UNIVERSIDAD MICHOACANA DE SAN NICOLAS DE HIDALGO

FACULTAD DE INGENIERIA ELECTRICA

TESIS

“Diseño y Construcción de un Módulo GPRS-GSM Transmisor de Información a un Servidor de Internet”

Que para obtener el Título de:

INGENIERO EN ELECTRÓNICA

Presenta:

Gilberto Mejía Romero.

Asesor de Tesis:

Ing. Félix Jiménez Pérez.

Mayo del 2011

DEDICATORIAS.

A mi padre el Dr. Gilberto Mejía Valencia, por ensañarme el valor del trabajo, la honestidad, la constancia, la honradez, por darme el ejemplo para salir adelante aun con adversidades, por enseñarme a siempre ver el lado racional de las cosas, por esto y más te dedico este trabajo y también te doy las gracias, te quiero Papá.

A mi madre la Profra. Gabriela Romero Hinojosa, por inculcarme la religión y sus valores, por mostrarme el lado espiritual de la vida, por enseñarme la bondad, la sencillez, el respeto, la tolerancia… por tus consejos, abrazos, caricias, por darme la vida, por las noche que no dormías por cuidarme de alguna enfermedad o simplemente porque no llegaba a casa, gracias por todo Mamá te quiero.

A mis hermanas Gabriela y Atzimba Mejía Romero, por el apoyo que me han dando y que se que me darán cuando lo necesite, saben que también cuentan con mi apoyo.

A mis abuelos Ma. Carmen Hinojosa Robles† e Ildefonso Romero Coss†, Por todo el cariño y el tiempo que me dieron en mi niñez, por enseñarme a querer y respetar todo lo que crece, por dejarme ayudarte abuelito a cosechar café, por todos esos desayunos tan ricos que solo tu abuelita podías hacer.

A Sandra Gonzales Soto por dedicarme parte de tu tiempo y darme tu cariño durante toda la carrera, por todas las cosas buenas y malas que aprendí a tu lado, por todos los bonitos sentimientos que despertaste en mí, siempre serás alguien muy importante en mi vida, te quiero mucho.

AGRADECIMIENTOS.

A la Universidad Michoacana de San Nicolás de Hidalgo y a la Facultad de Ingeniería Eléctrica, por formarme durante mi carrera, y convertirse en una segunda casa.

Al Ing. Félix Jiménez Pérez por la amistad y la confianza que me tuvo para terminar este proyecto.

Al Dr. José Juan Rincón Pasaye, Dr. Norberto Garcías Barriga, y M.C.I. Dionisio Buenrostro Cervantes, por el tiempo que invirtieron en la revisión de este trabajo.

A los “Gordos” Ing. Joel Abraham Gonzalez Vieyra, Ing. Israel Luna Reyes, Abraham Jaime Origel, por el gran equipo de trabajo que conformamos, por su compañía durante el tiempo que llevo realizar este trabajo y por todas la vivencias juntos.

A mis primos Gilberto, Alberto José y Arturo Amaro Romero, Pedro, Atziri, Nirani Corona Romero, por todas la vivencias que hemos pasado desde niños.

Al Ing. Adán Alberto Hurtado Olivares, por hacer del baile una parte importante en mi vida, por enseñarme a querer el tango y hacerlo un bonito vicio, por los viajes y aventuras en los que hemos sido compañeros.

A Sangre Latina (Montse, Lupita, Vanessa, Liz, Paco, Adán), por permitirme ser parte de este gran grupo, por todos los aplausos alegrías y tristezas que hemos vivido juntos.

A Fercho, Jimmy, Quique, Ruht, Laura, chuche, Piter, Tavo, Orlando, Ñañel, Fernanda, Roquero, Gufi, Emanuel, Quintero, Chaneke, Gus, Rosalia Rigo, Isra, Mayra, Rosy, Ricardo y demás amigos que no terminaría de nombrar.

RESUMEN.

En este trabajo se describe el diseño y construcción de un módulo transmisor de información a un servidor web utilizando el servicio GPRS (*General Packet Radio Service*) que ofrece la red celular GSM (*Groupe Special Mobile*), para el cual se utilizaron 5 dispositivos principales: Modem GPRS, tarjeta de evaluación para el modem GPRS, microcontrolador, memoria FRAM (*RAM* Ferroeléctrica) y display gráfico LCD (*liquid crystal* display).

En la Figura 1 se muestra un diagrama de bloques del sistema desarrollado. El módulo GSM-GPRS recibe los datos a transmitir por medio de un puerto de comunicación serie; esta información llega al microcontrolador y éste la envía a la memoria donde es almacenada de manera temporal. Cuando es necesario transmitir la información al servidor, el microcontrolador extrae la información de la memoria, la envía al modem GPRS y por medio de éste la envía al servidor.

El dsPIC30F4013, es el cerebro del sistema y contiene el software del mismo. Éste es el encargado de la comunicación serie con el cliente y el modem GPRS a través de dos de sus puertos USART (*Universal Synchronous/Asynchronous Receiver-Transmitter*), así como la comunicación SPI (*Serial Peripheral Interface*) con la memoria FRAM y el display gráfico LCD. Además es el responsable de ejecutar el protocolo para la conexión y envío de datos al servidor web.

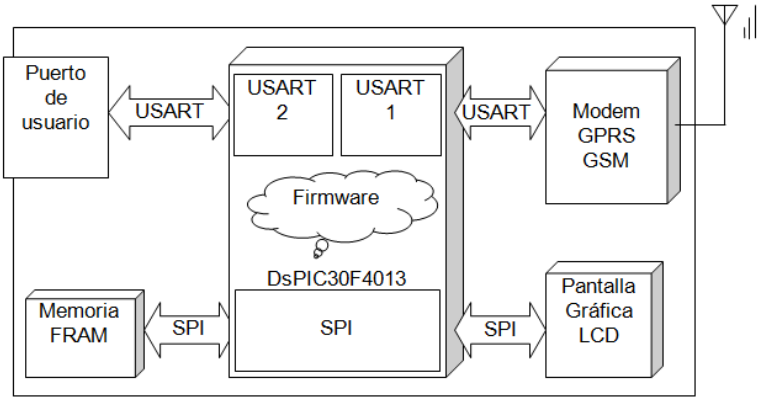


Figura 1: Diagrama a bloques general del módulo GSM-GPRS.

El modem GPRS permite utilizar la red GSM para el envío de los datos al servidor con un bajo costo; tiene facilidad de interconexión con el microcontrolador, y además cuenta con las siguientes características:

- Tiene implementado el protocolo TCP/IP.
- Interface de comunicación serie USART.
- Programación y configuración mediante comandos AT.
- Comunicación P2P mediante servicio GPRS.
- Acceso a página WEB entre otras características.

En la memoria FRAM, se almacena de manera temporal la información que llega del cliente a través del puerto serie (*USART1*). La memoria tiene una capacidad de almacenamiento de información de 32kb, de los cuales 30kb son utilizados para el almacenamiento de la información, y el resto del espacio es utilizado para guardar una tabla ASCII (*American Standard Code for Information Interchange*) utilizada por el display LCD y banderas necesarias para el funcionamiento de la memoria.

El display gráfico LCD, es utilizado como interfaz con el usuario y sirve para informar el estado del sistema, confirmar el correcto almacenamiento de la información, la conexión con el servidor de forma exitosa o fallida, si existe conexión con el cliente, entre otros avisos.

Para el desarrollo del sistema se utilizó una tarjeta de evaluación del modem GPRS, así como el circuito integrado MAX232 para acoplar la conexión de la tarjeta con el microcontrolador. La tarjeta y el MAX232 **no son parte del prototipo final.**

LISTA DE ABREVIACIONES Y SÍMBOLOS.

GPRS	*General Packet Radio Service*
GSM	*Groupe Special Mobile*
FRAM	*RAM* Ferromagnetica
RAM	*random-access memory*
LCD	*liquid crystal display*
USART	*Universal Synchronous/Asynchronous Receiver-Transmitter*
SPI	*Serial Peripheral Interface*
TCP	*Transmission Control Protocol*
AT	*Attention*
P2P	*peer-to-peer*
Kb	Kilobytes
ASCII	*American Standard Code for Information Interchange*
FTP	*File Transfer Protocol*
PHP	*Hypertext Pre-processor*
AM	*Amplitud Modulada*
FM	*Frecuencia modulada*
HF	*High Frequency*
VHF	Very High Frequency
CEPT	*Conférence européenne des administrations des postes et des télécommunications*
DCS	*Digital Combat Simulator*
TSG GERAN	*Technical Specification Group GSM EDGE Radio Access Network*
3GPP	*3rd Generation Partnership Project*
UMTS	*Universal Mobile Telecommunications System*
Km/h	Kilómetros por hora
m/s	Metros por segundo
SIM	*Subscriber Identity Module*
V	Voltaje
mA	Miliamperios
Amp	Amperios

W	Watts
Ω	Ohms
A	Amperios
Led	*Light-Emitting Diode*
CD	Corriente Directa
SMS	*Short Message Service*
PC	*Personal Computer*
ETSI	*European Telecommunications Standards Institute*
WAP	*Wireless Application Protocol*
RTC	*Real Time Clock*
TTL	*Logical Transistor a Transistor*
EEPROM	*Erasable Programmable Read-Only Memory*
° C	Grados centigrados
mSeg	Milisegundos
VDD	Voltaje Corriente Directa
MHz	Mega hertz
CPOL	*Clock Polarity*
CPHA	*Clock Phase*
MSB	*Most Significant Byte*
LSB	*Least Significant Bit*
MSM	Mensajes Miltimedia
MS	Estación Móvil
BSS	Estación Base
NSS	*Network and Switching Subsystem*
MSC	*Mobile Services Switching Center*
BTS	*Transceiver Station*
BSC	*Station Controller*
GMSC	*Gateway Mobile Services Switching Center*
HLR	*Home Location Registrer*
IP	*Internet Protocol*
VLR	*Visitor Location Registrer*

AuC	*Visitor Location Registrer*
EIR	*Equipment Identy Registrer*
IMEI	*International Mobile Equipment Identy*
GIWU	*GSM Interworking Unit*
Kbps	Kilo bites por segundo
seg	Segundo
CSD	conmutación de circuitos
UMTS	*Universal Mobile Telecommunications System*
SPC	S*ecure Copy*
SFTP	*Secure File Transfer Protoco*l
SSH	*Secure SHel*
RFC	*Request for Comments*
DTP	Proceso de Transferencia de Datos
Kbit/s	Kilo bites por segundo

ÍNDICE DE FIGURAS.

ÍNDICE DE TABLAS.

ÍNDICE

1 INTRODUCCIÓN.

El presente trabajo aborda el tema de la transmisión de datos digitales a través de la red de telefonía celular GSM, el cual es un medio de comunicación inalámbrico de gran cobertura.

Se describe la construcción de un sistema que almacena información y la envía a un servidor remoto de forma inalámbrica. El enlace inalámbrico utiliza la tecnología celular GSM mediante el protocolo GPRS para la transmisión de los datos.

La importancia de la transmisión de datos digitales a distancia, radica en la necesidad de recabar información y transmitirla a una estación base con posibilidad de contar con información en tiempo real, automatizar sin presencia del elemento humano. Esto reduce significativamente los costos de proyectos, en los cuales las tareas suelen ser rutinarias, y donde los lugares o zonas en los que se realizan las pruebas pueden ser de difícil acceso o localizarse a una gran distancia.

1.1 METODOLOGIA.

El módulo GPRS-GSM surge de la necesidad de crear un sistema de envío de información inalámbrico para el proyecto **"Sistema para la Evaluación del Potencial Eólico en el Estado de Michoacán"**. En este proyecto, se planteó colocar sensores de velocidad y dirección del viento en diferentes puntos del Estado, este trabajo se enfoca en un sistema capaz de enviar la información recabada de cada uno de los sensores a un servidor de Internet. En la primera parte del trabajo se investigó si existía un equipo con las características para realizar esta tarea, pero no se encontró. Entonces, se vio en la necesidad de realizar el diseño y construcción de un equipo con las características antes mencionadas.

Para el diseño de este módulo se consultaron páginas web relacionadas con la red GSM, el servicio GPRS, el protocolo y el servicio FTP (*File Transfer Protocol*). Para la

construcción del módulo se revisaron las hojas de datos de diferentes componentes como: El microcontrolador, la memoria FRAM, la tarjeta de evaluación, el modem GPRS y el MAX232.

La primera etapa del diseño del módulo se construyó en una tarjeta de prueba, se realizaron pruebas de funcionamiento de los componentes; las pruebas fueron realizadas por etapas, primero la comunicación entre el microcontrolador y la memoria, seguido de la comunicación entre el microcontrolador y el cliente, después la comunicación entre el microcontrolador y el modem GPRS y por último la comunicación entre el modem GPRS y el servidor de Internet. Ya comprobado el correcto funcionamiento de estas etapas se integraron para realizar la prueba de funcionamiento del sistema, una vez comprobado el correcto funcionamiento de éste, se construyó el módulo GSM-GPRS en un circuito impreso. Para la construcción de éste, primero se realizó el diseño en la computadora, y posteriormente se construyó el diseño final.

Los programas utilizados para realizar el diseño del módulo son:

- Para el software implementado en el microcontrolador se utilizó MPLAB IDE y el compilador C30.
- Para la depuración del módulo se requirió un analizador del puerto serie SERIAL SNIFFER.
- Para corroborar la comunicación entre el modem y el microcontrolador, se utilizó el programa PORTMON.
- Para el diseñó del circuito impreso se utilizó el software EAGLE de la empresa CADSOFT.
- Para realizar la página web que muestra la información enviada al servidor se utilizó el intérprete PHP (*Hypertext Pre-processor*).
- Se utilizó el servidor web Apache por ser uno de los más completos, el que mejor se integra con el intérprete PHP y el sistema operativo Linux.
- Se utilizó un servidor FTP gratuito para almacenar los datos en el servidor.

Finalmente se tiene el módulo GSM-GPRS armado y funcionando.

1.2 ANTECEDENTES DE COMUNICACIONES.

Con el desarrollo de la civilización y de las lenguas escritas surgió también la necesidad de comunicarse a distancia de forma regular. A raíz de esto surgieron los sistemas postales, los cuales fueron evolucionando con la aparición del ferrocarril, los vehículos de motor, los aviones y otros medios de transporte.

Con el descubrimiento de la electricidad en el siglo XVIII, se comenzó a buscar la forma de utilizar las señales eléctricas en la transmisión rápida de mensajes a distancia. A pesar de que la telegrafía supuso un gran avance en la comunicación a distancia, los primeros sistemas telegráficos sólo permitían enviar mensajes letra a letra. Por esta razón se seguía buscando algún medio de comunicación eléctrica. Los primeros sistemas telegráficos y telefónicos utilizaban el cable como soporte físico para la transmisión de los mensajes, pero las investigaciones científicas indicaban que podían existir otras posibilidades. En el año de 1906 tuvo lugar la primera emisión de información vía inalámbrica (utilizando ondas de radio).

Los primeros sistemas de telefonía móvil civil empiezan a desarrollarse a partir de finales de los años 40 en los Estados Unidos. Eran sistemas de radio analógicos que utilizaban en el primer momento modulación en amplitud (AM) y posteriormente modulación en frecuencia (FM). Se popularizó el uso de sistemas FM gracias a su superior calidad de audio y resistencia a las interferencias. El servicio se daba en las bandas de HF (*High Frequency*) y VHF (Very High Frequency). Los primeros equipos eran enormes y pesados, por lo que estaban destinados casi exclusivamente a usarse a bordo de vehículos. Generalmente se instalaba el equipo de radio en el maletero y se pasaba un cable con el teléfono hasta el salpicadero del coche. Una de las compañías pioneras que se dedicaron a la explotación de este servicio fue la americana *Bell*. Su servicio móvil fue llamado *Bell System Service*. No era un servicio popular porque era extremadamente caro, pero estuvo operando (con actualizaciones tecnológicas, por supuesto) desde 1946 hasta 1985.

El estándar GSM fue desarrollado a partir de 1982. En la conferencia de telecomunicaciones CEPT (*Conférence européenne des administrations des postes et des télécommunications*) de ese año fue creado el grupo de trabajo GSM, cuya tarea era desarrollar un estándar europeo de telefonía móvil digital. Se buscó evitar los problemas de las redes analógicas de telefonía móvil, que habían sido introducidos en Europa a fines de los años 1950, y no fueron del todo compatibles entre sí a pesar de usar, en parte, los mismos estándares. En el grupo GSM participaron 26 compañías europeas de telecomunicaciones. En 1990 se finalizaron las especificaciones para el primer estándar GSM-900, al que siguió DCS-1800 (*Digital Combat Simulator*) un año más tarde. En 1991 fueron presentados los primeros equipos de telefonía GSM como prototipos. De manera paralela, se cambió el nombre del grupo a *Standard Mobile Group* y a partir de este momento se usaron para el propio estándar. En 1992 las primeras redes europeas de GSM-900 iniciaron su actividad, y el mismo año fueron introducidos al mercado los primeros teléfonos celulares GSM, siendo el primero el Nokia 1011 en noviembre de este año. En los años siguientes, el GSM compitió con otros estándares digitales, pero se terminó imponiendo también en América Latina y Asia. En 2000, el grupo de trabajo para la estandarización del GSM se pasó al grupo TSG GERAN (*Technical Specification Group GSM EDGE Radio Access Network*) del programa de cooperación 3GPP (*3rd Generation Partnership Project*), creado para desarrollar la tercera generación de telefonía móvil (3G). El sucesor del GSM, UMTS (*Universal Mobile Telecommunications System*), fue introducido en 2001, sin embargo, su aceptación fue lenta, por lo que gran parte de los usuarios de telefonía móvil en 2010 siguen utilizando GSM[1].

Uno de los avances más espectaculares dentro de la comunicación de datos se ha producido en el campo de la tecnología de las computadoras. Desde la aparición de las computadoras digitales en la década de 1940, éstas se han introducido en los países desarrollados en prácticamente todas las áreas de la sociedad (industrias, negocios, hospitales, escuelas, transportes, hogares y comercios). Mediante la utilización de las redes informáticas y los dispositivos auxiliares, el usuario de un computador puede transmitir datos con gran rapidez. Estos sistemas pueden acceder a multitud de bases de datos. A

[1] http://www.monografias.com/trabajos15/telefonia-celular/telefonia-celular.shtml

través de la línea telefónica se puede acceder a toda esta información y visualizarla en pantallas.

1.3 JUSTIFICACIÓN.

La instrumentación a distancia ha cobrado en los últimos años gran importancia al permitir controlar y monitorear procesos industriales sin la presencia humana. Esto ha generado una reducción en los costos de mantenimiento y operación en los equipos, al poder monitorear de manera continua, aun cuando se localicen en lugares apartados. También se han optimizado los recursos humanos que poseen las empresas, al reducir el número de tareas en las cuales el elemento humano está subutilizado por ser tareas rutinarias.

Actualmente existe una gran cobertura celular en todo el país, esto sumado al costo accesible de transmisión de información, permite considerar el uso de esta infraestructura para la transmisión de información. El uso de la red celular permite hacer más eficientes los equipos de instrumentación a distancia, lo que da la pauta para que éstos puedan ser móviles y autónomos con el uso de baterías.

Este sistema forma parte integral del proyecto **"Sistema para la Evaluación del Potencial Eólico en el Estado de Michoacán", el cual es un proyecto aprobado por la coordinación de investigación científica de la Universidad Michoacana de San Nicolás de Hidalgo.**

1.3.1 APLICACIÓN AL DIAGNÓSTICO DE POTENCIAL EÓLICO**.**

La generación de energía eléctrica ha sido un tema de mucha importancia para la sociedad mundial. Siempre se han buscado formas de generarla por medios no contaminantes; un ejemplo es la generación por medio eólico. Esta fuente de energía ha tomado gran importancia en los últimos años, debido a que es una forma de generación

eléctrica no contaminante, barata y que tiene un impacto ambiental mínimo, en relación con otras formas de generación.

Un punto crucial en el diseño de las plantas de energía eólica, es la ubicación de la planta, ya que se requiere un lugar en donde el viento posea la suficiente velocidad para mover las hélices de los generadores eléctricos.

Para poder aprovechar la energía eólica es importante conocer las variaciones diurnas, nocturnas y estaciónales de los vientos, la variación de la velocidad del viento con la altura sobre el suelo, la cantidad de las ráfagas en espacios de tiempo breves, y valores máximos ocurridos en series históricas de datos con una duración mínima de 20 años. Para poder utilizar la energía del viento, es necesario que éste alcance una velocidad mínima que depende del aerogenerador que se vaya a utilizar, pero que suele empezar entre los 3 m/s (10 km/h) y los 4 m/s (14.4 km/h), velocidad llamada "*cut-in speed*", y que no supere los 25 m/s (90 km/h), velocidad llamada "*cut-out speed*".[2]

Todo lo anterior requiere de un equipo que sea capaz de capturar datos de la velocidad del viento de manera regular, dentro de la zona elegida como posible candidata para instalar una planta eólica. Dichas zonas tienden a ser áreas ubicadas fuera de las zonas urbanas y en planicies altas para captar la mayor cantidad de viento. Esto origina que la toma de mediciones sea una tarea muy compleja y costosa, ya que el traslado a dichas áreas suele ser complicado y de difícil acceso, debido a que muchas veces no existen caminos.

Con la ayuda de estaciones meteorológicas electrónicas, es posible realizar mediciones durante un cierto periodo de tiempo y almacenar los datos en alguna memoria. Con esto reducimos las veces que se tienen que visitar las zonas donde se encuentran las estaciones meteorológicas, pero aún así suele ser una tarea que podríamos eliminar, además de poder tomar mediciones más frecuentes y obtener datos más completos e incluso en tiempo real.

[2] http://es.wikipedia.org/wiki/Energía_eólica.

1.4 OBJETIVO.

Construir un módulo de comunicación inalámbrica para la transmisión de datos desde lugares remotos. El módulo debe transmitir la información recabada a una estación base o servidor WEB de acuerdo a la configuración que el usuario programe en dicho módulo. La Figura 2 muestra el diagrama general del sistema de envío de información.

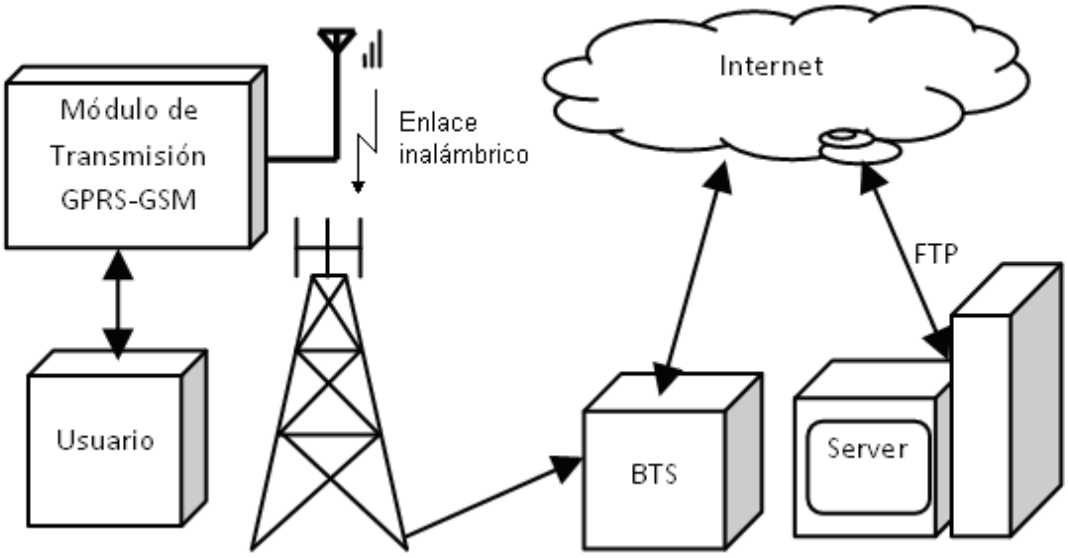


Figura 2: Diagrama general del sistema de envío de información.

1.5 DESCRIPCIÓN DE LOS CAPÍTULOS.

En el capítulo 1 se da una breve introducción a este trabajo, se muestra la problemática que se tiene con la adquisición de datos en una red de potencial eólico, y se propone una forma de solucionarla.

En el capítulo 2 se da una descripción breve de los componentes principales a ser utilizados para la interconexión de la red de potencial eólico y la conexión con la estación central.

En el capítulo 3 se describe el modem GSM a utilizar, se precisan características del mismo, así como las configuraciones necesarias para su utilización como medio de enlace en una red de módulos, mediante el protocolo GPRS.

En el capítulo 4 se presenta la construcción del módulo, la interconexión con el módulo de adquisición de datos y aspectos particulares de hardware; también se detallan aspectos como las configuraciones de software realizadas para la transmisión de los datos de forma remota.

En el capítulo 5 se presenta el software implementado en el módulo.

En el capítulo 6 se presentan las pruebas realizadas con los módulos, pruebas de alcance y calidad de la conexión.

En el capítulo 7 se presentan las conclusiones y trabajos futuros.

2 DESCRIPCIÓN DE LOS COMPONENTES.

2.1 PROBLEMÁTICA.

Para realizar el monitoreo a distancia, requerimos de alguna tecnología inalámbrica que permita la transferencia de la información de interés. Dos de las características que buscamos para tal propósito son: el bajo costo por volumen de datos transmitidos y la existencia de una infraestructura previa, para no invertir en una propia.

2.2 EL MÓDULO GPRS.

Se decidió utilizar la tecnología celular GSM ya que existe una gran cobertura en todo el país. Se utiliza el servicio GPRS para enviar la información pues el costo de envío es accesible, 1 peso por cada megabyte de información enviada. En la Figura 3 se presenta el módulo a utilizar, el cual es de la empresa Winity cuyo modelo es: WGM100.

Figura 3: Modem GSM Winity modelo WGM100.

Se decidió utilizar este módulo, debido a que presenta muchas características que buscamos para el proyecto, como son:[3]

[3] AG Electronica S.A de C.V, Tarjeta de Evaluación Módulo GSM-GPRS PLUTO, 2009.

- Bajo costo.
- Facilidad de interconexión con microcontroladores.
- Protocolo TCP/IP integrado.
- Múltiples conexiones TCP/UDP (hasta12 simultáneas).
- Transmisión de datos por GPRS: CS-1, CS-2, CS-3, CS-4
- Interface de comunicación serie USART.
- Programación y configuración mediante comandos AT.

Con estas características, el módulo nos permite realizar varios tipos de conexiones, entre las que destacan:

- Comunicación P2P mediante servicio GPRS.
- Acceso a página WEB.
- Posibilidad de comunicación basada en mensajes de texto (SMS).

Con el fin de agilizar las pruebas de este módulo y determinar la viabilidad del mismo, se hizo uso de una tarjeta de evaluación fabricada por la empresa AG ELECTRONICA la cual se muestra en la Figura 3; La tarjeta de evaluación es denominada KIT PLUTO, y presenta las siguientes características:

- Acceso a todos los periféricos del modem GSM mediante *headers*.
- Fuente de alimentación propia.
- Base para SIM (*Subscriber Identity Module*).
- Antena integrada.
- Convertidor TTL-RS232 para la comunicación con el modem.

En la Figura 4 se muestra el módulo con el modem GSM integrado.

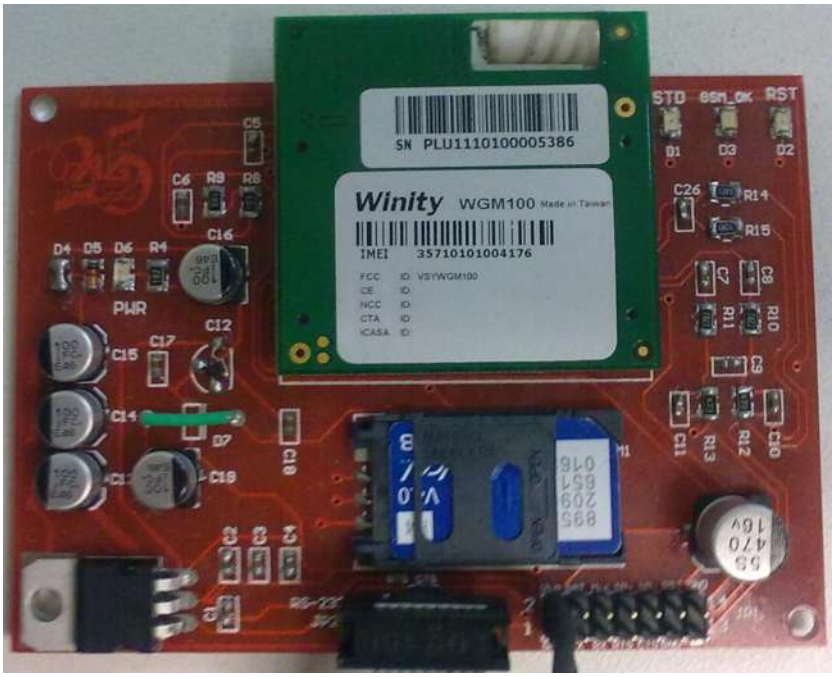


Figura 4: Tarjeta de evaluación para el modem GSM.

El estándar GPRS especifica 4 esquemas de codificación, llamados CS-1, CS-2, CS-3 y CS-4. Cada uno define el nivel de protección de los paquetes contra interferencias para evitar degradar la señal según la distancia entre las terminales móviles y las estaciones base. Cuanto mayor sea la protección, menor será el rendimiento. La Tabla 1 muestra el rendimiento y la protección de cada uno de los 4 esquemas. En el módulo GPRS-GSM, se utilizó el esquema de codificación 4, ya que es necesario que la velocidad de envío de datos al servidor, sea mayor que la velocidad de envío de datos del cliente al módulo.[4]

Tabla 1: Esquema de codificación GPRS.

Esquema de codificación	Rendimiento	Protección
CS-1	9.05 Kbit/s	Normal (señalización)
CS-2	13.4 Kbit/s	Ligeramente menor
CS-3	15.6 Kbit/s	Reducida
CS-4	21.4 Kbit/s	Sin protección

[4] http://es.kioskea.net/contents/telephonie-mobile/gprs.php3

2.3 TARJETA DE EVALUACIÓN PARA EL MODEM GSM.

La tarjeta de evaluación para el módulo PLUTO PCBA-GSM está diseñada para integrar de forma inmediata una comunicación telefónica GSM/GPRS, de bajo costo. Las características de la tarjeta de evaluación son:[5]

- Fuente de alimentación regulada: 5V a 1.5A y 2.8V a 80mA integrada en la tarjeta, para una entrada de CD de 6V a 12V a 1.5 Amp (mínimo).
- Comunicación serial, niveles 2.8V y RS-232 (Rx, Tx, CTS y RTS).
- Salida para altavoz de 16 Ω, 1 W.
- Entrada para micrófono de tipo electret.
- SIM holder.
- Leds (*Light-Emitting Diode*) monitores: Alimentación (PWR). Estado (STD) y Conexión a red (GSM_ok)
- Led monitor de GPIO (RST).
- Pin de RESET.
- Entrada para batería de reloj de tiempo real (calendario).
- Pin de PowerOn no disponible.
- Pines de teclado no disponible.
- No contempla protección ESD.

2.3.1 FUENTE DE ALIMENTACIÓN.

La alimentación deberá aplicarse a través del conector JP1, el cual se muestra en la Figura 5. Este conector se encuentra en la parte inferior derecha de la tarjeta de prueba. La alimentación es a través de los pines 1 y 2, empleando una fuente de alimentación de CD (corriente directa) con un rango de voltaje de 6V a 12V y corriente de 1.5 Amp (mínimo).

[5] AG Electronica S.A de C.V, Tarjeta de Evaluación Módulo GSM-GPRS PLUTO, 2009.

Al energizar la tarjeta deberá encender el LED indicador de “energizado”. Se recomienda no montar o desmontar el módulo mientras esté energizada la tarjeta, ya que puede dañarse.

Figura 5: Conector JP1.

2.3.2 COMUNICACIÓN SERIAL.

El conector JP1 de la tarjeta PLUTO cuenta con pines de comunicación serial. El nivel de operación es 2.8V, por lo que a la salida de TX y RTS se tendrá como máximo este voltaje en nivel alto, RX y CTS son entradas tolerantes a 3.0V.

El conector JP2 de la tarjeta se muestra en la Figura 5, éste se encuentra en la parte inferior de nuestra tarjeta PLUTO y es una salida de comunicación serial a niveles RS-232. Para comunicación con una PC (*Personal Computer*), se deben obtener las salidas en las terminales de comunicación serial del módulo PLUTO. En caso de comunicación con un microcontrolador, se deben cortocircuitar los puentes cercanos a JP1. La comunicación serial es de tipo DCE (conector DB9 hembra Data *Communication Equipment*). Dependiendo del tipo de conexión se debe seleccionar adecuadamente el cable de ésta. JP2 está optimizada para ser conectada a un cable serial con adaptador DB9 a cable plano y conector para cable plano de 10 terminales.

Figura 6: Conector JP2.

2.3.3 TARJETA SIM.

SIM1 es un conector estándar para SIM-card de 8 terminales. Para operar correctamente y conectarse a la red GSM debe insertarse una SIM válida para la red de su preferencia. Las terminales de detección de SIM insertada no están conectadas hacia el módulo PLUTO, pero la detección se hace por medio de software. En la Tabla 2 se nombran los pines de la SIM. **La tarjeta SIM forma parte del prototipo final.**

Tabla 2: Pines de la SIM.

Pin No.	Nombre
CS1	VSIM 1.8V / 3V
CS2	Reset
CS3	Reloj
CS4,CS8	Detección de SIM insertada
CS5	Tierra
CS6	No conectar
CS7	Entrada/salida

En la Figura 7. Se muestra la base para SIM de 8 terminales.

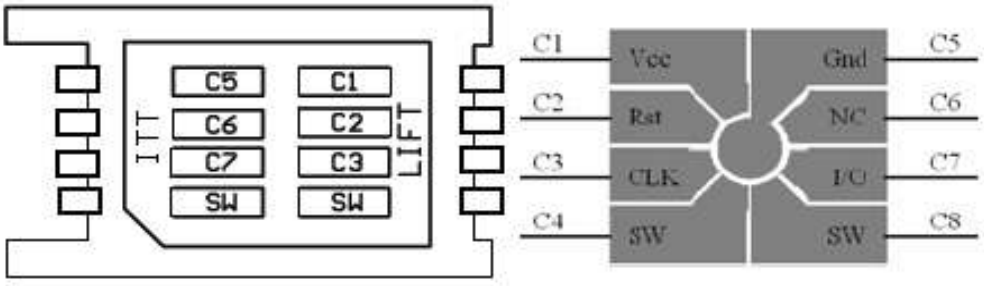


Figura 7: Base para SIM.

2.3.4 CONFIGURACIÓN DE LA COMUNICACIÓN SERIAL.

De fábrica, la comunicación serial del módulo PLUTO está configurada con los parámetros 115200, 8, N, 1, sin control de flujo. El modo de "llamada" al momento de encender el módulo se encuentra en modo "datos"; cabe resaltar que el modo CDS no es funcional en esta versión de firmware. Para la comunicación con una PC es posible emplear cualquier software de comunicación serial como hyperterminal, XCTU, Teraterm, etc.

2.4 COMANDOS AT.

Los comandos AT controlan las diversas funciones y configuraciones que posee el modem GSM. Estos comandos son parte de un estándar soportado por la organización ETSI (*European Telecommunications Standards Institute*) en la cual se establecen los diferentes protocolos para la comunicación en las redes GSM, así como los comandos que deben utilizar los modems y/o celulares que quieran utilizar la red GSM.

De acuerdo a las funciones que poseen los equipos móviles, existen diferentes versiones de comandos para controlar. Dichas versiones vienen dadas por las revisiones que hace ETSI para cada nueva función. Entre las diferentes revisiones destacan las siguientes:

- Comandos para mensajes SMS.
- Comandos para comunicación GPRS.
- Comandos básicos para configuración de ingreso en red GSM.
- Comandos para comunicación WAP (*Wireless Application Protocol*).

Existen también comandos propietarios de las empresas que fabrican los modems, a estos comandos se les denominan "comandos extendidos", los cuales varían dependiendo del fabricante y de las capacidades de los modems. Dichos comandos pueden controlar periféricos adicionales como convertidores analógico-digitales, RTC (Reloj de Tiempo Real), entre otros; pero también estos comandos pueden realizar funciones para simplificar

configuraciones diversas, como WAP o GPRS, que son un tanto complejas. Estos comandos simplifican en gran parte las configuraciones y hacen más fácil la interacción con la red GSM.

2.4.1 COMANDOS DEL MODEM GPRS.

Con los comandos AT, el modem GPRS puede realizar diversas tareas como son:

- Comprobar la comunicación serial.
- Ver el número telefónico de la SIM.
- Identificar el módulo.
- Realizar conexión a la red GSM y ver la calidad de la señal.
- Configuración del puerto RS-232.
- Configuración GSM.
- Enviar y recibir mensajes de texto.

En la Figura 8 se presenta una prueba realizada al modem con los comandos AT, mediante una sesión realizada con la hiperterminal, todos los "OK" son mensajes del confirmación del sistema.

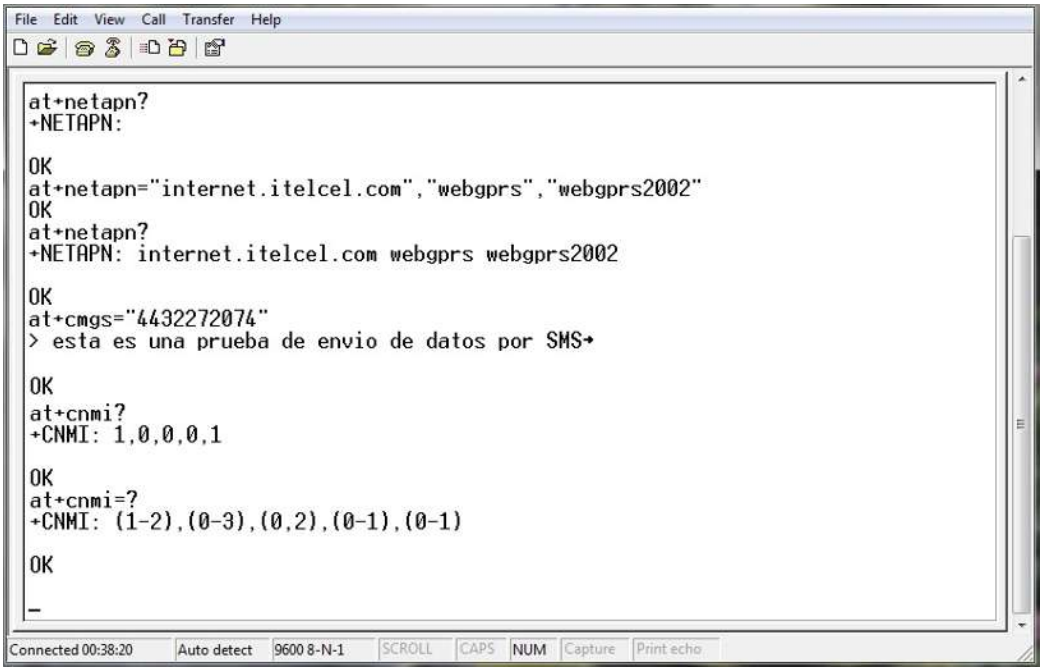


Figura 8: Sesión realizada con el modem GSM.

A continuación se describen los comandos utilizados en dicha sesión:

- At+netapn: Indica al modem los parámetros para conectarse a través del servicio GPRS a la red GSM (Dirección, Usuario, Contraseña).

- At+cmgs: Envía un mensaje de texto al número marcado entre comillas, para este ejemplo el mensaje enviado es "*esta es una lista de envio de datos por SMS*".

- At+cnmi: Configura al modem para recibir mensajes SMS y avisa en caso del arribo de un nuevo mensaje mediante un indicador.

Las ultimas 7 líneas de la Figura 8 son respuestas del sistema.

2.4.2 COMANDOS UTILIZADOS PARA LA INTERFAZ CON EL CLIENTE.

Se usaron diversos comandos, los cuales sirven al cliente para realizar diferentes funciones como:

- Comprobar conexión con el módulo.
- Guardar nombre del archivo.
- Guardar datos en el módulo.
- Borrar memoria FRAM.
- Enviar datos.

En la Tabla 3 se muestran los comandos del cliente.

Tabla 3: Comandos propios del sistema para el cliente.

Comando	*Formato*	*Descripción*	*Respuesta*	*Fallo*	*Comando ejecutado.*
ATT	ATT (enter)	Comprueba comunicación con el cliente.	OK	ERROR (comando escrito incorrectamente)	
ATN	ATN nombre_del_archivo.txt (enter)	Almacena los datos con el nombre_del_archivo.txt	OK	ERROR (nombre muy largo de más de 60 caracteres)	
ATG	ATG XX(enter)	Almacena el número de bytes XX en el módulo, donde XX es un número entre 00 y 90.	(espera datos)	ERROR(número de bytes mayor de 90)	OK
ATE	ATE(enter)	Envía los datos al servidor de Internet.	Sending…	ATE ERROR (no se transmitieron los datos)	ATE OK
EFRAM	EFRAM(enter)	Borra los datos existentes en el módulo.	OK		

En la Tabla 4 se muestra un ejemplo del uso de los comandos del cliente.

Tabla 4: Ejemplo del uso de los comandos del cliente

Cliente envía	Módem GSM-GPRS responde.	Comentarios
ATT (enter)	**OK**	*Prueba de conexión con el módulo.*
ATN archivo.txt (enter)	**OK**	*Graba nombre de archivo.*
ATN (enter)	**ERROR**	*En caso de un nombre incorrecto.*
ATG 10 (enter)	**(espera datos)**	*Guarda datos en el módulo.*
	0123456789	*Datos a enviar.*
	OK	*El número después del comando indica el número de datos que se envían.*
ATG 1(enter)	**ERROR**	*Sí el formato del comando es incorrecto.*
EFRAM(enter)	**OK**	*Borra memoria FRAM*
ATE(enter)	**Sending...**	*Envía los datos.*

2.5 EL MICROCONTROLADOR DSPIC30F4013.

La parte fundamental del módulo de transmisión de información es un microcontrolador de la empresa MICROCHIP, el cual posee las siguientes características[6]:

- Arquitectura HARDVARD modificada.
- 48Kb de memoria FLASH de programa.
- Hasta 30 MIPS.
- Detección programable de bajo voltaje.
- 2kb de memoria de datos.

Además presenta varios periféricos como son:

- Puertos serie USART a nivel TTL.
- Puerto de comunicación SPI.
- 2 puertos de entrada/salida digital de 8 bits.

En la Figura9, se presenta el diagrama de patitas del microcontrolador:

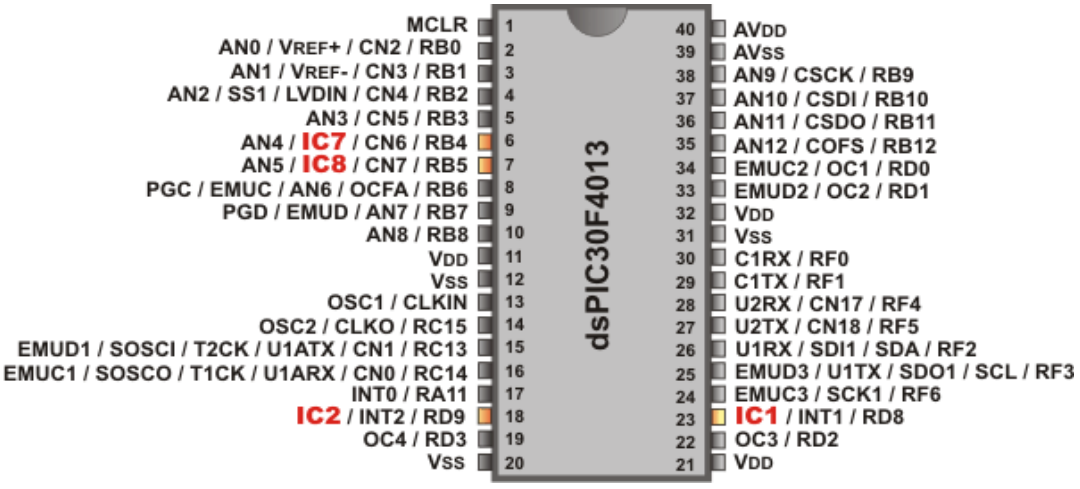


Figura 9: Diagrama del dsPIC30F4013.

[6] Microchip Technology Inc., dsPIC30F3014, dsPIC30F4013 Data Sheet, 2004.

Dadas las características anteriormente mencionadas, se optó por utilizar este dispositivo como elemento central de nuestro sistema, el cual realiza las siguientes tareas:

- Recepción de los datos para almacenar en memoria FRAM.
- Configuración del modem GPRS.
- Envío de los datos almacenados en FRAM al servidor FTP.
- Despliegue de información en pantalla gráfica.

2.6 LA MEMORIA FRAM.

Dado que el microcontrolador posee una cantidad de memoria limitada para el almacenamiento de datos, se optó por usar una memoria externa; esta memoria debe tener algunas características específicas debido a la aplicación:

- Capacidad de memoria al menos 32kb.
- Interface de comunicación SPI.
- Ciclo de borrado y grabado de información ilimitado.
- Retención de la información en caso de falla de la energía eléctrica.

Dadas estas características, se decidió utilizar una memoria del tipo FRAM (RAM ferromagnética), modelo FM25L256B del fabricante RAMTRON. La FM25L256B es una memoria de acceso aleatorio ferromagnética o FRAM de 256 kilobits no volátil, basada en un avanzado proceso ferroeléctrico. Realiza la lectura y escritura como una memoria RAM. Tiene una retención de datos hasta de 45 años, mientras que elimina los problemas de programación de las memorias, EEPROM (*Erasable Programmable Read-Only Memory*) y otras memorias no volátiles.

A diferencia de las memorias EEPROM, la memoria FRAM realiza la operación de escritura a la velocidad del bus. Los datos son escritos en la memoria inmediatamente

después de que cada byte se ha transferido al dispositivo. El tipo de memoria FRAM también presenta un consumo mucho menor de energía que la EEPROM.

Estas características hacen a la memoria FM25L256B ideal para aplicaciones donde se necesita el uso de memorias no volátil y donde se requieran frecuentes y rápidas escrituras con bajo consumo de potencia. Otra aplicación es en controles industriales que requieren un ciclo corto de escritura, donde el uso de una memoria EEPROM puede tener pérdidas.

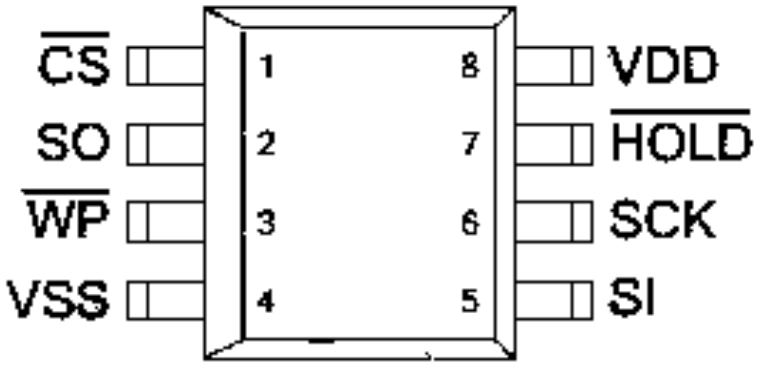


Figura 10: Diagrama de la FRAM.

La memoria FRAM opera a la velocidad del bus SPI. La memoria FRAM está diseñada para trabajar en un rango de temperatura de -40 ° C a +85 ° C. La memoria FRAM no es accesible por un período de tiempo (10 mSeg) después de conectarla a la energía. El usuario debe cumplir este tiempo mínimo después de conectar la memoria FRAM a VDD.

La configuración de los pines se muestra en la Figura 10 mientras que los pines se nombran en la Tabla 5.[7]

[7] Ramtron Inc., FM25L256B Data Sheet, 2009.

Tabla 5: Pines de la FRAM.

Pin	Función
/CS	Chip Select
/WP	Write Protect
/HOLD	Hold
SCK	Serial Clock
SI	Serial Data Input
SO	Serial Data Output
VDD	Supply Voltage (2.7 a 3.6V)
VSS	Ground

2.6.1 DESCRIPCIÓN DE LOS PINES.

- ***/CS:*** *Esta entrada es activa en bajo y tiene la función de activar el dispositivo. Cuando éste está en alto, el dispositivo entra en modo de espera consumiendo poca potencia e ignora otras entradas, y todas las salidas entran en tercer estado; cuando la entrada está en bajo, el dispositivo internamente activa la señal SCK, para enviar un op-code.*

- ***SCK:*** *Todas las I/O son sincronizadas con el reloj serie, las entradas funcionan en el flanco de subida y las salidas en el flanco de bajada, Puesto que el dispositivo es estático, la frecuencia de reloj puede ser cualquiera entre 0 y 20 MHz y puede ser interrumpida en cualquier momento.*

- ***/HOLD:*** *El pin /HOLD se utiliza cuando el dispositivo de control interrumpe una operación de memoria para otra tarea. Para esto, el pin /HOLD debe tener una señal en bajo. Cuando esto sucede, la operación actual se suspende. El dispositivo hace caso omiso de cualquier transición en SCK o /CS.*

- ***/WP:*** *Este pin activo en bajo, evita operaciones de escritura en el registro Status.*

- ***SI:*** *Todos los datos que entran a este pin son los datos almacenados en el dispositivo; el pin es muestreado solamente en el flanco de bajada del SCK. En cualquier otro momento los datos que entran en el pin son ignorados por el dispositivo. Siempre se debe trabajar a un nivel lógico válido para satisfacer las especificaciones.*

- ***SO:*** *Este es el pin de salida de datos, es utilizado durante las lecturas a la memoria y se mantiene en tercer estado en cualquier otro momento. Incluso cuando el pin /HOLD está en bajo, los datos serán leídos en el flanco de bajada del reloj serie.*

- ***VDD:*** *Fuente de alimentación, soporta un voltaje entre 2.7 a 3.6 v*

- ***VSS:*** *Tierra.*

En la Figura 10 se observa el diagrama de bloques de la estructura interna de la FRAM.

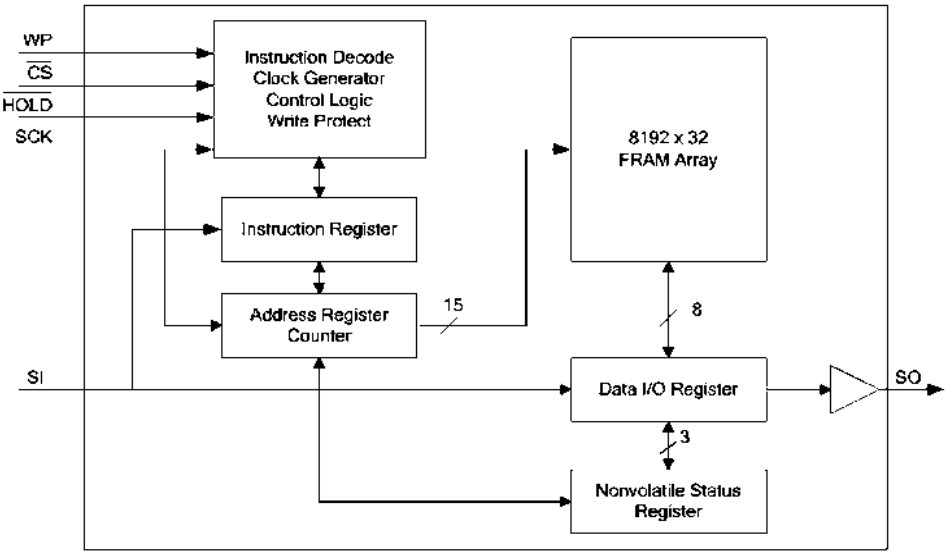


Figura 11: Diagrama de bloques interno de la FRAM.

El arreglo de la memoria se encuentra organizado como 32.768 x 8 y se accede mediante un estándar industrial del tipo SPI. La operación funcional de la FRAM es similar a la serie EEPROM. La diferencia importante entre la FM25L256B y la serie EEPROM es que la FRAM tiene un rendimiento de escritura superior y un menor consumo de energía.

2.6.2 ARQUITECTURA.

En la memoria FRAM, el usuario accede a más de 32000 localidades de memoria de 8 bits de datos cada uno. Estos bits de datos son ordenados en forma serial. El usuario accede a las direcciones mediante el protocolo SPI, que incluye un *Chip Select* (selección de chip, para permitir múltiples dispositivos en el bus), un código de operación, (op-code), y dos bytes de dirección. El bit más significativo del rango de direcciones es un valor "*don't care*" (no importa). La dirección completa de 15 bits especifica cada dirección de byte único.

La mayoría de las funciones de la FRAM son controladas por la interfaz SPI o son manipulados automáticamente por los circuitos que interactúan con ella. El tiempo de acceso para el funcionamiento de la memoria es casi cero, mucho menor, que el tiempo necesario para el protocolo. Es decir, la memoria se lee o escribe a la velocidad del bus SPI.

2.6.3 INTERFAZ SERIAL SPI.

La FRAM emplea el protocolo SPI, éste puede funcionar a velocidades de hasta 20 MHz. Este bus serie de alta velocidad proporciona un alto rendimiento de comunicación serial conectado a un microcontrolador. Muchos microcontroladores comunes tienen puertos SPI por hardware que permiten una conexión directa.

La interfaz SPI utiliza un total de cuatro pines: *SCK, data-in, data-out, chip Select*. Un sistema típico de configuración utiliza uno o más dispositivos FM25L256B con un microcontrolador que tiene un puerto dedicado SPI, como se ilustra en la Figura 11.

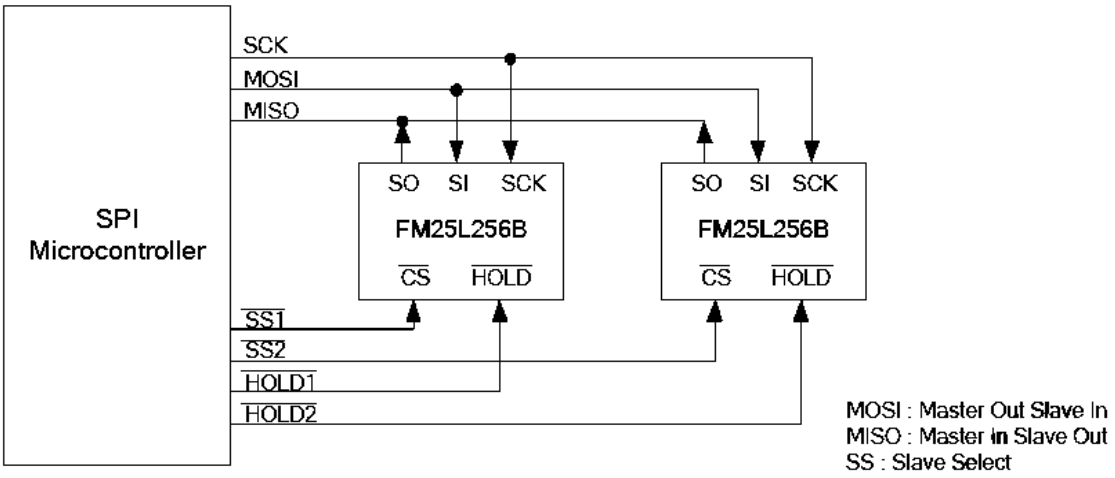


Figura 12: Conexión de varias FRAM a un Microcontrolador.

Es necesario que se tome en cuenta que los pines *SCK, MOSI (Master Out Slave In) y MISO (Master In Slave Out)*. Son comunes entre los dispositivos mientras que, los pines *Chip Select* y *Hold* deben ser manejados por separado para cada dispositivo FM25L256B.
Y que: *Clock* es SCK, *Data-in* es MOSI y *Data-out* es MISO.

2.6.4 DESCRIPCIÓN GENERAL DEL PROTOCOLO DE COMUNICACIÓN.

La interfaz SPI es una interfaz serial síncrona, porque trabaja con una señal de reloj. Su objetivo es soportar múltiples dispositivos en el bus. Cada dispositivo es activado por el microcontrolador utilizando una línea de control denominada *Chip Select*. Una vez que el pin *Chip Select* es activado por el bus maestro, la memoria FRAM comenzará a monitorear las líneas de reloj y datos.

Todas las transferencias de los datos, son sincronizadas por la línea de reloj. Un bit es transferido por cada ciclo de reloj. La mayoría de las interfaces SPI tienen 2 bits de

configuración, llamados *CPOL (Clock Polarity = Polaridad de Reloj)* y *CPHA (Clock Phase = Reloj de Fase). CPOL* determina si el estado de no transferencia de datos de la línea de reloj está en bajo *(CPOL=0)* o si se encuentra en un estado alto *(CPOL=1). CPHA* determina en qué flanco de reloj los datos son desplazados hacia dentro o hacia fuera. (Si *CPHA=0*, los datos sobre la línea MOSI son detectados cada flanco de bajada y los datos sobre la línea MISO son detectados cada flanco de subida). Cada bit tiene 2 estados, lo cual permite 4 diferentes combinaciones, las cuales son incompatibles entre si. Por lo que si dos dispositivos SPI desean comunicarse entre sí, éstos deben tener la misma Polaridad de Reloj *(CPOL)* y la misma Fase de Reloj *(CPHA).*[8]

Existen cuatro modos de reloj definidos por el protocolo SPI, éstos son mostrados en la Tabla 6:

Tabla 6: Modos del protocolo SPI

MODO	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Estos modos determinan el valor de la polaridad del reloj *CPOL* y el bit de fase del reloj *CPHA.* La mayoría de los dispositivos SPI pueden soportar al menos 2 modos de los 4 antes mencionados.

El bit de polaridad del reloj determina el nivel de estado de no transferencia de datos del reloj y el bit de fase de reloj determina qué flanco recibe un nuevo dato sobre el bus. El modo requerido para una determinada aplicación, está dado por el dispositivo esclavo. La capacidad de multi-modo combinada con un simple registro de desplazamiento, hace que el bus SPI sea muy versátil.

[8] López Pérez, Eric. "*Protocolo SPI, Teoría y aplicaciones*"

La FRAM soporta solo los modos 0 y 3. La Figura 12 muestra las relaciones entre las diferentes señales necesarias para éstos. En ambos modos, los datos se registran en la memoria FRAM en el flanco ascendente, después que el pin /CS es activado. Si el reloj inicia a partir de un estado alto, éste pasará al estado bajo antes del primer bit transferido en orden descendente, desde el bit más significativo (MSB) hasta el bit menos significativo (LSB), después de esto, el pin /CS debe pasar al estado alto.

El protocolo SPI es controlado mediante op-codes. Estos códigos de operación especifican los comandos para el dispositivo. Después de que el pin /CS se activa, el primer byte transferido desde el bus maestro, es el código de operación. Tras el código de operación, todas las direcciones y los datos son transferidos. Cabe mencionar que el pin /CS debe deshabilitarse después de que una operación se completa y antes de que un nuevo código de operación pueda ser emitido.

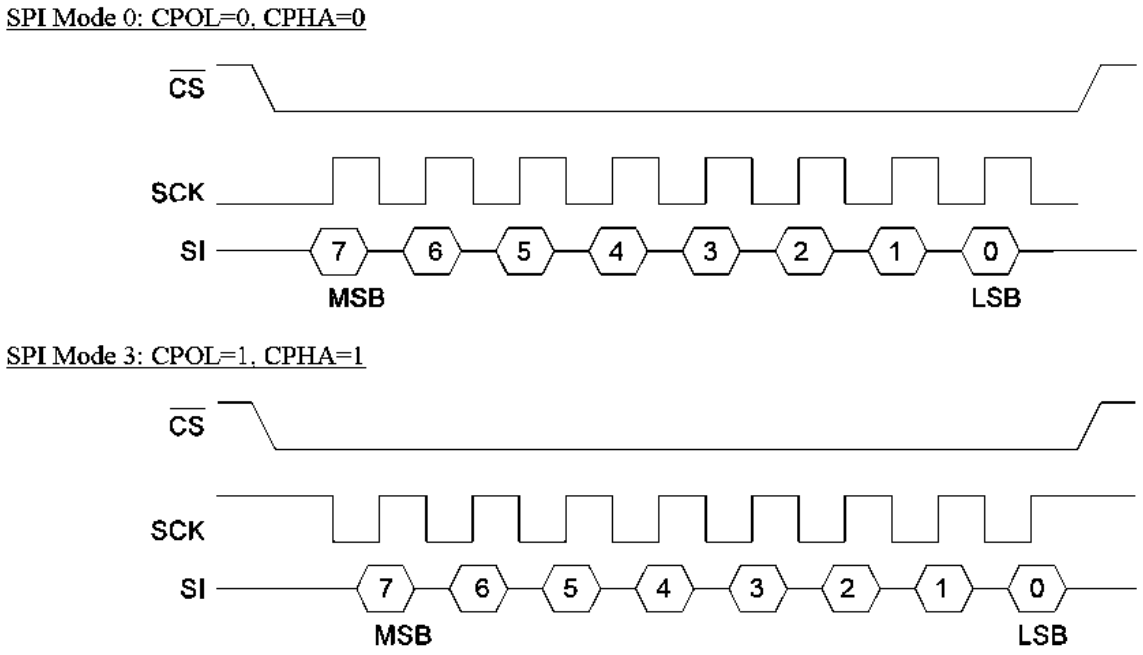


Figura 13: Protocolo SPI de la memoria FRAM.

2.6.5 ESTRUCTURA DE LOS COMANDOS.

Hay seis comandos que pueden ser usados por la memoria FRAM. Éstos se muestran en la Tabla 7. Estos códigos de operación controlan las funciones desempeñadas por la memoria y pueden ser divididos en tres categorías. En primer lugar, hay comandos que no tienen acciones futuras. Ellos realizan una sola función, por ejemplo, para permitir una operación de escritura. En segundo lugar están los comandos seguidos de bytes de información o datos. El tercer grupo incluye comandos para transacciones de la memoria seguido de la dirección y uno o más bytes de datos.

Tabla 7: Comandos de la memoria FRAM.

Nombre	Descripción	Op-code
WREN	Establece escritura habilitando LATCH	0000 0110b
WRDI	Habilita escritura	0000 0100b
RDSR	Lee el registro Status	0000 0101b
WRSR	Escribe en el registro Status	0000 0001b
READ	Lee los datos de la memoria	0000 0011b
WRITE	Escribe los datos en la memoria	0000 0010b

2.6.6 DESCRIPCIÓN DE LOS COMANDOS.

- WREN: La memoria tiene la capacidad de habilitarse. Este comando puede ser utilizado antes de cualquier operación de escritura. El uso del *op-code WREN* permitirá al usuario emitir posteriores *op-code* para cada operación de escritura, esto incluye la escritura del registro *Status* y la escritura en la memoria. El *op-code WREN* habilita un bit de bandera en el registro *Status* llamado *WEL*, éste indica el estado de *LATCH* mientras que *WEL* = 1 indica que es permitida la escritura. El intento de escribir el bit *WEL* en el registro *Status* no tiene ningún efecto en el estado de este bit. Al completar cualquier operación de escritura, este bit será limpiado automáticamente y evitará futuras escrituras sin comando *WREN*. La Figura 14 muestra la configuración del bus *WREN*.

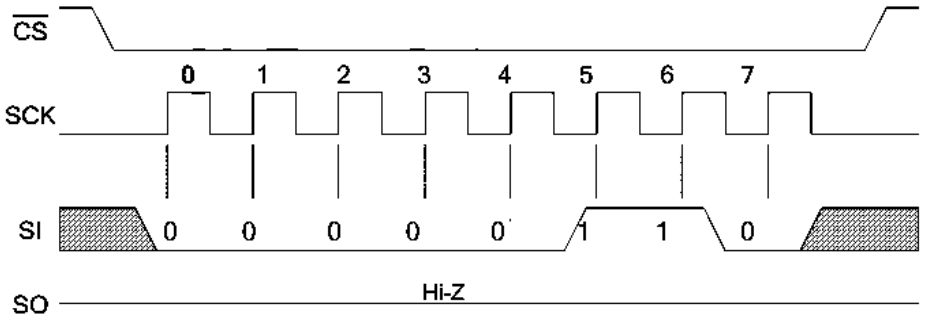


Figura 14: Configuración del Bus WREN.

- WRDI: Este comando desactiva todas las operaciones de escritura al bit *WEL* (*Write Enable Latch*). El usuario puede verificar que estas escrituras son deshabilitadas por el bit *WEL* y ver que *WEL*=0. En la Figura 15 se muestra el byte de configuración del comando *WRDI.*

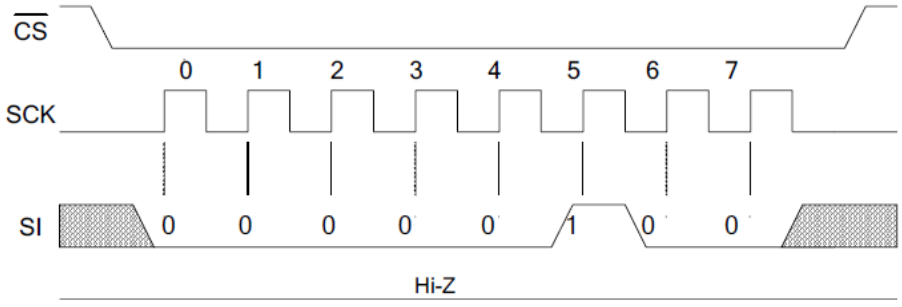


Figura 15: byte de Configuración del Comando *WRDI.*

- RDSR: Este comando permite que el bus maestro verifique el contenido del registro *Status*. La lectura del registro *Status* proporciona la información sobre el estado actual de las funciones de protección de escritura. Tras el código de operación *RSD*, la memoria devolverá un byte con el contenido del registro Status. En la Figura 16 se observa el byte de configuración del comando RDSR.

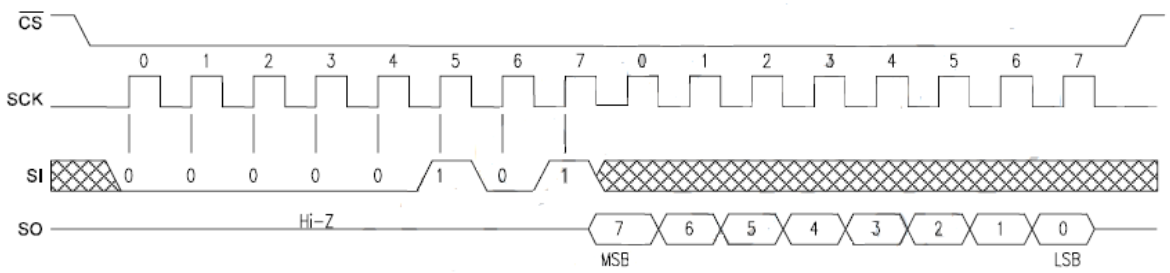


Figura 16: Lectura del registro STATUS.

- WRSR: Este comando permite al usuario seleccionar ciertas funciones de protección de escritura. Estas funciones se habilitan modificando el registro *Status.* Antes de usar el comando *WRSR*, el pin */WP* debe colocarse en alto y antes de enviar el comando *WRSR*, el usuario debe enviar el comando *WREN* para habilitar la escritura. Tome en cuenta que ejecutar el comando *WRSR* es una operación de escritura, por lo que es necesario limpiar el bit *WEL*. En la Figura 17 se muestra el byte de configuración del comando WRDR.

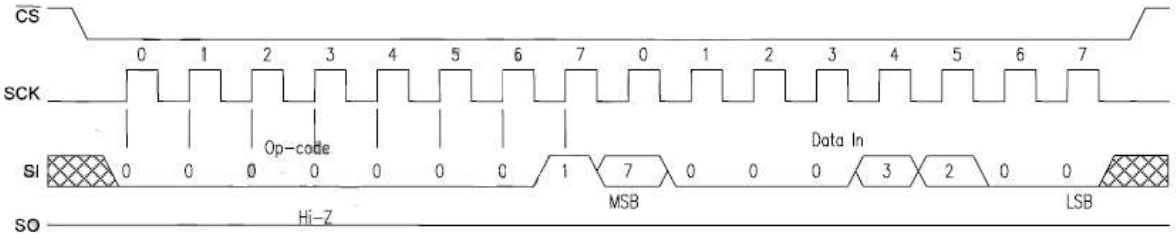


Figura 17: Comando de escritura del *WRSR*.

2.6.7 EL REGISTRO *STATUS* Y LA PROTECCIÓN CONTRA ESCRITURA.

Hay varias formas de protección contra escritura, colocar el pin */WP* a un estado lógico bajo, es una forma de protección utilizando hardware. Todas las operaciones de escritura son bloqueadas cuando el pin */WP* se encuentra en bajo. Como se describe anteriormente, la escritura al registro Status se lleva a cabo utilizando el comando *WRSR* y el pin */WP*. El registro *Status* es organizado de acuerdo a la Tabla 8.

Tabla 8: Registro Status.

Bit	7	6	5	4	3	2	1	0
Nombre	WPEN	0	0	0	BP1	BP0	WEL	0

Los bits 0, 4, 5 y 6 están en "0" lógico y no se pueden modificar (bloques protegidos). Los bits BP1 y BP0 controlan la escritura por medio de software, éstos son no volátiles. La bandera *WEL* indica el estado *Write Enable Latch*. El intento de escribir directamente al bit *WEL* en el registro *Status* no tendrá ningún efecto. Este bit es interno y se limpia mediante los comandos *WREN* y *WRDI* respectivamente. BP1 y BP0 son bits de bloqueo para protección de escritura a la memoria, con éstos se puede escoger la porción de la memoria que se desea bloquear, como se muestra en la Tabla 9.

Tabla 9: Protección de la FRAM.

BP1	BP0	Rango de direcciones protegidas.
0	0	Ninguna
0	1	6000h a 7FFFh (superior a ¼)
1	0	4000h a 7FFFh (superior a ½)
1	1	0000h a 7FFFh (todo)

Los bits BP1, BP0 y el bit *WEL* son los únicos mecanismos por software para protección contra escritura de la memoria.

El bit *WPEN* controla el efecto de pin /*WP*. Cuando el pin */WP* está en bajo, el pin /WP es ignorado. Cuando el bit *WPEN* está en alto, el pin */WP* controla el acceso para la escritura del registro Status. Así, el registro Status está protegido contra escritura si el bit *WPEN* = 1 y el pin */WP* = 0.

Lo anterior prevé un mecanismo de protección de escritura a la memoria bajo cualquier circunstancia. Esto ocurre si los bits BP1 y BP0 se establecen en 1, el bit *WPEN* se establece en 1 y el pin */WP* se establece en 0. Esto porque los bits de protección evitan

que se escriba en la memoria y la señal del pin */WP* en el hardware evita alterar los bits del bloque de protección (si el pin */WP* está en alto). En esta condición, el hardware debe permitir una operación de escritura. La Tabla 10 resume las condiciones de protección de escritura.

Tabla 10: Protección contra escritura.

WEL	WPEN	/WP	Bloques Protegidos	Bloques No Protegidos	Registro Status
0	X	X	Protegido	Protegido	Protegido
1	0	X	Protegido	No Protegido	No Protegido
1	1	0	Protegido	No Protegido	Protegido
1	1	1	Protegido	No Protegido	No Protegido

2.6.8 OPERACIÓN DE ESCRITURA.

Todas las escrituras en la memoria empiezan con un *op-code WREN*. El siguiente *op-code* es la instrucción *WRITE*, este *op-code* es seguido por una dirección de 2 bytes. El bit superior de la dirección es un bit sin cuidado, en total 15 bits especifican la dirección inicial de la operación de escritura. Los siguientes bits son datos y éstos son escritos secuencialmente.

Las direcciones son incrementadas internamente, siempre y cuando el bus maestro siga emitiendo pulsos de reloj. Si se llega a la dirección 7FFFh, el contador regresa a la dirección 0000h. Los datos se comienzan a escribir en el bit MSB. Una operación de escritura se muestra en la Figura 18. Todos los bite son leídos por la memoria en el flanco de subida de la señal SCK. Cuando /CS tiene un estado bajo, los primeros 8 pulsos de SCK son utilizados para introducir el op-code correspondiente a escritura, en los siguientes 16 pulsos del SCK, se introduce la ubicación de la memoria donde se quiere almacenar los datos, después de esto, los siguientes pulsos del SCK se utilizan para escribirlos.

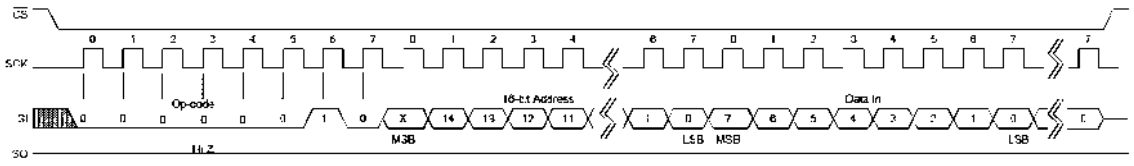


Figura 18: Operación de Escritura.

A diferencia de la EEPROM, cualquier número de bytes puede ser escrito secuencialmente y cada byte es escrito en la memoria inmediatamente, el dato es almacenado después del octavo ciclo de reloj. Los datos son aceptados en el flanco ascendente del pin /CS, con esto termina la operación *WRITE op-code*. Si el pin */WP* se activa en medio de una operación de escritura, no tendrá efecto hasta el siguiente flanco ascendente de */CS*.

2.6.9 OPERACIÓN DE LECTURA.

Antes del flanco ascendente del pin */CS*, el bus maestro puede emitir un *op-code READ*. Después de esta instrucción hay una dirección de doble byte, el bit superior de la dirección es un bit sin cuidado. En total, 15 bits especifican la dirección inicial del primer byte de la operación de lectura. Después que el *op-code* y la dirección son completados, el pin SI es ignorado. El bus maestro emite 8 pulsos de reloj, con un bit de lectura en cada uno. Las direcciones son incrementadas internamente siempre y cuando el bus maestro continúe emitiendo pulsos de reloj. Si se llega a la dirección 7FFFh, el contador regresa a la dirección 0000h. Los datos son leídos a partir del bit MSB, en el flanco ascendente de la terminal */CS*. Una operación de lectura se muestra en la Figura 19.

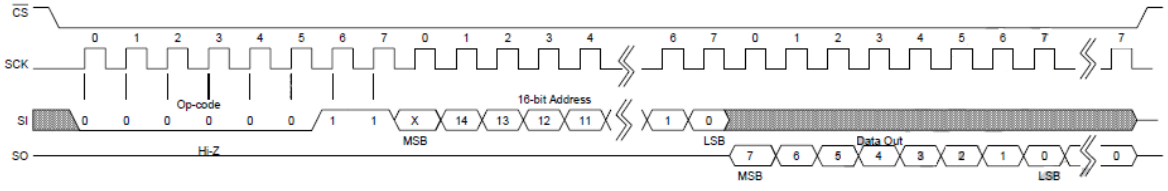


Figura 19: Operación de Lectura.

El pin */HOLD* puede ser usado para interrumpir una operación sin abortarla. Si el bus maestro pone el pin /*HOLD* en bajo mientras que *SCK* está en bajo, la operación será detenida. Tomando el pin */HOLD* en alto mientras *SCK* es bajo la operación será reanudada. La transición del pin */HOLD* debe ocurrir mientras *SCK* está en bajo.

2.7 ORGANIZACIÓN DE LA MEMORIA FRAM.

Para un funcionamiento eficiente de nuestro sistema, se organizaron las localidades de memoria destinando localidades para:

- Longitud del nombre del archivo.
- El nombre del archivo.
- Información.
- Apuntadores.
- Tabla *ASCII* de 5x7, para el display LCD.

Debido a que los archivos requieren de un nombre, debe identificar la información del mismo y no confundirla con la información de interés. Además se requieren localidades de la memoria para el código *ASCII* que utiliza el display para el despliegue de información, también localidades de memoria para banderas y apuntadores.

En la Figura 19, se observa cómo está particionada la memoria FRAM. Los primeros 64 bytes son utilizados para el nombre del archivo, éstos inician de la localidad 0x00 hasta la localidad 0x39. Los siguientes 32191 bytes son utilizados para almacenar los datos recibidos del usuario (datos de interés), éstos abarcan las localidades de memoria 0x40 hasta la localidad 0x7DFF. Los siguientes 490 bytes, son reservados para el código ASCII utilizado para que el Display muestre la información del estado del sistema. Éste código está almacenado en las localidades de memoria 0x7E00 hasta la 0x7FE9.

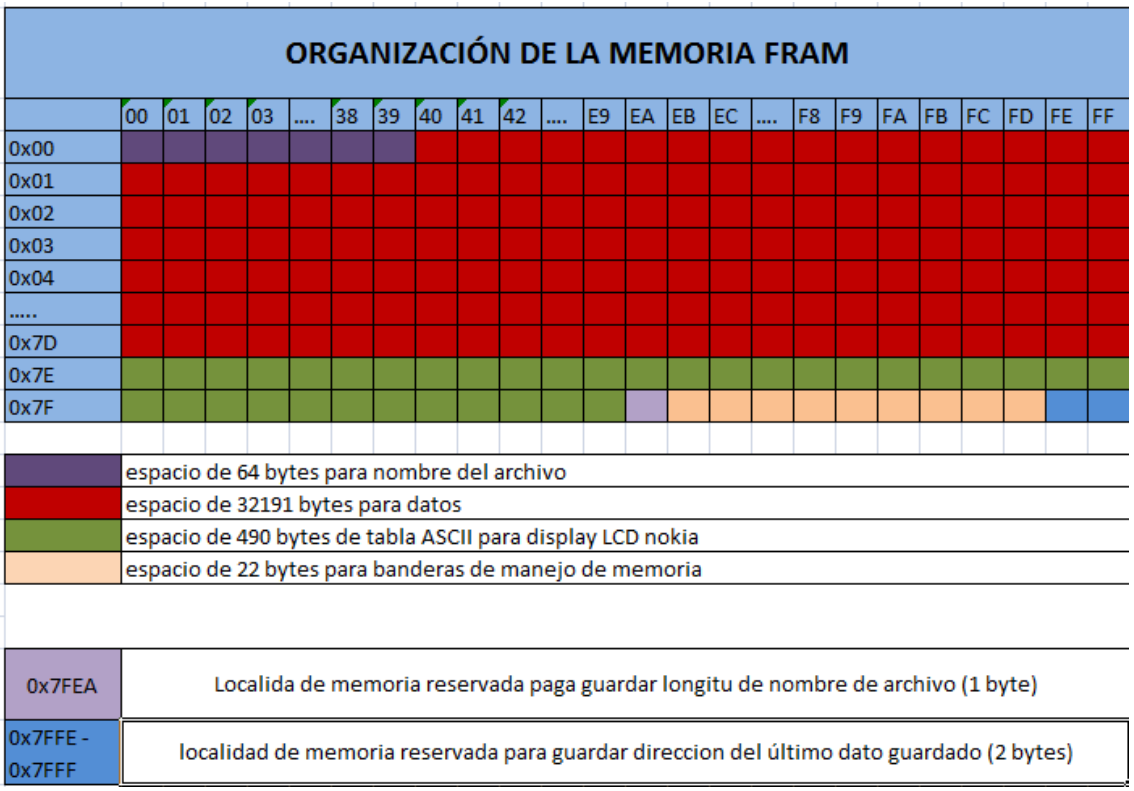


Figura 20: Organización de la Memoria.

El byte de la localidad de memoria 0x7FEA, está reservado para almacenar la longitud del nombre del archivo, esto sirve para una mayor eficiencia en el procesamiento de los datos. Los siguientes 22 bytes son banderas utilizadas para el manejo de la memoria, abarcan las localidades de memoria 0x7FEB a la 0x7FFD. Los 2 bytes siguientes son utilizados para guardar la dirección del último dato almacenado en la memoria, con el fin de tener una referencia, evitar que el sistema sobre escriba datos en alguna localidad de memoria ya ocupada y evitar la pérdida de información almacenada anteriormente, o que deje localidades de memoria vacías entre localidades utilizadas. Éstos ocupan las localidades de memoria, 0x7FFE y 0x7FFF.

2.8 EL DISPLAY GRÁFICO LCD.

En la Figura 20 se muestra el display Nokia utilizado.

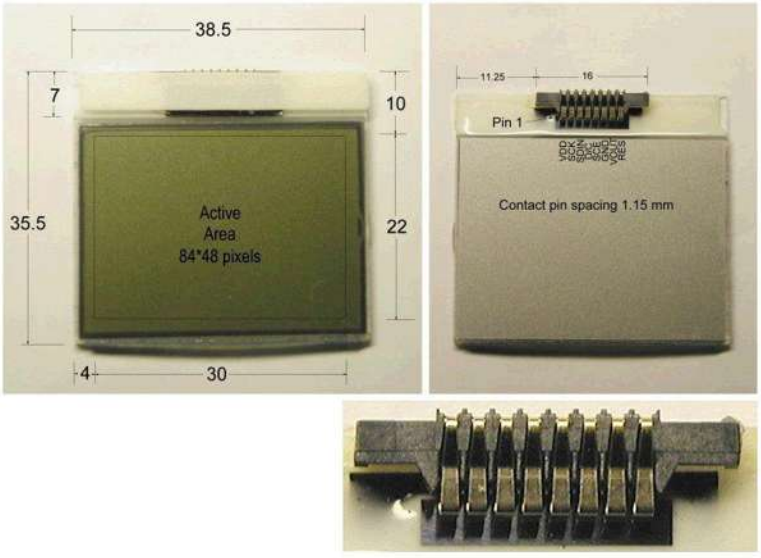


Figura 21: Display Nokia.

En nuestro sistema es muy útil proporcionar información del estado del módulo, el display utilizado es reciclado de teléfonos nokia de bajo costo, sus características son las siguientes:

- Controlador PCD8544.
- Bus SPI con línea de selección.
- Área activa de 84x48 píxeles.
- Capacidad de texto de 6x16 caracteres.
- Voltaje de alimentación de 3.3v.

2.9 EL SISTEMA INTEGRADO.

En la Figura 21 se puede observar el sistema ensamblado en una tarjeta de prueba, se observa el modem Winity con su tarjeta de prueba, el display LCD, la memoria FRAM, el

dsPIC y un circuito integrado MAX232, para la comunicación entre el dsPIC y la tarjeta del módulo Winity.

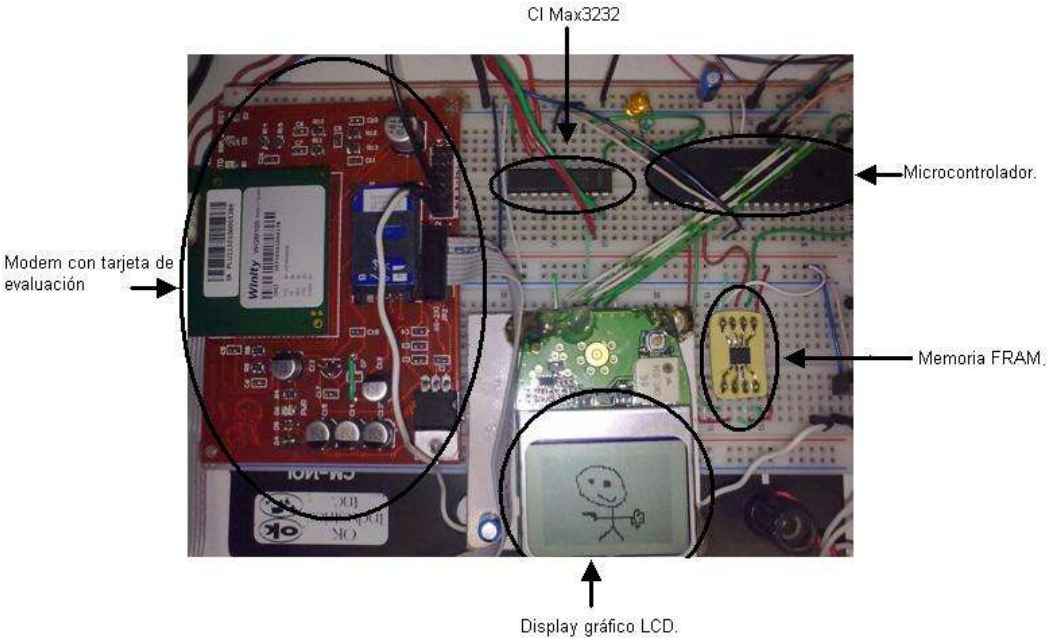


Figura 22: Integración del sistema en tableta de desarrollo.

3 LA RED DE TELEFONÍA CELULAR.

3.1 LA RED DE TELEFONÍA CELULAR GSM.

La red GSM es a comienzos del siglo XXI, el estándar de comunicación más usado de Europa. Se denomina estándar de segunda generación (2G) porque, a diferencia de la primera generación de teléfonos móviles, las comunicaciones se realizan de un modo completamente digital.

En 1982 se realizó el primer estándar y fue denominado "Groupe Special Mobile", ya para el año de 1991 se convirtió en un estándar internacional y sus siglas cambiaron a "Sistema Global de Comunicaciones Móviles".

En Europa, el estándar GSM usa las bandas de frecuencia de 900MHz y 1800 MHz. Sin embargo, en los Estados Unidos y Latinoamérica se usa la banda de frecuencia de 1900 MHz. Por esa razón, los teléfonos portátiles que funcionan tanto en Europa como en los Estados Unidos se llaman tribanda y aquéllos que funcionan sólo en Europa se denominan bibanda.

El estándar GSM permite un rendimiento máximo de 9.6 kbps, lo que permite transmisiones de voz y de datos digitales de bajo volumen, por ejemplo, mensajes de texto (SMS) o mensajes multimedia (MMS).[9]

En la Figura 23 se muestra la arquitectura de la red de telefonía celular GSM en la cual se puede observar las diferentes etapas que componen esta red.

[9] http://es.kioskea.net/contents/telephonie-mobile/gsm.php3

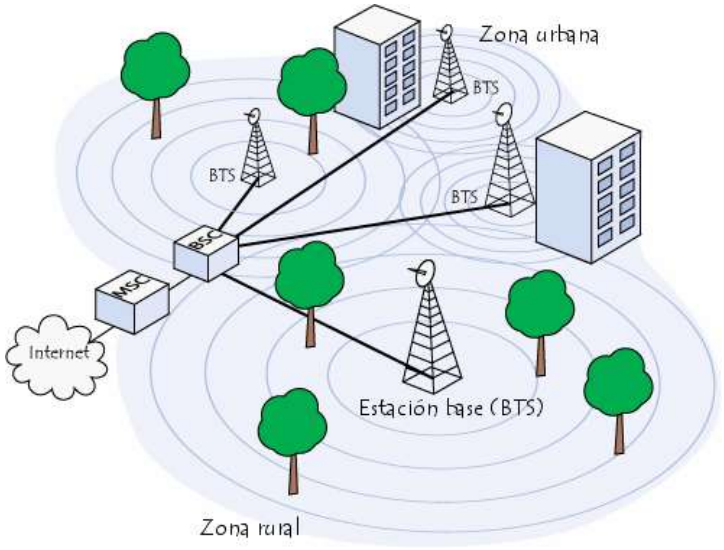


Figura 23: Arquitectura de la red celular GSM.

3.2 ESTRUCTURA DE LA RED DE TELEFONÍA CELULAR GSM**.**

En la Figura 24 se muestra la estructura de la red de telefonía celular GSM, ésta consta de 3 etapas: La estación móvil, el subsistema de estación base, y el subsistema de red.

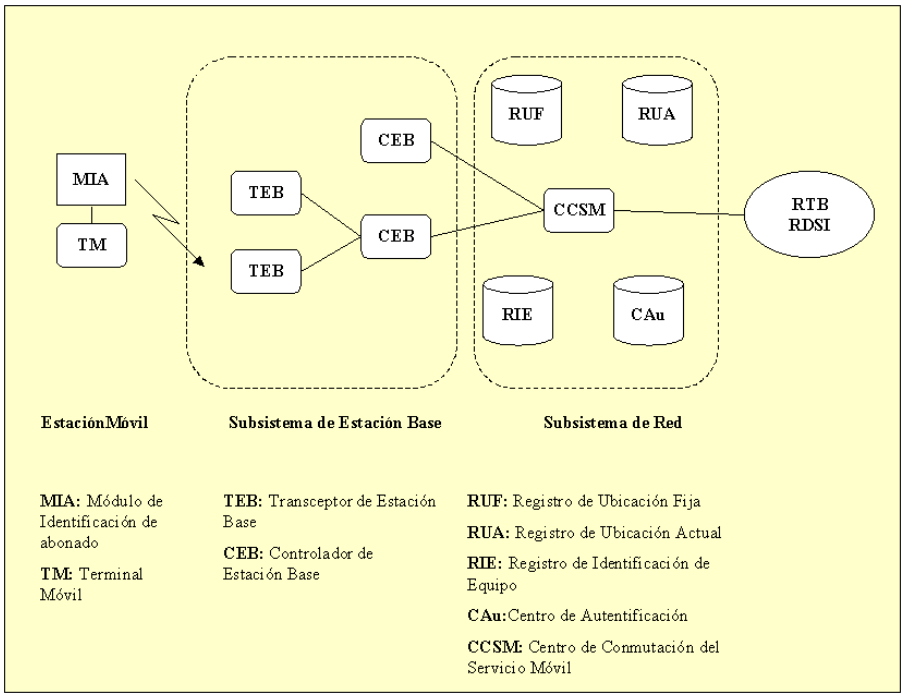


Figura 24: Subsistemas de Soporte y Operación GSM.

3.2.1 LA ESTACIÓN MÓVIL (MS).

Consta a su vez de dos elementos básicos, por un lado el terminal o equipo móvil y por otro lado el MIA. En cuanto a los equipos móviles podemos decir que existen de diferentes tipos y con diferentes características; una de las diferencias más grandes es la potencia que poseen, la cual va desde los 20 watts (generalmente instalados en vehículos) hasta los 2 watts de las terminales portátiles.

El MIA o SIM es una pequeña tarjeta inteligente que sirve para identificar las características de nuestra terminal. Esta tarjeta se inserta en el interior del móvil y permite al usuario acceder a todos los servicios disponibles por parte del operador, además que el equipo móvil posea el hardware necesario para soportar dichos servicios; sin la tarjeta MIA la terminal no sirve porque no podemos hacer uso de la red. El MIA está protegido por un número de cuatro dígitos que recibe el nombre de PIN (*Personal Identification Number*). Una vez que se introduce el PIN en el equipo móvil, éste comienza a realizar una búsqueda de redes GSM que estén disponibles y tratará de validarse en una de ellas, una vez que la red (generalmente la que tenemos contratada) ha validado nuestro equipo móvil, éste queda registrado en la célula que lo ha validado.

3.2.2 LA ESTACIÓN BASE (BSS).

La estación base se usa para conectar a las estaciones móviles con los Subsistemas de Conmutación y Red o *Network and Switching Subsystem* (NSS), además de ser los encargados de la transmisión y recepción. Estas estaciones también constan de dos elementos bien diferenciados: El transceptor de estación base (TEB) ó *Transceiver Station* (BTS) y el controlador de estación base (CEB) ó *Station Controller* (BSC). La BTS consta de transceptores y antenas usadas en cada célula de la red que suelen estar situadas en el centro de la célula, generalmente su potencia de transmisión determina el tamaño de ésta. Los BSC se utilizan como controladores de los BTS y tienen como función principal la de estar al mando de los "*handovers*". Éstos realizan la función del *roaming* en las células, los

frequency hopping y del control de las frecuencias de radio de los BTS.

3.2.3 EL SUBSISTEMA DE CONMUTACIÓN Y RED (NSS).

Este sistema se encarga de administrar las comunicaciones que se realizan entre los diferentes usuarios de la red. Para poder hacer este trabajo la NSS se divide en siete sistemas diferentes, cada uno con una misión dentro de la red:

- Centro de Comunicación del Servicio Móvil (CCSM) *ó Mobile Services Switching Center* (MSC). - Es el componente central del NSS y se encarga de realizar las labores de conmutación dentro de la red, así como de proporcionar conexión con otras redes.

- Puerta de Enlace al Centro de Comunicación Móvil (PECCM) *ó Gateway Mobile Services Switching Center* (GMSC): Un gateway es un dispositivo traductor (puede ser software o hardware que se encarga de interconectar dos redes haciendo que los protocolos de comunicaciones que existen en ambas redes se entiendan). Entonces, la misión del GMSC es esta misma, servir de mediador entre las redes de telefonía fijas y la red GSM

- Registro de Ubicación Fija (RUF) *ó Home Location Registrer* (HLR): El HLR es una base de datos que contiene información sobre los usuarios conectados a un determinado MSC. Entre la información que almacena el HLR tenemos fundamentalmente la localización del usuario y los servicios a los que tiene acceso.

- Registro de Ubicación Actual (RUA) ó *Visitor Location Registrer* (VLR): contiene toda la información sobre un usuario, esta información es necesaria para que dicho usuario acceda a los servicios de red. Forma parte del HLR con quien comparte funcionalidad.

- Centro de Autentificación (CAu) ó *Authentication Center* (AuC): Proporciona los parámetros necesarios para la autentificación de usuarios dentro de la red; también se encarga de soportar funciones de encriptación.

- Registro de Identificación de Equipo (RIE) ó *Equipment Identy Registrer* (EIR): También se utiliza para proporcionar seguridad en las redes GSM pero a nivel de equipos válidos. La EIR contiene una base de datos con todas las terminales que son válidos para ser usados en la red. Esta base de datos contiene los *International Mobile Equipment Identy* o IMEI de cada terminal, de manera que si un determinado móvil trata de hacer uso de la red y su IMEI no se encuentra localizado en la base de datos del EIR, no puede hacer uso de la red.

- *GSM Interworking Unit* (GIWU): sirve como interfaz de comunicación entre diferentes redes para comunicación de datos.

3.2.4 LOS SUBSISTEMAS DE SOPORTE Y OPERACIÓN.

Los OSS se conectan a diferentes NSS y BSC para controlar y monitorear toda la red GSM. La tendencia actual en estos sistemas es que, dado que el número de BSS se está incrementando, se pretenden delegar funciones que actualmente se encarga de hacerlas el subsistema OSS de los BTS, de modo que se reduzcan los costos de mantenimiento del sistema.

3.3 EL SERVICIO GPRS.

Dadas las características de las redes GSM, éstas presentan ciertas limitaciones para la transmisión de datos de alto volumen de información:

- Velocidad de transferencia de 9.6Kbps.
- Tiempo de establecimiento de la conexión de 15 a 30 seg.
- Las aplicaciones deben ser reiniciadas en cada conexión.
- Pago por tiempo de conexión.
- Problemas para mantener la conectividad en itinerancia (roaming).

La baja velocidad de transferencia limita la cantidad de servicios que internet nos ofrece; por ejemplo. A 9.6 Kbps no se puede navegar por internet de una manera satisfactoria. Si además, tenemos en cuenta que estamos pagando por tiempo de conexión, los costos se disparan. Esto representa un gran problema, pues no se puede comparar una hora de conversación con una hora de conexión a internet. La combinación de estos factores negativos hace que GSM sea una tecnología mayoritariamente utilizada para voz y no para datos.

Las redes GSM no se adaptan adecuadamente a las necesidades de transmisión de datos con terminales móviles. Para salvar este inconveniente se desarrolló una nueva tecnología para la transmisión de datos denominada GPRS que unifica el mundo IP (*Internet Protocol*) con el mundo de la telefonía móvil, creándose toda una red paralela a la red GSM y orientada exclusivamente a la transmisión de datos.

GPRS es una tecnología que comparte el rango de frecuencias de la red GSM utilizando una transmisión de datos por medio de “paquetes”. La conmutación de paquetes es el procedimiento más adecuado para transmitir datos debido a su naturaleza, ya que hasta ahora se habían transmitido mediante CSD (conmutación de circuitos), procedimiento más adecuado para la transmisión de voz.

Existen 2 características distintivas respecto a la forma en cómo se transmite la información por GPRS y se describen a continuación:

- ***Los canales se comparten entre los diferentes usuarios***.

En GSM, cuando se realiza una llamada, se asigna un canal de comunicación al usuario, dicho canal permanecerá asignado aunque no se envíen datos. En GPRS los canales de comunicación se comparten entre los distintos usuarios dinámicamente, de modo que un usuario sólo tiene asignado un canal cuando realmente se están transmitiendo datos. Para utilizar GPRS se precisa un teléfono que soporte esta tecnología. La mayoría de estos terminales soportarán también GSM, por lo que podrá realizar sus llamadas de voz utilizando la red GSM de modo habitual y sus llamadas de datos (conexión a internet, WAP,...) tanto con GSM como con GPRS.

La tecnología GPRS, o generación 2.5, representa un paso más hacia los sistemas inalámbricos de tercera generación o UMTS (*Universal Mobile Telecommunications System*). Su principal característica radica en la posibilidad de disponer de una terminal permanentemente conectada que tarifica únicamente por el volumen de datos transferidos (enviados y recibidos) y no por el tiempo de conexión.

- ***Obtiene mayor y mejor eficiencia de la red.***

De manera convencional, la transmisión de datos inalámbrica se realizaba utilizando un canal dedicado GSM a una velocidad máxima de 9.6 Kbps (restricción ocasionada por la arquitectura de la red GSM). Con el GPRS, no sólo la velocidad de transmisión de datos se ve aumentada desde 40 Kbps y hasta un máximo de 115 Kbps, sino que además la tecnología utilizada permite compartir cada uno de los canales por varios usuarios, mejorando así la eficiencia en la utilización de los recursos de red.

GPRS es una evolución no drástica de la actual red GSM, ya que no conlleva grandes inversiones y reutiliza parte de la infraestructura actual de GSM. Por este motivo, GPRS tiene la misma cobertura que la actual red GSM.

4 EL PROTOCOLO PARA LA TRANSFERENCIA DE DATOS.

4.1 INTRODUCCIÓN.

Para transferir información hacia Internet, es necesario seguir algunas reglas y protocolos que nos permitan garantizar que la información llegará de forma segura a su destino. Además se debe cumplir cierto estándar para que las computadoras puedan intercambiar la información independientemente de la plataforma y/o sistema operativo que los usuarios lleguen a utilizar.

En el mundo de la informática, existen diversos protocolos para la transferencia de archivos, los cuales poseen ventajas y desventajas respecto de otros debido a características como la encriptación de los datos entre otras. Para nuestro caso decidimos utilizar un protocolo básico, el cual garantiza la transferencia de la información además de realizarlo de una forma sencilla.

4.2 EL PROTOCOLO FTP

FTP (*File Transfer Protocol* – Protocolo de Transferencia de Archivos) en informática, es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (*Transmisión Control Protocol*), basado en la arquitectura cliente-servidor. Desde un equipo cliente se puede conectar a un servidor para descargar archivos desde él o para enviarle archivos, independientemente del sistema operativo utilizado en cada equipo.

El Servicio FTP es ofrecido por la capa de Aplicación del modelo de red TCP/IP, utilizando normalmente el puerto de red 20 y el 21. Un problema básico de FTP es que está pensado para ofrecer la máxima velocidad en la conexión, pero no la máxima seguridad, ya que todo el intercambio de información, desde el *login* y *password* del usuario en el servidor, hasta la transferencia de cualquier archivo, se realiza en texto plano sin ningún

tipo de cifrado, con lo que un posible atacante puede capturar este tráfico, acceder al servidor, o apropiarse de los archivos transferidos.

Para solucionar este problema son de gran utilidad aplicaciones como SPC (*secure copy*) y SFTP (*Secure File Transfer Protocol*), incluidas en el paquete SSH (*Secure SHel*), que permiten transferir archivos pero cifrando todo el tráfico.

La versión original del FTP fue publicado como RFC 114 (*Request for Comments*) el 16 de abril de 1971, y más adelante reemplazado por el RFC 765 (junio de 1980) y el RFC 959 (octubre de 1985), la versión que se usa actualmente. Muchos han propuesto alternativas a la versión de 1985, como por ejemplo el RFC 2228 (junio de 1997) que propone extensiones de seguridad y la RFC 2428 (septiembre de1998) que añade soporte para Ipv6 y define un nuevo tipo de modo pasivo.

4.3 EL MODELO FTP.

En la Figura 25, se muestra el modelo simplificado de una conexión cliente servidor por FTP.

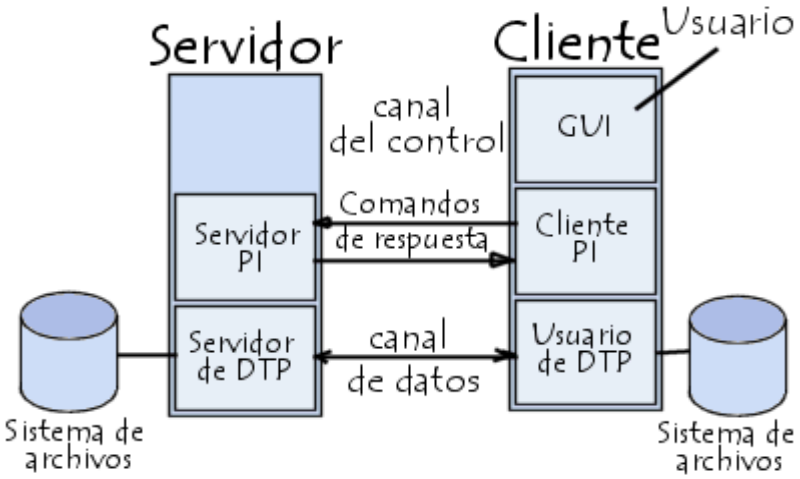


Figura 25: Modem conexión cliente servidor por FTP.

En el modelo, el PI (intérprete de protocolo) de usuario, inicia la conexión de control en el puerto 21. Las órdenes FTP estándar las genera el PI de usuario y se

transmiten al proceso servidor a través de la conexión de control. Las respuestas estándar se envían desde el PI del servidor al PI de usuario por la conexión de control como respuesta a las órdenes.

Estas órdenes FTP especifican parámetros para la conexión de datos (puerto de datos, modo de transferencia, tipo de representación y estructura) y la naturaleza de la operación sobre el sistema de archivos (almacenar, recuperar, añadir, borrar, etc.). El DTP (proceso de transferencia de datos) de usuario u otro proceso en su lugar, debe esperar a que el servidor inicie la conexión al puerto de datos especificado (puerto 20 en modo activo o estándar) y transferir los datos en función de los parámetros que se hayan especificado.

Vemos también en el diagrama que la comunicación entre cliente y servidor es independiente del sistema de archivos utilizado en cada computadora, de manera que no importa que sus sistemas operativos sean distintos, porque las entidades que se comunican entre sí son los PI y los DTP, que usan el mismo protocolo estandarizado.

También hay que destacar que la conexión de datos es bidireccional, es decir, se puede usar simultáneamente para enviar y para recibir comandos entre el cliente y el servidor.

4.4 EL SERVIDOR FTP.

Un servidor FTP es un programa especial que se ejecuta en un equipo servidor normalmente conectado a Internet (aunque puede estar conectado a otros tipos de redes, LAN, MAN, etc.). Su función es permitir el intercambio de datos entre diferentes servidores/ordenadores.

Por lo general, los programas servidores FTP no suelen encontrarse en los ordenadores personales, por lo que un usuario normalmente utilizará el FTP para conectarse remotamente a uno y así intercambiar información con él.

Las aplicaciones más comunes de los servidores FTP suelen ser el alojamiento web, en el que sus clientes utilizan el servicio para subir sus páginas web y sus archivos correspondientes; o como servidor de respaldo de los archivos importantes que pueda tener una empresa. Para ello, existen protocolos de comunicación FTP para que los datos se transmitan cifrados, como el SFTP.

4.5 EL CLIENTE FTP.

Un cliente FTP es un programa que se instala en el computador del usuario, y que emplea el protocolo FTP para conectarse a un servidor FTP y transferir archivos, ya sea para descargarlos o para subirlos.

Para utilizar un cliente FTP, se necesita conocer el nombre del archivo, el computador en que reside (servidor, en el caso de descarga de archivos), el computador al que se quiere transferir el archivo (en caso de subir información al servidor), y la carpeta en la que se encuentra.

Algunos clientes de FTP básicos en modo consola vienen integrados en los sistemas operativos, incluyendo Microsoft Windows, DOS, GNU/Linux y Unix. Sin embargo, hay disponibles clientes con opciones añadidas e interfaz gráfica. Aunque muchos navegadores tienen integrado FTP, es más confiable a la hora de conectarse con servidores FTP no anónimos, utilizar un programa cliente.

4.6 ACCESO AL SERVIDOR FTP.

4.6.1 ACCESO ANÓNIMO.

Los servidores FTP anónimos ofrecen sus servicios libremente a todos los usuarios, permiten acceder a sus archivos sin necesidad de tener un ‘USER ID’ o una cuenta de usuario. Es la manera más cómoda fuera del servicio web de permitir que todo el mundo

tenga acceso a cierta información sin que para ello el administrador de un sistema tenga que crear una cuenta para cada usuario.

Si un servidor posee servicio 'FTP anonymous' solamente con teclear la palabra "anonymous", cuando pregunte por el usuario, se tendrá acceso a ese sistema. No se necesita ninguna contraseña preestablecida, aunque se tendrá que introducir una sólo para ese momento, normalmente se suele utilizar la dirección de correo electrónico propia.

Solamente con eso se consigue acceso a los archivos del FTP, aunque con menos privilegios que un usuario normal. Generalmente solo podrás leer y copiar los archivos existentes, pero no modificarlos ni crear otros nuevos.

Normalmente, se utiliza un servidor FTP anónimo para depositar grandes archivos que no tienen utilidad si no son transferidos a la máquina del usuario, como por ejemplo programas, y se reservan los servidores de páginas web (http) para almacenar información textual destinada a la lectura en línea.

4.6.2 ACCESO MEDIANTE USUARIO.

Si se desea tener privilegios de acceso a cualquier parte del sistema de archivos del servidor FTP, de modificación de archivos existentes, y de posibilidad de subir nuestros propios archivos, generalmente se suele realizar mediante una cuenta de usuario. En el servidor se guarda la información de las distintas cuentas de usuario que pueden acceder a él, de manera que para iniciar una sesión FTP debemos introducir una autentificación (*login*) y una contraseña (*password*) que nos identifica.

4.6.3 ACCESO DE INVITADO.

El acceso sin restricciones al servidor que proporcionan las cuentas de usuario implica problemas de seguridad, lo que ha dado lugar a un tercer tipo de acceso FTP denominado invitado (*guest*), que se puede contemplar como una mezcla de los dos anteriores.

La idea de este mecanismo es la siguiente: se trata de permitir que cada usuario se conecte a la máquina mediante su *login* y su *password*, pero evitando que tenga acceso a partes del sistema de archivos que no necesita para realizar su trabajo. De esta forma accederá a un entorno restringido, algo muy similar a lo que sucede en los accesos anónimos, pero con más privilegios.

4.7 CLIENTES FTP BASADOS EN WEB.

Un "cliente FTP basado en WEB" no es más que un Cliente FTP al cual podemos acceder a través de nuestro Navegador Web sin necesidad de tener otra aplicación para ello. El usuario accede a un servidor web (http) que lista los contenidos de un servidor FTP. El usuario se conecta mediante http a un servidor web, éste se conecta mediante FTP al servidor FTP. El servidor web actúa de intermediario haciendo pasar la información desde el servidor ftp en los puertos 20 y 21 hacia el puerto 80 http que ve el usuario.

Este tipo de acceso se implementó debido a que siempre hay momentos en que nos encontramos fuera de casa, no llevamos el ordenador portátil encima y necesitamos realizar alguna tarea urgente desde un ordenador de acceso público, de un amigo, del trabajo, la universidad, etc. Lo más común es que no estén instaladas las aplicaciones que necesitamos y en muchos casos hasta carecemos de los permisos necesarios para realizar su instalación. Otras veces estamos detrás de un Proxy o Cortafuegos que no nos permite acceder a servidores FTP externos.

4.8 MODOS DE CONEXIÓN DEL CLIENTE MEDIANTE FTP.

FTP admite dos modos de conexión del cliente. Estos modos se denominan Activo (o Estándar, o PORT, debido a que el cliente envía comandos tipo PORT al servidor por el canal de control al establecer la conexión) y Pasivo (o PASV, porque en este caso envía comandos tipo PASV). Tanto en el modo Activo como en el modo Pasivo, el cliente establece una conexión con el servidor mediante el puerto 21, que establece el canal de control.

4.8.1 MODO ACTIVO.

En la Figura 26, se muestra el proceso para la conexión a un servidor mediante el modo activo.

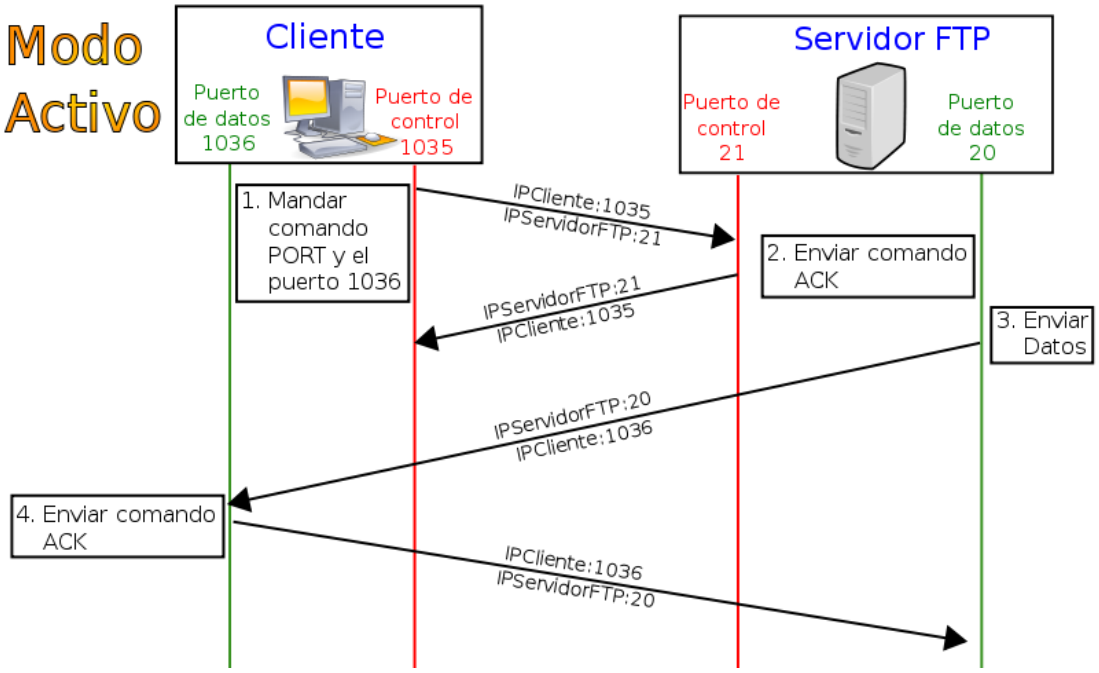


Figura 26: Conexión a un Servidor Mediante el Modo Activo.

En modo Activo, el servidor siempre crea el canal de datos en su puerto 20, mientras que en el lado del cliente el canal de datos se asocia a un puerto aleatorio mayor que el 1024. Para ello, el cliente manda un comando PORT al servidor por el canal de control indicándole ese número de puerto, de manera que el servidor pueda abrirle una conexión de datos por donde se transferirán los archivos y los listados, en el puerto especificado.

Lo anterior tiene un grave problema de seguridad, y es que la máquina cliente debe estar dispuesta a aceptar cualquier conexión de entrada en un puerto superior al 1024, con los problemas que ello implica si tenemos el equipo conectado a una red insegura como Internet. De hecho, los cortafuegos que se instalen en el equipo para evitar ataques

seguramente rechazarán esas conexiones aleatorias. Para solucionar esto se desarrolló el modo Pasivo.

4.8.2 MODO PASIVO.

En la Figura 27, se muestra el proceso para la conexión a un servidor mediante el modo pasivo.

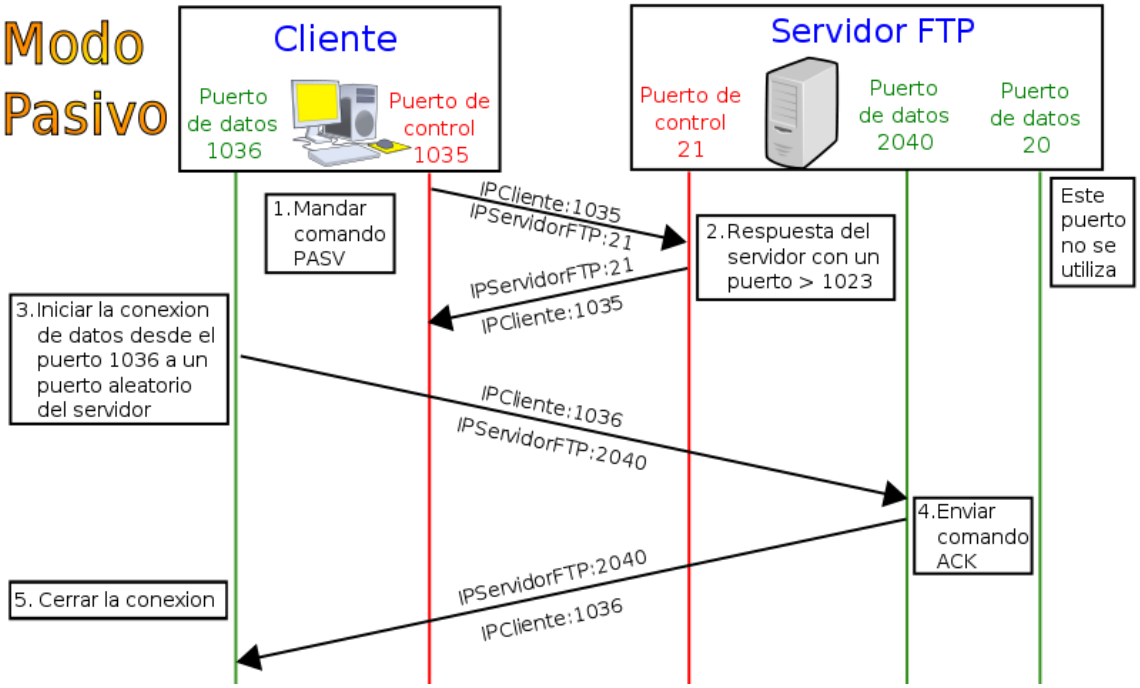


Figura 27: Proceso de conexión a un servidor en modo pasivo.

Cuando el cliente envía un comando PASV sobre el canal de control, el servidor FTP le indica por el canal de control, el puerto al que debe conectarse el cliente (mayor a 1023 del servidor). El cliente inicia una conexión desde el puerto siguiente al puerto de control, hacia el puerto del servidor especificado anteriormente.

Antes de cada nueva transferencia, tanto en el modo Activo como en el Pasivo, el cliente debe enviar otra vez un comando de control (PORT o PASV, según el modo en el

que se haya conectado), y el servidor recibirá esa conexión de datos en un nuevo puerto aleatorio (si está en modo pasivo) o por el puerto 20 (si está en modo activo).

4.9 TIPOS DE TRANSFERENCIA DE LOS ARCHIVOS.

Es importante conocer cómo debemos transportar un archivo a lo largo de la red. Si no utilizamos las opciones adecuadas podemos destruir la información del archivo. Por eso, al ejecutar la aplicación FTP, debemos recordar utilizar uno de estos comandos (o poner la correspondiente opción en un programa con interfaz gráfica):

- Tipo ASCII: Adecuado para transferir archivos que sólo contengan caracteres imprimibles (archivos ASCII, no archivos resultantes de un procesador de texto), por ejemplo páginas HTML, pero **no** las imágenes que puedan contener.

- Tipo binario: Este tipo es usado cuando se trata de archivos comprimidos, ejecutables para PC, imágenes, archivos de audio Etc.

4.10 PRUEBA DE TRANSFERENCIA DE DATOS MEDIANTE EL MODEM (GPRS Y FTP).

Los conceptos antes mencionados son la base para entender y realizar una conexión FTP entre un servidor FTP conectado a Internet y un modem GPRS. En esta parte se muestran las pruebas de conexión realizadas entre el modem GPRS y el servidor FTP gratuito. En la Figura 28 y Figura 29 se muestra la prueba realizada para la transferencia de datos a un servidor en Internet utilizando GPRS y una conexión a un servidor FTP. Todos los “OK” existentes en las Figura 28 y Figura 29, son respuesta del sistema indicando que se realizó correctamente la instrucción.

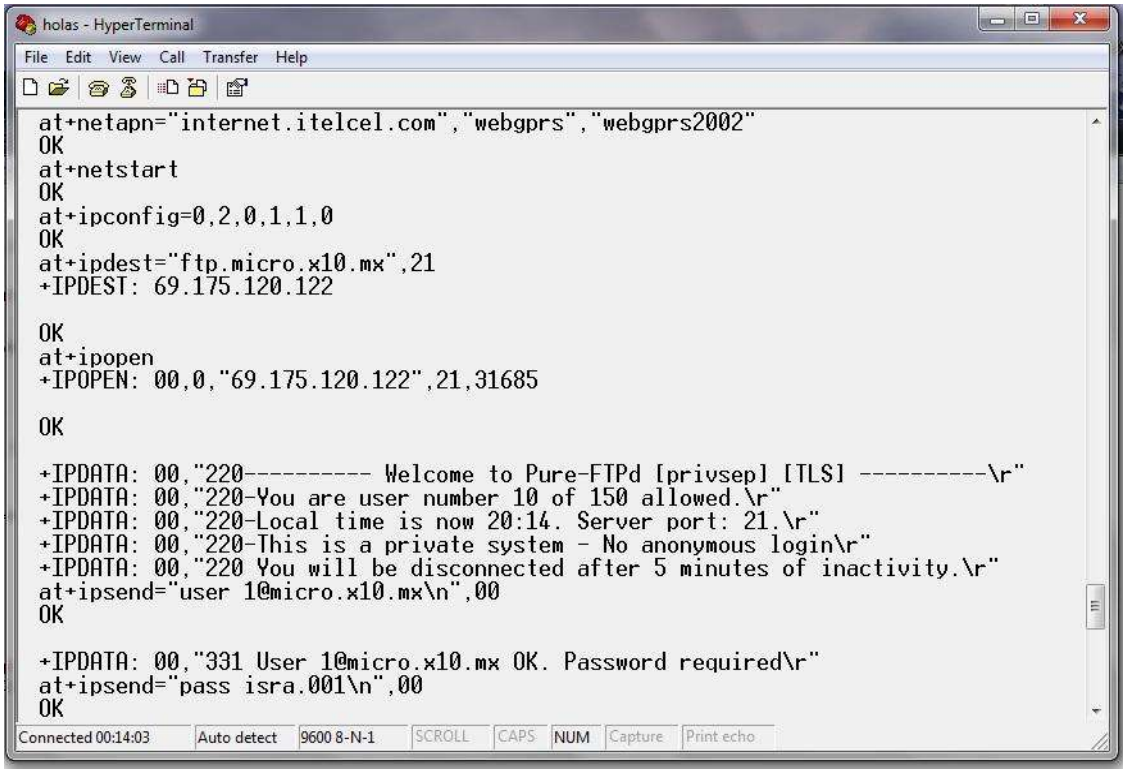


Figura 28: Transferencia de datos a un servidor en Internet.

A continuación se describen los comandos utilizados:

a) At+netapn: Le indica al modem los parámetros para conectarse a través del protocolo GPRS a la red GSM. Para esta prueba la configuración es la siguiente.
    a. APN (nombre del punto de acceso): internet.itelcel.com
    b. Username: webgprs
    c. Password: webgprs2002
b) At+netstart: Se conecta a la red GPRS con los parámetros del comando NETAPN.
c) At+ipconfig: Este comando configura varios aspectos de la interfaz de sockets de AT. En este caso es configurado de la siguiente forma:

    a. Type: 0 = TCP
    b. Multi-socket mode: 2 = *Múltiples sockets sin soporte auto-close cuando el soket alcanza su límite, + IPDATA muestra el ID del socket*.
    c. IPSEND format: *0 = envía información en ASCII plano.*
    d. IPDATA format: 1 = *recibe información en ASCII*

e. IPSEND timeout: *Tiempo de espera de entrada en segundos para el comando + IPSENDB. El valor predeterminado es 60. Se establece en 0 para desactivar el tiempo de espera.*

d) At+ipdest: Este comando configura las direcciones de destino para el socket del comando abierto. Está configurado de la siguiente forma:

   a. Dirección IP o nombre de dominio: ftp.micro.x10.mx
   b. Puerto : 21

e) At+ipopen: Abre un socket. Si la dirección DESTINO es "0.0.0.0", este comando iniciará un socket de servidor en el puerto de destino.
f) At+ipsend: El primero indica la dirección donde se enviará el archivo, la segunda envía la contraseña del cliente.
g) +IPDEST, +IPOPEN y +IPDATA: Son respuestas del sistema.

En la Figura 29 se envía un archivo al servidor con el nombre "prueba001.txt". Primero se configuran los parámetros para la conexión del modem:

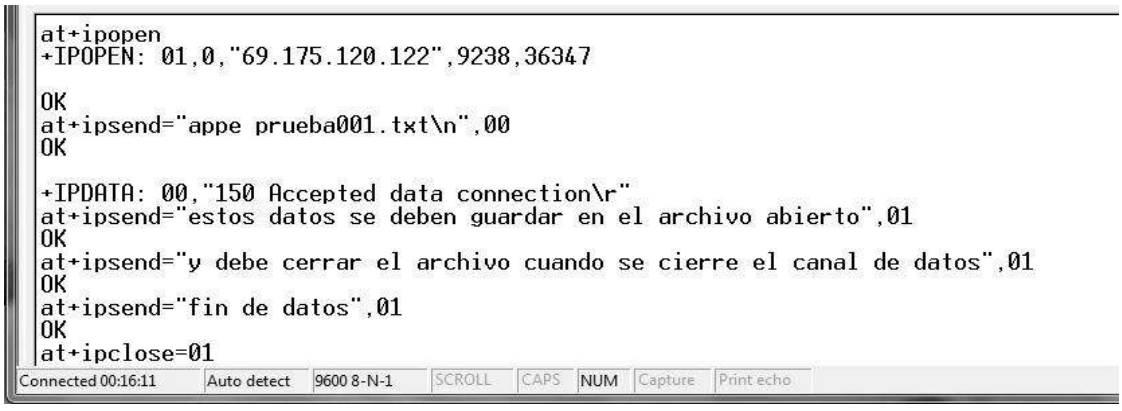


Figura 29: Transferencia de datos a un servidor FTP en Internet.

a) at+ipopen: *se indica al modem que va a configurar para la conexión.*
b) +IPOPEN: 01,0,0 “69.175.120.122”.9238,36347: *parámetros de conexión,*

Luego se indica el servidor a donde se enviará el archivo:

c) at+ipsend=“app prueba 001.txt\n”,00

Después se dan los datos que conforman el archivo, lo que se encuentra entre comillas son los datos que se enviaron al servidor.

d) at+ipsend=“estos datos se deben guardar en el archivo abierto”.

Por último se cierra la conexión.

e) at+ipclose=01

Los archivos enviados fueron revisados en un servidor web, la Figura 30 muestra el servidor utilizado y la Figura 31, muestra los datos guardados en el archivo “prueba001.txt”.

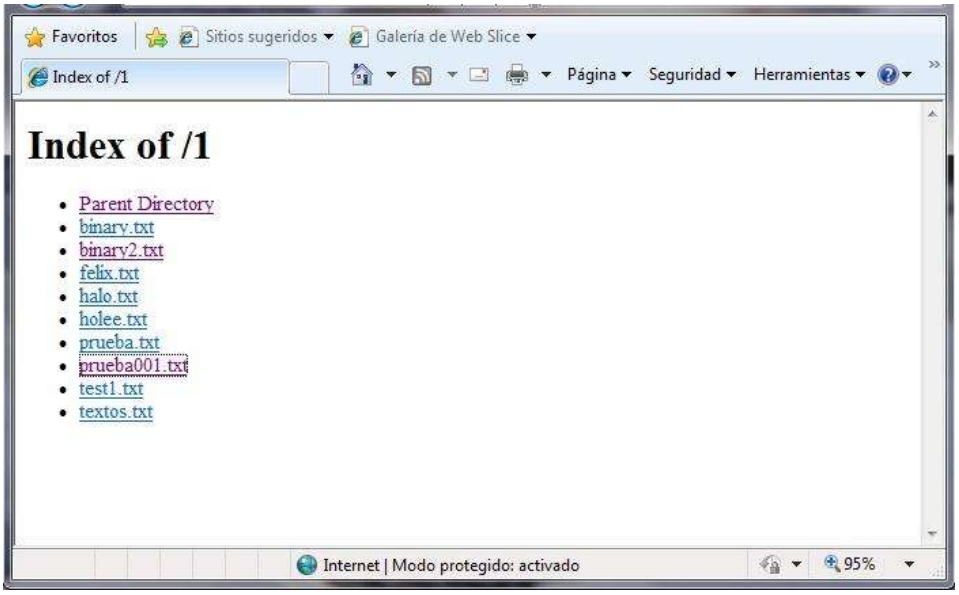


Figura 30: Servidor web utilizado para las pruebas realizadas al sistema.

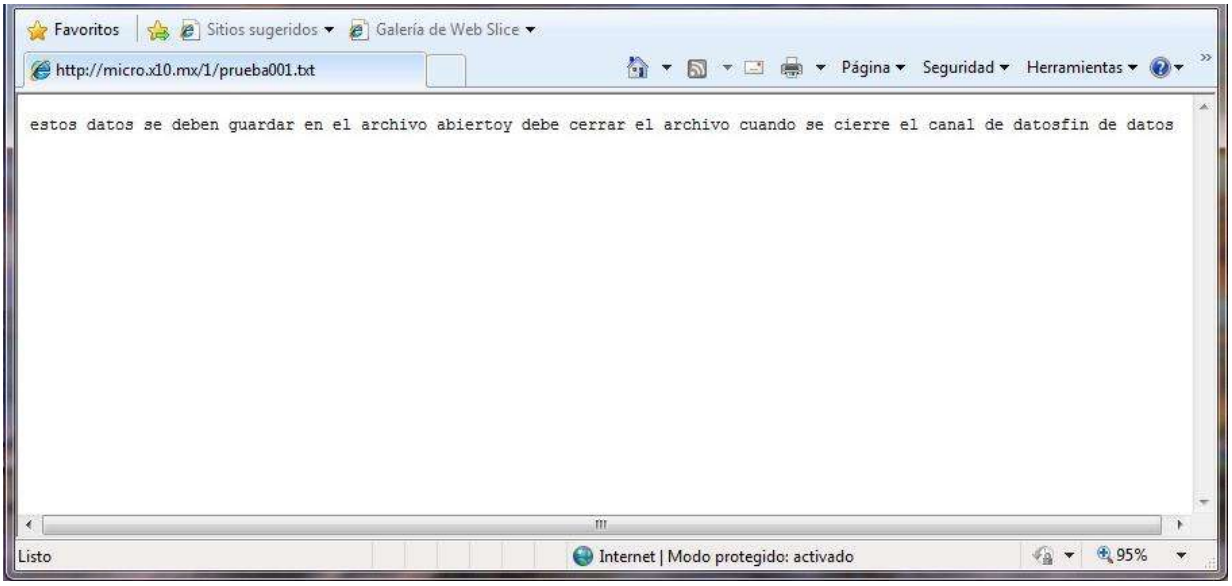


Figura 31: Archivo enviado al servidor visto con un navegador.

5 SOFTWARE IMPLEMENTADO.

5.1 INTRODUCCIÓN.

En este capítulo se presenta el software implementado en el microcontrolador dsPIC30F4011 para el módulo desarrollado. Este software está elaborado en base a una técnica de programación denominada multitarea y máquina de estados, esto con el objetivo de simplificar la programación. Se desarrollaron tareas para el control de los siguientes dispositivos:

- Display Gráfico LCD (Máquina de estados).
- Módulo GPRS (Multitarea).
- Memoria FRAM (Máquina de estados).

También se desarrolló una interfaz de comunicación con el "usuario" basada en comandos AT.

5.2 PSEUDOCÓDIGO.

El programa consta de un módulo principal y tres tareas principales, éstas son:

- Tarea USART1
    - Tarea de interfaz con el cliente.
- Tarea USART2
- Tarea de la memoria FRAM.

5.2.1 MÓDULO PRINCIPAL.

Este es un ciclo infinito que está ejecutando las tres tareas principales, a continuación mostramos el pseudocódigo del módulo principal:

-Definición de variables.

-Ciclo Principal.

-**Inicializa** display gráfico, memoria, timer 3 y puertos USART1 y USART2.

-**Limpia e inicializa** los puertos USART.

-**Despliega** información en el LCD.

-**Despliega** "U.M.S.N.H.
LAB. ELECTRONICA
MODULO G.P.R.S"

-Fin de ciclo.

-Ciclo infinito

-**Ejecuta** tarea del puerto USART 1.

-**Ejecuta** tarea del puerto USART 2.

-**Ejecuta** tarea de la memoria FRAM.

-**Configuración** del Timer 3 para monitoreo de actividad.

-Basado en el oscilador interno.

-1 mSeg si ocurre7670 cuentas.

-**Cuenta** 60000 para tener 1 minuto de tiempo base.

-Fin de cliclo

5.2.2 TAREA USART 1.

Esta tarea se encarga básicamente de corroborar si llegan comandos o datos al puerto serial. Si son comandos los ejecuta, si son datos los almacena. A continuación tenemos el pseudocódigo de la Tarea USART1.

-**Definición** de variables y banderas locales.

-**Definición** de variables y banderas globales.

-**Interrupción** RX del puerto USART1.

-**Revisa** si llegaron datos.

-Si está en modo de **comandos**:

-**Revisa** si llegó un comando válido y lo ejecuta.

-Si no es un comando válido envía un error al cliente.

-**Limpi**a el buffer.

-Si está en modo de **datos**:

- **Almacena** los datos del buffer.
- **Limpia** buffer.

- **Configuración** del puerto USART1

- Velocidad: 9600 bps.

- **Limpia** banderas de interrupción y de transmisión de TX1 y RX1.

-**Habilita** interrupción RX1.

-**Configuración** para limpiar el buffer y banderas de USART1.

-Tarea del puerto de recepción.

-**Declaración** de variables.

-**Corrobora** si llegó el comando de conexión (ATT).

-Si es comando **válido**:

-**Limpia** variables usart 1.

-**Despliega** "ATT-->OK" en display.

-**Envía** un "OK" al cliente.

-**Corrobora** si llegó comando de recepción de nombre. (ATN).

-**Corrobora** longitud del nombre.

-Si la **longitud es incorrecta**:

-**Limpia** variables usart1.

-**Despliega** "ATN-->ERROR" en display.

-**Envía** "OK" al cliente.

-Si la **longitud es correcta**:

-**Envía** los datos a la memoria.

-**Limpia** las variables USART1.

-**Despliega** “ATN-->OK” en display.

-**Corrobora** si llegó comando para **guardar** datos en la memoria.

-**Corrobora** longitud máxima de datos continuos (90).

-Si la **longitud es correcta**:

-**Limpia** variables USART1, espera datos.

-**Despliega** “ATG-->”en el display.

-**Envía** nueva línea para recepción de datos.

-Si la **longitud es incorrecta**:

-**Limpia** variables usart 1.

-**Despliega** “Command Error” en display.

-**Envía** “OK” al cliente.

-**Corrobora** si llegó el comando para **enviar** datos de la memoria.

-Si existe comando, **activa** bandera para enviar datos:

-**Limpia** las variables USART1.

-**Despliega** “ATE > Send” en display.

-**Envía** “OK” al cliente.

-**Corrobora** si llegó comando para **borrar** la memoria.

-Si **existe** comando válido:

-**Borra** la memoria FRAM.

- **Limpia** variables USART1.

-**Despliega** “ERASED FRAM” en display.

- **Envía** un “OK” al cliente.

-Si **no existe** un comando válido:

- **Limpia** variables usart 1.

-**Despliega** “Command Error” en display.

- **Envía** un “OK” al cliente.

-**Corrobora** si llegó comando para **grabar** en FRAM.

-Si **existe** comando válido:

-**Guarda** los datos en FRAM.

-**Despliega** “ATG-> Data Saved” En display.

-**Envía** un “OK” al cliente.

-Si los datos se **transmitieron** al servidor:

- **Envía** un “OK” al cliente.
- **Borra** la memoria FRAM.

-Si los datos **no se transmitieron**:

-**Envía** un mensaje de error al cliente.

-Si el **comando no es válido envía** un mensaje de error de comando al cliente.

5.2.3 TAREA USART 2.

Esta tarea es la responsable de la conexión GPRS y el envío de los datos al servidor. A continuación tenemos el pseudocódigo de esta tarea.

-**Definición** de constantes.

-**Definición** de cadenas de configuración del modem.

-**Definición** de cadenas de configuración de conexión FTP.

-**Definición** de cadenas locales del puerto serie.

-**Definición** de variables.

-**Interrupción** de tiempo base.

-Basado en oscilador interno.

-1mSeg si ocurren 7670 ciclos de reloj.

-**Cuenta** 60000 cuentas para tener 1 minuto de tiempo base.

-Si transcurren 15 minutos se **cierran** las conexiones activas.

-**Asigna** al valor del socket para la conexión de datos.

-Interrupción de recepción de datos USART2

-**Lee el dato**

-Si llega un retorno de carro o un “Enter” **incrementa** el número de comandos.

-En caso contrario es un carácter y lo **almacena** en el buffer b_Usart2.

-Si llega al final del buffer lo **limita** al máximo.

-Máquina de estados para transmisión de datos GPRS.

-Si existe **petición** de envío de datos:

-**Inicializa** Buffer.

-**Inicializa** el periodo de transmisión. (15 min)

- **Comprueba conexión** con el modem (AT)

- **Espera respuesta** “ok” o “error” del modem, en caso de “error” regresa al caso anterior.

- **Configura** parámetros GPRS.

- **Comprueba** si el modem responde “ok” o “error”, en caso de “error” regresa al caso anterior.

-**Configura** IP de la conexión GPRS.

- **Espera respuesta** “ok” o “error” del modem, en caso de “error” regresa al caso anterior.

- **Conecta** el modem a la red GPRS.

- **Comprueba** si el modem responde “ok” o “error”, en caso de “error” regresa al caso anterior.

-**Realiza la petición de conexión** al servidor FTP.

- **Espera respuesta** “ok” o “error” del modem, en caso de “error” regresa al caso anterior.

-**Abre el socket** de control para sesión FTP.

- Comprueba si se abrió el socket, en caso contrario regresa al caso anterior.

-**Envía** datos para ingresar a la cuenta FTP.

-**Confirma** que el servidor responda “ok” o “error” a la petición de LOGIN. De responder “error” regresa al caso anterior.

-**Envía** el password al servidor FTP.

-**Confirma** que el servidor responda “ok” o “error” al password. En caso de “error” regresa al caso anterior.

-**Cambia** el formato de datos del servidor de ASCII a Binario.

- **Espera respuesta** “ok” o “error” del servidor, en caso de “error” regresa al caso anterior.

-**Hace la petición** de sesión FTP en modo pasivo.

-**Confirma** que el servidor responda “ok” o “error” a la petición. En caso de “error” regresa al caso anterior.

-**Envía petición** de conexión de datos al servidor FTP.

-**Comprueba** si el modem encontró el servidor. Si no, regresa al caso anterior.

-**Abre el socket** de conexión de datos en el servidor FTP.

-**Espera respuesta** “ok” o “error” del servidor a la petición anterior. En caso de “error” regresa al caso anterior.

-**Envío** de datos GPRS.

-Se **realiza** la petición para almacenar archivos en servidor FTP.

-**Espera** la cadena del modem. De no recibir cadena regresa al caso anterior.

-**Prepara** las variables para el envío de datos.

-**Comienza** petición de envío de datos con la cadena.

    -Si hay más de 1000 datos en la memoria, se requieren varias llamadas a la función.

-**Espera** a que el modem responda correctamente, de no ser así regresa al envío de datos.

-**Envía** los datos byte por byte.

-**Comprueba** si el modem aceptó los datos.

    -Si **existe** un error en el envío, reinicia el apuntador al valor previo del envío y regresa a la función Envío de datos GPRS.

-Si se **envían** todos los datos, se cierra el socket de datos.

-**Revisa** que cierre el socket de datos y que éstos sean correctos, de lo contrario regresa al caso anterior.

-**Cierra** sesión FTP.

-**Prueba** que cierre el socket de control, de no ser así regresa al caso anterior.

-**Revisa** que los datos se enviaron de forma correcta.

-**Desconecta** el modem GPRS de la red.

5.2.4 TAREA MEMORIA FRAM.

Esta tarea básicamente revisa el número de datos almacenados en la memoria, si éste es mayor a 15000 datos indica al usuario que la memoria está al 50% de su capacidad, si hay más de 30000 datos guardados, indica al usuario que la memoria está a un 95% de su capacidad y bloquea la escritura de la misma. A continuación tenemos el pseudocódigo de esta tarea.

-**Define** comandos para la memoria FRAM.

-**Declaración** de funciones para lectura, escritura y habilitación de la memoria FRAM.

-**Declaración** de variables locales.

-**Configuración** del dsPIC, para protocolo SPI.

-Pin RD2 del dsPIC controla pin /CS de FRAM.

-Trabaja a un 1MHZ.

-**Tarea para borrar** FRAM, e inicializar valores de apuntadores.

-**Desbloquea** FRAM en caso de estar protegida.

-**Habilita** la escritura FRAM.

-**Activa** memoria FRAM.

-**Escribe** comando para escritura en memoria.

-**Apunta** a la localidad de memoria 0x0000 (2 registros de 8 bits).

-**Desactiva** memoria FRAM.

-**Escribe** los apuntadores de la primera localidad de memoria vacía.

-**Borra** longitud de nombre de archivo.

-**Tarea para grabar** datos en la FRAM.

-Si desea **guardar** el nombre del archivo.

-Los datos los **guarda** a partir del msb.

-Se **guarda** el tamaño del nombre.

-Si desea **guardar datos**.

-**Realiza** la lectura de la localidad de memoria para guardar los datos.

-**Escribe** datos en la memoria.

-**Habilita** la escritura a FRAM.

-**Activa** memoria FRAM.

-**Escribe** comando para escritura en memoria.

-**Apunta** a la localidad de memoria vacía conformada por msb y lsb. (2 registros de 8 bits).

-**Escribe** todos los datos en la FRAM.

-Se **elimina** el comando ATN de la cadena de datos.

-**Incrementa** apuntador de memoria vacía.

-**Genera** los 2 bytes que se almacenarán como apuntadores a memoria vacía.

-Tarea para **escribir** a una localidad de memoria específica en la FRAM.

-**Habilita** la escritura a FRAM.

-**Activa** memoria FRAM.

-**Envía** comando para escritura en memoria.

-**Apunta** a la localidad de memoria.

-**Escribe** la información en el banco de memoria.

-**Desactiva** memoria FRAM.

-Tarea para **enviar** datos a la memoria FRAM.

-**Limpia** bandera.

-**Escribe** comando a la memoria.

-**Espera** a que se transmitan los datos.

-**Limpia** bandera y buffer.

-Tarea para **habilitar o deshabilitar** escritura.

-**Activa** memoria FRAM.

-**Escribe** comando para habilitar escritura.

-**Desactiva** memoria FRAM.

-Tarea para **leer** un byte de la localidad de memoria especificada.

-**Habilita** memoria.

-**Escribe** comando para la lectura de la memoria.

-**Apunta** a la localidad de memoria a leer.

-**Escribe** ciclos de reloj para leer la información.

-**Deshabilita** memoria.

-Tarea para **leer** el registro STATUS de la memoria FRAM.

-**Activa** memoria FRAM.

-**Escribe** comando para leer registro.

-**Escribe** ciclos de reloj para leer la información.

-**Desactiva** memoria FRAM.

-Tarea para **escribir** en el registro STATUS de la memoria FRAM.

-**Activa** memoria FRAM.

-**Escribe** comando para escritura en STATUS de memoria.

-**Escribe** dato.

-**Desactiva** memoria FRAM.

-Tarea para **revisar** cantidad de memoria utilizada.

-**Declaración** de variables.

-**Revisa** si la memoria se encuentra al 50% de su capacidad.

-**Revisa** si la memoria se encuentra a un 95% de su capacidad, de ser así, la protege contra escritura.

5.2.5 TAREA LCD.

La TAREA LCD, es utilizada en las tareas USART1, USART2 y memoria FRAM, sirve básicamente para desplegar la información en el LCD cada vez que es necesario imprimir el estado del sistema. En el pseudocódigo de las tareas anteriores se menciona dónde es utilizado el LCD, por lo tanto no es necesario colocar un pseudocódigo específico.

6 PRUEBAS REALIZADAS.

6.1 INTRODUCCIÓN.

Las diferentes pruebas realizadas al módulo, permiten asegurar cierta confiabilidad y detectar las posibles fallas del mismo. Debido a que el objetivo de este módulo es ser instalado en lugares remotos y en algunas ocasiones de difícil acceso y tomando en cuenta la información que maneja, su correcto funcionamiento es crítico. Así que este capítulo es fundamental para el desarrollo del sistema.

6.2 MEDICIÓN DE VOLTAJE Y TRANSMISIÓN DE LA INFORMACIÓN.

La primera prueba realizada del módulo GPRS consistió en conectarlo a otro módulo sensor de voltaje, con el objetivo de comprobar el funcionamiento del módulo GSM-GPRS, tanto en la recepción de los datos como en el envío y el almacenamiento de los mismos. Éste se muestra en la Figura 32.

Figura 32: Módulo sensor de voltaje.

Este módulo sensor de voltaje contiene:

- 2 potenciómetros.
- 1 microcontrolador 16F877A.
- 2 botones.
- 1 display.

También se observa la herramienta Pic kit 2 utilizada para programar los microcontroladores.

Los potenciómetros tienen el objetivo de variar el voltaje, estos voltajes entran al módulo sensor de voltaje por el convertidor analógico-digital del microcontrolador 16F877A. Éste hace un procesamiento digital de las señales de voltaje, estos valores se muestran en el display del módulo sensor. Los datos adquiridos por el módulo sensor son enviados al módulo GSM-GPRS por medio del puerto USART del microcontrolador 16F877A, y a su vez adquiridos por el puerto USART 1 del microcontrolador dspic18F4550; el conector para esta comunicación se muestra en la Figura 33.

Cuando se encienden los módulos, los datos adquiridos por el módulo sensor son enviados al módulo GPRS, éste los almacena en la memoria FRAM. Durante este proceso, el display del módulo GPRS está indicando que se están almacenando los datos con los siguientes comandos desplegados en su display:

- ATG--->Data saved
- ATG---->
- ATT---->OK

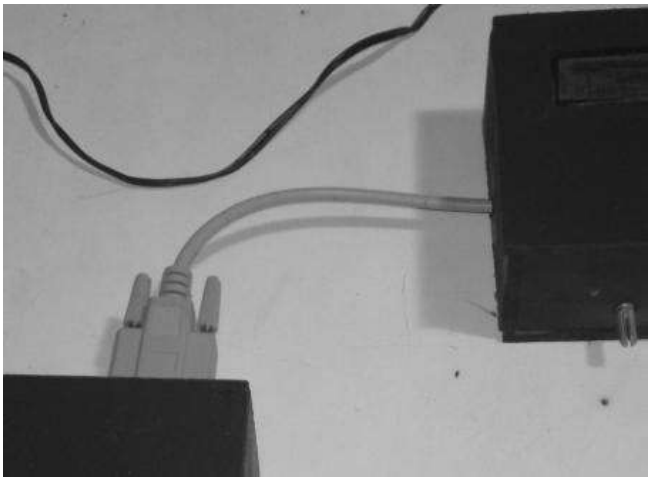


Figura 33: Comunicación entre los módulos.

En la Figura 34 se observan los comandos desplegados en el display del módulo GPRS.

Los botones del módulo sensor, fueron etiquetados como DATOS y ENVÍO; al presionar el botón DATOS, el display del módulo sensor muestra el número de datos adquiridos hasta el momento de presionar el botón. En la Figura 35, se observa el display cuando se presionó el botón DATOS en una de las pruebas.

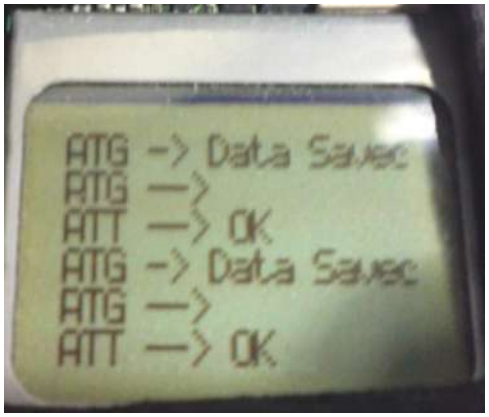


Figura 34: Display del módulo transmisor.

Figura 35: Display del módulo de adquisición de datos.

Al presionar el botón ENVÍO, el módulo GPRS envía los datos almacenados en la memoria FRAM al servidor de Internet. En el display del módulo se indica al usuario que se está realizando el envío de los datos. En la Figura 36 se muestra el mensaje que presenta el display mientras se envían los datos al servidor remoto.

En la Figura 37 se muestran los módulos conectados y funcionando. En la parte inferior izquierda se encuentra el módulo GPRS, en la parte superior derecha se encuentra el módulo sensor de voltaje.

Figura 36: Envío de datos al servidor.

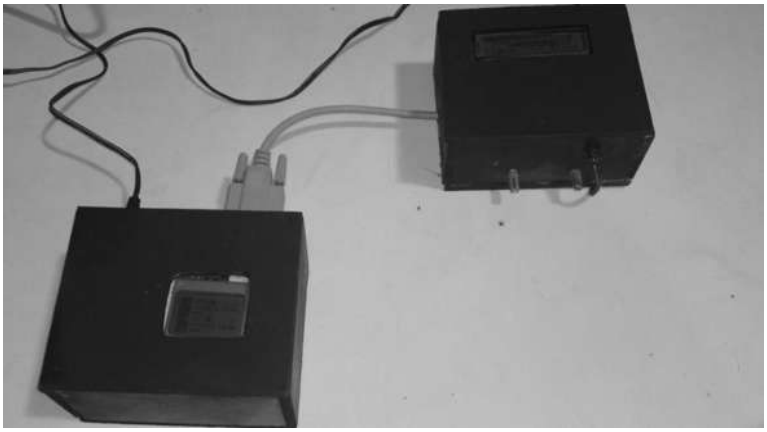


Figura 37: Módulos GSM-GPRS y sensor funcionando.

Se generó una página WEB, para poder visualizar los datos. La dirección de esta página es: http://www.micro.fie.umich.mx/, en ésta se encuentran varias ligas, la liga de interés para nosotros se llama "revisar datos y graficarlos"; si se accede a este link aparece una serie de nuevas ligas en las cuales se encuentran los archivos generados por los datos enviados por el módulo GPRS al servidor web. Si accedemos a la liga COECYT_29_10_2010_1m_05, nos muestra la gráfica de la Figura 38, la cual contiene los datos adquiridos y enviados por nuestro módulo GPRS en una de las pruebas realizadas.

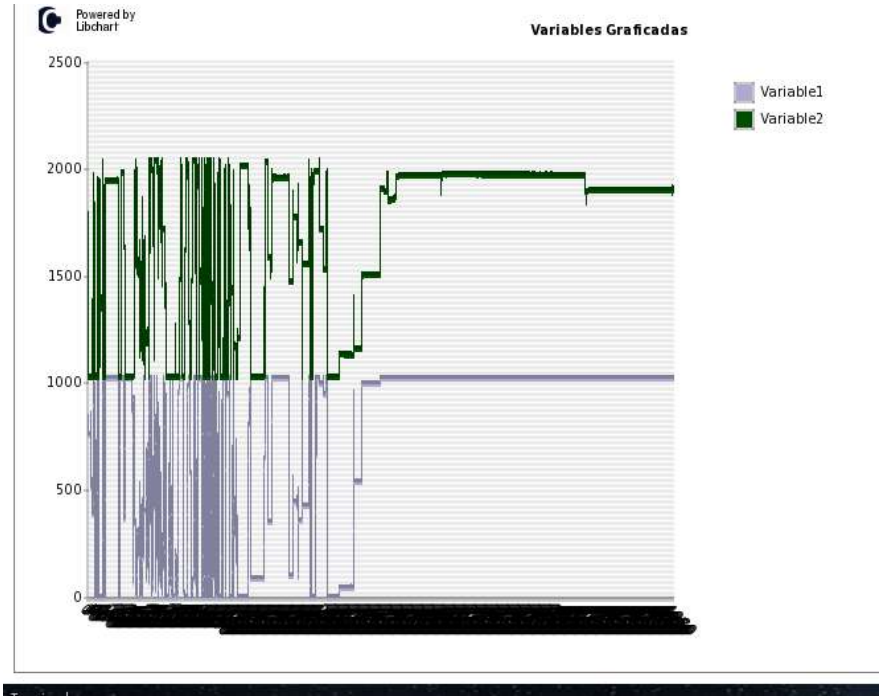


Figura 38: Datos adquiridos y mostrados en el servidor html.

En la parte superior de la gráfica de la Figura 38 existe la opción de mostrar datos, si ésta es seleccionada, despliega una tabla con las mediciones adquiridas y enviadas al servidor.

6.3 MEDICIÓN DE HUMEDAD Y TEMPERATURA Y ENVÍO DE LA INFORMACIÓN.

La segunda prueba realizada al módulo GPRS, consiste en conectarlo a un módulo sensor de temperatura y humedad, el objetivo es comprobar el funcionamiento del módulo GSM-GPRS con una aplicación más cercana a la aplicación final que tendrá el módulo. A diferencia del módulo sensor anterior, éste cuenta con sensores de humedad y temperatura conectados al microcontrolador. Este módulo está compuesto por:

- 2 Botones
- 1 Sensor de humedad y temperatura modelo: HMZ433A1.
- 1 microcontrolador 16F877A.
- 1 Display alfanumérico LCD.

Al energizar ambos módulos, el módulo sensor inicia con la adquisición de los datos de los sensores de humedad y temperatura, y los envía al módulo GPRS. En la Figura 39, se pueden observar los módulos conectados. En la parte izquierda el módulo sensor de humedad y temperatura, en la parte derecha el módulo GPRS.

Los botones del módulo sensor se etiquetaron como Envío y Visualización. Al presionar el botón Visualización, el display muestra la información de cuántos datos se han enviado y almacenado en el módulo "Transmisor de información" hasta ese momento. Al presionar el botón Envío, se da la orden al módulo "Transmisor de información" de enviar los datos al servidor web, y el display indica que los datos se están enviando al servidor. El

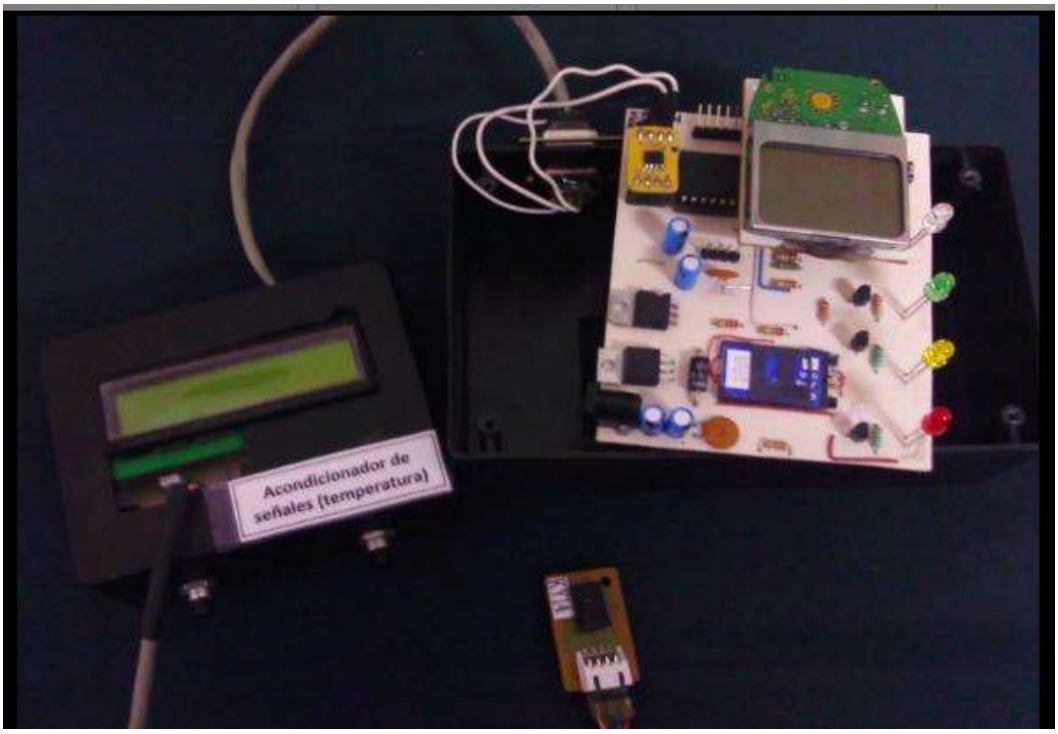


Figura 39: Segunda prueba realizada.

sensor HMZ433A1, tiene la función de convertir las variables físicas (humedad y temperatura) en variables eléctricas. La función del microcontrolador es básicamente la conexión serie entre los módulos.

Los archivos enviados al servidor pueden ser vistos desde una página web. En la Figura 40, se puede ver uno de los archivos enviados, los datos son mostrados en forma gráfica. Los datos también pueden ser vistos en una tabla como se muestra en la Figura 41.

Esta prueba fue muy importante ya que se detectó una falla en el módulo, la cual no tuvo un gran impacto en el funcionamiento del mismo, solamente afectó la interfaz con el usuario.

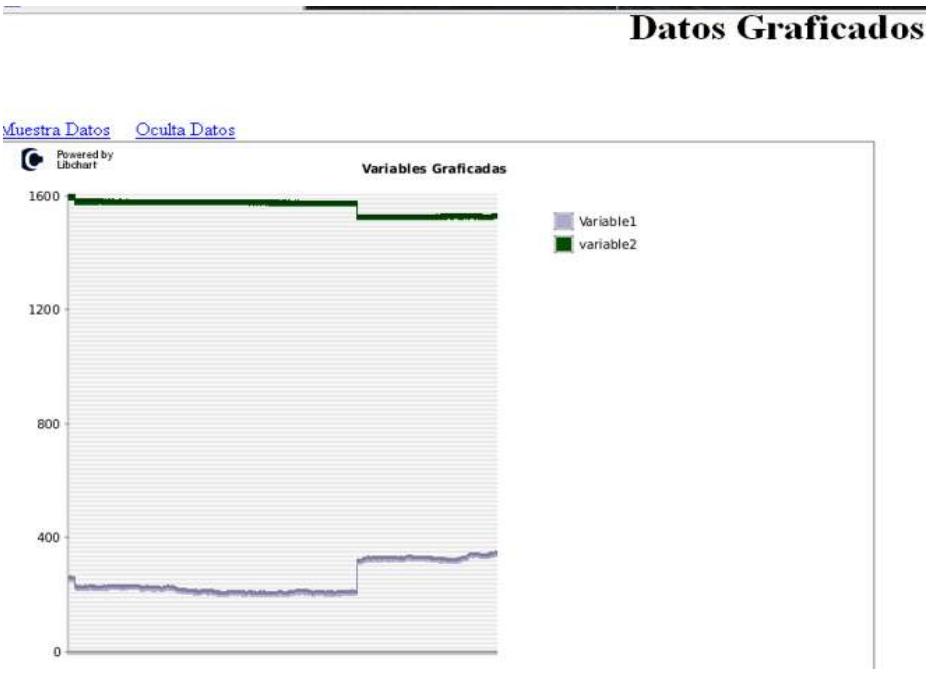
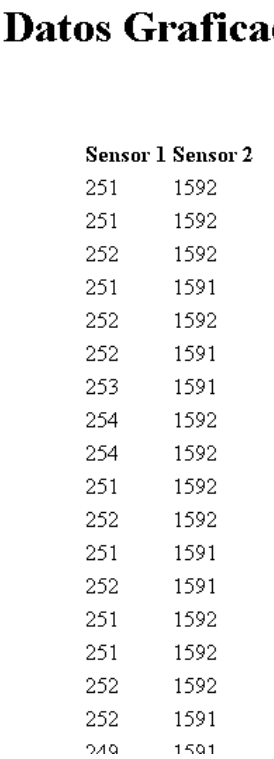


Figura 40: Gráfica de datos enviados al servidor.



Datos Graficados

Sensor 1	Sensor 2
251	1592
251	1592
252	1592
251	1591
252	1592
252	1591
253	1591
254	1592
254	1592
251	1592
252	1592
251	1591
252	1591
251	1592
251	1592
252	1592
252	1591
249	1591

Figura 41: Tabla parcial de los datos enviados al servidor.

6.4 REPORTE DE FALLAS DURANTE PRUEBAS DE CONFIABILIDAD.

En el software se estableció un bloqueo en la escritura de la memoria FRAM, el cual consiste en que cuando ésta llega a un 95% de su capacidad, no permite que se continúen escribiendo datos en ella. Este bloqueo falló, lo que ocasionó que se corrompieran las localidades de memoria reservadas para el código ASCII utilizado por el display para informar el estado del sistema. Después de esto, el display solo desplegaba ruidos en lugar de caracteres ASCII, esta falla no tuvo una consecuencia en el envío y la recepción de los datos. Esta falla en el firmware fue depurada de manera correcta, se siguieron haciendo pruebas para comprobarlo y el sistema funcionó correctamente.

7 CONCLUSIONES Y TRABAJOS FUTUROS.

7.1 CONCLUSIONES.

El servicio GPRS que ofrece la infraestructura de red celular GSM nos permite transmitir información a un costo muy accesible. Actualmente el costo de envío es de 1 peso por cada megabyte de información enviado al servidor. Es un medio de comunicación muy confiable dado que los estándares que maneja, son muy similares a los de las redes de Internet, solo que acondicionados para medios inalámbricos.

El módulo tiene un tiempo de conexión y desconexión variable, ya que estos tiempos dependen de la red GSM. Si al momento que el módulo envía los datos, la red se encuentra saturada debido a que está siendo utilizada por otros usuarios, puede demorar unos minutos para tener una conexión con el servidor, o inclusive no tener éxito. Se estableció un periodo de tiempo para establecer la conexión, de hasta 15 minutos, Si pasado este tiempo el modem no consigue tener conexión con el servidor, el sistema cierra todo enlace existente y avisa al usuario. También se estableció un rango de tiempo para desconectar al módulo del servidor, hasta un máximo de 5 minutos.

El hardware del módulo tiene un costo aproximado de 1600.00 pesos, resaltando que el modem GPRS es el 70% del costo total. Su peso es de 300 gr. El tiempo de desarrollo fue de 5 meses aproximadamente. El módulo es un prototipo único, ya que el modem GPRS utilizado ya no se encuentra en el mercado, lo que obliga a realizar otro diseño con el mismo funcionamiento pero adaptado al nuevo modem.

Resumen de características técnicas del prototipo.

1- Conexión serie con el cliente.
   Capacidad para almacenar y enviar datos.
2- 1 página web para monitorear 2 variables físicas.

3- Enlace GSM-GPRS a Internet.
4- Protocolo del cliente FTP.
5- Conexión al servidor de FTP.
6- Display LCD para visualizar estado de conexión (cliente-módulo-servidor) y características de los datos.
7- Capacidad para almacenamiento de datos de hasta 30 Kbyte.
8- El consumo de energía en modo de espera y recepción de datos es de 2.5 watts hora y en modo transmisión de datos es de 15 watt-hora.

El módulo tiene algunas características que es importante resaltar, cuenta con un puerto serie RS232, el cual está configurado con una velocidad de 9600bps. Se decidió utilizar esta velocidad ya que tiene que ser menor que la velocidad de transmisión del modem GPRS que es de 21400 bps, cuenta también con 8 bits de datos y un bit de parada, no cuenta con paridad ni control de hardware. Para que el cliente pueda enviar datos al módulo, éste debe de confirmar que se enviaron los datos anteriores. Para que la página web funcione correctamente el cliente debe proporcionar un nombre de archivo, así como la longitud del mismo.

7.2 TRABAJOS FUTUROS.

En un futuro se pretende que el módulo cuente con las siguientes características adicionales:

- Tenga un control de flujo de información bidireccional, esto es, que pueda realizar acciones de control en un sistema, como encendido de ventiladores, focos, alarmas etc. Ya sea por vía Internet, GPRS, de manera local, o con un mensaje de texto SMS.

- Envíe señales de alarma a un equipo móvil celular.

- Sea autónomo, acoplando una celda solar y una batería.

- Reemplazar el modem GPRS por otro con programación en Pitón (debido a que el actual modem está descontinuado).

- Posiblemente utilizar EDGE o 3G.

- Adecuar el prototipo para su uso a la intemperie.

APÉNDICES.

A. TABLAS DE COMANDOS FTP.

Comandos de control de acceso	
Comando	Descripción
USER	Cadena de caracteres que permite identificar al usuario. La identificación del usuario es necesaria para establecer la comunicación a través del canal de datos.
PASS	Cadena de caracteres que especifica la contraseña del usuario. Este comando debe ser inmediatamente precedida por el comando USER. El cliente debe decidir si esconder la visualización de este comando por razones de seguridad.
ACCT	Cadena de caracteres que especifica la cuenta del usuario. El comando ente no es necesario. Durante la respuesta que acepta la contraseña, si uesta es 230, esta etapa no es necesaria; Si la respuesta es 332, sí lo es.
CWD	Change Working Directory (Cambiar el directorio de trabajo): este comando permite cambiar el directorio actual. Este comando requiere la ruta de acceso al directorio para que se complete como un argumento.
CDUP	Parent Directory (Cambiar al directorio principal): este comando resar al directorio principal. Se introdujo para resolver los problemas nación del directorio principal según el sistema.
SMNT	Structure Mount (Montar estructura):
REIN	Reinitialize (Reinicializar):
QUIT	Comando que permite abandonar la sesión actual. Si es necesario, el servidor espera a que finalice la transferencia en progreso y después proporciona una respuesta antes de cerrar la conexión.

Comandos de parámetros de transferencia	
Comando	Descripción
PORT	Cadena de caracteres que permite especificar el número de puerto utilizado.
PASV	Comando que permite indicar al servidor de DTP que permanezca a la espera de una conexión en un puerto específico elegido aleatoriamente entre los puertos disponibles. La respuesta a este comando es la dirección IP del equipo y el puerto.
TYPE	Este comando permite especificar el tipo de formato en el cual se enviarán los datos (binario o ASCII).
STRU	Carácter Telnet que especifica la estructura de archivos (F de File [Archivo], R de Record [Registro], P de Page [Página]).
MODE	Carácter Telnet que especifica el método de transferencia de datos (S de Stream [Flujo], B de Block [Bloque], C de Compressed [Comprimido]).

Comandos de servicio FTP	
Comando	Descripción
RETR	Este comando (RETRIEVE [RECUPERAR]) le pide al servidor de DTP una copia del archivo cuya ruta de acceso se da en los parámetros.
STOR	Este comando (store [almacenar]) le pide al servidor de DTP que acepte los datos enviados por el canal de datos y que los almacene en un archivo que lleve el nombre que se da en los parámetros. Si el archivo no existe, el servidor lo crea; de lo contrario, lo sobrescribe.
STOU	Este comando es idéntico al anterior, sólo le pide al servidor que cree un archivo cuyo nombre sea único. El nombre del archivo se envía en la respuesta.
APPE	Gracias a este comando (append [adjuntar]) los datos enviados se concatenan en el archivo que lleva el nombre dado en el parámetro si ya existe; si no es así, se crea.

ALLO	Este comando (allocate [reservar]) le pide al servidor que reserve un espacio de almacenamiento lo suficientemente grande como para recibir el archivo cuyo nombre se da en el argumento.
REST	Este comando (restart [reiniciar]) permite que se reinicie una transferencia desde donde se detuvo. Para hacer esto, el comando envía en el parámetro el marcador que representa la posición en el archivo donde la transferencia se había interrumpido. Después de este comando se debe enviar inmediatamente un comando de transferencia.
RNFR	Este comando (rename from [renombrar desde]) permite volver a nombrar un archivo. En los parámetros indica el nombre del archivo que se va a renombrar y debe estar inmediatamente seguido por el comando RNTO.
RNTO	Este comando (rename from [renombrar a]) permite volver a nombrar un archivo. En los parámetros indica el nombre del archivo que se va a renombrar y debe estar inmediatamente seguido por el comando RNFR.
ABOR	Este comando (abort [cancelar]) le indica al servidor de DTP que abandone todas las transferencias asociadas con el comando previo. Si no hay conexión de datos abierta, el servidor de DTP no realiza ninguna acción; de lo contrario, cierra la conexión. Sin embargo, el canal de control permanece abierto.
DELE	Este comando (delete [borrar]) permite que se borre un archivo, cuyo nombre se da en los parámetros. Este comando es irreversible y la confirmación sólo puede darse a nivel cliente.
RMD	Este comando (remove directory [eliminar directorio]) permite borrar un directorio. El nombre del directorio que se va a borrar se indica en los parámetros.
MKD	Este comando (make directory [crear directorio]) permite crear un directorio. El nombre del directorio que se va a crear se indica en los parámetros.
PWD	Este comando (print working directory [mostrar el directorio actual]) hace posible volver a enviar la ruta del directorio actual completa.

LIST	Este comando permite que se vuelva a enviar la lista de archivos y directorios presentes en el directorio actual. Esto se envía a través del DTP pasivo. Es posible indicar un nombre de directorio en el parámetro de este comando. El servidor de DTP enviará la lista de archivos del directorio ubicado en el parámetro.
NLST	Este comando (name list [lista de nombres]) permite enviar la lista de archivos y directorios presentes en el directorio actual.
SITE	Este comando (site parameters [parámetros del sistema]) hace que el servidor proporcione servicios específicos no definidos en el protocolo FTP.
SYST	Este comando (system [sistema]) permite el envío de información acerca del servidor remoto.
STAT	Este comando (Estado: [estado]) permite transmitir el estado del servidor; por ejemplo, permite conocer el progreso de una transferencia actual. Este comando acepta una ruta de acceso en el argumento y después devuelve la misma información que LISTA pero a través del canal de control.
HELP	Este comando permite conocer todos los comandos que el servidor comprende. La información se devuelve por el canal de control.
NOOP	Este comando (no operations [no operación]) sólo se utiliza para recibir un comando OK del servidor. Sólo se puede utilizar para no desconectarse después de un período de inactividad prolongado.

B. COMANDOS ATE IMPLEMENTADOS EN EL MODULO GSM-GPRS.

- **ATT:** ***Comando para comprobar conexión.***

- **ATN XXX...XXXX:** ***indica el nombre de archivo en memoria FRAM (max.60 caracteres).***

- **ATG XX:** ***Graba xx datos en FRAM (max. 90 bytes por comando).***

    - **Donde xx es un número entre 00-90.**

- **ATE:** ***Envía los datos en FRAM al servidor FTP.***

- **EFRAM:** ***Borra la memoria.***

C. COMANDOS AT DEL MODEM GPRS.

- **At +ipconfig:** ***Este comando configura varios aspectos de la interfaz de sockets de AT como son:***
  - *Type.*
  - *Multi-socket mode.*
  - *IPSEND.*
  - *IPDATA .*
  - *IPDATA subformat.*
  - *IPSENDB timeout*:

- **At +netstart.** ***Este comando conecta el modem a la red GPRS.***

- **At +ipdest.** ***Este comando configura las direcciones de destino para el socket del comando abierto.***

- **At +ipopen.** ***Abre un socket y regresa un "ok" una ves que se ha establecido la conexión .***

- **At +ipsend.** ***Envía los datos al socket abierto, puede enviar una cadena de ASCII hasta de 256 caracteres.***

- **At +netstop.** ***Desconecta el modem de la red GPRS.***

- **At +ipsendb:** ***Envía datos binarios al socket abierto puede mandar hasta 1024 bytes.***

- **At +ipclose:** ***Cierra el socket abierto.***

D. PLATAFORMAS UTILIZADAS.

- Para las siguientes aplicaciones se utilizó el sistema operativo Windows.
    - Para el software implementado en el microcontrolador se utilizo MPLAB IDE y el compilador C30.
    - Para la depuración del módulo se requirió un analizador del puerto serie SERIAL SNIFFER.
    - Para corroborar que la comunicación entre el módem y el microcontrolador fuera correcta se utilizo el programa PORTMON.
    - Para el diseñó del circuito impreso se utilizo el software EAGLE de la empresa CADSOFT.
- Para las siguientes aplicaciones se utilizó el sistema operativo Linux.
    - Para realizar la pagina web que muestra la información enviada al servidor se utilizo el interprete PHP.
    - Se utilizo el servidor web Apache por ser uno de los más completo, el que mejor se integra con el intérprete PHP y el sistema operativo Linux.
    - Se utilizo un servidor FTP gratuito para almacenarlos datos en el servidor.

E. SOFTWARE.

A continuación, se presenta el programa del microcontrolador.

main.c:

```
#include<p30F4013.h>

unsigned char F_Envia_Datos;
unsigned int m_seg;
unsigned char min;


int main(void)
{
	F_Envia_Datos=0;

	_TRISF1=0;

	Config_Lcdnokia();	//inicializa el display grafico
	Config_Fram();		//inicializa la memoria
	Config_Usart1();	//inicializa el puerto serie 1
	Clean_Usart1(0);	//limpia variables del puerto serie 1
	Config_Usart2();	//inicializa el puerto serie 2
	Clean_Usart2();		//limpia variables del puerto serie 1
	Config_Timer3();


	Escribe_Lcd("");
	_LATF1=1;

	while(1)
	{
		Tarea_Usart1();
		Tarea_Usart2();
```

```
                if(min==5)
                {
                        _LATF1=~_LATF1;
                        Tarea_Fram();
                        min=0;
                }
        }
}

void Config_Timer3()
{
        IFS0bits.T3IF=0;
        IEC0bits.T3IE=1;
        IPC1bits.T3IP=1;
        TMR3=57866;
        T3CON=0x8000;
        m_seg=0;min=0;
}

void __attribute__((__interrupt__)) _T3Interrupt(void)
{

        TMR3=57866;
        m_seg++;

        if(m_seg==60000)
        {
                m_seg=0;min++;
        }

        IFS0bits.T3IF=0;
}
```

Usart1.c

```
//          PUERTO SERIE 1
//C_usart.- contador de caracteres 0x0d y 0x0a
//P_Usart1.- apuntador al buffer de la usart 1
//B_Usart1.- buffer de caracteres usart 1
//F_Usart1.- bandera de comandos/datos escritos a usar1
//T_Usart1.- bandera de tiempo de recepcion de datos
//L_Usart1.- longitud de datos a recibir en usart1

#include<p30F4013.h>

//variables y banderas locales
unsigned char C_Usart1,P_Usart1,B_Usart1[100];
unsigned char F_Usart1,L_Usart1;
const unsigned char error[]="\r\nERROR\r\n";
const unsigned char ok[]="\r\nOK\r\n";
const unsigned char newline[]="\r\n>";


//variables y banderas globales

extern unsigned char F_Envia_Datos;

void __attribute__((__interrupt__)) _U1TXInterrupt(void)
{
        IFS0bits.U1TXIF=0;
}


void __attribute__((__interrupt__)) _U1RXInterrupt(void)
{
        unsigned char trash;

        trash=U1RXREG&0xff;

        if(F_Usart1==0)
        {
                if(trash==0x0d)
```

```
		{
			C_Usart1=1;
		}
		else
		{
			B_Usart1[P_Usart1]=trash;
			P_Usart1++;
		}
	}

	else if(F_Usart1==1)
	{
		B_Usart1[P_Usart1]=trash;
		P_Usart1++;

		if(P_Usart1==L_Usart1)
		{
			C_Usart1=2;
		}
	}
	if(P_Usart1==100)
		P_Usart1=0;

	IFS0bits.U1RXIF=0;
}

void Config_Usart1(void)
{
	U1BRG=47;			//configura USART 1
	U1MODE=0x8000;			//a 9600bps =>u1brg= 47;
	U1STA=0x8400;			//con pines TX y RX alternativos
	U1MODE=0x8400;

	IFS0bits.U1TXIF = 0;		 //Limpia bandera de transmision TX1
	IEC0bits.U1TXIE = 0;		 //deshabilita interrupcion de TX1
	IFS0bits.U1RXIF = 0;		 //Limpia bandera de interrupcion RX1
	IEC0bits.U1RXIE = 1;		 //habilita interrupcion RX1
	IPC2bits.U1TXIP = 1;		//prioridad de interrupcion #1 para TX
```

```
	IPC2bits.U1RXIP = 7;			//prioridad de interrupcion #7 para RX mas alta...
}

void WriteStr_Usart1(const unsigned char string[ ])
{
	unsigned char i=0;
	while(1)
	{
		while(U1STAbits.UTXBF);
		U1TXREG=string[i];
		i++;
		if(string[i]==0x00)
			break;
	}
}


void WriteChr_Usart1(unsigned char letra)
{
	while(U1STAbits.UTXBF);
	U1TXREG=letra;
}

void Clean_Usart1(unsigned char data_command)
{
	unsigned char i;

	C_Usart1=0;
	P_Usart1=0;
	for(i=0;i<100;i++)
	{	B_Usart1[i]=0;	}

	switch(data_command)
	{
		case 0:
			F_Usart1=0;
			L_Usart1=0;//limpia variable de long. datos
			break;
```

```
                case 1:
                        F_Usart1=1;
                        break;

                default:
                        F_Usart1=0;
                        L_Usart1=0;
                        break;
        }
}

void Tarea_Usart1(void)
{
        unsigned char i;
        unsigned int x;

        if((F_Usart1==0)&&(C_Usart1==1))
        {
                if((B_Usart1[0]=='A')&&(B_Usart1[1]=='T'))
                {
                        switch(B_Usart1[2])
                        {
                                case 'T':
                                        Clean_Usart1(0);
                                        Escribe_Lcd("ATT --> OK");
                                        WriteStr_Usart1(ok);
                                        break;

                                case 'N':
                                        if((P_Usart1<=5)||(P_Usart1>65))
                                        {
                                                Clean_Usart1(0);
                                                Escribe_Lcd("ATN --> ERROR");
                                                WriteStr_Usart1(error);
                                        }

                                        else
```

```
                {

                        SaveData_Fram(B_Usart1,4,P_Usart1,0);
                        Clean_Usart1(0);
                        Escribe_Lcd("ATN --> OK");
                        WriteStr_Usart1(ok);
                }
                break;

        case 'G':

if((B_Usart1[3]==0x20)&&(B_Usart1[4]>=0x30)&&(B_Usart1[4]<=0x38)&&
(B_Usart1[5]>=0x30)&&(B_Usart1[5]<=0x39))

        {
                L_Usart1=10*(B_Usart1[4]-0x30)+(B_Usart1[5]-0x30);
                Clean_Usart1(1);
                Escribe_Lcd("ATG -->");
                WriteStr_Usart1(newline)
        }
                        else
                        {
                                Clean_Usart1(0);
                                Escribe_Lcd("Command Error");
                                WriteStr_Usart1(error);
                        }
                        break;

        case 'E':
                        if(F_Envia_Datos==0)
                        F_Envia_Datos=1;

                        Clean_Usart1(0);
                        Escribe_Lcd("ATE > Send");
                        WriteStr_Usart1("\r\nSending..\r\n");
                        break;

        default:
```

```
                                Clean_Usart1(0);
                                Escribe_Lcd("Command Error");
                                WriteStr_Usart1(error);//envia OK
                                break;
                        }
                }

                else if((B_Usart1[0]=='E')&&(B_Usart1[1]=='F')&&(B_Usart1[2]=='R')&&
                (B_Usart1[3]=='A')&&(B_Usart1[4]=='M'))

                {
                        Erase_Fram();
                        Clean_Usart1(0);
                        Escribe_Lcd("\r\nERASED FRAM\r\n");
                        WriteStr_Usart1(ok);
                }

                        else
                {
                        WriteStr_Usart1(error);
                        Escribe_Lcd("Command Error");
                        Clean_Usart1(0);
                }

}

                else if((F_Usart1==1)&&(C_Usart1==2))
{
                //el buffer B_Usart1 contiene los datos a guardar en la FRAM
                //comienzan en la localidad 0x00, y la longitud de los datos
                //esta almacenada en L_Usart1.

                //0 es el offset de los datos, como inician en 0x00 no debe existir ofsset
                //L_Usart1 es el numero de datos a guardar
                //1 indica que se van a guardar datos en la FRAM

                SaveData_Fram(B_Usart1,0,L_Usart1,1);
                Clean_Usart1(0);//limpia variables para esperar comando
```

```
                Escribe_Lcd("ATG -> Data Saved");
                WriteStr_Usart1(ok);//envia OK
        }
        if(F_Envia_Datos==3)
        {
                WriteStr_Usart1("\n\rATE OK\r\n");
                Erase_Fram();
                F_Envia_Datos=0;
        }

        if(F_Envia_Datos==10)
        {
                WriteStr_Usart1("\r\nATE ERROR\r\n");
                F_Envia_Datos=0;
        }
}
```

Usart2.c

```
#include<p30F4013.h>

// constantes locales del puerto serie
#define enter 0x0d

//constantes para el modem gprs
const unsigned char at[]="at";
const unsigned char server[]="+netapn=\"internet.itelcel.com\",\"webgprs\",\"webgprs2002\"";
const unsigned char ipconfig[]="+ipconfig=0,2,0,1,1,0";
const unsigned char netstart[]="+netstart";
const unsigned char ipdest[]="+ipdest=\"ftp.micro.x10.mx\"";
const unsigned char ipopen[]="+ipopen";
const unsigned char ipsend[]="+ipsend=";
const unsigned char netstop[]="+netstop";
const unsigned char ipsendb[]="+ipsendb=";
const unsigned char ipclose[]="+ipclose=";
```

```
//constantes para el servidor FTP
const unsigned char append[]="\"appe ";
const unsigned char user[]="\"user 1@micro.x10.mx\\n\"";
const unsigned char pass[]="\"pass isra.001\\n\"";
const unsigned char type[]="\"type i\\n\"";
const unsigned char pasv[]="\"pasv\\n\"";
const unsigned char quit[]="\"quit\\n\"";
const unsigned char S_FtpC[]={0x32,0x31,0x00};          //socket de conexion de control


//variables locales del puerto serie
unsigned char F_Conexion,trash;                 //bandera de tarea de usart2
unsigned char C_Enters;                         //contador de caracteres de fin de linea (0x0d 0x0a)
unsigned char B_Usart2[100];                    //buffer de recepcion de datos del modem
unsigned char C_Comm[3],C_Data[3];              //conexion abierta de datos y control para ftp
unsigned char S_FtpD[6];                        //socket de conexion de datos
unsigned char P_Usart2;                         //apuntador para buffer de recepcion de datos
unsigned int P_Data;                            //primer localidad de informacion en Fram
unsigned int P_Aux;                             //apuntador auxiliar de P_Data
unsigned int P_Fram;                            //apuntador a la memoriaauxiliar de P_Data
unsigned int N_Data;                            //numero de datos a enviar
unsigned char B_Alt[5];                         //buffer alterno para datos
unsigned int m_segundos;                        //banderas para contar tiempo
unsigned char minutos;


//variables globales externas
extern unsigned char F_Envia_Datos;

//definicion de funciones
void Config_Timer1(void);
void Clean_Usart2(void);
void Tarea_Usart2(void);
void Config_Usart2(void);
void Write_Usart2(unsigned char);
void WriteStr_Usart2(const unsigned char[]);
void Sockets_Locales(unsigned char);
```

```
unsigned int Scan_String(unsigned char,unsigned char);
void __attribute__((__interrupt__)) _U2TXInterrupt(void);
void __attribute__((__interrupt__)) _U2RXInterrupt(void);
void __attribute__((__interrupt__)) _T1Interrupt(void);

int main(void)
{
	Config_Usart2();
	Config_Lcdnokia();
	Config_Fram();
	Clean_Usart2();
	F_Envia_Datos=1;

	Escribe_Lcd("hola chinitoo!!");
	while(1)
	{
		Tarea_Usart2();
	}
}

void Config_Timer1()
{
	IFS0bits.T1IF=0;
	IEC0bits.T1IE=1;
	IPC0bits.T1IP=1;
	TMR1=57866;//para contar 1ms
	T1CON=0x8000;
	m_segundos=0;minutos=0;
}

void __attribute__((__interrupt__)) _T1Interrupt(void)
{

	TMR1=57866;
	m_segundos++;

	if(m_segundos==60000)
	{	m_segundos=0;minutos++;}
```

```
		if(minutos==15)
		{
			F_Conexion=50;
			m_segundos=0;minutos=0;
			F_Envia_Datos=2;
			T1CON=0x00;
		}

		IFS0bits.T1IF=0;
}

void __attribute__((__interrupt__)) _U2TXInterrupt(void)
{
		IFS1bits.U2TXIF=0;
}

void __attribute__((__interrupt__)) _U2RXInterrupt(void)
{
		trash=U2RXREG&0xff;

		if((trash==0x0d)||(trash==0x0a))
			C_Enters++;
		else
		{
			B_Usart2[P_Usart2]=trash;
			P_Usart2++;
		}

		if(P_Usart2==100)
			P_Usart2=99;

		IFS1bits.U2RXIF=0;
}

void Config_Usart2(void)
{
		U2BRG=47;				//configura USART 2
```

```
        U2MODE=0x8000;                  //a 9600 con oscilador interno
        U2STA=0x8400;                   //de 7.37MHz
        U2MODE=0x8000;                  //no tiene pins alternativos p/TX y RX

        IFS1bits.U2TXIF = 0;            //Reset TX1 interrupt flag
   IEC1bits.U2TXIE = 0;                 //Disable TX1 Interrupt Service Routine
        IFS1bits.U2RXIF = 0;            //Reset TX1 interrupt flag
   IEC1bits.U2RXIE = 1;                 //Enable TX1 Interrupt Service Routine
        IPC6bits.U2TXIP = 1;
        IPC6bits.U2RXIP = 6;
}

void WriteStr_Usart2(const unsigned char string[])
{
        unsigned char i=0;

        while(1)
        {
                while(!U2STAbits.TRMT);

                U2TXREG=string[i];
                i++;

                if(string[i]==0)
                break;
        }
}

void Write_Usart2(const unsigned char string)
{
        while(U2STAbits.UTXBF);
        U2TXREG=string;
}

void Clean_Usart2(void)
{
        unsigned char i;
```

```
	for(i=0;i<100;i++)
	{	B_Usart2[i]=0;}

	P_Usart2=0;
	C_Enters=0;

}


void Socket_FtpData(unsigned int valor)
{
	S_FtpD[0]=(valor/10000)+0x30;
	valor=valor%10000;

	S_FtpD[1]=(valor/1000)+0x30;
	valor=valor%1000;

	S_FtpD[2]=(valor/100)+0x30;
	valor=valor%100;

	S_FtpD[3]=(valor/10)+0x30;

	S_FtpD[4]=valor%10+0x30;

	S_FtpD[5]=0;
}

void Sockets_Locales(unsigned char puerto)
{
	unsigned char a=0,b=0,c[2];

	while(B_Usart2[a]!=':')
	{	a++;}

	while(B_Usart2[a]!=',')
	{
		if((B_Usart2[a]>=0x30)&&(B_Usart2[a]<=0x39))
		{	c[b]=B_Usart2[a];b++;}
```

```
            a++;
        }

        if(b==0)
        {       C_Comm[0]=0x39;C_Comm[1]=0x39;
                C_Data[0]=0x39;C_Data[1]=0x39;}

        else
        {
                switch(puerto)
                {
                        case 0:
                                C_Comm[0]=c[0];C_Comm[1]=c[1];
                                break;

                        case 1:
                                C_Data[0]=c[0];C_Data[1]=c[1];
                                break;
                        default:
                                C_Comm[0]=c[0];C_Comm[1]=c[1];
                                break;
                }
        }

        C_Comm[2]=0;C_Data[2]=0;

}

//funcion para encontrar un tipo de respuesta en el buffer Usart2
//0.- busca en el buffer la cadena "OK" con el numero de enters esperado
//1.- busca en el buffer la cadena ""+IPDATA: aa,"XXX ---------e (aaa,aaa,aaa,aaa,aaa,aaa)\r"
//2.- busca en el buffer la cadena "+IPDATA:aa,"aaa ------(aaa,aaa,aaa,aaaa,XXX,YYY)
//3.- busca en el buffer la cadena "0x0d 0x0a >"
// las respuestas son:
// 0.- numero de enters incompletos
// 1.- cadena encontrada, numero de enters completos
// 10.- cadena no encontrada, numero de enters completos
unsigned int Scan_String(unsigned char orden,unsigned char enters)
```

```
{
	unsigned char a=0,b=0;
	unsigned int x=0,y=0;

	switch(orden)
	{
		case 0:
			if(C_Enters>=enters)
			{
				while(a<100)
				{
					if((B_Usart2[a]=='O')&&(B_Usart2[a+1]=='K'))
					{x=1;break;}
					a++;
				}

				if(a==100)
					x=10;
			}

			else
				x=0;

			break;

		case 1:

			if(C_Enters>=enters)
			{
				while(a<100)
				{
					if((B_Usart2[a]=='+')&&(B_Usart2[a+6]=='A'))
			{
						while(B_Usart2[a]!="")
						{
						a++;
						}
	x=100*(B_Usart2[a+1]-0x30)+10*(B_Usart2[a+2]-0x30)+(B_Usart2[a+3]-0x30);
```

```
                        break;
                }
                a++;
        }
        if(a==100)
                x=10;
}
else
        x=0;
break;
case 2:
        if(C_Enters>=enters)//numero de enters encontrados
        {
                while(a<100)
                {       if(B_Usart2[a]=='(')
                        {
                        break;
                        }
                        a++;
                }

                if(a<100)
                {       while(b<4)
                        {
                                if(B_Usart2[a]==',')
                                {
                                b++;
                                }
                                a++;
                        }
                        while(B_Usart2[a]!=')')
                        {       if(B_Usart2[a]==',')
                                {       y=x*256;x=0;    }

                                else if((B_Usart2[a]>=0x30)&&(B_Usart2[a]<=0x39))
                                {       x=10*x+B_Usart2[a]-0x30;}
                                a++;
                        }
```

```
                        x+=y;
                }

                else
                        x=10;
        }

        else
                x=0;
        break;

case 3:
        if(C_Enters>=enters)
        {
                while(a<100)
                {       if(B_Usart2[a]=='>')
                        {       x=1;break;      }
                        a++;
                }

                if(a==100)
                {
                        a=0;
                        while(a<100)
                {
                if((B_Usart2[a]=='E')&&(B_Usart2[a+1]=='R')&&(B_Usart2[a+2]=='R'))
                                {       x=10;break;     }
                                a++;
                }
                        if(a==100)
                                x=0;
                }
        }
        else
                x=0;
        break;
}
return x;
```

```
}


void Tarea_Usart2(void)
{
	unsigned int x;
	unsigned char i;

	if(F_Envia_Datos==1)
	{
		Clean_Usart2();
		Config_Timer1();
		F_Envia_Datos++;
		F_Conexion=0;
	}


	else if(F_Envia_Datos==2)
	{
		switch(F_Conexion)
		{
			case 0:
				WriteStr_Usart2(at);Write_Usart2(enter);
				F_Conexion++;
				break;

			case 1:
				x=Scan_String(0,4);

				if(x==1)
				{	F_Conexion++;Clean_Usart2();Escribe_Lcd("AT --> OK");}

				else if(x==10)
				{F_Conexion--;Clean_Usart2();Escribe_Lcd("AT --> ERROR");}
			break;

			case 2:
				WriteStr_Usart2(at);WriteStr_Usart2(server);
```

```
        Write_Usart2(enter);F_Conexion++;
        break;

case 3:
        x=Scan_String(0,4);
        if(x==1)//ok
        {F_Conexion++;Clean_Usart2();Escribe_Lcd("server-->OK");}

        else if(x==10)
        {F_Conexion--;Clean_Usart2();Escribe_Lcd("server error");}
break;

case 4:
        WriteStr_Usart2(at);WriteStr_Usart2(ipconfig);
        Write_Usart2(enter);F_Conexion++;
        break;

case 5:
        x=Scan_String(0,4);
        if(x==1)
        {       F_Conexion++;Clean_Usart2();Escribe_Lcd("config OK");}

        else if(x==10)
        {       F_Conexion--;Clean_Usart2();Escribe_Lcd("config error");}
break;

case 6:
        WriteStr_Usart2(at);WriteStr_Usart2(netstart);
        Write_Usart2(enter);F_Conexion++;
        break;

case 7:
        x=Scan_String(0,4);
        if(x==1)//ok
        {       F_Conexion++;Clean_Usart2();Escribe_Lcd("net ok");}

        else if(x==10)
        {       F_Conexion--;Clean_Usart2();Escribe_Lcd("net error");}
```

```
break;

case 8:
        WriteStr_Usart2(at);WriteStr_Usart2(ipdest);Write_Usart2(',');
        WriteStr_Usart2(S_FtpC);Write_Usart2(enter);F_Conexion++;
        break;

case 9:
        x=Scan_String(0,8);
        if(x==1)
        {       F_Conexion++;Clean_Usart2();Escribe_Lcd("ftp 1 ok");}

        else if(x==10)
        {       F_Conexion--;Clean_Usart2();Escribe_Lcd("ftp 1 err");}
break;

case 10:
        WriteStr_Usart2(at);WriteStr_Usart2(ipopen);
        Write_Usart2(enter);F_Conexion++;
        break;

case 11:
        x=Scan_String(1,22);
        if(x==220)//ok
        {
        Sockets_Locales(0);
        F_Conexion++;Clean_Usart2();Escribe_Lcd("ipopen1 ok");
        }

        else if(x==10)/
        {       F_Conexion--;Clean_Usart2();Escribe_Lcd("ipopen1 err");}
        break;

case 12:
        WriteStr_Usart2(at);WriteStr_Usart2(ipsend);
        WriteStr_Usart2(user);Write_Usart2(',');
        WriteStr_Usart2(C_Comm);
        Write_Usart2(enter);F_Conexion++;
```

```
        break;

case 13:
        x=Scan_String(1,8);
        if(x==331)
        {       F_Conexion++;Clean_Usart2();Escribe_Lcd("login ok");}

        else if(x==10)
        {       F_Conexion--;Clean_Usart2();Escribe_Lcd("login err");}
        break;

case 14:
        WriteStr_Usart2(at);WriteStr_Usart2(ipsend);
        WriteStr_Usart2(pass);Write_Usart2(',');
        WriteStr_Usart2(C_Comm);
        Write_Usart2(enter);F_Conexion++;
        break;

case 15:
        if(x==230)
        {       F_Conexion++;Clean_Usart2();Escribe_Lcd("pass ok");}

        else if(x==10)
        {       F_Conexion--;Clean_Usart2();Escribe_Lcd("pass err");}
        break;


case 16:
        WriteStr_Usart2(at);WriteStr_Usart2(ipsend);
        WriteStr_Usart2(type);Write_Usart2(',');
        WriteStr_Usart2(C_Comm);
        Write_Usart2(enter);F_Conexion++;
        break;

case 17:
        x=Scan_String(1,8);
        if(x==200)//ok
        {       F_Conexion++;Clean_Usart2();Escribe_Lcd("type ok");}
```

```
        else if(x==10)
        {       F_Conexion--;Clean_Usart2();Escribe_Lcd("type err");}
break;

case 18:
        WriteStr_Usart2(at);WriteStr_Usart2(ipsend);WriteStr_Usart2(pasv);
        Write_Usart2(',');WriteStr_Usart2(C_Comm);
        Write_Usart2(enter);F_Conexion++;
        break;

case 19:
        x=Scan_String(1,8);
        if(x==227)//ok
        {       x=Scan_String(2,8);Socket_FtpData(x);
                F_Conexion++;Clean_Usart2();Escribe_Lcd("passM ok");
        }

        else if(x==10)
        {       F_Conexion--;Clean_Usart2();Escribe_Lcd("passM err");}
        break;

case 20:
        WriteStr_Usart2(at);WriteStr_Usart2(ipdest);Write_Usart2(',');
        WriteStr_Usart2(S_FtpD);Write_Usart2(enter);F_Conexion++;
        break;

case 21:
        x=Scan_String(0,8);
        if(x==1)
        {       F_Conexion++;Clean_Usart2();Escribe_Lcd("ftp2 ok");}

        else if(x==10)
        {       F_Conexion--;Clean_Usart2();Escribe_Lcd("ftp2 err");}
        break;

case 22:
        WriteStr_Usart2(at);WriteStr_Usart2(ipopen);
```

```
                Write_Usart2(enter);F_Conexion++;
                break;

        case 23:
                x=Scan_String(0,8);
                if(x==1)
                {       Sockets_Locales(1);
                        F_Conexion++;Clean_Usart2();Escribe_Lcd("ipopen2 ok");}

                else if(x==10)
                {       F_Conexion--;Clean_Usart2();Escribe_Lcd("ipopen2 err");}
                break;

//----------------------------- envio de datos via GPRS ------------------------------//

        case 24:
                WriteStr_Usart2(at);WriteStr_Usart2(ipsend);
                WriteStr_Usart2(append);

                x=READ_MEM(0x7FEA);

                for(i=0;i<x;i++)
                {       Write_Usart2(READ_MEM(i));}

                WriteStr_Usart2("\\n");Write_Usart2("");Write_Usart2(',');
                WriteStr_Usart2(C_Comm);Write_Usart2(enter);
                F_Conexion++;
                break;

        case 25:
                x=Scan_String(0,8);
                if(x==1)
                {F_Conexion++;Clean_Usart2();Escribe_Lcd("ipsendb ok");}

                else if(x==10)
                {       F_Conexion--;Clean_Usart2();Escribe_Lcd("ipsendb err");}
                break;
```

```
case 26:

        P_Data=0x40;
        P_Aux=0x40;
        N_Data=0;
        F_Conexion++;
        break;

case 27:

        Clean_Usart2();
        x=(READ_MEM(0x7FFE)<<8)|READ_MEM(0x7FFF);

        if(P_Data==x)
                F_Conexion=31;
        else if(x>P_Data+1000)
        {
                B_Alt[0]=0x31;B_Alt[1]=0x30;B_Alt[2]=0x30;B_Alt[3]=0x30;
                WriteStr_Usart2(at);WriteStr_Usart2(ipsendb);
                WriteStr_Usart2(B_Alt);Write_Usart2(',');
                WriteStr_Usart2(C_Data);Write_Usart2(',');
                Write_Usart2(0x30);Write_Usart2(enter);

                N_Data=1000+P_Data;
                P_Aux=P_Data;
                F_Conexion++;
        }
        else
        {
                N_Data=x-P_Data;B_Alt[0]=0x30;
                B_Alt[1]=N_Data/100+0x30;N_Data=N_Data%100;
                B_Alt[2]=N_Data/10+0x30;B_Alt[3]=N_Data%10+0x30;

                WriteStr_Usart2(at);WriteStr_Usart2(ipsendb);
                WriteStr_Usart2(B_Alt);Write_Usart2(',');
                WriteStr_Usart2(C_Data);Write_Usart2(',');
                Write_Usart2(0x30);        Write_Usart2(enter);
```

```
                N_Data=x;
                P_Aux=P_Data;
                F_Conexion++;
        }
        break;

case 28:
        x=Scan_String(3,2);
        if(x==1)
        {       F_Conexion++;Clean_Usart2();Escribe_Lcd("0D 0A>");      }

        else if(x==10)
        {       F_Conexion=27;Clean_Usart2();   }

        break;

case 29:
        if(P_Data==N_Data)
                F_Conexion++;

        else
        {       Write_Usart2(READ_MEM(P_Data));
                P_Data++;}
        break;

case 30:
        x=Scan_String(0,8);
        if(x==1)//datos OK
        {       F_Conexion=27;Clean_Usart2();}

        else if(x==10)
        {
                P_Data=P_Aux;
                F_Conexion=27;
                Clean_Usart2();
        }
        break;
```

```
case 31:
        WriteStr_Usart2(at);WriteStr_Usart2(ipclose);
        WriteStr_Usart2(C_Data);
        Write_Usart2(enter);F_Conexion++;

        break;

case 32:

        x=Scan_String(1,10);
        if(x==226)
        {       F_Conexion++;Clean_Usart2();Escribe_Lcd("close dat");}

        else if(x==10)
        {       F_Conexion--;Clean_Usart2();    }
        break;


case 33:
        WriteStr_Usart2(at);WriteStr_Usart2(ipsend);
        WriteStr_Usart2(quit);Write_Usart2(',');
        WriteStr_Usart2(C_Comm);
        Write_Usart2(enter);F_Conexion++;

        break;

case 34:
        x=Scan_String(1,10);
        if(x==221)
        {F_Conexion++;Clean_Usart2();Escribe_Lcd("close comm");}

        else if(x==10)
        {       F_Conexion--;Clean_Usart2();    }
        break;

case 35:
        T1CON=0x00;
        C_Comm[0]=0x30;C_Comm[1]=0x30;
```

```
                C_Data[0]=0x30;C_Data[1]=0x30;
                Socket_FtpData(0);
                Escribe_Lcd("Data sent");
                WriteStr_Usart2(at);
                WriteStr_Usart2(netstop);
                Write_Usart2(enter);
                F_Conexion++;
                break;

        case 36:
                x=Scan_String(0,4);
                if(x==1)
                {       F_Conexion=0;
                        F_Envia_Datos=3;
                        Escribe_Lcd("GPRS OFF");
                }
                break;

//--------------------------------error en tiempo de envio de datos----------------------//

        case 50:
                C_Comm[0]=0x30;C_Comm[1]=0x30;
                C_Data[0]=0x30;C_Data[1]=0x30;
                Socket_FtpData(0);
                Escribe_Lcd("Unsent Data");
                WriteStr_Usart2(at);
                WriteStr_Usart2(netstop);
                Write_Usart2(enter);
                F_Conexion++;
                break;

        case 51:
                x=Scan_String(0,4);
                if(x==1)
                {       F_Conexion=0;
                        F_Envia_Datos=10;
                        Escribe_Lcd("GPRS OFF");
                }
```

```
                                    break;
                        }
                }
}
```

fram.c

```
#include<p30F4013.h>

//*************** comandos para la memoria FRAM ***************************
#define WRITE_ENABLE 0x06           //opcode para habilitar escritura en FRAM
#define WRITE_DISABLE 0x04          //opcode para deshabilitar escritura FRAM
#define READ 0x03                   //lee el banco de memoria
#define WRITE 0x02                  //escribe en el banco de memoria
#define READSR 0x05                 //lee el registro STATUS de la memoria
#define WRITESR 0x01                //escribe al registro STATUS de la memoria
#define Clocks    0xaa              //ciclos de reloj para respuesta
#define FRAM_PTR1     0x7FFE        //penúltima localidad de memoria y última localidad de
#define FRAM_PTR2     0x7FFF        //memoria; entre los 2 hacen un registro de 16 bits
                                    //para apuntador de memoria libre
#define FRAM_NAME    0x7FEA         //localidad para guardar longitud de nombre de archivo
#define FRAM _LATD2                 //pin ChipSelect para el control de la memoria FRAM
#define ENABLE          0           //activar memoria FRAM
#define DISABLE         1           //desactivar memoria FRAM


//declaracion de funciones

void WRITES(unsigned int,unsigned char);
unsigned char READ_SR(void);                //lee el registro STATUS de la memoria
unsigned char READ_MEM(unsigned int);       //lee datos del banco especificado de la FRAM
void WRITE_SR(unsigned char);               //escribe al registro STATUS de la memoria
void WRITE_ED(unsigned char);               //habilita o deshabilita escritura a la memoria FRAM
unsigned char Orden(unsigned char);         //comando de escritura al buffer SPI
void Config_Fram(void);
```

```
void Erase_Fram(void);
void SaveData_Fram(unsigned char[],unsigned char,unsigned char,unsigned char);
void Tarea_Fram(void);

//declaracion de variables locales

unsigned char mem_mid[]="\r\nMEM 50%\r\n";
unsigned char mem_full[]="\r\nMEM FULL\r\n";



void Config_Fram(void)
{
	_TRISD2=0;
	SPI1CON=0x0122;
	SPI1STAT=0x8000;
	FRAM=DISABLE;
}

void Erase_Fram(void)
{
	unsigned char trash;
	unsigned int x;

	WRITE_SR(0x80);
	WRITE_ED(WRITE_ENABLE);
	FRAM=ENABLE;
	Nop();Nop();Nop();Nop();
	trash=Orden(WRITE);
	trash=Orden(0x00);
	trash=Orden(0x00);

	for(x=0;x<=0x7DFF;x++)
	{	trash=Orden(0xff);	}

	FRAM=DISABLE;

	WRITES(FRAM_PTR1,0x00);
```

```
	WRITES(FRAM_PTR2,0x40);
	WRITES(FRAM_NAME,0x00);
}


void SaveData_Fram(unsigned char datos[],unsigned char inicio,unsigned char long_datos,unsigned char
name_data)
{
	unsigned char trash,msb,lsb,x;
	unsigned int MEM;


	if(name_data==0)
	{
		msb=0x00;
		lsb=0x00;
	}

	else
	{
		msb=READ_MEM(FRAM_PTR1);
		lsb=READ_MEM(FRAM_PTR2);
		MEM=(msb<<8)|lsb;
	}

	//**********************escribe en la memoria FRAM**************************
	WRITE_ED(WRITE_ENABLE);
	FRAM=ENABLE;
	Nop();Nop();Nop();Nop();
	trash=Orden(WRITE);
	trash=Orden(msb);
	trash=Orden(lsb);
	for(x=0;x<long_datos-inicio;x++)
	{	trash=Orden(datos[x+inicio]);}


	FRAM=DISABLE;
```

```
        if(name_data==0)
        WRITES(FRAM_NAME,long_datos-inicio);
        else
        {
                MEM+=long_datos;
                msb=MEM>>8;
                lsb=MEM&0xff;
                WRITES(FRAM_PTR1,msb);
                WRITES(FRAM_PTR2,lsb);
        }
}
void WRITES(unsigned int local,unsigned char dats)
{
        unsigned char msb,lsb,trash;

        msb=local>>8;
        lsb=local&0xff;

        //**********************escribe en la memoria FRAM**************************

        WRITE_ED(WRITE_ENABLE);
        FRAM=ENABLE;
        Nop();Nop();Nop();Nop();
        trash=Orden(WRITE);
        trash=Orden(msb);
        trash=Orden(lsb);
        trash=Orden(dats);
        FRAM=DISABLE;
}
unsigned char Orden(unsigned char X)
{
        unsigned char data;
        IFS0bits.SPI1IF=0;
        SPI1BUF=X;
        while(!IFS0bits.SPI1IF);
        IFS0bits.SPI1IF=0;
        data=SPI1BUF;
        return data;
```

```
}
void WRITE_ED(unsigned char enadis)
{
	unsigned char data;

	FRAM=ENABLE;
	Nop();Nop();Nop();Nop();
	data=Orden(enadis);
	FRAM=DISABLE;
}
unsigned char READ_MEM(unsigned int banks)
{
	unsigned char msb,lsb,data;

	msb=banks>>8;
	lsb=banks&0xff;
	FRAM=ENABLE;
	Nop();Nop();

	data=Orden(READ);
	data=Orden(msb);
	data=Orden(lsb);
	data=Orden(Clocks);
	FRAM=DISABLE;

	return data;
}
unsigned char READ_SR(void)
{
	unsigned char data;
	FRAM=ENABLE;
	Nop();Nop();
	data=Orden(READSR);
	data=Orden(Clocks);
	FRAM=DISABLE;

	return data;
}
```

```
void WRITE_SR(unsigned char comm)
{
	unsigned char data;

	WRITE_ED(WRITE_ENABLE);
	FRAM=ENABLE;
	Nop();Nop();Nop();Nop();
	data=Orden(WRITESR);
	data=Orden(comm);
	FRAM=DISABLE;
}
void Tarea_Fram(void)
{
	unsigned int memory;
	unsigned char msb,lsb;

	msb=READ_MEM(FRAM_PTR1);
	lsb=READ_MEM(FRAM_PTR2);
	memory=(msb<<8)|lsb;

	if(memory>16000)
		WriteStr_Usart1(mem_mid);

	else if(memory>32000)
	{
		WriteStr_Usart1(mem_full);
		WRITE_SR(0x8c);
	}
}
```

Lcd_nokia_dspic.c

Nokia_CLK	B1
Nokia_DIN	B2
Nokia_DC	B3
Nokia_CS	B4

```
Nokia_RST                B5


#include<p30F4013.h>

#define Nokia_CLK       LATBbits.LATB1
#define Nokia_DIN       LATBbits.LATB2
#define Nokia_DC        LATBbits.LATB3
#define Nokia_CS        LATBbits.LATB4
#define Nokia_RST       LATBbits.LATB5

//declaracion de funciones

void Config_Lcdnokia(void);
void LCD_Nokia_Command(unsigned char);
void LCD_Nokia_Data(unsigned char);
void Nokia_Write_Byte(unsigned char);
void LCD_Nokia_Clean(void);
void LCD_Nokia_XY(unsigned char,unsigned char);
void Delay(unsigned char);
void SetPixel(unsigned char,unsigned char);
void CleanMemVideo(void);
void RefreshVideo(void);
void LCD_Nokia_txt_xy(unsigned char, unsigned char);
void LCD_Nokia_txt_Char(unsigned char);
void LCD_Nokia_txt_String(unsigned char *);
void Escribe_Lcd(unsigned char datos[]);


// variables locales para el display grafico

unsigned char MemVideo[504];
unsigned int LCD_address;

//funciones del LCD

void Config_Lcdnokia(void)
{
```

```
        _TRISB1=0;_TRISB2=0;_TRISB3=0;
        _TRISB4=0;_TRISB5=0;
        Nokia_DC=1;
        Nokia_CS=1;
        Nokia_RST=0;

        Delay(36);
        Nokia_RST=1;
        LCD_Nokia_Command(0x21);
        LCD_Nokia_Command(0xc2);
        LCD_Nokia_Command(0x13);
        LCD_Nokia_Command(0x20);
        LCD_Nokia_Command(0x09);
        LCD_Nokia_Clean();
        LCD_Nokia_Command(0x08);
        LCD_Nokia_Command(0x0c);
}

void LCD_Nokia_Command(unsigned char Command)
{
        Nokia_DC=0;
        Nokia_CS=0;
        Nokia_Write_Byte(Command);
        Nokia_CS=1;
}

void LCD_Nokia_Data(unsigned char Data)
 {
  Nokia_DC=1;
  Nokia_CS=0;
  Nokia_Write_Byte(Data);
  Nokia_CS=1;
 }

void Nokia_Write_Byte(unsigned char Data)
{
        unsigned char i;
        for (i=8;i>0;i--)
```

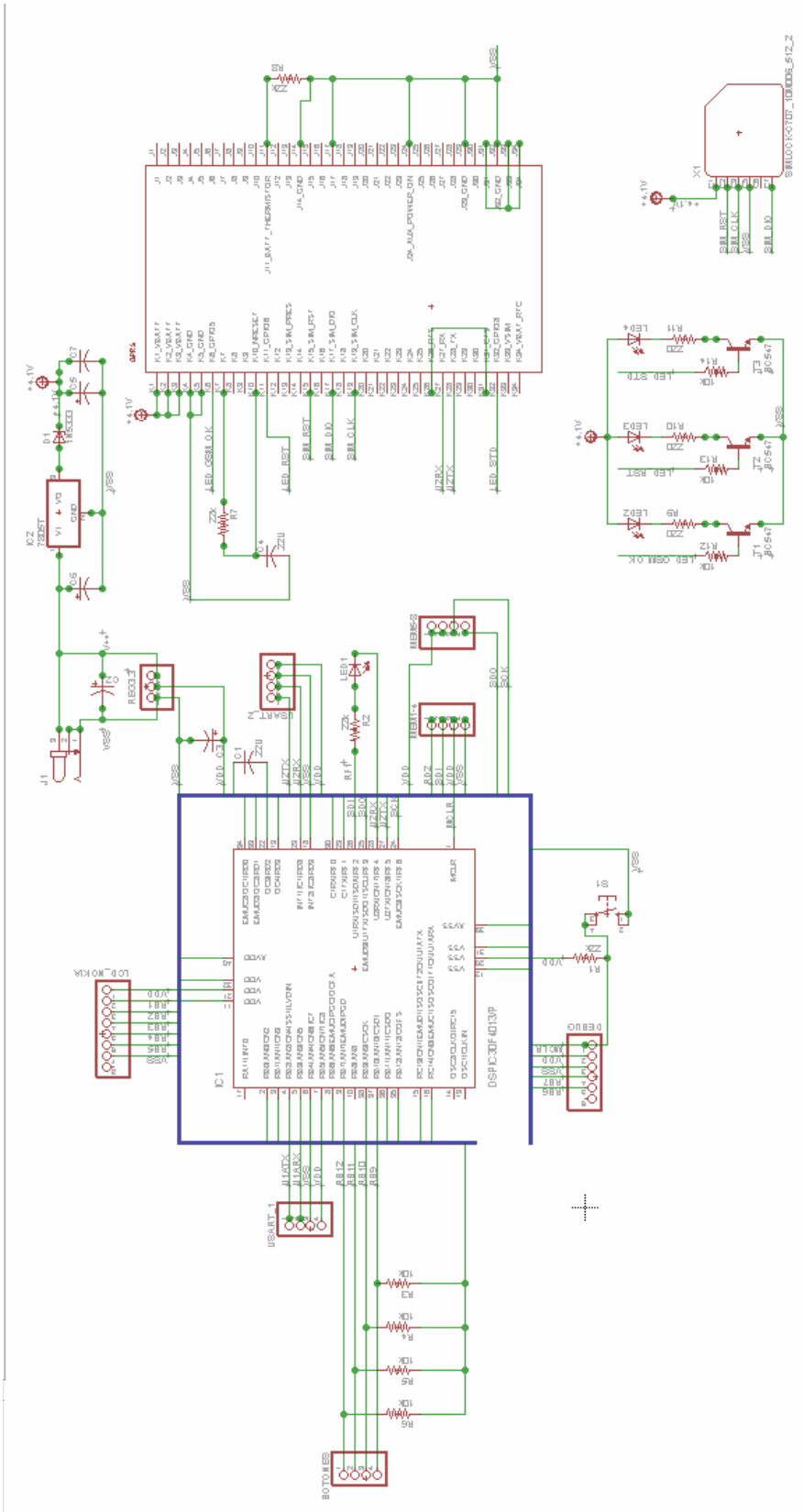
```
        {
                Nokia_CLK=0;
        if ((Data&0x80) == 0)
                {
                        Nokia_DIN = 0;
                }
                else
                {
                        Nokia_DIN = 1;
    }
        Nokia_CLK = 1;
        Data = Data << 1;
        }
}

void LCD_Nokia_Clean(void)
{
        unsigned int i;
        LCD_Nokia_XY(0,0);
        for (i=504;i>0;i--)
                LCD_Nokia_Data(0x00);
}
void LCD_Nokia_XY(unsigned char x,unsigned char y )
{
        LCD_Nokia_Command(0x40|(y&0x07));
        LCD_Nokia_Command(0x80|(x&0x7f));
}

void Delay(unsigned char tiempo)
{
                unsigned int y;

                while(tiempo!=0)
                {
                        y=7380;
                        while(y!=0)
                        {       y--;}
```

```
                        tiempo--;
                }
}


// manejo de la memoria de video con funciones especiales //
//*********************************************************//

void SetPixel(unsigned char X,unsigned char Y )
{
        unsigned char Pixel;
        unsigned char *Pointer;

        Y = 47-Y;
        Pointer = MemVideo + (int)(Y/8)*84+X;
   Pixel = Y % 8;
   Pixel = 1 << Pixel;
        *Pointer = *Pointer | Pixel;
}

void CleanMemVideo(void)
{
unsigned int i;
unsigned char *pointer;
        pointer = MemVideo;
   for(i=0;i<504;i++)
        {
                *pointer=0;
                pointer++;
        }
}

void RefreshVideo(void)
{
unsigned int i;
unsigned char *pointer;
        pointer = MemVideo;
        LCD_Nokia_XY(0,0);
```

```
        i=0;
   do
        {
                LCD_Nokia_Data(*pointer);
                pointer++;
                i++;
        }while(i<504);
}

void LCD_Nokia_txt_xy(unsigned char x, unsigned char y)
{
        LCD_address = (unsigned int) y * 84;
        LCD_address = LCD_address + x;
}


void LCD_Nokia_txt_Char(unsigned char caracter)
{
unsigned int posicion;
unsigned char i;
        posicion = caracter - 0x20;
        posicion = posicion * 5;

        for(i=0;i<5;i++)
        {
                MemVideo[LCD_address] = READ_MEM(posicion+0x7E00);
                posicion++;
                LCD_address++;
        }
}

void LCD_Nokia_txt_String(unsigned char *cadena)
{
        while(*cadena != 0)
        {
                LCD_Nokia_txt_Char(*cadena);
                cadena++;
        }
```

```
}

void Escribe_Lcd(unsigned char datos[])
{
	unsigned int i_MV;

	i_MV=503;
	do
	{
		MemVideo[i_MV]=MemVideo[i_MV-84];
		i_MV--;
	}while(i_MV>=84);

	for(i_MV=0;i_MV<84;i_MV++)
		MemVideo[i_MV]=0;

	LCD_Nokia_txt_xy(0,0);
	LCD_Nokia_txt_String(datos);
	RefreshVideo();
}
```

F. DIAGRAMA DEL MODULO GPRS.



G. HOJA DE DATOS DE LA MEMORIA FRAM

FM25L256B

256Kb FRAM Serial 3V Memory

RAMTRON

Features

256K bit Ferroelectric Nonvolatile RAM
- Organized as 32,768 x 8 bits
- Unlimited Read/Write Cycles
- 45 Year Data Retention
- NoDelay™ Writes
- Advanced High-Reliability Ferroelectric Process

Very Fast Serial Peripheral Interface - SPI
- Up to 20 MHz Frequency
- Direct Hardware Replacement for EEPROM
- SPI Mode 0 & 3 (CPOL, CPHA=0,0 & 1,1)

Write Protection Scheme
- Hardware Protection
- Software Protection

Low Power Consumption
- Low Voltage Operation 2.7V – 3.6V

Industry Standard Configurations
- Industrial Temperature -40°C to +85°C
- 8-pin SOIC and 8-pin TDFN Packages
- "Green"/RoHS Packaging

Electrical Specifications

Absolute Maximum Ratings

Symbol	Description	Ratings
V_{DD}	Power Supply Voltage with respect to V_{SS}	-1.0V to +5.0V
V_{IN}	Voltage on any pin with respect to V_{SS}	-1.0V to +5.0V and $V_{IN} < V_{DD}+1.0V$
T_{STG}	Storage Temperature	-55°C to + 125°C
T_{LEAD}	Lead Temperature (Soldering, 10 seconds)	300° C
V_{ESD}	Electrostatic Discharge Voltage - Human Body Model (JEDEC Std JESD22-A114-D) - Charged Device Model (JEDEC Std JESD22-C101-C) - Machine Model (JEDEC Std JESD22-A115-A)	 4kV 1kV 200V
	Package Moisture Sensitivity Level	MSL-1

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only, and the functional operation of the device at these or any other conditions above those listed in the operational section of this specification is not implied. Exposure to absolute maximum ratings conditions for extended periods may affect device reliability.

DC Operating Conditions (T_A = -40°C to +85°C, V_{DD} = 2.7V to 3.6V unless otherwise specified)

Symbol	Parameter	Min	Typ	Max	Units	Notes
V_{DD}	Power Supply Voltage	2.7	-	3.6	V	
I_{DD}	Power Supply Current					1
	@ SCK = 1.0 MHz		-	0.5	mA	
	@ SCK = 20.0 MHz		-	10.0	mA	
I_{SB}	Standby Current					
	@ T_A = 25°C	-		1.0	µA	2, 4
	@ T_A = 55°C	-		2.5	µA	2, 4
	@ T_A = 70°C	-		5.0	µA	2, 4
	@ T_A = 85°C	-		10.0	µA	2
I_{LI}	Input Leakage Current	-		±1	µA	3
I_{LO}	Output Leakage Current			±1	µA	3
V_{IH}	Input High Voltage	0.7 V_{DD}		V_{DD} + 0.5	V	
V_{IL}	Input Low Voltage	-0.3		0.3 V_{DD}	V	
V_{OH}	Output High Voltage	V_{DD} – 0.8		-	V	
	@ I_{OH} = -2 mA					
V_{OL}	Output Low Voltage	-		0.4	V	
	@ I_{OL} = 2 mA					

Notes

1. SCK toggling between V_{DD}-0.3V and V_{SS}, other inputs V_{SS} or V_{DD}-0.3V.
2. SCK = SI = /CS=V_{DD}. All inputs V_{SS} or V_{DD}.
3. $V_{SS} \le V_{IN} \le V_{DD}$ and $V_{SS} \le V_{OUT} \le V_{DD}$.
4. This parameter is characterized but not 100% tested.

AC Parameters (T_A = -40° C to +85° C, V_{DD} = 2.7V to 3.6V, C_L = 30pF)

Symbol	Parameter	Min	Max	Units	Notes
f_{CK}	SCK Clock Frequency	0	20	MHz	
t_{CH}	Clock High Time	22		ns	1
t_{CL}	Clock Low Time	22		ns	1
t_{CSU}	Chip Select Setup	10		ns	
t_{CSH}	Chip Select Hold	10		ns	
t_{OD}	Output Disable Time		20	ns	2
t_{ODV}	Output Data Valid Time		22	ns	
t_{OH}	Output Hold Time	0		ns	
t_D	Deselect Time	60		ns	
t_R	Data In Rise Time		50	ns	2,3
t_F	Data In Fall Time		50	ns	2,3
t_{SU}	Data Setup Time	5		ns	
t_H	Data Hold Time	5		ns	
t_{HS}	/Hold Setup Time	10		ns	
t_{HH}	/Hold Hold Time	10		ns	
t_{HZ}	/Hold Low to Hi-Z		20	ns	2
t_{LZ}	/Hold High to Data Active		20	ns	2

Notes

1. $t_{CH} + t_{CL} = 1/f_{CK}$.
2. This parameter is characterized but not 100% tested.
3. Rise and fall times measured between 10% and 90% of waveform.

Power Cycle Timing (T_A = -40° C to +85° C, V_{DD} = 2.7V to 3.6V)

Symbol	Parameter	Min	Max	Units	Notes
t_{PU}	Power Up (V_{DD} min) to First Access (/CS low)	10	-	ms	
t_{PD}	Last Access (/CS high) to Power Down (V_{DD} min)	0	-	µs	
t_{VR}	V_{DD} Rise Time	50	-	µs/V	1
t_{VF}	V_{DD} Fall Time - For V_{DD} above 2.0V - For V_{DD} below 2.0V	 50 1	 - -	 µs/V ms/V	1

Notes

1. Slope measured at any point on V_{DD} waveform.

Capacitance (T_A = 25° C, f=1.0 MHz, V_{DD} = 3.3V)

Symbol	Parameter	Min	Max	Units	Notes
C_O	Output Capacitance (SO)	-	8	pF	1
C_I	Input Capacitance	-	6	pF	1

Notes

1. This parameter is characterized and not 100% tested.

AC Test Conditions

Input Pulse Levels	10% and 90% of V_{DD}
Input rise and fall times	5 ns

H. HOJA DE DATOS DEL MICROCONTROLADOR.

MICROCHIP

dsPIC30F3014, dsPIC30F4013
Data Sheet

High-Performance
Digital Signal Controllers

FIGURE 1-1: dsPIC30F3014 BLOCK DIAGRAM

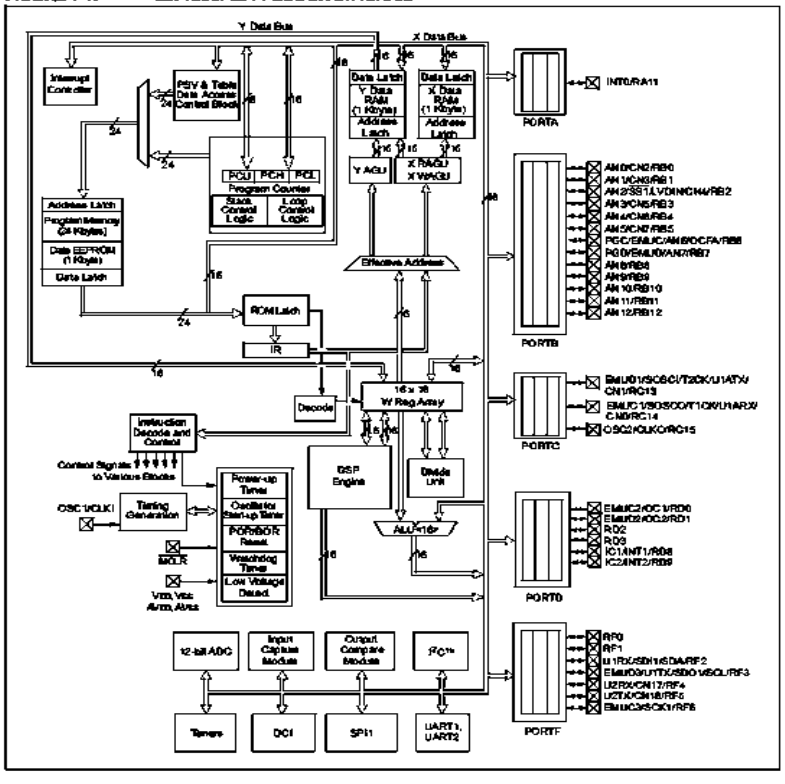
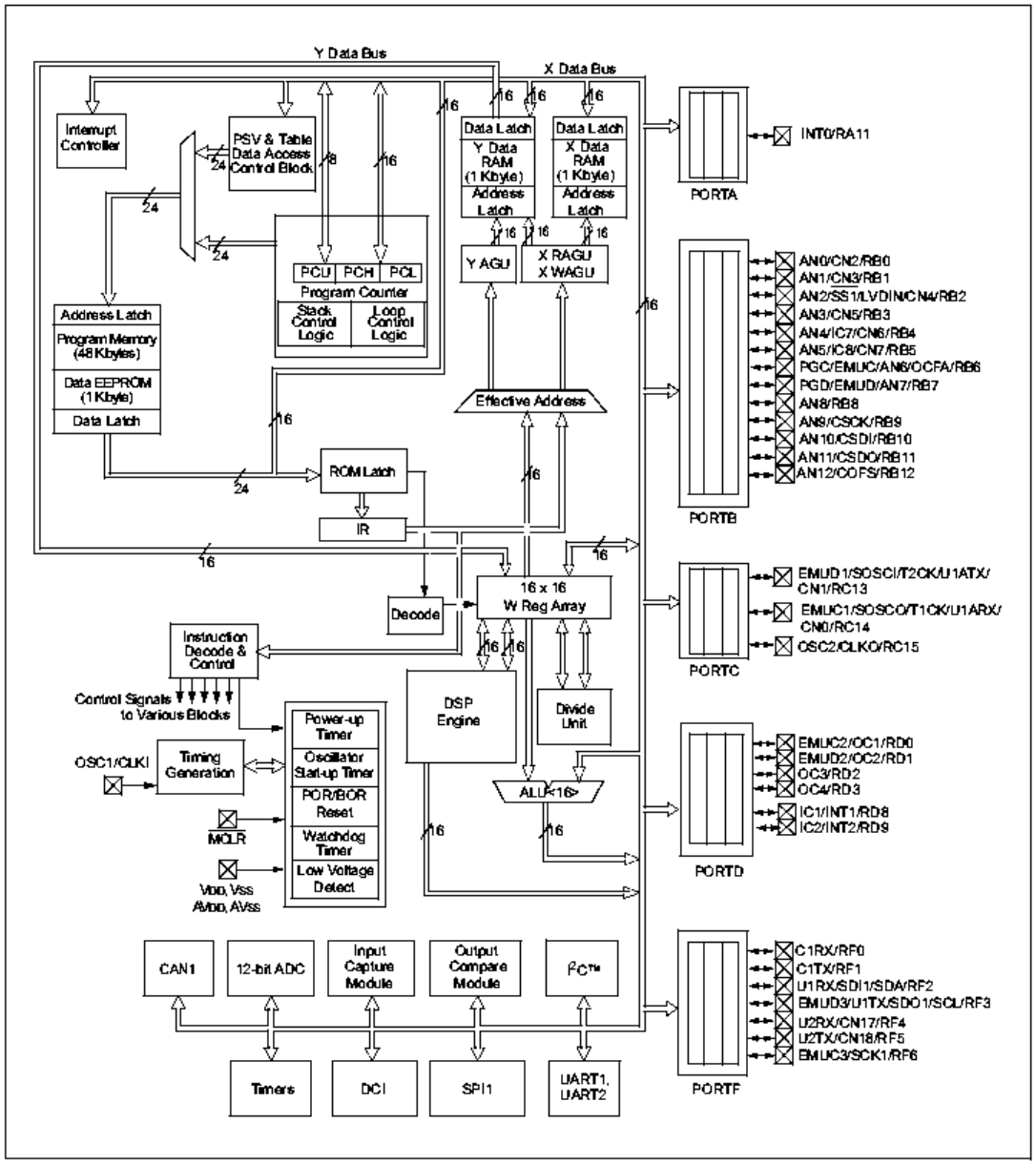


FIGURE 1-2: dsPIC30F4013 BLOCK DIAGRAM



I. DIAGRAMA DEL MODEM GSP PLUTO.

tiempos de espera o los dedicados a leer una página no le cuestan nada al cliente.

2.3 Comandos AT.

El protocolo empleado por los modems GSM para configurarlos y controlarlos está basado en el set de comandos AT Hayes, pero hay comandos específicos que han sido adaptados para los servicios que ofrece GSM tal es el caso de los mensajes de texto, manejo de memoria, funciones especiales de modem.
Los comandos AT para modems compatibles están definidos en los documentos ETSI GSM 07.07 y GSM 07.05.
GSM 07.07 define los comandos generales de GSM y GSM 07.05 define los comandos para manejo de mensajes escritos

2.3 SMS.

La especificación para SMS está dada en los documentos GSM 03.40 y GSM 03.38. Un SM puede tener hasta 160 caracteres de longitud, donde cada carácter es de 7 bits y corresponde a un carácter del alfabeto de 7 bits. En el caso de 8 bits el SM puede tener un máximo de 140 caracteres, empleado generalmente para mensajes inteligentes (imágenes y ringtones).

La estructura dentro del cual viaja un SM se le denomina PDU (Protocol Data Unit) la cual además de llevar información del propio texto lleva otra serie de caracteres con las que se pueden hacer funciones de control en la presentación del mensaje. La principal ventaja de este modo es que el mensaje antes de ser enviado a la red, debe pasar por un algoritmo el cual hace una codificación de 7 a 8 bits.

Al momento de emplear comandos AT los modems dan la opción de emplear el modo de envío/recepción PDU ó TEXTO, lo mas sencillo es manejar modo texto ya que en modo PDU hay que armar la cadena en forma comprimida. En el envío de Mensajes cortos esta involucrado el llamado SMSC (Centro de servicio de Mensajes Cortos) el cual se encarga del almacenamiento y envío de dicho mensaje, este se pude configurar por comandos AT, pero generalmente el modem GSM lo configura al conectarse a la red que le es asignada.

3. Módulo GSM/GPRS PLUTO[1].

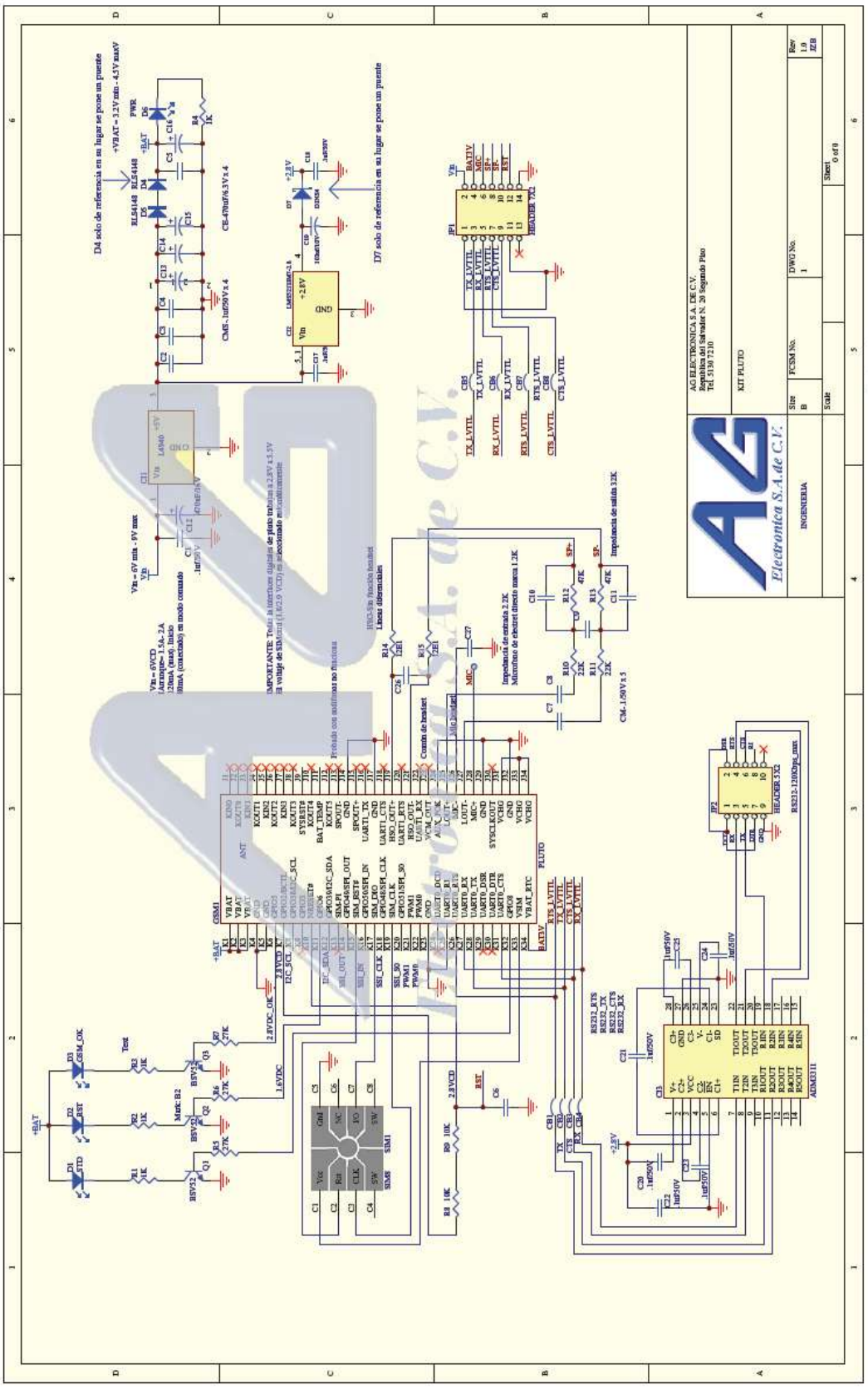
Pluto es un módulo GSM/GPRS basado en ARM9, el sistema operativo es Nucleus Plus el cual implementa un set de comandos AT compatibles con GSM y una pila TCP/IP.

Las características principales son:

- Módulo cuatribanda GSM850, E-GSM 900, DCS 1800, PCS1900.
- GPRS Clase B, Clase 1~12, con Esquemas de codificación CS1 a CS4.
- SMS soporta modo Texto y PDU.
- Interfaces:
  - Líneas de comunicación para tarjeta SIM 1.8/3V
  - Teclado de 6x4
  - 1 UART 115200 bps (maximo).
  - 5 GPIO.
  - Pin de reset
  - Pin de PowerOn
  - Entrada para micrófono de electret.
  - Salida para lamavoz.
  - 1 Salida 26 MHZ o 32.768KHz.
  - Alimentación de RTC.
- Antena helicoidal integrada.

- La pila TCP/IP no contempla APIs para SMTP, PoP3 o FTP.
- No funciona en modo CSD.

Fig. 7. Módulo PLUTO-PCBA-GSM.


AG ELECTRONICA S.A. DE C.V.
Electronica S.A. de C.V.
INGENIERIA
KIT PLUTO
www.agelectronica.com

BIBLIOGRAFÍA.

[Microchip 2004]

Microchip Technology Inc., dsPIC30F3014, dsPIC30F4013 Data Sheet, 2004.

[AG Electronica 2009]

AG Electronica S.A de C.V, Tarjeta de Evaluación Módulo GSM-GPRS PLUTO, 2009.

[Ramtron 2009]

Ramtron Inc., FM25L256B Data Sheet, 2009.

[CCww 2009]

Communications Consultants Worldwide Ltd, GSM/GPRS AT-Command User Guide, 2009.

[CCASIA 2009]

Communications Consultants, STAR-Lite GSM/GPRS Modem User Gide, 2009.

[Página web]

http://es.kioskea.net/contents/internet/ftp.php3

.

[Página web]

http://es.wikipedia.org/wiki/File_Transfer_Protocol

[Página web]

http://es.wikipedia.org/wiki/Energía_eólica.

[Página web]

http://es.kioskea.net/contents/telephonie-mobile/gsm.php3

[Articulo]

López Pérez, Eric. "*Protocolo SPI, Teoría y aplicaciones*".