Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Desarrollo de un sistema convertidor de protocolos para comunicación inalámbrica vía GSM

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura

Miguel Ángel Fonseca Porras

2010

INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PROYECTO DE GRADUACIÓN

TRIBUNAL EVALUADOR

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Ing. William Marín Moreno

Profesor lector

Ing. Aníbal Coto Cortés

Profesor lector

Ing. Johan Carvajal Godínez

Profesor asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica

Lugar y fecha de la presentación: Cartago 22/06/2010

Declaratoria de autenticidad

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía, he procedido a indicar las fuentes mediante las respectivas citas bibliográficas.

En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Cartago, 15/06/2010

Firma del autor

Miguel Ángel Fonseca Porras

Céd: *1-1204-0324*

Resumen

Este documento describe el proceso de investigación e implementación de un sistema de comunicaciones inalámbricas de larga distancia que se utilizó para conectar la plataforma de redes inalámbricas de sensores CRTECMote, con cualquier punto de interés; mediante una red de área amplia (*Wide Area Network* o *WAN*, de su traducción al inglés).

Los procedimientos seguidos para la ejecución del proyecto se basaron en una amplia investigación sobre protocolo USB y dispositivos de comunicación inalámbrica de tipo GSM. Producto de esta investigación, se desarrollaron rutinas de diseño de sistemas con las cuales se logró administrar un dispositivo de comunicaciones inalámbricas de tipo USB-GSM mediante un controlador de tipo USB.

En base de la administración del dispositivo de comunicaciones USB-GSM, se creó un sistema que permitió enviar cadenas de información por medio del sistema de mensajes cortos de texto. El canal de transmisión de la información se creó a través de una línea de telefonía celular conectada al servicio de última generación 3G brindado por el Instituto Costarricense de Electricidad.

La aplicación diseñada en este proyecto, cumplió los requerimientos de bajo consumo de potencia y transmisión eficiente de la información, lo cual la convierte en una aplicación de gran utilidad en los sistemas de monitoreo de redes inalámbricas de sensores.

La importancia de este proyecto es la de brindar un servicio de comunicación de la información contenida en una red de sensores; es por esto, que la solución de este problema llega a solventar la necesidad de transmitir información de carácter importante de una forma rápida y confiable.

Palabras claves: Sistema Operativo de Tiempo Real (RTOS), USB 2.0, Sistema de Comunicaciones Globales, Microcontrolador, EDGE-MODEM, Lenguaje C.

Summary

This document describes the process of research and implementation of a system of long-distance wireless communication that was used to connect the platform CRTECMote wireless sensor networks, with any point of interest, through a wide area network or WAN.

The procedures for the implementation of the project were based on extensive research into USB protocol and wireless GSM communication devices. As a result from this research, system design routines were developed which made possible to manage a USB-driver GSM wireless communications device via a USB controller.

On the basis of the administration of the USB-GSM communications device, a system that allowed sending information strings through a short text messages system was created. The information's transmission channel is created through a cellular telephone line connected to the next generation 3G services provided by the Instituto Costarricense de Electricidad.

The application designed meets the requirements of low power consumption and efficient information transmission, which makes it a very useful tool when monitoring wireless sensor network systems

The importance of this project is to provide a reporting service covering long distances, which is why the solution of this problem comes to solve the need to transmit significant information in fast and reliable manner.

Keywords: Real-Time Operating Systems (RTOS), USB 2.0, Global Communications System, Microcontroller, EDGE-MODEM, Language C.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Capítulo 1: Introducción

En los últimos años la protección del medio ambiente ha sido un factor clave en el desarrollo social y económico de muchos países alrededor del mundo. Debido a esto, muchas organizaciones han vuelto su mirada hacia la conservación y monitoreo de nuestros recursos naturales para evitar la explotación y destrucción de grandes aéreas protegidas.

Costa Rica es un país privilegiado al conservar una gran parte de su territorio, unos 13.000 kilómetros cuadrados, en parques y reservas nacionales . Aún así, se estima que en Costa Rica más del 25% de la madera que se consume es producto de la tala ilegal de arboles. A esta problemática, también se suman altos índices de explotación desmedida de los recursos naturales a nivel nacional. [5]

Por lo anterior, en Costa Rica existe la necesidad de realizar estudios ambientales que ayuden a proteger y monitorear los recursos protegidos dentro de los parques nacionales. Sin embargo, mantener un buen programa de estudio ambiental, trae consigo altos costos de pago de salarios y traslados de personal al campo de trabajo.

Una posible solución para disminuir los costos de este tipo de investigación, es el diseño de una red inalámbrica de sensores que se pueda instalar en el lugar donde se deben realizar los estudios de campo. Esto garantizaría una labor de monitoreo eficiente y se reduciría el costo de pago de personal.

La Escuela de Ingeniería en Electrónica del Instituto Tecnológico de Costa Rica ha puesto en marcha un proyecto de investigación para diseñar una red de sensores capaz de cumplir con las labores de monitoreo que se requieran realizar en las zonas de protección ambiental.

Este proyecto de graduación es una propuesta para desarrollar el sistema de comunicaciones inalámbricas que permita conectar las redes de sensores con cualquier punto de interés a larga distancia mediante una red de área amplia (*Wide*

Area Network o *WAN*, de su traducción al inglés). Esto con el objetivo de transmitir la información recopilada por los sensores hacia un punto de fusión de datos. De este modo, se podrán adquirir los datos ambientales valiosos tanto para investigadores como a autoridades de control, logrando un aporte significativo a la reducción de los gastos de traslados y del tiempo invertido por el personal; y garantizando así, un servicio de tiempo real que permita mejorar el desempeño en labores de control y estudio ambiental.

1.1 Entorno del Proyecto

La plataforma de redes inalámbricas CRTECMote actualmente cuenta con un sistema operativo capaz de administrar los recursos de hardware y software propios de la plataforma. Este sistema operativo lleva el nombre de SIWA RTOS, y fue desarrollado por el Ingeniero Norwin Leiva en su proyecto final de graduación durante II semestre del 2009. Su trabajo se basó en modificar el sistema operativo FreeRTOS, con el objetivo de lograr reducción en el consumo de potencia de un microcontrolador (MCU por sus siglas en inglés) PIC32MX. Concretamente logró una reducción de un 30% en el consumo de potencia al modificar el microkernel del FreeRTOS optimizando los algoritmos de ejecución del sistema operativo, dando paso así a SIWA RTOS.

Simultáneamente al desarrollo del sistema operativo SIWA RTOS, también se creó el nodo receptor de datos de la plataforma, también llamado Load Point. Este nodo fue desarrollado por el Ingeniero Dennis Rodríguez y su objetivo fue el de crear un sistema capaz de organizar la información recibida proveniente de la red inalámbrica de sensores mediante el uso de una base de datos creada en la aplicación MySQL. El aporte de este proyecto permitió a CRTECMote contar con un sistema de información que le permite a los usuarios el acceso a la información proveniente de la red de los sensores de una forma ordenada y fácil de administrar.

Gracias al aporte de estos proyectos, CRTECMote ha logrado crear las bases necesarias para desarrollar las últimas etapas propuestas del sistema.

1.2 Problema existente e importancia de su solución

1.2.1 Definición del Problema

CRTECMote es un proyecto dirigido por el Ingeniero Johan Carvajal Godínez y cuyo objetivo es el de crear una red inalámbrica de sensores capaz de monitorear variables ambientales.

Para dicho propósito se ha adquirido un microcontrolador de la empresa Microchip (PIC32MX460512), el cual funciona como un nodo receptor de todos los datos adquiridos por la red de sensores. Parte importante del diseño de la red inalámbrica de sensores es, que esta debe tener la capacidad de transmitir la información generada en el nodo de sensores, hacia cualquier punto de interés; el cual puede estar ubicado a unos cientos de metros o bien a muchos kilómetros de distancia del nodo principal.

Para crear esta posibilidad de comunicación inalámbrica, el sistema operativo SIWA-RTOS debe tener la capacidad de administrar el periférico de comunicaciones de largas distancias de tipo WAN; por lo tanto, el problema a resolver en este proyecto se divide en dos partes fundamentales:

1. Debido a que la red de sensores se encuentra en desarrollo, no se cuenta con un controlador de software funcional que permita comunicar el nodo sumidero de la red de sensores con el punto de acceso de recolección de datos.

2. No existe un diseño a nivel de hardware que permita conectar un dispositivo de comunicaciones con el nodo sumidero de la red de sensores.

1.2.2 Síntesis del Problema

Carencia de una interfaz de hardware y software que permita comunicar el nodo sumidero de las redes inalámbricas de sensores CRTECMote con el sistema de fusión de datos.

1.2.3 Importancia de la Solución

- ✓ Se podrá contar con un sistema que permita comunicar la información desde el lugar donde se encuentren los sensores, hasta cualquier punto de interés.
- ✓ Futuros diseñadores podrán utilizar el controlador de software para crear nuevos diseños y aplicaciones que permitan generar nuevos proyectos de investigación.
- ✓ Al cumplir con el objetivo general del proyecto, el proyecto CRTECMote podrá lanzar una versión completa de una red inalámbrica de sensores con comunicación WAN.
- ✓ Se abrirá paso a una nueva tecnología que permitirá proteger nuestros recursos naturales.

1.3 Solución seleccionada

La plataforma de redes inalámbricas CRTECMote es un sistema que debe cumplir con una serie de requerimientos de eficiencia, portabilidad y confiabilidad para transmitir la información contenida en los nodos de sensores. El diseño de cualquier aplicación que se quiera adaptar a la plataforma CRTECMote debe tener como prioridad un bajo consumo de energía y además ser aplicaciones que puedan ser administradas por medio de un sistema operativo.

Para brindar una solución que permita enviar la información desde el nodo sumidero de sensores, se debe cumplir con las exigencias descritas anteriormente. Esto implica, que se debe desarrollar una aplicación que sea eficiente en el uso de la energía y además permita un alto nivel de confianza en la información transmitida desde largas distancias.

Actualmente la plataforma de redes no cuenta con un sistema adecuado de comunicaciones para enlaces de larga distancia, y adquirir uno o varios transmisores elevaría el costo final del proyecto. Por lo tanto, la solución propuesta se basa en utilizar el servicio de telecomunicaciones que ofrece el Instituto Costarricense de Electricidad (ICE). Esta selección de solución se da en base a que el ICE cuenta con una infraestructura de telefonía celular con una amplia cobertura del territorio nacional, por lo tanto; la plataforma CRTECMote podrá contar con un servicio de información que podrá cubrir grandes distancias por medio de el servicio de mensajería que ofrece el instituto.

En síntesis con la solución planteada se espera recopilar la información contenida en el nodo sumidero y crear paquetes de información que contengan etiquetas con la fecha, hora, identificación y valor contenido en los sensores. Luego de recopilar esta información, se espera enviar estos paquetes de información a través de un MODEM GSM conectado a una línea celular. De este modo se utilizará el servicio de mensajería instantánea de texto como canal de comunicación entre el nodo sumidero de datos de los sensores y el nodo de fusión de datos.

El nodo de fusión de datos ya se encuentra implementado y su función es la de crear una interfaz visual que decodifica los datos recibidos e informa de manera accesible al usuario del sistema. En la figura 1 se presenta un esquema de la solución propuesta funcionando en conjunto con el nodo receptor de datos.

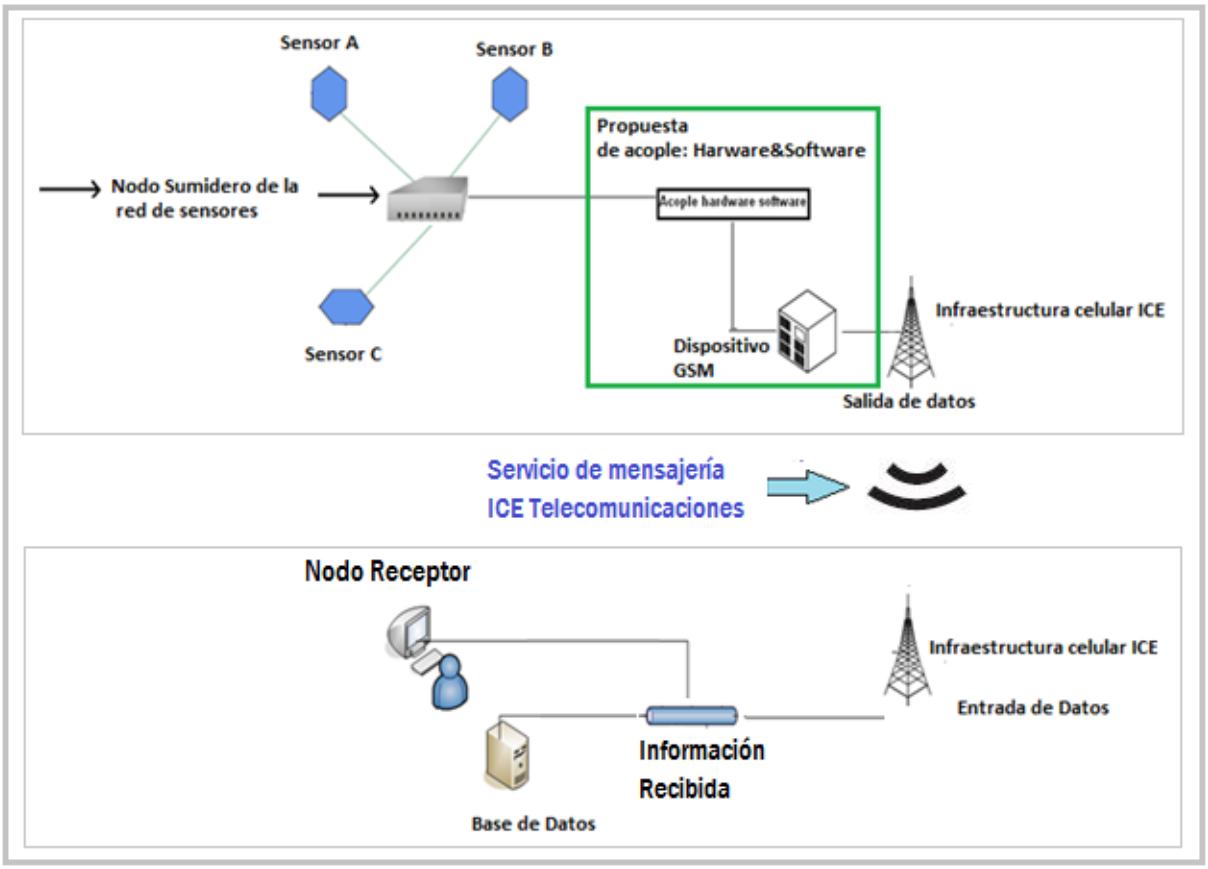


Figura 1. Esquema general del proyecto CRTECMote.

Como transmisor de la información de la plataforma, se selecciono un modem de comunicaciones globales de tipo USB de la marca EDGE, ya que es un modem de bajo consumo de potencia y además cumple con una serie de requisitos que lo hacen ideal para operar en la infraestructura celular del ICE.

Por último se escogió el controlador PIC32MX460512 de la empresa Microchip, debido a que es un controlador capaz de soportar el sistema operativo SIWA-RTOS y cumplir con las características solicitadas para el proyecto.

2 Capítulo 2: Meta y Objetivos

2.1 Meta

Lograr que el proyecto de redes inalámbricas de sensores CRTECMote cuente con un sistema de comunicación de larga distancia utilizando para ello la red de telefonía móvil que provee el ICE.

2.2 Objetivo general

Desarrollar el hardware y software necesarios para el acoplamiento correcto de un modem de comunicación de datos a la red inalámbrica de sensores CRTECMOTE.

2.3 Objetivos específicos

- Investigar las características operativas del modem de comunicación de datos proporcionado, para la identificación de los parámetros de configuración y control críticos del hardware.
- Desarrollar las bibliotecas de configuración y control de operación del modem, que se adapten correctamente con la red de telefonía móvil del ICE y el sistema operativo del nodo sumidero, para el envío de información hacia el sistema de fusión de datos.
- Adaptar el protocolo de comunicación de datos del nodo sumidero para que sea compatible con el sistema de fusión de datos y se logre el establecimiento de una comunicación adecuada entre estos puntos.

3 Capítulo 3: Marco Teórico

3.1 Descripción del sistema a mejorar, CRTECMote

CRTEMote es un proyecto diseñado como una plataforma de redes inalámbricas que permite monitorear variables de tipo ambiental tales como temperatura, humedad, caudal de agua, entre otros.

El proyecto cuenta con un nodo que se encarga de recopilar datos contenidos en una red de sensores. Esta red está diseñada bajo el protocolo MIWI creado por la compañía Microchip y su función principal se basa en organizar el tráfico de información generada desde los sensores. También se encarga de generar las tramas de información que serán enviadas al nodo sumidero encargado de colocar esta información en la etapa de comunicaciones a través de una red WAN.

Además la plataforma actualmente cuenta con un nodo receptor de datos que básicamente es una base de datos que se encarga de recibir y organizar la información enviada desde la red de sensores. Esta base cuenta con una interfaz grafica que permite visualizar los datos recibidos según su fecha, hora e identificación del valor de los sensores que se encuentran bajo estudio.

Por el momento, el sistema cuenta con un sistema operativo de tiempo real que garantiza un bajo consumo de potencia, lo cual es vital para el proceso de monitoreo remoto de variables. Este sistema operativo se denomina SIWA RTOS y está diseñado para adaptarse a las aplicaciones de multitarea necesarias para dar soporte a la plataforma.

En esta etapa de desarrollo se espera contar con una versión comercial del producto ya que sus últimas etapas están por concluir. Por lo tanto se podrá crear una interfaz visual comerciable que permita a CRTECMote ser parte y competencia de los productos existentes en el mercado que cumplen con este tipo de labor.

3.2 Antecedentes Bibliográficos

En parte de la investigación realizada para efectos de recopilar información para este proyecto se encontraron proyectos que efectivamente envían un mensaje de texto utilizando un modem de tipo GSM-RS-232. Sin embargo ninguno de ellos fue utilizado en algún aspecto como referencia en para el desarrollo de este proyecto, ya que ninguno cumple con las características ya especificadas de una interfaz de tipo USB y transmisión de datos en la banda de los 850Mhz correspondiente a las líneas de última generación 3G.

El aporte bibliográfico de mayor aporte para el desarrollo de este proyecto definitivamente fue la extensa documentación encontrada sobre el protocolo USB 2.0. Esta información fue recopilada desde el sitio web del estándar y amplia documentación sobre el protocolo fue recopilada desde la empresa Microchip específicamente en el sitio web en el apartado USB.

3.3 Descripción de los principales principios físicos y/o electrónicos relacionados con la solución del problema

3.3.1 Protocolo USB 2.0

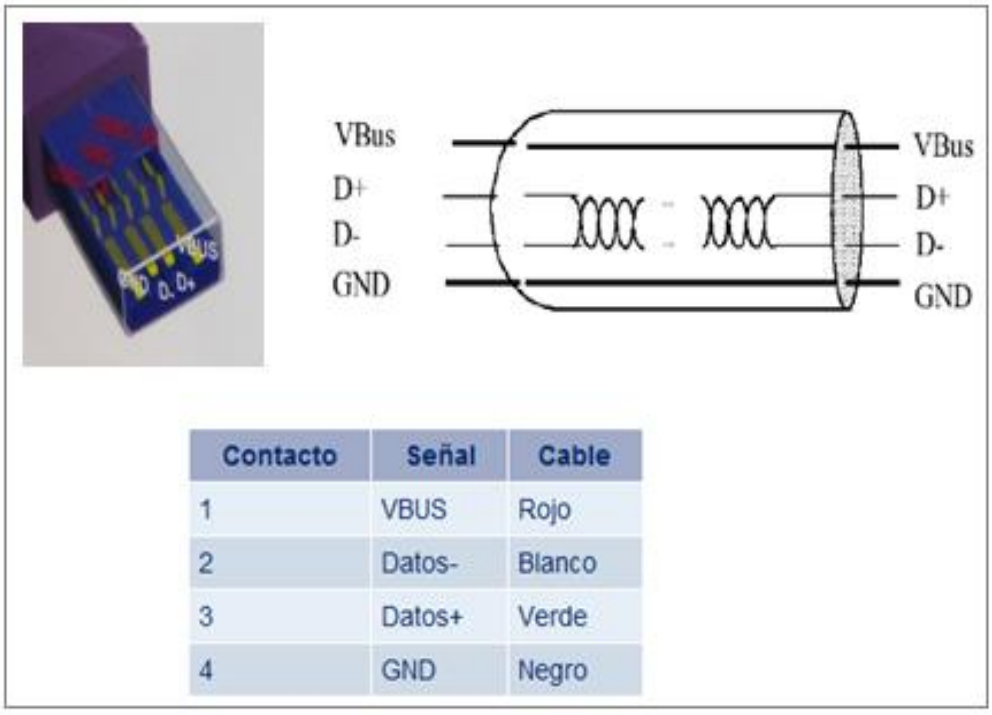
- **USB** (**Universal Serial Bus**) es una especificación que permite establecer una comunicación punto a punto entre dispositivos y sistemas administradores de recursos (USB HOST) de un computador.[19]
- El protocolo surgió como respuesta a una serie de limitantes presentadas por los protocolos de comunicación serie tales como puerto serie (**RS-232**) y puerto paralelo (***EPP***), los cuales presentaban muchos problemas al utilizarlos a alta velocidad debido a fenómenos parásitos producidos a altas frecuencias de transmisión.

- El protocolo fue diseñado en conjunto por un grupo de investigadores conformado por un conjunto de empresas líderes en alta tecnología: Hewlett-Packard Company, Intel Corporation, LSI Corporation, Microsoft Corporation, NEC Corporation and ST-Ericsson.

- Transceptores USB: consisten en módulos capaces de decodificar la información transmitida desde un dispositivo USB hasta un dispositivo USB HOST. [16]

- El tipo de transmisión de datos es Half-Duplex debido a que el envío de datos es bidireccional pero no simultáneo.

- **El cable USB:** consiste de cuatro conductores, dos conductores de potencia y dos de señal, D+ y D-. Los cables de media y alta velocidad están compuestos por un par trenzado de señal, además de GND(Tierra) y Vbus. [2]



Contacto	Señal	Cable
1	VBUS	Rojo
2	Datos-	Blanco
3	Datos+	Verde
4	GND	Negro

Figura 2 Diseño interno de las señales de control del bus USB [2].

- Un dispositivo USB está conformado por 2 tipos de unidades de almacenamiento de datos, dedicadas exclusivamente para brindar y recibir información.
- La primer unidad se denomina **Descriptores de Unidad** y es utilizada para almacenar la información acerca del dispositivo.
- La segunda unidad de almacenamiento y salida de datos se denomina **ENDPOINT**, y esta es utilizada como buffer de recepción y envió de datos desde el dispositivo USB hasta el HOST USB.
- Es importante destacar que únicamente los ***dispositivos USB*** poseen endpoints, así como descriptores; un host USB carece de ellos. [15]

A continuación se presente el diagrama de bloques que describe la estructura interna de un dispositivo USB. [2]

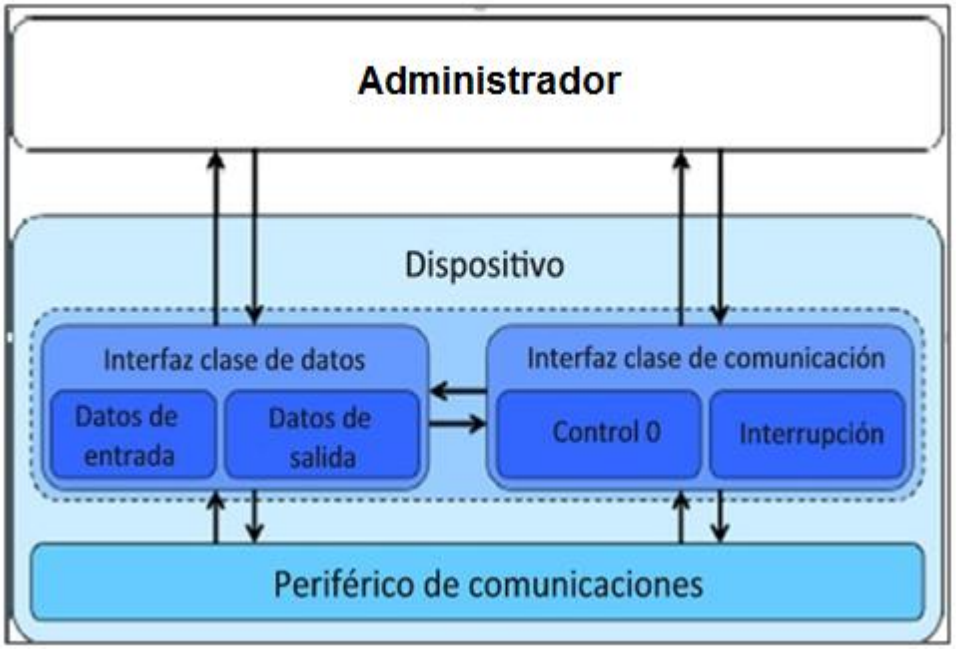


Figura 3 Diagrama de bloques de un dispositivo USB.

- Los dispositivos pueden ser separados según su funcionalidad por medio de su Class Descriptor.

Tabla 1 Códigos de un descriptor de Comunicaciones [18].

Offset	Field	Size	Value	Description
4	*bDeviceClass*	1	02h	Communication Device Class code as defined in Table 14.
5	*bDeviceSubClass*	1	00h	Communication Device Subclass code, unused at this time.
6	*bDeviceProtocol*	1	00h	Communication Device Protocol code, unused at this time.

- Tambien pueden ser seleccionados según su interfaz de comunicaciones ya que esta puede ser de diferentes tipos según sea la aplicación.

Tabla 2 Etiquetas de un descriptor de Comunicaciones [18].

Offset	Field	Size	Value	Description
5	*bInterfaceClass*	1	Class	Communication Interface Class code, as defined in Table 15.
6	*bInterfaceSubClass*	1	SubClass	Communication Interface Class SubClass code, as defined in Table 16.
7	*bInterfaceProtocol*	1	Protocol	Communication Interface Class Protocol code, which applies to the subclass, as specified in the previous field, is defined in Table 17.

- A cada una de las interfases se les asigna un código de tipo hexadecimal con el objetivo de agilizar el proceso de enumeración llevado a cabo por el Host USB.

Tabla 3 Descriptor de clase [18].

Offset	Field	Size	Value	Description
5	*bInterfaceClass*	1	0Ah	Data Interface Class code, as defined in Table 18.
6	*bInterfaceSubClass*	1	00h	Data Class SubClass code.
7	*bInterfaceProtocol*	1	Protocol	Data Class Protocol code, which applies to the subclass, as specified in the previous field, is defined in Table 19.

- También esxiste asignaciones de bits en los descriptores que informan a el Host USB sobre las cacidades del dispositivo.

Tabla 4 Asignaciones de valores de clase [18].

Offset	Field	Size	Value	Description
0	*bFunctionLength*	1	Number	Size of this functional descriptor, in bytes.
1	*bDescriptorType*	1	Constant	CS_INTERFACE
2	*bDescriptorSubtype*	1	Constant	Abstract Control Management functional descriptor subtype as defined in Table 25.
3	*bmCapabilities*	1	Bitmap	The capabilities that this configuration supports. (A bit value of zero means that the request is not supported.) D7..D4: RESERVED (Reset to zero) D3: 1 - Device supports the notification Network_Connection. D2: 1 - Device supports the request Send_Break D1: 1 - Device supports the request combination of Set_Line_Coding, Set_Control_Line_State, Get_Line_Coding, and the notification Serial_State. D0: 1 - Device supports the request combination of Set_Comm_Feature, Clear_Comm_Feature, and Get_Comm_Feature. The previous bits, in combination, identify which requests/notifications are supported by a Communication Class interface with the SubClass code of Abstract Control Model.

3.3.2 Sistemas Embebidos

Un sistema embebido es un dispositivo utilizado para controlar aplicaciones diseñadas para manejar una o más tareas, generalmente son muy utilizados en industrias y en equipos comerciales tales como teléfonos móviles, tarjetas de red inalámbricas, sistemas de posicionamiento global, equipos de cómputo, entre otros [7].

Además, su diseño permite tener una gran flexibilidad para manejar un gran rango de usuarios, desde grandes industrias hasta usuarios de propósito general. Los sistemas embebidos son controlados por un procesador central, este mismo puede variar desde un microcontrolador hasta un DSP (Procesador Digital de Señales).

Entre sus principales ventajas se encuentra su capacidad de soportar sistemas operativos y por ende la posibilidad de realizar numerosas tareas con un corto periodo de ejecución. Además estos sistemas operativos pueden ser compatibles con muchas aplicaciones, lo cual genera una alta flexibilidad en la funcionalidad de un sistema embebido.

3.3.3 Arquitectura General GSM Modem

Un Modem GSM es un dispositivo diseñado para la transmisión y recepción de señales bajo el estándar del Sistema Global para las Comunicaciones Móviles (Global Sistem for Mobile, en su traducción al inglés) mediante procesos de modulación y demodulación de señales.

Modular una señal consiste en modificar alguna de las características de esa señal, llamada portadora, de acuerdo con las características de otra señal llamada moduladora.

La modulación facilita el proceso de transmisión de las señales eléctricas por diferentes medios de propagación. [9] La demodulación permite cifrar la información recibida (modulada) y transmitirla a cualquier etapa de control para ser procesada (microcontrolador).

Existen diferentes diseños de arquitecturas basadas en brindar una mejor interfaz con el usuario, partiendo desde teclados, pantallas, cámaras de video, speaker o alta voz; esto permite diversificar la oferta de aplicaciones dependiendo del interés del usuario.

A continuación se muestra un diagrama de bloques con una propuesta de arquitectura de la empresa ERICSSON para un sistema embebido basado en comunicaciones bajo el estándar GSM. [3]

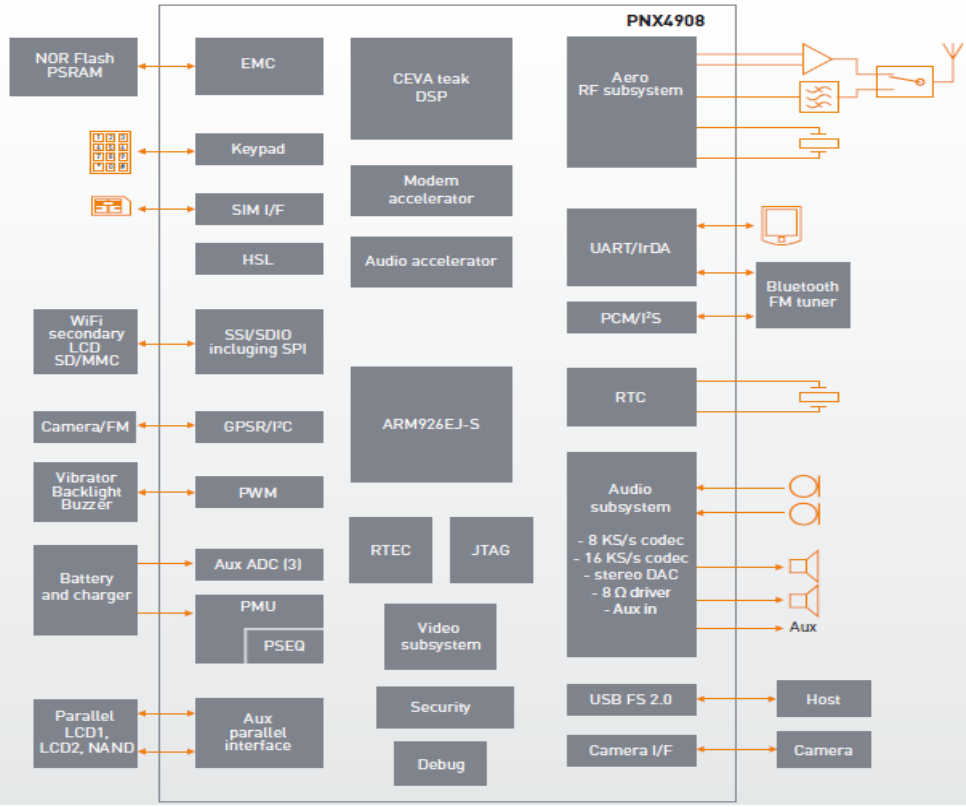


Figura 4 Diagrama de bloques de la arquitectura Modem-GSM-Ericsson. [3]

3.3.4 Comandos AT

Los comandos AT fueron desarrollados en 1977 por Dennis Hayes como un interfaz de control y comunicación con un MODEM telefónico de tipo analógico. Más adelante, con el avance del la tecnología digital, fueron las compañías Microcomm y US Robotics las que siguieron desarrollando y expandiendo el juego de comandos hasta universalizarlos en MODEMS digitales. La abreviatura AT proviene de la palabra Atención (Atenttion de su traducción al inglés). [21]

Estos comandos se utilizan para realizar una comunicación entre cualquier dispositivo que actúe como host de la aplicación y el modem GSM. [12]

A continuación se presentan algunas de las acciones que se pueden realizar utilizando los comandos AT:

a) Establecer conexión de datos o voz desde un modem remoto ATD, ATA)
b) Enviar y recibir fax (ATD, ATA, AT+F*).[12]
c) Recuperar configuraciones de fabrica para varias configuraciones del modem (AT+CRES) y (AT+CSAS). [12]

3.3.5 Centro de mensajes de texto

El elemento más utilizado en el servicio de mensajería es el Centro de Mensajes (MXE) ya que entre sus múltiples funciones se encuentra la de envío y recepción de mensajes de texto.

De manera general es un nodo que provee servicio de voz integrada, fax, correo electrónico y mensajería instantánea. Específicamente el MXE se utiliza para el correcto manejo del servicio de mensajes cortos por medio del Centro de Mensajes Cortos (SMSC). El SMSC se conecta a la base de datos HLR por medio del Punto de Transferencia de Señales (STP). [9]

3.3.6 Arquitectura de la red de telefonía móvil

La arquitectura de la red del sistema global para comunicaciones móviles se divide en tres grandes sistemas: el Sistema de Conmutación (SS), el Sistema de Base-Estación (BSS) y el Sistema de Operación-Soporte (OSS).

- El sistema de conmutación es el responsable de ejecutar el proceso de la llamada y realizar las funciones propias del operador.
- Todas las funciones relacionadas con radio frecuencia se realizan en este módulo.
- Un Centro de Operaciones y Mantenimiento (OMC) se conecta a todo el equipo del Sistema de Conmutación y al Sistema Base-Estación. El proceso de implementar el OMC se conoce como el Sistema de Operación y Soporte (OSS)

La figura. 5 muestra el diagrama de bloques de una arquitectura de la red GSM [9].

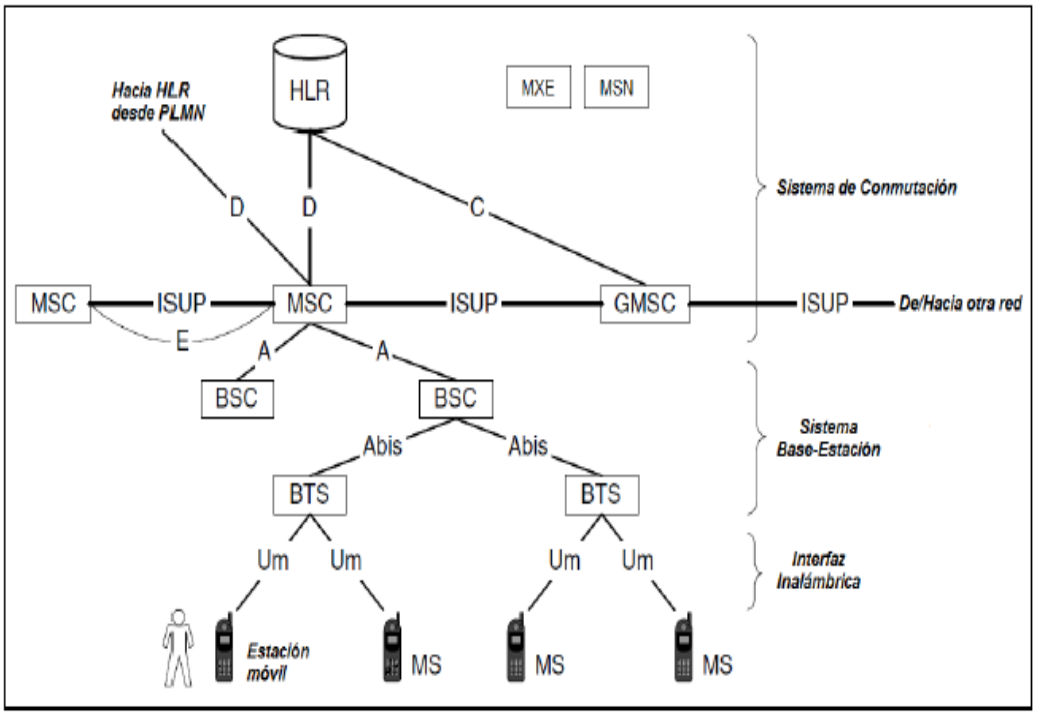


Figura 5 Arquitectura de para un sistema global para comunicaciones móviles [9].

3.3.7 Descripción de sistema operativo SIWA RTOS

- **Definición**

  Un Sistema Operativo es el programa encargado de ejercer el control y coordinar el uso del hardware entre diferentes programas de aplicación y los diferentes usuarios. Es un administrador de los recursos de hardware del sistema. [7]

- **Características Generales**

  ✓ Multi-tareas, ya que administran el tiempo de ejecución para cada aplicación

- ✓ Muchos tienen tiempos de respuesta predecibles para eventos electrónicos.

- **Diseño**

  1. Un sistema operativo guiado por eventos sólo cambia de tarea cuando un evento necesita el servicio.
  2. Un diseño de compartición de tiempo cambia de tareas por interrupciones del reloj y por eventos.

- **Interrupciones:** Para que el programa cumpla con su cometido de ser tiempo real es necesario que el sistema atienda la interrupción y procese la información obtenida antes de que se presente la siguiente interrupción. Como el microprocesador normalmente solo puede atender una interrupción a la vez, es necesario que los controladores de tiempo real se ejecuten en el menor tiempo posible.

  Esto se logra no procesando la señal dentro de la interrupción, sino enviando un mensaje a una tarea o solucionando un semáforo que está siendo esperado por una tarea. El programador se encarga de activar la tarea y esta se encarga de adquirir la información y completar el procesamiento de la misma. [8]

- **Calendarización**

  El reto principal de la calendarización es el de crear una estructura diseñada para minimizar el tiempo invertido para ejecutar una aplicación en el peor de los casos. En los diseños típicos, una tarea tiene tres estados: ejecución, preparada y bloqueada.

  La mayoría de las tareas están bloqueadas casi todo el tiempo. Solamente se ejecuta una tarea por UCP. La lista de tareas preparadas suele ser corta, de dos o tres tareas como mucho [8].

- **Comunicaciones**

  El sistema operativo crea la posibilidad de tener varios puertos de comunicación, por ejemplo comunicación de tipo USB, Ethernet, Serial, Wireless, permitiendo una buena administración en tiempo real de los periféricos en uso. [8]

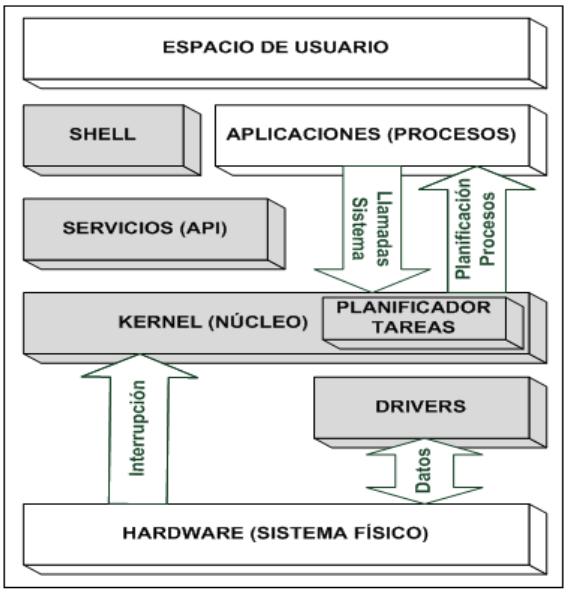


Figura 6 Diagrama de bloques del sistema operativo SIWA RTOS.

4 Capítulo 4: Procedimiento Metodológico

4.1 Investigación y Diseño

4.1.1 Analizar el protocolo USB para determinar los pasos a seguir a la hora de controlar un dispositivo USB.

Se analizó el protocolo USB utilizando el texto USB Complete [15] y las notas de aplicación brindadas por la empresa Microchip, obteniendo de esta manera la información necesaria para empezar a configurar el controlador PIC32MX460512 con la pila de aplicaciones USB CDC-Host. En esta parte de la investigación se tomaron las notas correspondientes de los procesos de enumeración de un dispositivo USB con el objetivo de reconocer cada uno de estos procesos en el momento de conectar el dispositivo al controlador.

Además, se hizo el estudio correspondiente a los procesos de configuración de un Modem USB por medio del modelo de control ACM (Abstract Control Model). Por medio de este estudio se reconocieron los procesos a seguir para configurar un dispositivo Modem USB en modo ACM-Serial Emulation para poder enviar los comandos AT de control por medio de la etapa de datos del controlador del dispositivo.

4.1.2 Caracterizar el controlador PICMX460512 para sustentar su capacidad de funcionar como un Host USB.

Por medio de la información recopilada en el manual de usuario del controlador PIC32MX460512 se determino que módulo USB OTG es capaz de administrar un dispositivo cuyo consumo de corriente alcanza un máximo de 50mA.

También se tomaron las notas correspondientes a la configuración del modulo tanto como en la interfaz de hardware, en el cual se cumplió con remover el jumper JMP2

de la tarjeta de desarrollo PIC-USB modelo DM320003 para eliminar la posibilidad de que se dañara el controlador por corrientes de retroalimentación.

A nivel de software se configuró el modulo USB-OTG por medio de la herramienta USBTools: USB-CONFIG. Por medio de esta configuración se seleccionó el modo de operación del módulo como Host USB, en la sección de resultados se presentarán las configuraciones utilizadas en esta aplicación.

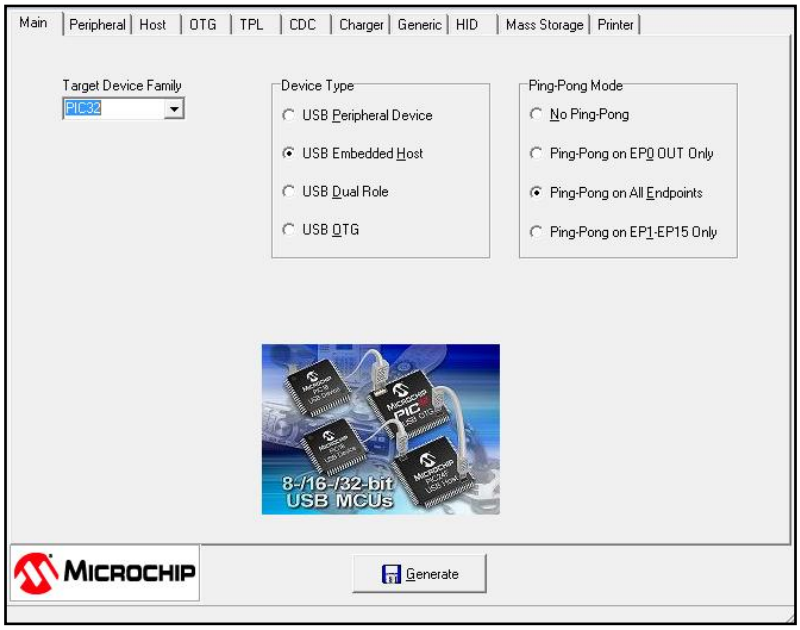


Figura 7 Sofware de Configuración del Módulo USB para controladores PIC32MX460512

4.1.3 Seleccionar un Modem GSM de tipo USB que cumpla con el estándar USB 2.0.

En esta etapa se selecciono el dispositivo Modem-USB de la marca EDGE ya que cumple con el estándar USB 2.0, y además puede ser controlado por el set de comandos V2.5ter AT Hayes.

Otro de los factores decisivos para seleccionar este dispositivo fue el de su capacidad de operar en la bandas de 850Mhz y 1900Mhz, esto aseguró que el

dispositivo pudiera ser utilizado bajo la infraestructura de comunicación GSM-3G de el Instituto Costarricense de Electricidad.

4.1.4 Investigar los comandos AT Hayes con el objeto de seleccionar los comandos a utilizar para enviar y recibir mensajes de texto a través de una red GSM.

Por medio de una revisión bibliográfica se recopilo el manual de usuario de los comandos AT permitidos por el Modem EDGE-USB [12]. En este manual se analizaron los comandos utilizados para configurar el modem en modo de envío y recepción de mensajes de texto. Con el estudio de estos comandos, se crearon las propuestas de los diagramas de flujo que controlaran el envío y recepción de los mensajes desde el controlador PIC32MX460512.

4.2 Implementación y Desarrollo

4.2.1 Adaptar aplicación de CDC-USB-Host que brinda la empresa Microchip a la tarjeta de desarrollo PIC-USB modelo DM-320003

El código de demostración de USB-CDC Host que viene incluido en la pila de aplicaciones USB de la empresa Microchip, se utilizó como base de operación para comenzar a trabajar con la etapa de programación del controlador PIC32MX460512. Se reconocieron las etapas de enumeración propuestas por el estándar USB 2.0 y se comenzó a filtrar paso a paso cada uno de los procesos realizados por la pila de aplicaciones CDC-USB-HOST por medio de herramientas de depurador de código diseñadas para los controladores de la empresa Microchip.

Con la herramienta de depurador se insertaron mensajes de control en cada una de las etapas del código CDC-USB-HOST para analizar el estado de cada una de las peticiones establecidas por el protocolo USB 2.0. También se insertaron mensajes de confirmación de estado de los registros involucrados en cada proceso de administración de datos de entrada y salida.

4.2.2 Administrar un el Modem EGDE-USB utilizando un controlador PIC32MX460512

Una vez concluida la etapa de enumeración del dispositivo se procedió a configurar el modem por medio de modelo de control ACM (Abstract Control Model) diseñado en la etapa de investigación. Este diseño se creó de configuración en base al la especificación USB CDC Class [18] y en base a analizadores de protocolos USB, específicamente USBLyser Y USBTrace en sus versiones de prueba.

Añadiendo a lo anterior, después de configurar con éxito el Modem USB se implementaron las secuencias de inicialización y control del dispositivo; esto con el objetivo de mantener bajo control el estado del Modem en caso de desconexión, errores en el bus o peticiones incorrectas en dirección USB Host- USB Device.

4.2.3 Análisis de las rutinas de programación que permiten enviar y recibir mensajes de texto a través de un modem EDGE-USB utilizando Hiperterminal de Windows

En este punto se verificaron las rutinas planteadas en la etapa de investigación por medio del software de comunicaciones del sistema operativo de Windows Hiperterminal. Con esta aplicación se creó una simulación de la rutina de envió de mensajes de texto mediante el uso de los comandos AT y se obtuvieron conclusiones vitales para el desarrollo de las rutinas de servicio de mensajería que se crearon para el controlador PIC32MX460512.

También, se tomaron muestras de tiempo de respuesta del modem a distintas solicitudes por parte del Host USB. Estos registros de tiempo fueron la base de tiempo para el desarrollo de las rutinas de mensajería.

4.2.4 Escribir una aplicación en el lenguaje de programación C que permita controlar el PIC32MX460512 para el envío y recepción de mensajes de texto.

Para esta parte del procedimiento se implementaron las rutinas planteadas para el envío y recepción de mensajes de texto en el controlador PIC32MX460512 utilizando las estimaciones de tiempo recopiladas en el análisis de las secuencias por medio de Hyperterminal de Windows.

Una vez finalizada la aplicación se procedió a graficar los consumos de potencia para diferentes casos de envío y recepción de mensajes de texto. También se realizaron comprobaciones de cantidad de errores producidos en el bus durante todo el proceso de ejecución del programa con el fin de cuantificar la eficiencia de las peticiones de entrada y salida de datos que el bus USB.

4.2.5 Migrar la aplicación descrita en el punto 4.2.4 a un sistema operativo de tiempo real SIWA RTOS

Se procedió a migrar la pila de aplicación de USB-CDC en conjunto con la aplicación de envío y recepción de mensajes de texto a la última versión del sistema operativo SIWARTOS el objetivo de compilar y correr con éxito ambos códigos en una sola aplicación.

El proceso de unir ambas aplicaciones se llevó a cabo mediante un prueba de envió de un mensaje de texto mediante la creación de una tarea en el calendarizador del sistema operativo, cuyo objetivo fue el de inicializar el modem USB y configurarlo para mantenerlo en línea y esperar a que el proceso o la tarea de envío de un mensaje de texto se ejecute.

4.3 Verificación de solución

4.3.1 Verificación del funcionamiento el controlador PIC32MX460 como host USB

Por medio de herramientas de depuración de código e indicadores de tipo lumínico (leds), se crearon etapas de comprobación de código para verificar que efectivamente el código produjera los resultados esperados desde el proceso de enumeración hasta el envío de un mensaje.

4.3.2 Verificación del funcionamiento la aplicación creada de envío y recepción de mensajes de texto

La verificación de envío de los mensajes de texto se llevo a cabo por medio de una solicitud de tipo USB-Host a USB-Device, en la cual se preguntó al modem sobre el de estado del mensaje enviado; en base a estas respuesta se planteó la ejecución del programa.

De igual manera, se procedió a verificar la recepción de mensajes de texto por medio de una secuencia que valida la recepción exitosa de mensajes y se adaptaron mensajes de tipo lumínico de tipo led para informar al usuario sobre el estado actual del sistema

4.3.3 Comparar los mensajes de texto enviados desde el controlador PIC32MX460 con los mensajes recibidos en la terminal de recepción.

La comparación se llevo a cabo por medio de estaciones de recepción de datos con las cuales se logro autenticar que el dato enviado por el controlador es los mismos que se envió desde el nodo central con el controlador PIC32MX460512

5 Capítulo 5: Descripción Detallada de la Solución

A continuación se presentará la información correspondiente al desarrollo de la solución propuesta para el problema planteado.

5.1 Descripción General

Para la solución del problema se utilizó un controlador PIC32MX460512. Esto debido a su diseño como administrador de dispositivos de tipo USB y también por la capacidad de soportar un sistema operativo de tiempo real como SIWA-RTOS.

La aplicación USB seleccionada para la solución fue la de tipo CDC-USB-HOST de la empresa Microchip, ya que es una buena aproximación al comportamiento que se desea del dispositivo EDGE-USB. La aplicación CDC-USB-HOST USB funciona bajo el modelo USB Embedded Host Firmware de la figura 6. En este diagrama se basó la solución planteada al problema.

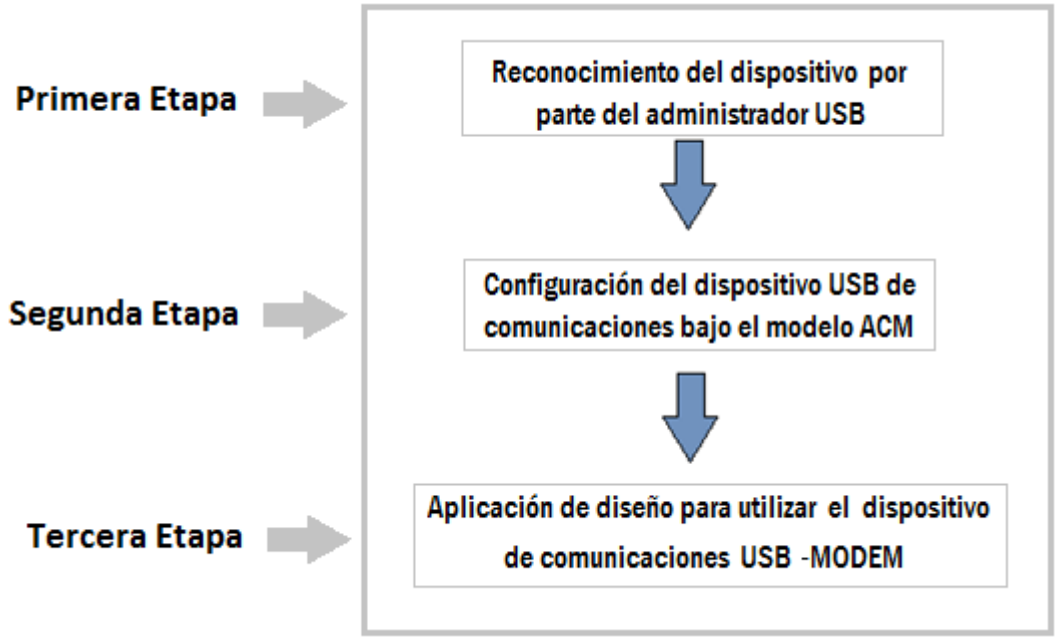


Figura 8 Arquitectura del la pila de aplicaciones Host-USB de Microchip

El dispositivo seleccionado como interfaz de comunicaciones fue un Modem de la marca EDGE, ya que está diseñado para ser controlado por comandos AT Hayes y también por funcionar en las frecuencias de operación de las redes de telefonía celular del Instituto Costarricense de Electricidad.

Por último se selecciono el sistema operativo SIWA-RTOS ya que es un sistema diseñado para ser de bajo consumo de potencia, lo cual es un factor determinante en el diseño y desarrollo de futuras aplicaciones de la plataforma de redes inalámbricas CRTECMote.

5.2 Primera Etapa: Reconocimiento del dispositivo de comunicaciones Modem USB por parte del controlador Host USB PIC32MX460512

5.2.1 Reconocimiento por interrupción de hardware

Para verificar el primer estado de reconocimiento del modem EDGE-USB, se localizó la primera bandera de interrupción de conexión del dispositivo, con la cual se logró comprobar que efectivamente la lógica de resistencia de pull up encargada de informar al controlador sobre una nueva conexión en el bus USB estaba funcionando.

```
if (((usbHostState & STATE_MASK) != STATE_DETACHED) && !U1IRbits.ATTACHIF)
```

Figura 9 Instrucción de control para la interrupción por hardware.

5.2.2 Reconocimiento a nivel de software

Luego de verificar que la interrupción de hardware se llevó a cabo con éxito, se procedió a identificar los estados de enumeración por software que contempla el protocolo USB 2.0; específicamente la aplicación USB-CDC Host. En el diagrama de la figura 9 se presenta la secuencia completa de los pasos de enumeración de un dispositivo USB conectado al controlador PIC32MX460512. Cada uno de estos procesos fue etiquetado con mensajes equivalentes a break-points de programación,

con el objetivo de imprimir en el Output Window de MPLAB el estado del proceso y controlar el estado del programa. Ver figura 8.

```
#define SendMessage(l,m)          DBPUTS(m)
```

Figura 10 Etiqueta de control de flujo de programa

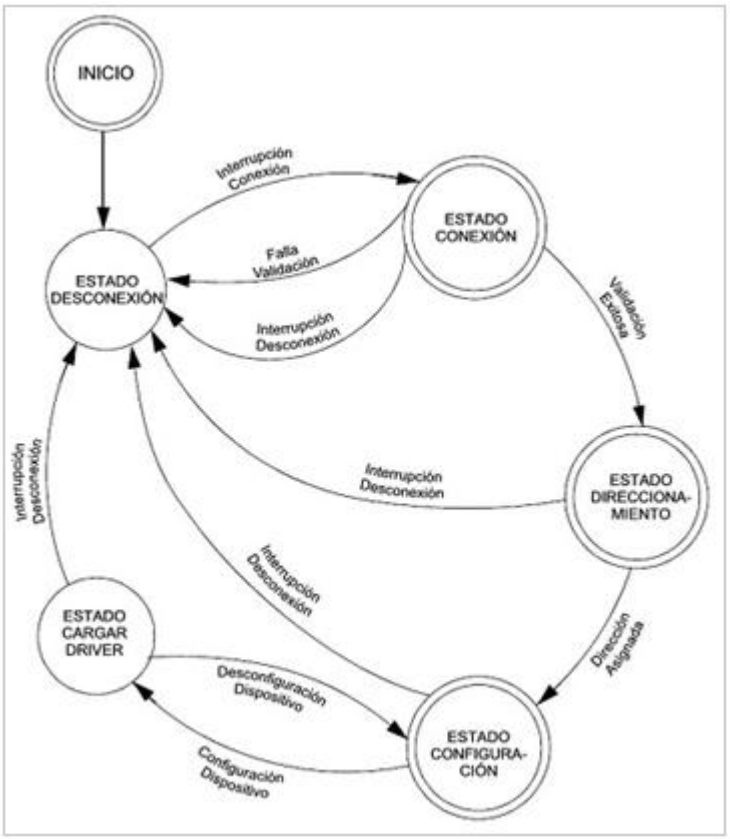


Figura 11 Diagrama de flujo de la enumeración de un dispositivo USB

Además se insertaron etiquetas de depuración de datos (figura 10), las cuales se crearon para conocer el valor de los registros asociados a las solicitudes de tipo USB-Host > USB Device y viceversa.

```
#define IMPRIME()                 DBPRINTF()
```

Figura 12 Etiqueta de control de valor de registros.

Estas etiquetas se insertaron en los procesos de enumeración descritos en el esquema de la figura 9, y de esta manera se logró controlar el estado de reconocimiento del dispositivo por parte del Host USB y decidir si el dispositivo estaba listo para ser configurado.

5.2.3 Petición de descriptores del modem EDGE USB

Como parte de los resultados obtenidos en la etapa de investigación, se determinó que un dispositivo USB enumerado como Comunication Device Class (Dispositivo de Comunicaciones), está diseñado para identificarse como un dispositivo de almacenamiento masivo de datos.

Por lo tanto se procedió a reconocer cada una de las etapas de petición de descriptores con el objetivo de identificar el paso en el cual el Host USB decide qué tipo de interface se utilizará para controlar el dispositivo y así poder seleccionar la interfaz de comunicación de datos en lugar de la interfaz de almacenamiento de datos. A continuación se presenta el desglose de las peticiones de descriptores más relevantes del proceso de enumeración:

- **Obtener Descriptor del Dispositivo**

La solicitud se realizó como una transacción dirigida al End-Point cero, y seguidamente se verificó que los datos recibidos fueran congruentes con los datos recopilados en la etapa de investigación por medio de los analizadores de protocolo USBlyser y USBTrace.

```
case SUBSTATE_GET_DEVICE_DESCRIPTOR_SIZE:

_USB_InitControlRead( usbDeviceInfo.pEndpoint0, pEP0Data, 8, pEP0Data, 8 );
_USB_SetNextSubSubState();
SendMessage( 2, "HOST: Getting Device Descriptor size.\r\n" );
```

Figura 13 Código de revisión de descriptor de interfaz del dispositivo

- **Validar VID & PID, o Class Type**

Para realizar la validación del dispositivo se plantearon dos soluciones, la primera de ellas fue la de tratar de validarlo por sus etiquetas Vendor ID y Product ID (VID-PID). Los valores de las etiquetas VID-PID para el dispositivo de comunicaciones EDGE-USB fueron asignados como 0x0471 y 0x1237 de su valor en hexadecimal. Estos valores se introdujeron en la primera tabla de identificación del dispositivo, figura 12.

```
USB_TPL usbTPL[NUM_TPL_ENTRIES] =
{
//    VID & PID or                          Client
//    Class, Subclass & Protocol      Config Driver  Flags
    { INIT_VID_PID( 0x0471, 0x1237 ),   1,     0,    {TPL_SET_CONFIG} }  // Validación por VID-PID
};
```

Figura 14 Validación por etiquetas VID-PID

Una vez insertados los valores de las etiquetas, se procedió a verificar el proceso de reconocimiento de dispositivo por medio de una declaración global del código denominado *ALLOW GLOBAL_VID_PID.*

```
#ifdef ALLOW_GLOBAL_VID_AND_PID
if (((usbTPL[i].device.idVendor  == pDesc->idVendor ) &&
     (usbTPL[i].device.idProduct == pDesc->idProduct)) ||
    ((usbTPL[i].device.idVendor  == 0xFFFF) &&
     (usbTPL[i].device.idProduct == 0xFFFF)))
#else
if ((usbTPL[i].device.idVendor  == pDesc->idVendor ) &&
    (usbTPL[i].device.idProduct == pDesc->idProduct)   )
#endif
{
    SendMessage( 2, "HOST:Device validated VID-PID\r\n" );
    usbDeviceInfo.flags.bfUseDeviceClientDriver = 1;
}
```

Figura 15 Estructura de control de identificación del dispositivo por etiquetas VID-PID

El segundo intento de validación se creó a partir de una tabla que lista los códigos predeterminados por el estándar USB 2.0 para la clase de tipo Comunication Class Device (CDC).

Estos códigos identifican al dispositivo de una forma más concreta según la aplicación para la cual este diseñado.

```
USB_TPL usbTPL[] =
{
    { INIT_CL_SC_P( 2ul, 2ul, 1ul ), 0, 0, {TPL_CLASS_DRV} }, // Communication Interface
     // Communication Device/Interface Class , Abstract Control Mode , Common AT Commands
    { INIT_CL_SC_P( 0x0Aul, 0ul, 0ul ), 0, 0, {TPL_CLASS_DRV} }  // Data Interface
};
```

Figura 16 Validación por etiquetas por clase de dispositivo.

Este tipo de validación también fue autenticado en la rutina de identificación de dispositivo por medio de de la etiqueta "*Host: Device validated by class"*

```
if ((usbTPL[i].device.bClass    == pDesc->bDeviceClass    ) &&
    (usbTPL[i].device.bSubClass == pDesc->bDeviceSubClass) &&
    (usbTPL[i].device.bProtocol == pDesc->bDeviceProtocol)    )
{
                SendMessage( 2, "HOST:Device validated by class\r\n" );
    #ifdef DEBUG_MODE
        UART2PrintString( "HOST: Device validated by class\r\n" );
    #endif
    usbDeviceInfo.flags.bfUseDeviceClientDriver = 1;
}
```

Figura 17 Estructura de control de identificación del dispositivo por etiquetas clase

- **Crear un espacio en memoria para dar soporte al dispositivo**

Al localizar la etapa de direccionamiento, se reconoció como prioridad identificar la dirección de memoria que el controlador asignó al dispositivo. También se analizaron las peticiones específicas al END-POINT cero relacionadas con la etapa de direccionamiento y tamaño de espacio en memoria para controlar el dispositivo.

```
case SUBSUBSTATE_SEND_SET_DEVICE_ADDRESS:
            SendMessage( 2, "HOST: Setting device address.\r\n" );
    usbDeviceInfo.deviceAddress = USB_SINGLE_DEVICE_ADDRESS;

    // Set up and send SET ADDRESS
    pEP0Data[0] = USB_SETUP_HOST_TO_DEVICE | USB_SETUP_TYPE_STANDARD | USB_SETUP_RECIPIENT_DEVICE;
    pEP0Data[1] = USB_REQUEST_SET_ADDRESS;
    pEP0Data[2] = usbDeviceInfo.deviceAddress;
    pEP0Data[3] = 0;
    pEP0Data[4] = 0;
    pEP0Data[5] = 0;
    pEP0Data[6] = 0;
    pEP0Data[7] = 0;
    _USB_InitControlWrite( usbDeviceInfo.pEndpoint0, pEP0Data, 8, NULL, 0 );
    _USB_SetNextSubSubState();
    break;
```

Figura 18 Petición de direccionamiento de dispositivo

- **Iniciar Configuración en base a la información contenida en los descriptores**

Se procedió a autenticar el estado de configuración del dispositivo. Con este estado de configuración el controlador PIC32MX460512 verifica si es capaz de controlar el dispositivo USB por medio de los controladores previamente configurados para la aplicación. Esta petición de configuración llevó la etiqueta "*Host: Set Configuration*" y con ella se identifico si efectivamente el controlador PIC32MX460512 alcanzó a realizar la petición con éxito.

```
case SUBSUBSTATE_SEND_SET_CONFIGURATION:
    SendMessage( 2, "HOST: Set configuration.\r\n" );
    #ifdef DEBUG_MODE
        UART2PrintString( "HOST: Set configuration.\r\n" );
    #endif

    // Set up and send SET CONFIGURATION.
    pEP0Data[0] = USB_SETUP_HOST_TO_DEVICE | USB_SETUP_TYPE_STANDARD | USB_SETUP_RECIPIENT_DEVICE;
    pEP0Data[1] = USB_REQUEST_SET_CONFIGURATION;
    pEP0Data[2] = usbDeviceInfo.currentConfiguration;
    pEP0Data[3] = 0;
    pEP0Data[4] = 0;
    pEP0Data[5] = 0;
    pEP0Data[6] = 0;
    pEP0Data[7] = 0;
    _USB_InitControlWrite( usbDeviceInfo.pEndpoint0, pEP0Data, 8, NULL, 0 );
    _USB_SetNextSubSubState();
    break;
```

Figura 19 Petición de configuración del dispositivo

Por último, se insertó una etiqueta denominada "*HOST: Configuratión Complete. EDGE MODEM (GSM-SMS)*" con el objetivo de comprobar que la etapa de enumeración había concluido.

```
case SUBSUBSTATE_SET_CONFIGURATION_COMPLETE:
    SendMessage( 2, "HOST:Configuratión Complete. EDGE MODEM (GSM-SMS).\r\n" );
    // Clean up and advance to the next state.
    _USB_InitErrorCounters();
    _USB_SetNextSubSubState();
    break;
```

Figura 20 Estado de confirmación de la configuración del dispositivo

5.3 Segunda Etapa: Configuración del dispositivo Modem USB bajo el modelo ACM (Abstract Control Model-Serial Emulation)

5.3.1 Inicio de configuración del controlador del Modem EDGE-USB

Después de asegurar que el dispositivo EDGE-USB estaba enumerado correctamente, el siguiente paso de la solución se basó en configurar el modem EDGE-USB de tal manera que pudiese entender el estándar de V.25ter (AT Hayes). El primer paso de la configuración consistió en verificar que la inicialización la interfaz de comunicaciones se estaba llevando a cabo y analizar el valor asignado a las direcciones de los END-POINTS de entrada y salida de datos al dispositivo USB.

```
BOOL USBHostCDCInitialize( BYTE address, DWORD flags, BYTE clientDriverID )
{
    BYTE   *descriptor         = NULL;
    WORD    i                  = 0;
    WORD    endpointINsize     = 0;
    WORD    endpointOUTsize    = 0;
    BYTE    device             = 0;
    BYTE    endpointIN         = 0;
    BYTE    endpointOUT        = 0;
    BYTE    validCommInterface = 0;
    BYTE    validDataInterface = 0;
SendMessage( 2, "CDC: USBHostCDCInitialize\r\n" );
    #ifdef DEBUG_MODE
        UART2PrintString( "CDC: USBHostCDCInitialize(0x" );
        UART2PutHex( flags );
        UART2PrintString( ")\r\n" );
    #endif
```

Figura 21 Inicialización de la etapa de configuración del dispositivo.

El estudio de los valores registrados se hizo por medio de la secuencia de comandos presentados en la figura 20.

```
DBPRINTF ("bLenght:%x\r\n", descriptor[i+0]);
DBPRINTF ("bDescriptorType:%x\r\n", descriptor[i+1]);
DBPRINTF ("bInterfaceNumber:%x\r\n", descriptor[i+2]);
DBPRINTF ("bAlternateSetting:%x\r\n", descriptor[i+3]);
DBPRINTF ("bNumEndpoints:%x\r\n", descriptor[i+4]);
DBPRINTF ("bInterfaceClass:%x\r\n", descriptor[i+5]);
DBPRINTF ("bInterfaceSubClass:%x\r\n", descriptor[i+6]);
DBPRINTF ("bInterfaceProtocol:%x\r\n", descriptor[i+7]);
DBPRINTF ("biInterface:%x\r\n", descriptor[i+8]);
```

Figura 22 Depuración del descriptor de comunicaciones del dispositivo

5.3.2 Reestructuración del modelo ACM contenido la aplicación USB-CDC-HOST

El modelo de control del dispositivo ACM-Data Interface propuesto por la empresa Microchip, se escogió como una buena aproximación del comportamiento que se requería para administrar el dispositivo EDGE-USB. Sin embargo, este modelo presentó una desventaja ya que su función principal es llevar los datos desde el Host-USB hasta la interfaz de datos del dispositivo EDGE-USB. Esta desventaja obligó a reestructurar el modelo ACM-Data Interface a un modelo ACM-Serial Emulation.

Esta reestructuración de modelo fue necesaria ya que un modem USB ejecuta los comandos AT solo si estos son insertados en la etapa de control, y para ello existen dos formas de hacerlo. La primera de ellas es escribir los comandos directamente en el modulo de control por medio de peticiones al END-POINT cero. La forma de control se basa en escribir los comandos en la interfaz de datos y **crear un puente** entre la etapa de datos y el modulo de control; a este segundo método se le denomina ACM-Serial Emulation.

En la figura 21 se presenta la estructura interna de control un dispositivo de comunicaciones, y con la cual se planteo la reestructuración del modelo ACM-CDC-

USB. La secuencia indicada en la figura 21 indica el modelo planteado por la empresa Microchip en su aplicación USB-CDC-HOST el cual administra el dispositivo por medio de la **interfaz de datos**.

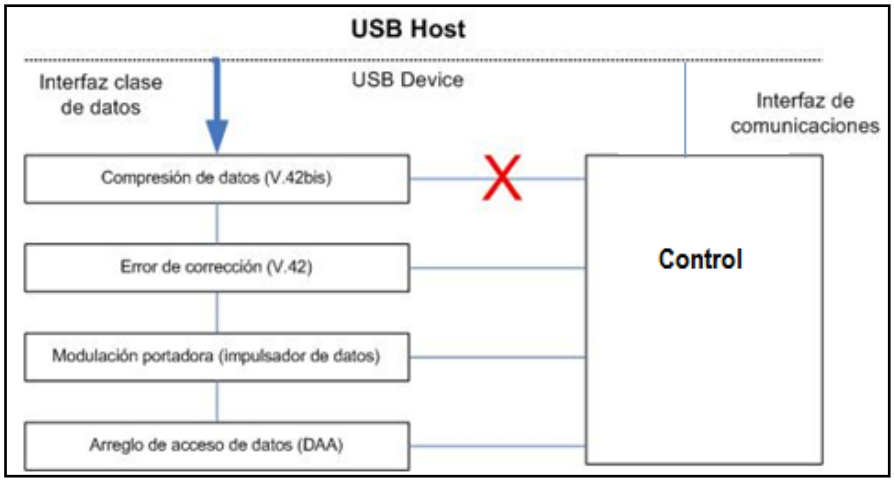


Figura 23 Modelo ACM-Data Interface

La propuesta de reestructuración se basó en el modelo Abstract Control Model Serial Emulation descrito en el documento de definición de clases de dispositivos de Comunicaciones de tipo USB [18], y su diseño se muestra en la figura 22. En esta figura se aprecia la unión de la interfaz de datos con la interfaz de control y comunicaciones.

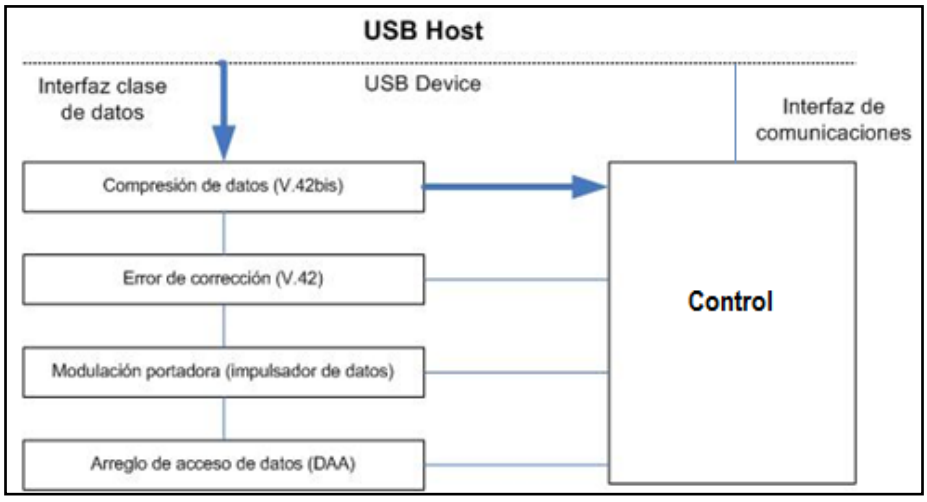


Figura 24 Modelo ACM-Serial Emulation

Para crear la reestructuración del modelo, se estudió el modelo ACM-Serial Emulation con las herramientas de análisis de protocolos USB llamadas USBLyser y USBTrace. El estudio consistió en analizar la etapa de configuración que ejecuta el programa de comunicaciones de Hyperterminal de Windows, ya que este funciona bajo el modelo ACM-Serial Emulation.

En el capítulo 6, específicamente en la sección 6.1.3, se presentan los resultados obtenidos con los analizadores de protocolos USBLyser y USBTrace. Con este estudio se reconocieron las solicitudes necesarias para configurar el dispositivo bajo el modelo ACM-Serial Emulation. En la tabla 5 se presenta el estándar de solicitudes que debe soportar un dispositivo de comunicaciones de tipo USB

Tabla 5 Estándar de peticiones que debe soportar un dispositivo de comunicaciones para ser configurado bajo el modelo ACM

Petición	Código	Descripción	Req/Op
Send Encapsulated Command	00h	Ejecuta el comando bajo el protocola establecido	Req
Get Encapsulated Response	01h	Solicita el comando bajo el protocola establecido	Req
Set_Line_Coding	20h	Configura DTE rate, bits de parada, numero de datos	Op
Get_Line_Coding	21h	Solicita DTE rate, bits de parada, numero de datos	Op
Set__Control_Line_State	22h	RS-232 señal utilizada para informar DCE presente	Op
Send_Break	23h	Envía la señal de parada	Op

El diagrama de flujo que se presenta en la figura 23 se describe la secuencia de comandos de configuración enviados al modem EDGE-USB bajo el modelo **ACM-Data Interface**.

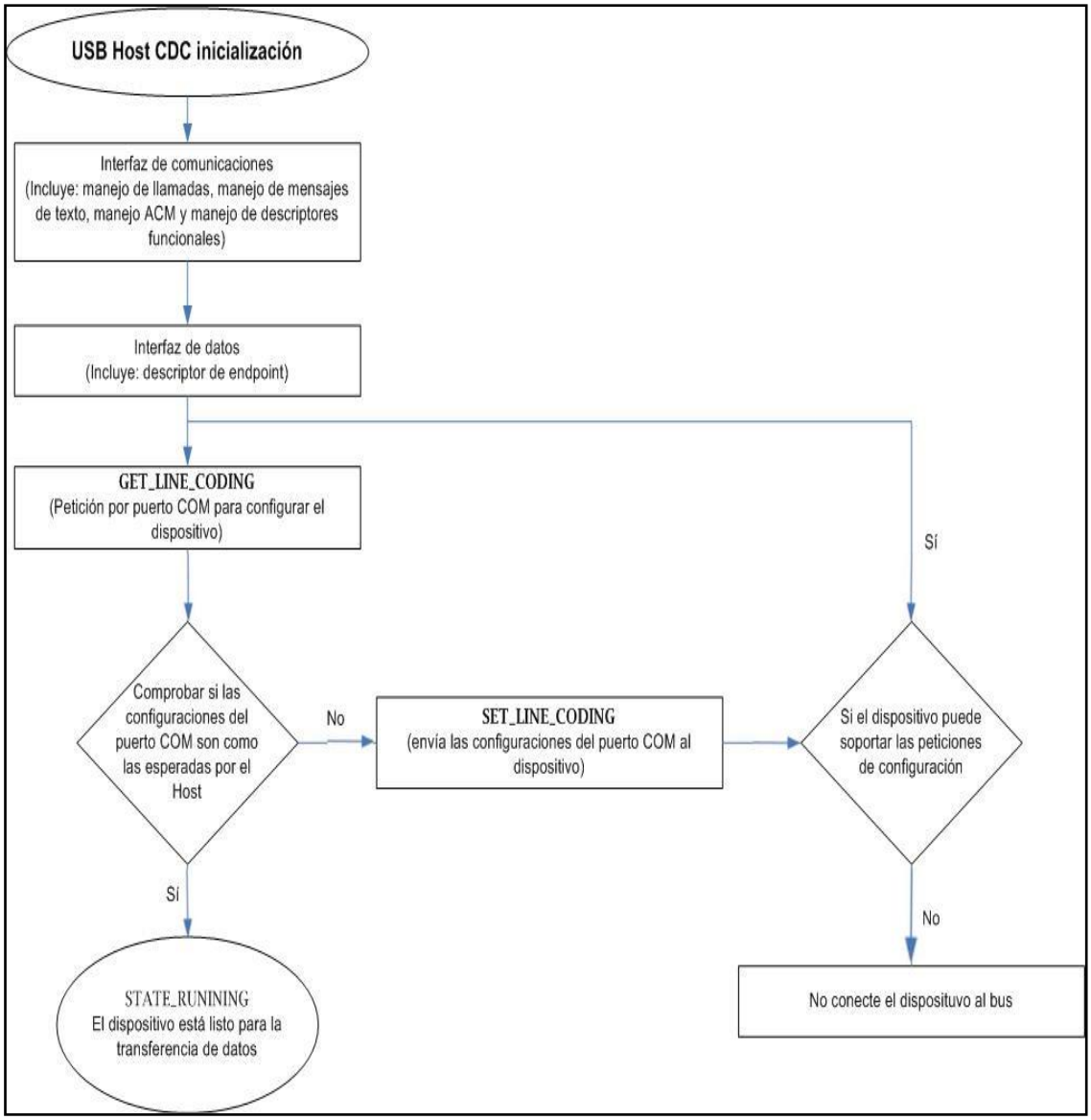


Figura 25 Esquema de configuración bajo el modelo AMC-Data Interface.

De esta manera se creó el diagrama de flujo de la figura 24 que representa la secuencia de peticiones involucradas en el modelo ACM-Serial Emulation y con este modelo se configuró el modem EDGE-USB seleccionado para este proyecto.

En la figura 24 se presenta la reestructuración propuesta para el modelo **ACM-Serial Emulation.**

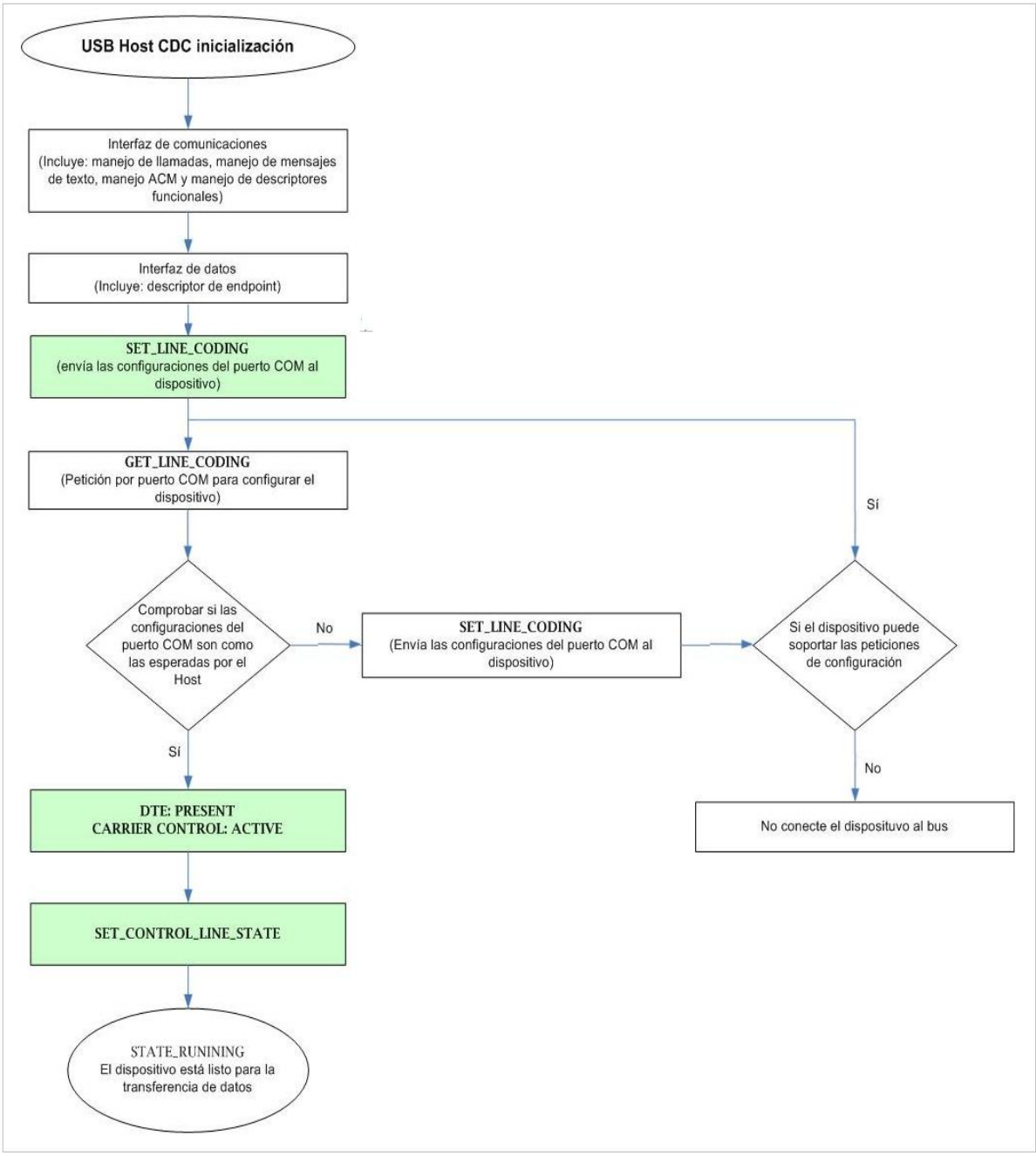


Figura 26 Esquema de configuración bajo el modelo ACM-Serial Emulation.

5.4 Tercera Etapa: Administración del dispositivo bajo una rutina de control del estado del Modem EDGE-USB

Luego de configurar el modem bajo el modelo de control reestructurado ACM-CDC-USB, el modem se encuentra listo para ser utilizado como dispositivo de comunicaciones inalámbricas.

En este punto varios procesos deben llevados a cabo regularmente para mantener bajo control los posibles cambios de estado del dispositivo, o bien atender cualquier evento relacionado solicitudes e interrupciones en el bus USB. El conjunto de estos procesos se resumió en la función USBTasks () y debió ser utilizada con regularidad para verificar el estado del modem USB con cierta periodicidad. En la figura 25 se presenta la estructura de administración de eventos que pueden ocurrir en cualquier momento y los cuales debieron ser atendidos con el orden establecido por la pila de aplicaciones embebidas USB, creado por la empresa Microchip.

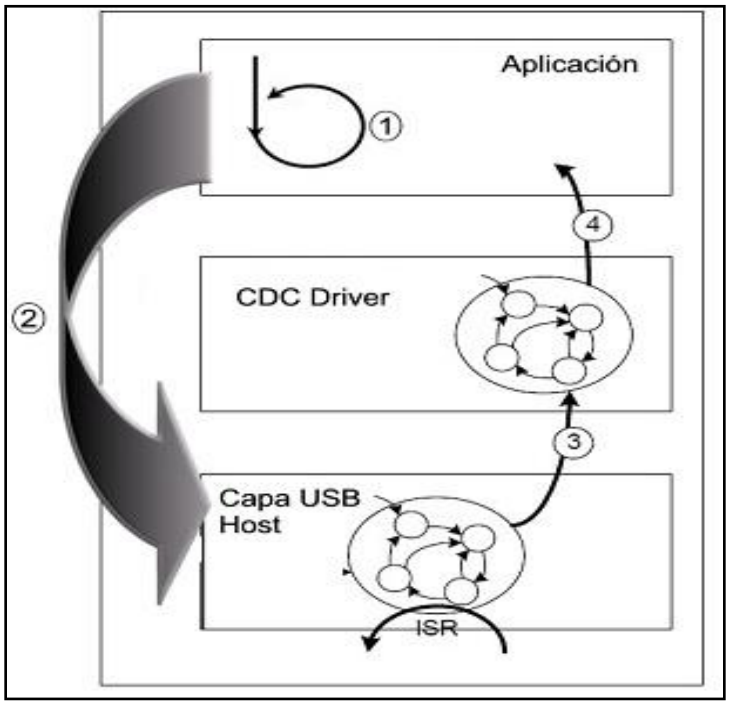


Figura 27 Capas de control para manejar los eventos del dispositivo.

Cabe destacar que cada uno de estos eventos fue etiquetado con instrucciones de control para determinar la causa de cada evento y poder dar rápida atención de ser el caso.

```
// CDC Eventos Específicos

case EVENT_CDC_NONE:
case EVENT_CDC_ATTACH:return TRUE;break;
case EVENT_CDC_COMM_READ_DONE:    SendMessage( 2, "COMM_READ_DONE\n" );return TRUE; break;
case EVENT_CDC_COMM_WRITE_DONE:   SendMessage( 2, "COMM_WRITE_DONE\n" );return TRUE; break;
case EVENT_CDC_DATA_READ_DONE:    SendMessage( 2, "DATA_READ_DONE\n" );return TRUE; break;
case EVENT_CDC_DATA_WRITE_DONE:   SendMessage( 2, "DATA_WRITE_DONE\n" );return TRUE; break;
```

Figura 28 Posibles eventos que deben ser atendidos por el controlador host USB

5.5 Desarrollo de una aplicación que permita enviar y recibir mensajes de texto desde el controlador PIC32MX460512 a través de un Modem USB

Para desarrollar las rutinas de envío y recepción de mensajes de texto, se tomó en cuento los análisis hechos mediante las pruebas previas hechas en el programa de comunicaciones Hyperterminal de Windows. Mediante esas pruebas se determino la se secuencia de comandos a ejecutar para configurar un modem USB en estado de envío y recepción de mensajes de texto.

5.5.1 Diseño de la temporización de los comandos V2.5ter (AT Hayes) enviados desde el controlador PIC32MX460512

Como parte de la investigación realizada en el tema de comandos AT, se identificó que cualquier dispositivo de comunicaciones que funcione bajo el protocolo V2.5ter (AT Hayes) tiene distintos tiempos de respuesta para cada comando solicitado. Esto implica que para enviar con éxito los comandos AT desde el controlador PIC32MX460512, este debe tener tiempos de espera específicos para cada tipo de comando que se desee enviar.

La temporización de los comandos se realizó en base a distintas pruebas de control, con el objetivo de identificar el tiempo de espera adecuado según el caso.

En el diagrama de la figura 27 resume el formato de envío de comandos desde el controlador PIC32MX460512, y el mismo es la base para todos los comandos AT se enviaron desde el controlador.

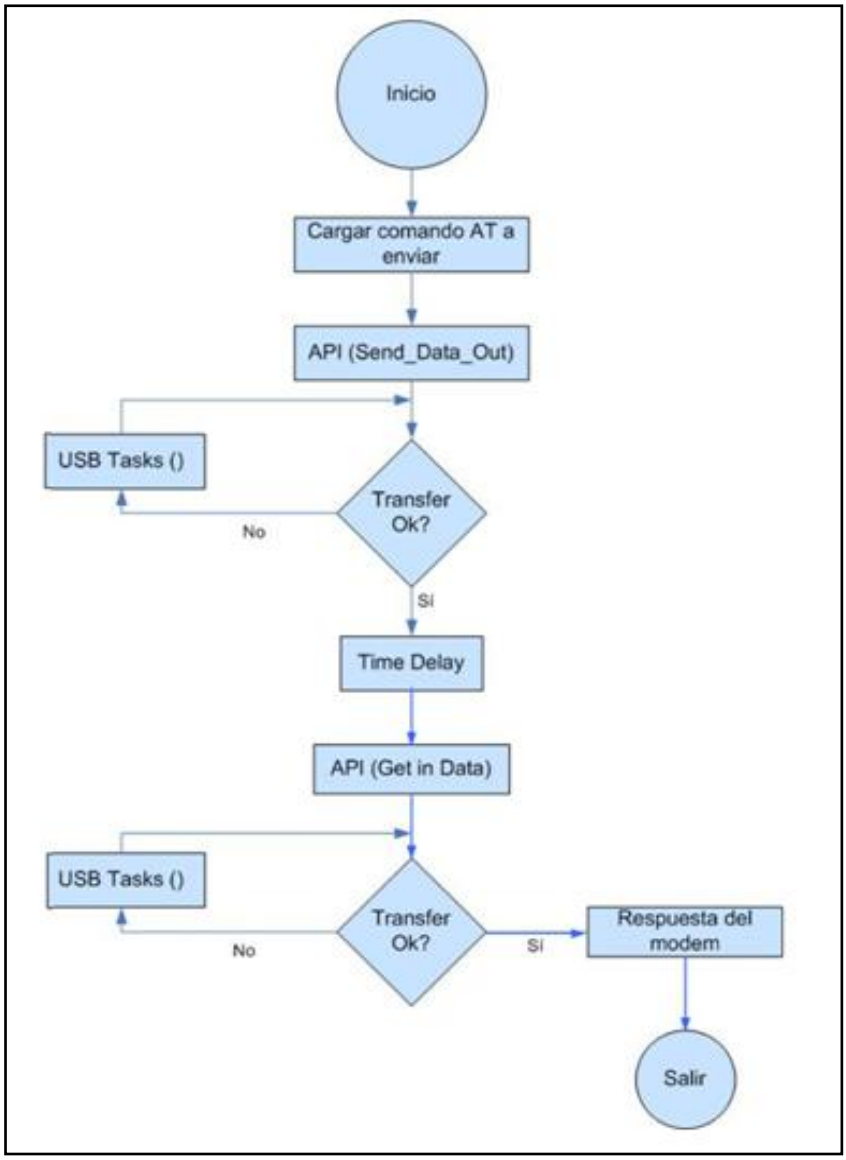


Figura 29 Esquema de diseño de envío de comandos AT desde el controlador host USB.

5.5.2 Diseño de las rutinas de comandos los AT en base al esquema de temporización de la figura 27

1. AT+ATE0

Este es el primer comando que se envió al modem ya que elimina los ecos de respuesta enviados desde el modem al controlador PIC32MX460512.

La función CDC_Api_Send_Data_Out ejecuta una transferencia de tipo Bulk y envía el valor contenido en el string ATEO.

```
  for( i=0; i<(sizeof (ATE0)+1);i++)
                        { USB_CDC_OUT_Data_Array[i] = ATE0[i];
                        }
USBHostCDC_Api_Send_OUT_Data(sizeof (ATE0) / sizeof (ATE0[0]),USB_CDC_OUT_Data_Array);
while (!USBHostCDC_ApiTransferIsComplete(&ErrorDriver,&NumOfBytesRcvd))
 {USBTasks();};
 USBTasks();
Delay10us(100000);
USBHostCDC_Api_Get_IN_Data(MAX_NO_OF_IN_BYTES,USB_CDC_IN_Data_Array);
while (!USBHostCDC_ApiTransferIsComplete(&ErrorDriver,&NumOfBytesRcvd))
 {USBTasks();};
```

Figura 30 Código diseñado para enviar el comanda ATE0

2. AT+CSCS

Este comando establece el tipo de caracteres que acepta el modem EDGE-USB que en este caso se seleccionó GSM-Internacional.

```
 for( i=0; i<(sizeof (ATCSCS)+1);i++)
                        { USB_CDC_OUT_Data_Array[i] = ATCSCS[i];
                        }
USBHostCDC_Api_Send_OUT_Data(sizeof (ATCSCS) / sizeof (ATCSCS[0]),USB_CDC_OUT_Data_Array);
while (!USBHostCDC_ApiTransferIsComplete(&ErrorDriver,&NumOfBytesRcvd))
 {USBTasks();};
 USBTasks();
 Delay10us(100000);
USBHostCDC_Api_Get_IN_Data(MAX_NO_OF_IN_BYTES,USB_CDC_IN_Data_Array);
while (!USBHostCDC_ApiTransferIsComplete(&ErrorDriver,&NumOfBytesRcvd))
 {USBTasks();};
```

Figura 31 Código diseñado para enviar el comanda AT+CSCS

3. AT+CPIN

Con este comando se carga el número de código de la tarjeta SIM insertada en el modem USB. Cabe destacar que este comando ejecuta las peticiones de conexión con la red de telefonía celular del ICE; por lo tanto, este comando requirió de mayores tiempos de espera para cada algunas de las respuestas del modem.

```
USBHostCDC_Api_Send_OUT_Data(sizeof (ATCPIN) / sizeof (ATCPIN[0]),USB_CDC_OUT_Data_Array);
while (!USBHostCDC_ApiTransferIsComplete(&ErrorDriver,&NumOfBytesRcvd))
 {USBTasks();};
 Delay10us(100000);
mLED_2_Off();
USBHostCDC_Api_Get_IN_Data(MAX_NO_OF_IN_BYTES,USB_CDC_IN_Data_Array);
while (!USBHostCDC_ApiTransferIsComplete(&ErrorDriver,&NumOfBytesRcvd))
 {USBTasks();};
Delay10us(100000);
   mLED_2_On();
   //UART2PrintString( "SETUP MENU\r\n" );
USBHostCDC_Api_Get_IN_Data(MAX_NO_OF_IN_BYTES,USB_CDC_IN_Data_Array);
while (!USBHostCDC_ApiTransferIsComplete(&ErrorDriver,&NumOfBytesRcvd))
 {USBTasks();};
    Delay10us(200000);
   mLED_2_Off();
USBHostCDC_Api_Get_IN_Data(MAX_NO_OF_IN_BYTES,USB_CDC_IN_Data_Array);
while (!USBHostCDC_ApiTransferIsComplete(&ErrorDriver,&NumOfBytesRcvd))
 {USBTasks();};
Delay10us(200000);
```

Figura 32 Código diseñado para enviar el comanda AT+CPIN

De igual manera se trabajaron los comandos AT+CSCA, AT+CMGF, AT+CSGS ya que los tiempos de entrada y salida de datos son básicamente los mismos.

5.5.3 Rutina de configuración para utilizar el servicio de mensajería de texto

La configuración del modem para habilitar el servicio de mensajes de texto requirió de lograr una conexión efectiva con las redes de telefonía celular del Instituto Costarricense de Electricidad. Los primeros dos comandos son los más importantes de la etapa de configuración, ya que se eliminan los ecos de respuesta del modem y se levanta la conexión con "*ICE Servicios*" por medio del comando AT+CPIN.

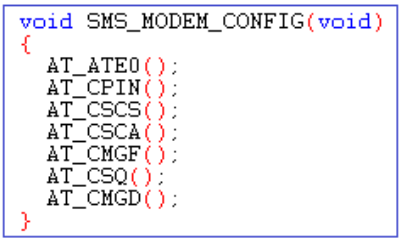


Figura 33 Secuencia de comandos ejecutados para configurar el modem en servicio de mensajería de texto

El comando AT+CSCA se encarga de almacenar el numero de centro de mensajes del ICE para líneas de tipo 3G y por último el comando AT+CMGF selecciona el servicio de mensajería del modem EDGE-USB.

5.5.4 Rutina para el envío de mensajes de texto utilizando el modem EDGE-USB

Después de configurar el modem en servicio de mensajería de texto, se procedió a implementar la rutina de programación que permitió enviar mensajes de texto desde el controlador PIC32MX460512 utilizando el modem EDGE-USB. La secuencia de los pasos implementados se encuentra descrita en el diagrama de flujo presentado en la figura 32, y se realizó en base al estudio realizado en la fase de investigación de este proyecto.

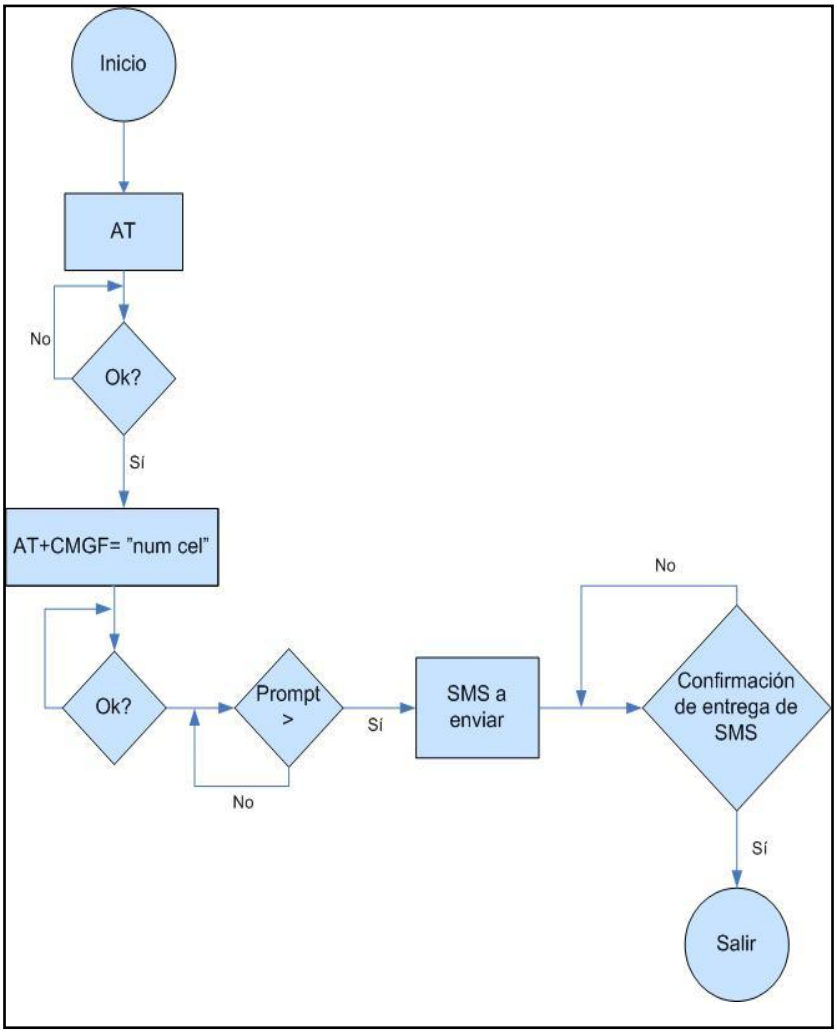


Figura 34 Diagrama de flujo para enviar un mensaje de texto

5.5.5 Rutina para la recepción de mensajes de texto con el modem EDGE-USB

Esta rutina controla la recepción de mensajes de texto por medio de una etapa de revisión del estado de llegada de nuevos mensajes de texto al controlador.

Por medio de la función API (Get Data In) se detecta cualquier nuevo informe del modem. En el momento que un nuevo informe es presentado al controlador, el siguiente procedimiento es el de almacenar el contenido del mensaje para luego borrar el registro de entrada con el objetivo de liberar el un espacio en memoria para asegurar que se podrá atender cualquier nuevo mensaje que ingrese al sistema.

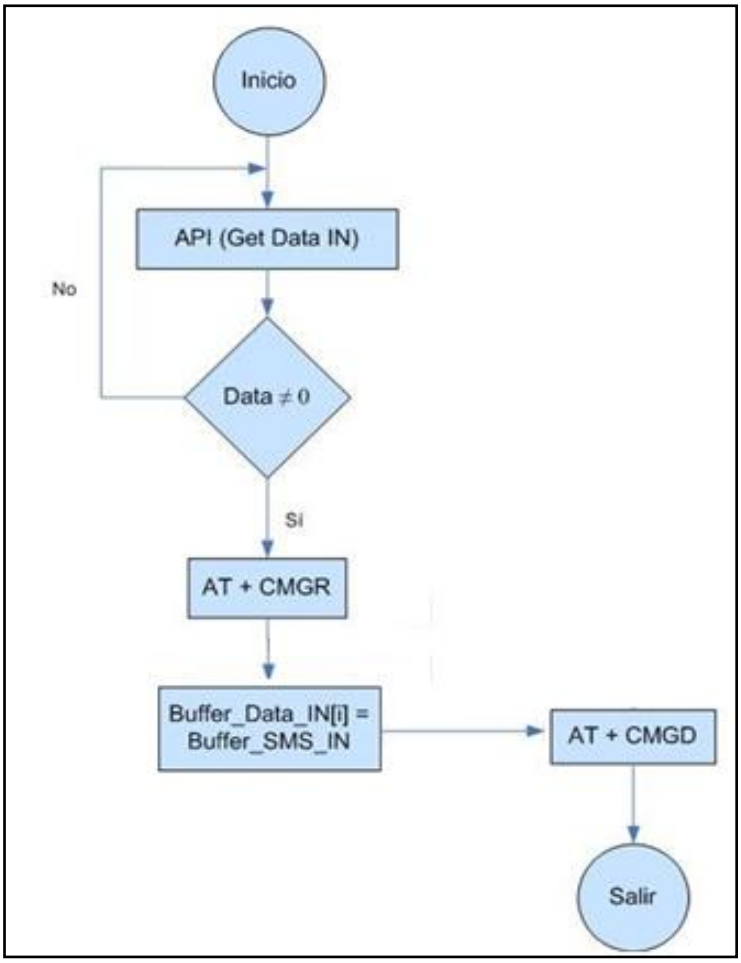


Figura 35 Diagrama de flujo para recibir un mensaje de texto

5.6 SIWA- RTOS y Servicio de Mensajes de Texto

La última parte de la solución propuesta se basó en migrar la aplicación de creada para el servicio de mensajería de texto a el sistema operativo de tiempo real SIWA-RTOS.

Esta migración se realizó por medio de la siguiente secuencia de procedimientos:

a. Se movieron todos los Header Files contenidos en la aplicación CDC-USB-HOST a la carpeta de los archivos a incluir de SIWA-RTOS.
b. Se creó con MPLAB el workspace del SIWA-RTOS.
c. Se cargó los archivos USB_host.c, USB_config.c, USB_CDC_Host.c el código de aplicaciones del SIWA-RTOS.
d. Se crearon las nuevas direcciones de búsqueda de los archivos a incluir.

La aplicación creada para comprobar los procedimientos mencionados se creó en base al diagrama de flujo presentado en la figura 34.

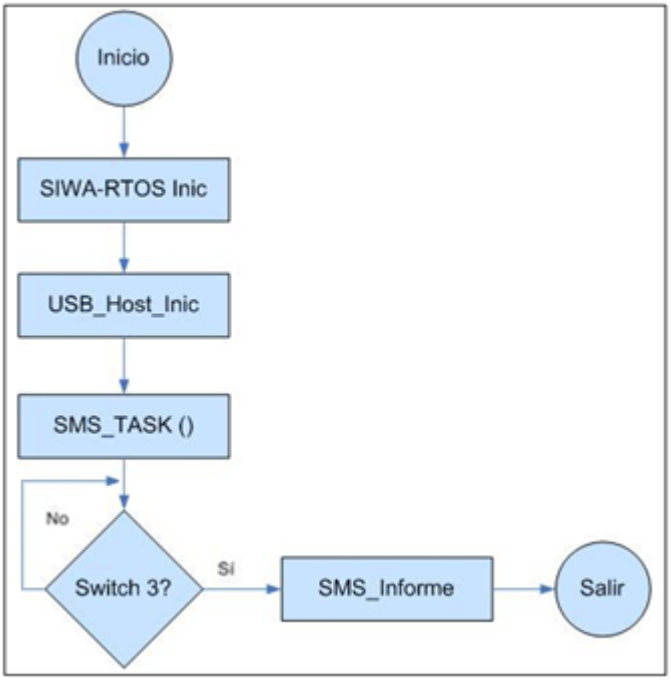


Figura 36 Esquema del proceso para ejecutar la tarea de enviar un mensaje de texto utilizando el sistema operativo SIWA-RTOS

A nivel de programación, se creó una primera función de configuración del modem EDGE-USB (Modem_USB_INIT()). Esta función se incluyó en el código principal de programa ya que debe ser ejecutado en la etapa de configuración del sistema. Luego se incluyó una nueva tarea denominada SMS_GSM en el planificador del sistema operativo

```
int main( void )
{
    Modem_USB_INIT();
    /* Configure both the hardware and the debug interface. */
    vSetupEnvironment();
    xTaskCreate( SMS_GSM, "SMS_GSM", 240, NULL, 1, NULL );//Tarea para enviar SMS
    /* Start the scheduler so our tasks start executing. */
    vTaskStartScheduler();
    for( ;; );
    return 0;
}
```

Figura 37 Código principal de la ejecución de la tarea SMS_GSM en SIWA-RTOS

Por medio de la tarea SMS_GSM, se creó una aplicación que envía un mensaje de texto con una notificación sobre el estado actual del sistema operativo.

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

void SMS_GSM( void *pvParameters )
{
const char *pcTaskName = "SMS_GSM\n";
volatile unsigned long ul;
    for( ;; )
    {   UART2Init();
        vPrintString( pcTaskName );
        if (!mGetSW3){
                      SMS_TASK();
                      }
        for( ul = 0; ul < mainDELAY_LOOP_COUNT; ul++ )
        {

        }
    }
}
```

Figura 38 Tarea SMS_GSM

6 Capítulo 6: Análisis y Resultados

En este capítulo se describen los análisis y resultados de las etapas desarrolladas en la solución propuesta del proyecto. Se presentarán figuras y gráficos que muestran el comportamiento del sistema en distintas circunstancias con el objetivo de calificar el funcionamiento del sistema en cada una de sus características de diseño.

Se expondrán cada una las secuencias de depuración de datos por medio de la aplicación Output-Window la cual forma parte del software MPLAB de la empresa Microchip. Esto debido a que fue la herramienta ideal para estudiar el comportamiento de los procesos involucrados con el controlador PIC32MX460512.

Además, se presentarán los resultados obtenidos en el envío de mensajes de texto desde el controlador PIC32MX460512 por medio de una estación de recepción de mensajes de texto llamada NOKIA PC SUITE, la cual funcionó como interfaz de despliegue visual para la información transmitida a través de la red de telefonía celular del Instituto Costarricense de Electricidad.

Por último, se presentará un estudio de consumo de potencia del controlador se realizado por medio del analizador de fuentes de potencia de la marca KEITHLTY que se encuentra a disposición del Laboratorio de Electrónica Aplicada a la Protección del Medio Ambiente de la escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica.

6.1 Resultados

6.1.1 Verificación de la enumeración del dispositivo (etapa de reconocimiento)

En la figura 37 se muestra el procedimiento ejecutado por el controlador PIC32MX460512 para enumerar con éxito el dispositivo de comunicaciones EDGE-USB. Este resultado corresponde a la primera etapa de reconocimiento de dispositivos USB y fue obtenido por medio del la ventana de salida de datos de MPLAB.

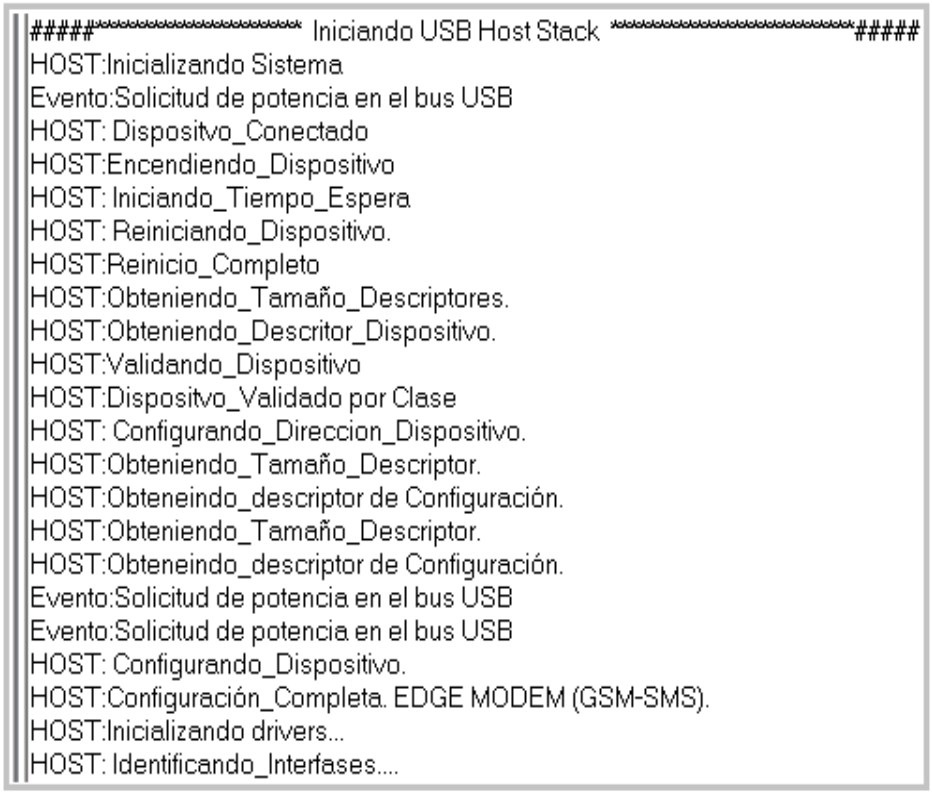


Figura 39 Proceso de enumeración del dispositivo USB

6.1.2 Verificación de la enumeración del dispositivo (etapa de configuración)

La verificación del funcionamiento de etapa se dio por medio de la inserción de etiquetas de control presentadas el figura 38, y en ella se presenta el valor del uno de los descriptores de el dispositivo que se encarga de informar al controlador PIC32MX46512 sobre la clase a la cual pertenece el dispositivo y el código del protocolo con el cual se puede administrar.

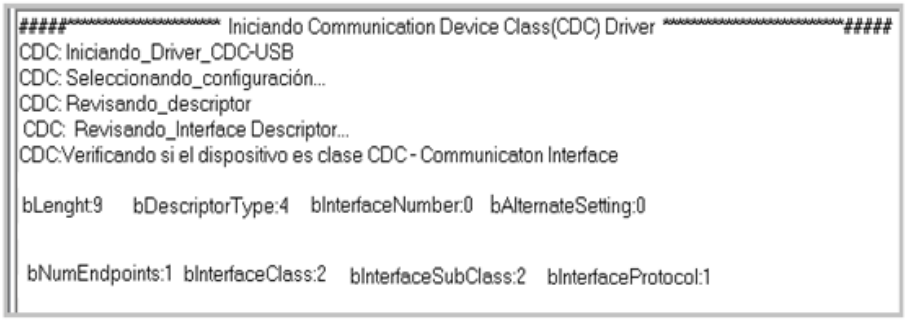


Figura 40 Proceso de reconocimiento del descriptor de Comunicaciones

6.1.4 Resultados de la configuración del modem EDGE-USB bajo el modelo ACM-Serial Emulation

En la figura se muestran los procedimientos del modelo ACM-Serial Emulation. Luego de la ejecución de este modelo, el modem EDGE-USB está listo para ejecutar los comandos AT.

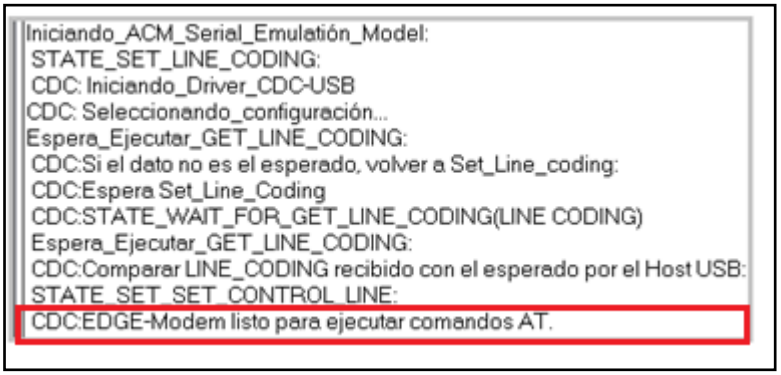


Figura 41 Ejecución del modelo ACM-Serial Emulation

6.1.3 Resultados los estudios realizados con los analizadores de protocolos USBLyser y USBTrace.

a) En la primera parte del estudio del modelo ACM-Serial Emulation, se obtuvieron los resultados del formato de los paquetes enviados por medio de Hipeterminal de Windows al modem EDGE-USB.

En la figura 40 se pueden apreciar el tipo de transferencia realizada, el tipo de comando y la etapa de datos; los cuales son parte del protocolo USB.

Request	Request Details	Raw Data
Class Interface	Set Line Coding (Ifc:0)	00 08 07 00 00 00 08
Class Interface	Set Line Coding (Ifc:0)	00 08 07 00 00 00 08
Control Transfer	Set Line Coding (Ifc:0)	
Control Transfer	Set Line Coding (Ifc:0)	
Class Interface	Get Line Coding (Ifc:0)	
Class Interface	Get Line Coding (Ifc:0)	
Control Transfer	Get Line Coding (Ifc:0)	00 08 07 00 00 00 08
Control Transfer	Get Line Coding (Ifc:0)	00 08 07 00 00 00 08
Class Interface	Set Control Line State (Ifc:0)	
Class Interface	Set Control Line State (Ifc:0)	
Control Transfer	Set Control Line State (Ifc:0)	
Control Transfer	Set Control Line State (Ifc:0)	
Class Interface	Set Line Coding (Ifc:0)	00 08 07 00 00 00 08
Class Interface	Set Line Coding (Ifc:0)	00 08 07 00 00 00 08

Figura 42 Secuencia de configuración obtenida con el programa USBLyser.

b) El siguiente paso se basó en obtener los resultados de los detalles de cada comando; en la figura 41 se presenta los comandos Set Line Coding, Get Line Coding y Set Control Line State

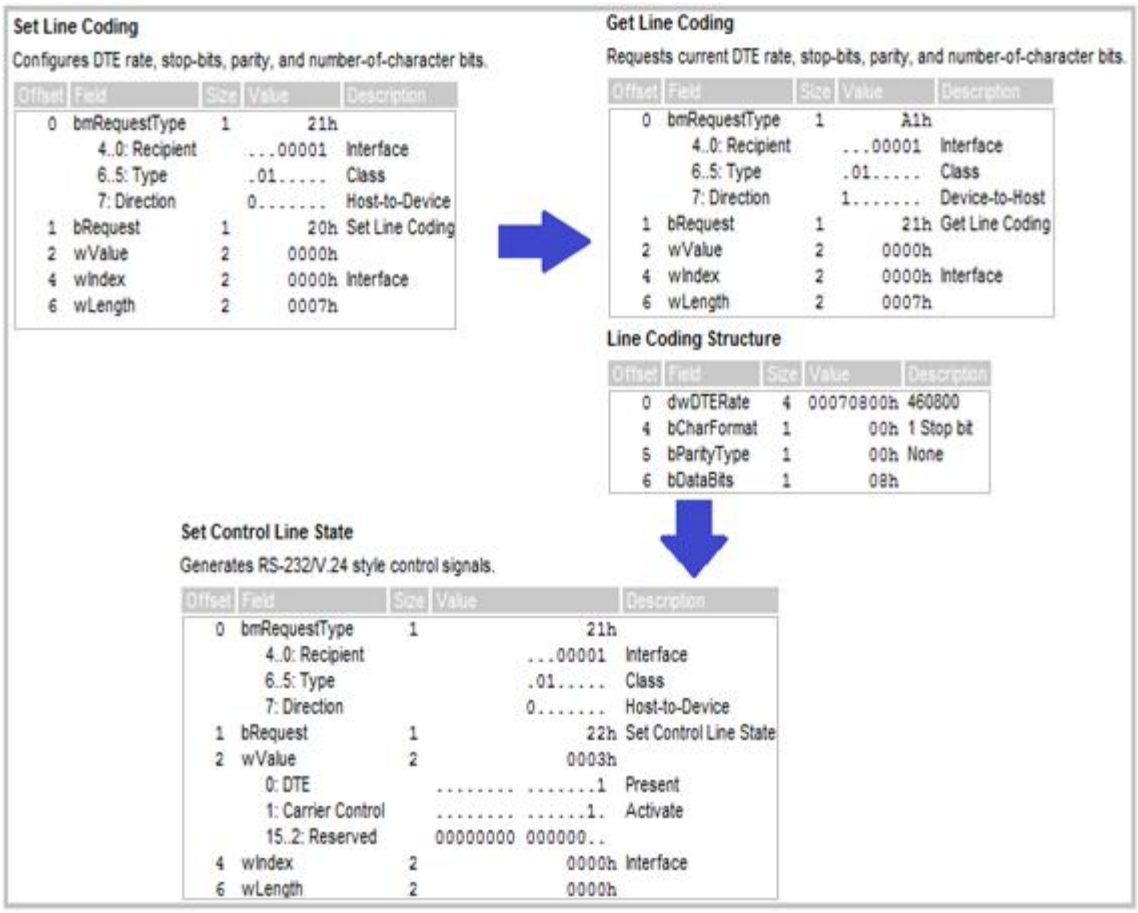


Figura 43 Comandos del modelo ACM-Serial Emulation obtenidos con USBLyser

6.1.5 Resultados de la implementación del diseño de la temporización de los comandos V2.5ter (AT Hayes) enviados desde el controlador PIC32MX460512

El resultado del diseño de la rutina de temporización de los comandos ejecutados, se demostró con las respuestas enviadas desde el modem al controlador PIC32MX460512. Un ejemplo de estas respuestas se presenta en la figura 42, la cual presenta una respuesta del proceso de conexión con las radio bases del Instituto Costarricense de Electricidad.

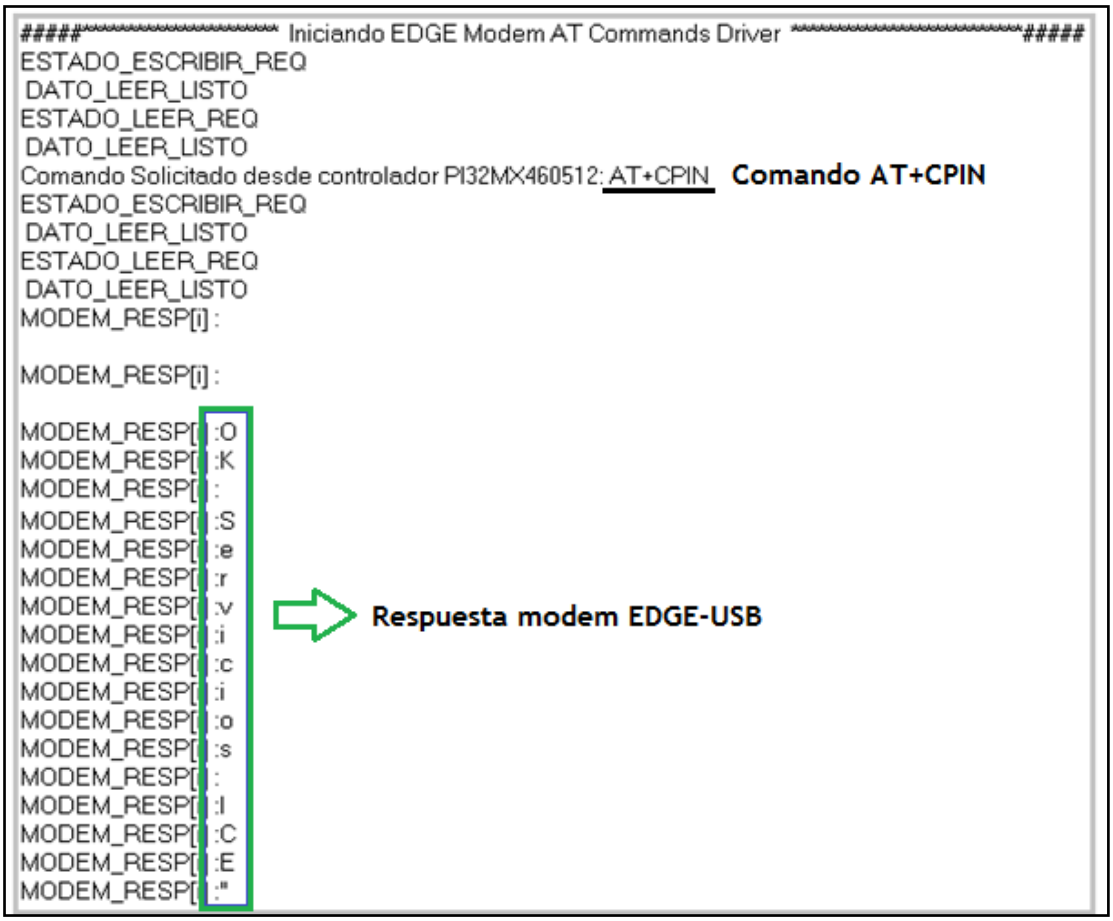


Figura 44 Verificación de la efectividad de conexión del modem con la infraestructura del ICE

6.1.6 Resultados de la implementación de las rutinas de recepción de datos desde el SIWA-RTOS a través de un Modem USB

- ✓ Confirmación de la recepción de un mensaje de texto enviado desde la estación NOKIA-PC-SUITE al controlador PIC32MX460512. En este resultado se puede apreciar la confirmación de recepción del modem EDGE-USB por medio de la cadena de datos **CMTI**.

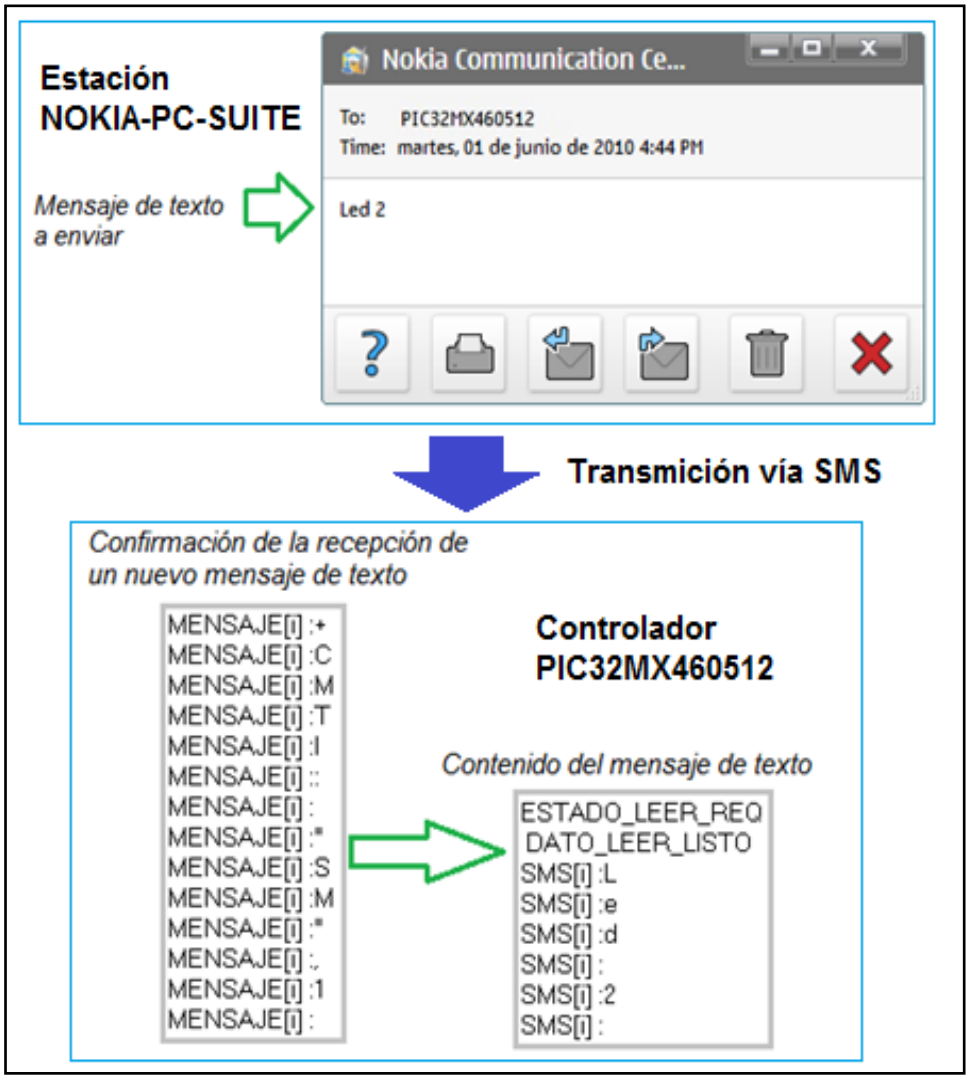


Figura 45 Estructura de recepción de un mensaje de texto en SIWA-RTOS

6.1.7 Resultados de la implementación de las rutinas de envío de datos desde SIWA-RTOS a través de un Modem USB

- ✓ Confirmación de la trasmisión de un mensaje de texto enviado desde el controlador PIC32MX460512 a la estación NOKIA-PC-SUITE. En este resultado se puede apreciar la confirmación de la transmisión del modem EDGE-USB por medio de la cadena de datos **CMGS**.

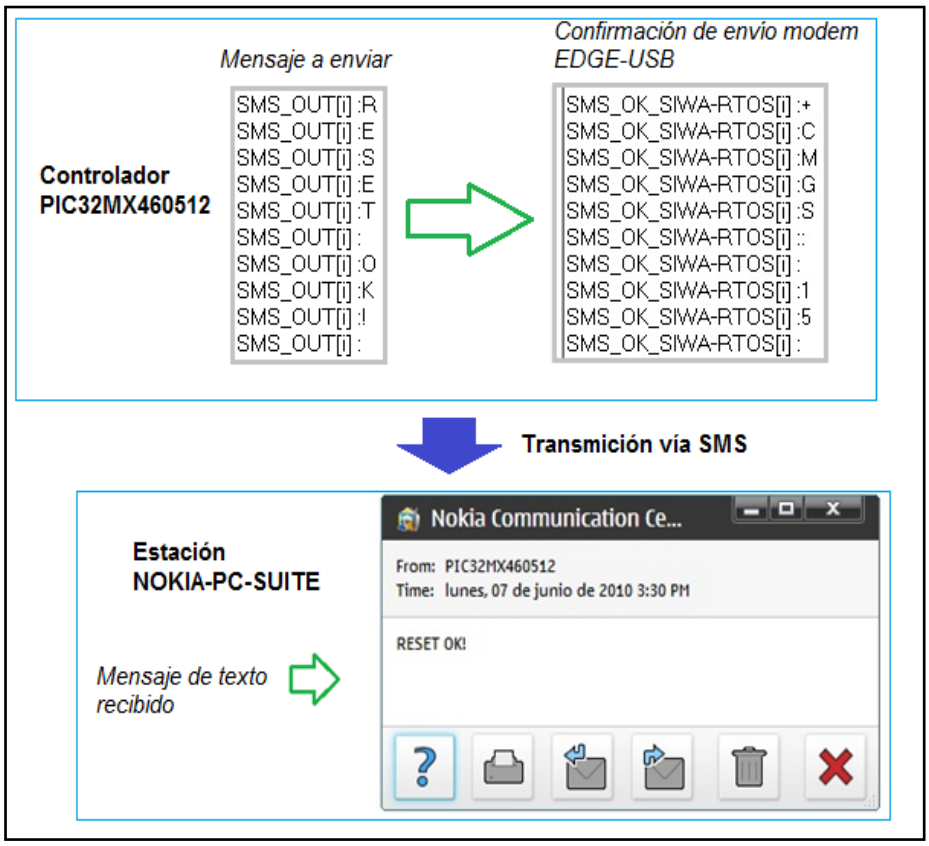


Figura 46 Estructura envío de un mensaje de texto en SIWA-RTOS

6.1.8 Verificación de la tasa de transferencias exitosas entre el dispositivo USB y el Host USB

Para determinar el porcentaje de transferencias exitosas, se cuantificaron la cantidad de peticiones USB-Host > USB Device que generaron algún tipo de error. Cabe

destacar que la única petición que creó un error en el bus, es la petición en la cual se le pide una respuesta al modem y el modem no tiene ninguna respuesta para el Host.

Figura 47 Validación de el acople de transmisión de datos desde el modem EDGE-USB al controlador PIC32MX460512

6.1.9 Verificación de la efectividad de transferencias de datos por medio de la red de telefonía celular por medio del servicio de mensajería de texto.

En la tabla 6 se presenta una prueba de envío de mensajes de texto desde el controlador PIC32MX460512. En ella se encuentra la trama de envío y la trama de datos recibida y una confirmación de la efectividad de los datos transmitidos.

Tabla 6 Comparación de tramas de datos enviadas desde el controlador hasta la estación de verificación de datos.

Trama de Datos Enviados	Trama de Datos Recibidos	Resultado
SMS_OK!	SMS_OK!	OK
RESET OK!	RESET OK!	OK
LED OK!	LED OK!	OK

6.1.10 Tasa de transferencia efectiva de datos enviados desde el nodo sumidero hasta el nodo receptor de la red o Load Point.

Se realizaron pruebas de control para identificar la tasa de transferencia efectiva de datos enviados. Los resultados se muestran en la tabla 7 y cada una de las pruebas se realizó con la transferencia máxima de 160 bytes por mensaje enviado.

Tabla 7 Tasa de transferencia efectiva de datos enviados desde el nodo sumidero

Tiempo de espera entre envío de mensajes (s)	Cantidad de mensajes enviados	Cantidad de mensajes recibidos	Porcentaje de error %	Tasa de transmisión (bps)
3	5	2	60	424
5	5	3	40	256
8	5	5	0	160

6.1.11 Cálculo de tiempo de autonomía del sistema utilizando una fuente de energía portable, puntualmente una batería de 9V.

En esta prueba se realizó utilizando una secuencia de envío de mensajes de texto con un lapso de 1 minuto entre cada mensaje enviado. Los Resultados se muestran en la tabla 8.

Tabla 8 Duración de descarga de la batería durante el envío de mensajes de texto

Inicio de Prueba	Fin de Prueba	Tiempo efectivo (h)	Cantidad de Mensajes Enviados
11:39 pm	12:39 am	1	62

- La corriente promedio de consumo del sistema se estimó en 200mA para la rutina de envío de mensajes de texto.

6.1.12 Gráficas de consumo de corriente para la aplicación de envío y recepción de mensajes de texto

En la figura 46 se presenta la gráfica del estudio del consumo de corriente del controlador en un período de tiempo de muestreo de 73 segundos aproximadamente.

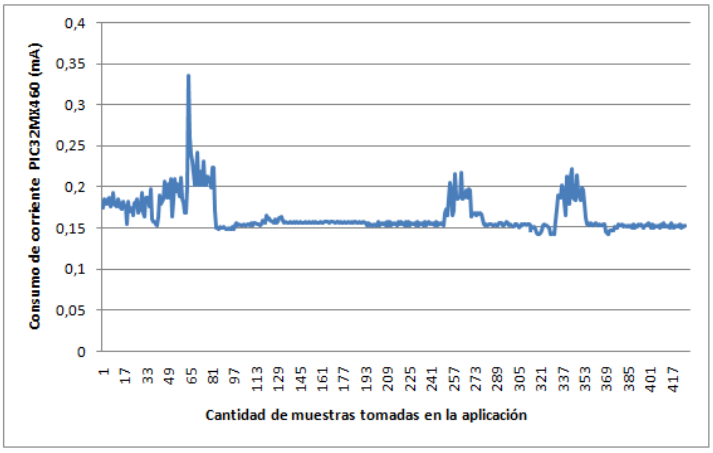


Figura 48 Consumo de corriente del controlador durante la ejecución de la aplicación de mensajes de texto

En la gráfica 47 se presenta un filtro de los datos de la figura 46 cuyo objetivo es identificar cada una de las etapas de la aplicación de conexión, configuración, envío y recepción de los datos a través del modem EDGE-USB

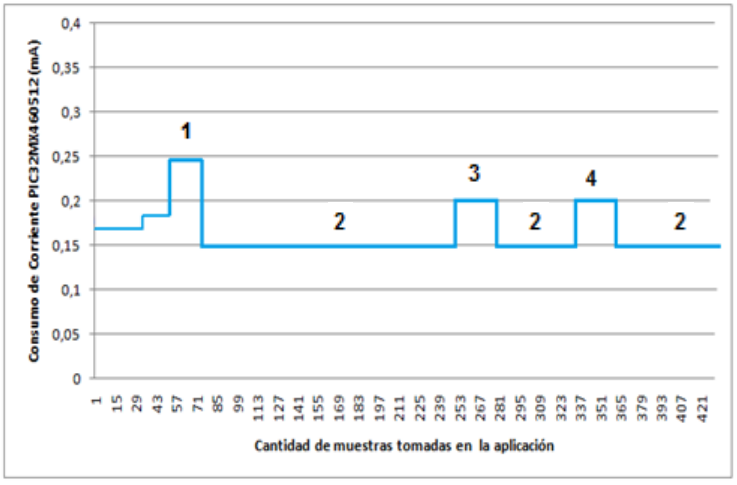


Figura 49 Descripción de las etapas de 1. Conexión, 2. USBTasks, 4. Nuevo SMS, 4. Salida SMS

Debido a que la señal de la figura 48 se presenta de una forma periódica, el análisis matemático para obtener los valores del consumo de potencia se basó en una integral definida por los límites establecidos en los rangos de las muestras para cada una de las ejecuciones de la aplicación. La ecuación 1 presenta el método para calcular la potencia promedio que consume el controlador

$$Potencia\ Promedio = 1/T \int p(t)\, dt \qquad (1)$$

$$Potencia\ Promedio\ Conexión\ ICE = \frac{3.3\ V}{73\ s} \int_{0}^{4} 0.168A\ dt = 30mW$$

$$Potencia\ Promedio\ Conexión\ ICE = \frac{3.3\ V}{73\ s} \int_{4}^{7} 0.176A\ dt = 23mW$$

$$Potencia\ Promedio\ Conexión\ ICE = \frac{3.3\ V}{73\ s} \int_{7}^{12} 0.25A\ dt = 56mW$$

$$Potencia\ Promedio\ USB\ Task = \frac{3.3\ V}{73\ s} \int_{0}^{48} 0.15A\ dt = 325mW$$

$$Potencia\ Promedio\ Nuevo\ SMS = \frac{3.3\ V}{73\ s} \int_{57}^{62} 0.2A\ dt = 45mW$$

$$Potencia\ Promedio\ Salida\ SMS = \frac{3.3\ V}{73\ s} \int_{43}^{48} 0.2A\ dt = 45mW$$

Tabla 9 Consumo de potencia para cada uno de los procesos de la aplicación de servicio de mensajes de texto

Ejecución	Rango de Muestras (s)	Consumo de corriente Max (mA)	Consumo de Potencia Promedio (mW)
1. Conexión ICE	0.00-11.7	250	109
2. USB Tasks	11.7-43.5	150	325
3. Nuevo SMS	43.48.15	200	45
4. Salida SMS	57.5-62.2	200	45

6.2 Análisis de Resultados

En esta sección del documento se presentarán los análisis de los resultados segmentados en base a los resultados obtenidos durante el desarrollo de la solución a este problema.

6.2.1 Reconocimiento de un dispositivo USB utilizando la pila de aplicaciones embebidas USB de la empresa Microchip

Uno de los primeros pasos para el desarrollo de la solución fue el de crear una secuencia lógica para la enumeración del dispositivo. Con primer paso de reconocimiento se procedió a identificar el momento cuando el controlador PIC32MX460512 reconoció a nivel de hardware, que un nuevo dispositivo había sido conectado con éxito en el modulo USB-OTG.

Luego se procedió a verificar la primera solicitud de potencia realizada por el dispositivo. En esta petición se le asigna un valor inicial de 100 mA al dispositivo ya que es un requisito del protocolo de enumeración establecido por el estándar USB 2.0. Al validar esta petición, se confirmó visualmente el resultado por medio de un indicador visual de tipo lumínico (Led) incorporado en el modem EDGE-USB. La validación de esta petición se comprobó por medio de los resultados descritos en la figura 37.

En este punto se analizó el proceso de reinicio del modem por medio el sub-estado RESET_DEVICE, este reinicio cumplió la función de borrar las configuraciones actuales en los módulos de tipo END-POINT, y de esta manera se comprobó que se podía iniciar un nuevo proceso de enumeración del dispositivo desde un estado conocido.

El siguiente paso fue el de verificar el contenido de cada uno de los descriptores solicitados por el Host USB. Los resultados de las peticiones de los primeros

descriptores fueron exitosos ya que el valor obtenido en el Host-USB PIC32MX460512 fueron los esperados en base al estudio previo hecho por medio de las herramientas e análisis de protocolos USBLyser y USBTrace.

Continuando con la validación del modem USB por parte del controlador PIC32MX460512, se encontró el primer problema específico de identificación del dispositivo EDGE-USB; la validación por medio de etiquetas del fabricante VID o PID, o bien la validación por clase de dispositivo (Comunication Device Class).

El primer intento de validar el dispositivo por medio de las etiquetas VID-PID presentó la ventaja de ser efectivo a la hora del reconocimiento por parte del Host USB. Sin embargo más adelante este tipo de identificación presentó el inconveniente de no informar al controlador USB-Host sobre las capacidades y características para las cuales fue diseñado el dispositivo USB.

Por lo tanto, se utilizó el método de validación por clase de dispositivo, ya que el protocolo USB 2.0 establece códigos en hexadecimal que permiten identificar mejor las capacidades de cada dispositivo USB según sea su aplicación.

Debido a que este método es de tipo genérico, se realizó una depuración de las rutinas de reconocimiento para mejorar el tiempo de enumeración del dispositivo en el host USB. Para verificar el estado de la validación, se insertó la etiqueta Dispositivo Validado por Clase, la cual se presenta en la figura 37.

Durante la última petición de potencia realizada por dispositivo USB, el dispositivo solicitó un aumento en el nivel máximo de corriente para pasar de 100mA a 500mA con el objetivo de garantizarse contar con la corriente suficiente para realizar las tareas para las cuales está diseñado. Por efectos de mantener bajo en consumo de potencia de la plataforma CRTECMote, se intentó reducir este máximo pero con este el cambio, el dispositivo rechaza las peticiones de enumeración por lo tanto se decidió establecer los 500mA como máximo. Más adelante se presentaran los

resultados que comprueban que el consumo de corriente del dispositivo no supera los 350 mA durante toda la ejecución de la aplicación.

6.2.2 Reestructuración del modelo ACM al modelo ACM-Serial Emulation

Como se explica en el capítulo 5, la reestructuración del modelo se creó en base al estudio de la aplicación de este modelo por parte de la aplicación de comunicaciones Hiperterminal de Windows. Es importante destacar que el principal problema durante la reestructuración del modelo surgió al tratar de cumplir con uno de los requisitos de configuración llamado Set Control Line State.

A diferencia de los comandos Set Line Coding y Get line Coding que tienen la capacidad de transmitir la información por medio de la etapa de datos establecida por el protocolo USB 2.0, el Set Control Line State comando debe transmitir la trama de datos de control por medio de la etapa de Setup del protocolo USB 2.0.

Este problema se soluciono reescribiendo el encabezado de datos de la función USBHost-Issue-DeviceRequest específicamente en la configuración del encabezado W-Value por el valor 0x0003 o 0x0001 hexadecimal.

De esta manera se le informó al modem EDGE-USB sobre la presencia de un terminal capaz de controlar las respuestas que el dispositivo genera ante cada petición.

Concluida esta etapa de configuración, se procedió a verificar si efectivamente el modem EDGE-USB estaba listo para ejecutar comandos del protocolo V2.5ter AT Commands. Esta prueba se realizó por medio de la función USB_CDC_OUT, en la cual se escribió el comando AT y se envió al modem por medio de una solicitud de transferencia de tipo bulk. La prueba se determinó como exitosa ya que en pocos instantes después de hacer enviado el comando, el modem EDGE-USB efectivamente respondió bajo el formato de respuesta del protocolo V2.5ter AT Commands. La respuesta obtenida para este comando fue “OK”.

Por medio de los datos mostrados en la figura 39 se comprobó que el método de reestructuración del dispositivo en base al modelo ACM-Serial Emulation funcionó como se esperaba para la etapa de configuración.

6.2.3 Diseño de la temporización de los comando AT para conexión con la red de telefonía celular y envío de mensajes de texto.

El diseño de la temporización de estas rutinas se realizó en base a una serie de pruebas de ejecución de comandos desde el controlador PIC32MX460512.

El comando cuyo diseño de temporización dictó el comportamiento de la mayoría de los comandos implementados en la solución de este proyecto fue el comando AT+CPIN. Esto debido que una vez ejecutado, se da inicio a la conexión con las radio bases de comunicación del Instituto Costarricense de Electricidad. Ver figura 42.

Esto presentó un mayor tiempo de espera de respuesta del modem ya que el tiempo no es siempre el mismo. Por lo tanto el tiempo de espera de la respuesta se estableció relativamente alto para evitar errores de conexión con la infraestructura de telecomunicaciones.

En la aplicación también se incorporó el comando AT+CSQ, este comando se diseño de tal manera que pudiese brindar la información de la intensidad de la señal del modem en una escala de 0 a 30; donde 30 representa el máximo de intensidad posible que se puede traducir en un rango de 0 dbm a los -60 dbm. El apéndice A.2 contiene el gráfico que los caracteriza los posibles valores que se pueden tener por medio del comando AT+CSQ.

El comando AT+CMGR, es el segundo tipo de comando que se diseño bajo otro tipo de temporización ya que este comando ejecuta la acción de enviar un nuevo mensaje de texto desde el controlador. Esto debido a que en algunas ocasiones el mensaje de texto no puede ser enviado debido a circunstancias ajenas al modem y

por lo tanto la temporización y el diseño del envió del comando se realizaron contemplando un tiempo máximo de entrega de 2 segundos.

6.2.4 Diseño de las rutinas temporizadas para envío y recepción de mensajes de texto desde el controlador PIC32MX460512

Para verificar el resultado de el diseño realizado de las rutinas de envío y recepción de datos, se implementó una estación de recepción de mensajes de texto que funciono como interfaz de visualización de la información recibida desde el controlador o bien la información enviada al controlador. Esta estación se basó en la aplicación NOKIA-PC-SUITE y presentó la ventaja de dar una interfaz visual adecuada para analizar las tramas de datos.

La rutina de envío de mensajes de texto desde el controlador, se basó en almacenar la cadena de datos en un arreglo de caracteres en base al código ASCII para comunicaciones GSM, el cual es una variante del código ASCII internacional.

La plataforma de redes inalámbricas CRTECMote cuenta con un nodo receptor de datos que identifica una secuencia mediante separadores de tipo “@”, por lo tanto se creó una secuencia que sintetiza la el valor obtenido de los sensores, la identificación de cada uno de los nodos y la fecha y la hora en la cual se envió la información.

La rutina de recepción de mensajes de texto se diseño por medio de un lazo infinito de programación que se encuentra constantemente enviando una solicitud de transferencias de entrada al bus USB. La rutina fue verificada con el envió de un mensaje de texto con el comando Led 2, cuya función fue la de encender la cantidad de leds empaquetada en el mensaje de texto.

En las figuras 43 y 44 se comprueban los resultados esperados para las rutinas implementadas para enviar y recibir mensajes de texto desde el sistema operativo SIWA-RTOS mediante la transmisión de datos en una red de conexión de datos de área amplia.

6.2.5 Tasa de transferencia de datos efectiva y calculo de autonomía del sistema utilizando una batería

En la tabla 7 se presentan los resultados del estudio realizado para determinar la tasa de transferencia efectiva obtenido por el sistema. En la prueba se enviaron cadenas de 5 mensajes de texto con diferentes tiempos de envío entre cada mensaje. En la tabla se puede observar que la mayor tasa de transferencia que se logró obtener con el sistema fue de 424bps. Sin embargo, esta tasa de transferencia genera un 60% de error con respecto al número de mensajes enviados y el número de mensajes recibidos. Debido a esto se intento reducir el porcentaje de error por medio de una ampliación a 5 segundos en el lapso de tiempo entre cada mensaje. El resultado fue una mejoría en el porcentaje de error a un 40% con una tasa de transferencia de 256 bps. Por último se procedió a determinar el lapso de tiempo requerido para reducir el porcentaje de error a 0%. Esto se logró ampliando el lapso de envío entre cada mensaje a 8 segundos como mínimo para obtener una taza de transferencia de 160 bps.

Con respecto al nivel de autonomía del sistema utilizando una batería de 9V, se utilizó una rutina de envío de mensajes con una iteración de un minuto de lapso entre cada mensaje enviado. Esto generó un tiempo de descarga de la batería de una hora aproximadamente.

6.2.6 Consumo de corriente durante la aplicación de descrita para enviar y recibir mensajes de texto

En esta etapa se procedió a graficar el consumo de corriente del dispositivo mediante el analizador de fuentes de potencia KEITHLEY. En la figura 46 se observa el comportamiento del consumo de corriente del controlador durante 70 segundos que duró aproximadamente la ejecución del programa.

Durante las primeras 81 muestras de la aplicación, se logra observar un cambio importante en el comportamiento de la gráfica de corriente. Esto se debió a que durante ese período de tiempo, el modem EGDE-USB ejecuta las peticiones de

conexión a la infraestructura de telecomunicaciones del ICE. Durante este proceso el consumo de potencia alcanza aproximadamente el valor de 109mW el cual se justifica en base al periodo de tiempo que dura la conexión de la línea.

En el segundo comportamiento de la gráfica, se logra observar un periodo de estabilidad de la señal, en la cual el consumo de corriente no supera los 150 mA instantáneos. Esta disminución en el consumo de la corriente se debe a que el modem se encuentra en estado de configuración interna en conjunto con el controlador Host USB. A pesar de la disminución en el consumo de la corriente, este periodo en el cual la aplicación se encuentra en estado de bajo consumo, en total es la etapa en donde existe un mayor consumo de potencia. Esto debido a que el factor tiempo de ejecución toma un papel importante en esta medida de potencia. Esta etapa consume aproximadamente 325mW, ya que su tiempo de ejecución supera los 30 segundos de duración, mientras que las otras ejecuciones no superan los 12 segundos de operación.

En las etapas 3 y 4 de la figura 47, se denota el comportamiento del consumo de corriente durante los procesos de recepción y envío de mensajes de texto. Se puede resaltar que el tiempo de ejecución de estos procesos es relativamente pequeño y por ello se justifica un consumo aproximado de 45mW por aplicación. Ver tabla 9.

6.2.7 Verificación de la funcionalidad de la aplicación para acoplar un modem EDGE-USB por medio de una interfaz de hardware y software utilizando SIWA-RTOS.

La última etapa de este análisis está dirigida a corroborar la funcionalidad del el sistema completo que se propuso como solución del problema.

Por medio de la validación de datos transmitidos desde el modem EDGE-USB hasta el controlador PIC32MX460512 se comprobó que la efectividad de la transmisión de los datos es prácticamente de un 100% ya que por medio de los resultados obtenidos en la figura 45 se puede destacar que el único error que comete la aplicación es al

sobrepasar el tiempo de espera NAK propuesto por el protocolo USB 2.0, cuyo valor típico de tiempo es de 20ms aproximadamente. Sin embargo este tiempo se modificó hasta un máximo de 30ms debido a que algunas respuestas del modem no duran siempre lo el mismo tiempo y también porque el protocolo establece los 30ms como un máximo de tiempo de espera.

La verificación de las cadenas de datos transmitidas a través de la red WAN-GSM fue exitosa ya que por medio de varias pruebas de control, se validó que la información enviada desde el controlador siempre fue la misma que se recibió en la estación de control. Ver tabla 6.

En última instancia, se comprobó que el sistema operativo de tiempo real se acopló en un 100% a la interfaz de comunicaciones. Esta comprobación solo se pudo realizar asignando un nivel de máxima prioridad a la tarea para enviar un mensaje de texto.

Esta asignación de prioridad se debe a que la ejecución de esta tarea se lleva a cabo en un amplio período de tiempo. Por lo tanto, si existe otra tarea de mayor prioridad en el momento de la ejecución de la tarea SMS_TASK, el mensaje de texto no será enviado debido al desfase de tiempos de respuesta que existirá entre el modem EDGE-USB y el controlador host PIC32MX460512.

7 Capítulo 7: Conclusiones y Recomendaciones

7.1 Conclusiones

1. La adaptación de la aplicación para enviar y recibir mensajes de texto desde el sistema operativo de tiempo real SIWA-RTOS, permitió brindar un servicio de comunicación de datos con un alcance de transmisión proporcional cobertura de telecomunicaciones del ICE. Aproximadamente un 70% del territorio nacional.

2. La transmisión de datos por medio de el acople a nivel de hardware y software entre el controlador y el modem EDGE-USB tuvo un porcentaje de efectividad del 100%.

3. El consumo de potencia durante el envío de un mensaje de texto se cuantificó en unos 42mW mientras que el consumo durante recepción de un mensaje fue de 43mW aproximadamente. Por lo tanto, se concluyó que la aplicación de envío y recepción de mensajes de texto es una rutina de bajo consumo de potencia y por lo tanto, es ideal para futuras aplicaciones en la plataforma de redes inalámbricas CRTECMote.

4. La información enviada desde el controlador es efectivamente la misma recibida en la estación de control, esto comprueba una índice de efectividad en la transmisión de un 100%.

5. La reestructuración al modelo de configuración ACM-Serial Emulation, permitió ejecutar comandos bajo el protocolo V2.5ter AT en un modem de tipo GSM-USB.

7.2 Recomendaciones

1. Configurar el modem EDGE-USB para crear una conexión de internet y utilizar el servicio de correos electrónicos.

2. Utilizar el controlador PIC32MX795512 para manejar el sistema operativo y las aplicaciones que se deseen incorporar a la red de sensores.

3. Para obtener una mayor eficiencia en la rutina de recepción de mensajes de texto, se recomienda incorporar las transferencias de tipo interrupción del protocolo USB 2.0.

4. Mantener actualizado la aplicación USB-Host con las nuevas actualizaciones de Microchip.

5. Otorgar un nivel máximo de prioridad de ejecución a la tarea de envío de mensajes de texto a la hora de configurar SIWARTOS

6. Tomando en cuenta que los tiempos de respuesta de las radio bases del ICE no son siempre los mismos, se recomienda mantener un mínimo de 10 segundos entre el envío en cadena de mensajes de texto.

7. Se recomienda realizar un estudio más detallado, y bajo características mejor diseñadas para caracterizar el consumo de potencia de el sistema funcionando en modo autónomo.

Bibliografía y Referencias

[1] Escuela de Ingeniería Electrónica. ***Guía Informe Final*** [en línea]. [Cartago: Instituto Tecnológico de Costa Rica], 16 noviembre 2006. <http://www.ie.itcr.ac.cr/acarrasquilla/Proyecto/ > [Consulta: 10 octubre de 2009]

[2] Developershome. **Introduction to GSM / GPRS Wireless Modems**. [En Línea]. Última modificación 26 noviembre 2009. Última visita: 10 diciembre 2009. URL: http://www.developershome.com/sms/GSMModemIntro.asp

[3] Ericsson. **Block Diagrams** [En Línea]. Última modificación 26 noviembre 2009. Última visita: 10 diciembre 2009 URL: http://www.stericsson.com/block_diagrams/pnx4908_block_diagram.JPG

[4] CMP Books. **Real-Time Concepts for Embedded Systems.** [En Línea]. Última modificación:Enero 2003. Última visita: 10 diciembre 2009. URL: http://vinodkrishnankutty.com/share/docs/RTOS.pdf

[5] Guías Costa Rica. **Areas Protegidas (Parques Nacionales).** [En Línea]. Última modificación: Enero 2003. Última visita: 10 diciembre 2009. URL: http://www.guiascostarica.com/areas.htm

[6] Stremler, F.G. **"Introducción a los sistemas de comunicación". 2da Ed.,** Addison Wesley Iberoamericana

[7] EURAM – Informática. **¿Qué es un Sistema Operativo?** [En Línea]. Última modificación: Junio 2000. Última visita: 14 diciembre 2009. URL: http://www.euram.com.ni/pverdes/verdes_informatica/informatica_al_dia/que_es_un_so_144.htm

[8] Alexander Leiva. **Sistema de administración en tiempo real de los recursos de hardware y software de un nodo de sensado** [En Línea]. Última visita: 12 diciembre 2009. URL: http:// http://groups.google.co.cr/group/crtecmote?hl=es.

[9] Dennis Rodríguez. **Organización de la información generada por una red inalámbrica de sensores: Load Point CRTECMOTE** [En Línea]. Última visita: 24 enero 2010. URL: http:// http://groups.google.co.cr/group/crtecmote?hl=es.

[10] Microchip Technology Inc.. **PIC32MX460F512L Datasheet**. Revisión E. Julio 2008.

[11] Microchip Tecnology Inc... **PIC18f4550 Datasheet**. Revision E. Agosto 2008

[12] Multi-Tech Systems, **Inc. Multitech Wireless EDGE Modem**. Revisión A. Julio 2005.

[13] Microchip Technology Inc.. **Pic_32_Compiler_Library_Guide** Revisión D. Julio 2009.

[14] Microchip Technology Inc.. **PIC32_Compilers_User_guide**. Revisión B. Julio 2009.

[15] Jan Axelson **"USB COMPLETE". 4ta Ed.,** LakeView Research.

[16] Bose (Boseji), Abhijit. **L.A.T.H.I.** [En Línea]. Última visita: 18 Febrero 2009. Última actualización: 28 octubre 2009. URL: http://sourceforge.net/projects/lathi/.

[17] Di Jasio, Lucio. (2008). "***Programming 32-bit Microcontroller in C, Exploring the PIC32***".

[18] USB Implementers' Forum. **Universal Serial Bus Class Definitions For Communication Devices**. Revisión 1.1. Enero 1999.

[19] USB Implementers' Forum. **Universal Serial Bus Specification**. Revisión 2.0. Abril 2000.

[20] Barry, Richard. (2009). "***A Practical Guide: Using The FreeRTOS Real Time Kernel***". Versión 1.0.4. FreeRTOS.org.

[21] Google Books. **Comandos AT.** [En Línea]. Última modificación: Enero 2006. Última visita: 6 junio 2010. URL: http:*alarmagsm.googlecode.com/files/**COMANDOS**%20AT.doc*

Apéndices

A.1 Glosario y Abrebiaturas

WAN: (Wide Area Network o Red de Area Amplia) Término que describe un red que cubre una gran área a nivel local.

GSM: (Global System for Mobile Communications) originalmente de *Groupe Spécial Mobile*),es un standard de telefonía celular.

RTOS: (*Real Time Operating System* en inglés), es un sistema operativo que ha sido desarrollado para aplicaciones de tiempo real.

API: *Application Programming Interface*. En español: “Interfaz de Programación de Aplicaciones”. Es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Half-Duplex: Sistema de comunicación que transmite y recibe datos pero no simultáneamente.

ASCII (American Standard Code for Information Interchange): es un código de caracteres basado en el alfabeto latino tal como se usa en inglés moderno y en otras lenguas occidentales.

Microcontrolador: Dispositivo programable usado en electrónica para aplicaciones especificas. Trabaja en forma muy similar a un microprocesador pero con la diferencia de que cuenta internamente con otros dispositivos tales como memorias, convertidores analógico-digitales, temporizadores, comparadores, etc.

Host USB: Se denomina Host USB al controlador encargado de reconocer y administrar los recursos operativos de un dispositivo USB.

MODEM (Modulador-Demodulador): es un periférico utilizado para transferir información entre varios equipos a través de una técnica de transmisión y recepción de señales eléctricas denominada Modulación y Demodulación.

A.2 Hoja de información de la empresa

Información del estudiante

Nombre: Miguel Fonseca Porras

Cédula: 1-1204-0324 **Carné ITCR:** 200236802

Dirección de su residencia en época lectiva: Heredia, San Francisco

Dirección de su residencia en época no lectiva: Heredia, San Francisco

Teléfonos en época lectiva: 87042899 **Teléfonos en época no lectiva:** 22372235

Email: migue.fonseca@gmail.com **Fax:** 22372235

Información del Proyecto **Duración en meses: 6**

Nombre del Proyecto: Desarrollo de un sistema convertidor de protocolos para comunicación inalámbrica vía GSM, arquitectura abierta CRTECMote

Área del Proyecto: Investigación

Profesor Asesor: Johan Carvajal Godínez

Información de la Empresa

Nombre: Instituto Tecnológico de Costa Rica

Zona: Cartago, Cartago, Central

Dirección en la cual se ubica Ud. dentro de la empresa:

Su teléfono en la empresa: **Su extensión en la empresa:**

Fax: **Apartado:**

Actividad Principal de la empresa:

Información del asesor en la empresa

Nombre: Ing. Johan Carvajal Godínez

Puesto que ocupa: Profesor

Departamento: Ingeniería Electrónica

Profesión: Ing. Electrónico **Grado académico:** Licenciado

Teléfono: 25509171 **Ext.:** **Fax:**

Email: johan.carvajal.g@gmail.com

A.2 Intensidad de señal del comando AT+CSQ en dbm