

Velocidades típicas de los accesos de conexión a Internet (Abril, 2006)	
Dispositivo de conexión	**Velocidad**
Módem RTB	56 kbps = 7kBps
Módem ADSL	1024 kbps (nominal 1 Mbps) = 128 kBps
Cable	2400 kbps = 300 kBps
VSAT	600 kbps = 75 kBps
Telefonía móvil	3 G = 384 kbps = 48 kBps
Nota: Estas velocidades son brutas. En la práctica, la velocidad neta disponible para el usuario, suele ser entre un 10 y un 15% menor, debido al ancho de banda consumido por las cabeceras y las colas de los protocolos.	

Tabla 1.1. Ejemplos de tasas de transmisión de datos.

La figura 1.7 ilustra las principales fuentes de datos y sus tasas de transmisión asociadas. Las fuentes están clasificadas en audio, video y datos. Los servicios operando por debajo de los 2 Mbps son de banda angosta, mientras que los de mayores tasas son llamados de banda ancha.

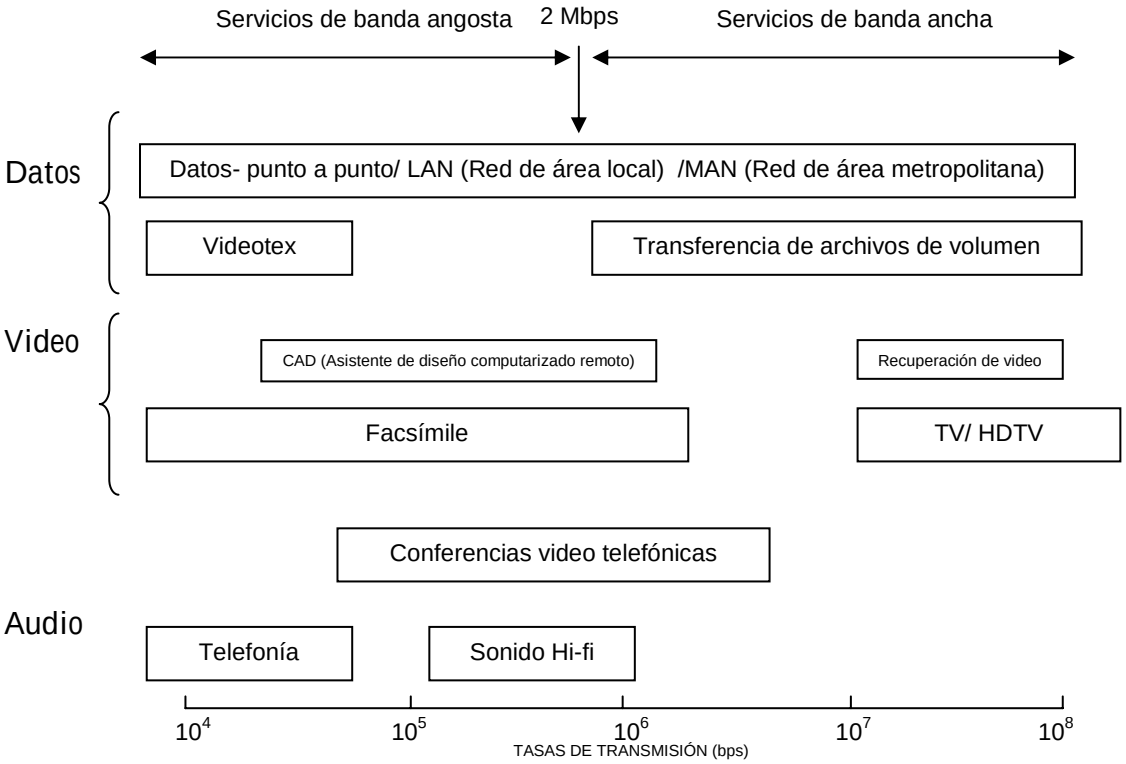


Figura 1.7. Tipos y fuentes de datos.

La videotelefonía ha llegado a ser comercialmente disponible en la forma de *videoteléfonos* donde la imagen del usuario de la red telefónica aparece en el dispositivo de comunicación. Las videoconferencias son un método para establecer una conferencia utilizando cámaras de televisión y monitores entre dos ó más sitios.

La televisión de alta definición (HDTV) es una extensión de los servicios de la televisión tradicional (TV), con una mayor definición consiguiendo con ello, un mayor detalle en las imágenes.

El Diseño Asistido por una Computadora Remota (CAD) permite que la información de una computadora remota sea transmitida. Tal información procede de una gran variedad de aplicaciones de Asistencia por Computadora Remota frecuentemente asociadas con diseños de ingeniería, tales diseños por ejemplo, pueden ser diagramas ó instrucciones para programas.

1.1.2. Modelo de comunicaciones

Las comunicaciones de datos están predominantemente asociadas con comunicaciones soportadas entre dos ó más sistemas computacionales interconectados. Algunas de las principales tareas necesarias para una comunicación exitosa de datos son las siguientes:

- Inicialización de un vínculo de comunicaciones.
- Sincronización entre estaciones transmisoras y receptoras con objeto de interpretar las señales correctamente, por ejemplo al inicio o al término de un paquete de datos.
- Protocolos de intercambio de información para gobernar las comunicaciones. Por ejemplo, un protocolo necesita indicar si una estación puede transmitir y recibir simultáneamente o de forma alternada.
- Control de errores, para determinar si se han recibido señales ó mensajes que estén libres de errores. Donde los errores son evidentes, se aplicaría alguna clase de acción correctiva.
- Direccionamiento y ruteo. Estas funciones aseguran que ocurre un ruteo apropiado dentro de una red, para conectar una estación que envía hacia una estación receptora con la dirección correcta.
- Formato del mensaje. Es lo concerniente con el formato o la codificación utilizada para representar la información.

Aunque esta lista no es exhaustiva, ilustra en algo la amplia gama de complejidad involucrada en una transferencia de datos entre dos entidades. Con objeto de construir un sistema de comunicaciones, la tarea de comunicación debe ser segmentada dentro de un número de sub tareas manejables y su interrelación claramente definida. Una aproximación común para tal análisis, es representar las tareas y sus interrelaciones en la forma de un modelo de comunicaciones conceptual. Si el modelo es lo suficientemente bueno, entonces el sistema puede ser desarrollado satisfactoriamente.

Aunque un sistema de comunicaciones puede ser modelado en un buen número de formas, las comunicaciones computacionales se apoyan fuertemente en modelos estratificados. Cada una de las capas del modelo representa una serie relacionada de tareas, las cuales son un sub grupo del número

total de tareas involucradas en un sistema de comunicaciones. La figura 10 ilustra un modelo de tres capas conectando a dos sistemas.

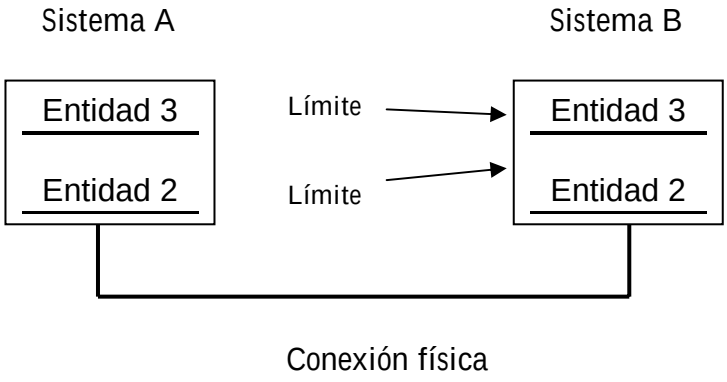


Figura 1.8. Un modelo de comunicaciones estratificado.

1.1.3. Comunicación de datos

Se define *dato* como cualquier entidad que junto con otros, al organizarse, brindan algún tipo de información. Por su origen podemos agruparlos en *analógicos* y *digitales*. Los datos de tipo *analógicos*, como el mostrado en la figura 1.9, pueden tomar cualquier valor en un intervalo continuo, por ejemplo, el video y la voz; contienen valores cuya intensidad varía con el tiempo. La mayoría de los datos que se capturan con sensores son analógicos. Por otra parte, los datos *digitales*, que se muestra en la figura 1.10, toman valores discretos, como por ejemplo los números enteros.

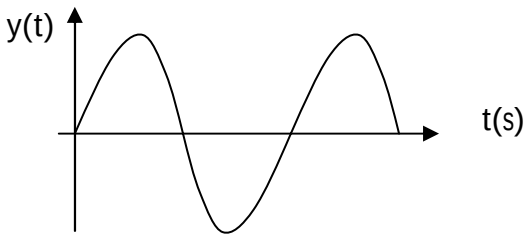


Figura 1.9. Dato analógico.

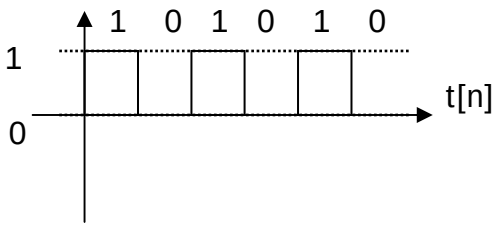


Figura 1.10. Dato digital.

Todos los formatos que se consideran de información, tales como voz, datos, imágenes, video, etcétera, se pueden representar en forma de señales eléctricas. Dependiendo del medio de transmisión y del entorno donde se realicen las comunicaciones, pueden ser utilizadas señales analógicas o digitales para realizar el transporte de datos.

Uno de los problemas más importantes que se presentan cuando se diseña un sistema de comunicación de datos, reside en la dificultad de transmitir información a través de las líneas de comunicación, o debido principalmente a los defectos que éstas presentan, como son: la atenuación, distorsión de retardo, la diafonía y el ruido (tanto en sus variantes de ruido impulsivo como ruido de intermodulación).

Las dificultades en la transmisión de señales analógicas causan efectos azarosos que degradan la calidad de la información transmitida y pueden afectar su inteligibilidad; en el caso de las señales digitales, durante el envío de datos pueden introducirse bits erróneos en la recepción.

Al diseñar un sistema de comunicación de datos se deben tomar en cuenta cuatro factores determinantes, los cuales son:

1. El ancho de banda de la señal.
2. La velocidad a la que se transmiten los datos.
3. La cantidad de ruido presente en el proceso de envío de datos.
4. La porción o tasa de errores que se pueden tolerar.

El ancho de banda disponible está limitado por el medio a través del cual se transmite, así como por la necesidad de evitar interferencias con señales cercanas. Debido a que el ancho de banda es un recurso escaso, es conveniente maximizar la velocidad de transmisión para el ancho de banda del cual se dispone. La velocidad de transmisión está limitada por el ancho de banda, la presencia de defectos en las líneas de transmisión, como por ejemplo el ruido, y por la tasa de errores que se tolera.

El éxito en la transmisión de datos depende fundamentalmente de 2 factores:

1. Calidad de la señal transmitida.
2. Características del medio de transmisión.

1.1.4. Canales de comunicación

Un canal de comunicación es una vía sobre la cual la información puede ser conducida. Puede ser definido como un medio conductor a través del cual se enlazan los dispositivos de comunicación. La información que se envía a través de él tiene una fuente de donde se origina la información que va a ser transportada y un destinatario al cual se entrega dicha información. Aunque la información se origina de una sola fuente, puede haber más de un destinatario, dependiendo de cuantas estaciones receptoras se encuentren ligadas al canal y de cuanta energía posea la señal transmitida.

En un canal de comunicaciones digitales, la información que fluye es representada por paquetes individuales de datos *(bits),* los cuales pueden ser agrupados dentro de unidades de mensaje más largas (multibits). Como se muestra en la figura 1.11, estas agrupaciones de *bits* reciben el nombre de *byte* cuando la agrupación es de 8 *bits.* La agrupación de 2 *bytes* representa lo que se llama una *palabra (word).* Ambos ejemplos representan unidades de mensajes que pueden ser conducidos a través de un canal de comunicaciones digital.

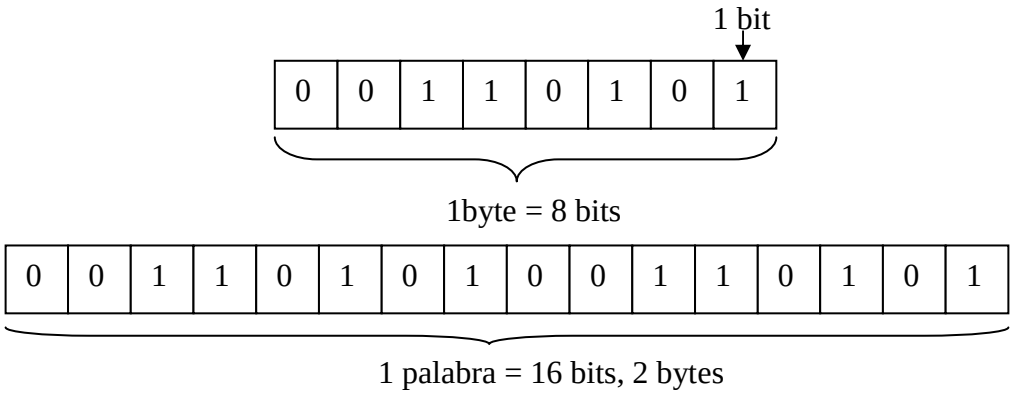


Figura 1.11. Ejemplo de bit, byte y palabra.

Una colección de bytes puede por sí mismo ser agrupado dentro de tramas u otras unidades de mensaje de mayor nivel. Tales niveles de agrupamiento facilitan el manejo de mensajes dentro de una compleja red de comunicación de datos.

En los canales de comunicación existen 3 métodos de envío/recepción de señales: símplex, dúplex o semidúplex y full dúplex.

Símplex

En el método símplex el transmisor y el receptor están perfectamente definidos y la comunicación es unidireccional. Este tipo de comunicaciones se emplean usualmente en redes de radiodifusión, donde los receptores no necesitan enviar ningún tipo de dato al transmisor.

Figura 1.12. Comunicación Simplex.

Dúplex ó semi dúplex

En el caso dúplex ambos extremos del sistema de comunicación cumplen funciones de transmisor y receptor; los datos se desplazan en ambos sentidos pero no de forma simultánea. Este tipo de comunicación se utiliza, con frecuencia, en la interacción entre terminales y una computadora central.

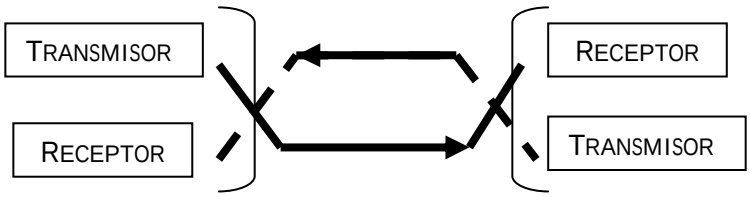


Figura1.13. Comunicación dúplex ó semi dúplex.

Comunicación Full dúplex

El sistema full dúplex es similar al dúplex, pero los datos se desplazan en ambos sentidos de forma simultánea. Para ello ambos transmisores poseen diferentes frecuencias de transmisión o dos canales de comunicación separados. Por ejemplo, para el intercambio de datos entre computadoras, este tipo de comunicaciones es más eficiente que la transmisión *semi dúplex.*

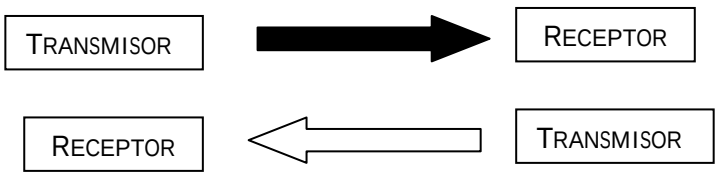


Figura 1.14. Comunicación full dúplex.

En relación con las tres topologías mostradas anteriormente, podemos mencionar que el término *dúplex* es utilizado en el argot de las telecomunicaciones para definir a un sistema que es capaz de mantener una comunicación bidireccional, enviando y recibiendo mensajes de forma simultánea. La capacidad de transmisión en modo dúplex está condicionada por varios factores:

a) Medio físico. La capacidad para transmitir en ambos sentidos.
b) Sistema de transmisión. Capacidad del sistema de enviar y recibir datos a la vez.
c) Protocolos o normas de comunicación empleados por los equipos terminales.

1.1.5. Modos de transmisión de datos

Una señal digital se compone de una sucesión de señales eléctricas de duración fija denominada pulsos ó elementos de señal. El intervalo constante de tiempo que ocupa cada elemento de señal es el *intervalo de señalización.* En el caso más simple, los elementos de señal suelen consistir en la variación de una cierta magnitud física – como la amplitud de una corriente eléctrica o una tensión, por ejemplo- que adopta uno de entre un conjunto finito de valores.

La correspondencia uno a uno entre los símbolos que genera la fuente de información y las señales eléctricas que se transmiten dentro de un intervalo de señalización se conoce con el nombre de *codificación de línea* y debe de elegirse de acuerdo a las características del medio de transmisión. Aunque la binaria es la forma más frecuente de transmisión digital, tal restricción no tiene por qué ser inconveniente para algunos medios de transmisión y, en un caso genérico, un elemento de señal puede codificar más de un bit de información.

Una vez que se ha elegido la codificación de línea, la efectividad de un sistema de transmisión de datos depende de las características de propagación del medio de transmisión y de la calidad de la señal que se recibe. Pero esta calidad, naturalmente, se degrada a medida que aumenta la distancia entre el transmisor y el receptor. En consecuencia, la separación física entre los dispositivos que intercambian información es uno más de los factores que influyen en los métodos y técnicas que se requieren para la transmisión de datos.

Cuando la separación física entre dos dispositivos de comunicaciones es pequeña, de un máximo de 5 m., la manera más común de realizar la transmisión es en *paralelo:* transmisor y receptor se conectan por medio de "n" circuitos idénticos, que se utilizan de forma simultánea para transmitir; en el caso de emplear la transmisión binaria, un bit por cada uno de ellos durante cada intervalo de señalización.

Por razones de costos, la utilización de un sistema de transmisión en paralelo para la interconexión de dispositivos no es una técnica con la que se pueda transmitir información a distancias grandes, ya que en tal caso, la solución natural que se adopta, es la de emplear un circuito único de transmisión por el que se emiten en secuencia todos los bits, uno tras otro. Esta técnica recibe el nombre de *transmisión serie* y es la que se usa mayormente en la transmisión de datos. Hasta la revisión C de la norma EIA/TIA 232, de la cual se obtiene el nombre común para este protocolo de comunicación de datos, se manejaba una longitud máxima de 15 m. Sin embargo, con el advenimiento de nuevas tecnologías sobre todo en conductores, la revisión D de la mencionada norma EIA/TIA 232, maneja que para una velocidad de 19200 bps, puede utilizarse un cable con una longitud máxima de 15 m, para una velocidad de 9600 bps se debe utilizar una longitud máxima de 150 m, para una velocidad de 4800 bps se debe utilizar una longitud máxima de 300 m y, finalmente, para una velocidad de 2400 bauds se debe utilizar un conductor con una longitud máxima de 900 m.

Como una pequeña consideración mnemotécnica, se dice que se utilizará la técnica de transmisión en paralelo *cuando la velocidad de transferencia de datos que se requiere debe ser alta y la distancia de separación entre transmisor y receptor es pequeña,* y se utiliza la transmisión en serie *cuando la distancia de separación entre transmisor y receptor es grande, con objeto de economizar medios de transmisión*, *teniendo como desventaja la disminución en la velocidad del intercambio de datos.*

Los distintos sistemas de transmisión de datos en serie difieren en cuanto a la codificación de línea, la técnica de modulación digital y la manera de conseguir la sincronización entre un transmisor y un receptor.

1.2. Planteamiento del problema y propuesta de solución

En el presente trabajo, se propone el diseño de una estación remota para fines de telemetría, la cual será enlazada con una estación central a través de varios canales de comunicación.

Al ser la telemetría una tecnología que permite la medición remota de magnitudes físicas y su posterior envío hacia un puesto o estación central, el envío de información dentro del esquema de un sistema telemétrico, se realizará mediante comunicaciones de tecnología tanto alámbrica como inalámbrica.

El diseño de la estación remota que se propone realizar pretende contar con 8 canales de entrada de señales analógicas, memoria de datos no volátil, reloj de tiempo real, alimentación de energía a partir de elementos renovables y no renovables y unidades de comunicación serial RS232 y USB, lo cual permitirá el envío periódico de datos almacenados en la estación remota hacia una estación central a través de diferentes medios de comunicación de corto y largo alcance, que son: módem telefónico, radio módem, USB y RS232.

Mientras que en la estación base se plantea desarrollar el software necesario para recuperar las tramas de datos enviadas desde la estación remota y ser almacenadas en una base de datos para su análisis ulterior.

Como todo desarrollo tecnológico, es necesario plantear antes que nada, una plataforma de conocimientos básicos relacionados con todos y cada uno de los componentes que integrarán al sistema, lo cual será cubierto en el siguiente capítulo.

2.1. MICROCONTROLADORES

En muchas ocasiones es muy común escuchar que los términos microprocesador, microcontrolador y microcomputadora se utilizan de forma arbitraría, sin embargo, es necesario que sean aclarados estos puntos: ¿qué es un microprocesador?, ¿qué es un microcontrolador? y ¿qué es una microcomputadora?

Un microprocesador es una unidad central de procesamiento (CPU) en forma de un solo circuito integrado. En el pasado la CPU fue diseñada utilizando una arquitectura a base de circuitos de media y alta escala de integración (MSI y LSI). El primer microprocesador de un solo *chip* fue el Intel 4004 (figura 2.1.) de 4 bits, contenía en un solo encapsulado todos los componentes de una CPU: unidad aritmético-lógica (ALU), decodificador de instrucciones, circuito de control de bus, etcétera.

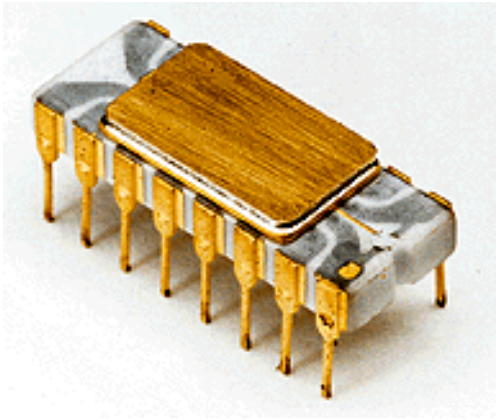


Figura 2.1. Microprocesador Intel 4004.

Cuando se reúne un microprocesador y a toda una serie de dispositivos periféricos de soporte asociados como componentes de entrada y salida (I/O) y memoria , realizando cómputos específicos, como por ejemplo de adquisición de datos y aplicaciones de control, se le llama microcomputadora. Si a todos los elementos que conforman una microcomputadora, se les encapsula dentro de un dispositivo de silicio, se le llama microcontrolador. Texas Instruments fue el creador del primer microcontrolador comercial, el microcontrolador de la serie TMS1000. El microcontrolador TMS1000 (figura 2.2.) tenía suficiente memoria de sólo lectura y de lectura/escritura (ROM y RAM) así como diversos

componentes periféricos de entrada y salida, el cual fue utilizado inicialmente como controlador de hornos de microondas, temporizadores industriales y en calculadoras.

Figura 2.2. Microcontrolador Texas Instruments TMS1000.

Con base en lo descrito en los párrafos anteriores, para la ubicación de un microprocesador, microcontrolador o microcomputadora en su correcta acepción, podemos entonces decir que un microcontrolador es un circuito integrado que incluye las tres unidades básicas de una computadora: CPU, memoria y unidades de entrada y salida (figura 2.3). Contiene prestaciones limitadas pero a cambio, un alto nivel de especialización.

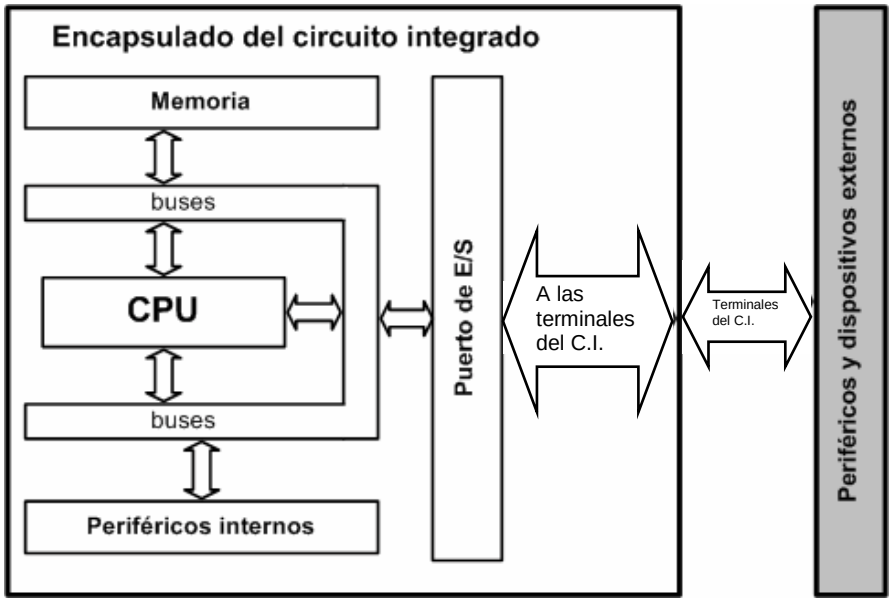


Figura 2.3. Arquitectura básica de un microcontrolador.

Un microcontrolador requiere de un mínimo de componentes externos para ponerlo en funcionamiento. Un microcontrolador típico posee memoria de tipo RAM, ROM, EPROM, EEPROM, dispositivos I/O, convertidores analógico a digitales, circuitos de módulación por ancho de pulsos (PWM), temporizadores, unidades síncronas- asíncronas de recepción y transmisión serial (USART,

anteriormente UART) y buses de interfaz especializados de transferencia de datos como SPI, TWI (I^2C), CAN, USB, ZigBee, etcétera.

2.2. ARQUITECTURAS

Un microcontrolador puede enmarcarse dentro de dos tipos de arquitecturas básicas, en función de la capacidad de expansión respecto de los dispositivos periféricos internos y externos asociados a él que son:

a) Arquitectura abierta

Una arquitectura abierta es aquella que puede ampliarse aún después de la construcción de un sistema, generalmente añadiendo circuitos adicionales, como por ejemplo: módulos extras de memoria o bien, conectando al sistema principal un circuito integrado con un nuevo microprocesador. En una arquitectura abierta, las especificaciones del sistema se hacen públicas, lo cual permite que otras empresas puedan fabricar productos de expansión.

b) Arquitectura cerrada

Una arquitectura cerrada es aquella que es básicamente lo opuesto a una arquitectura abierta, esto es, frecuentemente se utilizan en computadoras o equipos especializados que no requieren de mayores expansiones.

Por otra parte, tomando en cuenta el número de instrucciones o comandos con los que trabaja, un microcontrolador se puede clasificar como:

a) **CISC (Complex Instruction Set Computer).** Es una computadora con un conjunto de instrucciones complejas.
b) **RISC (Reduced Instruction Set Computer).** Es una computadora con un conjunto de instrucciones reducidas.

c) **MISC (Minimal Instruction Set Computer).** Es una computadora con un conjunto de instrucciones mínimas.

Un esquema comparativo entre las arquitecturas RISC y CISC se muestra en la figura 2.4, en la cual se puede observar que la dinámica, en cuanto al manejo de las instrucciones, es más compleja en el caso de una arquitectura CISC que en una RISC.

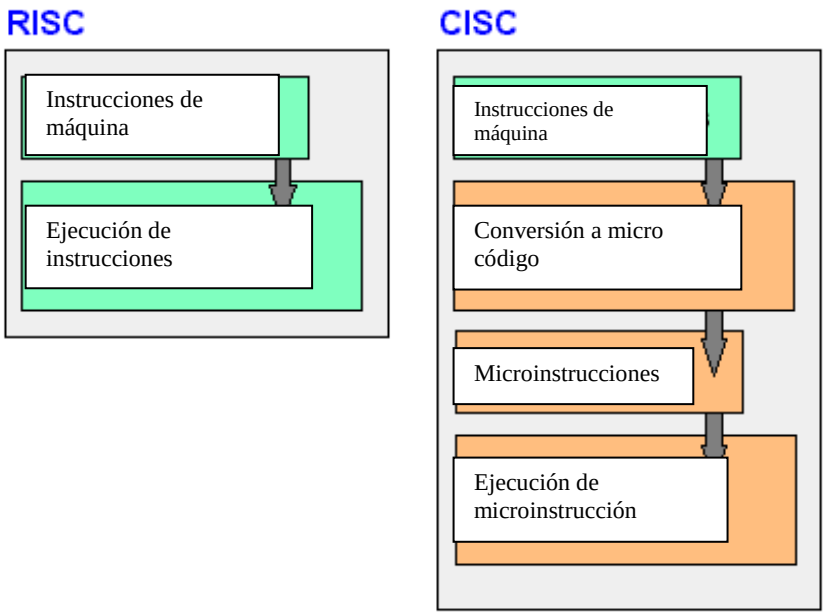


Figura 2.4. Comparativa entre un microcontrolador RISC y un CISC.

Finalmente, otra clasificación de los microcontroladores se da en función de la forma de programarlos y de cómo se puede acceder a su memoria, siendo éstas las arquitecturas Von Neumann y Harvard.

a) Arquitectura Princeton o Von Neumann

La arquitectura Von Neumann se maneja un modelo unificado de memoria, figura 2.5, en el cual el dispositivo de almacenamiento se utiliza tanto para gestión de instrucciones como para datos.

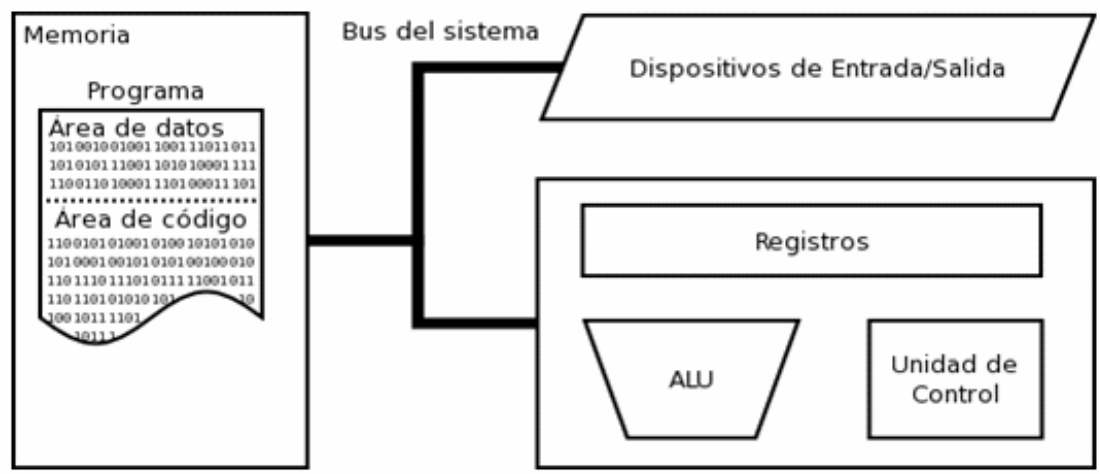


Figura 2.5. Arquitectura Von Neumann.

b) Arquitectura Harvard

La arquitectura Harvard concibe a una computadora en la cual, el dispositivo de almacenamiento es independiente tanto para la gestión de datos como para las instrucciones, figura 2.6.

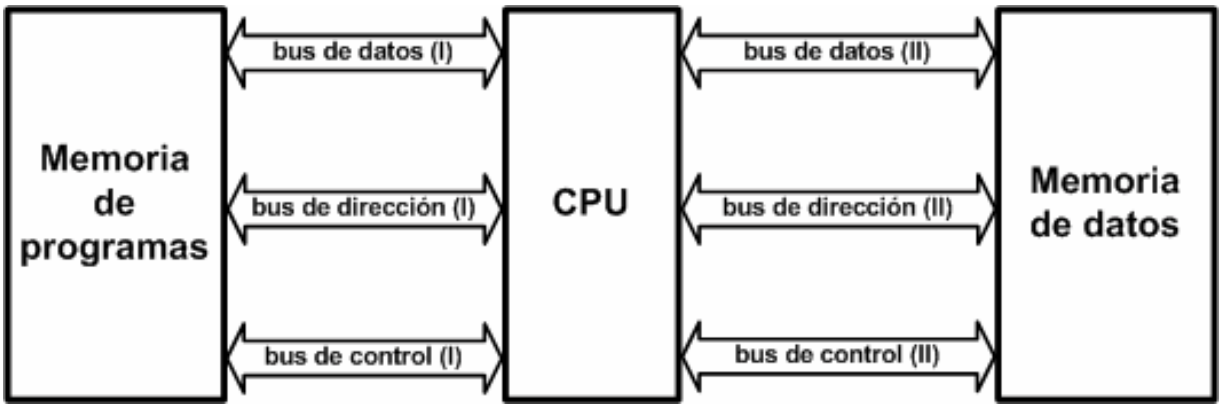


Figura 2.6. Arquitectura Harvard.

2.3. SUBSISTEMAS PERIFÉRICOS

Un microcontrolador está compuesto, dependiendo de la familia, por una serie de subsistemas periféricos que le permiten al dispositivo realizar diversas funciones tanto de procesamiento interno de datos externos como de intercambio y gestión de datos con dispositivos externos, bajo ciertas reglas de transacción denominadas protocolos.

De forma genérica, podemos afirmar que un microcontrolador cuenta dentro de sus subsistemas periféricos, con al menos uno de los que se mencionan a continuación:

- ✓ **CPU.** Es el corazón del microcontrolador. Aquí son almacenadas las instrucciones dentro de la memoria de programa, las cuales se decodifican para su posterior ejecución. La CPU está compuesta básicamente por registros, ALU, decodificadores de programa y circuitos de control.
- ✓ **Memoria de programa.** En la memoria de programa se almacenan las instrucciones que constituyen el programa del micro *per se.* Para acomodar programas extensos, en algunos microcontroladores, la memoria de programa se particiona en dos bloques: como memoria interna de programa y memoria externa de programa. La memoria de programa usualmente es de tipo no volátil y es de tipo EEPROM, EPROM o FLASH.
- ✓ **Memoria de lectura y escritura.** Es un espacio de memoria del microcontrolador, el cual es usado por él para el almacenamiento de datos. La CPU utiliza memoria de tipo RAM para el almacenamiento de variable. La estructura de datos, conocida como pila, es utilizada por la CPU para almacenar las direcciones de memoria que se obtienen de regreso después de haber ejecutado una rutina de algún programa, o bien, después de haber invocado el uso de una subrutina de interrupción.
- ✓ **Reloj.** El microcontrolador ejecuta el programa fuera de la memoria de programa a una cierta velocidad. Esta velocidad está determinada por la frecuencia del oscilador, también conocida como frecuencia del reloj. La señal del reloj puede provenir de un circuito oscilador interno de tipo RC o bien de un oscilador externo, tales como cristales de cuarzo, circuitos resonantes tipo LC, o en algunos casos circuitos de tipo RC. Tan pronto como se energice al microcontrolador, el oscilador inicia su funcionamiento.
- ✓ **Circuito de Reestablecimiento y de Reducción temporal de alimentación (*Reset y Brownout*).** El circuito de reestablecimiento (RESET) en un microcontrolador asegura que al inicio de la ejecución del programa de un microcontrolador, todos los componentes así como los circuitos

de control del microcontrolador arranquen dentro de un estado inicial definido y que todos los registros inicialicen apropiadamente. En el caso del circuito de reducción temporal de alimentación, es un circuito detector que monitorea el voltaje de alimentación del micro, y en caso de presentarse una caída en su magnitud, reestablece al microcontrolador y evita que esa caída en la tensión de alimentación corrompa el contenido de los registros, lo cual podría desembocar en una falla generalizada en la operación del microcontrolador.

- ✓ **Puerto serie.** El puerto de comunicaciones seriales es un componente muy importante en la arquitectura del microcontrolador. Es utilizado para comunicarse con dispositivos externos basado en una comunicación de tipo serie, donde la transmisión y recepción de datos se hace enviando *bit a bit* a través del canal de comunicación. Dependiendo de la frecuencia de la base de tiempo del microcontrolador, éste puede trabajar en velocidades de transferencia de datos desde 2400 bps hasta 2.5 Mbps.
- ✓ **Puertos digitales I/O.** El microcontrolador cuenta con terminales que soportan líneas de entrada y salida de datos, que con previa programación para funcionar como entradas o salidas de propósito general o de dispositivos periféricos internos, permiten la emisión de datos hacia el exterior del microcontrolador desde un periférico interno o, la adquisición de datos desde el exterior hacia un periférico interno con una lógica de control asociada. Todos los microcontroladores generalmente destinan algunas de sus terminales al soporte de líneas de entrada y salida digitales, comúnmente en grupos de 8, y que son reconfigurables vía software mediante la manipulación del estado lógico de ciertos bits en sus registros de control y de datos.
- ✓ **Puerto de conversión analógico a digital.** La conversión de una señal analógica a digital es realizada precisamente por un dispositivo llamado convertidor analógico a digital (ADC). Algunos modelos de microcontroladores están equipados con este tipo de subsistemas periféricos y/o comparadores analógicos, los cuales son utilizados bajo el control de software para la realización de la conversión de una señal analógica a una digital. El convertidor analógico a digital es utilizado para la adquisición de datos provenientes de sensores de temperatura, de presión, de humedad, de velocidad, de desplazamiento, etcétera, los cuales producen una salida que es proporcional a la cantidad física sensada.
- ✓ **Temporizadores.** El temporizador es utilizado por el microcontrolador para controlar el tiempo que dura o en que se ha de producir algún evento. De igual forma, el temporizador puede ser utilizado para llevar la cuenta de la ocurrencia de ciertos eventos tanto internos como externos, en tal caso al temporizador se le llama contador.

- ✓ **Temporizador de vigilancia.** Un temporizador de vigilancia es un temporizador especial que tiene una función específica. Usualmente es utilizado para prevenir fallas en el software del microcontrolador. Funciona de la siguiente forma, si el temporizador guardián no es reestablecido periódicamente por el usuario, el propio temporizador de vigilancia, reestablecerá al microcontrolador con objeto de corregir alguna falla durante el tiempo de ejecución en el software del microcontrolador.
- ✓ **Reloj de tiempo real (Real time clock).** Un reloj de tiempo real es un temporizador especial que tiene la tarea de mantener la cuenta del día, horas, minutos y segundos del microcontrolador.

La figura 2.7 muestra un microcontrolador típico, estos dispositivos vienen en una gran cantidad de tamaños y con varias prestaciones. Como los microprocesadores, los microcontroladores son clasificados como componentes de 8, 16, 36 y 64 bits. Esta clasificación tiene que ver con el tamaño de los registros y el acumulador. Usualmente se entiende que un sistema de 8 bits se conecta a sus diversos componentes a través de *buses* de 8 bits.

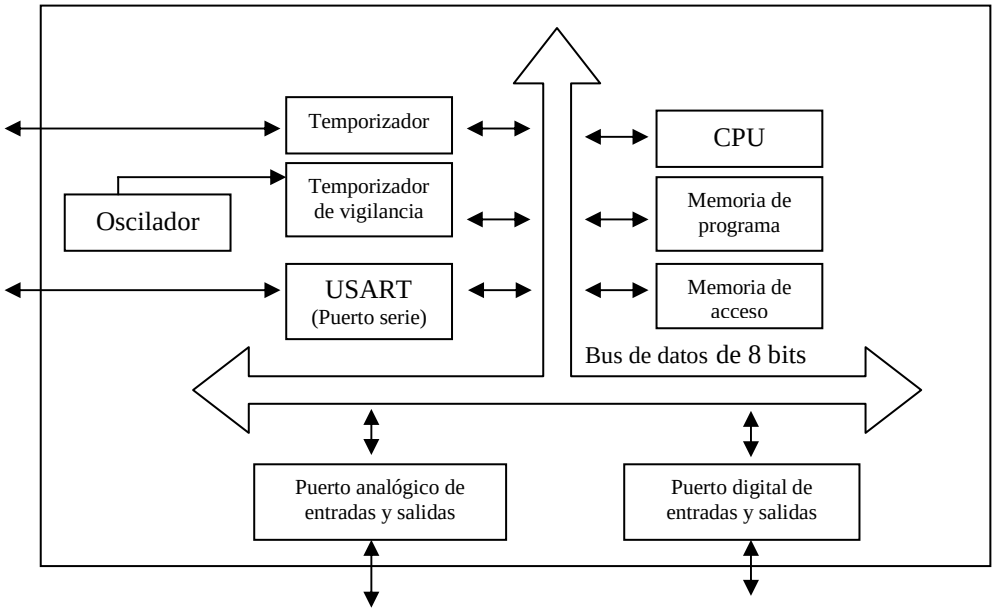


Figura 2.7. Microcontrolador de 8 bits.

Los microcontroladores con grandes *buses* de datos pueden llegar a tener un mejor desempeño que un microcontrolador similar con un *bus* de datos menor. Sin embargo, los microcontroladores con *buses* de menor capacidad resultan ser herramientas de desarrollo más baratas, comparadas con

microcontroladores con grandes *buses* de datos. Actualmente, los microcontroladores de 8 bits han resultado ser de entre los más populares, no sólo por su menor costo, sino porque es relativamente sencillo encontrar software y otras herramientas de forma gratuita para estos dispositivos.

2.4. FAMILIAS DE MICROCONTROLADORES

Literalmente existen cientos de microprocesadores y microcontroladores en el mercado, algunos de ellos con más prestaciones que otros y, muchos de ellos, subagrupados a su vez en gamas bajas, medias y altas de acuerdo a los subsistemas periféricos con los que cuentan, memoria de programa, memoria de almacenamiento de datos, protocolos de comunicación con dispositivos externos, etcétera.

Como corolario de lo antes mencionado, se muestra a continuación un cuadro comparativo, obtenido de *Freertos.org,* la cual es una organización civil estadounidense que agrupa a muchos especialistas en el ámbito del desarrollo de sistemas embebidos a partir de microcontroladores, donde se puede ver a grandes rasgos, una breve comparación en cuanto a capacidad de procesamiento de cuatro microcontroladores de diferentes familias: Texas Instruments con el MSP430, ATMEL con el ATMega323, Microchip con el PIC18F452 y Rabbit con el Rabbit 2000. Algunos de ellos, como en el caso del ATMEL, de la gama alta de su línea de microcontroladores ATMEGA.

Tarea	Tiempo empleado para realizar la tarea (μs)							
	Empresa	Modelo	Empresa	Modelo	Empresa	Modelo	Empresa	Modelo
	T.I.	MSP430	ATMEL	ATMega323	Microchip	PIC18F452	Rabbit	Rabbit 2000
Suma de 16 bits	27		40.4		71.6		63.6	
Multiplicación de 16 bits	72.4		60.8		193		80	
División de 16 bits	480		538		940		608	
Multiplicación de 32 bits	182		191		344		286	
Resta de 32 bits	57.2		75.6		76.4		172	
Ordenamiento de burbuja	992		834		3330		6380	
Movimiento y comparación de bloques de memoria	6750		5800		12400		6360	
Saltos condicionales	131.2		143.6		169		242	
Pushing & Popping	314		258		412		426	
Frecuencia de prueba	8 MHz		8 MHz		20 MHz		22.1 MHz	

Tabla 2.1. Comparativo entre diferentes familias de microcontroladores.

En la tabla 2.1. se observan los resultados de una serie de pruebas en cuanto al tiempo de procesamiento que ocupa cada microcontrolador en realizar ciertas tareas, como por ejemplo operaciones aritméticas y movimiento de bloques de datos. Los detalles de cada una de las pruebas, son los siguientes:

- En las operaciones aritméticas en 16 bits, se utilizó la suma de 2 números de 16 bits declarados como volátiles no signados y el resultado fue guardado en una variable también de 16 bits. La operación fue realizada 20 veces dentro del periodo de prueba marcado para cada modelo de microcontrolador.
- En el algoritmo de burbuja la operación consistió en el ordenamiento, de orden descendente a ascendente, de 20 bytes, declarados como volátiles no signados. La operación se realizó sólo una vez en el periodo correspondiente a cada microcontrolador y fue escrito en lenguaje C estándar.
- En lo que corresponde a los movimientos de bloques de datos, la operación de esta prueba hizo uso de la biblioteca estándar de C, la cual se suministra con funciones de las herramientas de desarrollo para copiar un bloque de 512 bytes de un *buffer* a otro, para luego comparar los valores máximos alcanzados por ambos *buffers,* para asegurarse de que alcanzaron el mismo valor. Esta operación se realizó en 5 ocasiones en el periodo establecido para cada microcontrolador.
- En el caso de los saltos condicionales, la operación se inicia con la limpieza de una variable recursiva de cuenta, cargándola con un valor cero antes de hacer la primera llamada a una función recursiva. Posteriormente, la función incrementa la cuenta de la variable para ver si ésta ha llegado a 5. Si el valor de la variable es menor a 5, la función se llama a sí misma. En caso de que la variable haya alcanzado el valor de 5, la función regresa y llama a la pila. Además del salto condicional, esta operación pone a prueba la sobrecarga de la llamada a una función en C. Una función recursiva es necesaria para impedir que algún compilador, al optimizar el código, remueva la llamada a todas las funciones juntas. La operación se realizó 10 veces, en el periodo de cada microcontrolador, haciendo 50 llamadas a funciones en total.
- En relación a los movimientos *pushing* y *popping* de bloques de datos, esta fue la única operación que incluyó algunas líneas de lenguaje ensamblador. La operación simplemente mete y saca 25 registros en la pila. La operación se repitió 20 veces en el periodo correspondiente, haciendo un total de 500 instrucciones *push* y 500 instrucciones *pop.*

Cabe señalar que las frecuencias utilizadas para las pruebas con los microcontroladores están, en algunos casos, por debajo de la frecuencia real de trabajo, ya que en el caso del MSP430, la frecuencia de trabajo real es de 16 MHz, para el ATMEGA323 es de 8 MHz, en el PIC18F452 es de 40MHz y, para el Rabitt 2000 es de 30 MHz. A pesar de que no se están comparando las características propias de cada microcontrolador, dentro de los modelos comparados podemos mencionar que en su mayoría cuentan casi todos con los mismos periféricos y, solo en algunos casos, como en el del MSP430 que cuenta con un convertidor analógico a digital de 12 bits de resolución o el Rabbit 2000 que trae integrado como módulo de comunicaciones el protocolo TCP/IP, presentan características sobresalientes. Ahora bien, el precio de cada uno de los modelos mencionados deja mejor posicionado al ATMEGA323, con un precio de $6.90 USD, el cual por cierto, a partir del año 2002 dejó de estar en existencia en el mercado para transformarse en el modelo ATMEGA32 el cual, hasta el momento de la redacción de este documento, se encuentra disponible en el mercado. Los costos de los otros integrados son: el MSP430 con $8.04 USD, el Rabbit 2000 con $8.75 USD y el PIC 18F452 con $9.95 USD.

2.5. LENGUAJES DE PROGRAMACIÓN

Para realizar la programación de un microcontrolador se necesita, además del hardware adecuado (tarjeta programadora y cable de conexión serie o USB con la PC), un compilador, que sobre alguna plataforma de programación, permita crear el programa que será guardado vía hardware en la memoria de programa del microcontrolador. Existen muchos lenguajes de programación a través de los cuales es posible realizar la codificación para el microcontrolador, sin embargo, lenguajes como el *ensamblador, C y BASIC,* se han vuelto los más populares para llevar a cabo dicha tarea.

Así mismo, varios fabricantes han desarrollado sus propios compiladores, los cuales permiten traducir el código capturado dentro de la estructura de algún lenguaje de programación en un lenguaje "máquina", que le permita al microcontrolador entender lo que el programador ha escrito en un lenguaje nativo.

En general, los programadores se han inclinado por dos plataformas de programación: lenguaje de máquina y lenguaje C. Cada uno de ellos con sus ventajas y respectivas limitantes, sin embargo, como también fue comentado anteriormente, compiladores como por ejemplo el AVR STUDIO 4.0 de ATMEL, ha incluido ambos lenguajes en el mismo software, por lo que el programador puede

seleccionar desarrollar su aplicación en ASM o C, con el único detalle que el compilador, el cual se descarga de forma gratuita de su sitio en Internet, contiene únicamente la opción de ASM, mientras que, para el caso de C, es un componente *plug in,* el cual también se descarga de forma gratuita y se instala en su compilador para, de esta forma, contar con los dos lenguajes en el mismo compilador.

La tabla 2.2 concentra algunas de las opiniones en contra y a favor, obtenidas de diversos foros de discusión, respecto al uso de ASM o C, para el desarrollo de programas para microcontroladores.

ASM		C	
Ventajas	Desventajas	Ventajas	Desventajas
- Es el lenguaje de programación natural de los microcontroladores - Aprovecha eficientemente los recursos del microcontrolador - Se pueden crear macros, para después simplificar el código en varios subprogramas	- Enormes tiempos de desarrollo - Difícil detectar fallas en el programa. Si este es muy grande, se complica la organización del código en bloques. - El set de instrucciones es privativo de los productos de cada compañía y no es genérico	- Es un lenguaje de alto nivel, más cercano al lenguaje de máquina - Se pueden construir rutinas matemáticas de forma sencilla - En modelos de gama alta, puede combinarse con directivas ASM - Se puede agrupar el código en bloques para su mejor entendimiento - Es aceptado por la mayoría de los compiladores y microcontroladores actuales	- Al compilarse, algunos programas, tienden a ser algo extensos y utilizar mayor cantidad de memoria de programa

Tabla 2.2. Comparación entre lenguajes ASM y C.

2.6. MEMORIA DE ALMACENAMIENTO DE DATOS

Una de las partes más importantes de un sistema embebido es la que está dedicada al almacenamiento de los datos que llegan al sistema. Esta es la tarea que desempeñan las memorias: *son dispositivos que son capaces de proveer los medios para el almacenamiento temporal de información.*

Las memorias son los circuitos que permiten el almacenamiento y recuperación de la información. Aunque conceptualmente parezcan sencillas presentan una relativa amplia diversidad de tipos, tecnologías, estructuras, prestaciones, protocolos de comunicación y costo entre todos los componentes de un sistema embebido. Pueden ser de varios tipos *PROM, EPROM, EEPROM* (también llamada *flash*) y *ROM*. Todos estos tipos de memoria son de las llamadas "no volátiles", lo cual significa que su contenido permanece incluso aún después de desconectar la energía de alimentación.

Podemos considerar a una memoria como un conjunto de *M registros* de *N bits* cada uno de ellos. Estos registros ocupan las posiciones desde el valor 0 hasta M-1. Para acceder a cada registro es necesaria una "lógica de selección". En general, para cada registro se pueden realizar procesos de lectura y de escritura. Para realizar todas estas operaciones son necesarios los siguientes terminales:

• Terminales de datos (de entrada y de salida): Estas terminales permiten la conexión física y la transferencia de datos entre dispositivos que pueden llegar a enmarcarse dentro de una estructura maestro- esclavo, donde generalmente, el dispositivo maestro es un microcontrolador que gestiona y controla dicho flujo de datos.

• Terminales de direcciones: Cada dispositivo está diseñado para funcionar con algún *bus* de comunicación (SPI, I^2C) y una única dirección de acceso, que puede venir preestablecida por el fabricante. Existen dispositivos que permiten establecer externamente parte de la dirección de acceso. Esto permite que una serie del mismo tipo de dispositivos se puedan conectar en un mismo *bus* sin problemas de identificación.

• Terminales de control. Son los que permiten especificar si se desea realizar una operación de escritura o de lectura así como seleccionar el dispositivo.

• Selección del circuito: La terminal de selección del circuito (*chip select*) es la que permite la interacción con el dispositivo (habitualmente con nivel bajo).

El uso de las memorias dentro de un sistema embebido se debe a que, algunos microcontroladores no tienen sistemas de almacenamiento masivo de datos para, por ejemplo, cargar un programa de arranque o almacenar una gran cantidad de datos provenientes de dispositivos periféricos internos del propio microcontrolador.

2.7. SENSOR DE TEMPERATURA

La temperatura es una magnitud física referida a las nociones comunes de frío o calor. Físicamente, es una magnitud escalar dada por una función creciente del grado de agitación de las partículas de los materiales. A nivel microscópico, la temperatura se define como el promedio de la energía de los movimientos de una partícula individual por grado de libertad.

Un sensor es un dispositivo capaz de transformar magnitudes físicas o químicas, llamadas variables de instrumentación, en magnitudes eléctricas. Las variables de instrumentación dependen del tipo de sensor y pueden ser, por ejemplo la temperatura, la intensidad luminosa, la distancia, la aceleración, el desplazamiento, etcétera.

Un sensor de temperatura es entonces un dispositivo que convierte la magnitud física temperatura en una magnitud eléctrica, generalmente un voltaje o resistencia eléctrica.

Existen una amplia variedad de sensores de temperatura que, como fue comentado anteriormente, generan una señal de salida en términos de voltaje o resistencia eléctrica, las cuales varían en función de la temperatura de la sustancia, líquida o gaseosa, que rodee al sensor. De entre los sensores de temperatura más comunes se encuentran: los termistores, los RTD's, las termocuplas y algunos circuitos integrados especializados.

Termistores

Un termistor es una resistencia cuyo valor varía en función de la temperatura (figura 2.8). Existen dos clases de termistores: los de coeficiente de temperatura negativo (NTC) cuyo valor se decrementa a medida de que aumenta la temperatura y los de coeficiente de temperatura positivo (PTC) cuyo valor aumenta a medida de que aumenta la temperatura.

Figura 2.8. Termistores.

Detectores de temperatura de tipo resistivo

Los detectores de temperatura de tipo resistivo (RTDs) están basados en un conductor de platino y otros metales (figura 2.9), que se utilizan para medir temperaturas por contacto e inmersión y en especial para un intervalo de temperaturas elevadas, donde no se pueden utilizar semiconductores u otros materiales sensibles. Su funcionamiento se basa en el hecho de que un metal, cuando sube la temperatura aumenta su resistencia eléctrica.

Figura 2.9. Termo resistencias (RTD's).

Termocuplas

El sensor de *termocupla* está formado por la unión de 2 piezas de metales diferentes. La unión de los metales genera un voltaje muy pequeño que varía con la temperatura. Su valor está en el orden de los milivolts, y aumenta en proporción con la temperatura. Este tipo de sensores cubre un rango amplio de temperaturas, desde -180°C hasta 1370°C, (figura 2.10).

Figura 2.10. Termocuplas.

Circuitos integrados para medir temperatura

Existe una amplia variedad de circuitos integrados sensores de temperatura, en muchos de ellos el elemento básico de sensado lo representa un diodo. Es bien sabido que el diodo es el sensor de temperatura de menor costo que se puede hallar, sólo es necesario hacer una buena calibración y mantener una corriente de excitación estable. El voltaje sobre un diodo conduciendo corriente en polarización directa tiene un coeficiente de temperatura que ronda los 2.3 mV/°C y su variación, dentro de un intervalo, es lineal. Los circuitos integrados que aprovechan esta característica de los diodos tienen la ventaja que ya vienen preparados, por medio de terminales adecuadas para realizar sobre ellos las conexiones necesarias tanto para la alimentación de energía así como para obtener la señal de salida utilizable que, frecuentemente, se presenta en forma de un voltaje que es proporcional a la temperatura sensada por el dispositivo. También existen otro tipo de dispositivos, los cuales se mencionan más adelante, cuya señal de salida se da en forma de una resistencia o una corriente que, al igual que el voltaje, su variación guarda cierta proporcionalidad con la temperatura que se mide.

Los sensores de temperatura de tipo circuito integrado se agrupan en cuatro categorías principales:

- circuitos integrados con salida de voltaje
- circuitos integrados con salida de corriente
- circuitos integrados con salida de resistencia
- circuitos integrados con salida digital

Los sensores de temperatura con salida de voltaje los podemos encontrar en circuitos de uso común como el LM35 (figura 2.11) cuyo voltaje de salida es proporcional a la temperatura en °C, el LM34, (figura 2.12) cuyo voltaje de salida es proporcional a la temperatura en °K.

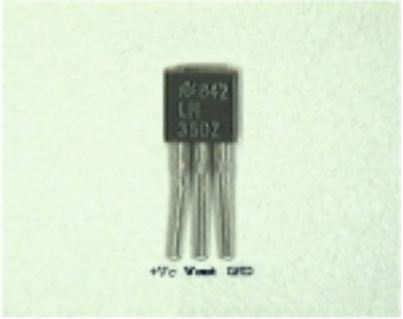


Figura 2.11. Sensor de temperatura LM35.

Figura 2.12. Sensor de temperatura LM34.

Los sensores de temperatura con salida de corriente en circuito integrado más conocidos es el AD590 (figura 2.13), y con salida digital los circuitos LM56 y LM75 (figura 2.14).

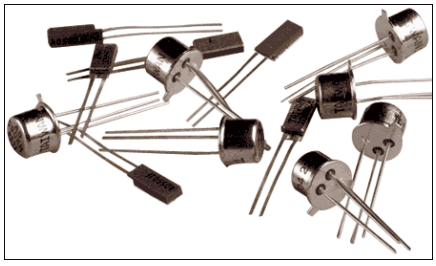


Figura 2.13. Sensor de temperatura AD590.

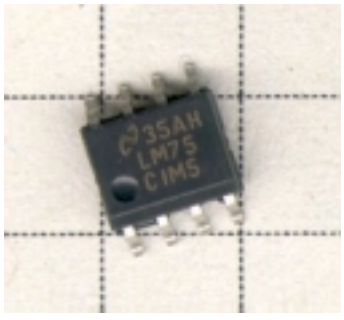


Figura 2.14. Sensor de temperatura LM75.

Finalmente, existen también sensores de temperatura en circuito integrado de señal de salida en forma de resistencia variable, los cuales son menos comunes y que son fabricados por Phillips y Siemens.

2.8. PANELES SOLARES

El panel solar fotovoltaico es el elemento clave en la conversión de la energía solar a energía eléctrica (figura 2.15). Su principio de funcionamiento se basa en el efecto fotoeléctrico, dicho efecto se produce cuando sobre materiales semiconductores convenientemente tratados incide la radiación solar, produciéndose energía eléctrica.

Figura 2.15. Paneles solares.

Es posible clasificar a los paneles en dos grandes grupos, cada uno de ellos atendiendo a los materiales con los cuales están fabricados o bien, por la forma que tienen.

Clasificación en función del material del que están construidas

Existen diferentes tipos de paneles solares en función de los materiales semiconductores y los métodos de fabricación que se emplean. Los tipos de paneles solares más frecuentemente encontrados en el mercado son: de silicio puro monocristalino y policristalino.

a) Panel de silicio puro monocristalino

Los paneles de silicio puro monocristalino están basados en secciones de una barra de silicio perfectamente cristalizado en una sola pieza. En ambiente de laboratorio se han alcanzado con este tipo de paneles rendimientos del orden de los 24.7%, siendo en los modelos comerciales de apenas un 16%, figura 2.16.

Figura 2.16. Panel solar de Silicio monocristalino.

b) Panel solar policristalino

Por las características físicas del silicio cristalizado, los paneles fabricados siguiendo la tecnología policristalina presentan un grosor considerable (figura 2.17). Mediante el empleo del silicio con otra estructura o de otros materiales semiconductores es posible conseguir paneles más finos y versátiles que permiten incluso, en algunos casos, su adaptación a superficies irregulares. Estos últimos se conocen como paneles de lámina delgada.

Figura 2.17. Panel solar policristalino.

Los tipos de paneles de lámina delgada son:

1) Silicio amorfo

2) Telurio de Cadmio
3) Arseniuro de Galio
4) Diseleniuro de Cobre en Indio

Existen también los llamados paneles *tandem,* los cuales combinan dos tipos de materiales semiconductores distintos. Debido a que cada tipo de materiales aprovecha sólo una parte del espectro electromagnético de la radiación solar, mediante la combinación de dos o tres tipos de materiales es posible aprovechar una mayor parte del mismo. Con este tipo de paneles se han llegado a lograr rendimientos del orden del 35%. Teóricamente, con la unión de tres materiales podrían llegarse a rendimientos del 50%.

La mayoría de los módulos comercializados están fabricados de silicio monocristalino, policristalino y amorfo. El resto de materiales se emplean para aplicaciones más específicas y son más difíciles de encontrar comercialmente.

Mención especial en este apartado merece una nueva tecnología que está llamada a revolucionar el mundo de la energía solar fotovoltaica. Se trata de un nuevo tipo de panel solar, muy fino, muy barato de producir y, según sus desarrolladores, presenta el mayor nivel de eficiencia de todos los materiales. Este nuevo tipo de panel solar está basado en el Cobre-Indio-Galio-Diselenido (CIGS) y se tiene previsto que, debido a su competitiva relación producción de energía/costo, pueda llegar a sustituir a los combustibles fósiles en la producción de energía.

Clasificación en función de la forma del panel

Es posible también clasificar a los paneles solares en función de su forma. Empleándose cualquiera de los materiales anteriormente mencionados, se fabrican paneles en muy distintas formas para adaptarse a una u otra aplicación en concreto o bien, para lograr un mayor rendimiento. Algunos ejemplos de formas distintas del clásico son: paneles con sistemas de concentración, paneles en forma de teja y paneles bifaciales.

a) Paneles con sistemas de concentración.

Los paneles con sistemas de concentración, mediante una serie de superficies reflectantes, concentra la luz sobre los paneles fotovoltaicos, figura 2.18.

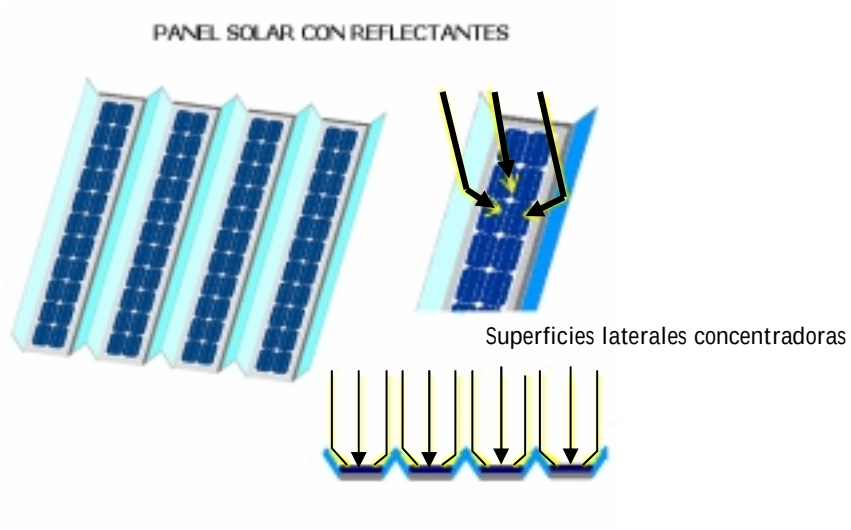


Figura 2.18. Paneles con sistemas de concentración.

b) Paneles en forma de teja.

Los paneles en forma de teja son de tamaño pequeño y están diseñados para combinarse en gran número, para así cubrir las grandes superficies que ofrecen los tejados de las viviendas. Son aptos para cubrir grandes demandas energéticas en los que se necesita una elevada superficie de captación, figura 2.19.

Figura 2.19. Panel solar en forma de teja.

c) Paneles bifaciales.

Los paneles bifaciales están basados en un tipo de panel que es capaz de transformar en energía eléctrica la radiación solar que recibe por cualquiera de sus dos caras. Para aprovechar convenientemente esta cualidad, el panel se coloca sobre dos superficies blancas que refleja la luz solar hacia la parte trasera del mismo panel, una imagen de este tipo de panel se muestra en la figura 2.20.

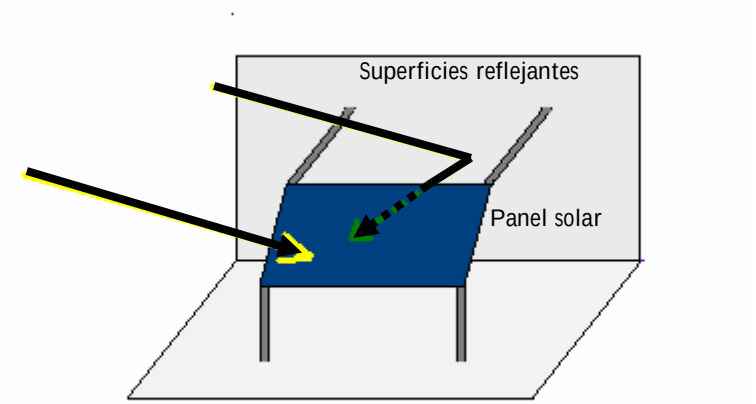


Figura 2.20. Paneles bifaciales.

2.9. BATERÍAS

Se le llama batería al dispositivo que almacena energía eléctrica utilizando procedimientos electroquímicos y que posteriormente devuelve casi en su totalidad; este ciclo puede repetirse un determinado número de veces. La función primordial de las baterías dentro de un sistema de captación solar es la de acumular la energía que se produce durante las horas de luminosidad, para poder ser utilizada por las noches o bien durante periodos prolongados de baja luminosidad solar. Otra importante función de las baterías es la de proveer a la carga una corriente superior a la que el panel de captación solar puede entregar.

El tipo de batería más utilizado actualmente, dado su bajo costo, es la batería de plomo y ácido sulfúrico con electrolito líquido. En ella, los dos electrodos están hechos de plomo y el electrolito está compuesto por una solución de agua destilada y ácido sulfúrico, esa es la razón por la cual se les llama comúnmente baterías de plomo-ácido. Cuando la batería está cargada, el electrodo positivo tiene un depósito de dióxido de plomo y el negativo es plomo. Al descargarse, la reacción química que ocurre

hace que, tanto la placa positiva como la placa negativa, tengan un depósito de sulfato de plomo. Las figuras 2.21 y 2.22 muestran estos estados.

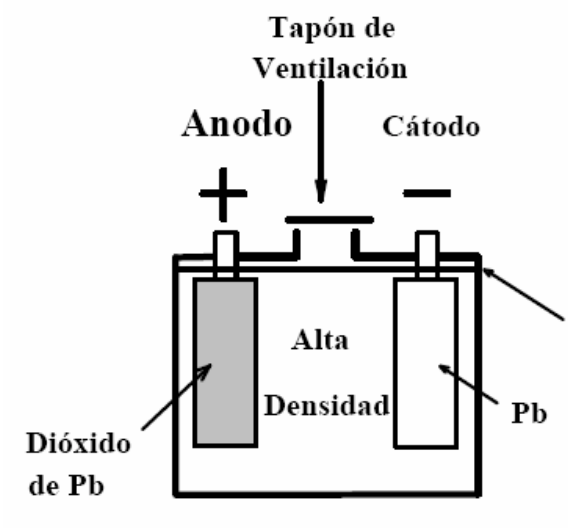


Figura 2.21. Batería cargada.

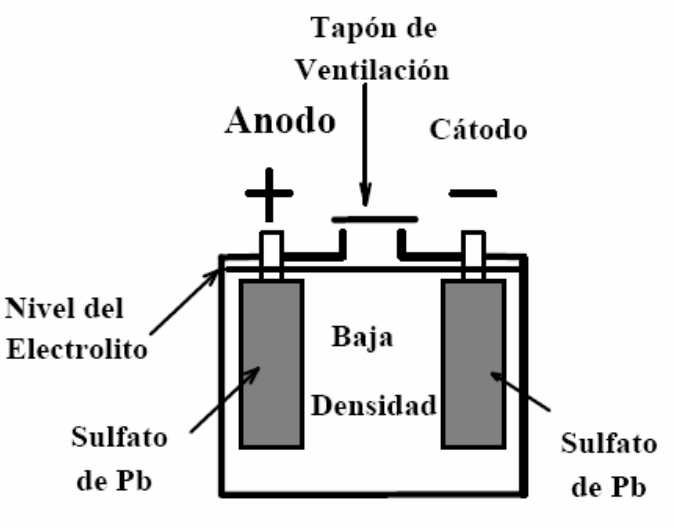


Figura 2.22. Batería descargada.

Debido a su bajo costo, las baterías de tipo plomo-ácido son preferidas en sistemas de captación solar sobre otras, por ejemplo las de tipo Níquel-Cadmio. Dentro del grupo de baterías de Plomo-ácido existen otras combinaciones como: las de plomo-antimonio, plomo-selenio y plomo-calcio, por mencionar a las más comunes.

La unidad básica de una batería es una celda de 2 V. La celda es el conjunto integrado por los electrodos y el electrolito. Dentro de la celda, el voltaje real de la batería depende de su estado de carga, si está cargado, descargando o a circuito abierto. En general, el voltaje de una celda varía entre los 1.75 V a 2.5V, siendo el promedio alrededor de los 2V, voltaje que se suele llamar nominal de la celda. Cuando las celdas de 2V se conectan en serie (positivo a negativo) los voltajes de las celdas se suman, obteniéndose de esta forma, baterías de 4, 6, 12, 24 y 48V; que son los voltajes más comunes encontrados en baterías de este tipo. Las celdas son colocadas dentro de cajas plásticas, las cuales además de ser resistentes a los impactos, son resistentes a la acción corrosiva de los electrolitos utilizados, existiendo dos tipos de cajas: de tipo *hermética o sellada y abierta*. Las baterías de tipo sellada o hermética proporcionan un alto grado de seguridad, ya que el electrolito no puede derramarse, no importando la posición de la misma, figura 2.23. Las baterías abiertas contienen tapones de ventilación, a través de los cuales libera los gases generados durante el proceso de carga, figura 2.24.

Figura 2.23. Baterías de plomo-ácido sellada. *Figura 2.24. Batería de plomo-ácido abierta.*

2.10. PROTOCOLOS DE COMUNICACIÓN DE DATOS

Un protocolo de comunicación es una serie de convenciones y reglas que establecen como deben comunicarse dos equipos o dispositivos electrónicos, por ejemplo dos computadoras a través de una red, con lo cual se reducen los errores de transmisión y permite que fluyan los datos entre ellas.

Las características principales de un protocolo de comunicación son:

1) El tiempo relativo al intercambio de datos entre dos sistemas de comunicación
2) El formato que el mensaje o los datos deben contener para que el intercambio entre dos computadoras, que emplean protocolos diferentes, puedan establecer una comunicación
3) Las acciones que deben realizarse en el caso de producirse un error en la comunicación
4) Las suposiciones del medio en el cual se ejecutará el protocolo

El objetivo del protocolo es establecer una conexión entre equipos terminales de datos (DTE), identificando el emisor y el receptor, asegurando que toda la información se transfiera correctamente.

Los modos de operación, la estructura de los datos, los tipos de órdenes y respuestas, constituyen las diferentes piezas constructivas de un protocolo.

Según la estructura de la información para su transmisión, los protocolos utilizan estructuras diferentes: orientadas a caracteres y orientadas a bits.

Protocolos orientados a caracteres

Los protocolos orientados a caracteres están basados en el uso del código ASCII, usado en transmisión asíncrona. La transmisión se controla por los códigos de control ASCII o EBCDIC. El carácter de control SYN sirve como agente de sincronización entre el transmisor (TX) y el receptor (RX), tiene la propiedad que en rotación circular no se repite a si mismo, sólo hasta después de que ha transcurrido un ciclo completo de 8 bits.

Protocolos orientados a bits

Los protocolos orientados a bits, a diferencia de los orientados a caracteres, se caracterizan por puntos como: independencia de códigos y/o alfabetos, actividad bidireccional, alta fiabilidad y formato único.

Existen aún muchos estándares o protocolos, pero en general, se aceptan sólo aquellos que responden a la arquitectura propuesta por la Organización Internacional de Estándares (ISO), esta arquitectura recibe el nombre de Protocolo de Interconexión de sistemas abiertos (OSI), la cual fue una respuesta de estandarización a la interconexión de equipos. Se trata de una estructura segmentada en 7 capas o niveles. A una determinada capa no le interesa como implementan sus servicios las demás capas. Hay independencia entre capas, de tal forma que a un nivel N sólo se preocupa de utilizar los servicios de la capa N-1 y de realizar los servicios para N+1. Una capa o nivel puede cambiar su estructura interna, pero no los servicios que recibe y que entrega.

El modelo OSI especifica un modelo de comunicaciones dividido en siete niveles, cada nivel define un conjunto de funciones que son necesarias para comunicar con otros sistemas similares. Se comunican únicamente con sistemas adyacentes. Cada uno añade valor a los niveles superiores, hasta que el nivel superior ofrece un abanico de servicios para las aplicaciones de comunicación. En el modelo OSI pueden modelarse o diferenciarse diversos dispositivos que reglamenta la ITU, con el fin

de poner orden entre todos los sistemas y componentes requeridos en la transmisión de datos, además de simplificar la interrelación entre fabricantes. De esta forma, todo dispositivo de cómputo y telecomunicaciones podrá ser referenciado al modelo y por ende concebido como parte de un sistema interdependiente con características muy precisas en cada nivel.

Esta idea da la pauta para comprender que el modelo OSI existe potencialmente en todo sistema de cómputo y de telecomunicaciones, pero que sólo cobra importancia al momento de concebir o llevar a cabo la transmisión de datos. En la tabla 2.3 se presenta un breve resumen de las funciones principales de las capas que conforman el modelo OSI.

Capa	Número	Función	Propósito
Física	1	Transmisión real de los datos a través de un medio físico	Lograr el intercambio de datos
De enlace de datos	2	Confiabilidad de la transmisión	Conseguir la transferencia útil de datos
De red	3	Enrutamiento de las conexiones a través de la red	Lograr la conexión entre terminales específicas de manera precisa y óptima
De transporte	4	Mantener la integridad entre extremos de los datos	Conseguir la comunicación completa y ordenada
De sesión	5	Controlar el diálogo entre dispositivos	Conseguir diálogos coherentes y con significado
De presentación	6	Codificación y formateo de los datos	Conseguir la compatibilidad de sistemas y mayor eficiencia de los canales de comunicación
De aplicación	7	Proporcionar servicios	Actuar como administrador general de la red

Tabla 2.3. Breve descripción del modelo OSI.

2.11. INTERFACES DE COMUNICACIÓN

Es el elemento de conexión que facilita el intercambio de datos, a nivel hardware, de un sistema informático, los podríamos ubicar en la forma de los puertos de comunicaciones de una computadora, los cuales sirven como canales de comunicación para el envío y/o recepción de señales desde un sistema hacia otro.

Para facilitar la conexión entre un equipo DTE y uno DCE (equipo de comunicación de datos) se han desarrollado múltiples estándares que determinan todas las características físicas, eléctricas, mecánicas y funcionales de la conexión, constituyendo lo que se denomina la definición de interfaz. Estos estándares constituyen un ejemplo de los protocolos del nivel físico, y se encuadrarían en el nivel más bajo del modelo de referencia OSI. Existen diversas interfaces de comunicación, destacando de entre ellas: RS232C, USB, IrDA, *Bluetooth,* WiFi (RF), etcétera.

2.11.1. RS232

La interfaz de comunicaciones RS232, comúnmente llamada *serie*, está basada en la norma RS-232-C; fue creada en el año de 1969 por la Asociación de fabricantes electrónicos (EIA), los Laboratorios Bell y los fabricantes de equipos de comunicaciones. En ella se definen tanto la interfaz mecánica, la distribución y disposición de terminales, las señales y los protocolos que debe cumplir la comunicación serial.

Todas las normas RS232 cumplen con los siguientes niveles de voltaje:

- Un “1” lógico es un voltaje definido entre -5 a -15 V en el transmisor y entre -3 a -25 V en el receptor.
- Un “0” lógico es un voltaje comprendido entre 5 a 15 V en el transmisor y entre 3 a 25 V en el receptor.

El envío de niveles lógicos (bits) a través de cables o líneas de transmisión, utilizando la interfaz RS232, necesita la conversión a voltajes apropiados. En los microcontroladores por ejemplo, para representar un “0” lógico se trabaja con voltajes inferiores a 0.8 V, y para un “1” lógico con voltajes mayores a 2 V.

Las velocidades de transmisión que puede soportar esta interfaz van desde los 0 bps hasta los 20 kbps. Con respecto a las distancias máximas se propone que no sean superiores a 15 metros. Aunque un diseño cuidadoso puede permitir distancias mucho mayores, hay que suponer además que, esta limitación teórica se puede manifestar en la práctica en dispositivos que cumplan la norma.

El conector que es utiliza en la interfaz RS232 está compuesto por 25 terminales, y es conocido como DB25; aunque en muchos equipos se utilizan conectores de 9 terminales, llamado DB9. En las figuras 2.25 y 2.26 se muestran los conectores para las versiones mencionadas.

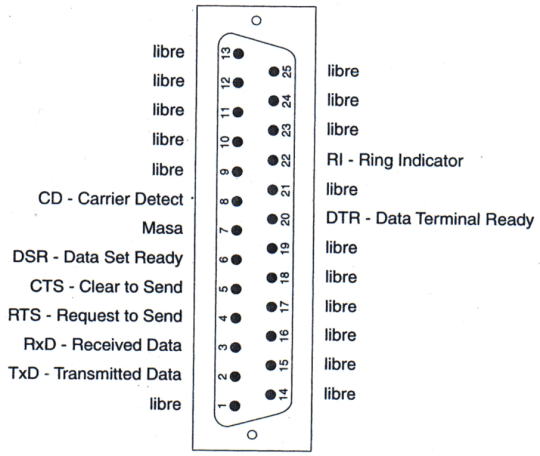


Figura 2.25. Conector DB25.

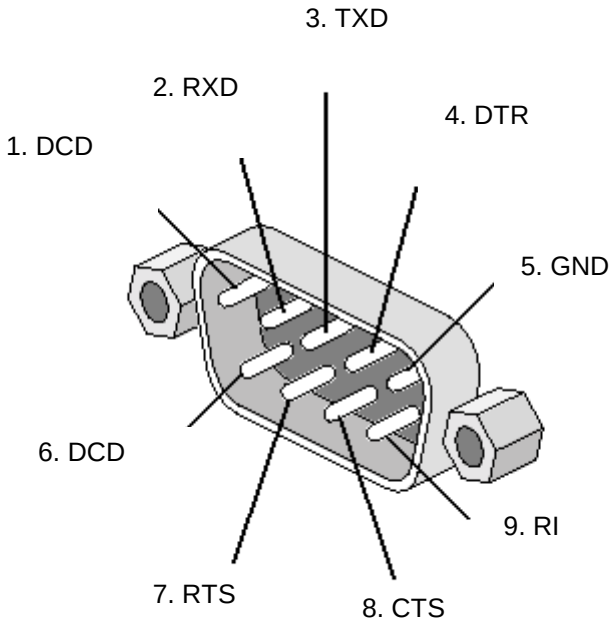


Figura 2.26. Conector DB9.

En cada versión mostrada se pueden encontrar terminales especializadas en el envío y transmisión de datos, así como en el control de flujo de los mismos. A continuación se mencionan los nombres y la función que desempeñan algunas de las terminales más importantes.

- **Transmisión de datos (TXD).** Línea por la que el DTE, por ejemplo una PC, envía los datos.
- **Recepción de datos (RXD).** Línea por que el DTE recibe los datos.
- **DTE preparado (DTR).** Línea por la que el DTE, por ejemplo una PC, indica al DCE, por ejemplo un módem, que está activo para comunicaciones con el modem.
- **DCE preparado (DSR).** Línea por la que el DCE indica al DTE que está activo para establecer la comunicación.
- **Petición de envío (RTS).** Con esta línea, el DTE indica al DCE que está preparado para recibir datos.
- **Preparado para enviar (CTS).** Tras un RTS, el DCE pone esta línea en 1 lógico tan pronto como esté preparado para recibir datos.
- **Tierra (GND).** Necesaria para que tenga lugar la transmisión de datos.

Los nombres de la señal y su número de terminal son exactamente los mismos tanto para el equipo de comunicación de datos como para el equipo terminal, con la salvedad de que funcionalmente son opuestos, esto es, una salida en el equipo terminal es una entrada en el equipo de comunicación y viceversa.

Es evidente que para que exista un flujo de datos entre dos dispositivos RS232 únicamente serán imprescindibles las terminales TXD, RXD y GND. En ciertos casos, como por ejemplo en el caso de conexiones con microcontroladores, las demás terminales podrán ser eliminadas, estableciendo una conexión de tipo *null-módem*, que anula el funcionamiento de las terminales de control cuya función en general, es dotar de un control de flujo en la transmisión y recepción de datos entre un DTE y un DCE.

En el siguiente cuadro se pueden observar la relación entre las terminales de los conectores DB9 y DB25, usado por muchos equipos.

Señal	DB9	DB25
DCD	1	8
RXD	2	3
TXD	3	2
DTR	4	20
GND	5	7
DSR	6	6
RTS	7	4
CTS	8	5
RI	9	22

Tabla 2.4. Relación entre las terminales de los conectores DB9 y DB25.

2.11.2. INTERFAZ DE COMUNICACIONES USB

Desde que IBM sacó a la venta la primera computadora, por razones de compatibilidad, algunas de sus características han permanecido sin cambios. Principalmente en lo que se refiere a conectores de salida como el paralelo, conocido también como *centronics,* la salida serie o RS232 y el conector para teclado y ratón PS2 han sufrido cambios muy pequeños. Si bien es cierto que, algunos de estos conectores, los cuales siguen aún vigentes en equipos antiguos, han comenzado a volverse obsoletos debido a su incapacidad de trabajar con equipos, cuyas velocidades de transferencia de datos, siguen cada día siendo más rápidas.

El bus universal serie (USB) nace como un estándar de entrada y salida de velocidad media/alta que va a permitir conectar dispositivos, que hasta el día de hoy requerían de un hardware especial, en muchos casos propietario, con el fin poderlo aprovechar al máximo, lo cual significaba por supuesto, un costo adicional en los equipos. USB proporciona un conector único que solventa casi todos los problemas de comunicación al exterior, teniendo la capacidad para formar una red de periféricos de hasta 127 elementos.

Al igual que con otro tipo de conectores como los ISA, que han tendido a desaparecer, e igual que con todos los citados al inicio de este apartado, con la inclusión del estándar USB será evidente la reducción de los tamaños de las tarjetas madres o de las tarjetas de expansión, hablando en el caso de las PC's, así como los respectivos controladores para dispositivos como el paralelo, serie, joystick, PS2, etcétera. Como es posible observar, USB es un estándar que, sin lugar a duda, ha venido a revolucionar el sentido de las interfases externas de los equipos actuales, y servido como plataforma de desarrollo para estándares de alta velocidad. Aunado a todas sus bondades, el estándar USB cuenta

con una característica, comúnmente llamada PNP (*plug and play*) lo cual permite las conexiones y desconexiones "en caliente", es decir, que se puede realizar la conexión y desconexión de los periféricos sin tener la necesidad de reiniciar la computadora.

Otras características del bus USB, incluyen:

- ✓ Altas velocidades de transferencia de datos. Para la versión USB 1.0 se definían 2 niveles de velocidad del bus, *full speed* a 12 Mbps y *low speed* a 1.5 Mbps, sin embargo a partir del año 2000, se dio a conocer la versión USB 2.0, el cual redefine el nivel de velocidad *full speed* a 480 Mbps, aproximadamente 40 veces más rápida que la versión *full speed* de la versión USB 1.0.
- ✓ Permite suministrar energía eléctrica a dispositivos que no tengan alto consumo y que no estén a más de 5 m de distancia, lo que elimina la necesidad de conectar dichos periféricos a la red eléctrica con sus correspondientes fuentes de alimentación.

La arquitectura USB se concibe como una topología de estrellas apiladas, como se muestra en la figura 2.27, donde un *hub* es el centro de cada estrella. Cada segmento de cable es una conexión punto a punto entre el *host* y los *hubs* o funciones, o un *hub* conectado a otro *hub* o función. Una función es un dispositivo USB que es capaz de transmitir y recibir datos o información de control sobre el bus.

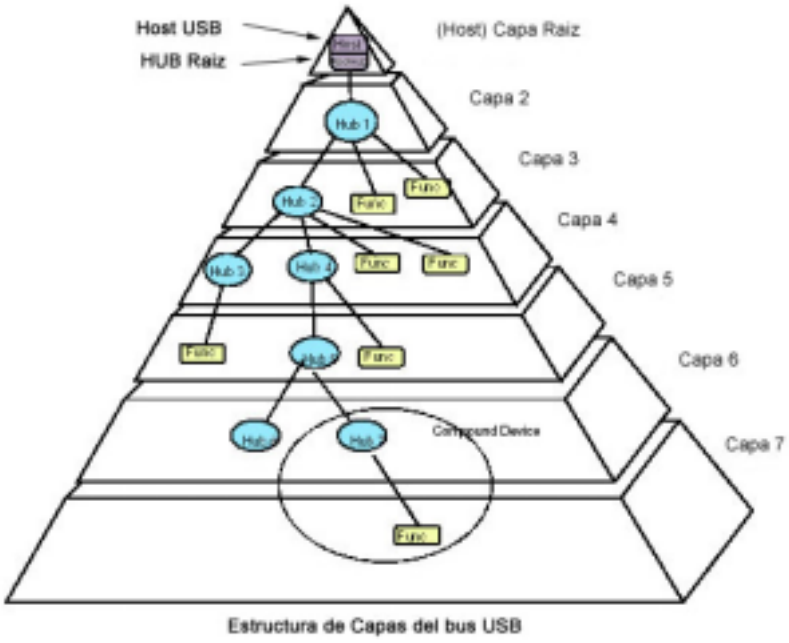


Figura 2.27. Arquitectura del USB.

El número máximo de dispositivos que puede conectar USB es de 127, pero debido a los retardos producidos por la propagación de la señal a lo largo del cable, el número máximo de capas permitidas es de 7 (incluida la raíz), sobre todo para el control de todo tráfico de información en el *bus*.

Físicamente, los conectores USB se dividen en tres tipos: A, B y mini USB, como se muestran en la figura 2.28.

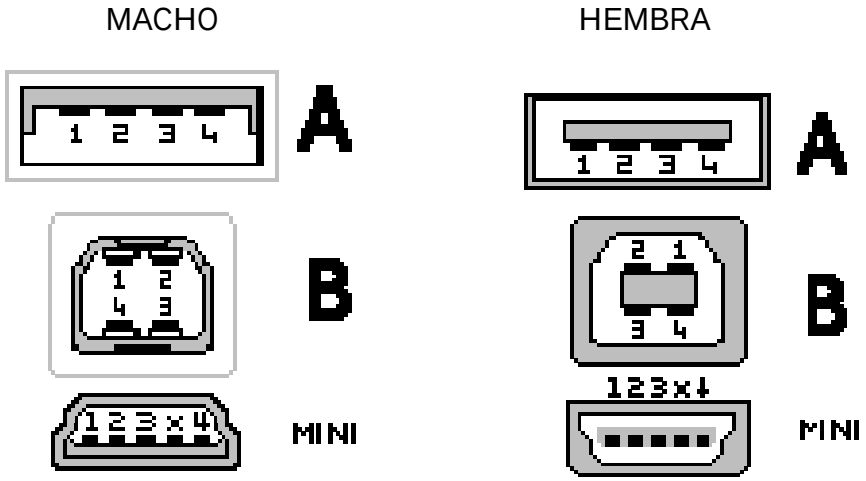


Figura 2.28. Formas de los conectores y terminales USB.

La función de cada uno de los pines de los conectores, se muestra en la tabla 2.4. Como puede observarse está integrada por 4 terminales físicas, dos de las cuales (VCC y GND), se encargan de proporcionar una polarización de 5V/ 100mA al dispositivo periférico que se conecte al bus. Las terminales D+ (*data plus*) y D-*(data minus*), son las terminales de un par trenzado, a través del cual se transporta los datos. Ambas terminales utilizan una señalización de tipo diferencial en modo *half dúplex* para combatir los efectos del ruido electromagnético en enlaces largos. D+ y D- operan de forma conjunta y no son conexiones de tipo *simplex.* La longitud máxima que pueden alcanzar los cables USB es de 5 m.

Terminal	Nombre	Color	Descripción
1	VCC	Rojo	+ 5 Vcd
2	D-	Blanco	Data minus (-)
3	D+	Verde	Data plus (+)
4	GND	Negro	Tierra

Tabla 2.5. Identificación de las terminales de los conectores USB.

Hasta aquí se han presentado, de forma general, todas las partes que integrarán el hardware de la estación remota, ubicado en bloques de adquisición, procesamiento, almacenamiento y envío de datos. En el siguiente capítulo se detalla la utilización de cada uno de estos dispositivos y protocolos de comunicación dentro del hardware antes mencionado.

3.1. DESCRIPCIÓN DE LA ESTACIÓN REMOTA

La arquitectura concebida para la estación remota está basada en un concepto modular, integrado por 5 unidades: unidad de procesamiento y conversión de señales, unidad de alimentación de energía, unidad de temporización, unidad de memoria de almacenamiento de datos y unidad de interfaces de comunicación, según se puede observar en la figura 3.1.

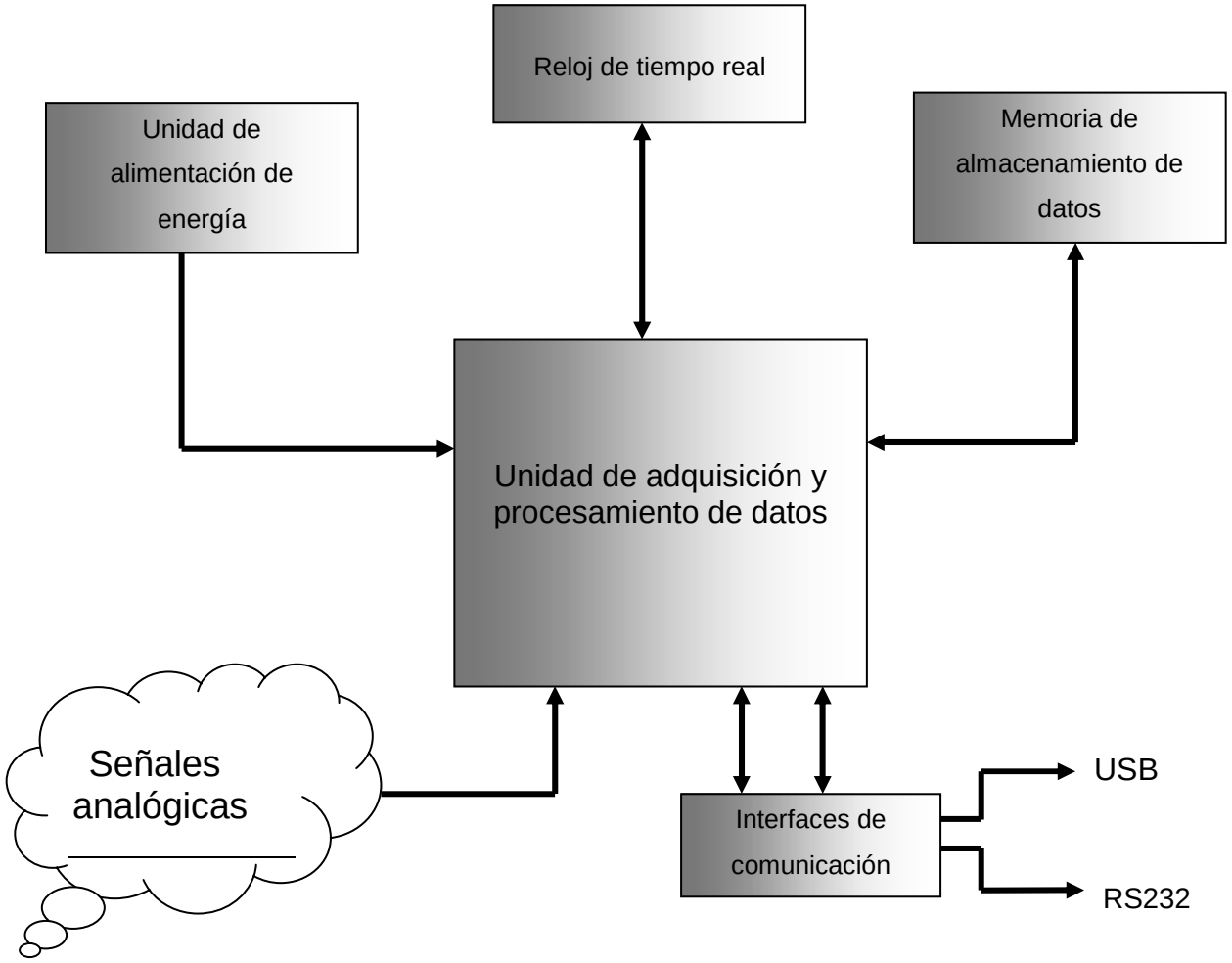


Figura 3.1. Diagrama de bloques de la estación remota.

La figura 3.1 presenta, también de forma esquemática, los requerimientos iniciales de diseño de la estación remota, mismos que consideran como elemento de adquisición y procesamiento de datos a un microcontrolador, el cual tiene que interactuar con dispositivos periféricos de almacenamiento de datos, referencia de tiempo (reloj en tiempo real) y la capacidad para comunicarse con otros dispositivos externos, por canales diferentes, a través de los protocolos de comunicación de alta y media velocidad de transferencia de datos, como el USB y RS232.

En los siguientes apartados se hace la descripción de cómo están compuestas cada una de las unidades que integran a la estación remota.

3.2. UNIDAD DE ADQUISICIÓN Y PROCESAMIENTO DE DATOS

Con base en las necesidades planteadas en el apartado 3.1 y en la disponibilidad de material y herramientas de desarrollo en el mercado nacional, el microcontrolador que se ajustó a nuestros requerimientos de diseño fue el microcontrolador ATMEGA128. Este microcontrolador, de la gama alta de la línea AVR, de arquitectura RISC de 8 bits y 64 terminales, cuenta, entre otras, con las siguientes características:

- 128 kBytes de memoria flash
- 4 kBytes de memoria EEPROM
- 4 kBytes de memoria SRAM
- 53 líneas I/O de propósito general
- 32 registros de propósito general
- 2 USART's
- *bus* de comunicaciones TWI
- *bus* de comunicaciones SPI
- ADC de 8 canales de 10 bits de resolución

Algunas de las particularidades de la familia de microcontroladores son presentadas en el siguiente apartado, lo cual nos llevará a entender la potencialidad del microcontrolador seleccionado.

3.2.1. CARACTERÍSTICAS GENERALES DE LOS AVR

Los microcontroladores AVR's son una familia de microcontroladores de 8 bits de la empresa ATMEL, cuya arquitectura básica fue concebida por Alf-Egil Bogen y Vergard Wollan, estudiantes en el Instituto Noruego de Tecnología (NTH), para posteriormente ser refinado y comercializado por ATMEL Norway, empresa subsidiaria de ATMEL y fundada por los dos arquitectos del circuito integrado. Como comentario curioso, respecto al origen del nombre AVR, propuesto por ATMEL para estos microcontroladores, versiones extraoficiales coinciden en que puede tener 2 orígenes, *Advanced*

Virtual Risc, por el tipo de tecnología del procesador o bien, por las iniciales de los nombres de los inventores, *Alf and Vegard RISC.* Sin embargo, la versión oficial de ATMEL es que, AVR no es ningún acrónimo y no tiene un significado en particular.

El AVR es una CPU de arquitectura Harvard, esto significa que, las instrucciones y los datos se almacenan en espacios de memoria separados con objeto de mejorar el rendimiento del microcontrolador. Como se muestra en la figura 3.2, los registros de propósito general y de control de cada uno de los subsistemas periféricos con los que cuenta el microcontrolador AVR, se encuentran separados del *bus* de datos de 8 bits.

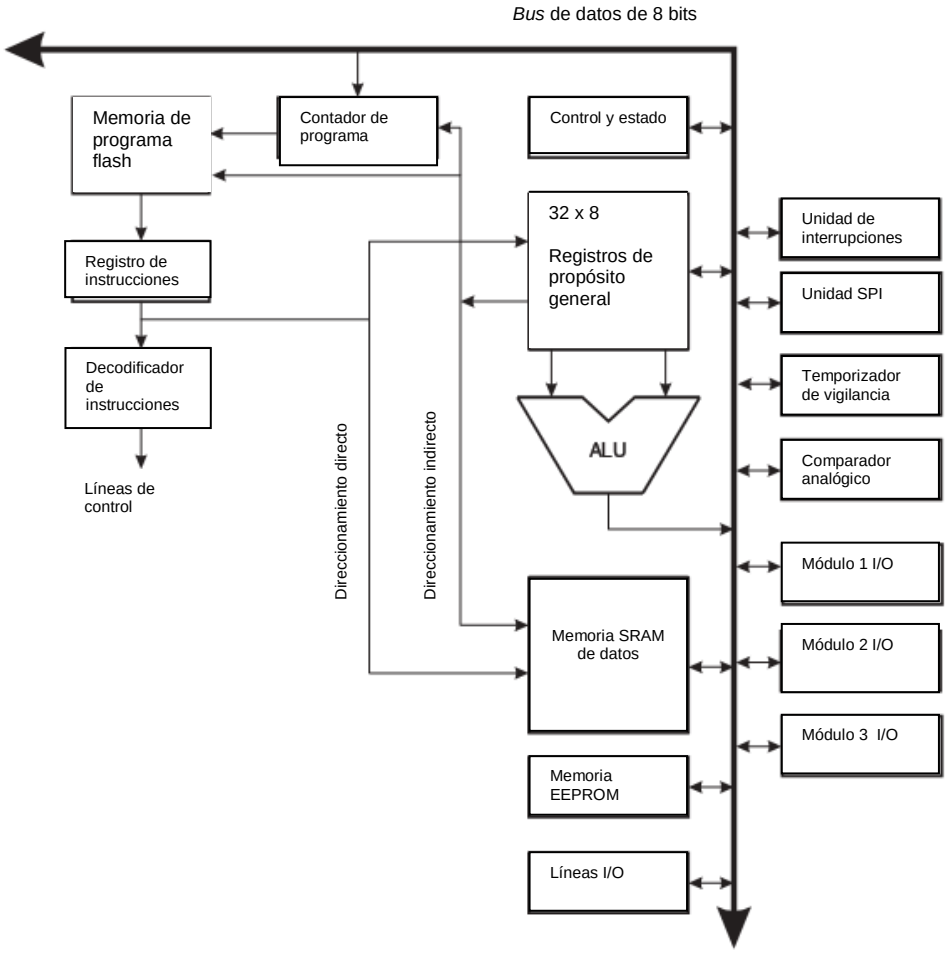


Figura 3.2. Arquitectura AVR.

Los microcontroladores AVR están disponibles en el mercado en diferentes modelos, que comparten el mismo núcleo pero tienen distintos subsistemas periféricos y cantidades de RAM y ROM. Es posible agrupar a la línea de microcontroladores AVR en tres gamas. La gama baja, la cual está integrada por los ya casi extintos AT90SXXXX y los actuales Tiny; estos últimos tienen como

característica que están diseñados para aplicaciones básicas, que no requieran una cantidad importante de memoria, además de no contar con una gran cantidad de subsistemas periféricos, un ejemplo de ellos es el Tiny11, que solamente tiene 1 kByte de memoria Flash y no tiene RAM (sólo cuenta con 32 registros de trabajo) y 8 terminales. La gama media, integrada por los diferentes modelos que van desde el ATMEGA8 hasta ATMEGA64, los cuales tienen como principal característica contar con mayor capacidad de memoria, tanto de programa como de datos, así como un mayor número de subsistemas periféricos; un ejemplo de estos microcontroladores es el ATMEGA32, que cuenta con 32 kBytes en memoria flash, 1024 en memoria EEPROM y 2 kB en SRAM interna. Finalmente la gama alta, comprendida por los modelos que van desde el ATMEGA103 hasta los actuales y gigantescos XMEGA; esta línea se caracteriza por contar con grandes cantidades de memoria tanto flash como EEPROM, además de muchos puertos I/O y subsistemas periféricos, que en algunos casos son múltiples, un ejemplo de esta línea es el modelo ATMEGA128, figura 3.3, el cual cuenta con 128 kBytes de memoria flash, 4 kBytes de memoria EEPROM, 4 kBytes en SRAM, 2 USART's, 53 líneas I/O de propósito general, etcétera. La compatibilidad entre los modelos se preserva en un grado muy razonable.

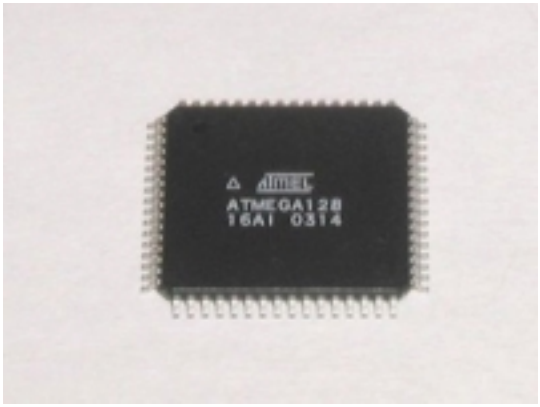


Figura 3.3. Microcontrolador ATMEGA128.

El *set* de instrucciones de los AVR es más regular que el de la mayoría de los microcontroladores de 8 bits, como por ejemplo los PIC's, teniendo en cuenta para los AVR lo siguiente:

- Los registros punteros X, Y y Z, registros 26 al 31, tienen modo de direccionamiento indirecto.

- Los primeros 15 registros son utilizados por instrucciones con modo de direccionamiento inmediato.

En la figura 3.4 se puede observar la distribución de los registros de propósito general de los microcontroladores AVR, así como las direcciones de memoria y la ubicación de los registros punteros.

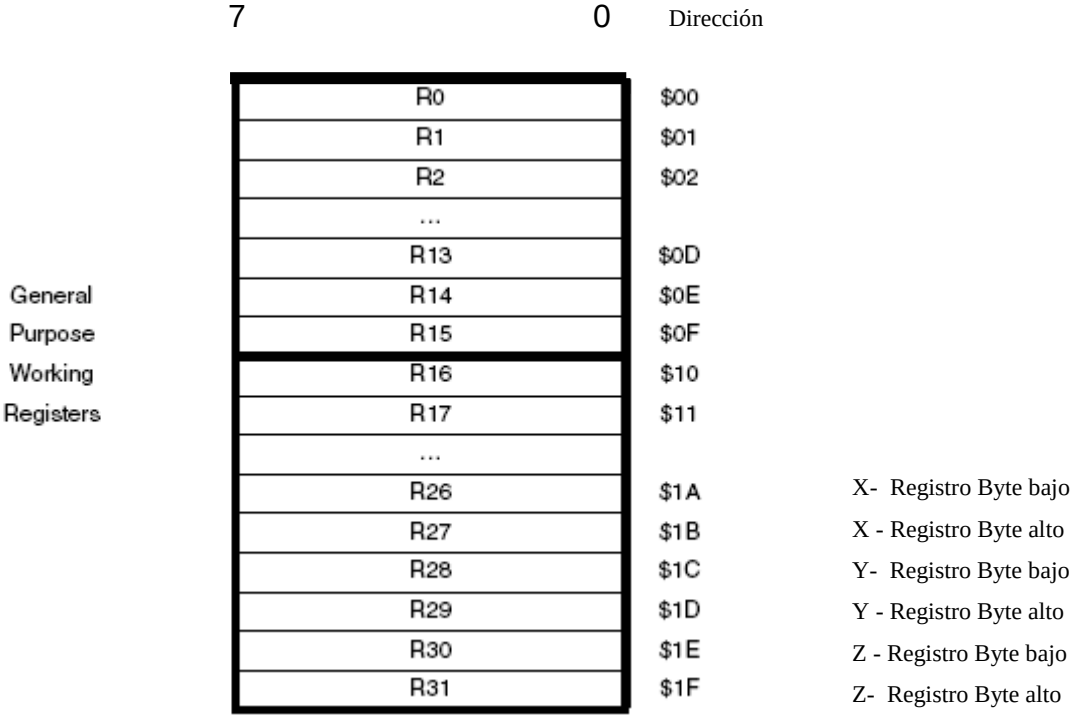


Figura 3.4. Registros de propósito general para los AVR.

3.2.2. ARQUITECTURA RISC

El hecho de que los microcontroladores AVR cuenten con una arquitectura computacional del tipo *de instrucciones reducidas* (RISC), le confiere las siguientes características:

a) Instrucciones de tamaños fijos y presentadas en un reducido número de formatos
b) Sólo las instrucciones de carga y de almacenamiento de datos acceden a la memoria correspondiente

Además, el contar con muchos registros de propósito general que hacen posible la segmentación y la ejecución en paralelo de instrucciones dan como resultado la reducción de accesos a la memoria.

Los microcontroladores AVR cuentan con una dinámica de procesamiento llamada *pipeline,* consistente en cargar y ejecutar, lo cual les permite ejecutar la mayoría de las instrucciones en un ciclo de reloj, lo cual los coloca entre los microprocesadores de 8 bits más rápidos en el mercado. Conceptualmente *pipeline* es el conjunto de actividades básicas en las que se divide la ejecución de una instrucción. Análogamente a una línea de ensamblado, la instrucción en un microcontrolador entra a la *pipeline* y cuando sale ya se ha ejecutado. La velocidad máxima de reloj que puede alcanzar el microcontrolador define la complejidad máxima del proceso asociado a cada instrucción en cada etapa de la *pipeline.* Entonces, mientras más *pipelines* tiene un procesador mayor cantidad de requerimientos puede resolver en paralelo.

La clave de la canalización en *pipeline* en los microcontroladores AVR, es que éstos puedan comenzar a leer la siguiente instrucción tan pronto como se termine de leer la última instrucción, lo cual significa, en este caso, que están trabajando simultáneamente dos instrucciones, una, la que está siendo leída, y otra, que está comenzando a ser decodificada, y en el siguiente ciclo habrán dos instrucciones que estarán siendo procesadas de forma casi simultánea, como puede apreciarse en la figura 3.5.

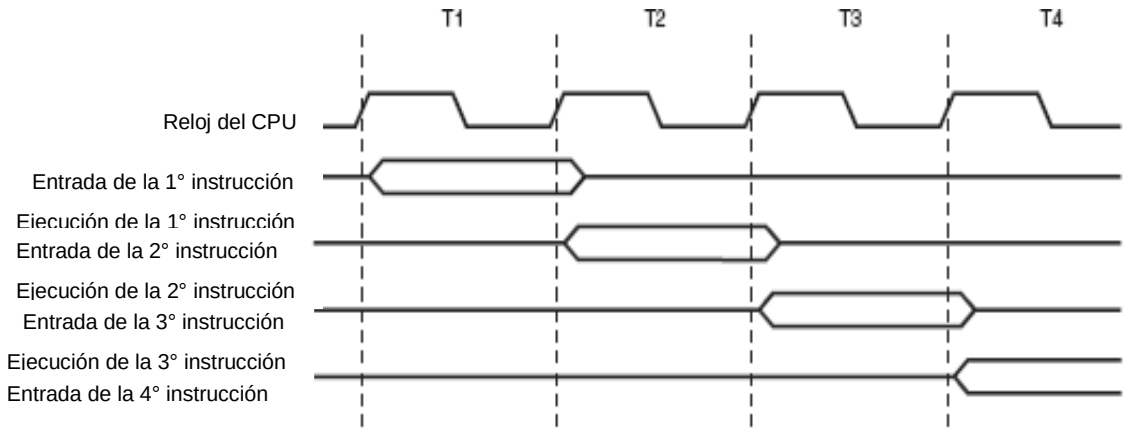


Figura 3.5. Dinámica de carga y ejecución de instrucciones para AVRs.

Esta dinámica les permite a los diseñadores una gran flexibilidad, lo cual facilita:

a) Incrementar el tamaño del conjunto de registros.
b) Implementar medidas para incrementar el paralelismo interno.
c) Añadir grandes espacios de memoria.

3.2.3. ESPACIOS DE MEMORIA

La arquitectura AVR tiene dos espacios principales de memoria, la memoria de programa de tipo flash y el espacio de memoria de datos, de tipo SRAM. Así mismo, se destaca un sector de memoria de datos externa EEPROM, en algunos modelos AVR, como se muestra en la figura 3.6. En dicha figura se puede observar que, cada sector de memoria está a su vez subdividido en pequeños sectores, algunos de los cuales contienen registros de control y de datos de dispositivos periféricos, terminales I/O, así como sectores muy especializados de memoria como la *boot section,* la cual, previa configuración, permite utilizarla una memoria temporal de datos o de programa.

Como puede observarse en la figura 3.6, la **memoria de datos (SRAM)** se divide, de manera general, en tres secciones, la inferior que ocupa las primeras 32 direcciones, que abarca el banco de registros de propósito general (direcciones $00 a $1F). Dentro de este grupo se pueden encontrar tres registros con características muy diferentes a todo el resto y que abarcan desde el registro 26 hasta el 31, los cuales son conocidos como registros punteros X, Y y Z.

La zona central de la memoria de datos cubre *n* direcciones y, dependiendo del modelo, las direcciones de los registros en donde están situados los registros asociados a los distintos subsistemas periféricos como la USART, el comparador analógico, el convertidor analógico a digital, el PWM, la TWI, el SPI, etcétera cambiarán de uno a otro. Así mismo, se encuentran los registros que permiten el control y configuración de los puertos I/O de propósito general del microcontrolador, como también los registros de control de las interrupciones internas y externas y los controles para la utilización de osciladores externos e internos.

La zona más alta (de arriba hacia abajo) de la memoria de datos, está asociada con la memoria **SRAM**, este espacio es utilizado en los modos de bajo consumo a los cuales puede ser puesto el microcontrolador. En este espacio de memoria se guardará la información necesaria para lograr la transición de un proceso de bajo consumo a un proceso normal de operación.

El sector correspondiente a la **memoria EEPROM** está organizado como un espacio de datos separado. Debido a que es una memoria no volátil, los datos que contiene permanecen inalterables aún ante la ausencia de energía eléctrica en el microcontrolador. Análogamente a la memoria flash,

para tener acceso a la memoria EEPROM, se cuentan con instrucciones específicas que permiten trabajar con datos junto con una cantidad de registros asociados que son utilizados por dichas instrucciones, lo que hace menos difícil el manejo de este espacio de memoria. De igual forma que el resto de los espacios de memoria, los cuales son grabados por medio de la utilización del bus SPI, tienen implícitos una cierta cantidad de ciclos de escritura y de borrado, que dependiendo del modelo del microcontrolador, ronda entre los 1000 a 100,000 ciclos.

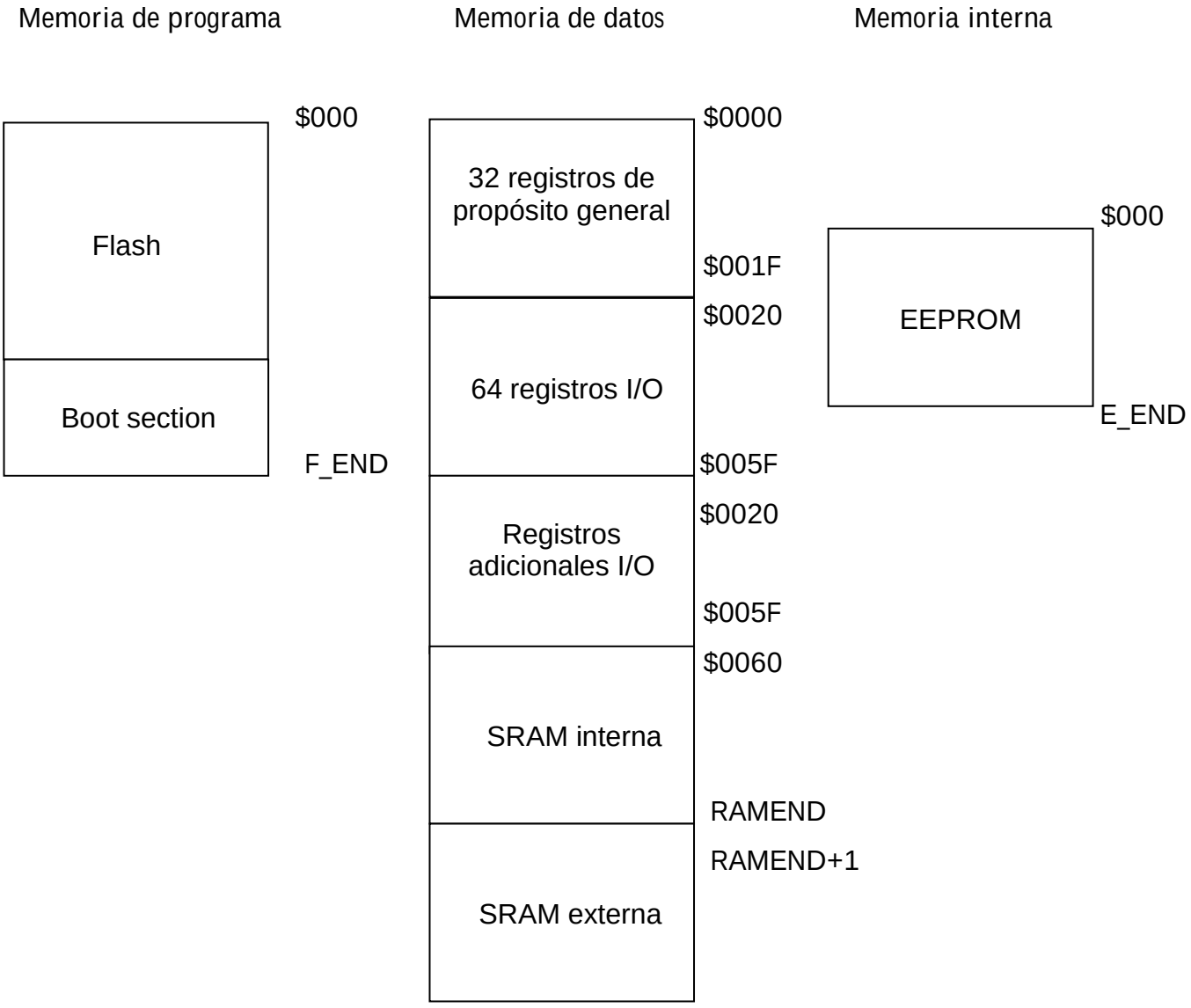


Figura 3.6. Mapa de memoria para microcontroladores AVR.

A modo de ejemplo, en la tabla 3.1, se presenta, para varios modelos de microcontroladores AVR, la dimensión real de cada uno de los sectores de memoria antes mencionados y el margen que comprende cada una de ellos.

Modelo	Memoria flash		Memoria RAM		Memoria EEPROM	
	F_END	Tamaño kB	RAMEND	Tamaño kB	E_END	Tamaño kB
ATMEGA8	$0FFF	8	$045F	1	$1FF	0.5
ATMEGA32	$3FFF	32	$085F	2	$3FF	1
ATMEGA64	$7FFF	64	$10FF	4	$7FF	2
ATMEGA128	$FFFF	128	$10FF	4	$FFF	4

Tabla 3.1. Comparativa de capacidad en memoria entre microcontroladores AVR.

Debido a que las instrucciones que operan sobre los microcontroladores AVR pueden ser de 16 o de 32 bits, en el espacio que ocupa la **memoria de programa,** la cual, como ya se comentó al principio de este apartado, es una memoria de tipo flash, se organiza como un espacio de *n* kB x 16 bits, dependiendo del modelo con el que se trabaje. Como medida de seguridad para el software, en algunos modelos de microcontroladores AVR, el espacio que ocupa la memoria de programa está a su vez dividida en dos sectores, la sección de memoria *boot* y el sector de programa de aplicación, como se muestra en la figura 3.7, para el caso del microcontrolador ATMEGA128.

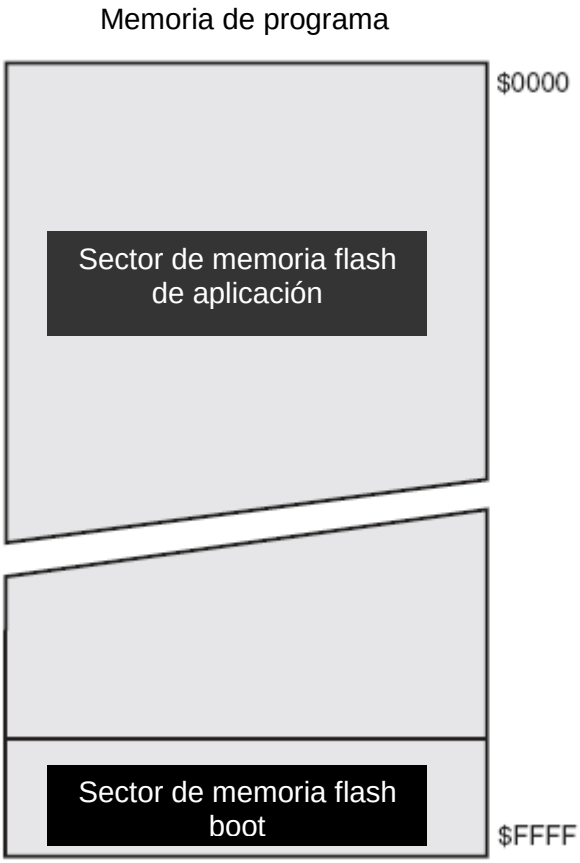


Figura 3.7. Mapa de memoria de programa para el microcontrolador ATMEGA128.

La memoria flash tiene, dependiendo del modelo del microcontrolador, una capacidad de escritura y borrado de 1000 a 10,000 ciclos. La característica de escritura/borrado, junto con una programación por sistema a través del bus SPI, hace que las modificaciones y la carga de programas a los microcontroladores AVR sean procesos relativamente sencillos, permitiendo con ello el rápido desarrollo de aplicaciones.

3.2.4. SEÑAL DE RELOJ

Para el funcionamiento del microcontrolador AVR se necesita de una fuente de pulsos de reloj, la cual se encarga de suministrar al AVR una frecuencia de trabajo a la CPU del microcontrolador. Este reloj de la CPU está ligado a los módulos de los registros de propósito general, registro de estado, registros de memoria de datos entre otros. Al detener el reloj del CPU, se inhibe al núcleo para realizar operaciones o cálculos.

En la figura 3.8 se muestran las diversas opciones de cómo están ubicadas las fuentes de señales de reloj con que cuentan los microcontroladores AVR y sus usos al interior dentro de la estructura del propio microcontrolador que, dependiendo del modelo del microcontrolador, pueden ser las señales de 1 MHz, 2 MHz, 4 MHz, 8 MHz y otros más. En dicha figura se puede observar, en la parte de abajo, las fuentes de reloj disponibles en la arquitectura del microcontrolador, las cuales pueden ser internas y externas; las internas, generadas por osciladores configurables por software y las externas, generadas, principalmente, por osciladores externos, basados en elementos piezoeléctricos, redes pasivas o generadores externos. En la parte superior de la figura 3.8 se pueden apreciar los elementos del microcontrolador que requieren alguna de estas fuentes de reloj, entre estos elementos podemos distinguir temporizadores, registros I/O de propósito general, subsistemas periféricos, CPU y sectores de memoria asociados.

Sin embargo, cuando se tienen aplicaciones específicas que demandan una señal de reloj de una frecuencia específica distinta a la que se puede conseguir por el propio microcontrolador, existe la posibilidad de asociar una señal de reloj externa, dentro de un intervalo especificado para cada modelo de AVR, que generalmente siempre es mayor al que puede generar internamente. El hardware adicional que se necesita para montar dicha fuente de reloj consiste, generalmente, en un oscilador generado con base en un cristal de cuarzo o un resonador cerámico, figura 3.9, y dos capacitores de valor pequeño, generalmente 33 pF, figura 3.10.

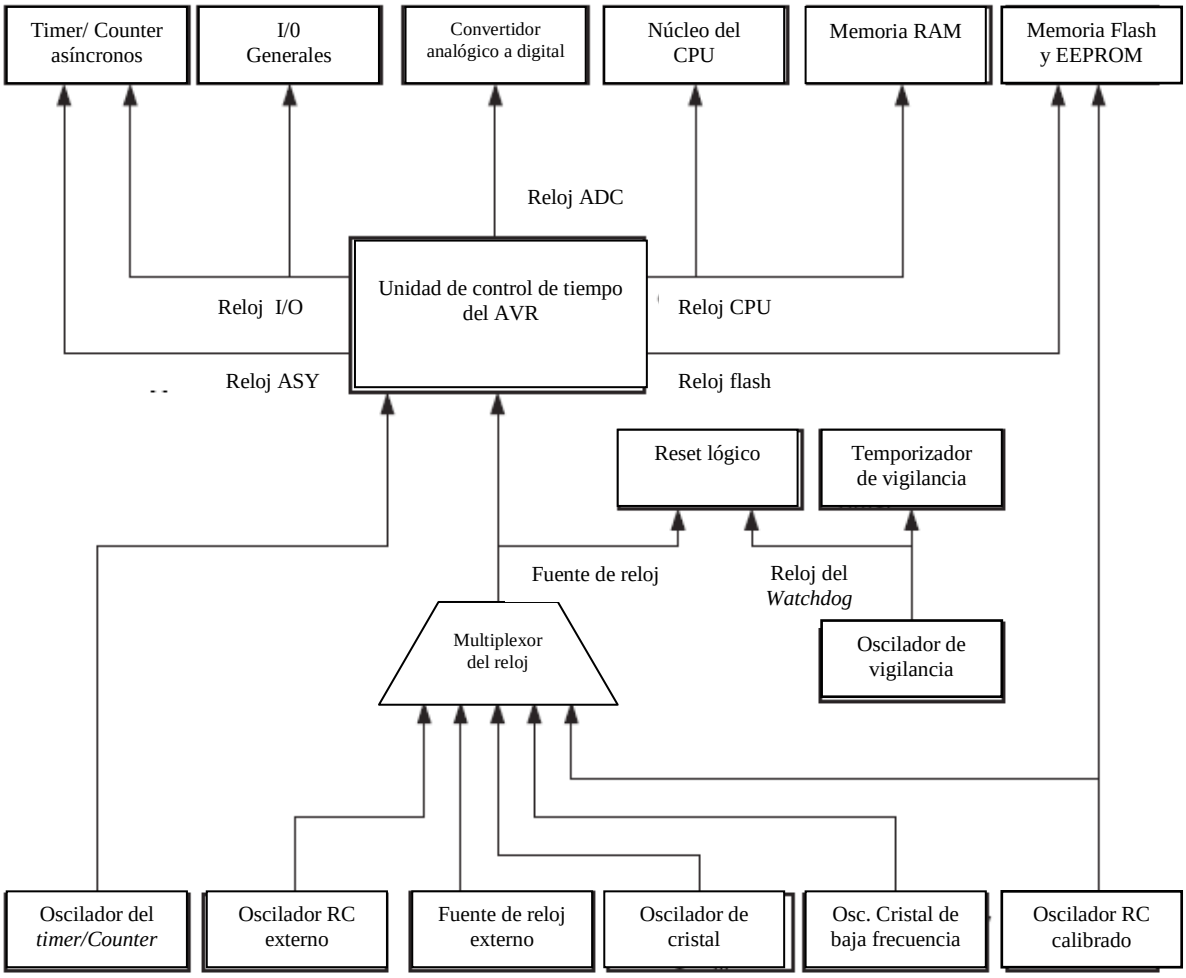
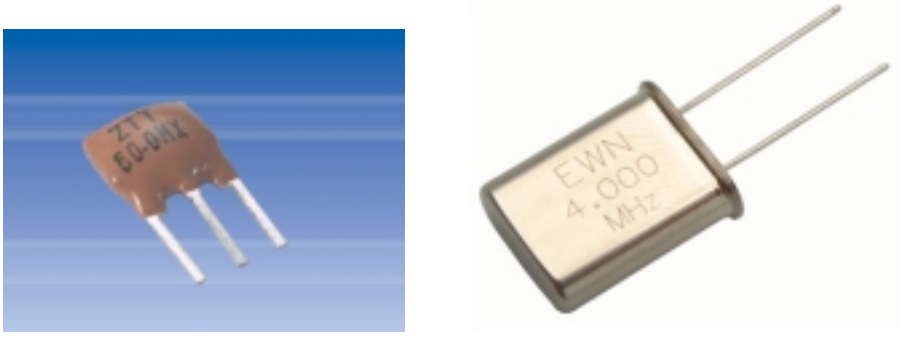


Figura 3.8. Opciones de reloj para los microcontroladores AVR.



(a) (b)

Figuras 3.9. (a) Resonador cerámico de 16 MHz y (b) Cristal de cuarzo de 4 MHz.

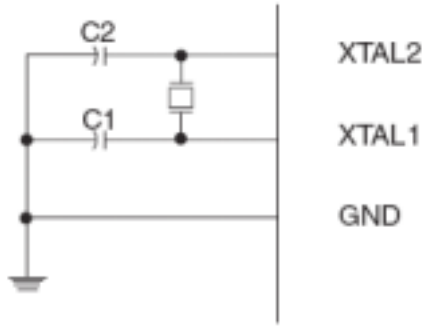


Figura 3.10. Montaje de una fuente de reloj externo al microcontrolador AVR.

Un buen criterio de selección para utilizar un oscilador con base en un cristal de cuarzo o con un resonador cerámico está relacionado con la precisión de la frecuencia; cuando la precisión en la frecuencia de oscilación no es crítica, en lugar de utilizar un cristal de cuarzo puede ser una buena opción el uso de un resonador cerámico, los que resultan en general más económicos. Mientras que en un cristal de cuarzo puede presentarse una desviación en su frecuencia nominal de hasta 0.005%, en un resonador cerámico dicha desviación puede llegar a ser de hasta el 0.5%.

Otra opción que se puede utilizar, y que resulta la más barata, es la que se muestra en la figura 3.11, donde se configura un oscilador externo mediante el uso de una red RC, mediante la cual se pueden generar valores de oscilación aproximados a los que entrega el reloj interno del microcontrolador. Sin embargo, el uso de redes RC como fuentes de señal de reloj, no siempre suele ser la mejor opción ya que, debido a las variaciones en el valor de los capacitores asociados a cambios por ejemplo, en temperatura, llevan a considerar a las redes RC como una fuente inestable de señales de reloj, recomendándose su uso únicamente en aplicaciones menores a los 5.5 MHz y en las que los requerimientos en cuanto a precisión en frecuencia no sea muy importante.

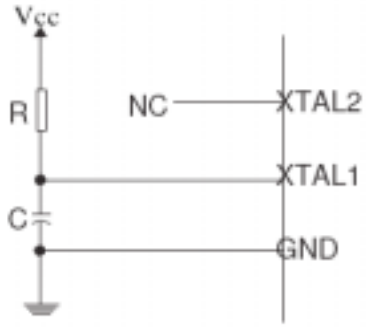


Figura 3.11. Configuración de una red RC externa.

Una última alternativa para una fuente de señal de reloj hacia con el microcontrolador, es contar con una fuente totalmente externa, por ejemplo un generador de señales, cuyo valor de frecuencia deberá estar en el orden de los valores que se tienen para redes RC o un oscilador de cristal de cuarzo, como se indica en la figura 3.12. En cualquier caso, se considerará una tolerancia entre el 1% y 2%.

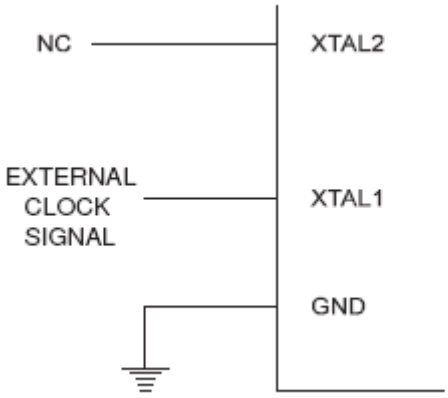


Figura 3.12. Configuración para una fuente de reloj externo.

3.2. 5. PUERTOS DE ENTRADA Y SALIDA

Todos los puertos de los microcontroladores AVR tienen la facilidad de poder funcionar como puertos de lectura o de escritura, según se necesite; además, cada puerto tiene asociados registros tanto de control como de datos que permiten su configuración, vía software, que les permite tomar la faceta de entrada o salida de datos, dichos puertos son:

- DDRx. Este registro configura el modo de operación del puerto. Escribir un "1" lógico a un bit de este registro, configura la terminal física correspondiente al bit como una salida. Escribir un "0" lógico lo hace una entrada.
- PINx. Este registro lee el estado de PORTx, independientemente del estado de DDRx. Fundamentalmente sirve para leer el estado de la terminal del puerto cuando ésta se ha configurado como entrada.
- PORTx. Si la terminal está configurada como salida, escribir un "1" o un "0" en el bit correspondiente de este registro, ocasiona que la salida en esta terminal sea un "1" o un "0". Si la terminal está configurada como entrada, escribir un "1" en el bit correspondiente de

este registro, habilita la resistencia de *pull-up.* Escribir un "0", estando configurado como entrada, deshabilita la resistencia de *pull-up*.

Las terminales de un puerto pueden llegar a tener funciones alternas dentro de la configuración de cada microcontrolador AVR, de hecho, pueden estar asociados a cualquiera de los subsistemas periféricos con los que cuente el modelo con el que se está trabajando. Por ejemplo, en el caso del modelo ATMEGA 32, las terminales del puerto A se pueden configurar como las terminales del convertidor analógico a digital, como lo muestra la tabla 3.2.

Terminal	Función alternativa
PA7	ADC7. Canal de entrada 7 del ADC
PA6	ADC6. Canal de entrada 6 del ADC
PA5	ADC5. Canal de entrada 5 del ADC
PA4	ADC4. Canal de entrada 4 del ADC
PA3	ADC3. Canal de entrada 3 del ADC
PA2	ADC2. Canal de entrada 2 del ADC
PA1	ADC1. Canal de entrada 1 del ADC
PA0	ADC0. Canal de entrada 0 del ADC

Tabla 3.2. Función alterna de las terminales del puerto A en el ATMEGA32.

3.2.6. INTERRUPCIONES

La base de una interrupción es la necesidad de un dispositivo periférico de enviar información al procesador principal del sistema. A nivel operativo, una interrupción tiene la ventaja de que delega la responsabilidad de comunicarse con el procesador al dispositivo periférico en lugar de gastar tiempo de operación en monitorear el estado de dicho periférico.

Las interrupciones en los microcontroladores AVR les permiten actuar solamente ante la ocurrencia de un evento en específico, sea de origen externo, por medio de hardware, por ejemplo con interruptores de tipo botón conectados a las terminales de interrupción externa correspondientes o bien, de origen interno, por medio de software, por ejemplo, debido al sobreflujo en el registro de cuenta de un temporizador o al estado lógico que guarde un bit en específico dentro del registro de control de algún periférico. Todo AVR cuenta con un *set* de interrupciones básicas, como las

interrupciones externas, interrupciones asociadas con contadores internos, las interrupciones generadas por comunicaciones seriales y una, de las más importantes, es el de restablecimiento (RESET), cuya función es inicializar todo el sistema. Un circuito práctico para la interrupción por *reset* se muestra en la figura 3.13.

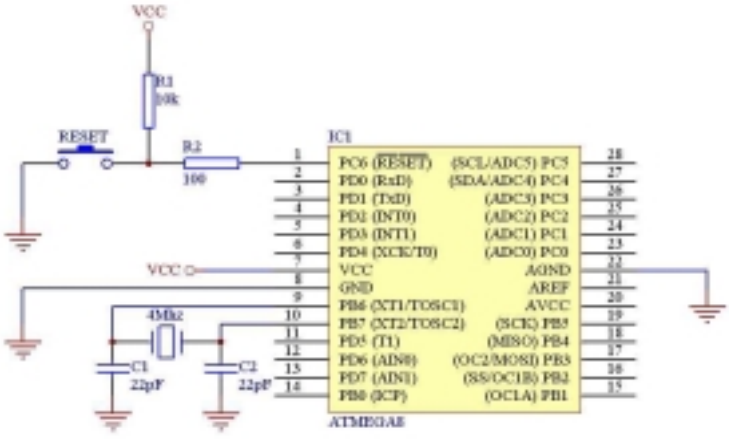


Figura 3.13. Circuito de interrupción externa por reset.

En la figura 3.13 podemos observar que el microcontrolador podrá ser sujeto de una interrupción externa de tipo *reset* externo, a través de la acción del interruptor de botón, en la que se aplica un nivel lógico bajo directamente a la terminal de *reset.*

Cada modelo de microcontrolador tiene asociada una lista de vectores de interrupción que, en algunos casos, están relacionados directamente con algunos de los subsistemas periféricos con los que dispone el modelo, en la hoja de especificaciones del fabricante, para cada modelo, puede encontrarse esta lista. Esta lista la podemos encontrar en la hoja de especificaciones del fabricante del microcontrolador, por ejemplo, para el modelo ATMEGA128, la tabla de vectores de interrupción es la que se muestra en la tabla 3.3.

Vector No.	Program Address[2]	Source	Interrupt Definition
1	$0000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	TIMER2 COMP	Timer/Counter2 Compare Match
11	$0014	TIMER2 OVF	Timer/Counter2 Overflow
12	$0016	TIMER1 CAPT	Timer/Counter1 Capture Event
13	$0018	TIMER1 COMPA	Timer/Counter1 Compare Match A
14	$001A	TIMER1 COMPB	Timer/Counter1 Compare Match B
15	$001C	TIMER1 OVF	Timer/Counter1 Overflow

Tabla 3.3. Vectores de interrupción para el microcontrolador ATMEGA128. (CONTINÚA)

16	$001E	TIMER0 COMP	Timer/Counter0 Compare Match
17	$0020	TIMER0 OVF	Timer/Counter0 Overflow
18	$0022	SPI, STC	SPI Serial Transfer Complete
19	$0024	USART0, RX	USART0, Rx Complete
20	$0026	USART0, UDRE	USART0 Data Register Empty
21	$0028	USART0, TX	USART0, Tx Complete
22	$002A	ADC	ADC Conversion Complete
23	$002C	EE READY	EEPROM Ready
24	$002E	ANALOG COMP	Analog Comparator
25	$0030[1]	TIMER1 COMPC	Timer/Counter1 Compare Match C
26	$0032[1]	TIMER3 CAPT	Timer/Counter3 Capture Event
27	$0034[1]	TIMER3 COMPA	Timer/Counter3 Compare Match A
28	$0036[1]	TIMER3 COMPB	Timer/Counter3 Compare Match B
29	$0038[1]	TIMER3 COMPC	Timer/Counter3 Compare Match C
30	$003A[1]	TIMER3 OVF	Timer/Counter3 Overflow

Vector No.	Program Address[2]	Source	Interrupt Definition
31	$003C[1]	USART1, RX	USART1, Rx Complete
32	$003E[1]	USART1, UDRE	USART1 Data Register Empty
33	$0040[1]	USART1, TX	USART1, Tx Complete
34	$0042[1]	TWI	Two-wire Serial Interface
35	$0044[1]	SPM READY	Store Program Memory Ready

Tabla 3.3. Vectores de interrupción para el microcontrolador ATMEGA128.

3.2.7. USART

El transmisor y receptor serial asíncrono/síncrono universal (USART), es un subsistema muy flexible de los microcontroladores AVR, principalmente de la gama media y alta, que permite, entre otras funciones, las siguientes:

- Operación full-dúplex (envío y recepción simultáneos)
- Operación síncrona y asíncrona
- Operación en modo maestro y esclavo con reloj asíncrono
- Soportar tramas (*frames*) de 5,6,7,8 y 9 bits de datos y 1 ó 2 bits de parada
- Generador de paridad par e impar
- Detección de errores (*data overrun, error frame*)
- Filtrado de ruido
- Generación de interrupciones por transmisión completa, por recepción completa o por registro de datos de transmisión vacío
- Comunicación multiprocesadores
- Doblador de velocidades en modo de comunicación síncrona

Para poder acoplar este subsistema a estándares de comunicación como el RS232, RS485 o el CANBUS, se requiere de dispositivos intermedios, como transceptores, que gestionen precisamente ese acoplamiento. De los transceptores más utilizados están los circuitos integrados MAX 232 y el MAX3222, cuya función es acoplar los niveles de tensión que se obtienen del subsistema USART, generalmente tipo TTL (0 a 5Vdc) con los niveles que maneja la norma RS232 (-12 a 12 Vdc), lo cual logra primero, duplicando los niveles TTL y después alternándolos. Un esquema de conexión típico, utilizando los circuitos integrados MAX232 y MAX3222, se muestran en las figuras 3.14 y 3.15.

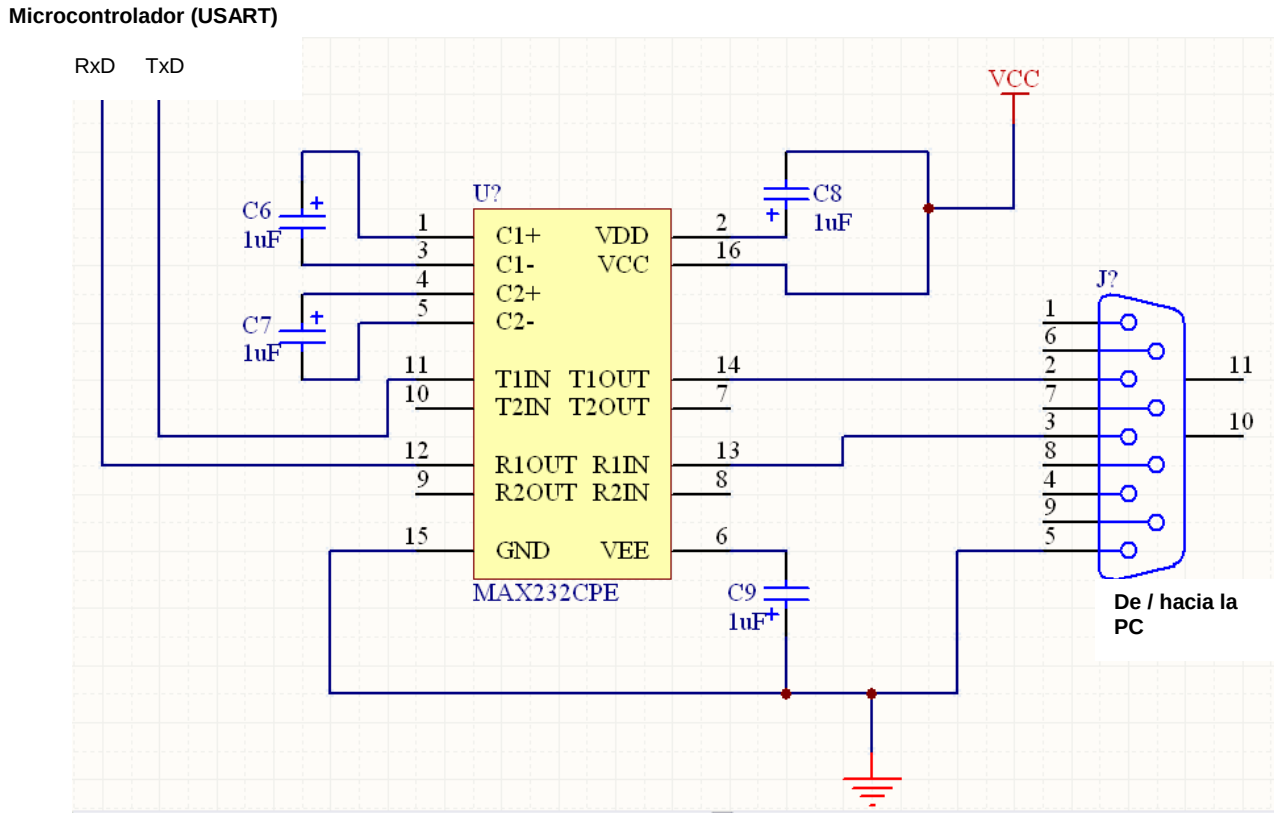


Figura 3.14. Configuración típica para el transceptor MAX232 y un microcontrolador.

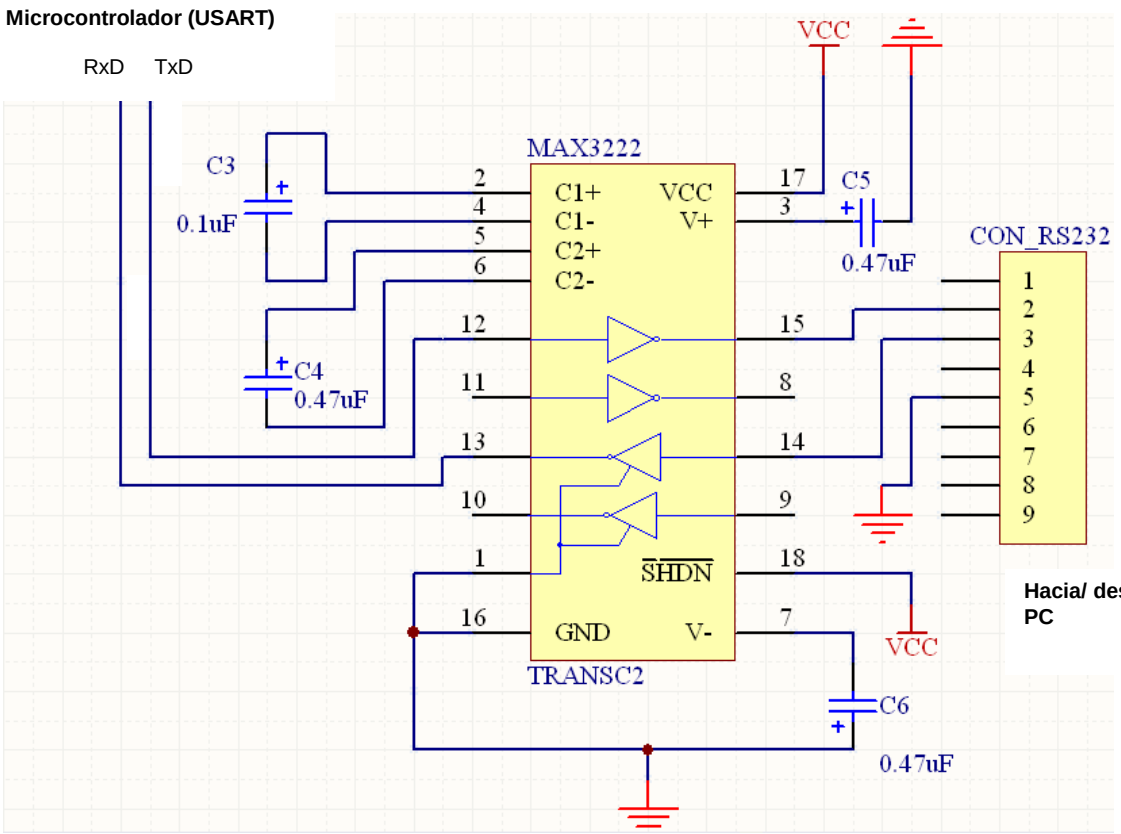


Figura 3.15. Transceptor MAX3222 y un microcontrolador.

Hasta aquí he presentado la descripción de algunas partes de los microcontroladores AVR, lo cual nos permite conocer, con cierto detalle, algunos puntos en cuanto a su arquitectura y las ventajas que tiene el utilizar uno de ellos en el desarrollo del prototipo de estación remota.

3.3. UNIDAD DE ALIMENTACIÓN DE ENERGÍA

Un elemento crucial después de la selección del microcontrolador es la alimentación de energía de toda la estación, ya que de ello dependerá el buen funcionamiento de todo el sistema. Para la estación remota se van a definir dos tipos de fuentes de alimentación, la primera, compuesta por un panel solar que, mediante el uso de un controlador de carga comercial va a cargar una batería y la segunda, compuesta por un eliminador de voltaje, alimentado por la red convencional.

El panel solar con el que cuenta la estación, figura 3.16, está fabricado de silicio monocristalino, de 10 W de potencia, voltaje a potencia máxima de 16.5 Vdc, corriente a potencia máxima de 0.60 A, voltaje a circuito abierto 20 Vdc, corriente de corto circuito 0.70 A, el cual se conecta al controlador de carga para la carga de la batería.

Figura 3.16. Módulo de captación solar de la estación remota.

El controlador de carga, figura 3.17, tiene las siguientes características:

- Está diseñado específicamente para trabajar con sistemas de captación solar
- Cuenta con un banco de terminales de fácil acceso para conexión
- Cuenta con un fusible de protección contra sobrecargas de salida
- Tiene un indicador luminoso de actividad solar
- Tiene un indicador luminoso de desconexión y reconexión de la carga

Figura 3.17. Controlador de carga para el panel solar.

El funcionamiento del controlador de carga consiste en sensar los niveles de voltaje en el banco de baterías para tomar las acciones que eviten una sobrecarga o descarga profunda que lleven a tener un mejor rendimiento en el sistema fotovoltaico.

La sobrecarga de la batería se alcanza cuando el controlador detecta un nivel de 14.5 ±2 Vdc, lo cual se evita con la desconexión del módulo solar, una vez desconectado el módulo solar, el nivel de la batería baja hasta 14.1 ±2 Vdc, momento en el cual, el controlador activa nuevamente el módulo solar, reiniciándose nuevamente la carga de la batería.

La descarga de la batería se controla por medio de la desconexión de la salida de carga, lo cual ocurre cuando el controlador sensa un nivel de voltaje de 11.3±2 Vdc. La salida de carga se restablece una vez que la batería ha superado su nivel de voltaje nominal y el controlador sensa un voltaje de 12.3 ±2 Vdc. Las conexiones físicas entre el controlador de carga y la batería en el interior de la estación remota se muestran en la figura 3.18.

Figura 3.18. Conexiones entre el controlador de carga y la batería en la estación remota.

El voltaje de la batería de 12 Vdc, se conecta a una bornera de la cual se toma la alimentación para la tarjeta de adquisición de datos, el sensor y para los medios de comunicación, como el radio módem. Los circuitos integrados que se encuentran en la tarjeta de adquisición de datos como el microcontrolador, el reloj de tiempo real, la memoria EEPROM, los transceptores USB y RS232 así como la sonda que contiene el sensor de temperatura se alimentan con voltajes de 5 y 2.5 Vdc respectivamente. Por lo tanto, resulta necesario abatir los 12 V que entrega la batería, para lo cual

se utilizan reguladores fijos y variables de voltaje, que permiten obtener niveles de tensión estables que alimenten a los dispositivos.

Los reguladores de voltaje, variables y fijos, utilizados en el segundo tipo de fuente de alimentación que tiene la estación remota, son de tecnología conmutada de tipo *buck,* es decir, reductores. El motivo de la elección de reguladores conmutados sobre otros, como por ejemplo los lineales, está basado en su alta eficiencia, alrededor del 75%, y su mínimo desprendimiento de calor aún a plena carga, lo cual redunda en la reducción del tamaño del disipador de calor y en algunos casos, en su omisión. Fundamentalmente en la arquitectura de los reguladores lineales se utilizan transistores operando en la zona lineal, conectados en serie con la corriente de la carga. Las ventajas que presenta este tipo de reguladores es un voltaje de rizo de salida muy pequeño además de poco ruido y su implementación es sencilla, sin embargo debido a la operación del transistor en la zona lineal, se comporta como una resistencia variable y su eficiencia es muy baja, típicamente alrededor del 40%. En los reguladores de voltaje conmutados, en lugar de controlar una resistencia variable, la salida del regulador es controlada mediante la rápida conmutación de transistores. El ciclo de trabajo de la señal producida por la conmutación, establece cuanta carga se transfiere a la carga, la cual es controlada por un mecanismo de retroalimentación. La eliminación de la resistencia variable permite incrementar la eficiencia eléctrica que ofrecen.

Para la estación remota se utilizó el regulador conmutado LM2575 en configuración fija y ajustable, los circuitos que se utilizaron e implementaron se muestran en las figuras 3.19 a 3.22.

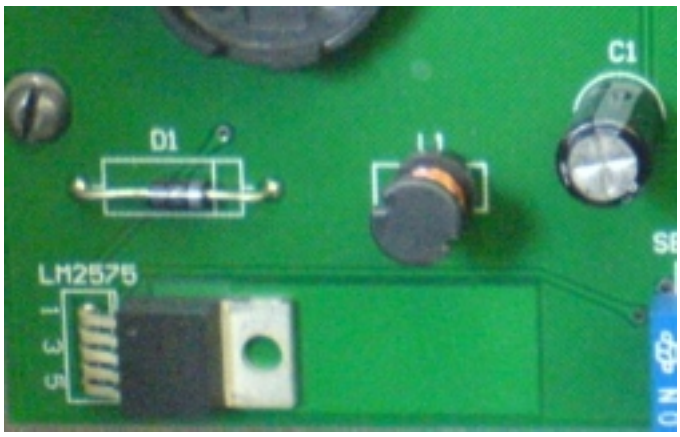


Figura 3.19. Implementación del regulador conmutado LM2575.

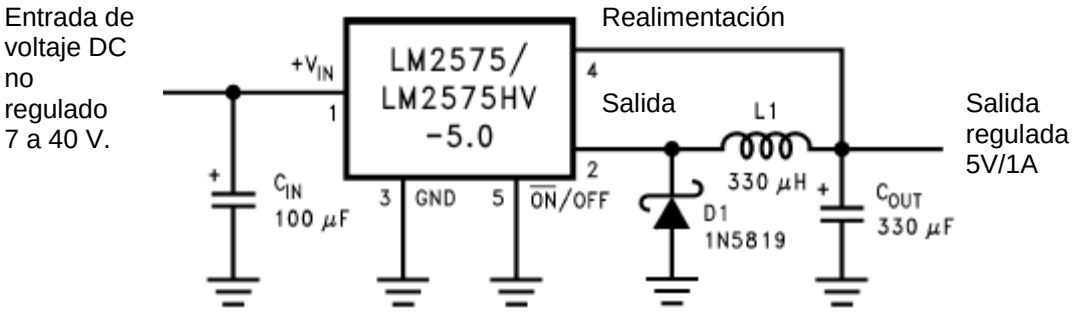


Figura 3.20. Configuración del regulador conmutado fijo 5V/1A.

Figura 3.21. Implementación de la fuente variable de 2.5V.

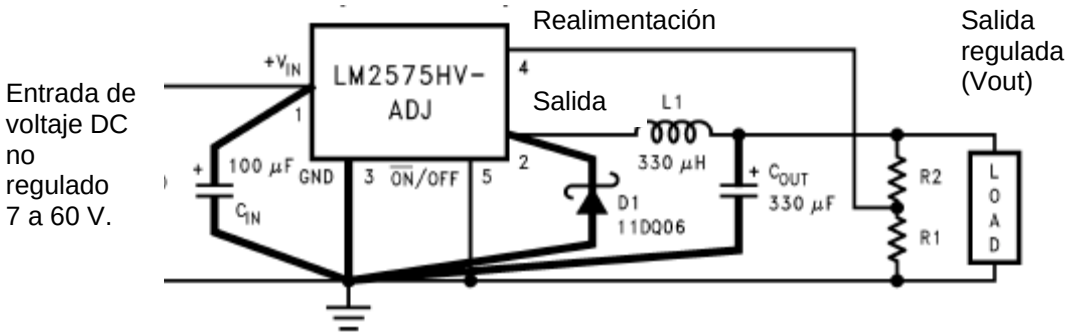


Figura 3.22. Configuración para el regulador conmutado variable.

Para diseñar los valores de los elementos indicados en el circuito de la figura 3.22, se aplican las ecuaciones 3.1 y 3.2 siguientes:

$$V_{out} = V_{REF}\left(1+\frac{R2}{R1}\right) \qquad \text{Ecuación 3.1.}$$

$$R2 = R1\left(\frac{V_{out}}{V_{REF}}-1\right) \qquad \text{Ecuación 3.2.}$$

Según el fabricante, se debe considerar para V_{REF} = 1.23 V y que 5kΩ>R1>1kΩ.

En el caso de la estación remota, considerando R1 = 4k7 con V_{OUT} = 2.56 V, utilizando las ecuaciones 3.1 y 3.2, se tiene:

$$R_2 = 4.7k\left(\frac{2.56}{1.23}-1\right) = 5.082k\Omega \quad \Rightarrow \quad V_{out} = 1.23\left(1+\frac{5.082k}{4.7k}\right) = 2.5599V \approx 2.56V$$

Una característica adicional con que cuenta este circuito integrado es que, la terminal marcada con $\overline{ON}$ /OFF, está disponible para poder ser utilizada por dispositivos de control como por ejemplo, microcontroladores, que tomen el control del regulador y que por medio de señales lógicas enviadas a través de una terminal del microcontrolador pueda encender o apagar al regulador. En los circuitos de las figuras 3.20 y 3.22 esta terminal se conectó a tierra con objeto de inhibir esta modalidad y permitir el funcionamiento normal del circuito integrado.

3.4. UNIDAD DE RELOJ DE TIEMPO REAL

La unidad de reloj de tiempo real de la estación remota tiene la función de asociar a los datos adquiridos una referencia de tiempo, con objeto de identificar las horas, los minutos, los segundos, el mes del año, el día de la semana en que fue adquirido cada uno de los datos que se envían a la estación base procedentes de la estación remota.

El circuito integrado que lleva a cabo la función mencionada en el párrafo anterior es el DS1307, el cual es un reloj de tiempo real. El circuito DS1307 se comunica con el microcontrolador por medio del bus serie I^2C (TWI en los microcontroladores AVR). Cuenta con 8 registros de 8 bits

para la configuración de hora, minutos, segundos, fecha, día de la semana, mes y año, además de un espacio de 56 Bytes de memoria SRAM no volátil y un generador de pulsos de frecuencia programable. Un mapa de direcciones de memoria del DS1307 y el detalle de configuración de los registros se presentan en las figuras 3.23 y 3.24.

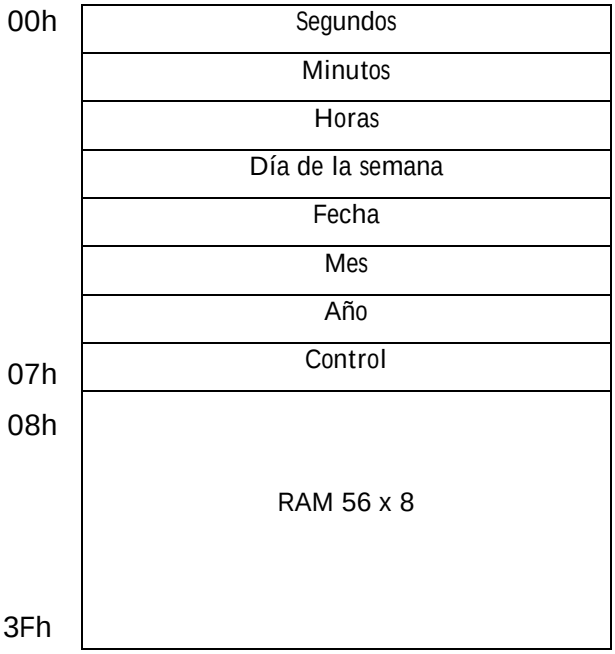


Figura 3.23. Mapa de direcciones y registros para el DS1307.

Dirección	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
00h	10 segundos				Segundos			
01h	10 minutos				Minutos			
02h	12	24	10 horas am/pm	10 horas	Horas			
03h	0	0	0	0	0	Día		
04h	0	0	Fecha		Fecha			
05h	0	0	0	Mes	Mes			
06h	Año				Año			
07h	Salida	0	0	SQWE	0	0	RS1	RS0

Figura 3.24. Descripción de los registros del DS1307.

La programación del DS1307 se hace por software utilizando los registros mostrados en la figura 3.23 y descritos en la figura 3.24. El hardware extra que se necesita para poner en funcionamiento al reloj en tiempo real es un cristal de cuarzo estándar, de bajo costo, de 32.768 kHz entre las terminales 1 y 2 del circuito integrado, figura 3.25, lo que le proporciona una base de tiempo exacta. Opcionalmente, se le puede conectar en la terminal 3, una batería de respaldo de 3 Vdc, asegurando que se mantendrán los datos cargados en los registros aunque esté desconectada la fuente de alimentación de energía del circuito principal. El circuito integrado DS1307 posee un mecanismo que automáticamente detecta la remoción de energía del circuito principal y obtiene la energía necesaria de las baterías de respaldo. Una batería de respaldo estándar tiene una duración aproximada de 10 años.

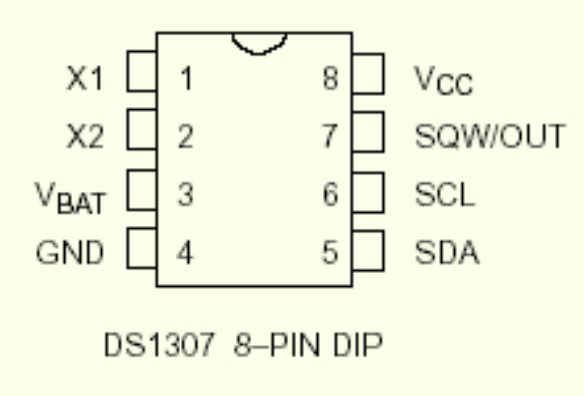


Figura 3.25. Diagrama del reloj en tiempo real DS1307.

Adicionalmente el circuito integrado DS1307 posee dos características interesantes: la terminal 7 es una salida a colector abierto, que puede ser programada para generar pulsos cada 1 Hz. Esto permite de un indicador luminoso (led) para monitorear las pulsaciones cada segundo, finalmente, el DS1307 cuenta con 57 bytes de memoria RAM de propósito general. Las terminales 5 y 6, correspondientes a SDA y SCL, permiten la conexión del circuito con un microcontrolador maestro, a través de las líneas de datos y de reloj, correspondientes al protocolo TWI.

Para la conexión del reloj en tiempo real con el microcontrolador central, se utiliza el *bus* TWI, cuya comunicación la gestiona el dispositivo maestro que, para la estación remota, es el microcontrolador ATMEGA128. El hardware asociado a esta conexión son dos resistencias, cuyo valor se calcula conforme a la ecuación 3.3 que está en función de la capacitancia del *bus* y del tiempo en el que dura el cambio de flanco entre las señales SDA y SCL del *bus* TWI.

$$R_{PU} = \frac{t_r}{C_b}$$ Ecuación 3.3.

De donde:

t_r: es el tiempo que dura el cambio de flanco entre las señales SDA y SCL en el bus TWI y cuyo valor típico es de 1000 ns.

C_b : es la capacitancia de cada uno de los *buses* TWI con que se comunica el reloj esclavo con el microcontrolador maestro. Tiene un valor típico de 400 pF.

Sustituyendo en la ecuación anterior los valores mencionados, tenemos:

$$R_{PU} = \frac{1000x10^{-9}}{400x10^{-12}} = 2.5\ \text{k}\Omega$$

La conexión de las resistencias R_{PU}, se muestran en la figura 3.27, como se puede apreciar en ella, ambas son conectadas a Vcc, y en la figura 3.27, observamos su implementación física en la tarjeta de adquisición de la estación remota.

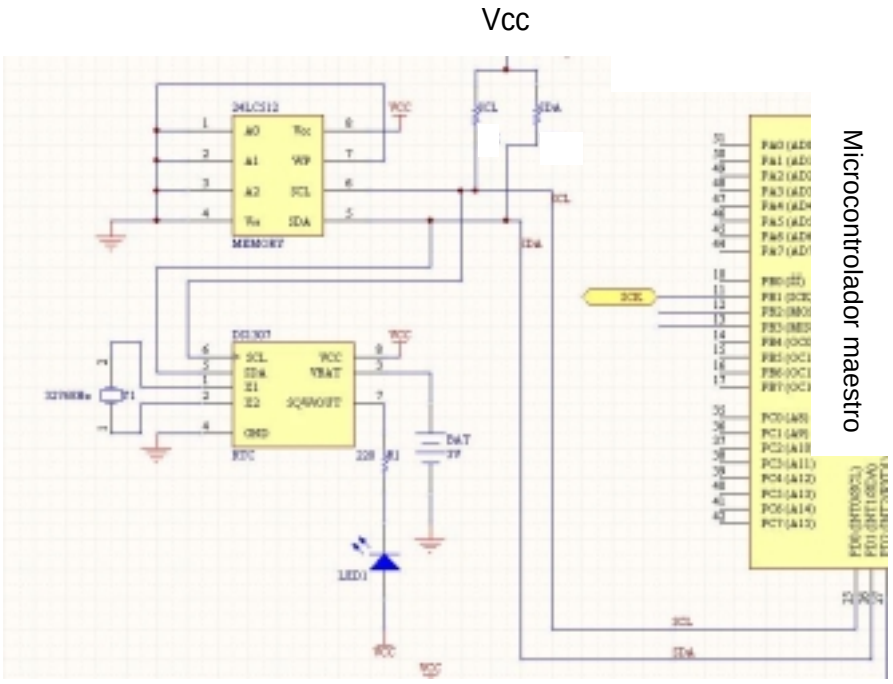


Figura 3.26. Conexión entre el microcontrolador maestro y los circuitos DS1307 y 24LC512.

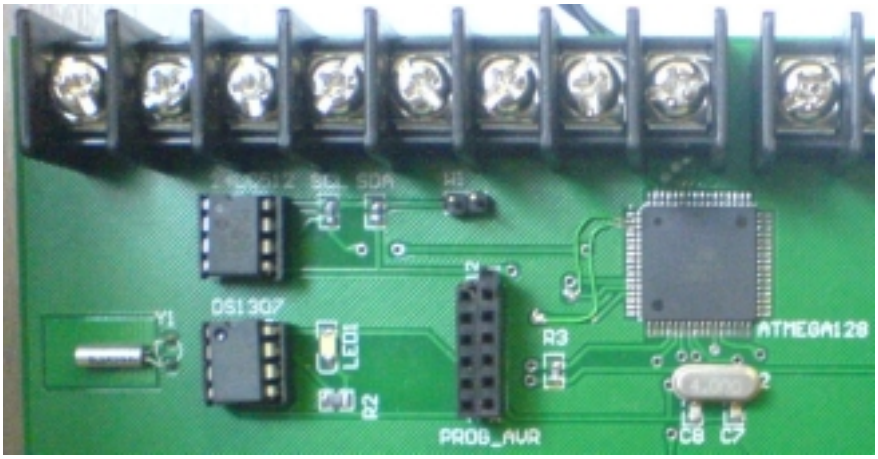


Figura 3.27. Implementación física del reloj DS1307 y la memoria 24LC512.

3.5. UNIDAD DE MEMORIA

La unidad de memoria de datos de la estación remota la integra el circuito integrado 24LC512, el cual es una memoria serie de tipo 64 k x 8 EEPROM, esto es, una memoria eléctricamente borrable, capaz de operar en un amplio intervalo de voltajes, de 2.5 a 5.5 V y a una frecuencia máxima de reloj de 400 kHz.

El circuito integrado 24LC512 permite realizar lecturas aleatorias y secuenciales de hasta 512 Kb, es decir, el espacio total direccionable.

La capacidad de la memoria puede ser ampliada hasta 4 Mb, mediante la utilización de tres líneas adicionales de direccionamiento, que permiten conectar hasta ocho dispositivos en un mismo *bus*, figura 3.28.

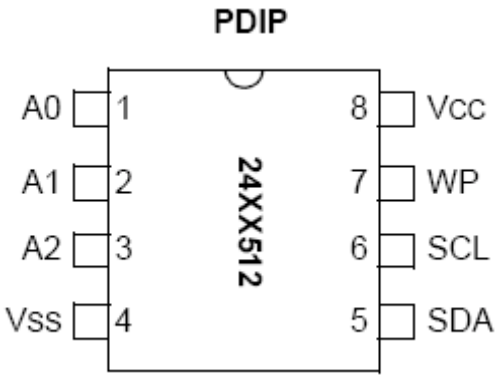


Figura 3.28. Memoria EEPROM 24LC512.

Algunas de las características del circuito integrado 24LC512 son las siguientes:

- Tecnología CMOS de baja potencia. La corriente máxima en escritura es de 5 mA a 5.5 V. La corriente máxima en lectura es de 400 μA a 5.5 V. La corriente de espera (STANDBY) es 100nA a 5.5 V.
- El *bus* de comunicaciones con un dispositivo maestro está formado por dos hilos, con una interfaz serie, compatible con el estándar I^2C (TWI).
- Protección de escritura por hardware.
- En un mismo ciclo pueden ser realizadas operaciones de lectura y escritura.
- La capacidad de cada página de escritura es de 128 bytes.
- Circuito interno supresor de ruido.
- Hasta 1, 000,000 de ciclos de lectura y escritura.
- El tiempo máximo de escritura por ciclo es de 5 ms.
- El circuito integrado cuenta con una protección contra las descargas de electricidad estática de 4 kV.
- Tiempo máximo de permanencia de datos de 200 años.
- Intervalos de temperaturas de funcionamiento: industrial, -40°C a 85°C, y automotriz -40°C a 125 °C.

El circuito integrado 24LC512 cuenta con 8 terminales y está disponible en encapsulados DIP, SOIC, DFN y en TSSOP, este último de 14 terminales. Las terminales más importantes del integrado, cuyo esquema se muestra en la figura 3.28, son las entradas A0, A1 y A2, la terminal de datos serie SDA, la terminal de reloj serie SCL y la terminal de protección contra escritura WP. Sus características más relevantes son:

- Entradas *A0, A1 y A2*. Las entradas *A0, A1 y A2* son utilizadas por el circuito 24LC512 para las operaciones con múltiples dispositivos. Los niveles lógicos en estas entradas corresponden a los bits de la dirección del esclavo. El circuito es seleccionado si y sólo si estos bits coinciden. Pueden conectarse hasta 8 dispositivos en el mismo *bus* usando combinaciones de bits del selector de circuito (*chip select*). Si estas terminales se dejan sin conectar, las entradas serán internamente conectadas a Vss mediante un circuito interno de *pull-down.* En la mayoría de las aplicaciones, como en el caso de la estación remota, las direcciones de entrada *A0, A1 y A2* son conectadas a un nivel lógico "0" ó "1". Para los usos en los cuales el control de estas

terminales es gestionado por un microcontrolador, u otro tipo de dispositivo de lógica programable, las terminales de selección de circuito deben ser puestos a un nivel lógico de "0" ó "1" antes de que la operación normal del circuito comience.

- Terminal de datos seriales (SDA). Esta es una terminal bidireccional que se usa para transferir direcciones y datos de entrada y salida de los dispositivos. Se trata de una terminal en colector abierto, por lo cual, el bus SDA requiere de una resistencia de *pull-up* para conectarse con Vcc (típicamente 10 kΩ para una frecuencia de transferencia de 100 kHz, 2 kΩ para 400 kHz y 1 MHz). Para una transferencia de datos normal, la terminal SDA permite el cambio sólo durante el nivel bajo de la señal SCL. Los cambios durante el estado alto de SCL están reservados para marcar las condiciones de START y STOP como parte del protocolo TWI.
- Reloj serie (SCL). Esta terminal se utiliza para sincronizar la transferencia de datos con el dispositivo maestro.
- Protección contra escritura. Esta terminal puede ser conectada a Vss o Vcc. El circuito interno de *pull-down* de esta terminal, mantendrá al dispositivo en un estado de no protección si permanece flotado o conectado a Vss, sin embargo, no se recomienda hacerlo. Si WP se conecta a Vcc, las operaciones sobre la memoria pueden ser realizadas, lectura o escritura de la memoria completa desde 0000 a FFFF.

En cuanto a la arquitectura interna del 24LC512 es posible resaltar el papel que juegan los registros internos. Dentro de los registros internos del 25LC512 podemos encontrar dos tipos de registros. Por un lado los registros de almacenamiento y por otro lado el registro de direcciones. Los registros que se pueden encontrar son los siguientes:

- Registro principal de almacenamiento. Tiene una capacidad de almacenamiento de 512 kb, distribuidos en 64,000 celdas de 8 bits cada una.
- *Buffer* de página. Es utilizado en el modo de escritura de página como *buffer* intermedio de almacenamiento, previo a la escritura en memoria de datos. Su tamaño es de 128 bytes.
- Registro de direcciones. Se trata de un puntero de dirección de 2 bytes, pues apunta a direcciones de memoria de esa misma dimensión.

Generalmente la forma de operación del 24LC512 se enmarcará dentro de un sistema "maestro-esclavo", donde el papel de maestro será usualmente ocupado por un microcontrolador y la memoria

en el de esclavo. El maestro controlará el *bus*, a través del cual tanto memoria como microcontrolador quedarán comunicados, generando la señalización necesaria y las condiciones de inicio y de paro.

De igual forma que el reloj de tiempo real, el 24LC512 transfiere sus datos y recibe órdenes del microcontrolador maestro por medio de los *buses* SDA y SCL del protocolo TWI. TWI es un protocolo serie, similar al I^2C, de hecho, algunas fuentes consultadas afirman que TWI es la primer versión del protocolo I^2C desarrollado por Phillips pero que por motivos de patente, ATMEL tuvo que bautizar con el nombre de interfaz de dos hilos o TWI. Para iniciar una comunicación entre dispositivos conectados al *bus* TWI, se debe respetar un protocolo. Tan pronto como el *bus* esté libre, un dispositivo maestro puede ocuparlo generando una condición de inicio (START). El primer byte transmitido después de la condición de inicio contiene 7 bits, que componen la dirección del dispositivo esclavo de destino seleccionado, y un octavo bit correspondiente a la operación deseada, lectura o escritura. Si el dispositivo cuya dirección se apuntó en los siete bits está presente en el *bus*, éste responde enviando un pulso o señal de confirmación (ACK). Inmediatamente después puede comenzar el intercambio de información entre los dispositivos.

El circuito 24LC512 puede operar de dos modos diferentes en cuanto a transmisión de datos se refiere, como transmisor o como receptor.

Cuando la señal de lectura o escritura (R/W) está previamente al nivel lógico bajo, el dispositivo maestro envía datos al dispositivo esclavo hasta que deja de recibir los pulsos de reconocimiento, o hasta que se hayan transmitido todos los datos. En el caso contrario, es decir cuando la señal de R/W está en el nivel lógico alto, el dispositivo maestro genera pulsos de reloj durante los cuales el dispositivo esclavo puede enviar datos. Luego de cada byte recibido el dispositivo maestro (que en este momento está recibiendo datos) genera un pulso de reconocimiento.

El dispositivo maestro puede dejar libre el *bus*, generando una condición de paro (STOP). Si se necesita seguir transmitiendo, el dispositivo maestro puede generar otra condición de inicio en lugar de una condición de paro. Esta nueva condición se llama de inicio repetido (REPEATED START) y se puede emplear para direccional un dispositivo esclavo diferente o para alterar el estado del bit R/W.

3.6. UNIDAD DE CONVERSIÓN ANALÓGICA A DIGITAL

La unidad de conversión analógica a digital (ADC), a pesar de no ser un módulo visiblemente aislado como en los casos descritos en los apartados anteriores, reviste una gran importancia dentro de la estructura funcional de la estación remota, ya que es la encargada de digitalizar las señales procedentes de los sensores hacia el microcontrolador, para su posterior procesamiento, almacenamiento y envío. La unidad de conversión analógica digital se encuentra embebida en el microcontrolador ATMEGA128. Está compuesta por 8 canales a los cuales se tiene acceso por medio de la bornera superior izquierda de la tarjeta principal de la estación remota, figura 3.29.

Figura 3.29. Entradas del convertidor A/D en la tarjeta principal.

El convertidor analógico a digital establece una relación entre su entrada y su salida, dependiendo de la resolución con que éste cuente. Con el valor de la resolución, si se conoce, se puede obtener el número de valores discretos que puede producir, el cual se suele expresar en bits, y que se asienta en la ecuación 3.4.

$$Número\ de\ valores\ discretos = 2^N$$ Ecuación 3.4.

Donde N es el número de bits de resolución del ADC.

Por ejemplo, un ADC que codifica a una entrada analógica en 1024 valores discretos tiene una resolución de 10 bits, ya que:

$$V_{\text{número de valores discretos ADC}} = 2^{10} = 1024$$

El convertidor analógico a digital del microcontrolador ATMEGA128 utiliza el método de conversión de aproximaciones sucesivas, el cual consiste en el uso de un comparador para rechazar voltajes que se van presentando, hasta que llega a un intervalo adecuado para ese nivel, al cual se le ha asignado un valor binario. Por ejemplo, la primera comparación pudiera determinar el bit más significativo (MSB) de la señal de salida, mientras que la siguiente comparación determinaría el segundo bit MSB y así, sucesivamente hasta llegar al bit menos significativo (LSB). Los convertidores de aproximaciones sucesivas se destacan por ofrecer una resolución elevada, así como por ofrecer velocidades de conversión relativamente altas, por lo que son utilizados en aplicaciones que demandan precisión y velocidad; por ejemplo, en los sistemas de adquisición de datos y controles de procesos, comúnmente utilizados en la automatización industrial. El ADC del microcontrolador ATMEGA128 cuenta con 8 canales de tipo *single-ended,* que pueden ser multiplexados, con estos mismos 8 canales se pueden formar 4 canales de tipo diferencial, en uno u otro caso la configuración se lleva a cabo vía software, habilitando los bits correspondientes a los registros de control y de datos del ADC.

En cuanto al tiempo de conversión, para realizar adecuadamente una conversión, la frecuencia del circuito de aproximaciones sucesivas debe estar entre 50 kHz y 200 kHz, para una resolución de 10 bits. Si es necesario un menor número de bits, se pueden utilizar frecuencias mayores a los 200 kHz. Para obtener frecuencias aceptables, el subsistema ADC contiene un circuito de preescala, configurable vía software. El convertidor puede configurarse de dos formas: *free running* y *single conversion*; la primera arranca una vez que se ha encendido el convertidor, comienza a realizar la digitalización de los datos analógicos, para el caso de la segunda forma de operación, se deben manejar, habilitar y deshabilitar y vigilar ciertos bits de un registro específico para el inicio de la conversión y el fin de la misma.

Una vez que se inicia una conversión, al ADC le toma 13 ciclos del reloj para llevarla a cabo, a excepción de la primera, que le toma 25 ciclos. La velocidad de conversión dependerá de una versión escalada del reloj de la CPU.

El voltaje de referencia en la terminal V_{REF} indica el margen de voltaje de la conversión; es importante agregar un voltaje de referencia estable, ya que éste es el que indica el nivel de voltaje correspondiente al máximo valor discreto de salida. Las referencias de voltaje pueden ser seleccionadas entre una referencia interna de 2.56, la cual se establece por software, o bien, referencias de voltaje externas, mediante su conexión en la terminal AREF. Si se utiliza la terminal de referencia AREF, se recomienda el uso de un capacitor entre la terminal y la tierra para aumentar la inmunidad al ruido del ADC. En el caso de la unidad de conversión analógica digital de la estación remota, se hace uso de la referencia interna de 2.56, aunque se encuentran disponibles conectores para cambiar esa referencia al voltaje de alimentación de la tarjeta, que es de 5V según se requiera, como se puede apreciar en la figura 3.30.

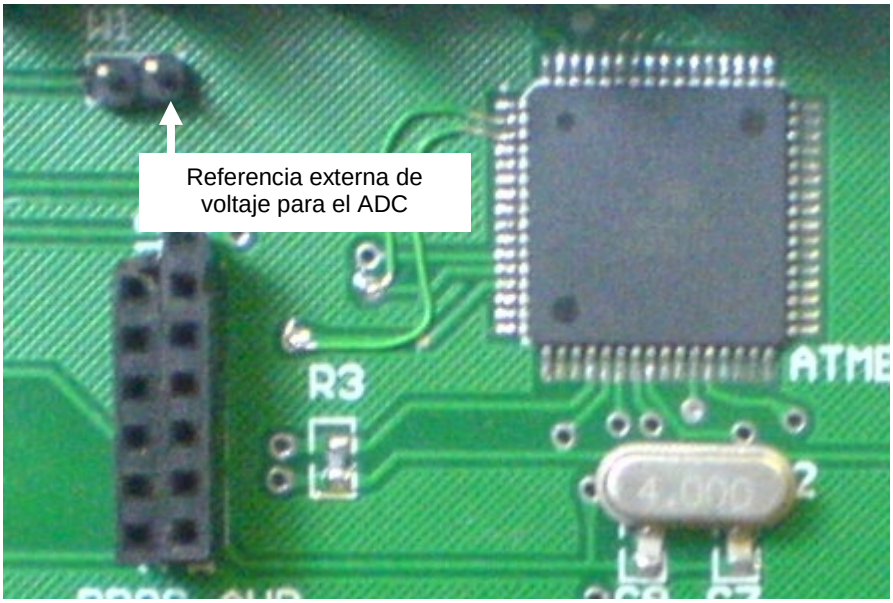


Figura 3.30. Conector para referencia externa del ADC.

3.7. INTERFACES DE COMUNICACIÓN

Uno de los subsistemas más importantes dentro de la arquitectura del hardware de la estación remota son, sin lugar a duda, las interfaces de comunicación. De forma general, el diseño de la estación remota plantea integrar dos tipos de protocolos de comunicación, uno de velocidad media, estándar en todavía algunos equipos de comunicación de corto y largo alcance como módems telefónicos y de radio frecuencia, y otro medio, de alta velocidad, de reciente desarrollo pero de notable crecimiento entre los equipos de comunicación, los protocolos de comunicación RS232 y USB, figura 3.31.

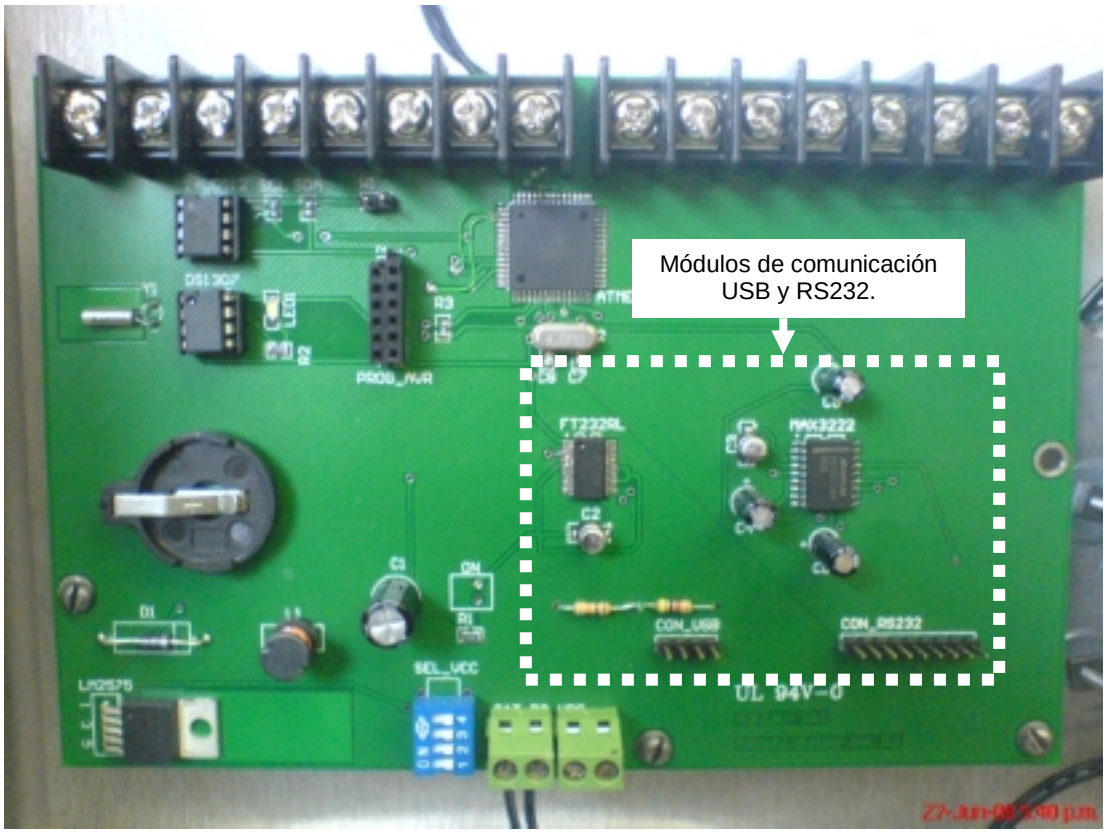


Figura 3.31. Protocolos de comunicación RS232 y USB.

Cada uno de los protocolos de comunicación que fueron integrados a la estación remota están basados en circuitos integrados especializados, los cuales interactúan, cada una de ellos, con las 2 unidades de comunicaciones seriales que posee el microcontrolador ATMEGA128, para el envío de datos hacia la estación base.

RS232

La comunicación a través del protocolo RS232 fue implementada en torno al circuito integrado MAX3222, cuya función es acoplar los niveles TTL, obtenidos en las terminales RX y TX del subsistema USART, con los niveles de -12 V a 12 V, propios del puerto serie de la computadora, o de los medios de comunicación integrados que cuenten con este protocolo, como se muestra en la figura 3.32.

Figura 3.32. Circuito integrado para el protocolo RS232.

El circuito integrado MAX3222 es un dispositivo transceptor, es decir, un dispositivo que realiza funciones tanto de envío como de recepción de señales, empleando elementos comunes del circuito para ambas direcciones de comunicaciones. La gama de voltajes de polarización, la opción de operación "en espera" (STANDBY) y el reducido número de componentes adicionales hacen de este integrado un dispositivo óptimo en aplicaciones embebidas, alimentadas a base de baterías, como en el caso de la estación desarrollada. El circuito mostrado en la figura 3.33 muestra la aplicación práctica de este circuito integrado, según la hoja del fabricante y en la tabla 3.4, el valor de los capacitores asociados a él.

Vcc (V)	C1 (uF)	C2, C3 y C4 (uF)
3.3 a 3.6	0.1	0.1
4.5 a 5.5	0.047	0.33
3.0 a 5.5	0.1	0.47

Tabla 3.4. Valores de capacitores recomendados por el fabricante.

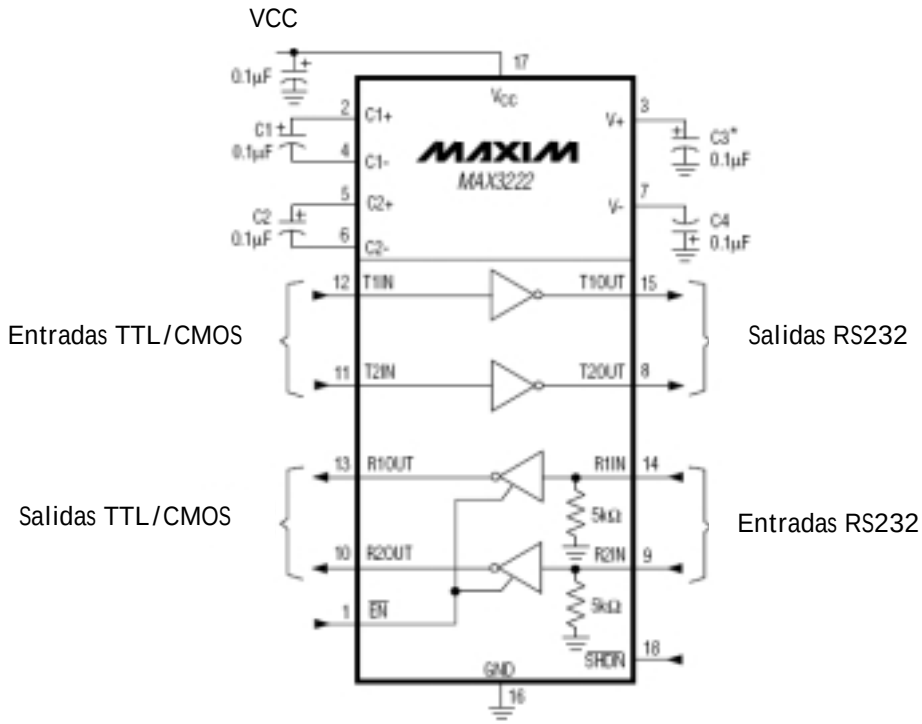


Figura 3.33. Aplicación típica del MAX3222.

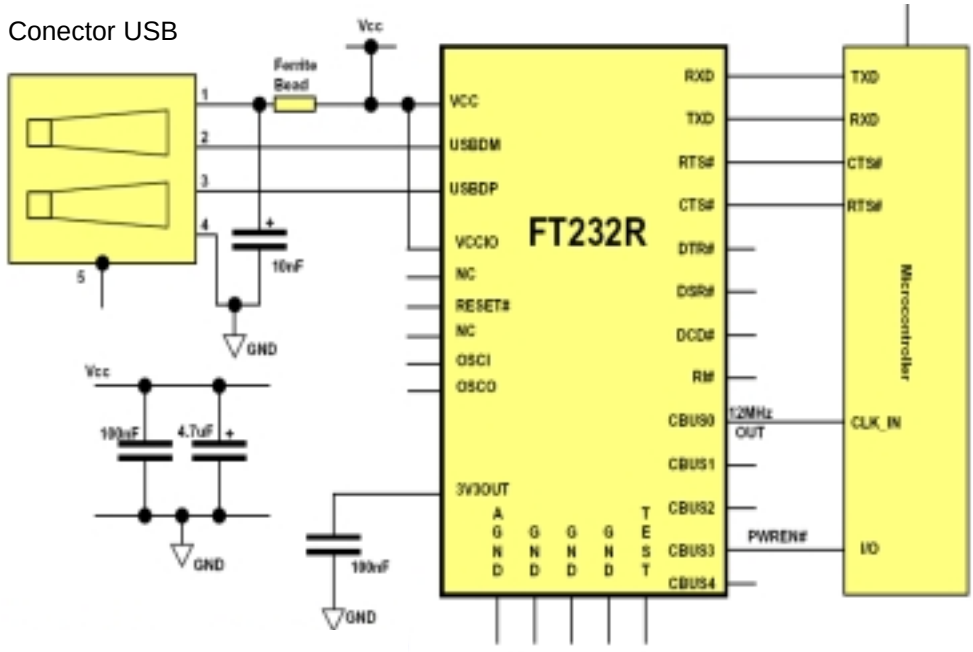
USB

El protocolo de comunicaciones USB fue desarrollado con base en el circuito integrado FT232RL. El FT232RL permite la comunicación de los datos entre una de las unidades del subsistema de comunicaciones seriales del ATMEGA128 y la estación base. El FT232RL fue seleccionado, además de por su fácil implementación práctica con un mínimo de componentes, por contar con controladores, herramientas de desarrollo y documentación de ayuda gratuitas y accesibles en el portal del fabricante. Las características técnicas del FT232RL son las siguientes:

- Un solo circuito integrado maneja tanto el protocolo USB como la transferencia serial síncrona.

- La UART del FT232RL admite datos en formato de 7 y 8 bits, 1 ó 2 bits de parada y otros mecanismos de control de flujo.
- Soporta tasas de transferencia de 300 Bauds a 1 MBauds en el protocolo RS232
- Soporta tasas de transferencia de 300 Bauds a 3 MBauds en los protocolos RS422, RS485 con niveles TTL.
- Soporta modo de bajo consumo (SLEEP) y modo en espera (STANDBY).
- Soporte para alimentar dispositivos directamente del *bus* USB a través de la terminal PWREN#.
- Circuito *power-on-reset* incluido.
- Voltaje de alimentación de 3.3 V a 5V.
- Compatible con el controlador de *host* UHCI/OHCI/EHCI.
- Compatible con los estándares USB 1.0 y USB 2.0.
- Registro en memoria, vía programación, de identificadores del circuito integrado y del equipo.
- Encapsulado pequeño SSOP-28.

En la figura 3.34 (a) se muestra su aplicación práctica dentro de la tarjeta principal de la estación remota y en la figura 3.34 (b) el diagrama recomendado por el fabricante.



(a)

(b)

Figura 3.34. (a) Conexión sugerida por el fabricante. (b) Circuito FT232RL en la tarjeta de la estación.

Cabe comentar que además de la comunicación *in situ*, la estación cuenta también con comunicación de largo alcance. Para comunicaciones de largo alcance se implementó el uso de módems de radio frecuencia. Se utilizaron 2 radio módems de RF, uno para la estación remota y otro para la estación base. Las características de los mencionados equipos se muestran a continuación, junto con una figura de ellos, figura 3.35.

Características técnicas del radio módem utilizado

- Alcance en interiores: 450 m.
- Alcance en exterior (con línea de vista): 11 km con antena de fábrica.
- Alcance en exterior (con línea de vista): 32 km con antena de alta ganancia.
- Potencia de transmisión: 100 mW (20 dBm).
- Tasas de transferencia de datos soportadas: 125 – 65,000 bps, configurable por software.
- Voltaje de alimentación: 7 a 18 V.
- Corriente en recepción: 70 mA.
- Corriente en transmisión: 170 mA.
- Frecuencia de transmisión: 902 a 928 MHz. Banda de radio aficionados, no necesita licencia por parte de la SCT.

Figura 3.35. Radio módem utilizado en la estación remota.

La selección de este equipo obedeció, principalmente, a su bajo costo, su muy bajo consumo de energía, su bajo voltaje de alimentación y la disponibilidad en el mercado nacional.

Los medios de comunicación *in situ* lo representan básicamente cables de transferencia de datos seriales y USB, que se conectan directamente con una computadora portátil y permiten, mediante el software diseñado, la adquisición de datos. Un componente opcional desarrollado, fue un convertidor USB a RS232, basado en el circuito integrado TUSB3410, que permitiría la conexión de una computadora que no tuviese integrado el puerto RS232 para la conexión con la estación, en el supuesto de un modelo de estación remota sin protocolo USB.

La arquitectura que sigue el circuito TUSB3410 es similar a la del antes tratado FT232RL, con la salvedad del requerimiento de un mayor número de componentes adicionales, como por ejemplo una base de tiempo, una propia fuente de alimentación y toda una serie de elementos resistivos y capacitivos que hacen más robusto el diseño utilizando este circuito, cuya circuito eléctrico se muestra en la figura 3.36 y su implementación práctica en la figura 3.37.

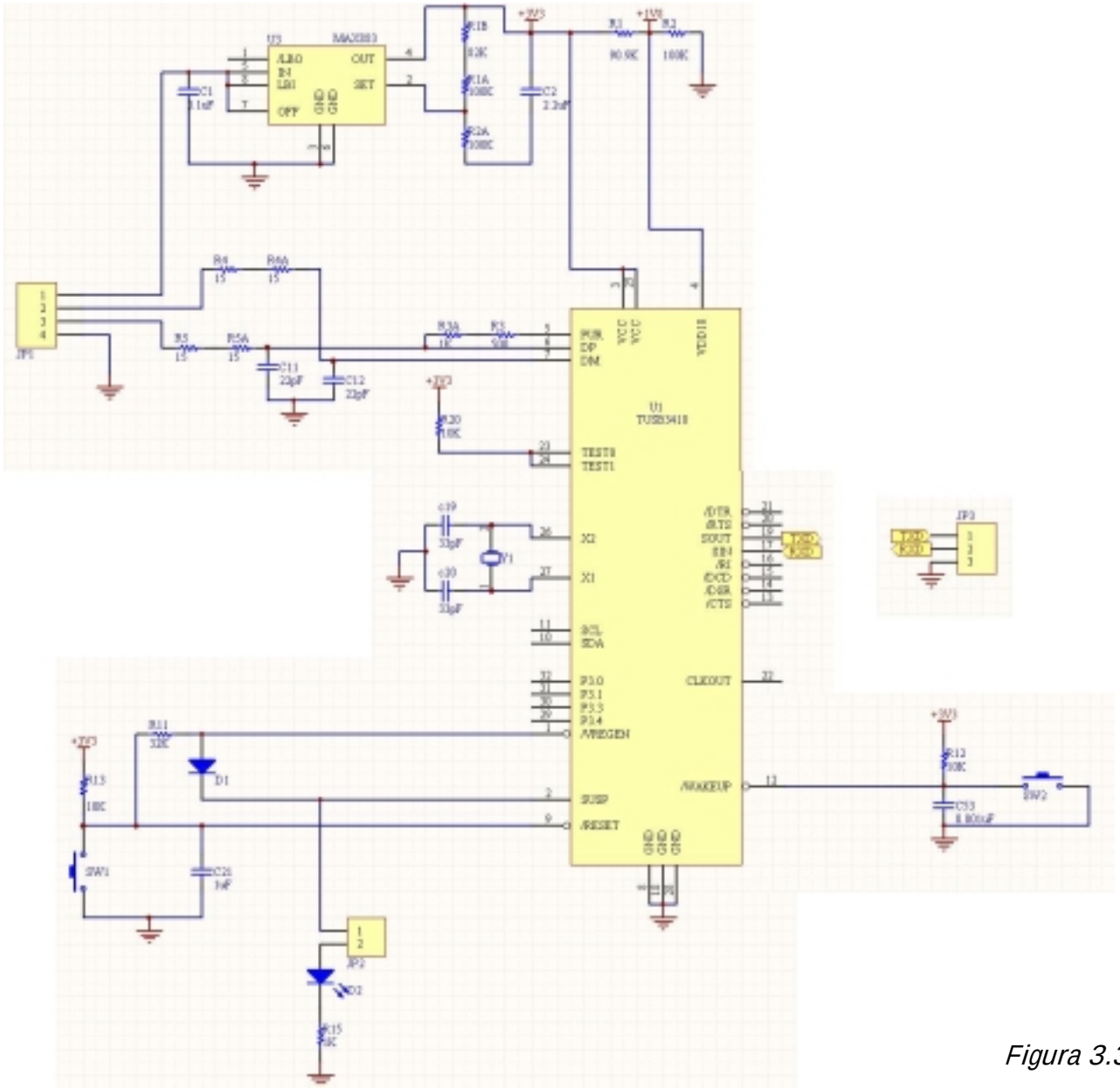


Figura 3.36. Convertidor USB a RS232

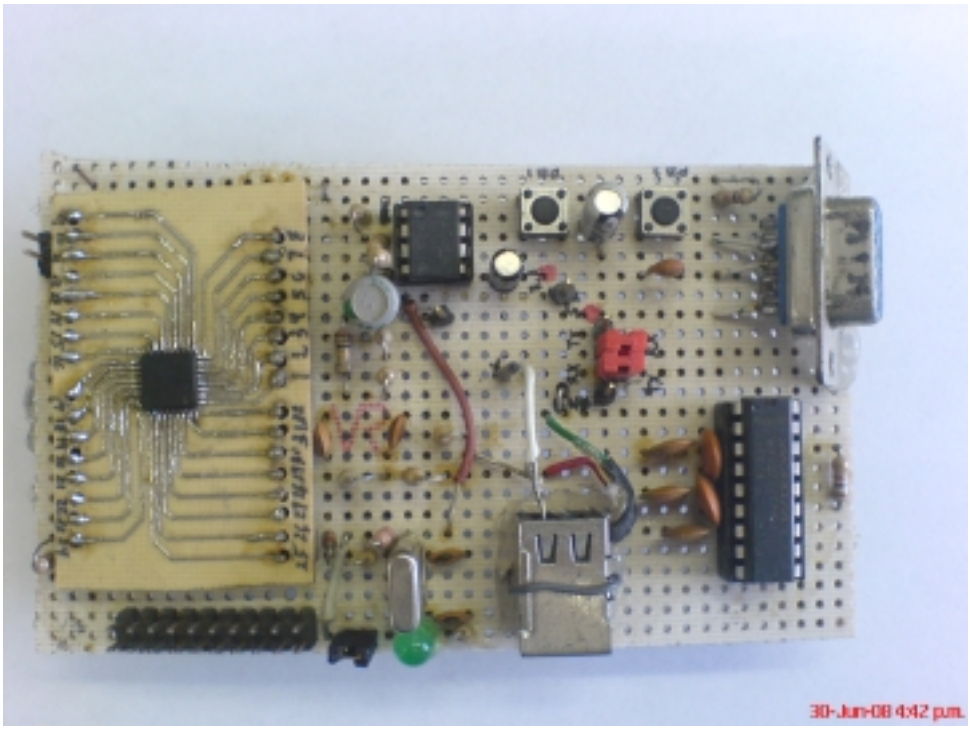


Figura 3.37. Convertidor USB a RS232.

3.8. SENSOR

La estructura planteada del proyecto "estación remota para fines de telemetría" plantea como filosofía principal el hecho de poder utilizar una arquitectura general básica, que permita aprovechar los módulos de adquisición, procesamiento, almacenamiento, alimentación de energía y envío de datos como una estructura funcional genérica, con la única particularización en la integración de los sensores, mismos que pueden ser del ámbito meteorológico, sísmico, hidrológico, etcétera, con, por supuesto, el acondicionamiento requerido de las señales de los sensores en cada caso.

Para el prototipo tratado en este trabajo, se integró un sensor de tipo meteorológico, en la forma de una sonda sensora de temperatura ambiente, figura 3.38, en cuyo interior, como elemento central de sensado, se encuentra un termistor.

Figura 3.38. Sonda de temperatura.

Para esta aplicación en particular, implicó ajustar la estación al ámbito meteorológico, sobre todo en lo referente a la adquisición, acondicionamiento y procesamiento de la señal proveniente del sensor. Aunque con un solo sensor, se trató en todo momento de respetar las disposiciones y estándares que el ámbito de aplicación tiene asociado, en este caso, las que tienen que ver con estaciones meteorológicas y que es emitida por la Organización Meteorológica Mundial (OMM), la cual marca como elemento de medición de temperatura ambiente, entre otros, al termistor, por poseer una alta sensibilidad al cambio de temperatura. En lo que corresponde al acondicionamiento de la señal proveniente de la sonda, ésta viene preparada con el termistor dentro de un arreglo en divisor de voltaje con resistencias de un valor tal que, compensan la no linealidad propia del termistor, y con cables listos para ser conectados a uno de los canales del convertidor analógico digital en configuración *single-ended.*

Otra de las normativas observadas fue la instalación de la sonda para el sensado de la temperatura ambiente, la cual se tiene que resguardar dentro de un recinto conocido como garita termométrica, la cual, según la "Guía de Instrumentos y métodos de observación meteorológicos", emitida por la OMM, permite el sensado preciso de la temperatura del aire y no de la propia sonda, debido a efectos de la radiación solar directa sobre ella, cuyo valor puede llegar a ser 25°C arriba de la del aire, siendo por

supuesto un valor distinto al que se requiere. Para tal efecto, la sonda se instaló dentro de un protector de radiación solar, figura 3.39, una versión miniatura de una garita termométrica en la cual se cumple con las normas de la OMM para la medición de la temperatura del aire. Una vista interior de la estructura del protector de radiación solar se muestra en la figura 3.40, y la instalación en la estructura de la estación se muestra en la figura 3.41.

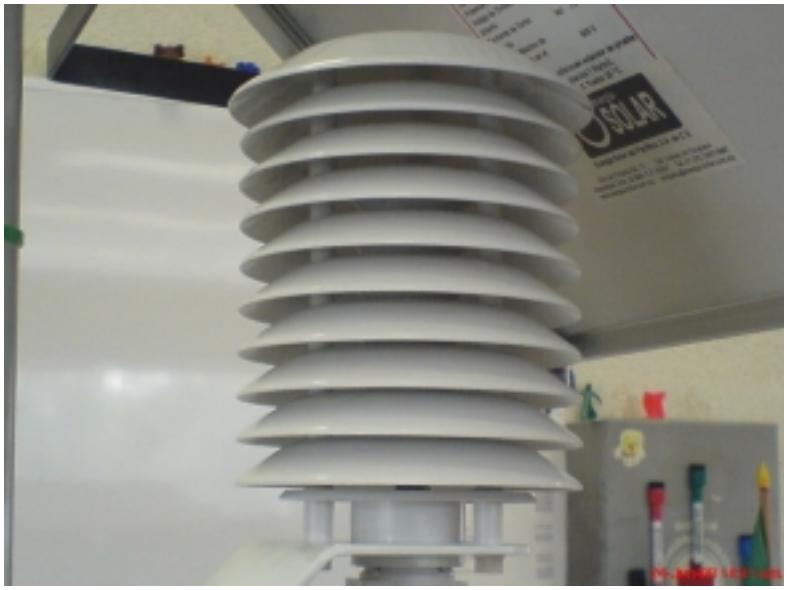


Figura 3.39. Protector de radiación solar.

Figura 3.40. Interior del protector de radiación solar.

Figura 3.41. Instalación en la estructura de la estación remota.

Hasta aquí se ha tratado todo lo referente al desarrollo e integración del hardware que compone a la estación remota, en el siguiente capítulo se presenta el software que complementa la estructura de la estación remota y a la estación base.

4.1. SOFTWARE DE LA ESTACIÓN REMOTA

El complemento del hardware, descrito en el capítulo anterior, lo representa el software, tanto de la estación base como de la estación remota el cual, en la estación remota, estará residente en la memoria de programa del microcontrolador ATMEGA128 y, en la estación base, estará residente en una computadora personal.

El software de la estación remota lo componen una serie de rutinas de adquisición, procesamiento, almacenamiento y envío de datos, todas ellas desarrolladas en lenguaje C y compiladas mediante el uso del compilador AVR GCC de la suite AVR STUDIO 4.12. El ambiente integrado de desarrollo (IDE) AVR STUDIO de ATMEL es el software que se utiliza para programar y depurar aplicaciones para microcontroladores AVR bajo las plataformas Windows 98, Windows XP, Windows ME, Windows 2000 y Windows NT. El compilador contiene una interfaz, mostrada en la figura 4.1, para el manejo de proyectos, un editor de código fuente y un emulador, éste último soporta una buena parte de modelos de la gama baja, media y alta de microcontroladores AVR.

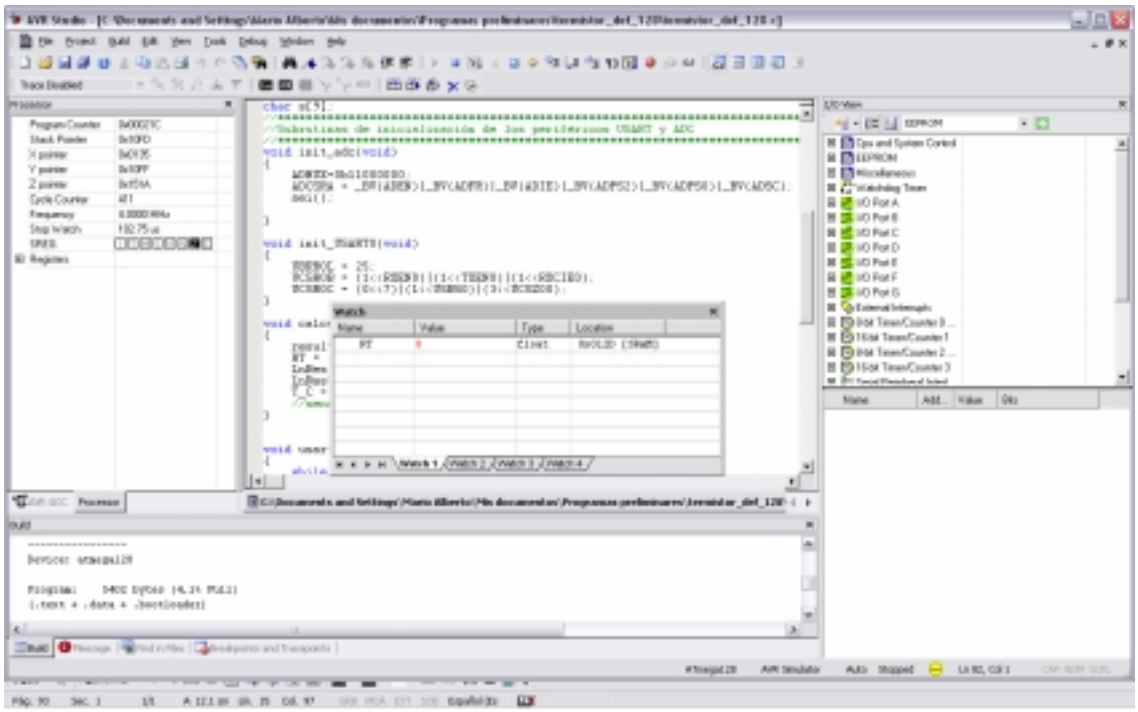


Figura 4.1. IDE de AVR STUDIO 4.12.

ATMEL, a través de su página de Internet, pone a disposición de los desarrolladores herramientas como WIN AVR, el cual es una suite de archivos ejecutables, de código abierto, para el desarrollo de aplicaciones basadas en microcontroladores AVR de ATMEL, que corre en la plataforma de Windows, el cual incluye el compilador GNU GCC para C y C++. WIN AVR, incluye todos los elementos necesarios para el desarrollo en microcontroladores AVR, incluyendo el compilador (AVR-GCC), el depurador (AVR-GDB) entre otros. El AVR STUDIO 4.12, o cualquiera de sus versiones anteriores sólo tienen el compilador en lenguaje ensamblador por defecto, por lo que para poder utilizar el compilador AVR-GCC, hay que descargarlo de su sitio web e instalarlo. Este compilador no tiene ningún costo.

Algunos de los módulos principales que maneja WIN AVR son:

- Utilidades de soporte para *Bootloader*
- Cómputo de control de redundancia cíclica
- Bucles de retraso y espera
- Manejo de la EEPROM
- Manejo del temporizador de vigilancia (Watchdog)
- Matemáticas
- Interrupciones y señales
- Tipos estándar para números enteros

Para trabajar con WIN AVR, se necesita de la inclusión de las librerías que se vayan requiriendo, las cuales pueden ser: *io.h, stdio.h, interrupt.h, math.h*, entre otras, las cuales trae asociado el compilador GCC y las que se pueden encontrar dentro de la subcarpeta AVR del propio software. Para poder ser incluidas en un proyecto, basta con hacer una inclusión de archivos .h tradicionales de los lenguajes C y C++, por ejemplo:

```
#include <avr/io.h>
#include <avr/interrupt.h>
```

La programación de los procesos de adquisición de datos, almacenamiento en memoria y la referencia de tiempo que se realiza en la estación remota, se describen a continuación.

Una vez capturado el código fuente y compilado mediante el AVR STUDIO 4.12, el programa se carga al microcontrolador a través de un programador. El programador utilizado es el AVR Titán EX, figura 4.2, el cual es un "programador en el circuito" (ISP).

Figura 4.2. Programador AVR Titán EX.

El programador AVR Titán EX tiene la ventaja de que aprovecha la tecnología ISP del microcontrolador AVR. Lo anterior se logra sin tener que sacar al microcontrolador del circuito impreso en el que ha sido montado y, a través de algunas terminales del microcontrolador; por ejemplo, en el ATMEGA128, a través de las terminales PDI, PDO y SCK es posible realizar la programación de éste, conectando las terminales MOSI, MISO y SCK del programador a la tarjeta de la estación remota, para descargarle el programa compilado en el AVR STUDIO 4.12. El AVR Titán EX sólo se inserta en un conector de la tarjeta de circuito impreso en la que fue montado el AVR cada vez que haya que hacerle modificaciones al programa y ser cargado de nuevo en el microcontrolador, como se muestra en las figuras 4.3 y 4.4.

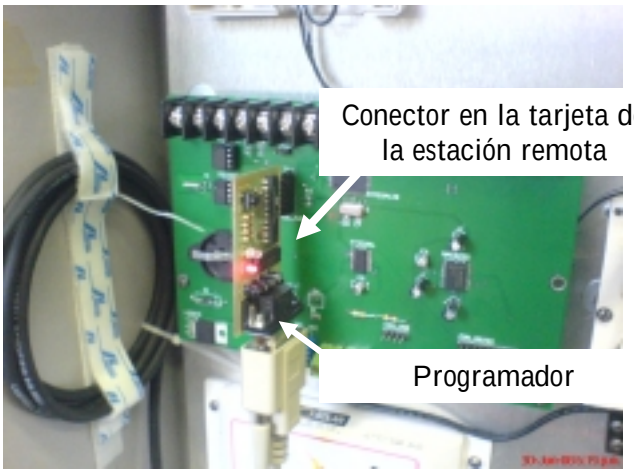


Figura 4.3. Conexión del programador a la estación remota.

La conexión del programador con la PC se hace a través del puerto RS232; en este modelo de programador ya también existe la versión USB. En caso de que la PC a la que se desee conectar el programador no cuente con puerto RS232, como ya es la generalidad de los modelos más recientes, es posible utilizar un convertidor USB a RS232, como el desarrollado y tratado en el capítulo anterior. La figura 4.4, ilustra la conexión del programador con una PC que no tiene puertos RS232 y necesita un convertidor USB a RS232.

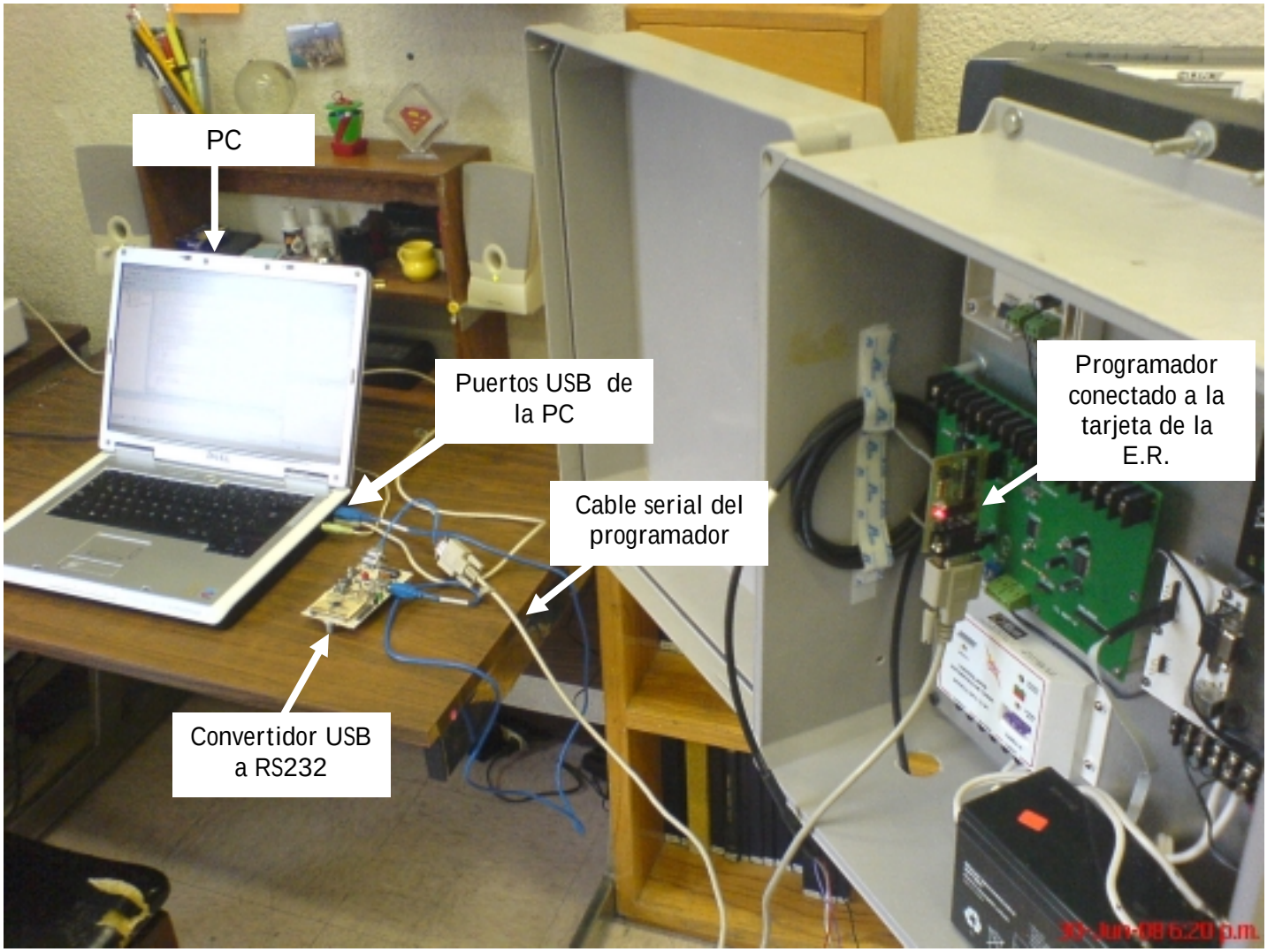


Figura 4.4. Conexión de la PC con el programador AVR Titán EX.

La programación de los procesos de adquisición de datos, almacenamiento en memoria y la referencia de tiempo (reloj en tiempo real) se describen a continuación.

Adquisición de datos

La señal proveniente del sensor de temperatura es un voltaje cuyo valor es proporcional a ésta. Dicho valor debe ser ingresado al microcontrolador central por medio del subsistema de conversión analógica a digital. En la figura 4.5 se muestra la conexión física correspondiente al conjunto sensor-microcontrolador.

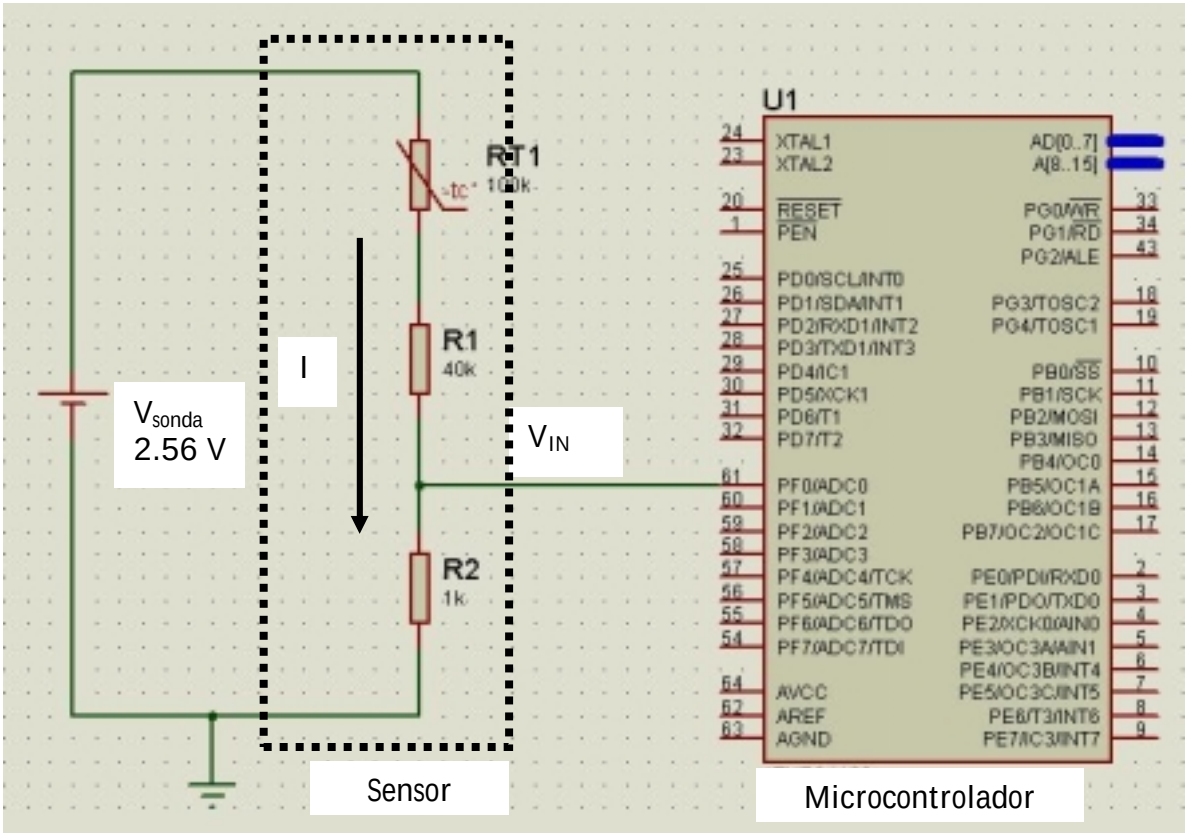


Figura 4.5. Conexión física entre la sonda y el microcontrolador.

Procesamiento de la señal

En la figura 4.5 el sensor está representado por el arreglo de resistencias RT, R1 y R2, las cuales forman una divisora de voltaje. La señal resultante del divisor se conecta a la terminal 61 (ADC0) del ATMEGA128. Dicha terminal corresponde al canal 1, en configuración *single-ended*, del convertidor analógico a digital del microcontrolador, con una referencia de voltaje interna de 2.56 V, establecida mediante software. Para poder hacer el acoplamiento de la sonda con el ADC se debe considerar lo siguiente:

De la figura 4.5, para el bloque correspondiente al sensor y utilizando las leyes de Kirchhoff, además de considerar que el valor de la impedancia de entrada del ADC, reportado por el fabricante, es muy grande, del orden de 100 MΩ , aplicando el concepto del divisor de voltaje y la ecuación 4.1, en el circuito formado por las resistencias que componen el bloque del sensor, se tiene que

$$I = \frac{V_{REF}}{(R_T + R_1) + R_2} \qquad \text{Ecuación 4.1}$$

Donde V_{REF} se refiere al voltaje de polarización de la sonda.

$$V_{IN} = R_2\left(\frac{V_{REF}}{R_T + R_1 + R_2}\right) \qquad \text{Ecuación 4.2}$$

Reacomodando un poco los términos de la ecuación 4.2, la podemos plantear finalmente como:

$$V_{IN} = \left(\frac{R_2}{R_T + R_1 + R_2}\right)V_{REF} \qquad \text{Ecuación 4.3}$$

La ecuación 4.3 representa el valor de la señal analógica del voltaje de entrada al ADC procedente de la sonda sensora de temperatura.

A continuación, se realizará un breve análisis que permitirá obtener las ecuaciones de recurrencia para el cálculo de la temperatura, como la ecuación característica del termistor, comúnmente conocida como ecuación de Steinhart-Hart, y parámetros asociados a ella.

Ahora, teniendo en cuenta la información para el convertidor analógico a digital del microcontrolador, se tiene la siguiente ecuación, proporcionada en la hoja de datos del fabricante, se tiene que el voltaje convertido está en función del voltaje analógico de entrada, de la referencia de voltaje utilizada por el ADC y del número máximo de valores discretos, que con 10 bits, éste puede alcanzar:

$$V_{adc} = \frac{V_{IN} x 1024}{V_{REF}} \qquad \text{Ecuación 4.4}$$

Si de la ecuación 4.4, se despeja V_{IN}, se tiene:

$$V_{IN} = \left(\frac{V_{adc}}{1024}\right) V_{REF} \qquad \text{Ecuación 4.5}$$

Como se utiliza el ADC del ATMEGA128, el cual tiene una resolución de 10 bits, los valores entre 0.0 y V_{REF} son cuantificados como valores desde 0 hasta 1024.

Si ahora igualamos las ecuaciones 4.3 y 4.5, se tiene lo siguiente:

$$\left(\frac{R_2}{R_T + R_1 + R_2}\right) V_{REF} = \left(\frac{V_{adc}}{1024}\right) V_{REF} \qquad \text{Ecuación 4.6}$$

Ya que prácticamente se está utilizando, muy aproximadamente, el mismo valor de voltaje de referencia, tanto en la sonda como en el ADC, podemos eliminar, en la ecuación 4.6, el término V_{REF} en ambos miembros, generando una ecuación más sencilla, independiente de V_{REF}, teniendo por tanto:

$$\left(\frac{V_{adc}}{1024}\right)=\left(\frac{R_2}{R_T+R_1+R_2}\right) \Rightarrow \qquad \therefore \quad R_T=\frac{(R_2)(1024)}{V_{adc}}-(R_1+R_2) \qquad \text{Ecuación 4.7}$$

Una vez obtenido el valor de R_T, es necesario conocer el valor de la temperatura. Este cálculo se realiza auxiliándonos de la ecuación característica del termistor. La ecuación, conocida como ecuación de Steinhart-Hart, ecuación 4.8, aproxima el comportamiento de la curva característica del termistor en un polinomio que está en función de la resistencia del termistor y de sus constantes de fabricación, las cuales pueden ser proporcionadas por el fabricante u obtenidas de forma experimental, de forma relativamente sencilla.

$$\frac{1}{T}=A+B\ln(R_T)+C\left(\ln(R_T)\right)^3 \qquad \text{Ecuación 4.8}$$

Donde las constantes A, B y C pueden ser calculadas de forma experimental, utilizando información del fabricante del termistor y resolviendo un sistema de ecuaciones simultáneas de tres incógnitas, T es la temperatura en K y R_T es la resistencia del termistor. En nuestro caso, utilizando °C como unidad de temperatura de uso común, se puede replantear la ecuación en términos de °C, como se muestra en la ecuación 4.9.

$$T=\frac{1}{A+B\ln(R_T)+C(\ln(R_T))^3)}-273.15 \qquad \text{Ecuación 4.9}$$

Para el caso de la sonda utilizada, una sonda modelo 108-L y fabricada por Campbell Scientific, proveedor y desarrollador de tecnologías para estaciones meteorológicas. La sonda tiene como características principales con un intervalo dinámico de temperatura de -5 a 95°C, una constante de tiempo de 200±10 segundos; y en el interior se encuentra un termistor *betatherm* de 100kΩ a 25°C, así como dos resistencias, una de compensación de 40 kΩ y otra, para acondicionamiento de 1 kΩ, las constantes A, B y C, proporcionadas por el fabricante, tienen los siguientes valores:

$A = 8.271111 \times 10^{-4}$; $B = 2.088020 \times 10^{-8}$; $C = 8.059200 \times 10^{-8}$

Con los valores disponibles de las constantes del termistor, las expresiones para el cálculo de la resistencia del termistor, en función del valor digitalizado del voltaje de entrada al ADC procedente del sensor, es posible plantear las siguientes ecuaciones iterativas:

$$V_{adc} = \frac{V_{IN} x 1024}{2.56} \qquad \text{Ecuación 4.10}$$

$$R_T = \frac{1024000}{V_{adc}} - 41000 \qquad \text{Ecuación 4.11}$$

$$T = (1/(8.271111 \times 10^{-4} + 2.088020 \times 10^{-4} \ln (R_T) + 8.059200 \times 10^{-8} (\ln (R_T))^3) - 273.15 \qquad \text{Ecuación 4.12}$$

Las cuales serán utilizadas para el procesamiento de los datos obtenidos por el ADC del microcontrolador. En la sección de apéndices, se encuentra un listado parcial de la rutina de procesamiento para el microcontrolador ATMEGA128.

Almacenamiento en memoria y referencia de tiempo

Dos dispositivos más, cuya presencia es importante dentro de la estructura de la estación remota son el reloj en tiempo real DS1307 y la memoria 24LC512. A nivel hardware, su inclusión no revistió mayor dificultad, sin embargo, en lo que corresponde a la comunicación con el microcontrolador central, soportado por software, vía el protocolo de comunicaciones seriales TWI, la programación no fue trivial, sobre todo, por la que toca a la dinámica de programación y reglas del propio protocolo que hay que respetar.

El *bus* TWI fue diseñado para la implementación de comunicación de datos a corta distancia. El protocolo TWI a través de tan sólo dos cables permite, a varios circuitos integrados y módulos de desarrollo (OEM), interactuar entre sí a velocidades relativamente lentas. Emplea una comunicación serie, utilizando un conductor para el manejo de la base de tiempo (pulsos de reloj) y otro para el intercambio de datos.

El bus TWI se basa en tres líneas:

- ✓ Datos de sistema (SDA). Es la línea por la cual viajan los datos entre dispositivos
- ✓ Reloj del sistema (SCL). Es la línea de datos por la cual transitan los pulsos de reloj que sincronizan al sistema

Las líneas SDA y SCL son del tipo *drain* abierto, similares a las de colector abierto pero asociadas a un transistor de efecto de campo (FET). Se deben llevar a un estado lógico alto, a través de la conexión de resistencias de *pull-up*, para construir una estructura de *bus,* de tal forma que se permita conectar en forma paralela múltiples salidas y entradas. Las dos líneas de comunicación disponen de niveles lógicos altos cuando están inactivas. Inicialmente, el número de dispositivos que se puede conectar al *bus* es ilimitado; sin embargo, las líneas cuentan, típicamente, con una capacitancia máxima de 400 pF, lo que las limita a un máximo de 128 dispositivos. El *bus* puede operar dentro de un margen de velocidades de transferencia de datos que abarca desde los 0 hasta los 400 kHz, sin embargo, algunos dispositivos sólo alcanzan los 100 kHz, como por ejemplo el circuito DS1307.

En el protocolo TWI se definen dos elementos: maestro y esclavo.

- ✓ Maestro (MASTER). Es el dispositivo que determina la temporización y la dirección del tráfico de datos sobre el *bus*. Es el único elemento dentro del protocolo que aplica los pulsos de reloj sobre la línea SCL. Cuando se conectan varios dispositivos maestros a un mismo *bus*, la configuración obtenida se llama multi-maestro.
- ✓ Esclavo (SLAVE). Es cualquier dispositivo conectado al *bus* controlado por el maestro. El esclavo es incapaz de generar pulsos de reloj. Reciben señales de comando y de reloj proveniente del dispositivo maestro. Se pueden conectar hasta 127 dispositivos, que son los dispositivos seleccionados por el maestro mediante 7 bits (dirección del esclavo). Para el reloj en tiempo real DS1307, la dirección de esclavo es 1101000_2, mientras que en el caso de la memoria 24LC512, es 1010_2 más la posición lógica que a las que se fijen las terminales A0, A1 y A2.

Ahora bien, los estados que puede presentar el *bus* TWI son: *bus* desocupado, comienzo, paro, dato inválido y dato válido.

- ✓ Bus desocupado (BUS FREE). El estado de *bus* desocupado se da cuando ambas líneas, SDA y SCL, están inactivas, presentando un estado lógico alto. Únicamente en este estado es cuando un dispositivo maestro puede comenzar a hacer uso del *bus*.
- ✓ Comienzo (START). Sucede cuando un dispositivo maestro toma el control del *bus*, generando esta condición. La línea de datos (SDA) toma un estado bajo mientras que la línea de reloj (SCL) permanece en alto.
- ✓ Paro (STOP). Un dispositivo maestro puede generar esta condición dejando libre el bus. La línea de datos toma un estado lógico alto mientras que la de reloj permanece también en este estado.
- ✓ Dato válido (VALID DATA). Esta condición sucede cuando un dato presente en la línea SDA permanece estable, mientras que la línea SCL está a un nivel lógico alto.
- ✓ Dato no válido (INVALID DATA). La condición de dato inválido sucede cuando un dato presente en la línea SDA es estable mientras que la línea SCL está a un nivel lógico bajo.

En cuanto a la lo referente al tráfico de datos a través del *bus*, se definen los siguientes conceptos del *bus* TWI:

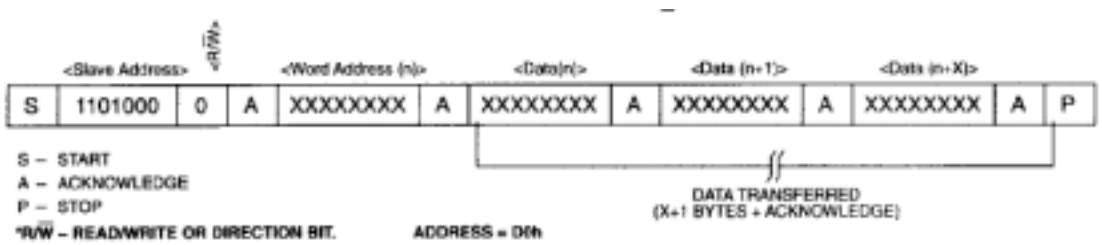
- ✓ Formato de datos (DATA FORMAT). La transmisión de datos a través de este *bus* consta de 8 bits de datos (1 byte). A cada byte le sigue un noveno pulso de reloj durante el cual, el dispositivo receptor del byte debe generar un pulso de reconocimiento, conocido como ACK. Esto se logra situando la línea de datos a un nivel lógico bajo mientras transcurre el noveno pulso de reloj.

- ✓ Dirección (ADDRESS). Cada dispositivo con protocolo integrado TWI o I^2C está diseñado para funcionar sobre las líneas SDA y SCL del protocolo, y dispone de su propia dirección de acceso, que viene preestablecida por el fabricante. Existen dispositivos que, mediante el uso de conexiones de ciertas terminales a niveles lógicos altos y bajos, permiten establecer externamente parte de la dirección de acceso, lo cual permite que una serie del mismo tipo de dispositivos se puedan

conectar en un mismo *bus* sin problemas de identificación. La dirección 00 es la denominada "de llamada general", por la cual responden todos los dispositivos que han sido conectados al *bus*.

- ✓ Bit de lectura y escritura (R/W). Cada dispositivo dispone de una dirección de 7 bits. El octavo bit, el menos significativo (LSB), enviado durante la operación de direccionamiento, corresponde al bit que indica el tipo de operación a realizar. Si este bit es alto, el dispositivo maestro lee la información proveniente de un dispositivo esclavo. En cambio, si este bit fuese bajo, el dispositivo maestro escribe información en un dispositivo esclavo.

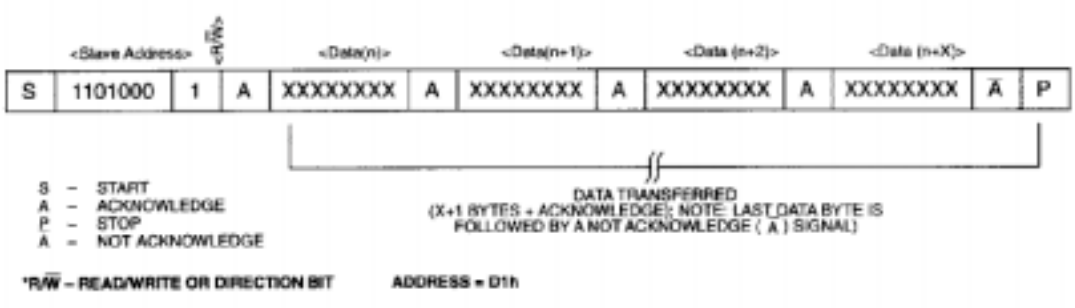
La figuras 4.6 y 4.7, muestran los procesos de escritura y de lectura para el reloj de tiempo real DS1307. Se observa que, en la estructura de datos que envía el dispositivo maestro al dispositivo esclavo, la primera trama de datos, después de ser enviado la señal de inicio (START) del protocolo TWI, se envía la dirección del dispositivo en 7 bits, que en este caso es 1101000_2, seguido de 1 bit, que define la dirección de comunicación de datos, que en este caso es de escritura, que según el protocolo TWI, es "0". Una vez enviado este paquete de 8 bits, el dispositivo esclavo envía una señal de confirmación por medio de una señal ACK. El siguiente paquete de datos contiene la dirección del registro de control, marcados en el mapa de direcciones de registros de control del circuito DS1307, que puede corresponder a la hora, los minutos, los segundos, etcétera. Enseguida, se espera confirmación (ACK) de parte del dispositivo esclavo hacia el maestro, para continuar con el envío del contenido del registro de control. Se continúa la definición del registro y la carga de su contenido, tantas veces como registros de control contenga el dispositivo esclavo. La operación termina cuando el dispositivo maestro envía al esclavo una señal de paro (STOP).



Si el bit R/W = 0, es un proceso de escritura.

Figura 4.6. Escritura del DS1307.

En la figura 4.7 se muestra el proceso de lectura del reloj de tiempo real DS1307. El algoritmo que se sigue para la lectura es muy similar al del proceso de escritura, con pequeños cambios. Uno de los cambios más notables es que, el bit que corresponde a la lectura ahora es "1". Le siguen señales de confirmación SCK enviadas por el dispositivo esclavo al maestro. En algunos dispositivos, como se verá más adelante, se requiere el envío de una condición REPEATED START, antes de enviar la dirección del registro que se necesita leer.



Si el bit R/W = 1, es un proceso de lectura.

Figura 4.7. Lectura del DS1307.

En la figura 4.8 se muestra el patrón de tramas de datos para el proceso de escritura a la memoria 24LC512. Como se puede observar, el primer dato que el dispositivo maestro envía es el que corresponde a una señal de START, seguido por un byte de control, que define la dirección del esclavo. Dicho byte está compuesto por 4 bits genéricos 1010_2, seguido por 3 bits más, *A0, A1* y *A2*, cuyas combinaciones, mediante su conexión a estados lógicos altos o bajos, permiten crear un banco de hasta 8 dispositivos de memoria, cada uno de ellos con una dirección distinta entre ellos. El último bit que conforma a este primer byte, es el bit de dirección de comunicación de datos, que para el caso de escritura es "0". A continuación, el dispositivo esclavo envía una señal de confirmación, ACK, al dispositivo maestro. La siguiente trama de datos que envía el maestro al esclavo contiene el byte alto de la dirección del registro al que se quiere escribir. A diferencia del reloj en tiempo real DS1307, en la memoria EEPROM 24LC512 la dirección de los registros de datos se especifica en 2 bytes. Una vez enviado el byte alto y bajo del registro al que se quiere escribir, el dispositivo esclavo envía la señal de confirmación correspondiente, ACK, hacia el dispositivo maestro. Concluida la transmisión de la dirección de los registros para la escritura se envían los datos que contendrán aquellas, acompañadas a su finalización,

por sus respectivas señales de confirmación, ACK. Se termina el proceso con el envío del maestro al esclavo de una señal de STOP.

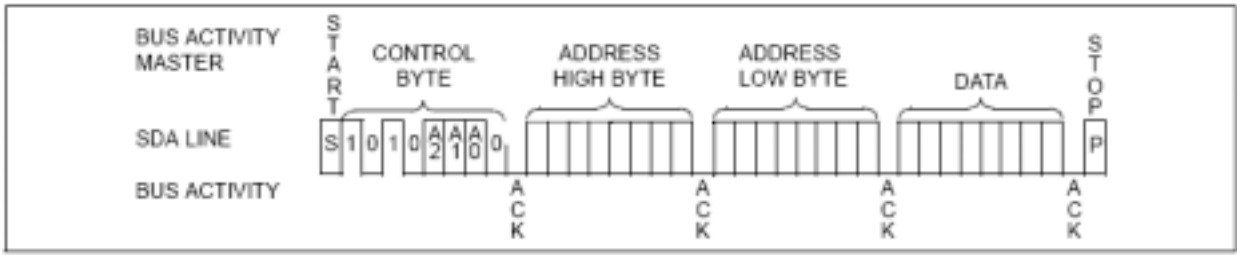


Figura 4.8. Escritura de un byte a la memoria 24LC512.

En la figura 4.9 se muestra el proceso de lectura en la memoria 24LC512. Dicho proceso comienza, al igual que en la escritura, con el envío de una señal de START, seguido por el byte de control, cuyo último bit será "1", correspondiente a la dirección de comunicación de datos de lectura. A continuación se recibe una señal de confirmación por parte del dispositivo esclavo, seguido por las partes alta y baja del registro inicial del dispositivo. En el caso de esta memoria EEPROM, para iniciar de lleno con el proceso de lectura de los registros correspondientes, antes se envía una condición de REPEATED START. Para finalizar el proceso, el dispositivo maestro envía una condición de STOP.

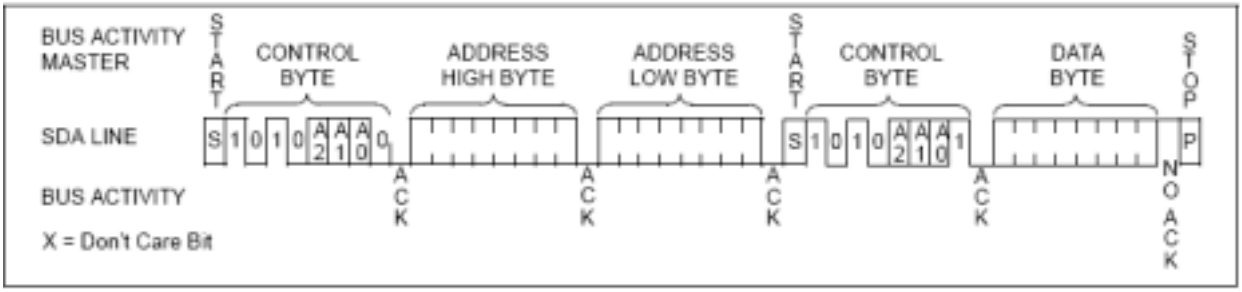


Figura 4.9. Lectura aleatoria en la memoria 24LC512.

Los listados parciales de las rutinas para el reloj en tiempo real DS1307 y para la memoria 24LC512 se pueden consultar en el apéndice A.

4.2. SOFTWARE DE LA ESTACIÓN BASE

La estación base está integrada por una computadora personal y un radio módem para comunicaciones de largo alcance, opcionalmente también se pueden integrar otros medios de comunicación, también de largo alcance, como módem telefónico, módem GSM, módem satelital, etcétera, según con los que también se cuente en la estación remota. Por defecto, se dispone además de los puertos de comunicación estándar como el USB o RS232, principalmente para comunicaciones de corto alcance o en sitio.

El software de la estación remota fue desarrollado en Visual Basic 6.0, el cual además de ser un lenguaje de alto nivel, permite, de forma relativamente sencilla, la creación de interfaces gráficas para el usuario. Así mismo, es un programa que puede vincularse con bases de datos, potencializando la utilidad de las aplicaciones allí creadas.

El desarrollo de la interfaz con el que cuenta la estación es básico, permite elegir de entre los puertos disponibles de la computadora residente, el puerto a través del cual se pueda comunicar con la estación remota, permite también la apertura manual y el cierre de los puertos para la comunicación con la estación remota, de forma directa a través de cable o bien, a distancia, a través de cualquier medio de comunicación de largo alcance, por ejemplo el radio módem. Uno de los componentes principales en la programación de la interfaz fue el uso del control MSCOMM de Visual Basic. El control MSCOMM permite tomar el control de los puertos de salida serie de la computadora, pudiendo a través de ellos comunicarse con la estación remota, vía los medios de comunicación alámbricos o inalámbricos correspondientes. En la figura 4.10 se muestra una figura del entorno de Visual Basic donde se creó la interfaz.

Cabe comentar que el software de diseño que nos permitió elaborar la interfaz para la estación remota, con su filosofía de programación orientada a objetos, es la base de algunos ambientes de diseño especializadas, como es el caso de Campbell Scientific, cuyo paquete llamado RTDM, en versión demo, permite la creación de interfaces gráficas para la adquisición de datos de sus estaciones meteorológicas comerciales. En la figura 4.11, se muestra el ambiente del software de Campbell Scientific para el desarrollo de interfaces gráficas.

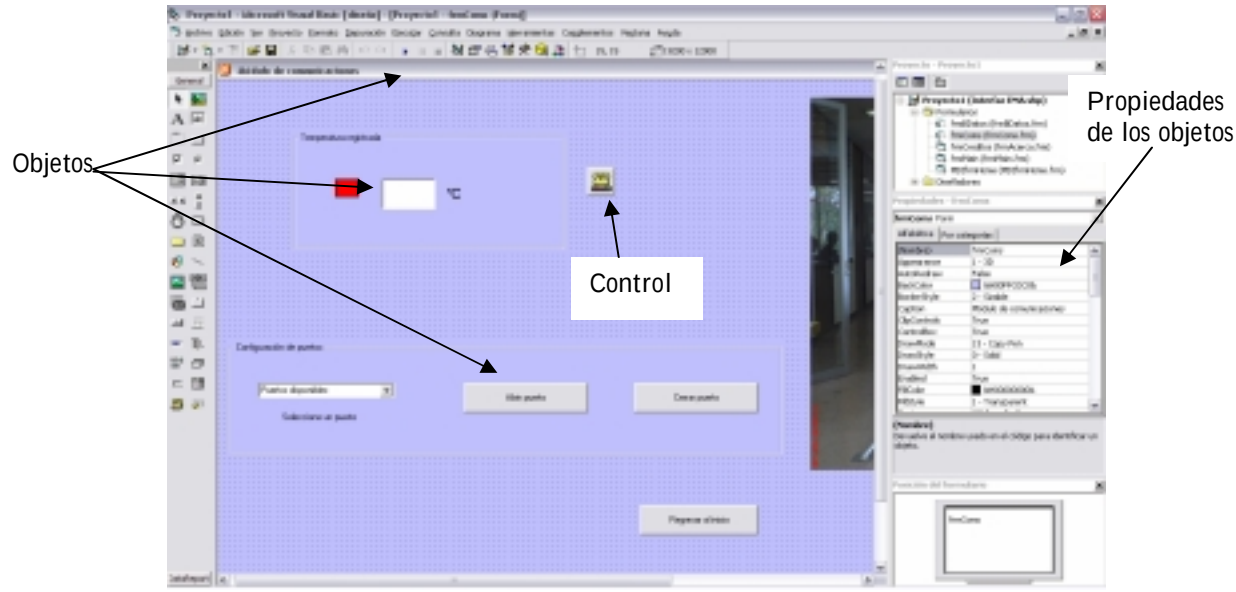


Figura 4.10. Ambiente de desarrollo Visual Basic.

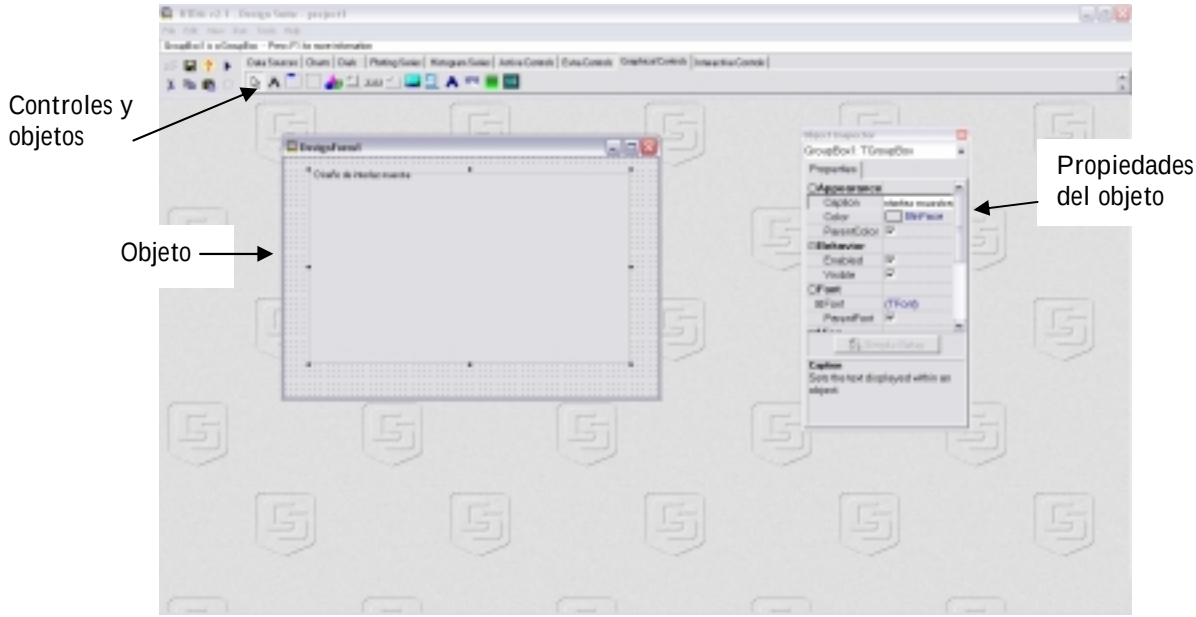


Figura 4.11. Ambiente de desarrollo RTDM (Campbell Sci.)

El software desarrollado está estructurado en ventanas, las cuales permiten una conceptualización modular del programa. Las figuras 4.12 a 4.14, muestran la estructura del software desarrollado.

Figura 4.12. Pantalla de bienvenida.

Figura 4.13. Pantalla principal.

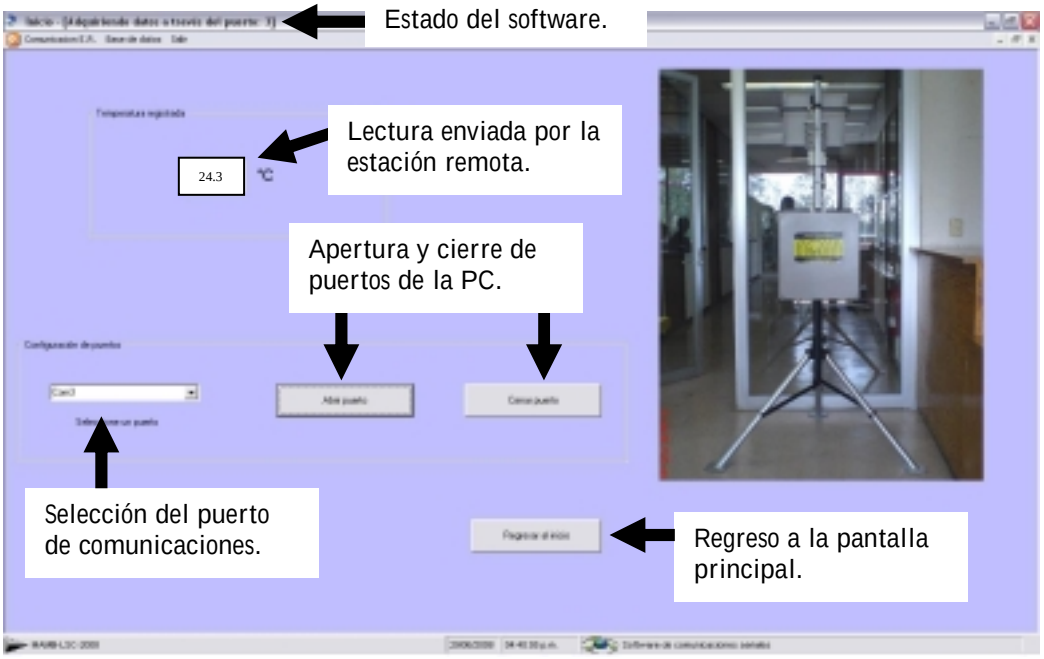


Figura 4.14. Módulo para comunicaciones seriales.

Un listado parcial del código fuente del software de la estación base se presenta en el apéndice B.

Como se ha visto hasta este momento, por un lado el hardware y ahora el software, son dos de los puntos más importantes que conforman a la estación remota. En el siguiente capítulo se analizará la integración de ambos elementos y se podrá evaluar el comportamiento del sistema.

5.1. DISEÑO DE LOS CIRCUITOS IMPRESOS

Uno de los medios de integración de todos los elementos de hardware, presentados en el capítulo 3, es el circuito impreso (PCB) de la estación remota, mismo que contendrá a las unidades de alimentación de energía, conversión analógica a digital, comunicación de datos, memoria y reloj de tiempo real, que permitirá la interacción entre estas unidades y su vinculación con el software desarrollado para la estación base. El objetivo entonces, de la tarjeta de circuito impreso, es la interacción funcional de todos los elementos, tanto de hardware como de software que integran a la estación remota.

La primera versión de la estación remota se ensambló en una tarjeta perforada de cobre (figura 5.1), práctica en su ensamblado, y con capacidad de expansión. En esta primera versión, el microcontrolador utilizado, de la gama media de la familia AVR (ATMEGA32), tenía el principal objetivo de experimentar con algunas rutinas básicas de comunicación y, servir como una tarjeta de pruebas de propósito general.

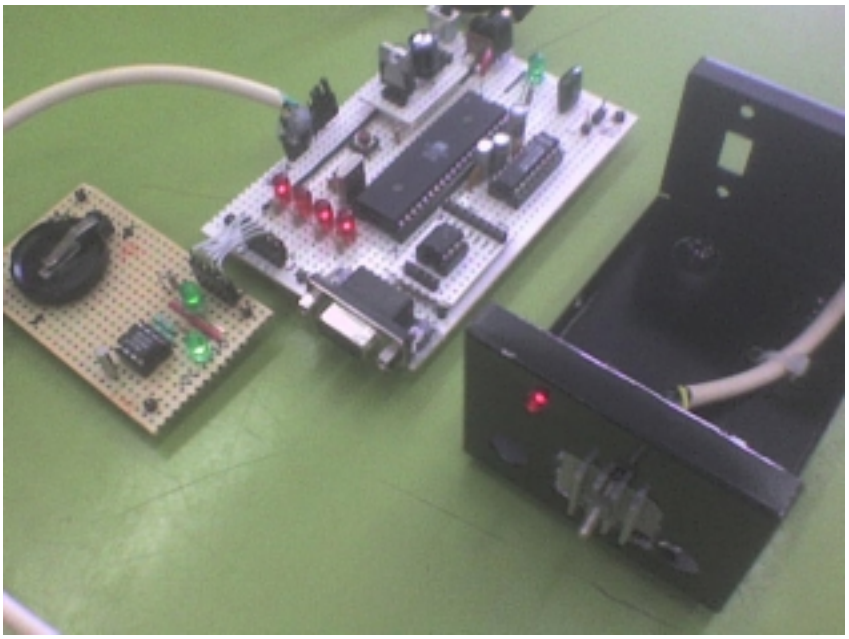


Figura 5.1. Primera versión de la estación remota.

Una vez definida la arquitectura final de la estación remota, fue necesario, el desarrollo de una tarjeta de circuito impreso. Dicha tarjeta albergaría a todos los componentes de la estación, algunos de los cuales, debido a su nula disponibilidad en el mercado nacional como elementos de simple inserción, tuvieron que seleccionarse de tipo montaje superficial (SMD), lo cual implicaba no sólo una mayor complejidad en el armado, sino una mayor complejidad en el diseño y fabricación de la propia tarjeta de circuito impreso.

Una de las herramientas utilizadas para el diseño de los circuitos impresos fue el uso del software PROTEL DXP, versión demo, el cual permite, de una forma relativamente sencilla, el diseño y construcción de tarjetas de circuito impreso a partir de la captura de un circuito esquemático. El ambiente de este software, bajo la plataforma Windows, ofrece al usuario, de una forma amigable, potentes herramientas para la creación de circuitos impresos. El ambiente de desarrollo del software se muestra en la figura 5.2.

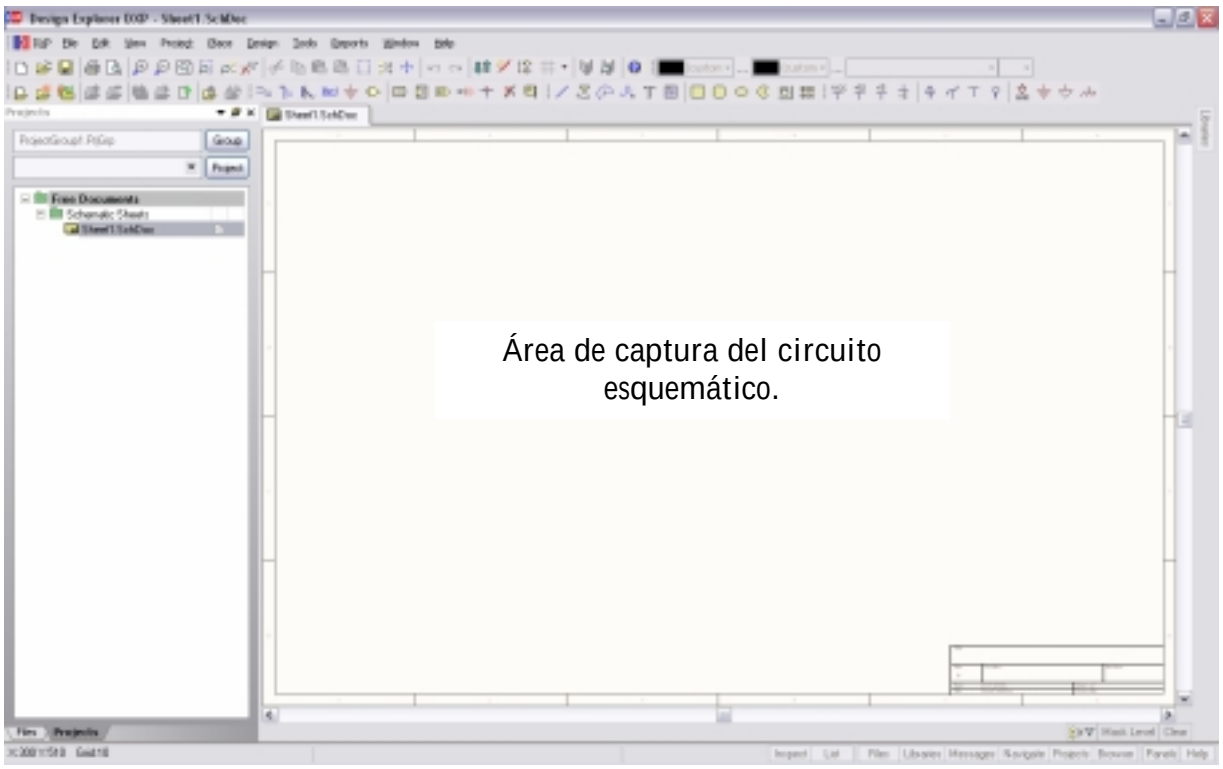


Figura 5.2. Ambiente de desarrollo del software para el diseño de PCBs.

De entre las herramientas y utilidades que el software CAD ofrece al usuario, está la disposición de cientos de bibliotecas conteniendo el símbolo gráfico de diferentes componentes y su respectiva imagen en el circuito impreso, en este mismo punto, permite la edición y creación de nuevas bibliotecas de componentes y la adición de otras, las cuales se pueden descargar, algunas de ellas sin costo, de su sitio en Internet. En lo que respecta a la generación del circuito impreso, el software ofrece utilidades de ruteo automático, por medio de algoritmos creados a partir redes neuronales y de ruteo interactivo, en el cual, el usuario manualmente va creando las mejores rutas entre los dispositivos del circuito. En cuanto a opciones de visualización, el software permite la vista en 3D del circuito impreso que se ha generado, figura 5.3, dando al usuario una presentación preliminar del circuito antes de ser impreso o enviado para su fabricación. En cuanto al formato de los archivos de salida para su fabricación, el software ofrece dos tipos básicos de formato, Gerber y NCDrill, utilizados por la mayor parte de los fabricantes de PCBs profesionales.

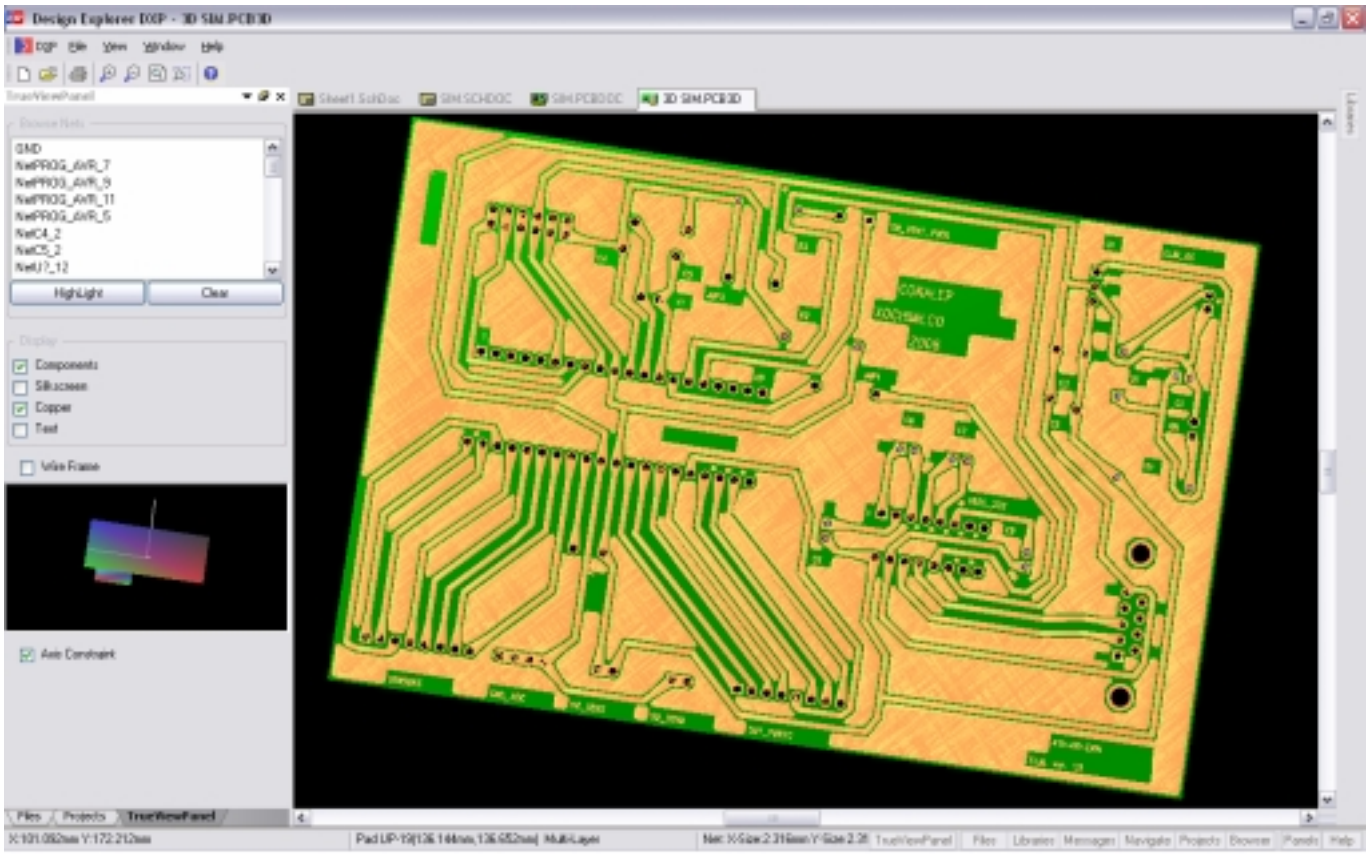


Figura 5.3. Vista 3D de una tarjeta PCB, creada con software especializado.

Una vez finalizada la etapa de proyección de ajustes de la estación remota, se inició el diseño de la PCB mediante el software CAD. El paso inicial consistió en la captura del circuito esquemático, cuya imagen se muestra en la figura 5.4.

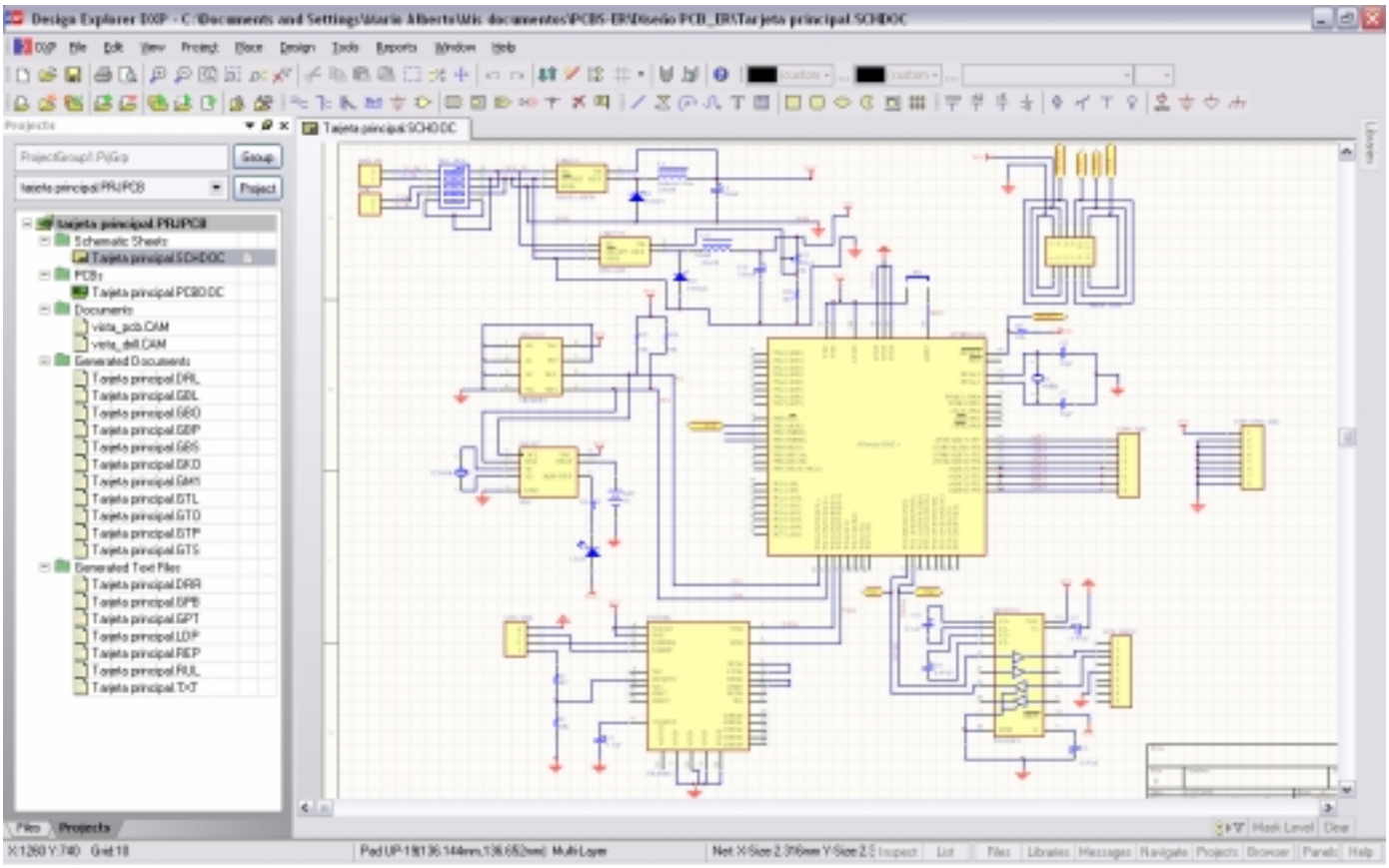


Figura 5.4. Captura del circuito esquemático de la estación remota.

Algunas de las herramientas utilizadas para el desarrollo del circuito esquemático fue el diseño de librerías que no contenía el programa originalmente, tal fue el caso del circuito integrado FT232RL, del cual se tuvo que crear su biblioteca, basándose en la información contenida en la hoja de especificaciones del fabricante, en relación con la funcionalidad de cada uno de las terminales que integran dicho circuito y de las dimensiones físicas del mismo. La biblioteca del componente FT232RL, esquemático e imagen de circuito impreso, se muestra en la figura 5.5.

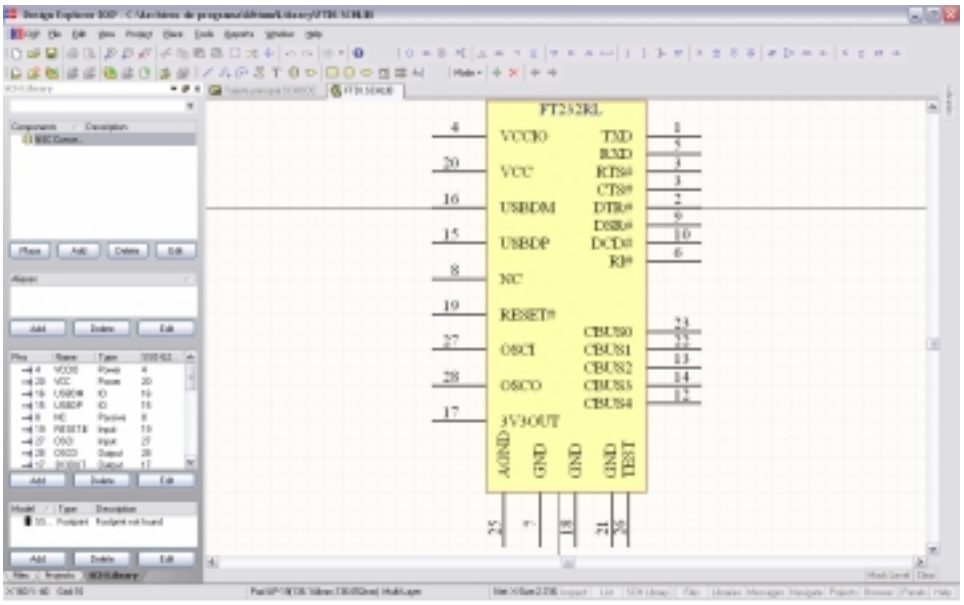


Figura 5.5. Creación de la biblioteca para el componente FT232RL.

Una vez que todas las bibliotecas de elementos estuvieron disponibles, capturado el circuito esquemático y definidas las reglas para el ruteo automático y la disposición física de componentes dentro de la tarjeta, se generó el circuito impreso, que se muestra en la figura 5.6.

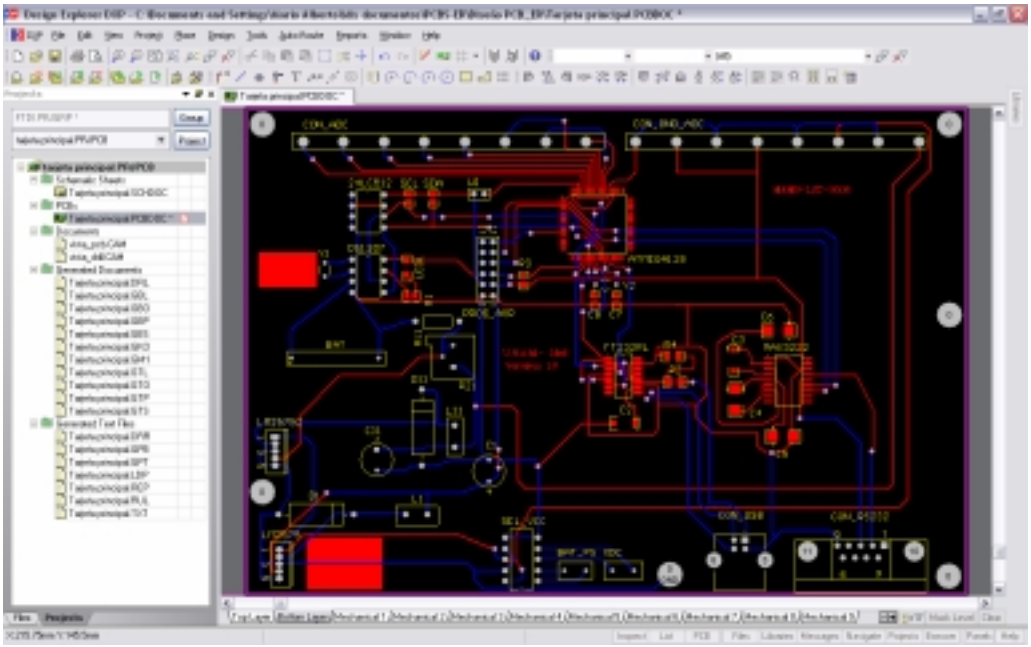


Figura 5.6. Circuito impreso generado por software.

Dada la complejidad de fabricación, debido a que es un circuito de doble cara, se recurrió a un fabricante externo (SWPLUS S.A. de C.V.) al cual se le proporcionó el diseño del circuito impreso en formato Gerber. Gerber es un formato estandarizado utilizado por la mayor parte de los fabricantes de circuitos impresos, el cual tiene la característica de que puede ser abierto por visualizadores que no necesariamente pertenezcan al ambiente de diseño en el que fue creado. Los archivos Gerber, generados se muestran en las figuras 5.7, 5.8 y 5.9 y corresponden a las caras superior e inferior de la PCB y a la disposición de las perforaciones.

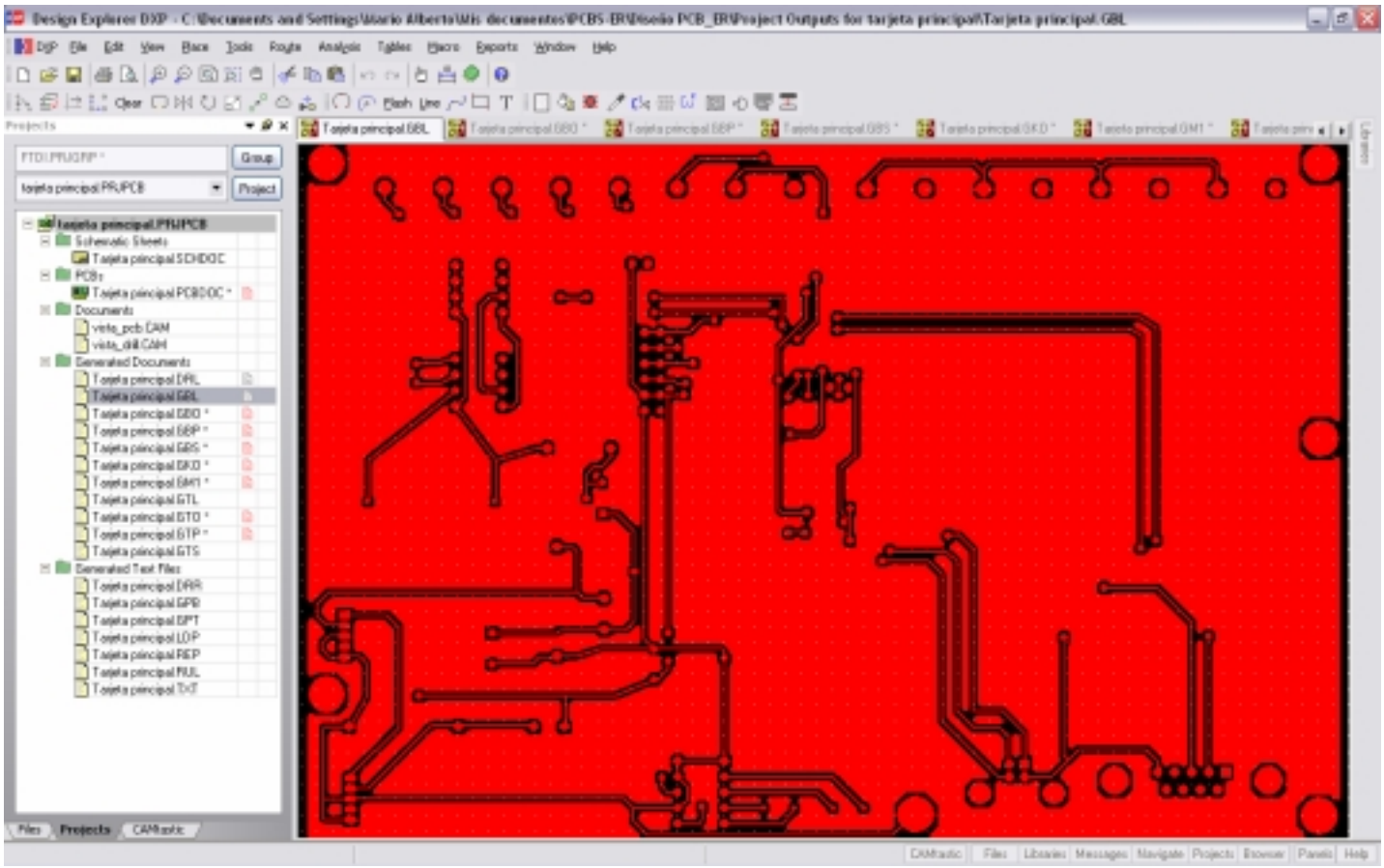


Figura 5.7. Archivo Gerber para la cara inferior de la PCB.

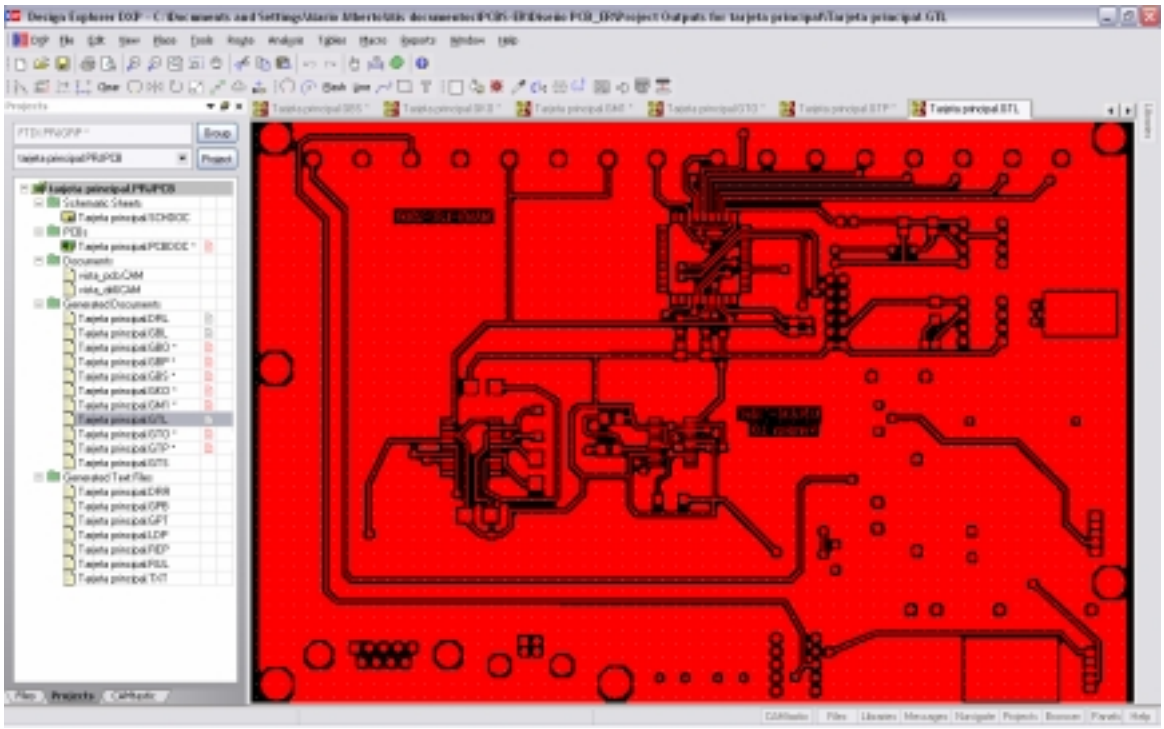


Figura 5.8. Archivo Gerber para la cara superior de la PCB.

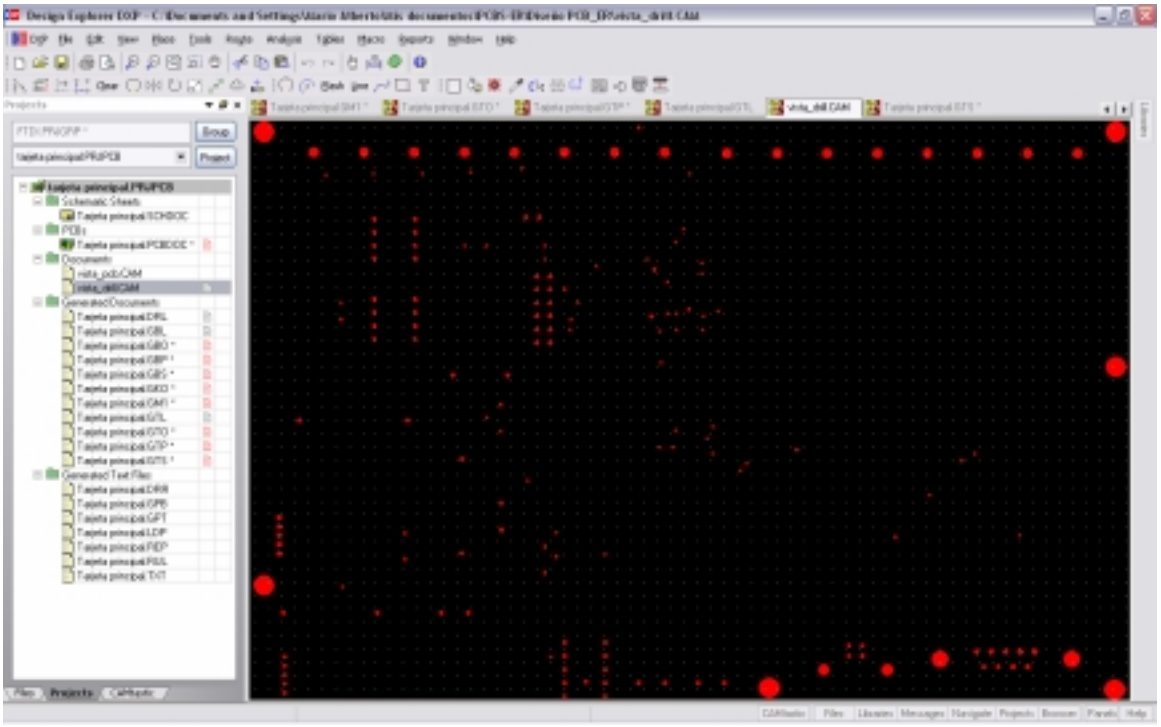


Figura 5.9. Disposición de perforaciones de la PCB.

Los circuitos impresos fabricados se que se muestran en las figuras 5.10, 5.11, 5.12 y 5.13, correspondientes a las dos versiones que al momento se tienen.

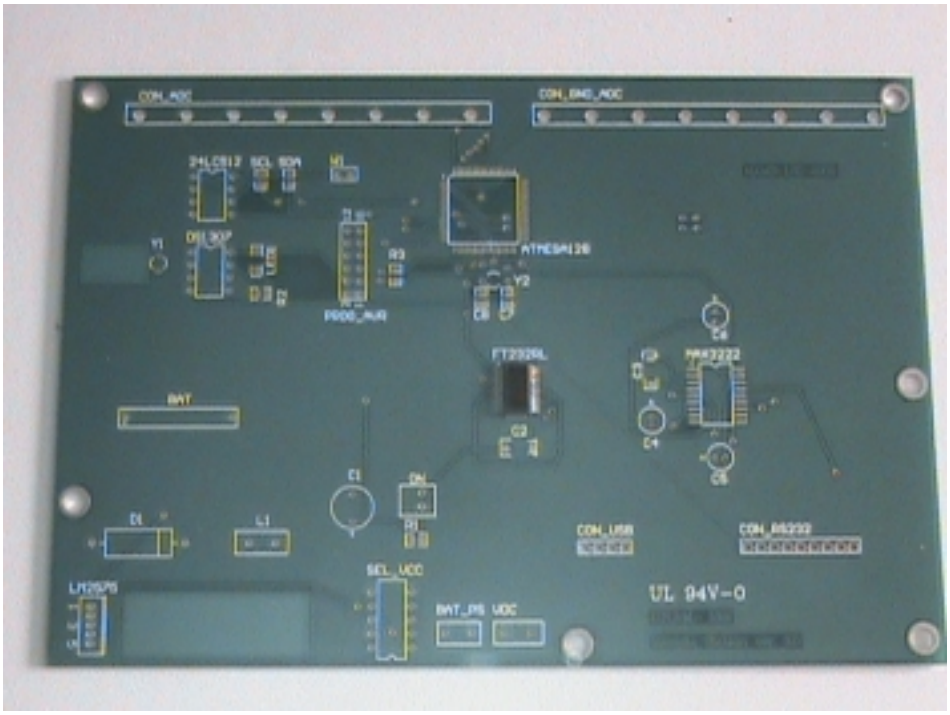


Figura 5.10. Circuito impreso de la estación remota versión 0.0 (vista superior).

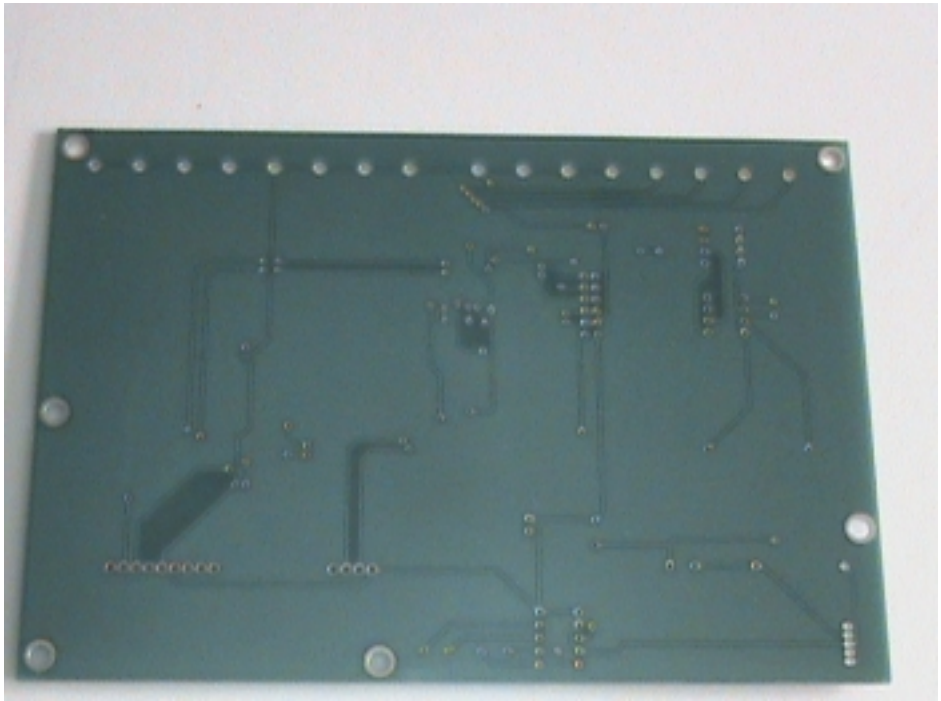


Figura 5.11. Circuito impreso de la estación remota versión 0.0 (vista inferior).

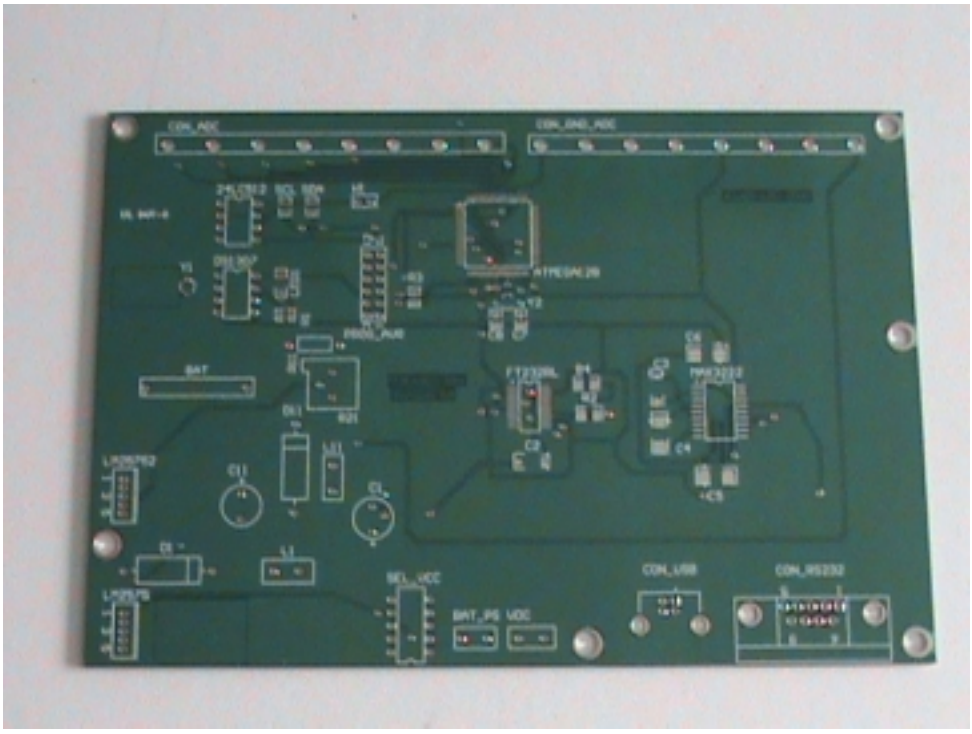


Figura 5.12. Circuito impreso de la estación remota versión 1.0 (vista superior).

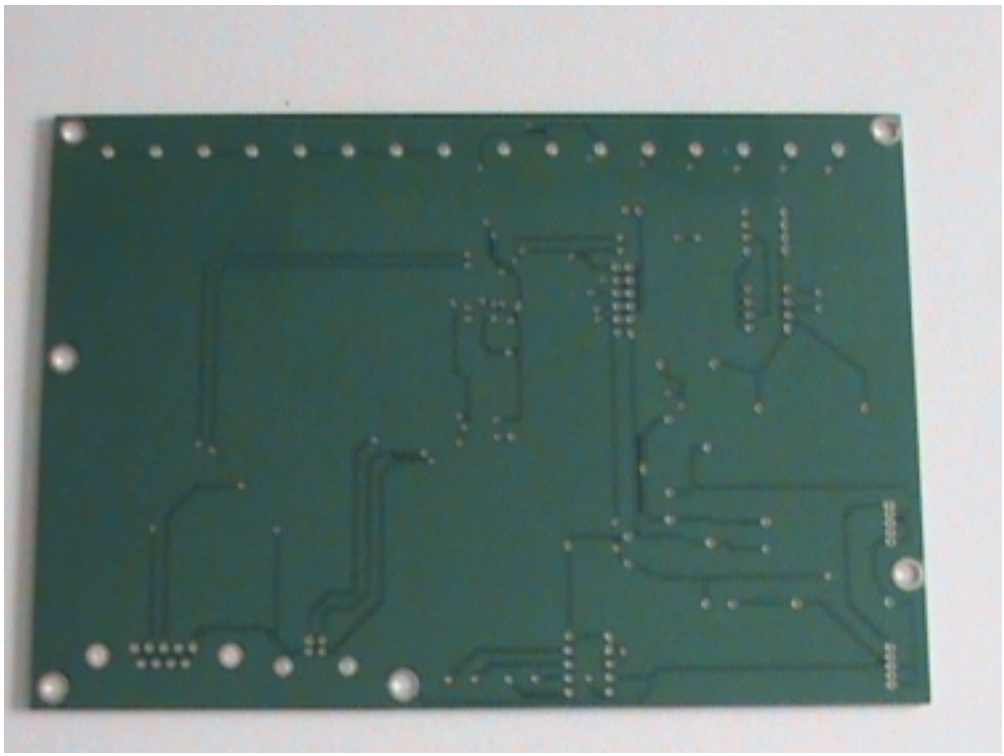


Figura 5.13. Circuito impreso de la estación remota versión 1.0 (vista inferior).

La figura 5.14 muestra a la tarjeta de la versión 0.0 con los componentes montados.

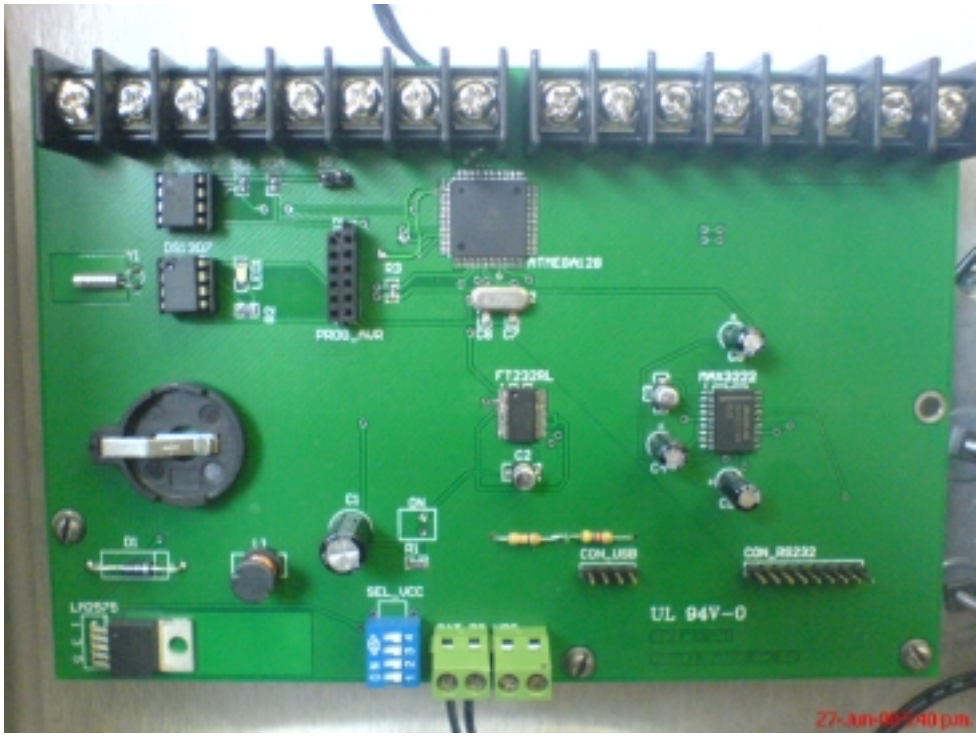


Figura 5.14. PCB versión 0.0 con componentes montados.

En cuanto al proceso de fabricación de los circuitos impresos, realizado por SWPLUS S.A. de C.V., a grandes rasgos, fue el siguiente:

1. Aplicación de cobre sobre la placa de fibra de vidrio.
2. Aplicación de material fotosensible
3. Fotograbado
4. Proceso de revelado de las caras superior (TOP) y posterior (BOTTOM)
5. Perforado
6. Inspección del perforado
7. Platinado de perforaciones
8. Aplicación de material fotosensible, para el *solder mask. Solder mask* es una capa aislante que es aplicada a la PCB que expone solamente a las zonas que deben ser soldadas
9. Proceso de revelado del *solder mask*

10. Grabado del *silk screen.* Silk screen es toda la serigrafía estampada sobre la o las capas de componentes del PCB
11. Prueba eléctrica

Otra tarjeta de circuito impreso fabricada fue la de la fuente de alimentación de 2.56 V, para el sensor de temperatura. Las figuras 5.15 y 5.16, muestran el proceso de desarrollo por software, mientras que la figura 5.17 muestra la tarjeta grabada y la figura 5.18 muestra la imagen de la tarjeta ensamblada.

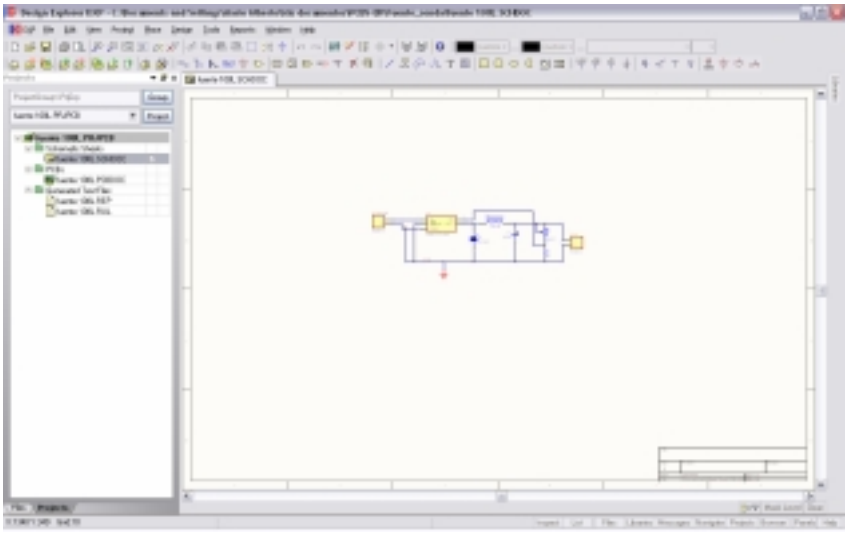


Figura 5.15. Captura del circuito esquemático para la fuente de 2.56V.

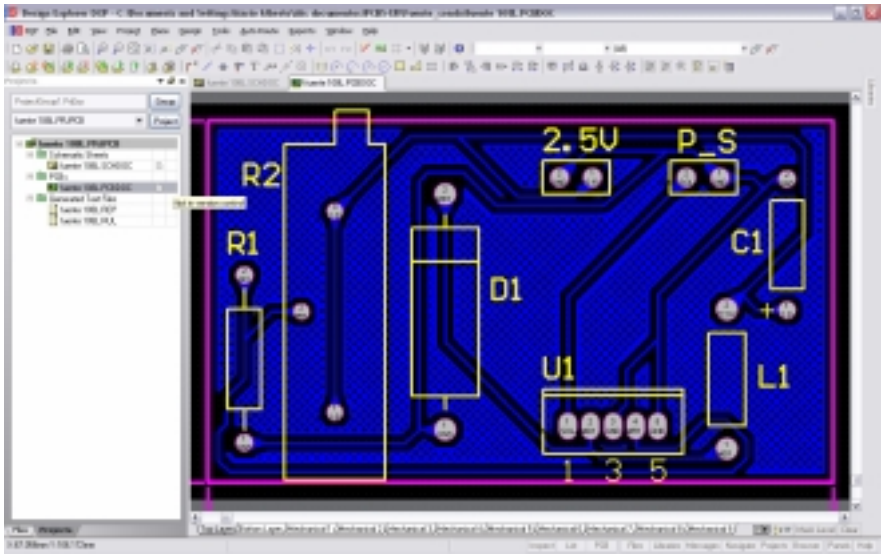


Figura 5.16. Circuito impreso de la fuente de 2.56 V generado.

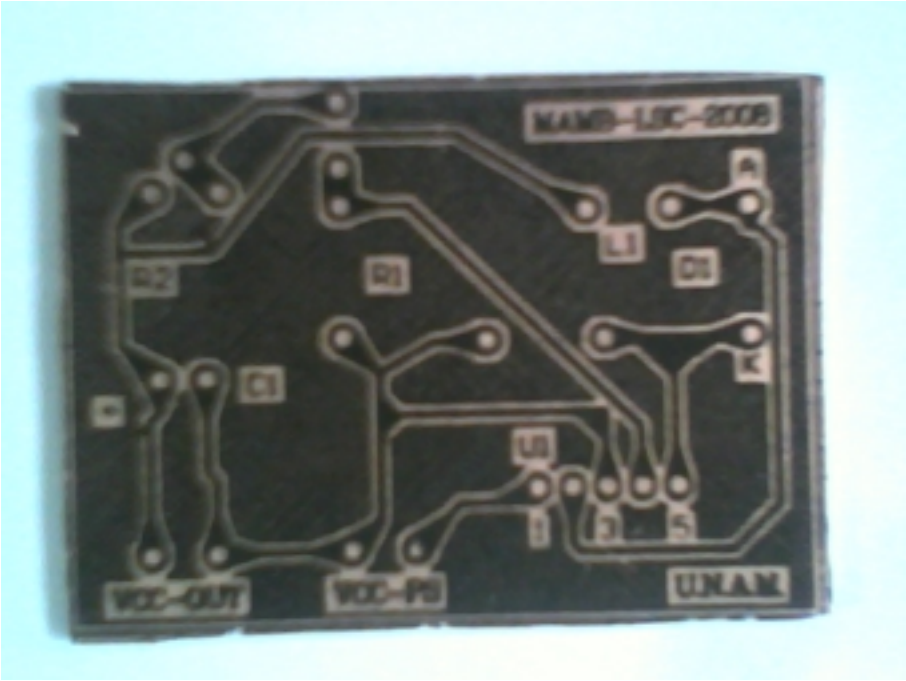


Figura 5.17. Circuito impreso de la fuente de 2.56 V.

Figura 5.18. Circuito impreso ensamblado de la fuente de 2.56 V.

5.2. PRUEBAS, AJUSTES Y EVALUACIÓN AL SISTEMA

Una vez fabricadas y armadas las PCBs, se colocaron al interior de un gabinete plástico que resguardará a los diversos componentes de la intemperie. En la figura 5.19 se observa que la caja está soportada por un tripié, y que éste además contiene al panel solar y al sensor de temperatura.

Las figuras 5.19, 5.20 y 5.21 muestran la instalación de la estación remota y de sus elementos de captación de energía solar y sensores.

Figura 5.19. Instalación de la estación remota.

Figura 5.20. Instalación de la tarjeta de circuito impreso dentro del gabinete de la estación remota.

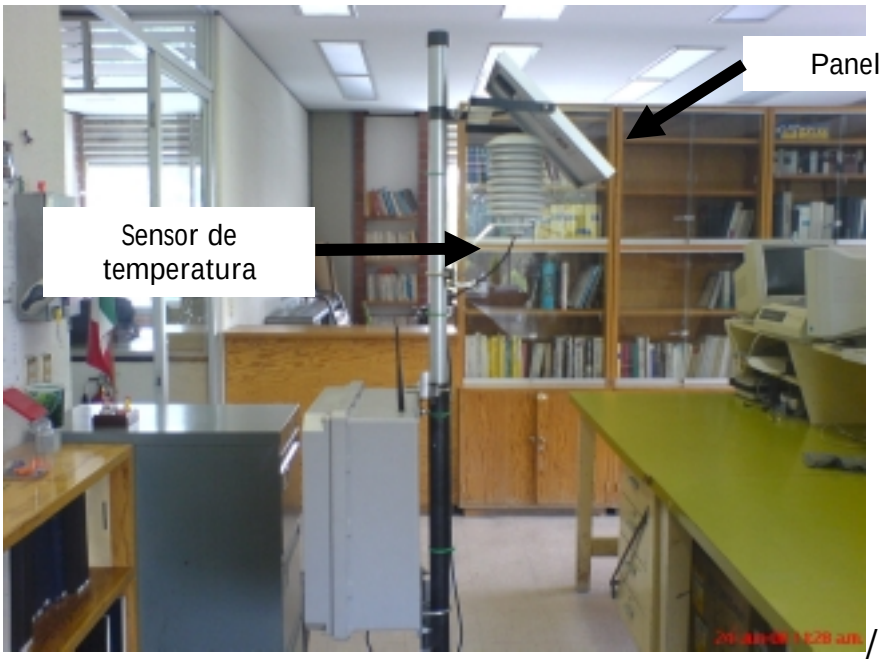


Figura 5.21. Instalación del panel solar y del sensor de temperatura en la estructura de la ER.

Una vez realizada la instalación de la estación remota, se procedió a realizar pruebas de enlace y comunicación entre la estación base (PC) y la estación remota, vía alámbrica e inalámbrica, como se muestra en la figura 5.22.

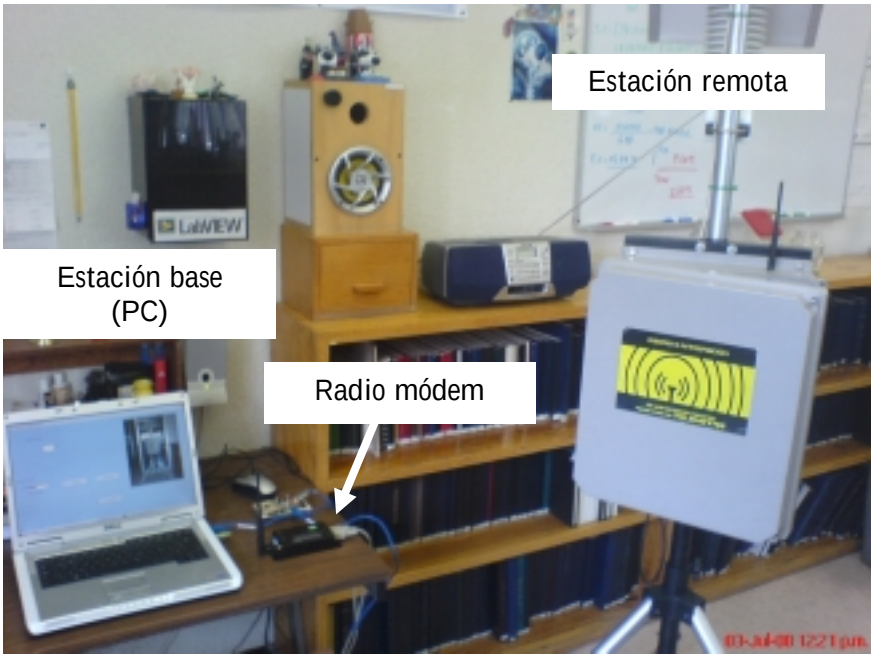


Figura 5.22. Pruebas de enlace entre la EB y ER.

En la comunicación inalámbrica, la estación remota se tuvo en operación 24 horas continuas, con el soporte de energía proporcionado por la batería y el panel solar, con objeto de verificar la operación de todos los módulos que la integran, observando un tiempo de carga completa de la batería en, aproximadamente, 5 horas, con un 80% de carga inicial en la batería; calentamiento nulo en los reguladores de voltaje que surten de energía tanto a la tarjeta principal de la estación remota como a la tarjeta auxiliar que alimenta a la sonda sensora de temperatura, con niveles de voltaje estables en 4.97 V y 2.56 V, respectivamente. Se adquirieron datos cada minuto de la temperatura ambiente y fueron enviados a la computadora de la estación base, vía radio módem. En relación con los medios alámbricos de comunicación, se probó el desempeño de las interfaces RS232 y USB para el flujo de datos en sitio, resultando satisfactoria desde el punto de vista de la integridad de los datos recibidos, con el detalle de que, en el caso de la interfaz USB, hubo que tomar en consideración, para el buen funcionamiento de la interfaz, tanto la configuración de parámetros tales como la asignación numérica del puerto, el número de bits de datos, la inclusión de bits de inicio y paro, la tasa de transferencia de datos y el tamaño del *buffer;* algunos de estos parámetros no sólo pertenecientes al controlador asociado con el circuito integrado de la interfaz USB, sino además, propios del puerto de la PC *host* , por el cual se lleva a cabo la comunicación de datos.

Con base en los resultados obtenidos y a las observaciones realizadas en el proceso, se tuvieron que realizar una serie de ajustes sobre la estación remota, en específico sobre la PCB principal, lo que llevó al diseño de la versión 1.0, la cual contiene correcciones a omisiones y errores detectados en la versión 0.0 De los principales correcciones, se tienen la referente a la alimentación de energía para el sensor de temperatura y a las interfaces de comunicación. Ya que en la versión 0.0 ambos módulos se encontraban como módulos aislados, lo cual además de demandar espacio físico dentro del gabinete, hacia más elaboradas las conexiones entre la PCB principal, los medios de comunicación y el sensor de temperatura. De hecho, las características de la versión 1.0 de la PCB de la estación remota, incluyen la integración de los conectores físicos USB y RS232, la disponibilidad de fuentes de 2.56, 5 y 12 V en la bornera principal, lo cual permitirá a futuro, la conexión de otro tipo de sensores. La figura 5.23 muestra los módulos adicionales entre la versión 0.0 y 1.0. y la figura 5.24 muestra la tarjeta de circuito impreso, de la versión 1.0, armada.

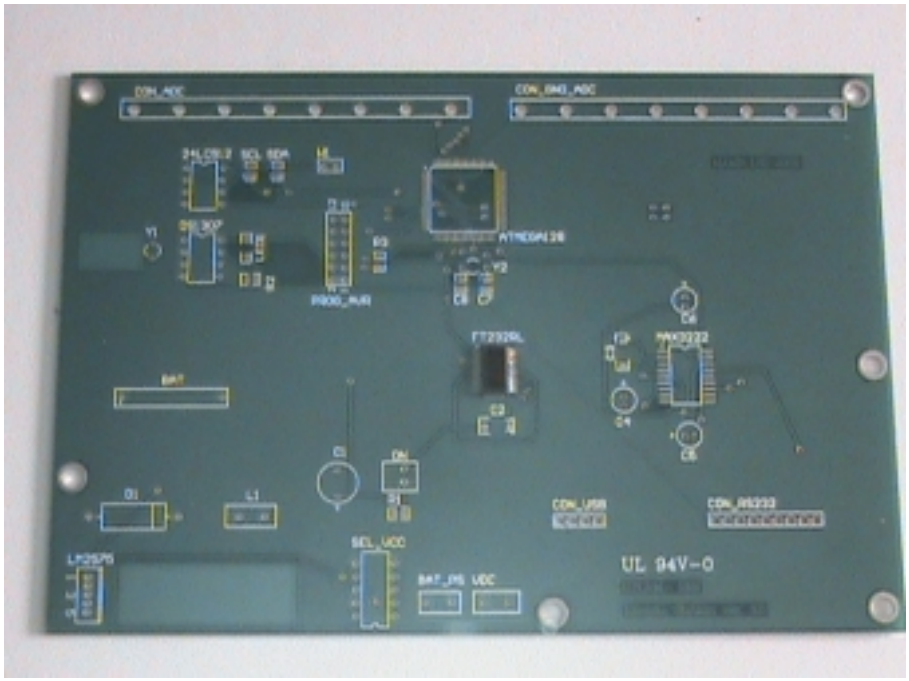


Figura 5.23. PCB de la estación remota versión 0.0.

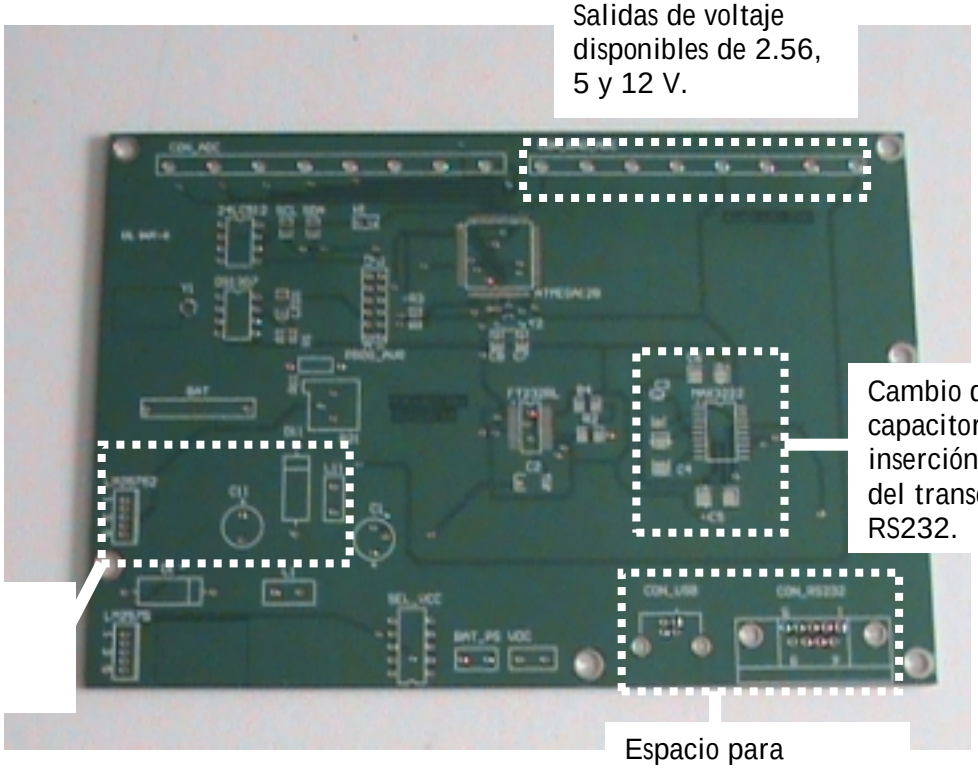


Figura 5.24. Mejoras hechas en la versión 1.0.

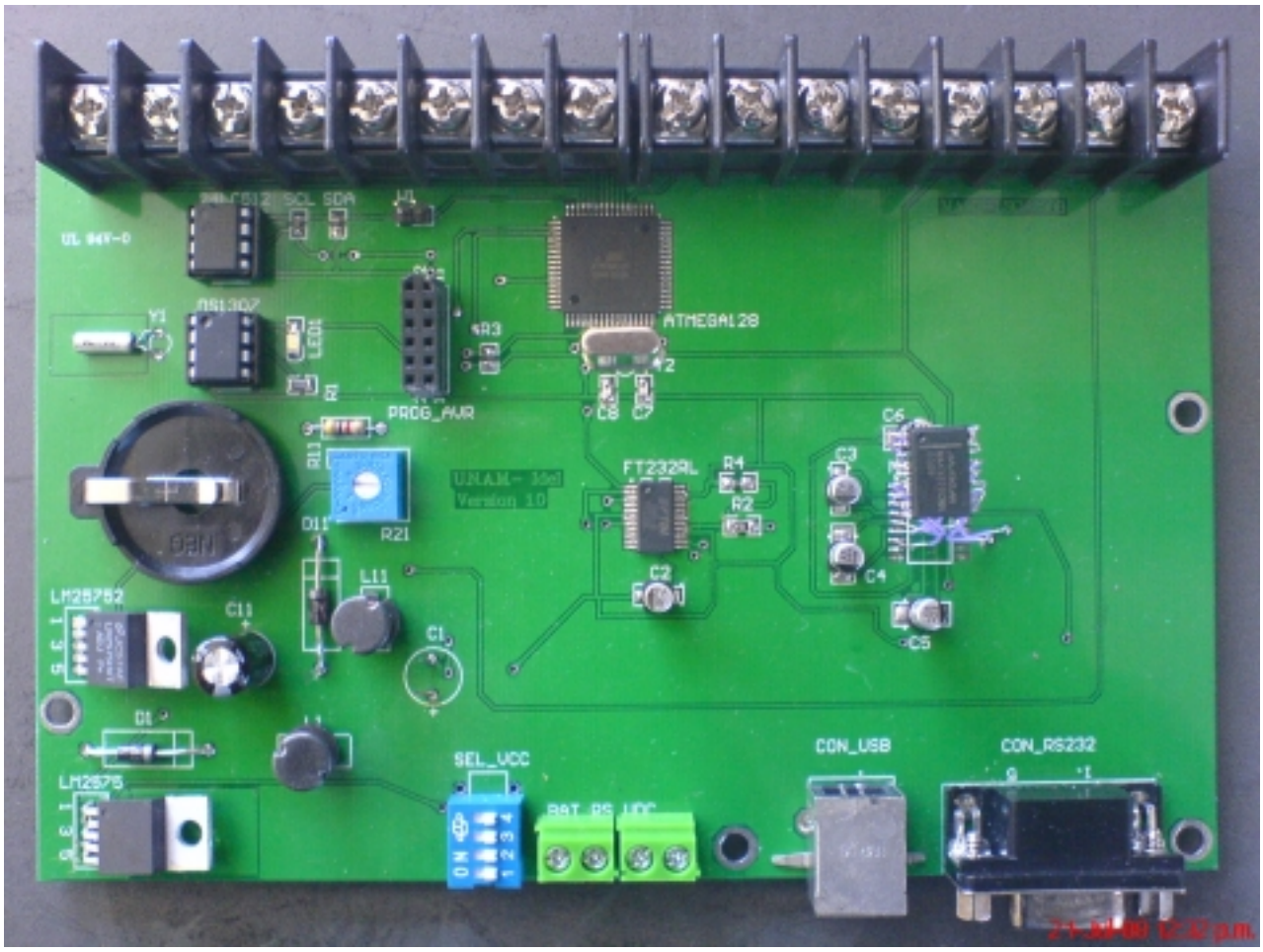


Figura 5.24. Tarjeta de circuito impreso versión 1.0 armada.

Cabe comentar que, funcionalmente hablando, con el desarrollo de la versión 1.0 uno de los logros más importantes fue la puesta a punto del módulo de interfaz USB, permitiendo realizar una transferencia de datos entre la tarjeta de la estación remota y la computadora de la estación base, misma que en la versión 0.0 no se logró por completo, así mismo se logró abatir el espacio que ocupa la tarjeta con la incorporación de los conectores físicos RS232 y USB, así como la fuente de alimentación para la sonda sensora de temperatura.

En el siguiente capítulo se encuentran reunidos los resultados finales y las conclusiones del desarrollo de la estación remota.

6.1. RESULTADOS

Con objeto de llegar a cubrir los objetivos iniciales planteados al inicio de este trabajo, éste fue programado, básicamente en 6 etapas de desarrollo:

La primera de ellas, de carácter general y práctico, consistente en el desarrollo de prototipos básicos, basados en dispositivos programables, con objeto de reforzar técnicas de ensamblado y optimización de circuitos.

La segunda etapa, ya más enfocados en el ámbito de desarrollo del tema de tesis, la documentación acerca de medios de comunicación y pruebas sencillas con algunos de ellos, como por ejemplo el radio módem y módem telefónico.

La tercera etapa consistió en la documentación sobre tópicos de programación y el desarrollo práctico de proyectos básicos con microcontroladores, con objetivo de familiarizarse con ellos, utilizando para ello, tarjetas de desarrollo de propósito general.

La cuarta etapa, consistente en la documentación y capacitación sobre programación en software orientado a objetos, para el desarrollo de interfaces gráficas.

La quinta etapa del proyecto, consistió en la investigación, ensamblado y evaluación de un sistema prototipo de adquisición de datos el cual, integraba ya algunos de los conceptos desarrollados en etapas anteriores, lo cual sirvió como plataforma para, en una sexta etapa integrar un sistema aún más completo y robusto de adquisición de datos, enmarcado dentro de un ámbito de aplicación específico y estructurado en bloques funcionales de conversión analógica a digital, almacenamiento de datos, interfaces de comunicación y referencia de tiempo, integrado dentro de una tarjeta de circuito impreso fabricada de forma profesional, lo que trajo consigo la generación de un primer prototipo.

Una vez que se terminó de integrar al prototipo, se prosiguió con una serie de pruebas de evaluación, mismas que llevaron a generar una serie de observaciones y correcciones, las cuales fueron

realizadas en una segunda versión, con la cual se obtuvieron mejores resultados a nivel "estético y funcional".

Concluida la etapa de evaluación y pruebas, se obtuvo el prototipo de una estación remota, que si bien para efectos demostrativos se ha caracterizado como una estación meteorológica, fue concebida bajo el concepto de ser un sistema genérico, el cual, previo cambio, configuración y acondicionamiento de los sensores, puede colocarse en otros ámbitos como el sísmico o hidrológico, por mencionar sólo algunos de ellos.

Entre las características que tiene el prototipo desarrollado se pueden mencionar las siguientes:

- Se tiene una estación remota para fines telemetría, con un costo aproximado, en material, de $19,090.304 M.N., en la tabla 6.1. se muestra el costo desglosado, la cual fue construida en un tiempo aproximado de 15 meses, a partir de algunos elementos de fácil adquisición en los mercados nacional e internacional, por medio de canales de comercialización como tiendas establecidas y compañías trasnacionales de venta de componentes electrónicos por catálogo con representantes en México.

Estación remota desarrollada en el Instituto de Ingeniería, U.N.A.M		
ITEM	**PRODUCTO**	**PRECIO**
1	Tripié	$ 1,000.00
1	PCB	$1,336.00
1	Sensor de temperatura	$1,421.00
1	Protector de radiación solar	$3,769.13
1	Radio Módem	$3,452.174
1	Controlador de carga 12V/3A	$239.00
1	Batería recargable 12V/7Ah	$370.00
1	Panel Solar 12V/10W	$703.00
	Circuitos integrados	$800.00
1	Gabinete	$5,000.00
	Material diverso	$1,000.00
Costo total		$19,090.304

Tabla 6.1. Costo desglosado de la estación remota.

- El sistema tiene la capacidad de sensar hasta 8 variables a través de sensores de terminación simple y/o 4 de terminación diferencial, con niveles de polarización de 2.56 V, 5V y 12V, con capacidad de extender los márgenes hacia otros niveles.

- Integra dos protocolos de transmisión de datos; uno de velocidad media (RS232) y otro de alta velocidad (USB). Esta última es una característica novedosa en equipos de su clase.

- Se tiene un sistema que se encuentra protegido contra la intemperie y soportado por su propia estructura mecánica de características robustas.

- La estación cuenta con 2 sistemas de alimentación de energía, uno con base en un panel fotovoltaico, que le permite una autonomía de hasta, aproximadamente, 24 horas, utilizado para aplicaciones en campo, y un sistema basado en un eliminador de voltaje, alimentado de la red convencional que le permite su alimentación en ambientes interiores como laboratorios o estancias.

- Presenta una interfaz de software amigable con el usuario, de fácil entendimiento, y que permite la generación de ejecutables, mismos que permiten su portabilidad.

- La Estación remota tiene la capacidad de expandir los medios de comunicación con los que cuenta actualmente, a dispositivos de telefónico convencional, red GSM y de tipo satelital, para transmisión y recepción de datos a distancia y en sitio.

- Se desarrolló un equipo de características de diseño propio, basado en ideas de equipos similares en el mercado internacional.

La comparativa contra otros equipos comerciales, por el momento no cabe, debido a que la estación remota construida no ha sido probada en algún laboratorio que permita su certificación, como sucede con equipos comerciales.

6.2. CONCLUSIONES

Se cumplió con el objetivo inicial del proyecto, el cual consistía en el diseño e integración de una estación telemétrica que permitiera el sensado remoto de variables, ya sean físicas o químicas, de fácil instalación y operación, la cual posee una arquitectura, con algunos de sus elementos que cumplen con estándares internacionales, como en el caso de la tarjeta de circuito impreso y del sensor de temperatura.

El poder contar con una estación remota de tipo "adaptativo", permite el sensado remoto de variables, físicas o químicas, en distintos ámbitos y en condiciones de tiempo prolongadas, en las cuales un operador humano no sería necesario.

En las pruebas de operación de la estación, se verificó que el equipo es funcional, confiable y seguro en su operación, sujeto todavía a mejoras y optimizaciones, como por ejemplo en hardware, diseñar e implementar un gabinete que contenga a la tarjeta principal, de la cual únicamente se tenga acceso a una bornera para la conexión de sensores y fuentes de alimentación de energía; en cuanto a software, inicialmente, desarrollar una base de datos robusta, que permita su consulta a través de Internet, desarrollar una interfaz para dispositivos móviles tales como *PDA*'s o *SmartPhone*'s. El sistema desarrollado puede servir como plataforma para futuros desarrollos en el ámbito de la telemetría, el cual en nuestro país está medianamente desarrollado.

6.3. RECOMENDACIONES

En cualquier desarrollo científico o tecnológico siempre habrá elementos por incorporar, corregir o suprimir, que le confieran vigencia y confiabilidad. Por lo tanto, el desarrollo de la estación remota es un proyecto sujeto a múltiples mejoras y optimizaciones.

Uno de los principales problemas que se tuvo en el desarrollo de este proyecto fue la falta de financiamiento. Por tanto, pudiera tenerse la impresión acerca de la limitación en cuanto a los alcances del prototipo, sobre todo en el aspecto, por ejemplo de los sensores; sin embargo, convendría hacerse de recursos que permitan subsanar esa falta de sensores, pudiendo acoplar sensores de humedad, velocidad y dirección del viento.

Internet es actualmente una vía de datos infranqueable, por lo tanto convendría diseñar un esquema de adquisición que permita su almacenamiento masivo y periódico en un equipo central y poder vincular esta información hacia una página dinámica de Internet, en la cual se pudieran visualizar datos en tiempo real, así como consultar y generar registros históricos.

Personalmente el desarrollo de este proyecto me dejó satisfecho, en el sentido de haber podido diseñar y construir un sistema funcional que presentó una alta complejidad, principalmente por las limitaciones iniciales en cuanto al conocimiento de tópicos relacionados al momento de arrancar el proyecto. Sin embargo, una vez que vista la estación en pié y funcionando, queda la satisfacción personal no solo de ver en pié un proyecto por el cual se ha luchado hasta el cansancio, sino por ver que, un equipo de las características con las cuales la estación remota fue concebida, haya podido ser construido utilizando diseños propios, mismo que nos demuestra por enésima vez que en México podemos desarrollar equipos propios, sólo que hace falta un "pequeño" ingrediente indispensable…el financiamiento.

```
//Programa para la digitalización de la señal analógica procedente de la
//sonda de temperatura de CSI 108-L.
//*****************************************************************************
#include <avr/io.h>
#include <stdio.h>
#include <avr/interrupt.h>
#include <math.h>
#include <stdlib.h>
//*****************************************************************************
//Definición de constantes a utilizar
//*****************************************************************************
const float A = 0.000827111;
const float B = 0.000208802;
const float C = 0.000000080592;
//*****************************************************************************
//Declaración de variables utilizadas en el programa
//*****************************************************************************
long int adc_value;
long int resultado;
float RT;
float LnRes;
float LnRes3;
float  T_C;
char s[9];
//*****************************************************************************
//Subrutinas de inicialización de los periféricos USART y ADC
//*****************************************************************************
void init_adc(void)
{
        ADMUX=0b11000000;
        ADCSRA = _BV(ADEN)|_BV(ADFR)|_BV(ADIE)|_BV(ADPS2)|_BV(ADPS0)|_BV(ADSC);
        sei();

}

void init_USART0(void)
{
        UBRR0L = 25;
        UCSR0B = (1<<RXEN0)|(1<<TXEN0)|(1<<RXCIE0);
        UCSR0C = (0<<7)|(1<<USBS0)|(3<<UCSZ00);
}

void calcule_temp(void)
{
        resultado = adc_value;
        RT = (1024000/resultado)-41000;
        LnRes = log(RT);
```

```
        LnRes3 = LnRes * LnRes * LnRes;
        T_C = (1/(A + B * LnRes + C * LnRes3))-273.15;
        //send_temp();
}


void usart_putc(unsigned char c)
{
        while(!(UCSR0A & (1<<UDRE0)));
        UDR0 = c;
}

void usart_puts(char *s)
{
        while(*s)
        {
                usart_putc(*s);
                s++;
        }
}

//*****************************************************************************************
//Subrutina de interrupción
//*****************************************************************************************
ISR(SIG_ADC)
{
    adc_value = ADC;
}
//*****************************************************************************************
//Declaración de la subrutina principal
//*****************************************************************************************
int main(void)
{
        init_USART0();
        init_adc();
        while(1)
        {
           calcule_temp();
                dtostrf(RT,6,1,s);
           usart_puts(s);
                usart_putc(0x0d);
        }
}
```

```
//Programa preeliminar para evaluar el comportamiento del reloj en tiempo real DS1307
//con el microcontrolador ATMEGA128.
//*****************************************************************************************
#include <avr/io.h>
#include <avr/interrupt.h>
//*****************************************************************************************
//Definición de los estados del bus TWI
//*****************************************************************************************
#define ADD_SL_W        0xD0
#define ADD_SL_R        0xD1
#define MT_START        0x08
#define MT_REP_START    0x10
#define MT_SLA_ACK      0x18
#define MT_SLA_NACK     0x20
#define MT_DATA_ACK     0x28
#define MT_DATA_NACK    0x30
#define MT_ARB_LOST     0x38
#define MR_DATA_ACK     0x58

typedef unsigned chay byte;

//*****************************************************************************************
//Definición de los avisos que se enviarán a través de la USART del ATMEGA128
//*****************************************************************************************
char string1[]="Datos escritos!";
char string2[]="Datos leídos:";
char string3[]="Error al escribir: MT_START";
char string4[]="Error al leer: MT_START";
char string5[]="Escritura y lectura del RTC DS1307";
char string6[]="Teclea una opción...";
char string7[]="E: escribe y L: lee";
char string8[]="Otro caso...";
char string9[]="Error en START";
char string10[]="Error: NO ACK";
char string11[]="Error: NO ADD SLA";
char string12[]="FIN DE DATOS";
//*****************************************************************************************
//Definición de los prototipos de funciones para el control de periféricos
//*****************************************************************************************
void twi_master_init(void)
{
        TWBR = 12;
        TWCR = _BV(TWEA)|_BV(TWEN)|_BV(TWIE);
        sei();
}

void init_USART(void)
{
        UBRR0L = 25;
```

```
        UCSR0B = (1<<RXEN0)|(1<<TXEN0)|(1<<RXCIE0);
        UCSR0C = (0<<7)|(1<<USBS0)|(3<<UCSZ00);
        sei();
}

void usart_putc(unsigned char c)
{
        while(!(UCSR0A & (1<<UDRE0)));
        UDR0 = c;
}

void usart_puts(char *string)
{
        while(*string)
   {
                usart_putc(*string);
                string++;
        }
}
//*****************************************************************************************
//Prototipos de funciones para el manejo del protocolo TWI
//*****************************************************************************************
int rtc_write_data(uint8_t add_mem)
{
begin:
                TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
                while(!(TWCR & (1<<TWINT)));
                if((TWSR & 0xF8) == MT_START)
                {
                        goto send_byte_ctrl_w;
                }
                if((TWSR & 0xF8) != MT_START)
                {
                        usart_puts(string9);
                }
//-----------------------------------------------------------------------------------------
send_byte_ctrl_w:
                TWDR = 0xd0;
            TWCR = (1<<TWINT)|(1<<TWEN);
                while(!(TWCR & (1<<TWINT)));
                if((TWSR & 0xF8) == MT_SLA_ACK)
                {
                        goto add_memo;
                }
                if((TWSR & 0xF8) == MT_SLA_NACK)
                {
                        goto error_ack;
                }
//-----------------------------------------------------------------------------------------
```

```
add_memo:
                TWDR = add_mem;
                TWCR = (1<<TWINT)|(1<<TWEN);
                while(!(TWCR & (1<<TWINT)));
                if((TWSR & 0xF8) == MT_DATA_ACK)
                {
                        goto send_dato1;
                }
                if((TWSR & 0xF8) == MT_DATA_NACK)
                {
                        goto error_data_ack;
                }
//-------------------------------------------------------------------------------
send_dato1:
                TWDR = 0x00;
                TWCR = (1<<TWINT)|(1<<TWEN);
                while(!(TWCR & (1<<TWINT)));
                if((TWSR & (0xF8) == MT_DATA_ACK)
                {
                        goto send_dato2;
                }
                if((TWSR & 0xF8) == MT_DATA_NACK)
                {
                        goto error_data_ack;
                }
//-------------------------------------------------------------------------------
send_dato2:
                TWDR = 0x00;
                TWCR = (1<<TWINT)|(1<<TWEN);
                while("(TWCR & (1<<TWINT)));
                if((TWSR & 0xF8) == MT_DATA_ACK)
                {
                        goto send_dato3;
                }
                if((TWSR & 0xF8) == MT_DATA_NACK)
                {
                        goto error_data_ack;
                }
//-------------------------------------------------------------------------------
send_dato3:
                TWDR = 0x69;
                TWCR = (1<<TWINT)|(1<<TWEN);
                while(!(TWCR & (1<<TWINT)));
                if((TWCR & 0xF8) == MT_DATA_ACK)
                {
                        goto send_dato4;
                }
                if((TWCR & 0xF8) == MT_DATA_NACK)
                {
```

```
                goto error_data_ack;
        }
//-------------------------------------------------------------------------------------
send_dato4:
        TWDR = 0x04;
        TWCR = (1<<TWINT)|(1<<TWEN);
        while(!(TWCR & (1<<TWINT)));
        if((TWCR & 0xF8) == MT_DATA_ACK)
        {
                goto send_dato5;
        }
        if((TWCR & 0xF8) == MT_DATA_NACK)
        {
                goto send_dato6;
        }
//-------------------------------------------------------------------------------------
send_dato5:
        TWDR = 0x28;
        TWCR = (1<<TWINT)|(1<<TWEN);
        while(!(TWCR & (1<<TWINT)));
        if((TWCR & 0xF8) == MT_DATA_ACK)
        {
                goto send_dato6;
        }
        if((TWCR & 0xF8) == MT_DATA_NACK)
        {
                goto error_data_ack;
        }
//-------------------------------------------------------------------------------------
send_dato6:
        TWDR = 0x02;
        TWCR = (1<<TWINT)|(1<<TWEN);
        while(!(TWCR & (1<<TWINT)));
        if((TWCR & 0xF8) == MT_DATA_ACK)
        {
                goto send_dato7;
        }
        if((TWCR & 0xF8) == MT_DATA_NACK)
        {
                goto error_data_ack;
        }
//-------------------------------------------------------------------------------------
send_dato7:
        TWDR = 0x08;
        TWCR = (1<<TWINT)|(1<<TWEN);
        while(!(TWCR & (1<<TWINT)));
        if((TWCR & 0xF8) == MT_DATA_ACK)
        {
                goto send_dato8;
```

```
		}
		if((TWCR & 0xF8) == MT_DATA_NACK)
		{
			goto error_data_ack;
		}
//-------------------------------------------------------------------------------------
send_dato8:
		TWDR = 0x10;
		TWCR = (1<<TWINT)|(1<<TWEN);
		while(!(TWCR & (1<<TWINT)));
		if((TWCR & 0xF8) == MT_DATA_ACK)
		{
			goto quit;
		}
		if((TWCR & 0xF8) == MT_DATA_NACK)
		{
			goto error_data_ack;
		}
//-------------------------------------------------------------------------------------
quit:
		TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
		usart_puts(string1);
		usart_putc(0x0d);
		return 1;
//-------------------------------------------------------------------------------------
error_ack:
		TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
		usart_putc(0x0d);
		usart_puts(string11);
		return -1;
//-------------------------------------------------------------------------------------
error_data_ack:
		TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
		usart_puts(string10);
		usart_putc(0x0d);
		return -1;
}
//*************************************************************************************
//Prototipo de función para la lectura de los datos del RTC
//*************************************************************************************
byte rtc_read_data(byte add_ds)
{
	uint8_t temp;

begin:
	TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
	while(!(TWCR & (1<<TWINT)));
	if((TWSR & 0xF8) == MT_START)
	{
```

```
                goto send_ctrl_byte_w;
        }
        if((TWSR & 0xF8) != MT_START)
        {
                usart_puts(string9);
        }
//-------------------------------------------------------------------------------
send_ctrl_byte_w:
                TWDR = 0xd0;
                TWCR = (1<<TWINT)|(1<<TWEN);
                while(!(TWCR & (1<<TWINT)));
                if((TWSR & 0xF8) == MT_SLA_ACK)
                {
                        goto tx_add_dir1;
                }
                if((TWSR & 0xF8) == MT_SLA_NACK)
                {

                }
//-------------------------------------------------------------------------------
tx_add_dir1:
                TWDR = add_ds;
                TWCR = (1<<TWINT)|(1<<TWEN);
                while(!(TWCR & (1<<TWINT)));
                if((TWSR & 0xF8) == MT_DATA_ACK)
                {
                        goto send_rep_start;
                }
                if((TWSR & 0xF8) == MT_DATA_NACK)
                {

                }
//-------------------------------------------------------------------------------
send_rep_start:
                TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTA);
                while(!(TWCR & (1<<TWINT)));
                if((TWSR & 0xF8) == MT_REP_START)
                {
                        goto send_ctrl_byte_r;
                }
                if((TWSR & 0xF8) != MT_REP_START)
                {

                }
//-------------------------------------------------------------------------------
send_ctrl_byte_r:
                TWDR = 0xd1;
                TWCR = (1<<TWINT)|(1<<TWEN);
                while(!(TWCR & (1<<TWINT)));
```

```
		if((TWSR & 0xF8) == MT_DATA_ACK)
		{
			goto tx_data;
		}
		if((TWSR & 0xF8) == MT_DATA_NACK)
		{

		}
//-------------------------------------------------------------------------------
tx_data:
		TWCR = (1<<TWINT)|(1<<TWEN);
		while(!(TWCR & (1<<TWINT)));
		if((TWSR & 0xF8) == MR_DATA_ACK)
		{
			temp = TWDR;
		}
		if((TWSR & 0xF8) != MR_DATA_ACK)
		{

		}
//-------------------------------------------------------------------------------
tx_stop:
		TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
//*******************************************************************************
//Transmisión y recepción por la USART
//*******************************************************************************
ISR(SIG_UART0_RECV)
{
	uint8_t data;
	uint8_t sec,min,hour,date,month,year;
	uint8_t add_ptr;
	data = UDR0;

	switch(data)
	{
		case('E'):
			usart_putc(0x0d);
			add_ptr = 0x00;
			rtc_write_data(add_ptr);
			break;
		case('L'):
			usart_putc(0x0d);
			sec = rtc_read_data(0);
			while(!(UCSR0A & (1<<UDRE0)));
			UDR0 = sec;
			usart_putc(0x0d);
			sec = rtc_read_data(1);
			while(!(UCSR0A & (1<<UDRE0)));
			UDR0 = min;
```

```
                usart_putc(0x0d);
                sec = rtc_read_data(2);
                while(!(UCSR0A & (1<<UDRE0)));
                UDR0 = hour;
                usart_putc(0x0d);
                sec = rtc_read_data(3);
                while(!(UCSR0A & (1<<UDRE0)));
                UDR0 = day;
                usart_putc(0x0d);
                sec = rtc_read_data(4);
                while(!(UCSR0A & (1<<UDRE0)));
                UDR0 = date;
                usart_putc(0x0d);
                sec = rtc_read_data(5);
                while(!(UCSR0A & (1<<UDRE0)));
                UDR0 = month;
                usart_putc(0x0d);
                sec = rtc_read_data(6);
                while(!(UCSR0A & (1<<UDRE0)));
                UDR0 = year;
                usart_putc(0x0d);
                usart_puts(string12);
                usart_putc(0x0d);
                usart_putc(0x0d);
                break;
        default:
                usart_puts(string8);
                usart_putc(0x0d);
        }
}
//*****************************************************************************************
//Declaración de la función principal
//*****************************************************************************************
int main(void)
{
        init_USART();
        twi_master_init();
        usart_puts(string5);
        usart_putc(0x0d);
        usart_puts(string6);
        usart_putc(0x0d);
        usart_puts(string7);
        usart_putc(0x0d)
        while(1)
        {
        }
}
```

```
//Programa para probar el funcionamiento en lectura y escritura de la memoria
//EEPROM 24LC512
//*****************************************************************************
#include <avr/io.h>            //Encabezados para el uso de io del microcontrolador
#include <avr/interrupt.h>
//*****************************************************************************
//Definiciones y códigos de estado para el protocolo TWI
//*****************************************************************************
#define ADD_SL_W     0xA0 //Dirección del esclavo + WRITE
#define ADD_SL_R     0xA1 //Dirección del esclavo + READ
#define MT_START     0x08 //Código para señal de START por el MASTER
#define MT_REP_START 0x10 //Código de señal de REPEATED START por el MASTER
#define MT_SLA_ACK   0x18 //Código de señal de confirmación del esclavo
#define MT_SLA_NACK  0x20 //Código de señal de no confirmación del esclavo
#define MT_DATA_ACK  0x28 //Código de señal de confirmación de datos por esclavo
#define MT_DATA_NACK 0x30 //Código de señal de no confirmación de datos por esclavo
#define MT_ARB_LOST  0x38 //Código de señal para reportar fallas en el bus de datos
#define MR_DATA_ACK  0x58 //Código de señal para reportar que el dato ha sido recibido
                                                    //y no se enviará una confirmación.
//*****************************************************************************
//Definición de las cadenas de aviso
//*****************************************************************************

char string1[]="Dato escrito!";                              //Cadenas de texto para ser transmitidas
char string2[]="Dato leido:";                               //por el subsistema USART y verificar el
char string3[]="Error al escribir MT_START";   //estado de transferencia de los datos
char string4[]="Error al leer MT_START";       //por medio del protocolo TWI.
char string5[]="Escritura a memoria EEPROM";
char string6[]="Teclea una opcion:";
char string7[]="E escribe y  L lee";
char string8[]="****************************";
char string9[]="ERROR:NO ACK";
char string10[]="Otro caso...";
char string11[]="Bye...";
//*****************************************************************************
//Definición de los prototipos de funciones
//*****************************************************************************

void twi_master_init(void)    //Inicialización del subsistema TWI del ATMEGA32 como
{                             //maestro.
        TWBR = 12;
        TWCR = _BV(TWEA)|_BV(TWEN)|_BV(TWIE);
        sei();                                  //Habilitación de las interrupciones internas
}

void init_USART(void)         //Inicialización del subsistema USART
{
        UBRRL0 = 25;                //4 MHz, 8,n,1,n.
        UCSR0B = (1<<RXEN0)|(1<<TXEN0)|(1<<RXCIE0);
```

```
        UCSR0C = (1<<URSEL0)|(1<<USBS0)|(3<<UCSZ00);
        sei();
}

void usart_putc(unsigned char c)  //Envío de caracteres ASCII a través de USART.
{
        while(!(UCSR0A & (1<<UDRE0)));
        UDR0 = c;
}

void usart_puts(char *string)    //Envío de cadenas de caracteres a través del USART.
{                           //Toma como función de apoyo usart_putc para enviar
        while(*string)              //caracter x caracter de la cadena vía USART a la PC.
        {
                usart_putc(*string);     //Invocación a la función putc para el envío de caracter
                string++;               //x caracter de la cadena a la que apunta *string
        }
}


//*****************************************************************************
//Prototipo de función para la escritura de un byte
//*****************************************************************************

int eeprom_write_byte(uint8_t add_H, uint8_t add_L, char buf) //Definición de la función de
{                                           //escritura de 1 byte, cuyos
         //argumentos de entrada son 3: byte de dirección BAJA/ALTA y el dato a escribir
        DDRA = _BV(DDA0)|_BV(DDA1)|_BV(DDA2)|_BV(DDA3)|_BV(DDA4); //Habilitación de leds

//testigos.
//-----------------------------------------------------------------------------
begin:                                                          //Envío de la señal de START para inicializar
                TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN); //la comunicación por el protocolo TWI.
                while(!(TWCR & (1<<TWINT)));          //Se utiliza la técnica de poleo para verifi-
      if((TWSR & 0xF8) == MT_START)         //car el estado de las banderas de habilita-
                        {                           //ción del bus, para inicializar la transfe-
                                PORTA = _BV(PA0);             //rencia de datos
                                goto send_byte_ctrl_W; //Si se envió la condición de START, salta a
                        }                       //siguiente subrutina
      if((TWSR & 0xF8) != MT_START)
                        {
                                while(!(UCSRA & (1<<UDRE)));
                                UDR = TWSR;           //En caso de error, muestra el contenido del TWSR
                                goto error1;          //y salta a subrutina de error.
                        }
//-----------------------------------------------------------------------------
send_byte_ctrl_W:          //Subrutina para envío de dirección del dispositivo esclavo +
                TWDR = 0xA0;              //bit de dirección de datos (R/W): 1010 + 0000 + 0
                TWCR = (1<<TWINT)|(1<<TWEN);
```

```
                while(!(TWCR & (1<<TWINT)));    //Por poleo, verifico que el bit de interrupción de
        if((TWSR & 0xF8) == MT_SLA_ACK)     //TWI se limpie, para que pueda seguir la ejecución
                  {                         //del programa. También reviso el código de estado
                        goto direccion_alta;    //que ha enviado el esclavo, en caso de estar bien,
                    }                       //salta a la siguiente subrutina.

        if((TWSR & 0xF8) == MT_SLA_NACK)
                    {
                        goto error;     //En caso de que algo haya salido mal, vete a subrutina de
                    }                   //error.
//---------------------------------------------------------------------------
direccion_alta:         //Subrutina de envío de la parte alta de la localidad de memoria
                TWDR = add_H;     //en la cual se ha de escribir el byte.
                TWCR = (1<<TWINT)|(1<<TWEN);
                while(!(TWCR & (1<<TWINT))); //Revisamos por poleo, que se limpie la bandera de --
                if((TWSR & 0xF8) == MT_DATA_ACK) //interrupción de la TWI, para después proceder a
                    {                       //verificar el código de estado. En caso de estar
                        goto direccion_baja;     //bien, salta a la siguiente subrutina.
                    }
                if((TWSR & 0xF8) == MT_DATA_NACK) //En caso de que el código de estado, marque
                    {                       //algún tipo de error, salta a la subrutina de
                        goto error;             //error.
                    }
//---------------------------------------------------------------------------
direccion_baja:         //Subrutina de envío de la parte baja de la localidad de memoria
                TWDR = add_L;    //en la cual se ha de escribir el byte.
                TWCR = (1<<TWINT)|(1<<TWEN);
                while(!(TWCR & (1<<TWINT))); //Revisamos por poleo, que se limpie la bandera de --
                if((TWSR & 0xF8) == MT_DATA_ACK) //interrupción de la TWI, para después proceder a
                    {                       //verificar el código de estado. En caso de estar
                        goto send_dato;         //bien, salta a la siguiente subrutina.
                    }
                if((TWSR & 0xF8) == MT_DATA_NACK) //En caso de que el código de estado, marque
                    {                       //algún tipo de error, salta a la subrutina de
                        goto error;           //error.
                    }
//---------------------------------------------------------------------------
send_dato:              //Subrutina para envío del dato que se va a escribir en la memoria
                TWDR = buf;     //buf es la variable que contiene el dato que se va a escribir
                TWCR = (1<<TWINT)|(1<<TWEN); //Se habilitan el bus y las interrupciones
                while(!(TWCR & (1<<TWINT))); //Por poleo, se verifica el estado de la interrupción
                if((TWSR & 0xF8) == MT_DATA_ACK) //cuando se haya limpiado el bit de interrupción
                      {             //comprobará el código de estado,
                        goto quit;              //en caso de ser el código esperado, salta a la
                    }                       //siguiente subrutina
                if((TWSR & 0xF8) == MT_DATA_NACK) //En caso de no ser el código esperado
                    {                       //salta a la subrutina de errores.
                        goto error;
                    }
```

```
//-------------------------------------------------------------------
quit:
                TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
                PORTA = _BV(PA1);
                usart_puts(string1);
                usart_putc(0x0d);
                return 1;
//*************************************************************************************
//Definición de rutinas, por si existieron problemas en la transfencia de datos
//*************************************************************************************
error1:
          TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
                PORTA = _BV(PA2);
                usart_putc(0x0d);
                while(!(UCSRA & (1<<UDRE)));
                UDR = TWSR;;
                usart_putc(0x0d);
                return -1;

error:
                TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
                usart_puts(string9);
                usart_putc(0x0d);
                return -1;
}

//*************************************************************************************
//Prototipo de función para la lectura de un byte
//*************************************************************************************

int eeprom_read_byte(uint8_t add_H_R, uint8_t add_L_R, char *buf)
{
        DDRD = _BV(DDD6);
//-------------------------------------------------------------------------
begin:
                TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
                while(!(TWCR & (1<<TWINT)));
                if((TWSR & 0xF8) == MT_START)
                {
                        goto send_ctrl_byte;
                }
                if((TWSR & 0xF8) != MT_START)
                {
                        goto error2;
                }
//-------------------------------------------------------------------------
send_ctrl_byte:
                TWDR = 0xa0;
                TWCR=(1<<TWINT)|(1<<TWEN);
```

```
                while(!(TWCR & (1<<TWINT)));
                if((TWSR & 0xF8) == MT_SLA_ACK)
                {
                        goto direccion_alta_R;
                }

                if((TWSR & 0xF8) == MT_SLA_NACK)
                {
                        goto error;
                }
//-------------------------------------------------------------------------------
direccion_alta_R:
                TWDR = add_H_R;
                TWCR = (1<<TWINT)|(1<<TWEN);
                while(!(TWCR & (1<<TWINT)));
                if((TWSR & 0xF8) == MT_DATA_ACK)
                {
                        goto direccion_baja_R;
                }

                if((TWSR & 0xF8) == MT_DATA_NACK)
                {
                        goto error;
                }
//-------------------------------------------------------------------------------
direccion_baja_R:
                TWDR= add_L_R;
                TWCR = (1<<TWINT)|(1<<TWEN);
                while(!(TWCR & (1<<TWINT)));
                if((TWSR & 0xF8) == MT_DATA_ACK)
                {
                        goto send_rep_start;
                }
                if((TWSR & 0xF8) == MT_DATA_NACK)
                {
                        goto error;
                }

//-------------------------------------------------------------------------------
send_rep_start:
                TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTA);
                while(!(TWCR & (1<<TWINT)));
                if((TWSR & 0xF8) == MT_REP_START)
                {
                        goto send_ctrl_byte_R;
                }
                if((TWSR & 0xF8) != MT_REP_START)
                {
                        goto error;
```

```
                }
//-------------------------------------------------------------------------------
send_ctrl_byte_R:
                TWDR = 0xa1;
                TWCR = (1<<TWINT)|(1<<TWEN);
                while(!(TWCR & (1<<TWINT)));
                if((TWSR & 0xF8) == MT_DATA_ACK)
                {
                        goto tx_byte;
                }

                if((TWSR & 0xF8) == MT_DATA_NACK)
                {
                        goto error;
                }
//-------------------------------------------------------------------------------
tx_byte:
                TWCR = (1<<TWINT)|(1<<TWEN);
                while(!(TWCR & (1<<TWINT)));
                if((TWSR & 0xF8) == MR_DATA_ACK)
                {
                        *buf = TWDR;
                }

                if((TWSR & 0xF8) != MR_DATA_ACK)
                {
                        goto error;
                }
//-------------------------------------------------------------------------------
quit:
                TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
                usart_puts(string2);
                usart_putc(0x0d);
                return -1;
//*******************************************************************************
//Definición de rutinas de detección de errores en la transferencia de datos
//*******************************************************************************

error:
                TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
                usart_puts(string9);
                return -1;
error2:
                TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
                PORTD = _BV(PD6);
                usart_puts(string4);

}
```

```
//*****************************************************************************
//Transmisión y recepción por USART
//*****************************************************************************

ISR(SIG_UART_RECV)
{
        char buffer;
        char buffer2;
        uint8_t add_targetH;
        uint8_t add_targetL;
        uint8_t data;
        data = UDR0;
        switch(data)
        {
                case('E'):
                        buffer=0x4d;
                        add_targetH=0x00;
                        add_targetL=0x00;
                        eeprom_write_byte(add_targetH,add_targetL,buffer);
                        buffer=0x41;
                        eeprom_write_byte(add
                        while(!(UCSRA & (1<<UDRE)));
                        UDR0 = 'W';
                        usart_putc(0x0d);
                        break;

      case('L'):
                        add_targetH=0x00;
                        add_targetL=0x00;
                        eeprom_read_byte(add_targetH,add_targetL,&buffer2);
                        while(!(UCSRA & (1<<UDRE)));
                        UDR0 = buffer2;
                        while(!(UCSRA & (1<<UDRE)));
                        UDR0 = 0x0d;
                        usart_putc(0x0d);
                        break;

                default:
                   usart_puts(string10);
                        usart_putc(0x0d);

   }
}

//*****************************************************************************************************
//Declaración de la función principal
//*****************************************************************************************************
int main(void)
{
```

```
        init_USART();
        twi_master_init();
        DDRD = _BV(DDD5);
        usart_puts(string5);
        usart_putc(0x0d);
         usart_puts(string6);
        usart_putc(0x0d);
        usart_puts(string7);
        usart_putc(0x0d);
        usart_puts(string8);
        usart_putc(0x0d);

        while(1)
        {
        }

}
```

APÉNDICE B: SOFTWARE DE LA ESTACIÓN BASE

```
'Extracto del código fuente del software de comunicaciones seriales.

Private Sub cmdAbrir_Click()
On Error GoTo manejo_errores

 If MSComm1.PortOpen = False Then
    If comboPuertos.ListIndex = 0 Then
       MSComm1.CommPort = 1
    ElseIf comboPuertos.ListIndex = 1 Then
       MSComm1.CommPort = 2
    ElseIf comboPuertos.ListIndex = 2 Then
       MSComm1.CommPort = 3
    ElseIf comboPuertos.ListIndex = 3 Then
       MSComm1.CommPort = 4
    ElseIf comboPuertos.ListIndex = 4 Then
       MSComm1.CommPort = 11
    End If

  MSComm1.Settings = "9600,N,8,1"
  MSComm1.InputLen = 0
  MSComm1.RThreshold = 1
  MSComm1.PortOpen = True
  Me.Caption = "Adquiriendo datos a través del puerto: " & MSComm1.CommPort

 End If

manejo_errores:
```

```
    If Err.Number <> 0 Then
        MsgBox ("Error al intentar abrir el puerto COM:" + Str(comboPuertos.ListIndex)), vbCritical, "Error
detectado"
        MsgBox ("Puerto No disponible: Seleccione por favor otro puerto "), vbInformation, "Descripción de error"
        Exit Sub
    End If

End Sub

Private Sub cmdCerrar_Click()
If MSComm1.PortOpen = True Then
    MSComm1.PortOpen = False
    Me.Caption = "Desconectado"
End If
End Sub

Private Sub cmdRegresa_Click()
If MsgBox("¿Estas seguro de regresar?", vbQuestion + vbYesNo, "Regresar al inicio") = vbYes Then
Unload Me
MDIfrmHome.Show
End If
End Sub

Private Sub Form_Load()

comboPuertos.AddItem "Com1", 0
comboPuertos.AddItem "Com2", 1
comboPuertos.AddItem "Com3", 2
comboPuertos.AddItem "Com4", 3
comboPuertos.AddItem "Com10", 4


End Sub

Private Sub MSComm1_OnComm()
Dim sdato As Variant, ibyte As Variant

Shape1.Visible = False

    If MSComm1.CommEvent = comEvReceive Then
        Shape1.Visible = True
        sdato = MSComm1.Input
        ibyte = Asc(sdato)
        Text1.Text = sdato
    End If

End Sub
```

bps (bits por segundo): En una transmisión de datos es el número de impulsos elementales (1 ó 0o) transmitidos cada segundo. Los bits por segundo, como unidad del sistema internacional de medidas, son utilizados para expresar la velocidad de datos o *bit rate.*

Bus: Conjunto de conductores común a varios dispositivos que permite distribuir corrientes de alimentación de energía e información.

Hardware: Conjunto de los componentes que integran la parte material de una computadora.

Host: Anfitrión. Es una computadora o dispositivo que funciona como punto de inicio y fin de la transferencia de datos.

Hot plug: Conexión en caliente. Es la capacidad de un periférico para ser conectado o desconectado cuando la computadora está encendida.

Hub: Concentrador. Es un dispositivo que permite centralizar el cableado de una red y poderla ampliar. Esto significa que dicho dispositivo recibe una señal y la repite, emitiéndolo por sus diferentes puertos.

Null modem: Es un método para conectar dos terminales usando un cable serie RS232. En la forma null modem, las líneas de transmisión y recepción están cruzadas. Existe más de una forma de realizar una conexión null modem ya que no hay ningún estándar que defina esta conexión.

PDA: Asistente Personal Digital. Es una computadora de mano; originalmente fue diseñada como agenda electrónica (calendario, lista de contactos, bloc de notas y recordatorio) con un sistema de reconocimiento de escritura.

Plug and play: Es una tecnología que permite a un dispositivo informático ser conectado a una PC sin tener que configurar manualmente hardware o software ni proporcionar parámetros a sus controladores. Para que esto sea posible, el sistema operativo con el que funciona la PC debe tener soporte para dicho dispositivo.

Plug in: Complemento. Es una aplicación que se relaciona con otra para aportarle una Función nueva y generalmente muy específica.

Smartphone: Son pequeños dispositivos que integran funcionalidades de teléfono móvil, con las Funcionalidades más comunes.

Software: Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

Tándem: Conjunto de dos elementos que se complementan.

BIBLIOGRAFÍA

Libros:

1. Duck, Michel, Bishop Richard, Read Richard. *Data communications for engineers*. Editorial Addison-Wesley. Inglaterra, 1996.
2. Gadre, V. Dhananjay. *Programming and customizing the AVR microcontroller*. Editorial Mc Graw Hill. Estados Unidos, 2001.
3. Kernighan, Brian W., Ritchie, Dennis M. *El lenguaje de programación C segunda edición.* Editorial Pearson Prentice Hall. México, 1991.
4. Torres, Manuel, Torres, Miguel Ángel. *Diseño e ingeniería electrónica asistida con Protel DXP.* Editorial Alfaomega-Rama, México 2005.
5. Vázquez Hernández, Javier. *Curso de microcontroladores AVR de ATMEL. Programación en C.* Haltica Automatización, México 2007.
6. Wolf, Stanley, Smith, F.M., Richard. *Guía para mediciones electrónicas y prácticas de laboratorio.* Pearson Educación, 2007.

Libros electrónicos y artículos:

7. Narváez V., Carlos A. *Artículo: Termómetro digital usando un termistor NTC y un PIC16F873.* Universidad de Oriente, Venezuela 2004.
8. *Medida y filtrado de temperatura con NTC: Una introducción al procesado de señales.*
9. *Notas del diplomado: Visual Basic y SQL Server*. Centro de Investigación Cibernética. México 2007.

Manuales:

10. *Hoja de especificaciones del microcontrolador ATMEGA 32*. Atmel Corporation. 2006.
11. *Hoja de especificaciones del microcontrolador ATMEGA128*. Atmel Corporation 2007.
12. *Hoja de especificaciones del circuito integrado LM2575*. Nacional Semiconductors 1996.
13. *Hoja de especificaciones del circuito integrado FT232RL*. Future Technology Devices International Ltd. Estados Unidos, 2005.

14. *Hoja de especificaciones del circuito integrado MAX3222*. MAXIM Integrated Products, 2007.
15. *Hoja de especificaciones del circuito integrado DS1307*. Dallas Semiconductor, 2005.
16. *Hoja de especificaciones del circuito integrado 24LC512*. Microchip Technology, 2004.
17. *Hoja de especificaciones de la sonda 108*-L. Campbell Scientific, 2007.

Páginas de Internet:

18. www.freerTOS.org
19. www.avrfreaks.net
20. www.lawebdelprogramador.com
21. www.atmel.com
22. www.wikipedia.org