ESCUELA POLITECNICA DEL EJÉRCITO

DEPARTAMENTO DE ELECTRICA Y ELECTRONICA

CARRERA DE INGENIERIA EN ELECTRONICA Y TELECOMUNICACIONES

PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERIA

CONFIGURACIÓN Y ANÁLISIS DEL SISTEMA DE TRANSMISIÓN Y RECEPCIÓN DEL PICOSATÉLITE CUBESAT KIT

Juan Fernando Balarezo Serrano

SANGOLQUI – ECUADOR

2012

CERTIFICACIÓN

Certificación por parte del Director y Codirectora de la elaboración del proyecto bajo su dirección, y pie de firmas.

Ing. Darío Duque
DIRECTOR

Ing. Vanessa Vargas
CODIRECTORA

DEDICATORIA

A Dios, por ser quién me guía en los caminos de la vida.

A mis padres, quienes en todo momento me han apoyado y motivado a sacar adelante todas las metas que me he propuesto tanto en el ambiente personal como en el profesional, gracias a sus enseñanzas he logrado ser quien soy. Gracias al amor, ejemplo y educación que he recibido siempre de ellos y seguiré recibiendo, han dado como resultado que alcance todos mis objetivos por más grandes que estos fuesen.

A mi hermana, que con su cariño y apoyo ha sido de gran ayuda para alcanzar las metas que me he propuesto, y que nunca olvide que de igual forma siempre contará con mi apoyo, cariño y ejemplo para llevar a cabo todo lo que se proponga.

A mis abuelitos que en vida siempre me dieron mucho amor y siempre con buenos consejos supieron guiarme mientras crecía. Y que ahora desde el cielo se me cuidan y velan por mi.

AGRADECIMIENTOS

Al CEINCI y al Dr. Alfonso Tierra quienes fomentaron y son iniciadores del proyecto *"ESPE-CubeSat"* con lo que se dio la oportunidad de que yo me involucre en el mismo para llevar a cabo mi proyecto de grado.

A los Ingenieros Darío Duque y Vanessa Vargas, director y codirectora de este proyecto de grado, por el apoyo y la guía brindada en todo el desarrollo del proyecto.

A mis amigos de universidad, con los que compartí 5 años de nuevas experiencias y una amistad sincera, que a su vez me demostraron que en la universidad sí se puede encontrar amigos verdaderos brindándome siempre su apoyo y cariño.

PROLOGO

Este proyecto de grado presenta la respuesta a las necesidades tecnológicas presentadas para fomentar e iniciar la investigación de las ciencias espaciales en la Escuela Politécnica del Ejército.

El proyecto está dividido en 7 capítulos: el primero es una introducción al proyecto *"ESPE-Cubesat"* en donde se abarca el alcance del proyecto y de este proyecto de grado; el segundo capítulo se realiza un estudio del estándar *CubeSat* para abarcar los características de los Picosatélites; en el tercer capítulo se realiza un análisis del hardware y software utilizado para llevar a cabo este proyecto de grado; el cuarto capítulo se enfoca en un estudio del sistema de referencia con el que contamos, es decir, el código que tenemos como base para realizar la configuración del microcontrolador MSP430; el quinto capítulo se refiere al análisis de la configuración e integración de hardware y software con el modem MHX 2420 para llevar a cabo la aplicación de comunicación de datos en tiempo real; el sexto capítulo abarca el análisis de los resultados obtenidos; el séptimo capítulo se enfoca en las conclusiones y recomendaciones generadas después de haber culminado el proyecto de grado.

ÍNDICE DE CONTENIDO

Contenido

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS

CAPITULO 1

1. INTRODUCCIÓN

1.1 ANTECEDENTES

Dado que en el Ecuador no se registran proyectos referentes a la investigación y desarrollo científico en cuanto a temas satelitales y espaciales, el Centro de Investigaciones Científicas de la Escuela Politécnica del Ejército (CIENCI) optó por llevar a cabo el proyecto *"CubeSat Kit"* que tiene varias fases para ser el pionero en este ámbito. El proyecto tiene como objetivo poner en órbita un Picosatélite, con fines de investigación científica orientado fundamentalmente al desarrollo de aplicaciones realizadas por estudiantes universitarios o de postgrado a bajo costo; siendo esta la gran ventaja frente a proyectos con satélites de mayor dimensión [1]. Este ambicioso proyecto consta de varias fases. Dentro de la primera fase, una parte primordial es la configuración e integración del *CubeSat Kit*, objeto del presente proyecto de grado.

Las especificaciones del Picosatélite CubeSat Kit se han convertido en una base para la mayoría de diseños de Picosatélites. Es así que en países como Colombia, Perú, Venezuela, Chile, proyectos similares han sido realizados con éxito [1].

1.2 JUSTIFICACIÓN E IMPORTANCIA

Si se compara el proyecto *CubeSat Kit* con las misiones satelitales tradicionales, se puede notar que los proyectos con una plataforma de desarrollo como el *CubeSat* tienen gran potencial para ayudar al auto aprendizaje de las personas involucradas en el proyecto e implementar misiones exitosas y de gran utilidad especialmente en los ámbitos científicos e industriales. Adicionalmente tiene la gran ventaja de que sus costos son mucho menores a aquellos requeridos por las misiones tradicionales antes mencionadas.

Las aplicaciones realizadas sobre la plataforma de desarrollo *CubeSat Kit* son generalmente únicas. Dado que se utiliza una base previa, el desarrollo de estos proyectos dura menos tiempo que una misión satelital tradicional y el *CubeSat Kit* está diseñado para realizar una misión completa y exitosa en la menor cantidad de tiempo posible. Sin embargo, requieren una planeación meticulosa y un trabajo minucioso para maximizar las probabilidades de éxito.

El desarrollo de este proyecto de grado impulsará el estudio aeroespacial en la Escuela Politécnica del Ejército, al ser el primero en este ámbito. Servirá como base para emprender proyectos de lanzamiento de Picosatélites al espacio, que servirán para investigación y estudios científicos de los cuales se verá beneficiada la Universidad, la comunidad científica y la sociedad en general, al desarrollarse aplicaciones que favorezcan al bien común.

Mediante este proyecto de grado se entenderá el funcionamiento y manejo del Picosatélite CubeSat-ESPE, se detallará como configurarlo e integrarlo de forma adecuada para un desempeño correcto, al optimizar los recursos de hardware y software que se van a manejar.

1.3 ALCANCE DEL PROYECTO

Este proyecto de grado permite configurar el software con el que se generarán aplicaciones tales como establecer comunicaciones en tiempo real con el módem MHX 2420. De esta forma se motivará el desarrollo e investigación espacial en el Ecuador para emprender aplicaciones tales como telemetría, análisis a través de sensores, mediciones para verificar el comportamiento climático, etc. Adicionalmente se pretende integrar el hardware en el case del *CubeSat* de tal forma que se encuentre armado de forma correcta.

A fin de configurar e integrar el sistema se realizará un programa en el ambiente de desarrollo de *CrossWorks* que funcionará en conjunto con el sistema operativo en tiempo real *Salvo*, el cual será cargado en el microcontrolador MSP430 de *Texas Instruments*. El microcontrolador va a ser el cerebro del módulo de desarrollo del *CubeSat*; al que se le va a integrar un módem MHX 2420, que es configurado mediante códigos AT y mhx programados en *CrossWorks*. Cabe mencionar que la configuración del módem es enviada desde el microcontrolador. Adicionalmente es necesario, para comprobar la transmisión de datos del *CubeSat*, configurar otro módem para poder establecer una comunicación punto a punto con el *CubeSat* y así ejecutar la aplicación de transmisión de datos en tiempo real. A fin de realizar la configuración de este módem extra se requiere de una computadora con conexión serial para ingresar los comandos AT y mhx, usando un terminal ASCII, y estos comandos serán enviados al módem. Una vez configurados todos los módulos se procederá a realizar la integración del hardware.

1.4 OBJETIVOS

1.4.1 General

Configurar e integrar el Picosatélite *CubeSat Kit*, basado en el RTOS (Sistema Operativo en Tiempo Real) *Salvo* usando el microcontrolador TI-MSP430, con módems MHX 2420 en la banda de 2.4 GHz.

1.4.2 Específicos

- Configurar el software de un sistema embebido usando la tarjeta de desarrollo *CubeSat Kit* con un microcontrolador RISC-16 bits TI-MSP430.
- Usar el RTOS *Salvo* para aplicaciones de tiempo real, en ambientes asincrónicos y multitarea para el *CubeSat-ESPE*.
- Configurar los módems MHX 2420 acoplados a la tarjeta de desarrollo.

CAPITULO 2

2. EL CUBESAT

Las especificaciones técnicas del CubeSat son una base para Picosatélites de medidas 10x10x10 cm y de hasta 1,33 kg. Las cuales fueron propuestas por las Universidades de Stanford, San Luis Obispo y la California Polytechnic State University en 1999 para proveer un estándar para el diseño de Picosatélites y de esta forma fomentar y ayudar a que las universidades de todo el mundo desarrollen proyectos de investigación científica y exploración en el espacio con costos y tiempos reducidos. El trabajar sobre un estándar permite promover un rápido desarrollo y además se cuenta con experiencias de otras personas sobre otras misiones.

Actualmente, el proyecto CubeSat se ha convertido en una organización (cubesat.org) que tiene por objetivo dar la oportunidad a las universidades de realizar misiones espaciales. Cuenta con la colaboración internacional de más de 100 universidades, colegios y empresas privadas, quienes trabajan en el desarrollo de Picosatélites que llevan cargas con aplicaciones científicas para empresas privadas y gubernamentales.

Con lo que el desarrollo e investigación en este ámbito mayoritariamente proviene de entidades académicas, sin embargo varias compañías privadas están interesadas en la investigación y desarrollo de estos Picosatélites. Los CubeSats pueden llevar cargas para misiones relativamente simples, por ejemplo cámaras de video con cargas como plantas para demostraciones de comportamiento de especímenes en el espacio. Además estos Picosatélites tienen un tiempo de vida aproximado entre 3 a 9 meses y se los lanza a la órbita LEO (Low-earth Orbit).

Entre las características de los Picosatélites las más importantes se puede decir que su infraestructura es simplificada, lo que permite diseñar y producir un satélite de bajo costo comparado con un tradicional, que requieren de millones de dólares en inversión.

Los CubeSats son un medio rentable e independiente para poner cargas en órbita. La mayoría de CubeSats llevan en sí por lo general uno o dos instrumentos de experimentación científica como su carga principal de la misión que se esté llevando a cabo. Algunas compañías e institutos de investigación ofertan regularmente oportunidades de lanzamientos grupales de varios cubos, empresas tales como ISC Kosmotras y Eurokot.

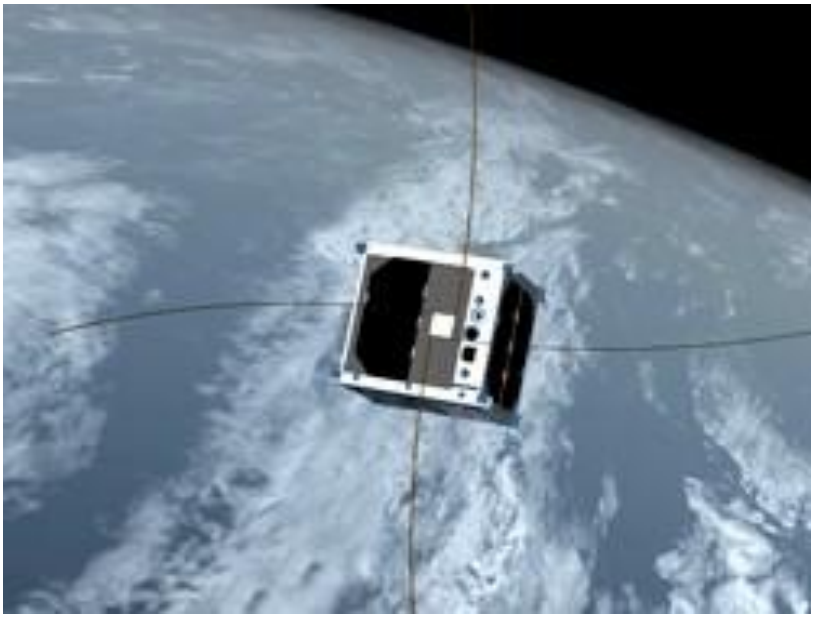

Fig. 2.1: CubeSat en el espacio. Fuente: www.cubesatkit.com

2.1 ESPECIFICACIONES DE DISEÑO

El término CubeSat es usado para clasificar a Picosatélites que cumplan con los estándares de diseño especificados en el *CubeSat Design Specification*, el cual fue publicado por la California Polytechnic con la colaboración y guía del profesor de ingeniería aeroespacial Jordi Puig-Suari, y de Bob Twiggs del departamento de aeronáutica de la Universidad de Stanford. Sus estudios se han centrado en desarrollar

CubeSats desde instituciones educacionales. Las especificaciones y características del CubeSat no se aplican ni son iguales que otros Picosatélites en forma de cubo como el Picosatélite *MEPSI* de la *NASA*, el cual es de mayor tamaño que el CubeSat.

El CubeSat básico antes mencionado de 10x10x10 cm también es conocido como CubeSat "1U", que quiere decir una unidad. Los CubeSat son escalables a lo largo de un solo eje y en incrementos de 1U. Por ejemplo CubeSats 2U tendrían dimensiones de 20x10x10 cm y para los 3U sus dimensiones serían 30x10x10 cm, ambos ejemplos ya han sido construidos y lanzados incluso.

Fig. 2.2: CubeSat. Fuente: www.cubesatkit.com

Debido a que los CubeSat son todos 10x10 cm, independientemente de su longitud (que es escalable), estos pueden ser lanzados y desplegados usando un sistema de transporte y despliegue común. Por lo cual los CubeSats son generalmente lanzados y desplegados desde un mecanismo común llamado *Poly-PicoSatellite Orbital Deployer (P-POD)*, también desarrollado y construido como su nombre así lo indica por la *Cal Poly(California Polytechnic)*.

Fig. 2.3: P-POD. Fuente: www.cubesatkit.com

2.1.1 Requerimientos Generales

- Para los CubeSats diseñados con alguna modificación respecto a las especificaciones de diseño de CubeSats se debe solicitar una aprobación de la Cal Poly para garantizar que pueden ser lanzados con éxito una vez estudiado su diseño.

- Todas las partes del Cubesat deben permanecer adheridas al mismo durante el lanzamiento, expulsión y operación. No se deben generar desechos espaciales.

- No se permite pirotecnia adherida al CubeSat, es decir productos inflamables.

- No se permitirán recipientes a presión donde esta sea mayor a 1.2 atm, además los recipientes a presión deberán tener un factor de seguridad mínimo de 4.

- El total de energía química almacenada no deberá superar a los 100 W/h.

- No se debe llevar como carga materiales peligrosos.

- Los materiales integrados al CubeSat deberán cumplir el siguiente criterio para reducir la emisión de gases que puede contaminar otros equipos durante la integración, pruebas y lanzamiento:

    - Pérdida Total de Masa debe ser menor o igual al 0,1%

    - El acumulado de materiales volátiles condensables debe ser menor o igual al 0,1%.

- La versión oficial de las especificaciones de diseño del CubeSat es la última revisión publicada (http://www.cubesat.org/images/developers/cds_rev12.pdf), la cual deben cumplir todos los desarrolladores de CubeSat.

- La Cal Poly enviará actualizaciones respecto a las especificaciones de diseño o cualquier tipo de actualización referente a los CubeSat a la lista de entidades registradas como desarrolladoras o que están desarrollando proyectos con CubeSats.

2.1.2 Requerimientos Mecánicos

Los CubeSats son Picosatélites cuya longitud nominal es de 100 mm por lado. Las características mecánicas generales de todos los CubeSat incluyen:

- En cuanto a las Dimensiones Externas el CubeSat deberá usar un sistema de coordenadas definido como en la Figura 2.4. El lado –Z del CubeSat debe ser ingresado primero en el P-POD.

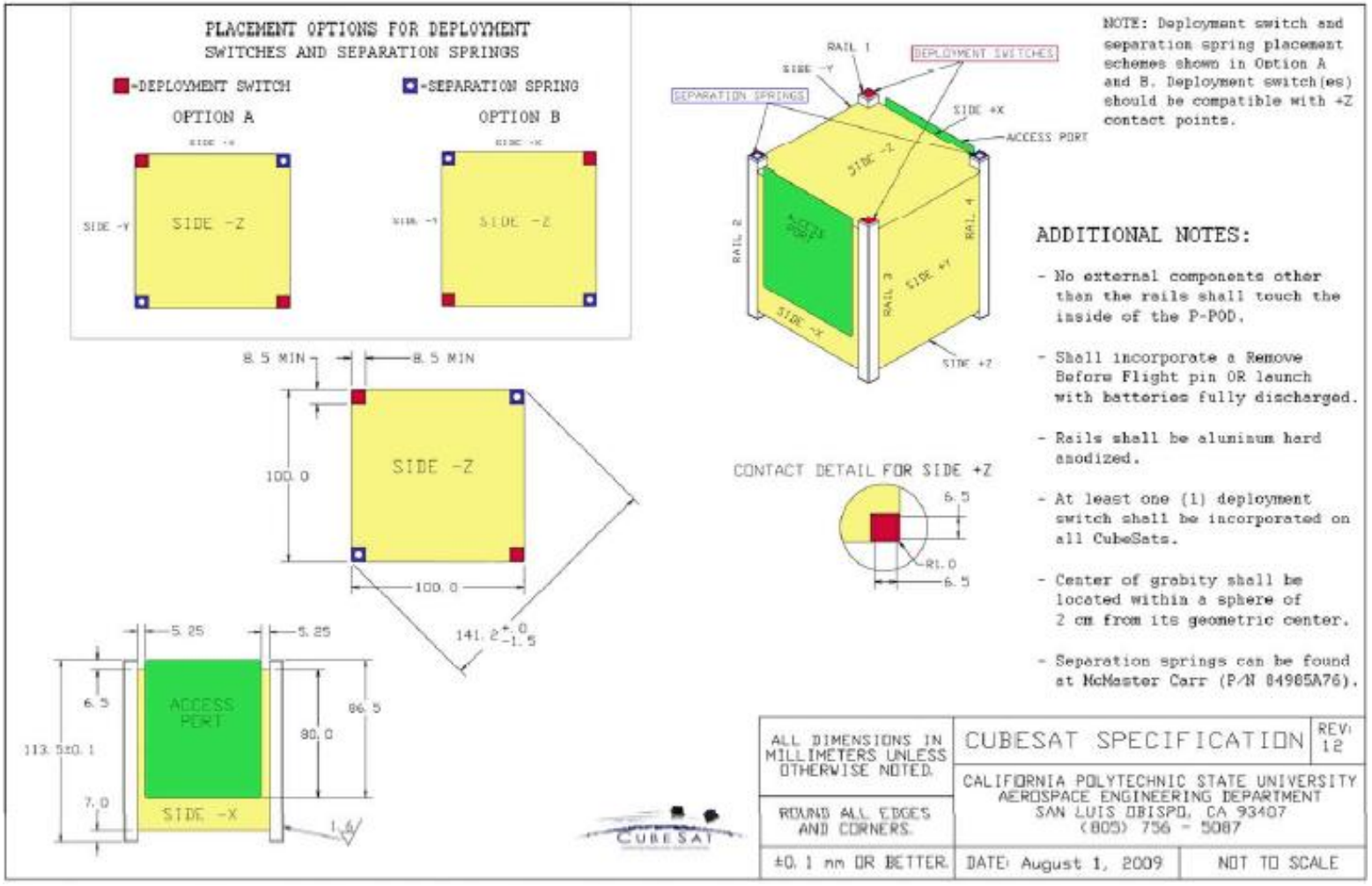

Fig. 2.4: Diagrama de las Especificaciones de Diseño del CubeSat. Fuente: www.cubesatkit.com

- Las dimensiones físicas deben ser según lo especificado en la Figura 2.4.
- El ancho del CubeSat será de 100.0 ± 0.1 mm. (Dimensiones de X y Y especificadas en la Figura 2.4).
- El alto de un Cubesat de 1U debe ser 113.5 ± 0.1 mm. (Dimensiones de Z según la Figura 2.4).
    - Un CubeSat de 3U debe tener 340.5 ± 0.3 mm de alto.
- Los componentes externos, que lleve el CubeSat no deberán tener contacto con la superficie interna del P-POD, únicamente los rieles podrán contactar dicha superficie.
- Si el CubeSat tuviese elementos desplegables, los mismos están restringidos de tal forma que no podrán tener contacto con el P-POD.
- Los rieles deben tener un espesor mínimo de 8.5 mm.
- La rugosidad de la superficie de los rieles no puede ser mayor a 1,6 um.
- Los bordes de los rieles deben tener al menos 1 mm de radio.
- Las terminaciones de los rieles en el lado +Z (Figura 2.4) deben tener un área mínima de contacto de 6,5 mm x 6,5 mm para colindar con las rieles del CubeSat.
- Por lo menos el 75% de la riel del CubeSat debe estar en contacto con las rieles del P-POD, el 25% restante de los rieles puede ser omitido, pero en ningún momento la riel deberá del Cubesat deberá exceder el 100% de su tamaño.
- Para CubeSats de 1U por lo menos debe haber un contacto con el riel de 85,1 mm; mientras que en CubeSats de 3U debe haber un contacto de al menos 255,4 mm.
- Cada CubeSat 1U no podrá exceder los 1,33 kg de masa, mientras que los de 3U no pueden exceder los 4,0 kg.

- El centro de gravedad del CubeSat deberá ser ubicado dentro de una esfera de 2 cm desde su centro geométrico.

- Los materiales usados para la estructura principal y rieles del CubeSat deben ser de Aluminio 7075 o 6061. Si se desean usar otros materiales es necesario reportarlo para verificar la factibilidad de usar los mismos.

- Las rieles y material del Cubesat que estén en contacto de igual forma con las rieles del P-POD, deben ser de aluminio anodizado para prevenir soldaduras en frío.

- El CubeSat deberá usar resortes de separación con las características establecidas en la Figura 2.5. Estos resortes proveen separación entre el despliegue de cada CubeSat del P-POD. No se requieren resortes de separación para CubeSats de 3U.

Characteristics	*Value*
Plunger Material	*Stainless Steel*
End Force Initial/Final	*0.5 lbs. / 1.5 lbs.*
Throw Length	*0.05 inches minimum above the standoff surface*

Fig. 2.5: Características Resortes de Separación. Fuente: www.cubesatkit.com

2.1.3 Requerimientos Eléctricos

Los sistemas electrónicos deben ser diseñados de acuerdo a las siguientes consideraciones de seguridad:

- Ningún elemento debe estar activado durante el lanzamiento para prevenir cualquier interferencia eléctrica o RF que afecte al vehículo de lanzamiento o las cargas principales. Los CubeSats deben estar totalmente desactivados durante el lanzamiento o se puede lanzar con baterías descargadas.

- El CubeSat debe incluir al menos un interruptor tipo *push button* conocido como *deployment switch* en un riel designado para apagar por completo la energía una vez desactivado al entrar en contacto con el riel de los P-PODs. Todos los sistemas deben estar apagados, incluyendo relojes de tiempo real.

- Para permitir que se carguen las baterías de los CubeSats y se lleven a cabo diagnósticos de los mismos una vez integrados a los P-PODs, los conectores deben estar dentro de la ubicación correspondiente al Puerto de Acceso. Estas tareas se realizarán mientras el interruptor de despliegue este desactivado.

- El CubeSat debe incluir un pin *Remove Before Flight (RBF)*, o debe ser lanzado con las baterías completamente descargadas. El pin RBF debe ser accesible desde el Puerto de Acceso, además debe cortar toda la energía del Picosatélite una vez que es insertado el listón RBF que activa el pin. El listón RBF no debe sobresalir 6,5 mm de los rieles cuando esta completamente insertado en el Picosatélite.

2.1.4 Requerimientos Operacionales

Los CubeSats deben cumplir con ciertos requisitos referentes a la integración y funcionamiento del mismo para cumplir con obligaciones legales y garantizar la seguridad de otros CubeSats.

- Los CubeSats que lleven baterías deben tener la capacidad de recibir un comando de apagado. Este requerimiento fue emitido por la Comisión Federal de Comunicaciones (FCC).

- Todos los elementos desplegables, como antenas y paneles solares, deben esperar al menos 30 minutos después de que el interruptor de despliegue haya sido activado después de la eyección del P-POD para empezar a desplegarse.

- Los transmisores RF cuya potencia sea superior a 1 mW, deben esperar un mínimo de 30 minutos después de que el interruptor de despliegue haya sido activado después de la eyección del P-POD para empezar a transmitir.

- Los operadores de los Picosatélites deben obtener y proveer la documentación pertinente en cuanto a licencias y permisos para el uso de frecuencias.

- El tiempo de vida de caída orbital de los CubeSats deberá ser inferior a 25 años después del final de la vida de la misión. Se deberá obtener y presentar la documentación de la aprobación de un plan de mitigación de desechos orbitales de la FCC.

- La Cal Poly llevará a cabo pruebas para verificar e inspeccionar el hardware que el CubeSat puede ser integrado al P-POD. Una prueba final de acoplamiento se llevará a cabo antes del lanzamiento. Se usará un checklist de aceptación del CubeSat estandarizado por la Cal Poly para verificar el cumplimiento de los requerimientos y especificaciones de los CubeSats.

2.2 APLICACIONES TIPICAS

Existe una gran gama de aplicaciones que se pueden desarrollar con los CubeSats, está en quienes participan de este tipo de proyectos el determinar que aplicación desarrollar y cuyas limitaciones son básicamente ajustarnos al volumen del Picosatélite.

2.2.1 Desarrollo de la tecnología CubeSat

- Se pueden desarrollar Picosatélites CubeSat para evaluar la tecnología del mismo y demostrar las capacidades del concepto del CubeSat, para así demostrar de manera exitosa al público el funcionamiento del CubeSat. Esto se podría llevar a cabo instalando una cámara a bordo, pero las limitaciones que se tendrían son la durabilidad de las baterías y la fuerza de las señales de radio que se reciba.

- Otra aplicación que serviría para demostrar las funcionalidades del CubeSat y su operación sería llevar a bordo un sensor de radiación que emita estos datos hacia la Tierra y así analizar los niveles de radiación que se tienen en el CubeSat una vez que está en el espacio.

- Se pueden generar experimentos para desarrollar experiencia en cuanto a aspectos diferentes de diseño y operación de los CubeSat por ejemplo diferentes formas de acoplar los paneles solares, antenas y demás periféricos, en los cuales se necesitaría más inversión para distintos lanzamientos y verificar que diseños de CubeSats tuvieron un mejor comportamiento en cuanto a su operación en el espacio.

- Se pueden llevar a cabo misiones que prueben control y tecnologías de navegación de los CubeSat.

- Además en los CubeSat se pueden acoplar paneles solares, los cuales recolectarán energía para así alargar la vida útil del CubeSat en el espacio y poder desarrollar aplicaciones y misiones que duren más tiempo.

2.2.2 Observación de la Tierra

Se pueden desarrollar aplicaciones para mejorar los mecanismos de detección de terremotos, mediante la evaluación de señales magnéticas, las cuales pueden mostrar valores que indiquen o pronostiquen el inicio de un terremoto. La compañía Quakefinder que se encuentra realizando un proyecto es este ámbito se encuentra recolectando datos sobre las fluctuaciones del campo magnético que podrían estar asociadas a los terremotos.

Además se podría realizar aplicaciones para detección de eventos naturales de otra índole como erupciones volcánicas. Esta aplicación sería muy útil para países y zonas geográficas con alta densidad de volcanes como lo es el Ecuador. Otra aplicación podría ser la detección de tormentas en zonas tropicales. El campo de observación de la Tierra conlleva un gran número de aplicaciones que pueden ser desarrolladas en sistemas CubeSat y que son de gran utilidad para la colectividad humana. Por lo tanto este tipo de aplicaciones es la que debería tener más apoyo económico por parte de entidades gubernamentales.

2.2.3 Biotecnología

Otro tipo de aplicaciones a desarrollar pueden entrar en ámbitos como la Biotecnología y la Biología, por ejemplo se puede llevar como carga dentro del CubeSat una cámara que registre el comportamiento de alguna especie en el espacio ya sea de flora o alguna especie de bacteria o parásito para fines investigativos.

Existe ya un Picosatélite lanzado en el 2006, el *Minotaur* de la NASA, para realizar investigaciones y experimentos genéticos. Se enfocaron estudiar y analizar el comportamiento y crecimiento de la bacteria E. Coli. La meta de este proyecto fue establecer métodos para estudiar los cambios genéticos provocados al estar expuestos al espacio exterior.

Actualmente el proyecto CubeSat tiene la colaboración internacional de más de 100 universidades, colegios y empresas privadas, quienes trabajan en el desarrollo de Picosatélites que llevan cargas con aplicaciones científicas para empresas privadas y gubernamentales.

2.3 P-POD (POLY PICOSATELLITE ORBITAL DEPLOYER)

El P-POD es el sistema estandarizado de despliegue para los CubeSats, es decir es el medio estándar por mediante el cual son lanzados los CubeSat. Es capaz de llevar tres CubeSats estándar y sirve como interfaz entre el CubeSat y el Vehículo de Lanzamiento. El P-POD es una caja rectangular que tiene una puerta y un mecanismo de resortes mediante el cual son expulsados los CubeSats. Los P-POD son hechos de aluminio anodizado, que quiere decir que su superficie posee una mayor protección contra la abrasión y la corrosión. Una vez que el mecanismo de despliegue del P-POD es activado por una señal enviada por el Vehículo de Lanzamiento, un conjunto de resortes de torsión en la bisagra de la puerta son usados para forzar y abrir la misma, una vez realizado esto los CubeSats son lanzados por el resorte principal deslizándose usando sus propias rieles y las rieles del P-POD.

Los Cubesats se deslizan a través de una serie de rieles antes que sean eyectados a orbita. Se debe asegurar que el CubeSat con el que se este trabajando sea compatible con el P-POD para asegurar el éxito de la misión, para ello es importante basarnos en las especificaciones de diseño del CubeSat publicadas por la *California Polytechnic State University*.

Los P-PODs son montados en un vehículo de lanzamiento y llevan a los CubeSats a órbita, y los despliega una vez recibida la señal del vehículo de lanzamiento. Los P-

PODs han desplegado alrededor del 90% de todos los CubeSats lanzados hasta la fecha, incluyendo lanzamientos fallidos. Y el 100% de CubeSats que han sido lanzados desde el 2006 fueron lanzados en P-PODs. El P-POD Mk III tiene la capacidad para tres CubeSats de 1U, sin embargo dado que el tamaño de tres CubeSat de 1U es igual al de 3U y dos de 1U son iguales a uno de 2U este P-POD puede desplegar CubeSats cuyas combinaciones de unidades y tamaños lleguen a un volumen máximo de 3U.

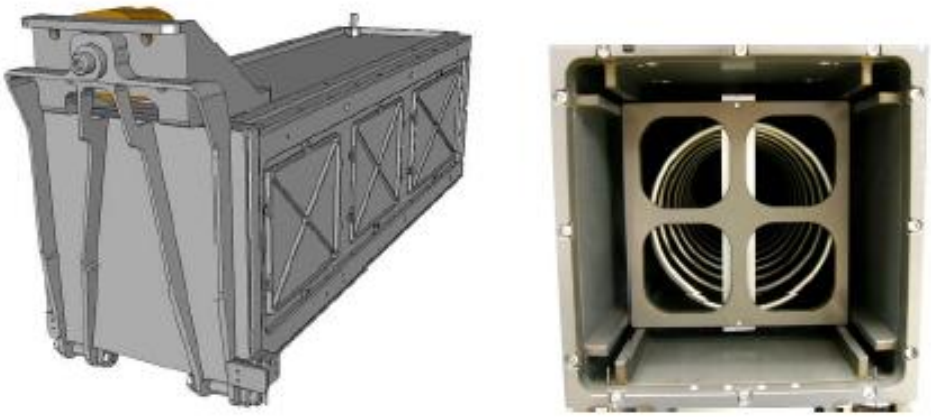

Fig. 2.6: Vistas del P-POD. Fuente: www.cubesatkit.com

CAPITULO 3

3. ANÁLISIS DE HARDWARE Y SOFTWARE DEL CUBESAT

Como todo sistema embebido, el CubeSat Kit consta de Hardware y Software, además de herramientas de desarrollo. En este capítulo se describirá y analizará el contenido del mismo.

3.1 CUBESAT KIT, PUMPKIN INC.

El CubeSat Kit de Pumpkin Inc. es un kit comercial que puede ser comprado de forma libre, diseñado exclusivamente para reducir tiempo y costos en la construcción de un satélite funcional con un diseño mecánico optimizado. La reducción de costos permite su accesibilidad a entidades académicas como universidades.

Fig. 3.1: CubeSat Kit de Pumpkin Inc. Fuente: www.cubesatkit.com

Pumpkin Inc. se asocio con MASSIF, empresa que aportó en el diseño mecánico, para en conjunto crear el CubeSat Kit. Pumpkin se encuentra comprometido en el desarrollo de proyectos relacionados con el espacio, todos sus esfuerzos están enfocados en desarrollar un excelente producto (CubeSat Kit) para así garantizar el éxito de las misiones y el lanzamiento de los CubeSat Kit.

La meta mundial de los proyectos CubeSat es entregar un paquete de 10cm de arista y cuya peso sea de hasta 1 kg que pueda ser lanzado a la órbita LEO de la Tierra a un precio bajo. Este reto se está logrando mediante un proceso de co-diseño de Hardware y Software del CubeSat Kit. Así se minimiza el tamaño y consumo de energía del sistema permitiendo maximizar la carga que se puede llevar. El Cubesat Kit de Pumpkin Inc. ofrece máxima flexibilidad mediante un diseño integrado.

El CubeSat Kit se adhiere y cumple por completo a las especificaciones de diseño de los CubeSats, incluyendo los requerimientos Eléctricos, Mecánicos y Operacionales. El módulo de vuelo FM430 cuenta con un microcontrolador RISC y un sistema de bus completo de 80 pines para expansión. El microcontrolador, que consume una cantidad realmente pequeña de energía menor a los 100mW, y puede ser pre programado con el Sistema Operativo en Tiempo-Real (RTOS) Salvo para desarrollar el software de forma más eficiente, óptima y rápida. Adicionalmente permite instalar módulos de comunicación (tranceivers) en el modulo de vuelo sin realizar ninguna modificación adicional.

Por otra parte, Pumpkin Inc. promueve al RTOS Salvo como la solución premier para microcontroladores embebidos en un solo chip. Salvo es óptimo para los

microcontroladores que pueden ser usados en el CubeSat Kit. De esta forma se mantienen optimizadas la electrónica y la distribución de elementos dentro del CubeSat a pesar de las limitaciones de energía y espacio en cada Cubesat.

El CubeSat Kit incluye el hardware y software necesario para un desarrollo y despliegue completo de la solución y misión.

3.1.1 Hardware

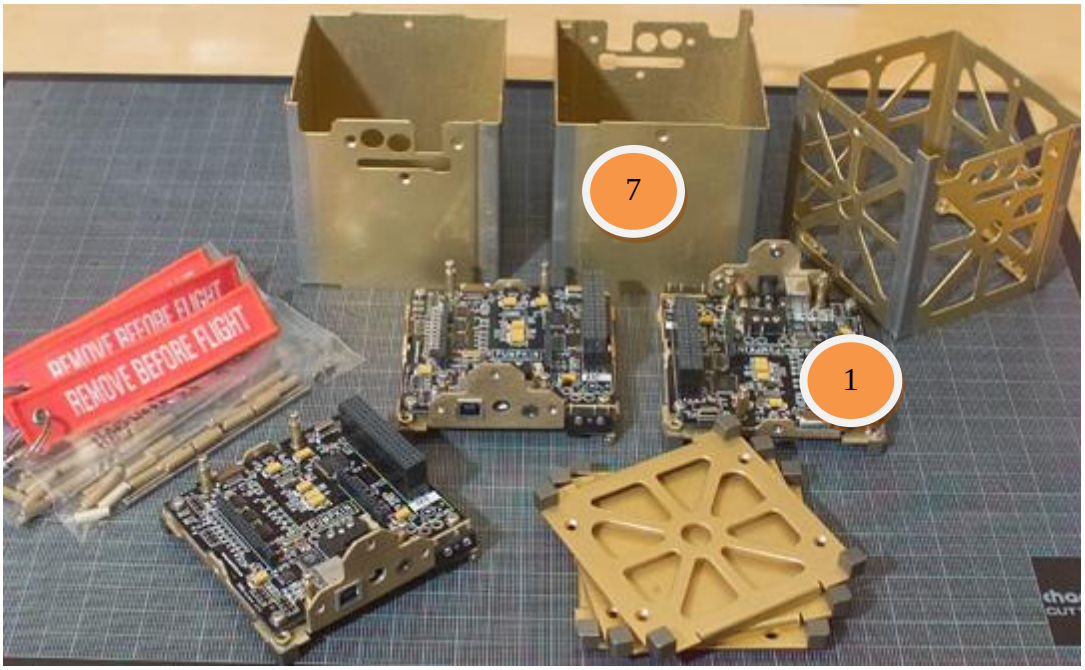

Fig. 3.2: Módulos de vuelo. Fuente: www.cubesatkit.com

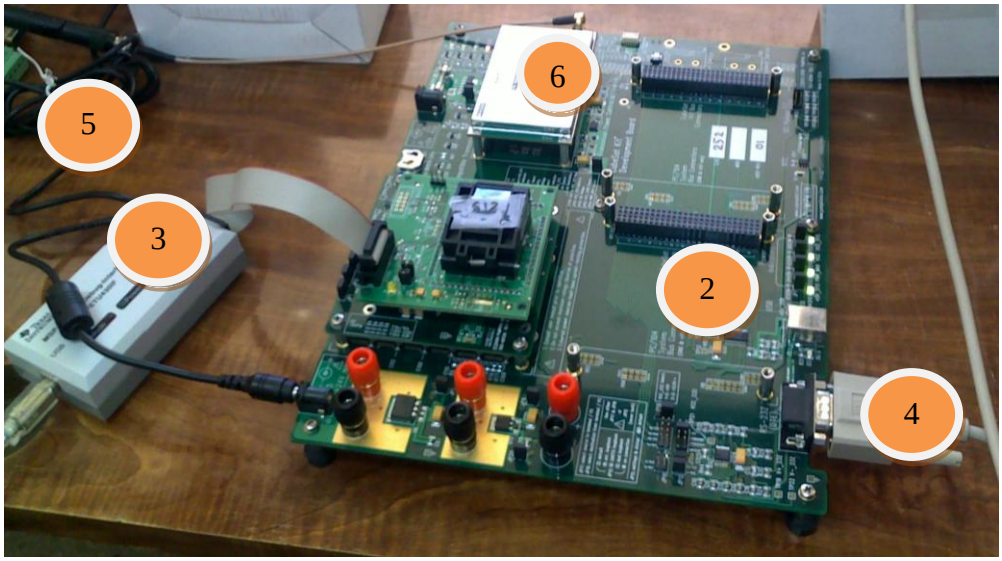

Fig. 3.3: Elementos CubeSat Kit. Fuente: www.cubesatkit.com

1 Módulo de Vuelo FM430.

2 Contiene tanto el Tablero de Desarrollo (Development Board) más el Módulo del Procesador (Pluggable Socketed Processor Module) para el desarrollo del proyecto en laboratorio y realización de pruebas. Microcontrolador Texas Intruments MSP430 F1612 de 16 bits.

3 MSP 430 USB Debug Interface (Marca Texas Instruments, MSP-FET U430IF, con cable USB incluido).

4 Cable Planar.

5 Fuente de poder (Input: 100-240 V AC, 0,5 A; Output: 8-11V DC, 1,63-2,25 A)

6 Módulo de Comunicaciones MHX2420-SL Radio OEM. El CubeSat Kit acepta una variedad de módulos de comunicación. La serie MHX 425/900/910/920/920A/2400/2420 de módulos de comunicación que manejan espectro ensanchado (spread-spectrum) del fabricante Microhard Corp. son soportados por el Cubesat Kit, y el módulo que se usó para el desarrollo de esta tesis es el MHX2420. Para optimizar el consumo de energía la alimentación del módulo de comunicación es controlado vía software mediante el microcontrolador.

7 Estructura CubeSat completa y lista para ser lanzada de tamaño 1U, con alta resistencia, liviano y un volumen interno grande para las cargas a llevar.

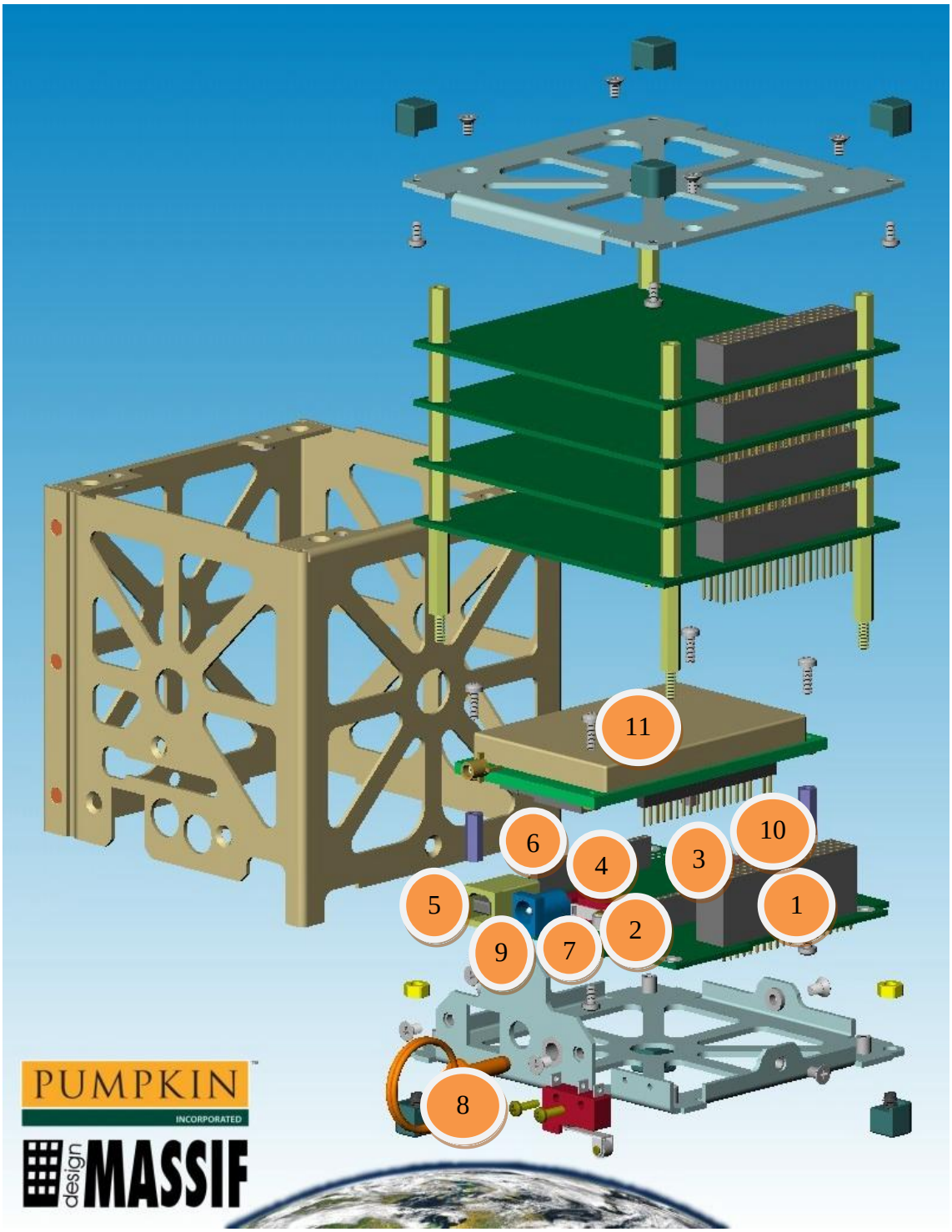

Fig. 3.4: Módulos de vuelo. Fuente: www.cubesatkit.com

El módulo de vuelo contiene:

1 Tarjeta Madre (Motherboard)

2 Módulo del Procesador (Pluggable Processor Module).

3 El Bus de comunicaciones de 104 pines del CubeSat Kit que interconecta todos los módulos del CubeSat Kit mediante un único conector confiable y apilable, no se requiere cables adicionales.

4 El conector PPM del CubeSat Kit de 100 pines para conectar los Módulos del Procesador.

5 Puerto USB 2.0

6 Socket compatible con el módulo de comunicación MHX

7 Socket para tarjeta SD

8 El interruptor *Remove Before Flight*

9 Conector de alimentación externa con protección contra sobrevoltajes, sobrecorrientes y voltajes inversos; conectores de energía y tierra PC/104.

10 Microcontrolador Texas Intruments MSP430 F1612 de 16 bits.

11 Módem MHX2420.

3.1.2 Software

- Ambiente de Desarrollo (IDE) CrossWorks MSP430, de Rowley Associates, que serán explicados más adelante en el capítulo 3.2.2.

- Pumpkin SALVO Pro RTOS para procesadores TI-MSP430, que serán explicados más adelante en el capítulo 3.2.21.

- Pumpkin CubeSat Kit MSP430 Software (código fuente, librerías). Cada CubeSat Kit tiene un set de librerías especificas para el procesador que se está usando, para controlar y manejar los recursos e interfaces del CubeSat Kit. Estas librerías se encuentran en código fuente e incluyen funciones como: Habilitar y controlar la interfaz del tranceiver, habilitar y controlar la interfaz USB, habilitar

y controlar la interfaz SD, controlar el encendido y apagado del equipo, realizar el apagado del sistema para un consumo mínimo de energía.

- Pumpkin Software Libraries para SALVO para microcontroladores TI MSP430. Cada CubeSat Kit además contiene un set de librerías específicas para el procesador que se está usando para la inicialización y uso de los periféricos del chip. Estas librerías se encuentran en código fuente e incluyen funciones para controlar los UART del chip, que son los quienes manejan los puertos y dispositivos seriales.

3.2 HARDWARE EMBEBIDO

A continuación se procederá a explicar el Hardware que usa el presente proyecto.

3.2.1 Tablero de Desarrollo

El Tablero de Desarrollo usado es de la cuarta generación de Pumpkin cuya característica principal es que son computadoras en una sola placa o tablero, diseñados para el uso en el CubeSat Kit u otras aplicaciones. El propósito del Tablero de Desarrollo es ser una plataforma de laboratorio para entrenamiento, desarrollo, depuración y realización de pruebas con el CubeSat Kit. Sus componentes están organizados de tal forma que se provee máximo acceso simultáneo a todos los subsistemas de la arquitectura del CubeSat Kit que están presentes en el Tablero de Desarrollo. Cabe recalcar que el Tablero de Desarrollo no está diseñado para que encaje en ningún tipo de CubeSat, es más los CubeSats son desarrollados usando los Tableros de Desarrollo, luego los módulos de los CubeSats y el software cargado en el Tablero de Desarrollo una vez probados y evaluados son migrados al módulo de vuelo del CubeSat en el proceso de integración.

El Tablero de Desarrollo cuenta con todas las características de la Tarjeta Madre del módulo de vuelo, y con otras características que lo hacen adecuado para trabajar en laboratorio. Entre estas características están: Jumpers en línea para aislamiento de circuitos y mediciones de corriente, dos fuentes de alimentación de de +5V y +3V, un puerto serial configurable RS-232 con conector DB-9, LEDs indicadores del estado de la alimentación, interruptor de Reset, todos los circuitos accesibles sobre un área grande que facilita las mediciones, el uso de osciloscopios, entre otras tareas.

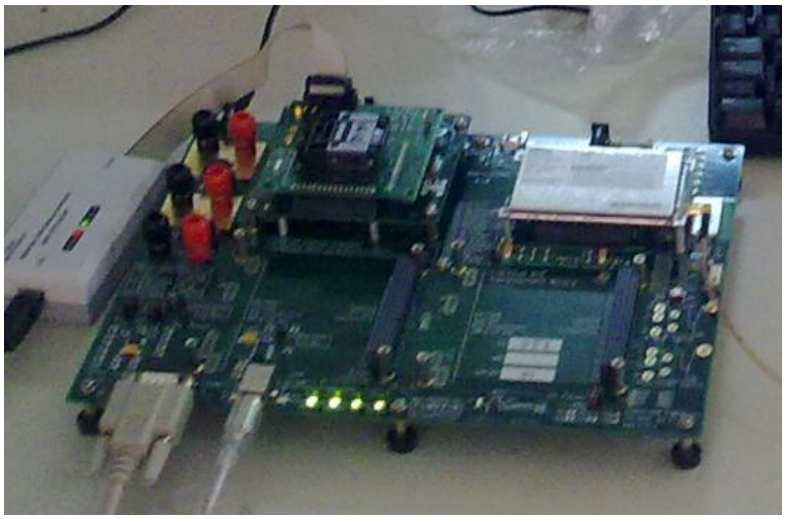

Fig. 3.5: Tablero de Desarrollo Rev. D

La configuración por defecto de los Jumpers del Tablero de Desarrollo es la siguiente:

Tabla 3.1: Configuración por defecto de los Jumpers del Tablero de Desarrollo

Jumper	Configuración
JP1	OFF
JP2	ON

Jumper	**Configuración**
JP3	ON
JP4	OFF
JP5	ON
JP6	OFF
JP7	ON
JP8	ON
JP9	ON
JP10	ON
JP11	OFF
JP12	OFF
JP13	OFF
JP14	2-3

El Tablero de Desarrollo acepta tensiones de entrada entre 6-24 V y requiere los siguientes valores:

Tabla 3.2: Tensiones de Funcionamiento del Tablero de Desarrollo

Señal	Ubicación	Valor
+5V	TP9	+5V
VCC	TP12	+3.3V
VCC_MCU	TP20	+3.3V
VCC_232	TP21	+3.3V
V+_232	TP19	>+5V
V-_232	TP22	<-5V
+5V_SW	TP10	0V
-RST/NMI	TP8	+3.3V

Al ser un elemento de desarrollo y evaluación el Tablero de Desarrollo tiene características y elementos adicionales comparados con la Tarjeta Madre, estos son:

- Un Bus de comunicaciones extra para soportar un segundo stack de módulos.
- Una segunda fuente de alimentación DC, con reguladores locales de +5V y +3.3V.
- Un puerto RS-232 configurable con conector DB-9.
- Variedad de Jumpers para configuración y aislamiento de circuitos.

- Posibilidad de aceptar procesadores tipo socket.

La Tarjeta Madre del módulo de vuelo es un subconjunto eléctricamente idéntico al Tablero de Desarrollo. Cuando son equipados con el mismo microcontrolador son compatibles completamente incluso en el código.

El Tablero de Desarrollo posee un socket para acoplar el módulo de procesador (Pluggable Proccesor Module, PPM), el mismo que puede ser de Pumpkin, otras marcas o puede ser creado por usuarios finales del CubeSat Kit. El sistema permite usar una amplia gama de microcontroladores con un PPM adecuado, en este caso se utiliza el MSP430 de Texas Instruments. El Tablero de Desarrollo tiene un esquema de energía flexible que permite el uso de PPMs con requerimientos diferentes de energía de entrada y salida.

El Tablero de Desarrollo provee un socket para el PPM al cual llegan las señales de elementos de entrada y salida y las señales de energía mediante el Bus de comunicaciones del CubeSat Kit, así como llegan algunas señales dedicadas y otras de propósitos especiales.

Todos los periféricos de entrada y salida que se encuentran en el Tablero de Desarrollo (Interfaz MHX, USB, Tarjeta SD) están diseñados para interactuar con los PPMs con cualquier voltaje de entrada y salida dentro del rango de +1.65V a +5.5V.

3.2.2 Módulos de Comunicación MHX2420

El MHX2420 es un modem inalámbrico que trabaja en la banda de 2.4 GHz usando la tecnología de modulación de Espectro Ensanchado por Saltos de Frecuencia (Frequency Hopping Spread Spectrum, FHSS). En el Ecuador la banda de 2.4 GHz no se encuentra licenciada por tanto el uso del espectro de dicha banda no es restringido y se puede usarlo para este proyecto. Este módem inalámbrico puede ser optimizado para establecer comunicaciones rápidas y de largas distancias, sobre los 50 Kilómetros y de alto rendimiento en una variedad de topologías de red.

Fig. 3.6: MHX 2420

Proveen transferencia inalámbrica de datos asíncrona de forma segura entre la mayoría de tipos de equipos que emplean interfaces RS232, RS422 o RS485.

A continuación en la figura 3.7 se muestra el Pin Out del MHX2420:

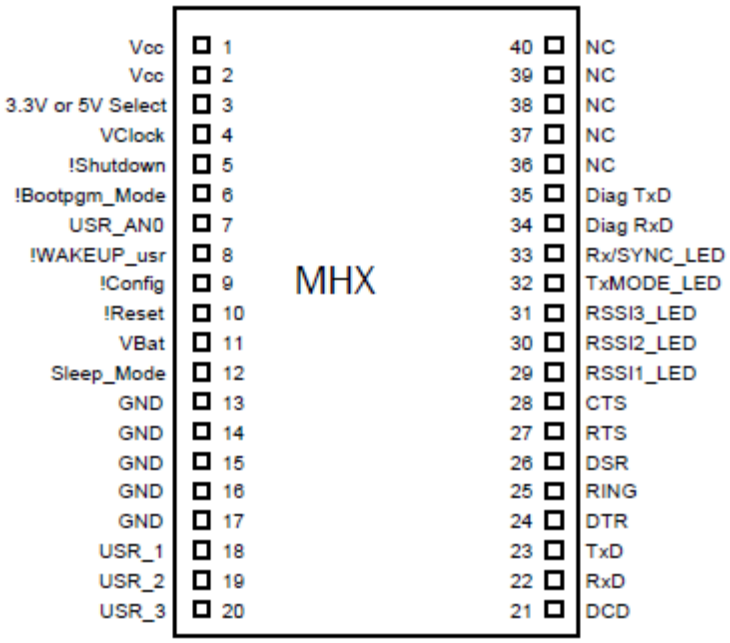

Fig. 3.7: Pin Out MHX 2420

En el Anexo C1 se detalla la descripción de pines.

Entre las características más importantes del MHX2420 están:

- Enlace transparente, de baja latencia, provee un throughtput continuo de 230 kbps.
- Comunicaciones con todo tipo de PLCs, RTUs y dispositivos seriales.
- Temperaturas industriales soportadas.
- Aplicaciones Punto a Punto, Punto a Multipunto, Repetidoras, TDM, etc.
- Potencia de transmisión máxima de 1W (30dBm).
- Bajo consumo de energía en Modo Sleep.
- 32 bits de CRC, tipo de FEC (Forward Error Correction) seleccionable, el FEC tiene la capacidad de retransmisión en caso de errores.
- Puerto separado para diagnósticos, los cuales son transmitidos de forma transparente para permitir la obtención de diagnósticos de forma remota y realizar control de la red en línea.

A continuación se detallan las especificaciones técnicas del modem MHX2420:

Tabla 3.3: Especificaciones Técnicas del Módulo MHX2420

Especificaciones Técnicas	
Frecuencia	2.400-2.4835 GHz
Tipo de Modulación	Frequency Hopping Spread Spectrum
Segmentos de Banda	Seleccionables mediante la opción de restricción de

Especificaciones Técnicas	
	Frecuencia
Detección de Errores	32 bits CRC. ARQ
Rango	Más de 50Km (Depende de la tasa de transmisión y de la línea de vista)
Potencia de Transmisión	Desde 100mW hasta 1 W (30dBm)
Sensibilidad	-108dBm
Ganancia del Sistema	142dB (Con antenas Rubber Duck, ver figura 3.8, el módulo soporta antenas Rubber Duck y Yagui)
Interfaz Serial	RS232/RS485/RS422
Velocidad de Transmisión de la interfaz Serial	300 bps hasta 230.4 kbps
Velocidad de Transmisión del enlace	19.2 kbps hasta 230.4 kbps
Modos de Operación	Punto-Punto, Punto-Multipunto, TDMA, Multimaster, Peer-to-Peer, Store & Forward Repeater
Interfaces	RxD1, TxD1, RTS, CTS, DCD, DSR, DTR, RxD2, TxD2, LEDs RSSI, LEDs Tx/Rx, Reset

Especificaciones Técnicas	
Diagnósticos	Voltaje, Temperatura, RSSI y Diagnósticos Remotos
Voltaje de Core	4.0VDC hasta 5.5VDC
Voltaje en las Interfaces de Entrada y Salida	3.3VDC hasta 5.5VDC
Conector de la Antena	MCX
Temperatura de Operación	-40ºC hasta +85ºC; 5-95% de humedad no condensada
Peso	55 gramos
Dimensiones	89mm x 53.4mm x 17.8 mm
Regulaciones de Radio	FCC Apartado 15.247; IC RSS210

Fig. 3.8: Antena Tipo Rubber Duck

Su pequeño tamaño y alto rendimiento permiten que sea ideal para algunas aplicaciones, entre ellas se encuentran:

- Redes SCADA.
- Telemetría Remota.
- Control de Tráfico.
- Control Industrial.
- Monitoreo Remoto.
- GPS.
- Transmisión Inalámbrica de Video.
- Robótica.
- Sistemas CubeSat Kit.

Para aplicaciones con el CubeSat Kit, el tablero de desarrollo está diseñado de tal forma que permite integrar directamente los módulos inalámbricos MHX2420 en la interfaz

destinada para el mismo. De igual forma su Módulo de Vuelo soporta este tipo de módulos de forma directa, es decir, no es necesario realizar ningún tipo de acople extra.

La interfaz en la que se debe integrar el módem MHX2420 es alimentada con +5V y requiere una cantidad de energía considerable del CubeSat. El módulo de vuelo ve al módulo de comunicaciones inalámbrico como DCE (Data Communications Equipment).

El módulo MHX2420 puede ser configurado de varias formas a continuación los métodos usados en el presente proyecto: Configuración mediante comandos AT y el otro método fue Configuración mediante el CubeSat Kit. Ambos métodos se detallan a continuación.

Configuración mediante comandos AT

Este proceso se realiza mediante comandos AT. Para realizar esta configuración en el módulo MHX2420 es necesario un computador con Terminal ASCII instalado (HyperTerminal, Putty, etc) y también se debe contar con el kit de desarrollo de Microhard Corp. que incluye:

- Tablero de desarrollo propio del fabricante del módem (Diferente al Tablero de Desarrollo del CubeSat Kit)
- Cable serial directo (9 pines Macho a 9 pines Hembra)
- Antena tipo Rubber Duck para ser conectada al módulo de comunicación MHX2420.
- 2 cables seriales con terminales RJ45 y DB9, cada uno con una diferente configuración y funcionalidad.

Una vez integrado este sistema como se ve en la figura 3.9, se procede a realizar la configuración.

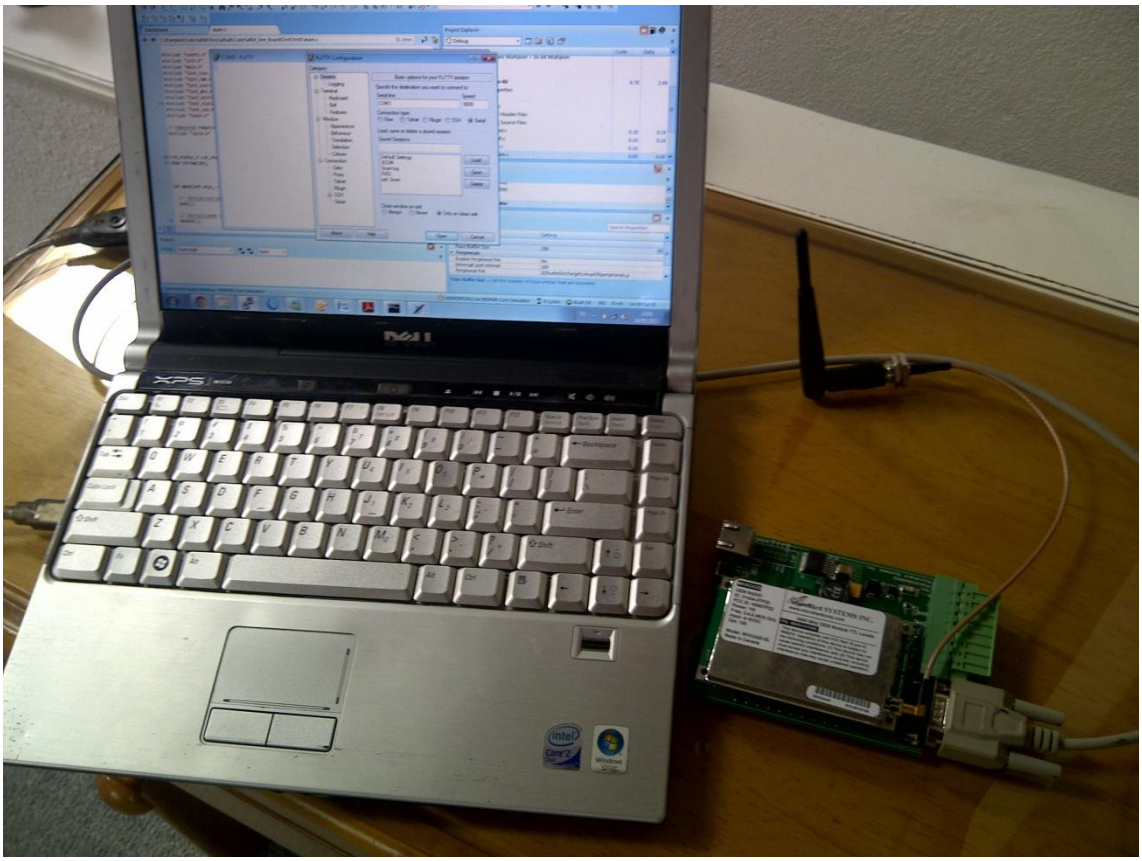

Fig. 3.9: Tablero de Desarrollo Microhard Corp. conectado a una computadora.

El tablero de desarrollo del módulo MHX2420 visto desde frente se muestra en la figura 3.10. En el cual se puede observar la interfaz de diagnósticos (SERIAL DIAG), la cual posee un conector RJ45 y es usada para dos propósitos:

- Obtener diagnósticos en línea y realizar configuraciones usando el software MHS (Propio del fabricante e incluido con la compra del kit de desarrollo MHX2420) y conectando el cable serial negro con terminales RJ45 a DB9 del tablero a una computadora.

- Realizar una actualización del firmware, usando el cable serial azul con terminales RJ45 a DB9.

Cabe recalcar que la interfaz SERIAL DIAG no es un puerto Ethernet ni soporta ingreso de comandos AT. La configuración que se puede realizar a través de esta interfaz es mediante el software MHS.

En la parte frontal también se puede encontrar el botón CFG que es usado para entrar al modo de configuración o ingreso de comandos del equipo.

Además el tablero cuenta con 6 LEDs indicadores los cuales son:

- LED de estatus del sistema. Está ubicado en el panel frontal del conector RJ45, debe estar encendido en color verde. Se ilumina cuando el sistema está encendido, la energía conectada y el estatus del sistema en conjunto no presenta problemas. Cuando se está en modo de comandos es el único LED encendido.

- LED de Transmisión (TX). Según la vista frontal, de izquierda a derecha es el primer LED, cuando está encendido en rojo indica que el módulo MHX2420 se encuentra transmitiendo datos sobre el aire.

- LED de Recepción (RX) y sincronismo (SYNC). Según la vista frontal, de izquierda a derecha es el segundo LED, cuando está iluminado en verde indica que el módem MHX2420 está sincronizado y que recibe paquetes válidos.

- LEDs indicadores de la fuerza de la señal recibida. Son los 3 LEDs del extremo derecho según la vista frontal de la figura 3.10. Los LEDs se iluminan de izquierda a derecha conforme aumente el nivel de la señal recibida. El nivel de la señal se calcula basándose en los 4 últimos paquetes validos recibidos con un valor correcto de CRC, este valor también se reporta en el registro S123 que almacena el módulo MHX2420.

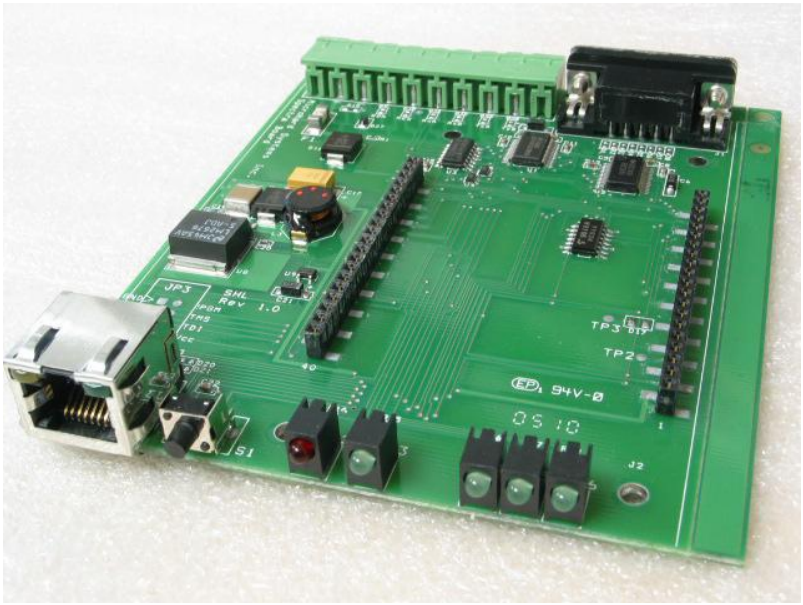

Fig. 3.10: Tablero de desarrollo MHX2420, Vista Frontal.

En la figura 3.11 se muestra el tablero de desarrollo visto desde la parte posterior, en el cual se encuentra el puerto RS232 (DCE), el cual es usado para:

- Transmisión de datos seriales RS232 (300-230, 400 bps) cuando se encuentra en modo de datos.
- Para configurar el módem cuando se encuentra en modo de comandos.

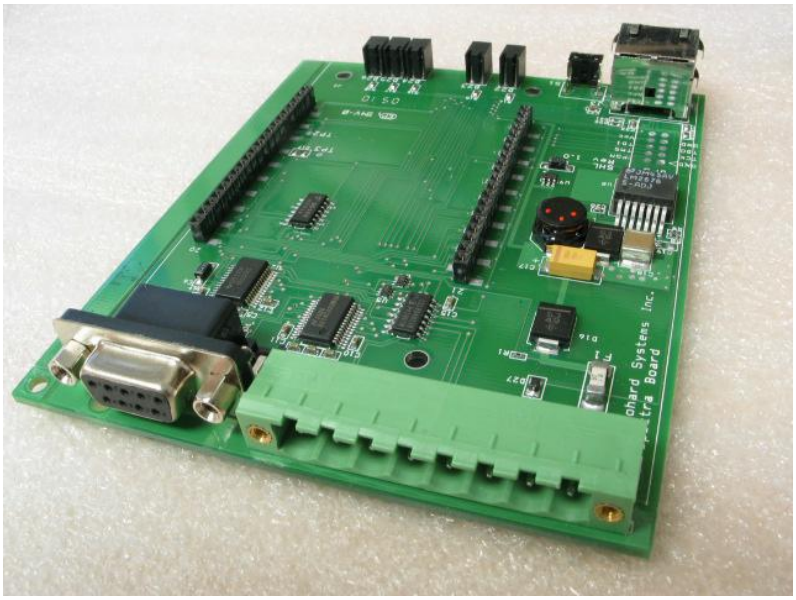

Fig. 3.11: Tablero de desarrollo MHX2420, Vista Posterior.

El puerto RS422/485 es usado para conectar a un equipo DTE (Data Terminal Equipment) que posea el mismo tipo de interfaz.

Tanto la interfaz RS232 como la interfaz RS422/485 son usadas para tráfico de datos. El Pin Out de la interfaz se muestra en la tabla 3.4.

Tabla. 3.4: Pin Out Interfaz RS232.

Número de PIN (Conector DB9)	Nombre	Entrada/Salida (I/O)
1	DCD	O
2	RXD	O
3	TXD	I
4	DTR	I
5	SG	
6	DSR	O
7	RTS	I
8	CTS	O
9	No usado	

El módulo MHX2420 usa un conector tipo MCX para la señal RF, el pigtail y cable coaxial viene incluido en el kit de desarrollo.

En el capítulo 5.4 se describe la configuración realizada por este método.

Configuración mediante el CubeSat Kit

La configuración a través del CubeSat Kit consiste en programar el microcontrolador integrado al tablero de desarrollo del CubeSat Kit, de tal forma que este envíe los comandos de configuración al módulo de comunicaciones MHX2420 acoplado a dicho tablero. Es decir, el encargado de controlar el radio MHX2420 es el microcontrolador Texas Instruments MSP430. Para realizar esta configuración es necesario tener instalado un compilador en C que sea compatible con el microcontrolador, en este caso se maneja el Ambiente de Desarrollo Crossworks MSP430 de Rowley Associates, el cual está dedicado exclusivamente para la serie del microcontrolador con el que se está trabajando.

De forma similar a la configuración mediante el tablero de desarrollo propio de Microhard Corp., los comandos deben ser enviados en el formato AT.

En el capítulo 5.3 se detalla la configuración realizada por este método.

3.3 SOFTWARE EMBEBIDO

3.2.1 Real-Time-Operating-System (RTOS) SALVO

La primera versión de SALVO fue únicamente para uso interno de Pumpkin Inc. y fue escrita en lenguaje Assembler y para uso exclusivo del microcontrolador PIC17C756 PICmicro en el año 1998. Esta versión ya contenerá la mayoría de las funcionalidades básicas que llevaron al desarrollo de futuras versiones. Después de realizar un análisis de mercadeo, Pumpkin Inc. decide reescribir la primera versión en lenguaje C. De esta forma surgieron otras opciones de configuración y optimización, ya que la versión escrita en C no solo es más robusta y flexible que su predecesor en Assembler, si no que es también completamente portable. En la tabla 3.5 se describen las versiones siguientes hasta llegar a la última.

Tabla. 3.5: Versiones SALVO.

Versión	Año	Características
2	2000	Es la primera versión comercial de SALVO, la cual poseía la característica de llevar a cabo multitareas basadas en prioridades, sin embargo fue dirigida exclusivamente para los microcontroladores PICmicro.
3	2002	Funciona con nuevos sistemas embebidos como el MSP430 de 16 bits.
4	2005	Soporta sistemas embebidos de 32 bits, tiene optimizaciones en cuanto al uso de memoria y recursos para obtener un mejor desempeño en tiempo real más flexible y robusto.

El código fuente de SALVO está escrito con el objetivo primordial de usar la menor cantidad de recursos como sea posible para el desarrollo de aplicaciones que requieran operar en tiempo real.

Salvo es el primer sistema operativo en tiempo real diseñado exclusivamente para sistemas embebidos que tienen limitaciones en cuanto a memoria y capacidad. Con Salvo se puede crear aplicaciones de bajo consumo de memoria, sin dejar de ser aplicaciones inteligentes y sofisticadas. Es propietario de Pumpkin Inc., es decir, los mismos desarrolladores del CubeSat Kit. Es decir, que SALVO es un sistema operativo en tiempo real probado, robusto y que posee un gran desempeño. Es una herramienta de software que ayudará a crear aplicaciones ambiciosas de gran alcance, confiables y sofisticadas dedicada a sistemas embebidos. Se escoge trabajar con un sistema operativo en tiempo real como SALVO, cuando la aplicaciones y operaciones que se deben llevar a cabo son de carácter criticas y deben ser ejecutadas y completadas de forma correcta en una cantidad establecida de tiempo

Pumpkin certifica que SALVO es compatible con varios microprocesadores y microcontroladores que poseen recursos limitados, ya que requiere entre 5 y 10 veces menos memoria que otros sistemas operativos en tiempo real, entre los microprocesadores y microcontroladores sobre los cuales se puede implementar SALVO se destacan los siguientes:

- ARM7TDMI de ARM.
- AVR y MegaAVR de Atmel.
- La familia S1C17 de Epson.
- M68HC11 de Motorola.
- MSP430 de Texas Instruments, el cual se está usando en el CubeSat Kit.
- PIC12/14000/16/17/18 PICmicro.
- PIC24 dsPIC.
- PIC32.
- TMS320C2000 de Texas Instruments.

Entre las principales características de SALVO están:

1 Posee una gran escalabilidad.

2 Requerimientos mínimos de memoria RAM y ROM.

3 El sistema operativo a través del kernel y los servicios que contiene es el encargado de manejar las tareas y eventos del sistema. La aplicación y programa que se realice trabaja como un solo sistema en conjunto con el RTOS. Basado en prioridades, multitareas, eventos, retardos en tiempo real, entre otras.

La cantidad de memoria ROM que requiere SALVO depende en cuanto se use el RTOS, es decir, que comandos y líneas de código específicos de SALVO se use en el programa diseñado. Los RTOS convencionales requieren de cantidades grandes de memoria RAM dedicada para cada tarea específica, mientras que SALVO al no requerir que se

mantengan siempre activas, es decir que se activan las tareas conforme se las vaya necesitando, las tareas requieren cantidades menores de memoria RAM. Con lo que la cantidad de RAM que requiere SALVO depende igualmente de la programación que se realice, pero en una aplicación como la que semaneja en el CubeSat Kit se requiere en promedio por cada tarea entre 4 a 12 bytes, cada evento necesita entre 3 a 4 bytes y entre 4 a 6 bytes más para la administración de las tareas configuradas, eventos y retardos. Lo que permite que SALVO no requiera de grandes cantidades de memoria ROM y RAM, es que no necesita almacenar datos que normalmente otros RTOS si lo requieren, al hablar de datos se refiere por ejemplo a variables locales, registros guardados y datos específicos de las tareas que dependan del compilador con el que se está trabajando.

SALVO es un RTOS netamente manejado por eventos en conjunto con multitareas, las cuales son basadas en prioridades, soportando 16 niveles independientes de prioridad. Además SALVO provee servicios empleando semáforos, mensajes, banderas, entre otros para establecer comunicaciones entre tareas y realizar una óptima administración de los recursos que se tienen.

Al referirse a tareas se quiere decir que son secuencias de instrucciones, que a veces se realizan de forma repetitiva, para realizar alguna acción específica, por ejemplo encender un LED, es decir, las tareas son programas pequeños dentro de uno más grande. Al trabajar con memoria limitada, una tarea debe tener para si misma todos los recursos del sistema sin que esto dependa de cuantas tareas se usen en la aplicación. La prioridad de una tarea indica la importancia relativa hacia otras tareas, puede ser fija o variable, única o compartida con otras tareas. Además se tiene la funcionalidad de realizar conmutación entre tareas, esto ocurre cuando una tarea deja de correr y otra empieza a ejecutarse en su lugar. El estado de una tarea describe que se encuentra haciendo la misma en determinado momento, las tareas pueden cambiar su estado a otro mediante reglas previamente definidas y configuradas. Se dice que un sistema está inactivo cuando no existen tareas ejecutándose.

Las interrupciones son eventos internos o externos que causan que la ejecución del programa se suspenda, es necesario habilitar las interrupciones para que estas puedan darse o sean reconocibles por el hardware y software. Una vez que el sistema se recupera de la interrupción, este continúa en donde quedo. Debido a su habilidad de

suspender el programa mientras se está ejecutando las interrupciones corren en primer plano, mientras que el resto del programa corre en segundo plano.

Un retardo es una cantidad de tiempo, por lo general especificada en milisegundos, durante el cual la ejecución de una tarea es suspendida. Mientras una tarea se encuentra en un retardo la misma usa una cantidad mínima de recursos para maximizar el desempeño de la aplicación que se está llevando a cabo, lo que es similar a ejecutar otras tareas que no fueron suspendidas en conjunto. Una vez que el tiempo de retardo terminó la tarea se vuelve a ejecutar en donde termino. Se puede especificar el tiempo del retardo y cuantas veces ocurrirá.

Un evento es que suceda algo que una tarea está esperando que ocurra. Cualquier parte del programa debe censar que se de un evento, con lo que permita informar a otras partes del sistema de que sucedió un evento.

La comunicación entre tareas es el paso de información de una tarea a otra. Para esto se pueden usar semáforos, semáforos binarios, mensajes y banderas.

Un *timeout* es una cantidad de tiempo, igualmente definida por lo general en milisegundos, durante el cual una tarea puede esperar a que ocurra un evento. Son opcionales puesto que las tareas pueden esperar a que suceda un evento en un lapso de tiempo indefinido.

Para trabajar con SALVO es necesario tener conocimientos básicos en lenguaje C y contar con un compilador de dicho lenguaje que sea compatible y certificado para trabajar con SALVO. Además es útil el conocer los fundamentos y características de un sistema operativo en tiempo real.

La nomenclatura que maneja SALVO para definir las librerías se muestra en la figura 3.12.

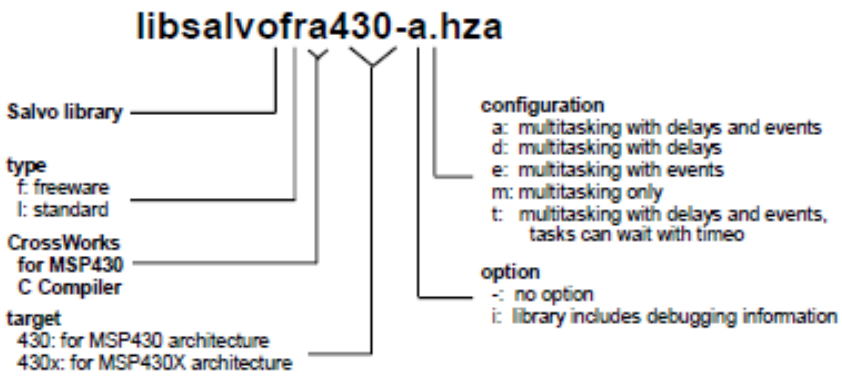

Fig. 3.12: Nomenclatura Librerías SALVO para Crosworks.

La versión de SALVO que se está usando es la 4.2.2, la cual posee las siguientes características:

- Posee 16 niveles dinámicos para dar prioridad a las tareas.
- Las tareas son manipuladas según el usuario lo requiera, es decir se pueden crear, editar y terminarlas según se requiera.
- Soporta: Semáforos binarios, semáforos, mensajes, banderas para eventos.
- Utiliza un sistema simple de temporizadores que controla los retardos de las tareas así como sus tiempos de espera de una manera muy eficiente.
- Entre 20 tareas y 30 eventos apenas llegan a consumir hasta el 15% de memoria RAM y 5% de memoria FLASH, dejando el resto para las demás aplicaciones.
- Provoca un bajo consumo de potencia, ya que el dispositivo se mantiene activo solo cuando se registra actividad en el sistema.
- SALVO se ejecuta de manera rápida y eficiente.
- Trabaja en conjunto con CrossWorks.

3.2.2 Ambiente de Desarrollo Integrado, CrossWorks de Rowley Associates

Crossworks es un compilador de lenguaje C robusto que lo convierte en un ambiente de desarrollo óptimo para la programación del microcontrolador MSP430 acoplado al CubeSat Kit, está diseñado para la programación de una gama de microcontroladores entre los cuales destacan:

- ARM.
- AVR.
- MAXQ.
- MSP430.

Se está trabajando con CrossWorks dedicado exclusivamente para el MSP430 de Texas Instruments, el cual provee un set de herramientas de desarrollo completo para el MSP430. Sirve para compilar lenguaje C ANSI, soporta macro assembler, lectura de librerías, maneja un depurador JTAG, características necesarias para configurar el CubeSat Kit.

CrossWorks permite optimizar el rendimiento del microcontrolador MSP430, ya que cumple con las normas ANSI e ISO, lo que le permite ser un compilador robusto. La versión 2 de CrossWorks soporta todas las variantes de la familia de microcontroladores MSP430, desde los MSP430 de 16 bits hasta el nuevo MSP430X. El modelo exacto del microcontrolador que se encuentra acoplado al CubeSat Kit es el MSP430F1612 el cual es soportado por el Ambiente de Desarrollo CrossWorks.

Los requerimientos para trabajar con CrossWorks son:

- Soporta los siguientes sistemas operativos de Windows: 7 (x86 y x64), Vista (x86 y x64), XP y 2000.
- Además soporta sistemas operativos de MAC OS X 10.6 Snow Leopard, 10.5 Leopard (Intel) y 10.4 Tiger (Intel).
- En Linux soporta kernel desde el 2.6 en adelante (x86 y x64), Crossworks también ha sido probado en todas las versiones de Ubuntu desde la 6.06 LTS

hasta la 11.04, es compatible con Linux openUSE 10.3, Debian 4.0 y PCLinux 2007.

- En cuanto a los requerimientos del equipo donde se va a instalar se requiere como mínimo un procesador Pentium 4 de 1GHz, con 512MB de RAM y 80MB de espacio libre en el disco.

El proceso de instalación se describe en el capítulo 5.1.2.

SALVO trabaja en conjunto con el Ambiente de Desarrollo CrossWorks MSP430 de Rowley Associates, es necesario cargar las librerías que corresponden al Sistema operativo en Tiempo Real SALVO dentro de los proyectos que se estén configurando dentro del compilador CrossWorks.

Las librerías que se están usando para el CubeSat Kit siguen la nomenclatura lalibsalvolra430it.hza, con lo que se especifica que el tipo de librerías son las estándar para el compilador C Crossworks para el microcontrolador MSP430, además las librerías incluyen información depuración, y suya configuración es apta para llevar a cabo multitareas con retardos y eventos.

CAPITULO 4

4. SISTEMA DE REFERENCIA

El sistema de referencia es el conjunto de código fuente, cabeceras, tareas y librerías ya existentes o creados por los fabricantes del CubeSat Kit, SALVO y CrossWorks, que sirvieron como guía para el desarrollo de la programación y configuración del microcontrolador MSP430 de Texas Instruments que es el cerebro del sistema.

Este proyecto describirá cada uno de los componentes que conforman el sistema de referencia y como ha sido utilizado dentro del proyecto. Cabe señalar que la plataforma de desarrollo sobre el cual se añaden estos componentes es el ambiente de desarrollo CrossWorks. Es necesario importar al proyecto todo el código que sea útil para la aplicación que se vaya a implementar, lo que permite optimizar el tiempo de trabajo partiendo de una plataforma base.

4.1 CÓDIGO FUENTE.

El código fuente de un software o programa, es el conjunto de líneas de texto o instrucciones que va a ejecutar el dispositivo inteligente. A fin de transformar a este conjunto de líneas en instrucciones entendibles por el dispositivo, se requiere de un compilador adecuado quien traducirá las líneas ingresadas de programación al lenguaje

que entiende el microcontrolador. En este caso, el cerebro del sistema es el microcontrolador MSP430, quien ejecutará la aplicación descrita en el código fuente.

4.1.1 Código Fuente CubeSat Kit

El CubeSat Kit provee de un conjunto de código fuente para trabajar con diferentes dispositivos incluyendo el MSP430. Así por ejemplo, para llevar a cabo la comunicación con el módulo inalámbrico MHX2420, es necesario importar el siguiente código fuente alstro proyecto:

- csk_io.c: Definen tareas o funciones que sirven para forzar a las interfaces de entrada y salida del CubeSat Kit[1] a tener un valor de voltaje en alto (VCC, +5V) o en bajo (GND).
- csk_led.c: Especifica tareas que permiten controlar los LEDs del tablero de desarrollo.
- csk_mhx.c: Define funciones que permiten establecer comunicación con el módulo MHX2420 y encender o apagar el módem MHX2420.
- csk_power.c: Especifica una tarea para realizar el apagado de todo el sistema del CubeSat Kit.
- csk_rand.c: Define una función que retorna un número aleatorio positivo.
- csk_usb.c: Define tareas que permiten establecer comunicación con la interfaz USB del Tablero de Desarrollo del CubeSat Kit.
- csk_wdt.c: Consta dos funciones, una permite forzar el reinicio (*reset*) del sistema mediante el *Watch Dog Timer(WDT)* del CubeSat Kit, la otra apaga el WDT.

[1] Las interfaces de entrada y salida del CubeSat Kit son 47.

En la figura 4.1 se aprecia la ubicación en el ambiente de desarrollo de los archivos de código fuente del CubeSat Kit.

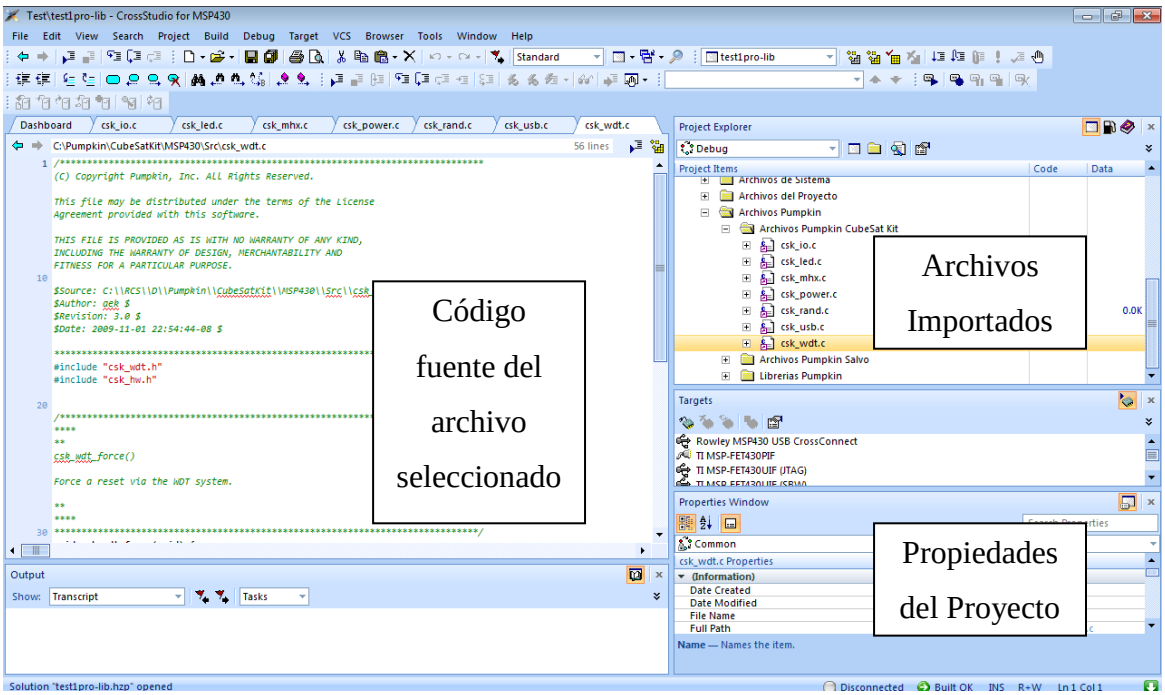

Fig. 4.1: Organización del Código Fuente CubeSat Kit en el ambiente de desarrollo

4.1.2 Código Fuente SALVO

Adicionalmente, al código fuente del CubeSat Kit, se requiere importar el código fuente del Sistema Operativo en Tiempo Real (RTOS) SALVO. A través de este se optimizará la administración de tareas para la aplicación desarrollada.

A continuación se describe el archivo de código fuente del RTOS SALVO que se uso en la configuración del CubeSat Kit:

- salvomem.c: Trabaja en conjunto con la cabecera salvoprg.h, la cual define pragmas específicos del compilador. Los pragmas sirven para controlar la ubicación de las variables globales de SALVO en el espacio designado de memoria RAM del cerebro, en este caso es el microcontrolador MSP430. El

contenido de este archivo está sujeto a los términos de licencia de SALVO, se puede usarlo mientras se cumpla con dichos términos. En la figura 4.2 se aprecia la organización del código fuente SALVO.

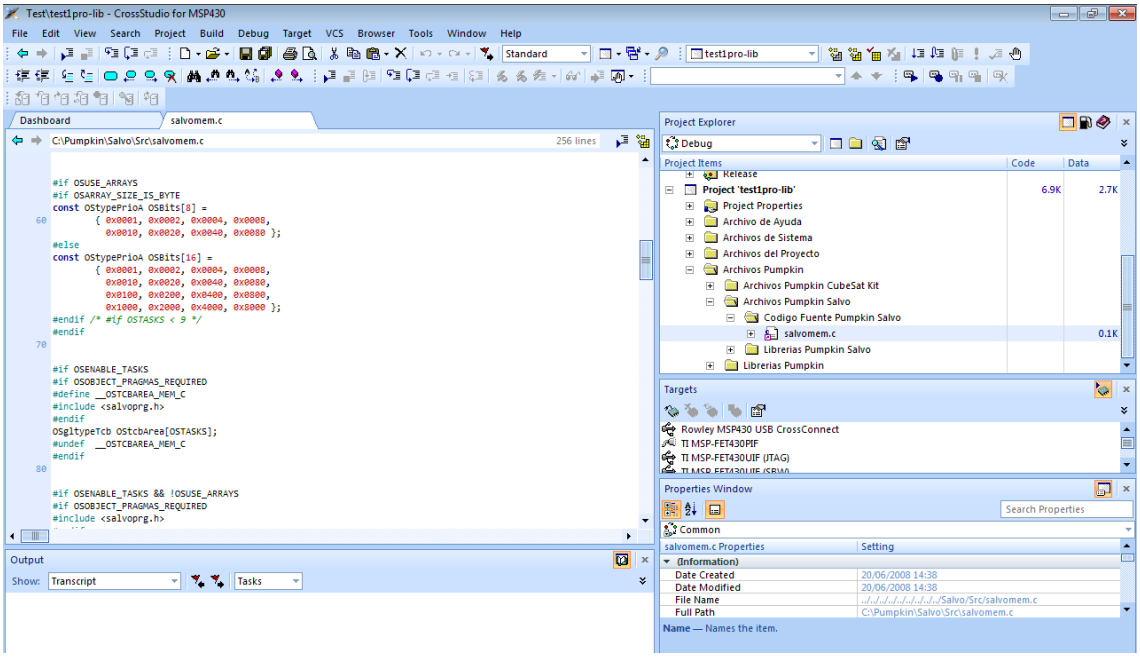

Fig. 4.2: Organización del Código Fuente SALVO en el ambiente de desarrollo

4.1.3 Código Fuente CrossWorks

Existen archivos de código fuente propios de CrossWorks que son dedicados exclusivamente para el uso del microcontrolador MSP430 del Tablero de Desarrollo del CubeSat Kit. Al momento de instalar CrossWorks es necesario descargar e instalar los archivos relacionados con el MSP430. El código fuente que se empleó para configurar el Tablero de Desarrollo se detalla a continuación:

- cmd.c: Administra el reloj del sistema del Tablero de Desarrollo del Cubesat.

- init.c: Define varias funciones básicas: a) inicializar el sistema; b) apagar el WDT; c) deshabilitar las interrupciones; d) activar en bajo las señales de control

del CubeSat Kit; e) cerrar las comunicaciones con el módem MHX2420, USB y LEDs; f) apagar el módulo MHX, USB y LEDs; g) inicializar el reloj a una frecuencia de 32.768kHz y h) establecer la configuración de la interfaz RS232 a 9600 bps, sin paridad, con 8 bits de datos y 1 de parada.

- isr.c: Define las interrupciones de servicios. ISR son las siglas en inglés de Interrupt Service Routine.

- task_onchip_temp.c: Medir la temperatura en grados centígrados del sensor de temperatura acoplado al Tablero de Desarrollo. La temperatura medida es la del microcontrolador.

En la figura 4.3 se muestra la organización del código fuente CrossWorks.

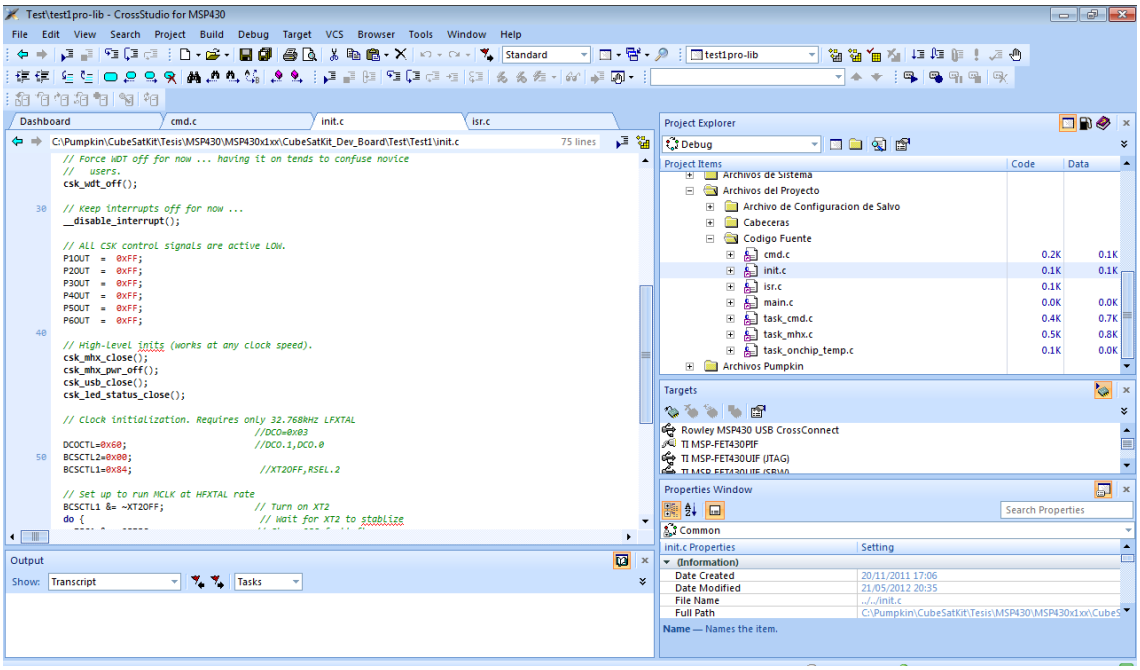

Fig. 4.3: Organización del Código Fuente CrosWorks en el ambiente de desarrollo

4.2 CABECERAS

Se denominan cabeceras a los archivos que tienen forma de código fuente que son incluidos de forma automática por el compilador. Y sirven para definir o especificar interfaces que aportan información explicativa de cómo usar los componentes declarados en el archivo. La inclusión de las cabeceras se especifica mediante el uso de pragmas al comienzo de un archivo de código fuente. Como se menciono anteriormente, los pragmas son sentencias especiales o directivas que controlan el comportamiento del compilador.

Las cabeceras por lo general contienen declaraciones de clases, subrutinas, variables entre otros identificadores. Si se desea declarar identificadores estándares en varios archivos de código fuente se puede colocar estos identificadores dentro de una sola cabecera. La cual debe ser incluida en los mismos.

4.2.1 Cabeceras SALVO

SALVO incluye una sola cabecera nombrada salvocfg.h, en la cual se definen que utilidades y herramientas del sistema operativo en tiempo real se van a usar y también se especifica la cantidad de eventos, banderas, mensajes y tareas se van a manejar en el programa. Es decir mediante está cabecera establecemos una configuración inicial del RTOS. Si bien el formato preestablecido de esta cabecera, es genérico, la forma de configurar el RTOS es independiente para cada proyecto o aplicación.

4.2.2 Cabeceras CrossWorks

Así como se maneja archivos de código fuente propio del ambiente de desarrollo CrosWorks también son necesarias las respectivas cabeceras en las cuales se definen variables, funciones y entre otros identificadores, las cabeceras utilizadas son:

Generales:

- config.h: Define las frecuencias de operación de los distintos relojes que maneja el sistema CubeSat Kit.
- CSK_cfg.h: Establece los valores de los buffer de almacenamiento que maneja el CubeSat Kit y se define la revisión del hardware del Tablero de Desarrollo, la cual es la revisión D.
- events.h: Define los eventos que el CubeSat Kit espera ocurran, estos son que se requiera y establezca comunicación a la interfaz MHX y que se detecte el ingreso de un carácter a través de la interfaz RS232 del CubeSat Kit.

En la figura 4.4 se muestra la organización de las cabeceras de CrossWorks.

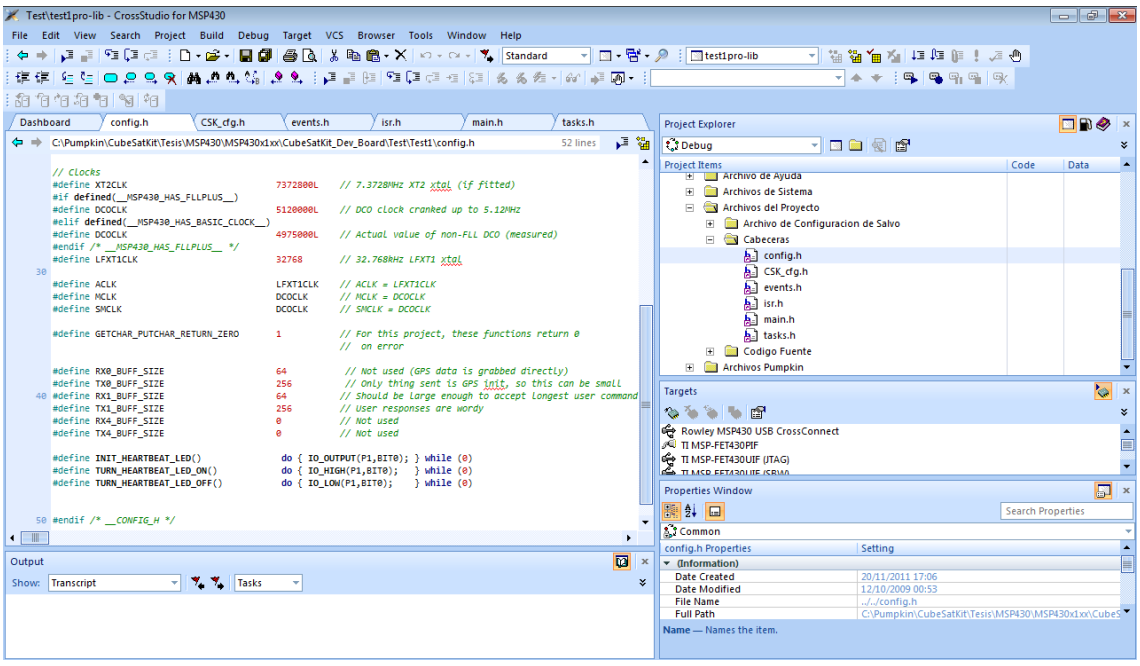

Fig. 4.4: Organización de las Cabeceras CrosWorks en el ambiente de desarrollo

Especificas para el MSP430:

Adicionalmente las cabeceras para el microcontrolador MSP430 se las obtiene al instalar los paquetes dedicados para el microcontrolador, mediante el ambiente de desarrollo CrossWorks. Sirven para especificar el modelo de microcontrolador con el que se está trabajando, las cabeceras necesarias son:

- msp430.h: Define todos los modelos posibles del microcontrolador MSP430 de Texas Instruments que soporta el ambiente de desarrollo CrossWorks. La cual debe ser incluida en todos los archivos tanto código fuente y cabeceras del proyecto desarrollado.

- msp430f1612.h: Es una cabecera exclusiva del microcontrolador MSP430 F1612 de Texas Instruments, donde se establecen los registros estándar y se definen los bits que maneja el microcontrolador.

En la figura 4.5 se muestra la organización de las cabeceras especificas para el MSP430.

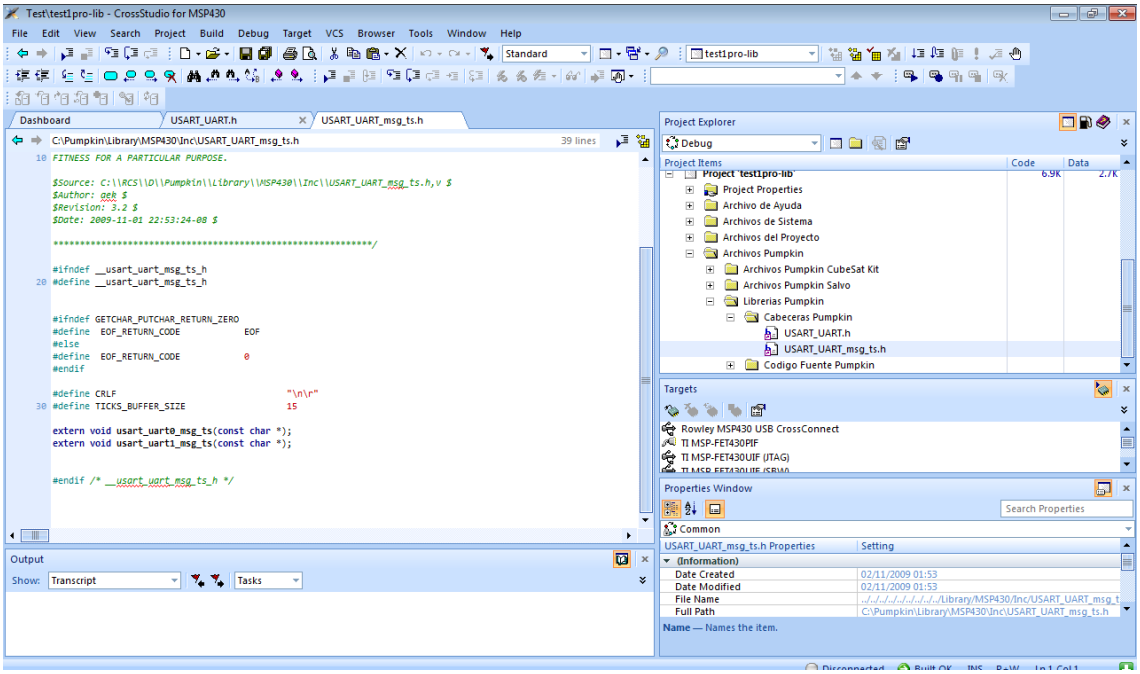

Fig. 4.5: Organización de las Cabeceras del MSP430 en el ambiente de desarrollo

4.3 LIBRERÍAS

Por otra parte es necesario importar las librerías que permiten tener acceso a los recursos del Cubesat Kit. Las librerías que son un conjunto de subprogramas propios del fabricante utilizados como herramientas para desarrollar el código de una aplicación. Las librerías contienen tanto código como datos, los cuales proporcionan servicios a programas independientes, pasando a formar parte de los mismos, de tal forma que el código y datos puedan ser compartidos y modificados de forma modular.

4.3.1 Librerías Pumpkin

Estas librerías vienen con el software del CubeSat Kit. Al instalar e integrar el software CubeSat Kit de Pumpkin se debe cargar el paquete de librerías de Pumpkin. Estas librerías se encuentran escritas en código fuente y además como cabeceras.

Las librerías en formato de código fuente que se importan son:

- usart_uart0.c: Define tareas que configuran, habilitan y deshabilitan el sistema de comunicaciones serial UART, que son las siglas de *Universal Asynchronous Receiver-Transmitter,* el cual controla los puertos y dispositivos seriales. El UART0 corresponde a la interfaz serial RS232 del CubeSat Kit.

- usart_uart0_msg_ts.c: Define una función que permite enviar mensajes al UART0.

- usart_uart1.c: Similar al archivo usart_uart0.c, pero para controlar el UART1. El UART1 corresponde a la interfaz que comunica el CubeSat Kit con el módulo MHX2420.

- usart_uart1_msg_ts.c: Define una función que permite enviar mensajes al UART1.

Mientras que las librerías que se usan en forma de cabecera son:

- USART_UART.h: Declara el tipo de microcontrolador que se está empleando y además se establecen los parámetros de inicialización para las UART, entre ellos los parámetros con los que pueden funcionar las interfaces seriales del CubeSat Kit (baudios, bits de datos, etc).

- USART_UART_msg_ts.h: Declara variables para el envío de mensajes hacia el UART0 y UART1.

En la figura 4.6 se muestra la organización de las librerías de Pumpkin.

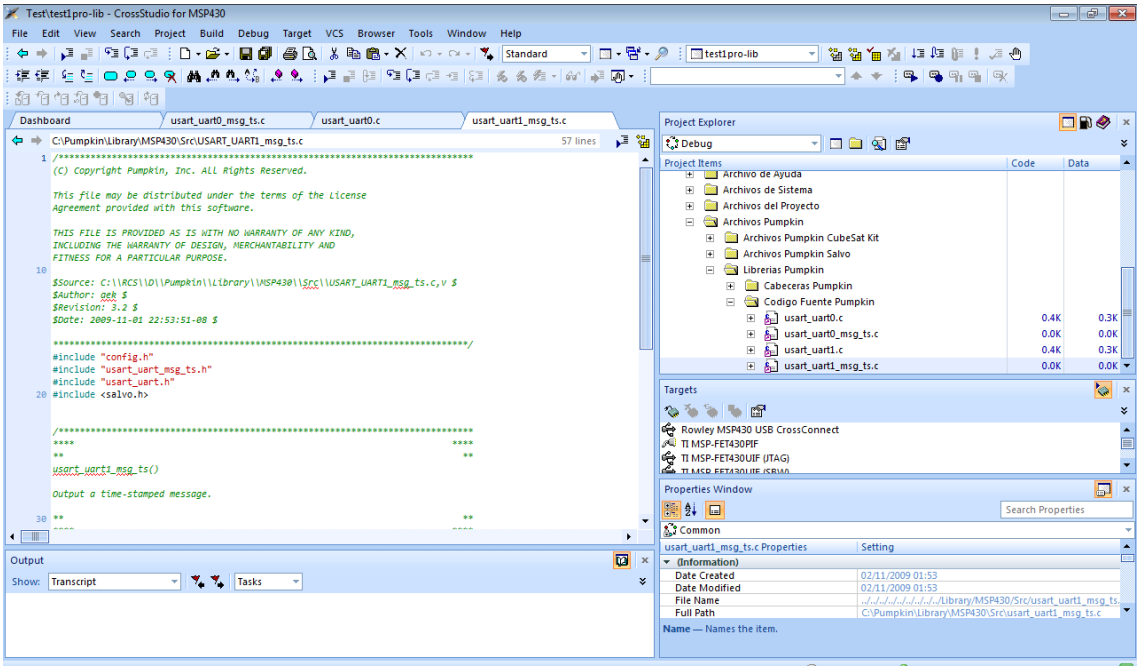

Fig. 4.6: Organización de las Librerías Pumpkin en el ambiente de desarrollo

4.3.2 Librerías SALVO

Para cargar las librerías del sistema operativo en tiempo real SALVO, se debe seguir la nomenclatura de la figura 3.10. Así la librería a importar según los requerimientos de la aplicación es libsalvolra430it.hza, donde:

- libsalvo: Indica que es una librería de SALVO.
- l: Define que el tipo de librería es estándar, que corresponde a las librerías que son incluidas en la licencia comprada e incluida en el CubeSat Kit.
- ra: Específica para que tipo de compilador están diseñadas las librerías, en este caso trabajan con el compilador de lenguaje C CrossWorks para el microcontrolador MSP430.
- 430: Determina el tipo de arquitectura maneja el microcontrolador, en este caso el microcontrolador tiene la arquitectura para la familia del microcontrolador MSP430.
- i: Señala si las librerías incluyen la información de depuración o no, en caso de que incluyan dicha información se usa la letra i.
- t: Determina la configuración de las librerías que se van a importar. La letra t corresponde a la configuración que permita usar multitareas con retardos y eventos, y las tareas soportan *timeouts*.
- Finalmente .hza corresponde a la extensión que manejan todas las librerías de SALVO para ser reconocidas como tales.

En la figura 4.7 se muestra la organización de las librerías SALVO en el ambiente de desarrollo.

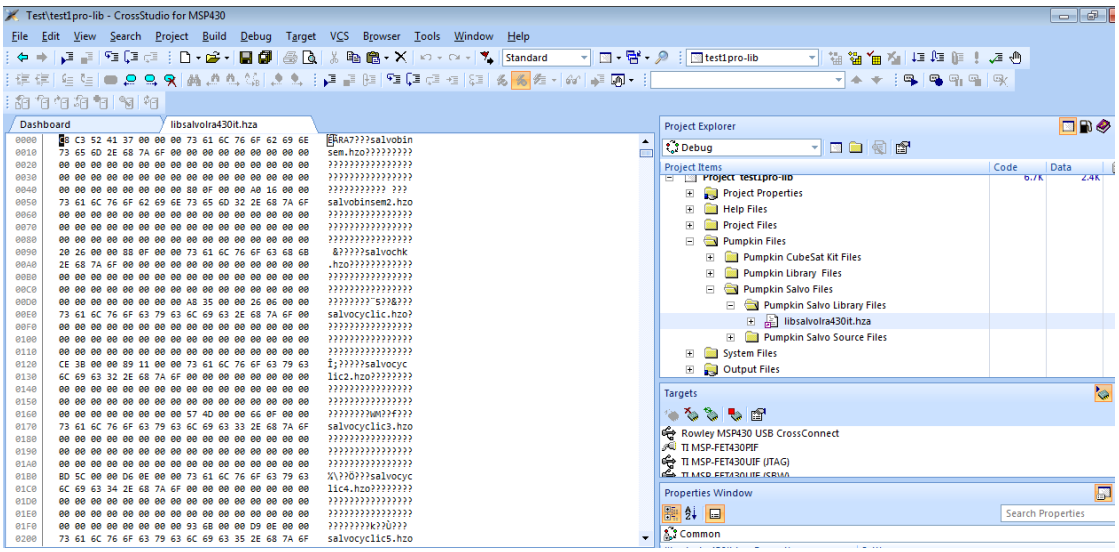

Fig. 4.7: Organización de las Librerías SALVO en el ambiente de desarrollo

CAPITULO 5

5. CONFIGURACIÓN E INTEGRACIÓN DE HARDWARE Y SOFTWARE DEL SISTEMA

El sistema de pruebas sobre el cual se desarrolló el presente proyecto se detalla en la figura 5.1.

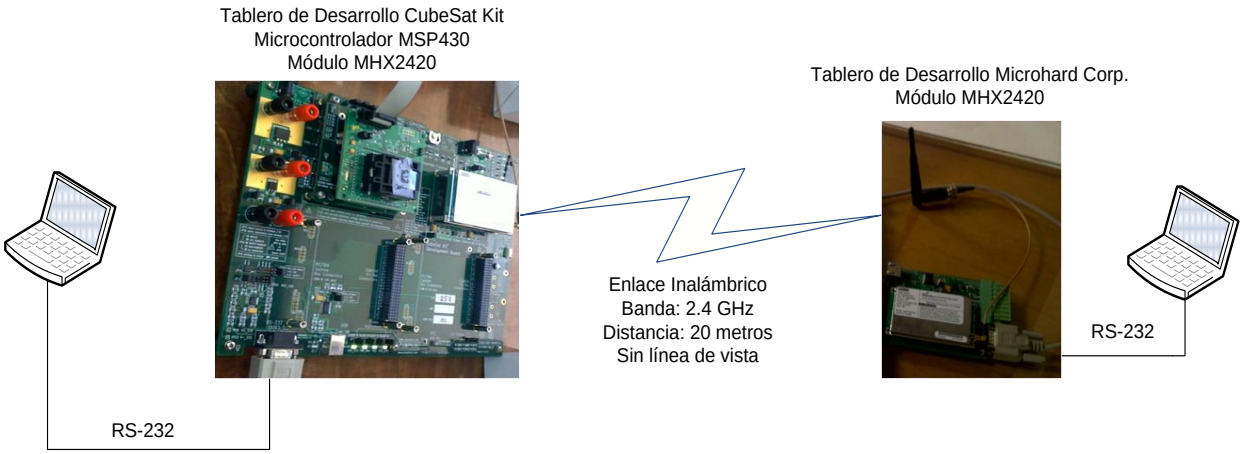

Fig. 5.1: Acceso zona privada de descargas Pumpkin Inc.

En el cual están involucrados el tablero de desarrollo del CubeSat Kit con en el microcontrolador MSP430 y el módulo de comunicaciones inalámbricas MHX2420 y el tablero de desarrollo de Microhard Corp. al cual se integra el módem MHX2420. Dos computadoras conectadas a las interfaces de ambos tableros de desarrollo respectivamente, la que se encuentra conectada al tablero de desarrollo del CubeSat Kit permite observar los procesos que se están llevando a cabo del sistema en general, y la

que se encuentra conectada al tablero de desarrollo de Microhard Corp. permite observar los datos transmitidos mediante el enlace inalámbrico establecido entre los módulos MHX2420.

Para realizar este proyecto fue necesario llevar a cabo una serie de procesos que permitieron configurar e integrar el sistema y establecer un enlace de radio punto a punto en la banda de 2.4 GHz entre los módulos MHX2420 del CubeSat Kit y del Tablero de Desarrollo de Microhard Corporation.

Los procedimientos que se llevaron a cabo se explican en el transcurso de este capítulo y se enumeran a continuación:

- Descarga del software necesario.
- Instalación del software y drivers.
- Verificación de integridad del hardware del sistema.
- Configuración del microcontrolador MSP430 de Texas Instruments mediante el ambiente de desarrollo CrossWorks.
- Configuración del Sistema Operativo en Tiempo Real Salvo.
- Configuración de los Módulos MHX2420 vía comandos AT.
- Integración del Hardware del Sistema.
- Transmisión de información.

Para establecer el enlace de comunicación inalámbrico se utilizaron una pareja de radios MHX2420. A fin de configurarlos, se usaron dos métodos:

- Uso del ambiente de desarrollo CrossWorks configurando el microcontrolador MSP430 del Tablero de Desarrollo del CubeSat Kit.
- Uso de comandos AT mediante el Tablero de Desarrollo de Microhard Corp.

5.1 CONSIDERACIONES DE SOFTWARE Y HARWARE

Para realizar la configuración del CubeSat Kit se debe considerar las necesidades en cuanto a hardware y software que se requieren.

5.1.1 Consideraciones de Software

Es muy importante considerar el Anexo A y B en los que se detalla la información de descarga e instalación del software respectivamente. Son de suma importancia puesto que ningún manual indica cómo hacerlo. La realización de este proyecto permitió establecer todos los pasos que se deben seguir para realizar los procedimientos de descarga e instalación del software de forma correcta.

Una vez instalado todo el software correspondiente al Cubesat Kit se puede realizar la configuración y programación del mismo, mediante la integración de librerías, código fuente, cabeceras, tareas y funciones de los programas instalados.

5.1.2 Consideraciones de Hardware

Las características mínimas de la computadora en la que se ejecutan los programas deben ser:

- Sistema Operativo Windows XP y versiones posteriores. (Puede ser de 64 bits o 32 bits)
- Procesador Intel Core 2 Duo 2.00 GHz.
- Memoria RAM: Mínimo 1Gb

- 80MB de espacio libre en el disco, y 40MB extra para guardar los proyectos que se realicen.
- Mínimo un puerto serial DB9 o puerto USB para conexión al Tablero de Desarrollo del CubeSat Kit y al Tablero de Desarrollo de Microhard Corp. De solo contar con el puerto USB es necesario un cable conversor de DB9 a USB.

En cuanto al Tablero de Desarrollo del CubeSat Kit se debe verificar que los Jumpers distribuidos en el mismo tengan la configuración detallada en la tabla 5.1

Tabla 5.1: Configuración por defecto de los Jumpers del Tablero de Desarrollo

Jumper	Configuración	Verificación
JP1	OFF	OK
JP2	ON	OK
JP3	ON	OK
JP4	OFF	OK
JP5	ON	OK
JP6	OFF	OK
JP7	ON	OK

Jumper	Configuración	Verificación
JP8	ON	OK
JP9	ON	OK
JP10	ON	OK
JP11	OFF	OK
JP12	OFF	OK
JP13	OFF	OK
JP14	2-3	OK

Para verificar el estado y correcta alimentación del Tablero de Desarrollo se debe conectar la fuente de alimentación de +5V al conector J1 y se debe medir las siguientes tensiones:

Tabla 5.2: Tensiones de Funcionamiento del Tablero de Desarrollo

Señal	Ubicación	Valor	Valor Medido
+5V	TP9	+5V	+4.8V

Señal	Ubicación	Valor	Valor Medido
VCC	TP12	+3.3V	+3.2V
VCC_MCU	TP20	+3.3V	+3.2V
VCC_232	TP21	+3.3V	+3.3V
V+_232	TP19	>+5V	+5.1V
V-_232	TP22	<-5V	-5.3V
+5V_SW	TP10	0V	0V
-RST/NMI	TP8	+3.3V	+3.4V

Todos los cables y accesorios (fuentes de poder, conectores, etc.) necesarios para configuración y demás aplicaciones del hardware adquirido con el CubeSat Kit y con los módems MHX2420, están incluidos en los mismos.

Dispositivo JTAG

Para comunicar la computadora con el Tablero de Desarrollo del CubeSat Kit específicamente con el microcontrolador MSP430, se utiliza el conversor JTAG USB-Serial a fin de programar y depurar la configuración realizada en el ambiente de desarrollo CrossWorks. Dicho dispositivo es propio de Texas Instruments y está incluido dentro de los elementos del CubeSat Kit. El modelo del equipo es MSP-

FET430UIF, el cual es una herramienta de emulación que permite iniciar el desarrollo de aplicaciones en el microcontrolador MSP430. El módulo requiere únicamente entre 1.8 a 3.6 V y 100mA por lo tanto se alimenta mediante el puerto USB de la computadora. En la figura 5.2 se aprecia el dispositivo JTAG conectado tanto al Tablero de Desarrollo como a una computadora con el ambiente de desarrollo CrossWorks ejecutándose. El JTAG posee dos LEDs indicadores; uno en rojo, que indica que el dispositivo está conectado y que se estableció comunicación al microcontrolador y se puede enviar la configuración, y el otro en verde, que indica que el dispositivo está encendido.

Fig. 5.2: Dispositivo JTAG conectado al Tablero de Desarrollo y a la computadora mediante el Ambiente de Desarrollo CrossWorks

5.2 CONFIGURACIÓN DEL MICROCONTROLADOR TI-MSP430 MEDIANTE CROSSWORS

A continuación se detalla la configuración del microcontrolador MSP430 usando el ambiente de desarrollo CrossWorks versión 2.0.8.

Para configurar el microcontrolador MSP430 se siguieron los siguientes pasos:

- Ejecutar el ambiente de desarrollo CrossWorks.
- Importar las librerías y archivos del sistema de referencia.
- Configurar el RTOS SALVO mediante la cabecera salvocfg.h.
- Desarrollo del código para la aplicación del sistema a configurar.

Todo lo que este a continuación de los caracteres // son comentarios añadidos para guiar que se está haciendo en cada segmento de código.

5.2.1 Código Fuente

5.2.1.1. Archivo main.c

Es el archivo de código fuente principal de la configuración del microcontrolador MSP430, en el cual se inicializa al dispositivo y sus periféricos, se carga el sistema operativo en tiempo real y se crean tareas, entre otros. A continuación se explica brevemente sus principales funcionalidades:

- Incluye las cabeceras necesarias para la configuración del MSP430, tanto las que se crearon para el proyecto como las del sistema de referencia. Además se incluye la cabecera del RTOS SALVO y la creación de 2 variables globales.

```
#include "events.h"

#include "init.h"

#include "main.h"

#include "task_cmd.h"

#include "task_mhx.h"

#include "task_onchip_temp.h"

#include "tasks.h"

// Cabeceras Pumpkin Salvo

#include "salvo.h"

csk_status_t csk_status;

char strTmp[80];
```

- Inicializa la función main que es de tipo integer. Se llama a la función init() que inicializa el sistema del CubeSat Kit y a la función OSInit(), la cual inicializa el sistema operativo en tiempo real SALVO.

```
int main(int argc, char *argv[]) {

//  Inicializacion.

init();

// Inicializando Salvo RTOS.

OSInit();
```

- Crea las tareas que tendrá el proyecto en el RTOS SALVO. Dándoles sus respectivas prioridades y especificando el puntero correspondiente a cada tarea.

  //Creando tareas

  - La tarea task_cmd_do se refiere a la función creada para verificar el estatus de la aplicación mediante la interfaz RS232.

    OSCreateTask(task_cmd_do, TASK_CMD_DO_P, 1);

  - La tarea task_mhx_talk se refiere a la función creada para establecer comunicación con el módulo MHX2420.

    OSCreateTask(task_mhx_talk, TASK_MHX_TALK_P, 7);

  - La tarea task_onchip_temp se refiere a la función usada para medir la temperatura del microcontrolador MSP430.

    OSCreateTask(task_onchip_temp, TASK_ONCHIP_TEMP_P, 11);

  - Crea eventos por los que el CuBeSat Kit espera que sucedan para realizar determinadas actividades. Se debe especificar el puntero correspondiente a cada evento y su prioridad.

    // Creación de eventos.

  - Mediante la creación del semáforo binario RSRC_ USB_MHX_IF_P se especifica el evento que permite establecer comunicación con el MHX2420.

    OSCreateBinSem(RSRC_USB_MHX_IF_P, 1);

  - Creando el semáforo SEM_CMD_CHAR_P se determina que el CubeSat Kit espera que se digite un carácter (teniendo comunicación a la interfaz RS232 del Tablero de Desarrollo) para realizar determinada tarea.

    OSCreateSem(SEM_CMD_CHAR_P, 0);

- Habilita las interrupciones en el sistema.

```
// Habilitar interrupcioness.

__enable_interrupt();
```

- Crea un lazo infinito para poder tener la opción de multitareas mediante el comando del RTOS SALVO OSSched(), el cual ejecuta la tarea con mayor prioridad en un principio, al estar en un lazo infinito permite que las demás tareas se lleven a cabo, esto es lo que se denomina multitareas.

```
// Multitareas

while (1) {

  OSSched();

} /* while */

} /* main() */
```

5.2.1.2. Archivo task_cmd.c

La tarea task_cmd.c permite establecer comunicación con la interfaz RS232 del Tablero de Desarrollo del CubeSat Kit. A su vez admite el ingreso de caracteres para determinadas funciones.

- Incluye las cabeceras que el archivo de código fuente maneja.

```
#include "main.h"

#include "cmd.h"

#include "events.h"
```

```
#include "task_cmd.h"

#include "task_onchip_temp.h"

#include "tasks.h"

#include <ctype.h>

// Cabeceras Pumpkin CubeSat Kit

#include "csk_power.h"

#include "csk_uart.h"

#include "csk_wdt.h"

// Cabeceras Pumpkin Salvo

#include "salvo.h"
```

- Crea la función o tarea tipo void cmd_explain que despliega el menú de posibles opciones de caracteres que pueden ser ingresados y su respectiva función. El código se muestra a continuación:

```
static

void cmd_explain(void) {

user_debug_msg(STR_CMD_EXPLAIN "h|?: Ayuda ");

user_debug_msg(STR_CMD_EXPLAIN "r:   reinicio de la aplicacion");

user_debug_msg(STR_CMD_EXPLAIN "t:   despliega temperatura del microcontrolador");

user_debug_msg(STR_CMD_EXPLAIN "v:   despliega versión del software");

user_debug_msg(STR_CMD_EXPLAIN "z:   pone el sistema en modo sleep");

} /* cmd_explain() */
```

- Crea la función tipo void task_cmd_do, que se ejecuta según el carácter ingresado al Tablero de Desarrollo del CubeSat Kit.

```
void task_cmd_do(void) {

unsigned char cmd;

user_debug_msg(STR_TASK_CMD_DO "Inicializando.");

OS_Delay(50);

csk_status.MCLKOutEnabled = 0;

OS_Delay(50);
```

Dentro de la función task_cmd_do se ejecutan las siguientes tareas:

- Llama a la función cmd_explain().

```
cmd_explain();

while (1) {
```

- Despliegue de la configuración del puerto RS232.

```
user_debug_msg(STR_TASK_CMD_DO "Control via RS-232 en " STR_BAUD_RATE ",N,8,1.");

OS_Delay(50);
```

- El programa espera constantemente el evento de ingreso de un carácter mediante el teclado de una computadora conectada a la interfaz RS232 del Tablero de Desarrollo.

```
OS_WaitSem(SEM_CMD_CHAR_P, OSNO_TIMEOUT);

  if ((cmd=csk_uart0_getchar())) {
```

- Dependiendo del carácter digitado se realizan las siguientes actividades.

```
switch (tolower(cmd)) {

// Ayuda.
```

- En caso de que se presione la letra h o el símbolo ?, se despliega el menú de ayuda llamando a la función cmd_explain().

```
case 'h':

case '?':

    cmd_explain();

    break;
```

- En el caso que se digite la letra r se realice el reseteo del Tablero de Desarrollo mediante el WDT.

```
// Reset.

case 'r':

    user_debug_msg(STR_TASK_CMD_DO "r: Reset en 1 s.");

    OS_Delay(100);

    csk_wdt_force();

    break;
```

- Si se presiona la letra t, se despliega la temperatura del microcontrolador llamando a la tarea onchip_temp_deg_c().

```
// Lectura del sensor de temperatura del microcontrolador

case 't':

    sprintf(strTmp, STR_TASK_CMD_DO "t: Temperatura medida en el microcontrolador %dC.", onchip_temp_deg_c());
```

```
user_debug_msg(strTmp);

break;
```

- Para desplegar la versión del software que se está usando se digita la tecla v.

```
// Version del software.

case 'v':

    user_debug_msg(STR_TASK_CMD_DO "v: Versión CrossWorks: 2.0.8");

    user_debug_msg(STR_TASK_CMD_DO "v: Versión SALVO: Pro MSP430 4.2.2 rc5");

    user_debug_msg(STR_TASK_CMD_DO "v: Versión Software CubeSat Kit: 1.2.2-rc2");

    break;
```

- Para apagar el sistema se debe presionar la tecla z. Todas las tareas terminan excepto task_cmd_do.

```
// Apagado del sistema.

case 'z':

    user_debug_msg(STR_TASK_CMD_DO "z: Apagado complete en 2 s.");

// Termina todas las tareas excepto task_cmd_d.

    for (cmd=2; cmd<=OSTASKS; cmd++) {

      OSStopTask(OSTCBP(cmd));

    }

    OS_Delay(100);
```

- El sistema muestra un mensaje indicando que el sistema se apago.

```
// Mensaje Final.

user_debug_msg(STR_TASK_CMD_DO "z: El sistema esta apagado.");

OS_Delay(100);
```

- Se ejecuta el comando propio del CubeSat Kit para apagar el sistema.

```
// Apagado completo del sistema

csk_power_shutdown();

break;
```

- En el caso que se digite otra letra o carácter a los indicados anteriormente se despliega un mensaje en el cual se avisa al usuario que el comando no es válido.

```
default:

    sprintf(strTmp, STR_TASK_CMD_DO "Comando desconocido: '%c' (0x%X)", cmd, cmd);

user_debug_msg(strTmp);

break;

} /* switch */

} /* if */

} /* while */

} /* task_cmd_do() */
```

5.2.1.3. Archivo task_mhx.c

Esta tarea permite establecer comunicaciones con el módem MHX2420 acoplado al CubeSat Kit y configurarlo:

- Se incluyen las cabeceras que el archivo de código fuente maneja.

```
#include "main.h"
#include "events.h"
#include "task_onchip_temp.h"
#include "task_mhx.h"
// Cabeceras Pumpkin CubeSat Kit
#include "csk_mhx.h"
#include "csk_rand.h"
#include "csk_uart.h"
// Cabeceras Pumpkin Salvo
#include "salvo.h"
```

- Inicio de la tarea, se inicializa la función tipo void.

```
void task_mhx_talk(void) {
```

- Se envía mensajes a la interfaz de debug, la cual es la interfaz RS232 del Tablero de Desarrollo del CubeSat Kit, para indicar que procedimientos se están llevando a cabo de manera informativa. Esto se hace mediante una de las funciones propias del CubeSat Kit que incluye en sus librerías. Dicha función se ejecuta de la siguiente forma: user_debug_msg(STR_TASK_MHX_TALK "Mensaje"); y de forma automática el mensaje es enviado a la interfaz RS232 del Tablero de Desarrollo del CubeSat Kit. Estos mensajes se pueden visualizar

conectando una computadora a la interfaz RS232 mediante un cable serial directo y usando un terminal ASCII.

```
user_debug_msg(STR_TASK_MHX_TALK "Inicializando Tarea MHX.");

// Empieza despues de 2.5 segundos.

OS_Delay(250);

while (1)  {
```

- Se ingresa un retardo de 5 segundo incluido un tiempo aleatorio. Se debe evitar ingresar un retardo en 0 puesto que esto terminaría con la tarea. Para llamar a retardos de tiempos se usa la función OS_Delay(TiempodeRetardo); por ejemplo si se desea insertar un retardo de 1 segundo se lo hace de la siguiente forma: OS_Delay(100). Todas las funciones que comiencen con OS son propias del sistema operativo en tiempo real (RTOS) SALVO.

```
// Espera 5s + un tiempo aleatorio. Evitar el

//  OS_Delay(0) porque eso detendría esta tarea.

OS_Delay(250);

OS_Delay(250);

OS_Delay(1 + ((csk_rand()>>9)& 0x007F));
```

- Se verifica si la interfaz MHX RF[2] está siendo usada mediante un semáforo binario propio del RTOS SALVO.

```
// Procede si la interfaz MHX RF no está siendo usada.

OS_WaitBinSem(RSRC_USB_MHX_IF_P, OSNO_TIMEOUT);

user_debug_msg(STR_TASK_MHX_TALK "Habilitando MHX RF.");
```

[2] Es la interfaz de comunicación entre el CubeSat Kit y el MHX2420

- Se habilita la comunicación hacia el módulo MHX2420 y se enciende el mismo. Mediante las funciones csk_mhx_open() y csk_mhx_pwr_on() respectivamente. Las funciones que empiezan con csk son propias de las librerías del CubeSat Kit. Y se coloca el valor 1 a la variable csk_status.mhx_connected, la cual sirve de indicador que se estableció comunicación con el módulo MHX2420.

  ```
  // Acceso al MHX.

  csk_mhx_open();

  csk_mhx_pwr_on();

  csk_status.mhx_connected = 1;
  ```

- Se envía un mensaje de aviso a la interfaz RS232 del Tablero de Desarrollo indicando que se estableció comunicación y se encendió el módulo MHX2420. Se espera 1 segundo para que encienda el módulo y se notifica mediante el respectivo mensaje.

  ```
  // Envio de un mensaje simple al puerto de debug.

  user_debug_msg(STR_TASK_MHX_TALK "MHX_PWR ENCENDIDO.");

  // Espera 1s para que el modem encienda.

  user_debug_msg(STR_TASK_MHX_TALK "Esperando que el modem encienda.");

  OS_Delay(100);
  ```

- Mediante el envío del comando mhx al módulo MHX2420 se lo coloca en modo de comandos, y espera un segundo por la respuesta del módem. Mediante el la línea csk_uart1_puts(“Comando”) se envía un comando al MHX2420.

  ```
  // Pone el modem en modo de comandos  mediante el envío del comando mhx y espera por respuesta, la respuesta que debe aparecer debe ser OK.
  ```

```
user_debug_msg(STR_TASK_MHX_TALK "\"mhx\": Modem en modo de comandos.");

csk_uart1_puts("mhx");

OS_Delay(100);
```

- Se envía el comando AT&F7&K0 al módem MHX2420 de tal forma que se lo configura como modo Esclavo. Mediante el comando AT&F7 se configura automáticamente los registros para que el equipo trabaje en dicho modo. Los demás registros tales como potencia de salida y tasas de transmisión, que no definen el tipo de modo de operación del módem son configurados con sus valores por defecto. Adicionalmente se añade &K0 al comando para deshabilitar el handshaking de hardware.

```
// Configura el modem en modo Esclavo y espera por respuesta.

user_debug_msg(STR_TASK_MHX_TALK "\"AT&F7&K0\": Modo esclavo, sin handshaking y esperando respuesta");

csk_uart1_puts("AT&F7&K0");

OS_Delay(50);
```

- Una vez configurado el módem conectado directamente al CubeSat Kit como Esclavo, se sale del modo de comandos enviando el texto ATA al MHX2420. Posteriormente, se configura al módem en modo de transmisión de datos para que pueda establecer la comunicación con el módulo remoto y enviar la información generada en el CubeSat Kit.

```
// Entra en modo de datos.

user_debug_msg(STR_TASK_MHX_TALK "\"ATA\": Entrando al modo de datos.");

csk_uart1_puts("ATA");
```

```
OS_Delay(150);
```

- Asumiendo que se establece el enlace entre los módulos MHX2420, se realiza el envío de datos en forma de texto plano, se envían mensajes informativos, y los datos de temperatura del CubeSat Kit medidos por la tarea propia del CubeSat Kit que realiza la medición de temperatura en grados centígrados del microcontrolador, la cual es onchip_temp_deg_c(). Para enviar datos al remoto se usa la línea sprintf(strTmp, "Texto a enviar"). Los datos se envían al módem MHX2420 para que el mismo los envíe mediante el enlace inalámbrico al módulo remoto y a su vez los datos son enviados a la interfaz RS-232 del Tablero de Desarrollo del CubeSat Kit con el fin de constatar la integridad de los datos recibidos en el equipo remoto, es decir comprobar que los datos enviados mediante el enlace inalámbrico sean los mismos que el microcontrolador genera.

```
// Envio de datos.

user_debug_msg(STR_TASK_MHX_TALK "Enviando datos.");

csk_uart1_puts(STR_CRLF);

sprintf(strTmp, "CubeSat Kit ESCUELA POLITECNICA DEL EJERCITO ..." );

data_debug_msg(strTmp);

user_debug_msg(STR_TASK_MHX_TALK "CubeSat Kit ESCUELA POLITECNICA DEL EJERCITO ...");

OS_Delay(100);

sprintf(strTmp, "Autor: Juan Fernando Balarezo" );

data_debug_msg(strTmp);

user_debug_msg(STR_TASK_MHX_TALK "Autor: Juan Fernando Balarezo");

OS_Delay(100);
```

```
sprintf(strTmp, "Directores:" );

data_debug_msg(strTmp);

user_debug_msg(STR_TASK_MHX_TALK "Directores:");

OS_Delay(100);

sprintf(strTmp, "Ing. Darío Duque" );

data_debug_msg(strTmp);

user_debug_msg(STR_TASK_MHX_TALK "Ing. Dario Duque");

OS_Delay(100);

sprintf(strTmp, "Ing. Vanessa Vargas" );

data_debug_msg(strTmp);

user_debug_msg(STR_TASK_MHX_TALK "Ing. Vanessa Vargas");

OS_Delay(100);

sprintf(strTmp, "TEMPERATURA MEDIDA EN EL CHIP:  %dC." , onchip_temp_deg_c());

data_debug_msg(strTmp);

user_debug_msg(strTmp);

OS_Delay(100);
```

- Después de haber enviado el texto al remoto, se vuelve al modo de comandos mediante el envío del comando +++ al MHX2420.

```
// Dejando el modo de datos despues de enviar el comando +++ al MHX.

user_debug_msg(STR_TASK_MHX_TALK "\"+++\": dejando el modo de datos.");
```

```
OS_Delay(100);

csk_uart1_puts("+++");
```

- Se apaga el segmento de transmisión del módem MHX2420.

```
// Finalizando tarea.

OS_Delay(100);

user_debug_msg(STR_TASK_MHX_TALK "Deshabilitando MHX RF.");

csk_mhx_pwr_off();

csk_mhx_close();

user_debug_msg(STR_TASK_MHX_TALK "MHX_PWR  OFF.");

csk_status.mhx_connected = 0;
```

- Finalmente, se deshabilita la comunicación entre el CubeSat Kit y su módulo MHX2420 acoplado y finaliza la tarea.

```
// Liberando la interfaz MHX RF.

OSSignalBinSem(RSRC_USB_MHX_IF_P);

user_debug_msg(STR_TASK_MHX_TALK "MHX RF LIBERADA.");

} /* while */

} /* task_mhx_talk() */
```

5.2.2 Cabeceras

5.2.2.1. Cabecera events.h

En esta cabecera se declaran los eventos que se usarán, a los que se debe asignar un nombre respectivamente. Sin embargo, existirá un puntero general de eventos llamado OSECBP(). Los eventos definidos son:

- RSRC_USB_MHX_IF_P: Cuando se requiera transmitir datos o establecer comunicación al módulo MHX2420.

  #define RSRC_USB_MHX_IF_P OSECBP(1)

- SEM_CMD_CHAR_P: Permite reconocer cuando un carácter es digitado mediante un computador conectado al puerto RS232 del Tablero de Desarrollo del CubeSat Kit.

  #define SEM_CMD_CHAR_P OSECBP(2)

5.2.2.2. Cabecera tasks.h

La cabecera task.h permite definir las tareas que van a ser creadas en el archivo de código fuente main.c. Las mismas serán administradas usando el RTOS SALVO y utilizan el puntero OSTCBP(). Son las siguientes:

- TASK_CMD_DO_P: Establece una tarea que permite el ingreso de caracteres a la interfaz serial RS232 del Tablero de Desarrollo del CubeSat Kit.

  #define TASK_CMD_DO_P OSTCBP(1)

- TASK_ONCHIP_TEMP_P: Declara la tarea que lee la temperatura registrada por el sensor que tiene acoplado el Tablero de Desarrollo.

  #define TASK_ONCHIP_TEMP_P OSTCBP(2)

- TASK_MHX_TALK_P: Establece la tarea que lleva a cabo la comunicación con el módulo MHX2420.

  #define TASK_MHX_TALK_P OSTCBP(3)

5.2.2.3. Cabecera main.h

Mediante la cabecera main.h se realiza lo siguiente:

- Declaración de símbolos a manera de constantes que pueden ser usados en cualquier parte del programa, siempre y cuando se incluya esta cabecera.

  #ifndef __main_h

  #define __main_h

  // Simbolos.

  #define STR_APP_NAME "Transmisión de datos mediante un enlace inalambrico"

  #define STR_VERSION " Elaborado el " __DATE__ " at " __TIME__

  #define STR_WARNING "Precaución: Use el comando 'z' con precaución "

  #define STR_BAUD_RATE "9600"

  #define STR_1TAB "\t"

```
#define STR_2TABS            "\t\t"

#define STR_CRLF             "\r\n"
```

- Se define la forma como serán llamadas las funciones que permiten enviar datos a la interfaz RS232 y al módulo MHX2420.

```
// Macros

#define user_debug_msg(x)     csk_uart0_msg_ts(x)

#define data_debug_msg(x)     csk_uart1_msg_ts(x)
```

- Se establece como se podrá llamar a un lazo infinito.

```
#define LOOP_HERE()           do { ; } while (1)

#endif /* __main_h */
```

5.3 CONFIGURACIÓN DEL RTOS USANDO SALVO

5.3.1 Cabeceras

Mediante la cabecera salvocfg.h se configura los recursos del sistema operativo en tiempo real que se van a usar.

- Se establece que tipo de librerías se usará, la cual es la estándar adquirida con la compra del software SALVO incluido en el paquete CubeSat Kit.

```
#define OSUSE_LIBRARY              TRUE

#define OSLIBRARY_TYPE             OSL
```

- Se define que el proyecto soportará multitareas con retardos y eventos. Las tareas pueden esperar la ocurrencia de eventos mediante *timeouts*.

  #define OSLIBRARY_CONFIG OST

- El número de eventos. En este caso son 2.

  #define OSEVENTS 2

- El uso de banderas. No se usan banderas ni mensajes en este caso.

  #define OSEVENT_FLAGS 0

  #define OSMESSAGE_QUEUES 0

- El número de tareas. En este proyecto se usaron 3.

  #define OSTASKS 3

5.3.2 Librerías

Las librerías utilizadas se encuentran detalladas en el capítulo 4.3.2.

Una vez realizada la configuración del módulo y enviada a través del comando AT&F7&K0. El módulo queda programado de la siguiente forma:

Tabla. 5.3: Configuración y Registros modo Esclavo.

Descripción	Registro	Valor
Modo de operación	S101=2	Modo Esclavo
Tasa de Transmisión Inalámbrica	S103=2	172800 bps
Dirección de Unidad	S105=2	2
Tiempo de Salto de Frecuencias	S109=9	20 ms
Tamaño Máximo del Paquete	S112=255	255 bytes
Dirección de Destino	S140=1	1 (Maestro)
Modo de Transmisión Serial	S142=0	Mediante la interfaz RS232
Brillo de los LEDS(%)	S149=100	100%

Descripción	Registro	Valor
Velocidad de Transmisión de le Interfaz Serial	S102=7	9600 bps
Dirección de Red	S104=1234567890	1234567890
Potencia de Transmisión(dBm)	S108=30	30 dBm
Formato de los Datos (Interfaz RS232)	S110=1	8N1
Número de Retransmisión de Datos	S113=5	5
Tipo de Red	S133=1	Punto a Punto
Handshaking	K0	Deshabilita Handshaking de Hardware

5.4 CONFIGURACIÓN DE LOS TRANCEIVERS MHX VIA COMANDO AT

Usando el tablero de desarrollo propio de Michrohard Corporation, se puede configurar el módulo MHX2420 mediante el conector RS232 usando un terminal ASCII o usando el conector SERIAL DIAG (Puerto RJ45) con el software de diagnósticos del sistema MHS.

Antes de realizar la configuración es necesario conectar el cable serial directo entre el puerto RS-232 del tablero de desarrollo y un puerto serial del computador con el que se esté trabajando[3]. Una vez conectado adecuadamente el tablero de desarrollo al computador se abre el terminal ASCII que se está usando[4], y se lo configura de la siguiente manera:

- Velocidad: 9600 baudios.
- Bits de Datos: 8.
- Bit de Parada: 1.
- Paridad: Ninguna.
- Handshaking: No.

[3] En caso de no poseer puerto serial el computador utilice un cnversor USB-Serial.

[4] Putty, Hyper Terminal, entre otros.

El procedimiento de conexión es:

1. Posteriormente se inicia la sesión en el terminal ASCII, y se comprueba que están bien establecidos los parámetros de conexión.

2. Se debe observar que en la esquina inferior izquierda del terminal ASCII haya salido la palabra *Connected.*

3. Se conecta el cable de alimentación al toma corriente de la pared mientras se tiene presionado el botón CFG del tablero de desarrollo.

4. Se conecta la energía al tablero de desarrollo al conector verde ubicado en la parte posterior del mismo (en el cual se indica a cual conector va el cable positivo y a cual conector va el negativo).

Después de haber realizado este proceso se despliega el mensaje *NO CARRIER OK* en el terminal ASCII, lo cual indica que se entró al modo de ingreso de comandos AT para poder realizar la configuración del módulo MHX2420. Cabe señalar que, en el modo de comandos el módulo MHX2420 no pasa datos por ninguna de sus interfaces, incluso la interfaz RF.

Previo a la configuración de los equipos es necesario considerar lo siguiente: qué tipo de aplicación se va a implementar, la topología y la distribución física de la red. También se debe considerar los registros a modificar para configurar los módulos según la función que vayan a desempeñar y las consideraciones antes mencionadas.

Una vez conectados y en modo de comandos se realizan cambios en la configuración del MHX2420 con comandos AT, los cuales en su mayoría modifican los valores de los registros de configuración. Los registros de configuración ya tienen establecidos valores por defecto de fábrica con lo cual, dependiendo de la aplicación y consideraciones mencionadas antes no es necesario cambiar todos los valores de los registros. El ingreso de los comandos se realiza digitando AT seguido del comando y posteriormente pulsando la tecla Enter, de la siguiente forma: AT<comando>[Enter].

A continuación se detallan algunos de los comandos más usados para la configuración y visualización del status del módulo MHX2420[5]:

- ATA o ATO: Cualquiera de los dos sirve para regresar al modo de datos una vez que se ha terminado de configurar el módulo.

- ATg o ATG: Cualquiera de los dos provee una herramienta muy útil que es un analizador de espectros. Esta herramienta despliega los niveles de señal recibidas dentro del entorno y rango de frecuencias en el que el módem está trabajando. El comando ATg toma un promedio de 256 muestras, mientras que el comando ATG 16000. Digitando el comando ATg se realiza un barrido de la banda en la que trabaje el módulo MHX2420 y proporciona información del valor medio y los picos de los niveles de señal detectados en cada canal. Los resultados que despliega el analizador de espectros se publican de la siguiente manera:

  ch 78 -137dBm *

  ch 80 -105dBm *****…

  El formato indica el número de canal, la potencia media, el número de asteriscos indica el número de – dBm del pico inferior y el número de puntos indica el número de + dBm del pico superior. Por lo tanto se interpreta que en el canal 78 no se recibió ninguna señal, puesto que los niveles de sensibilidad están entre -110dBm y -53dBm. Mientras que en el canal 80 se detectó una señal media de -105dBm, y sus picos son -110dBm (se obtiene restando el número de asteriscos, en este caso 5, del valor medio, -105-5=-110dBm) y el otro pico es -102dBm (se calcula sumando el número de puntos a continuación de los asteriscos, -105+3=-102dBm).

  De acuerdo al Datasheet se conoce que el canal 1 se encuentra en 2.4016GHz, existe una separación de 400kHz entre cada canal, hasta llegar al canal 202 a

[5] Los comandos se especifican tal como deben ser digitados en el terminal ASCII.

2.4820GHz. Con lo que si se desea calcular la frecuencia correspondiente al canal n, se usa la fórmula:

$$\text{Frecuencia} = 2401.6 + [(n - 1) \times 0.400] \text{ MHz} \quad \text{(Ec. 5.1)}$$

- ATI*n*: Este comando despliega información acerca del módulo MHX2420. Donde n puede ser un número del 1 al 7 y 255. El comando ATI255 indica las configuraciones de fábrica.

- ATN: Es un analizador de espectros avanzado, el cual provee un análisis detallado de un área específica del espectro de frecuencias en el que el módulo MHX2420 opera. Se puede ingresar la frecuencia de inicio y final para determinar el rango donde se realizará el análisis. También se puede indicar el incremento con el que se va a llevar a cabo el análisis y el tiempo que realiza el análisis en cada frecuencia. Se debe ingresar el comando de la siguiente forma:

  ATN Frec.inicial Frec.final S D[Enter]

  Donde:

  Frec.inicial= Frecuencia inicial en MHz (puede incluir entre 0 a 6 cifras decimales)

  Frec.final= Frecuencia final en MHz (puede incluir entre 0 a 6 cifras decimales)

  S= Incremento en kHz (desde 1 a 1000)

  D= Tiempo de análisis en cada frecuencia en ms (desde 1 a 1000)

- AT&F*n*: Carga las configuraciones por defecto para distintas aplicaciones, donde n va desde 1 a 9 y se refiere a cada tipo de aplicación según la tabla 5.4.

Tabla. 5.4: Tipos de configuraciones para el commando AT&F*n*.

Valor (n)	Tipo de Configuración
1	Modo Maestro Punto-Multipunto
2	Modo Esclavo Punto-Multipunto
3	Modo Repetidora Punto-Multipunto
4	Modo Maestro (Lento) Punto-Multipunto
5	Modo Esclavo (Lento) Punto-Multipunto
6	Modo Maestro Punto-Punto
7	Modo Esclavo Punto-Punto
8	Modo Maestro (Lento) Punto-Punto
9	Modo Esclavo (Lento) Punto-Punto

- AT&H0: Restringe al módulo el uso de ciertas frecuencias en la banda de 2.4GHz. Por defecto el módulo MHX2420 saltará en frecuencias a través de la banda entera en la que opera que va desde 2.400-2.835GHz. Sin embargo, para ciertas aplicaciones y ambientes de trabajo puede ser necesario evitar que el módem opere en ciertas frecuencias. Cabe recalcar que el módulo requiere un mínimo de ancho de banda para operar de forma óptima, por tanto no permitirá que se restrinjan muchas frecuencias.

- AT&V: Sirve para visualizar la configuración del módulo, es decir, muestra el nombre y valor de los registros.

- AT&W: Guarda y graba la configuración en la memoria no volátil del módem.

Por otra parte, existen comandos para cambiar la configuración de los registros. La mayor parte de la configuración se realiza mediante el cambio en los valores de estos registros. El formato para el cambio de los registros es:

Formato de Consulta: ATS<# de registro S>?[Enter]

Formato para cambio de registro: ATS<# de registro S>=<value>[Enter]

A continuación se detallan algunos de los comandos que manipulan los registros de configuración del módem MHX2420[6]:

- ATS0: Este registro determina el modo opera el módem al momento de ser encendido. Se puede configurar para que el módulo opere en modo de comandos cuando es encendido colocando el valor 0 en el registro. Por defecto está colocado el valor 1 en el registro con lo que al encender el módem el mismo opera en modo de transmisión de datos.

- ATS101: Define el rol del equipo, es decir, si funciona como Maestro (0), Esclavo (2) o Repetidora (1).

- ATS102: Sirve para establecer la velocidad a la que trabaja la interfaz RS232, por defecto se encuentra en 9600 bps. Las velocidades soportadas están especificadas en la tabla 5.5.

Tabla. 5.5: Velocidades de operación interfaz RS232.

Valor	Velocidad (bps)
0	230400

[6] Los comandos se especifican tal como deben ser digitados en el terminal ASCII

Valor	Velocidad (bps)
1	115200
2	57600
3	38400
4	28800
5	19200
6	14400
7	9600
8	7200
9	4800
10	3600
11	2400
12	1200
13	600
14	300

- ATS103: Este registro determina la tasa de transmisión en la que se darán las comunicaciones RF en una red determinada. Todos los módulos que pertenecen a una misma red deben estar configurados con la misma velocidad de transmisión. Se debe tomar en cuenta que incrementar la tasa de transmisión resulta la obtención de un mayor throughput. Sin embargo por cada incremento en la velocidad de transmisión del enlace se reduce en aproximadamente 1dB la sensibilidad. Por defecto la tasa de transmisión del enlace se encuentra configurada a 172800bps. Los valores posibles se resumen en la tabla 5.6.

Tabla. 5.6: Velocidades de operación interfaz RS232.

Valor	Velocidad (bps)
0	19200
1	115200
2	172800
3	230400
4	270000
5	340000

- ATS104: Es el registro de dirección de red. Todos los módulos MHX2420 que correspondan a una red determinada deben tener la misma dirección de red, la cual además de servir como una característica de seguridad. Esto permite que distintas redes en la misma área operen sin interferirse, añadiendo seguridad y

diferenciando las redes. La dirección de red puede ir desde 0 hasta 4000000000. El valor por defecto es 1234567890.

- ATS105: Es el registro de dirección de unidad. Debe ser un identificador único para cada módem en la red. Tiene 16 bits de longitud, por lo que su rango va desde 1 hasta 65534. Es importante conocer que el valor 1 está restringido para el módulo que opera como Maestro en la red y el valor 65535 es la dirección de broadcast.

- ATS107: Determina una máscara estática, cuya función es añadir seguridad a la red. Dicha máscara es colocada al momento de transmitir los datos y removida cuando es recibida en el módulo remoto. Puede contener hasta 32 caracteres.

- ATS108: Mediante este registro se establece la potencia de transmisión. La tabla 5.7 indica los posibles valores.

Tabla. 5.7: Potencias de salida del módem MHX2420.

Valor en dBm	Valor en mW
20	100
21	125
22	160
23	200
24	250
25	320

Valor en dBm	Valor en mW
26	400
27	500
28	630
29	800
30	1000

- ATS109: Este registro se aplica únicamente al equipo que está funcionando como Maestro, es el encargado de determinar el tiempo en el que todos los módems de una red cambian de frecuencia[7]. Al poner intervalos altos de tiempo se obtiene mayor throughput, sin embargo al colocar intervalos más pequeños de tiempo se disminuye la latencia particularmente de paquetes de datos pequeños. El valor por defecto es 20ms, el cual es satisfactorio para la mayoría de aplicaciones. Es importante considerar que si se realiza algún cambio en este registro de igual forma deben ser modificados los registros S102, S103 y S112. La tabla 5.8 refleja los posibles valores para este registro.

Tabla. 5.8: Tiempos de cambio de frecuencia.

Valor	Tiempo en milisegundos
0	1.5

[7] Salto de una frecuencia a otra

Valor	Tiempo en milisegundos
1	2.0
2	2.5
3	3.0
4	4.0
5	5.0
6	7.0
7	10
8	15
9	20
10	30
11	40
12	50
13	60
14	70

Valor	Tiempo en milisegundos
15	80
16	90
17	100
18	125
19	150

- ATS110: Mediante este registro se puede especificar el formato de los datos que serán transmitidos por el puerto serial RS232. Por defecto se encuentra configurado como 8 bits de datos, sin paridad y 1 bit de parada. Esta es la configuración utilizada para la aplicación implementada.

- ATS112: Determina el máximo número de bytes que debe ser encapsulado en un paquete. Paquetes de grandes tamaños dan como resultado el mejor throughput posible, sin embargo los paquetes pequeños tienen menor posibilidad de ser corrompidos, y en caso de una retransmisión tienen un menor impacto en el tráfico de la red. Los valores que pueden ser colocados van de 0 a 255. El valor que se está configurado por defecto es de 255.

- ATS113: Este registro determina el número máximo de retransmisiones de paquetes. En aplicaciones punto a punto el módem retransmitirá paquetes al remoto solo si es necesario. El paquete es descartado si las retransmisiones no son exitosas. El receptor eliminará los paquetes duplicados. El valor por defecto es 5, se lo puede configurar entre 0 a 255. En aplicaciones punto a multipunto el Maestro retransmite cada paquete el número de veces según lo configurado en el registro S113 y los esclavos retransmitirán solo si es necesario.

- ATS123: Este registro no es de configuración. Sirve para desplegar el valor promedio de la señal recibida en intervalos de 4 saltos de frecuencia. Se puede obtener valores entre -110 hasta -55 dBm.

- ATS133: Define el tipo de topología de la red. Puede ser Punto a Multipunto (0), Punto a Punto (1), P2P (3). Todos los módulos en una misma red deben tener el mismo tipo de topología de red.

- ATS140: Mediante este registro se configura la dirección de destino a la cual apunta el módem, es decir la dirección de unidad del equipo con el que se va a establecer el radio enlace. En aplicaciones Punto a Punto el esclavo debe apuntar a la dirección 1 siempre, mientras que el maestro debe apuntar a la dirección configurada en el esclavo.

- ATS142: Modificando este registro se define la interfaz física mediante la cual se va a realizar comunicación de datos. Por defecto está establecida la interfaz serial RS232. Las posibles opciones son: Interfaz RS232 (0), interfaz RS485 half-duplex (1), interfaz RS485full-duplex (2).

- ATS158: Determina el tipo de FEC utilizado, por defecto no se usa ningún tipo de FEC. La tabla 5.9 indica los posibles valores que se pueden configurar.

Tabla. 5.9: Tipos de FEC.

Valor	Tiempo en milisegundos
0	Sin FEC
1	Hamming (7,4)
2	Hamming (15,11)

Valor	Tiempo en milisegundos
3	Hamming (31,24)
5	Binario BCH (47,36)
6	Golay (23, 12, 7)
7	Reed-Solomon (15,11)

Estos son los registros de mayor importancia a ser configurados. Los demás registros se detallan en el Manual de Operación del MHX2420 en el Anexo C1.

5.4.1 Configuración del módem MHX2420 del receptor

En el receptor. A fin de configurar el módulo como Maestro sin handshake de hardware, con potencia de salida y tasa de transmisión por defecto se utilizan 2 comandos. El comando AT&F6 lo configura como Maestro dejando la potencia y tasa de transmisión con valores por defecto. En tanto que el comando AT&K0 deshabilita el handshaking. Con lo que los registros más significativos quedan configurados como se detalla en la tabla 5.10.

Tabla. 5.10: Configuración y Registros modo Maestro.

Descripción	Registro	Valor
Modo de operación	S101=0	Modo Maestro
Tasa de Transmisión Inalámbrica	S103=2	172800 bps
Tiempo de Salto de Frecuencias	S109=9	20 ms
Tamaño Máximo del Paquete	S112=255	255 bytes

Descripción	Registro	Valor
Dirección de Destino	S140=2	2 (Remoto)
Modo de Transmisión Serial	S142=0	Mediante la interfaz RS232
Brillo de los LEDS(%)	S149=100	100%
Velocidad de Transmisión de le Interfaz Serial	S102=7	9600 bps
Dirección de Red	S104=1234567890	1234567890
Potencia de Transmisión(dBm)	S108=30	30 dBm
Formato de los Datos (Interfaz RS232)	S110=1	8N1
Número de Retransmisión de Datos	S113=5	5
Tipo de Red	S133=1	Punto a Punto
Handshaking	K0	Deshabilita Handshaking de Hardware

5.5 INTEGRACIÓN DEL HARDWARE DEL CUBESAT KIT

Una vez configurado el microcontrolador del Tablero de Desarrollo del CubeSat Kit, se debe acoplar el módulo MHX2420 a dicho tablero y la antena tipo *Rubber Duck* a dicho módem. Adicionalmente se debe conectar una computadora mediante un cable serial DB9 hembra-hembra a la interfaz RS232[8]. El CubeSat Kit queda conectado como se muestra en la figura 5.3.

[8] De ser necesario se debe usar un conversor de interfaz de DB9 a USB

Fig. 5.3: Hardware del CubeSat Kit integrado.

Por otra parte, en el lado receptor, se acopla el módem con su respectiva antena al tablero de desarrollo de Microhard Corp., que a su vez se conecta a una computadora como se muestra en la figura 5.4.

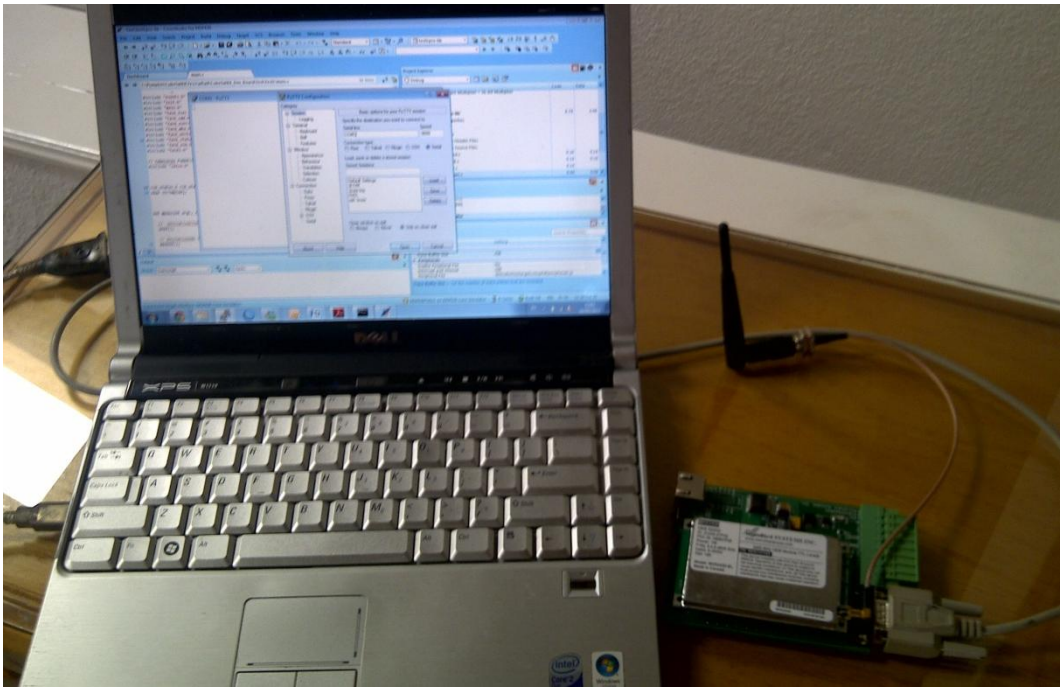

Fig. 5.4: Módulo MHX2420 integrado al tablero de desarrollo de Microhard Corp.

El proceso de transmisión será analizado en el capítulo 6.

CAPITULO 6

6. ANALISIS DE RESULTADOS

El presente capítulo muestra las pruebas y resultados obtenidos una vez establecido el enlace inalámbrico en la banda de 2.4GHz usando el método de transmisión *Frequency Hopping Spread Spectrum*, entre el Tablero de Desarrollo del CubeSat Kit y el Tablero de Desarrollo de Microhard Corp., mediante los módems MHX2420.

Fundamentalmente se llevaron a cabo 2 pruebas:

- Constatar que los datos recibidos por el remoto sean los correctos.
- Verificar el uso del espectro radioeléctrico del enlace.

6.1 ANÁLISIS DE DATOS RECIBIDOS POR EL REMOTO

El resultado de la aplicación de transmisión de texto lo que se detalla en la figura 6.1.

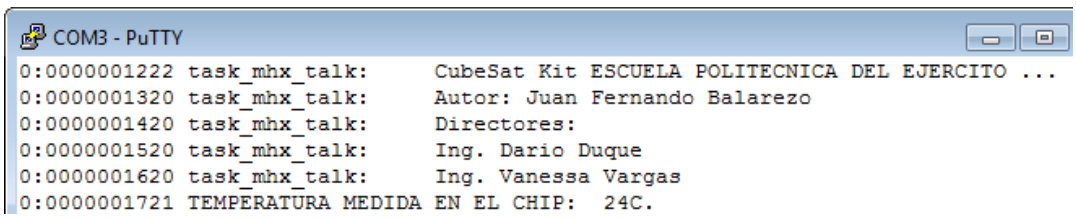

Fig. 6.1: Datos recibidos en el módulo MHX2420 remoto.

En la figura 6.1 se observan los mensajes que fueron recibidos por el módem MHX2420 remoto y estaban predeterminados a ser enviados en la tarea task_mhx.c.

Se puede comprobar la integridad de los datos recibidos en el módem remoto al comparar los mismos con los datos generados por el microcontrolador MSP430, utilizando para ello la interfaz RS232 de cada Tablero de Desarrollo. La figura 6.2 indica los datos transmitidos por el Tablero de Desarrollo del CubeSat Kit.

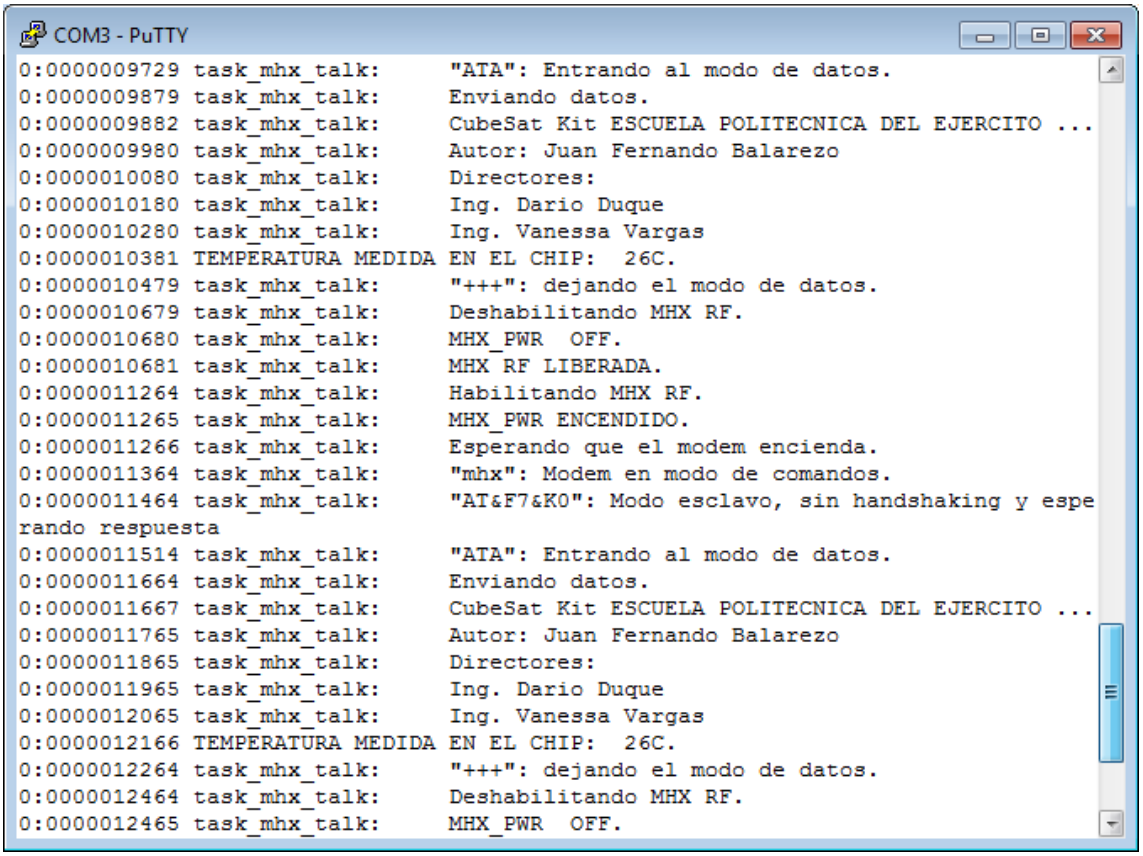

Fig. 6.2: Datos entregados por la interfaz RS232 del Tablero de Desarrollo del CubeSat Kit.

Una vez realizada la comparación se constató que los datos se recibieron de forma íntegra, sin errores ni variación en el contenido.

Cabe señalar que el tablero de desarrollo del CubeSat Kit puede ser monitorizado mediante su interfaz RS-232. Para ello se debe configurar la visualización del estado de tareas, funciones y procesos. También es necesario configurar un evento que permita censar el ingreso de caracteres mediante esta interfaz. Así desde el momento de ser

energizado, el tablero inicia la transmisión de información de los procesos y tareas que se están llevando a cabo por el sistema.

El haber configurado la visualización del estado de las tareas, funciones y procesos permite llevar un control del funcionamiento del mismo y de esta forma verificar que el sistema esté operando correctamente.

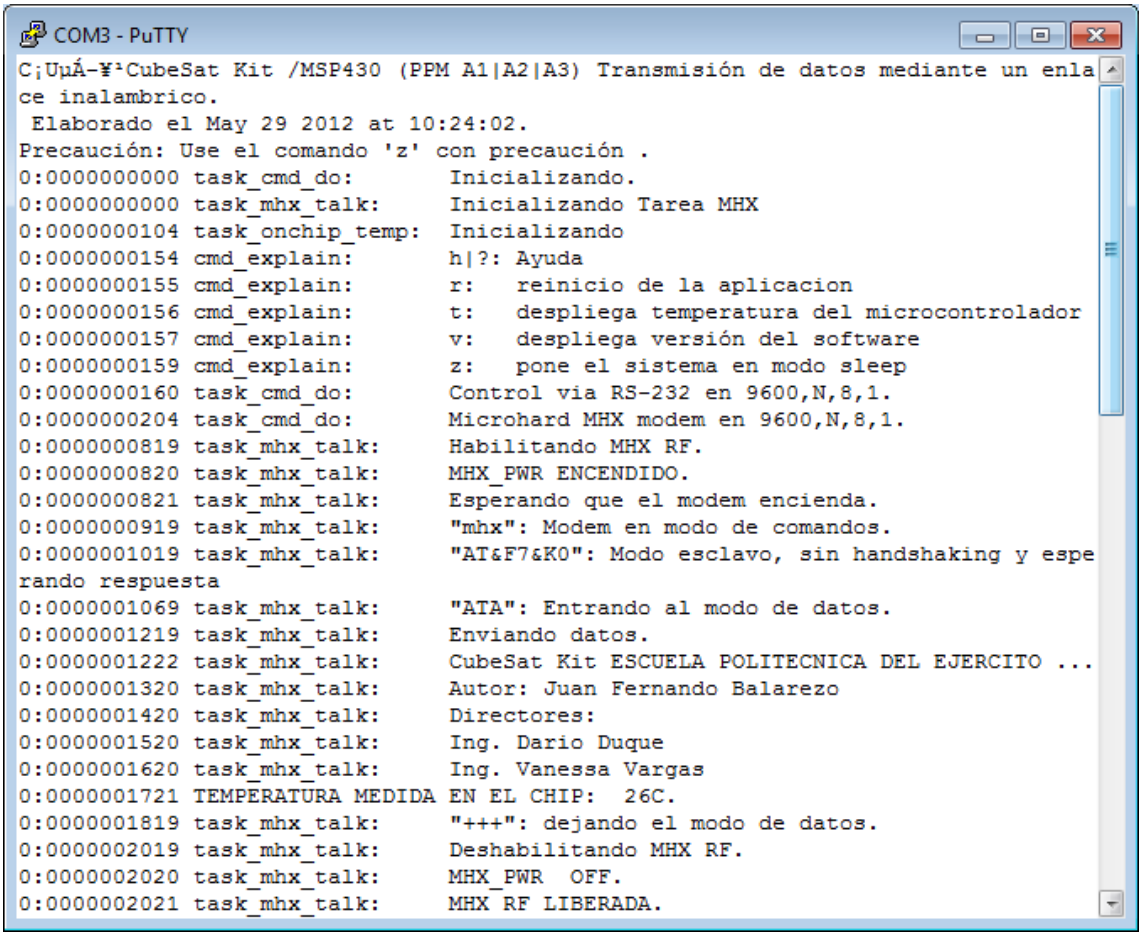

Fig. 6.3: Inicio de recepción de datos mediante la interfaz RS232 de CubeSat Kit.

En la figura 6.3 se observa el inicio de recepción de datos transmitidos por el tablero de desarrollo del CubeSat Kit hacia una computadora. Se despliega un mensaje informativo en el que se indica que se está recibiendo datos del CubeSat Kit regido por el microcontrolador MSP430 de Texas Instruments y se aprecia el título de la aplicación implementada. A continuación se envía la fecha y hora en la cual fue compilado y cargado por última vez el algoritmo en el microcontrolador que fue creado en el ambiente de desarrollo CrossWorks. Se despliega un mensaje de advertencia al digitar la tecla z puesto que esto llevara a que el sistema se apague.

En la figura 6.3 se observa las tareas que están siendo inicializadas, las cuales son: task_cmd_do (configurada dentro del archivo de código fuente task_cmd.c), task_mhx_talk (alojada dentro del archivo de código fuente task_mhx.c) y task_onchip_temp (configurada dentro del archivo de código fuente task_onchip_tem.c); en este orden basado en las prioridades asignadas a cada tarea. Después se observan dos mensajes con las configuraciones de las interfaces seriales del Tablero de Desarrollo del CubeSat Kit, estas son la interfaz RS232 (controlada por la UART0) y la interfaz serial que comunica el microcontrolador con el módem MHX2420 (controlada por la UART1). A partir de estos mensajes continúan ejecutándose las tareas programadas en el microcontrolador como se ve en la figura 6.3.

Por otra parte, también se puede enviar datos hacia la interfaz RS232 del tablero de desarrollo del CubeSat Kit mediante el teclado del terminal ASCII. Para conocer los comandos reconocidos por el tablero de desarrollo podemos observarlos al momento de inicializar el sistema, se muestran como aparece en la figura 6.3 o en cualquier momento se puede hacer uso del help como aparece en la figura 6.4, digitando h ó ?. En el caso de que se digite un carácter distinto se despliega un mensaje de aviso que dicho carácter no es reconocido por el CubeSat Kit, junto con el carácter que fue ingresado. Los posibles caracteres a ser ingresados son: h, ? (Menú de ayuda); r (Reseteo); t (Información de temperatura del microcontrolador); v (Versión del software); z (Apagado del sistema).

La información desplegada al ejecutar h indica información de ayuda y se muestra a continuación:

- h/?: Información de ayuda.

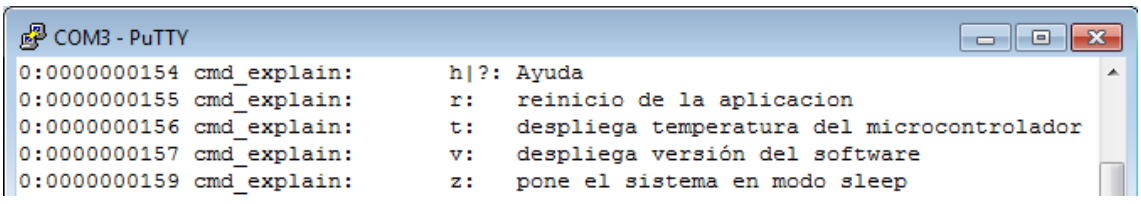

Fig. 6.4: Menú de Ayuda.

La información desplegada al ejecutar t indica información de la temperatura en el microcontrolador y se muestra a continuación:

- t: Información de la temperatura en el microcontrolador.

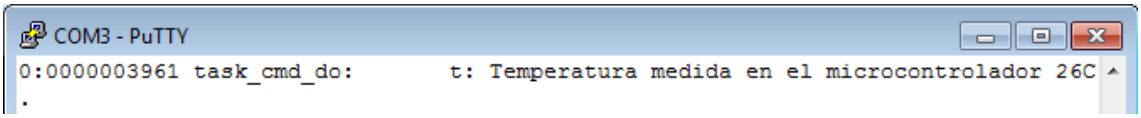

Fig. 6.5: Medición de la temperatura del microcontrolador.

La información desplegada al ejecutar v indica la versión del software y se muestra a continuación:

- v: Versión del software.

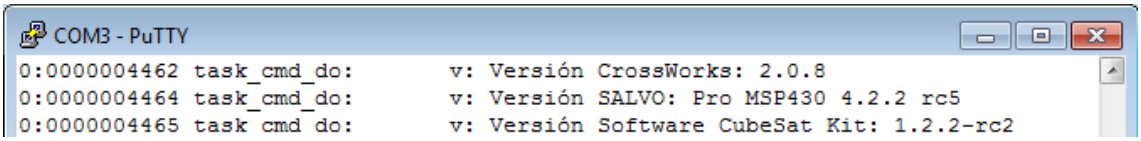

Fig. 6.6: Versión del Software.

Durante la ejecución de las pruebas se pudo observar un aumento de temperatura del microcontrolador en 4ºC. Así al inicio de la ejecución del programa la temperatura era de 24ºC tal como se muestra en la figura 6.1 en tanto que después de transcurridas aproximadamente 4 horas subió a 28ºC y se mantuvo constante como se muestra en la figura 6.7. Sin embargo este incremento no salió de los límites óptimos de funcionamiento del microcontrolador MSP430.

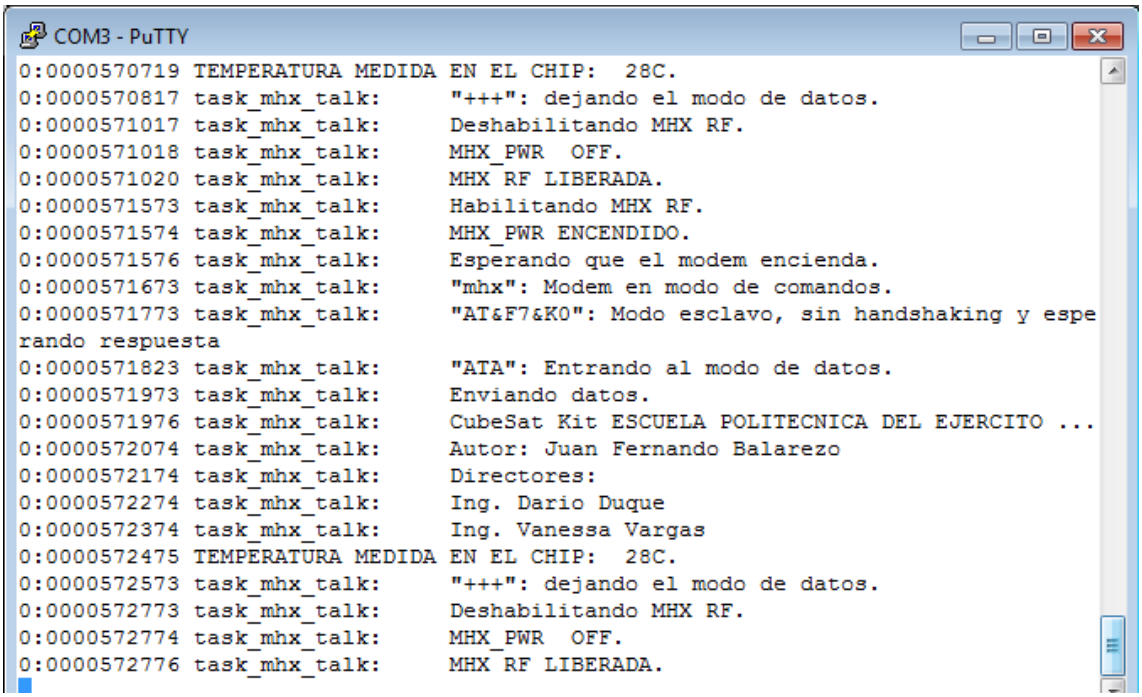

Fig. 6.7: Temperatura Pico alcanzada.

6.2 ANÁLISIS DEL USO DEL ESPECTRO RADIOELÉCTRICO

Se llevo a cabo un análisis del espectro que utilizan los módulos MHX2420 al transmitir y recibir datos usando un analizador de espectros. Como se ha indicado anteriormente, el módulo MHX2420 usa la tecnología de modulación de Espectro Ensanchado por Saltos de Frecuencia[9], realizando un salto de frecuencias cada 20ms según lo configurado en los registros de los módems. Los resultados de una de las pruebas realizadas se muestran en la figura 6.9. Mediante un analizador de espectros (como se aprecia en la figura 6.8) se pudo capturar las frecuencias de transmisión de algunos de los canales en los que el CubeSat Kit transmite información.

[9] *Frequency Hopping Spread Spectrum, FHSS*

Fig. 6.8: Analizador de espectros capturando datos del CubeSat Kit.

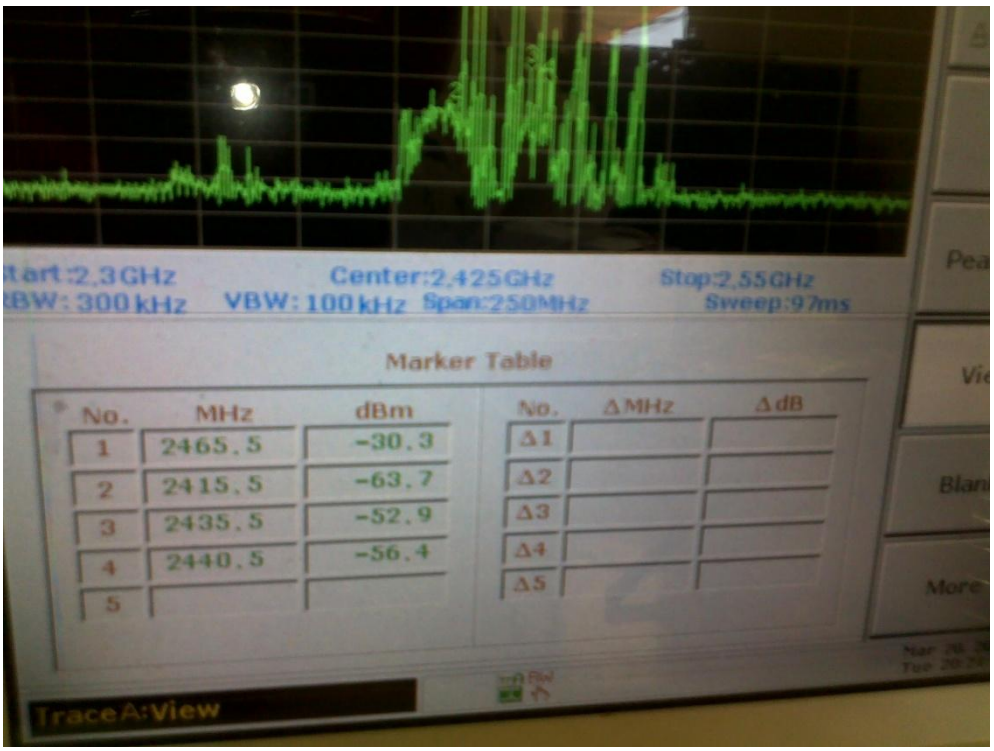

Fig. 6.9: Frecuencias capturadas por el Analizador de Espectros

A fin de obtener a que canal del módulo MHX corresponden los datos de las frecuencias medidas se utilizó la ecuación 5.1.

$$\text{Frecuencia} = 2401.6 + [(n - 1) \times 0.400] \text{ MHz} \quad \text{(Ec. 5.1)}$$

Y se despejo la variable del canal (n).

$$n=[(\text{Frecuencia}-2401.6)/0.400]+1 \quad \text{(Ec. 6.1)}$$

La frecuencia debe estar en MHz para usar la fórmula.

En la tabla 6.1 se puede ver a qué canal pertenecen las frecuencias capturadas en analizador de espectros.

Tabla 6.1: Frecuencias capturadas y su canal correspondiente.

Frecuencia(MHz)	Canal
2415.5	36
2435.5	86
2440.5	98
2465.5	161

De las pruebas realizadas se pudo concluir que la selección del canal que los módulos MHX2420 eligen para entablar comunicaciones entre sí se escogen de forma aleatoria entre los 202 canales de frecuencia disponibles. Como se explico en el capítulo 3, el MHX2420 posee 202 canales de frecuencia en los que establece la comunicación inalámbrica y en los cuales realiza el salto de frecuencias. El primer canal está ubicado en la frecuencia 2.4016GHz y existe una separación de 400kHz entre cada canal, siendo la frecuencia del último canal 2.4820GHz.

La potencia de transmisión usada en las pruebas realizadas fue de 30 dBm y la ganancia de la antena Rubber Duck es de 2dBi, aplicando la ecuación 6.2 se determina que el PIRE del sistema es de 32 dB asumiendo que no hay pérdidas ocasionadas por el cable.

PIRE=Potencia de transmisión + Ganancia de la Antena – Perdidas por cable (Ec. 5.2)

Nuestro sistema cumple las regulaciones FCC, que dicen que el PIRE no debe superar los 36 dB.

Los niveles de señal recibidos en otra prueba realizada con el analizador de espectros conectado a una computadora se detallan en la figura 6.10., las pruebas fueron realizadas a una distancia de 10 metros en el laboratorio.

MKR	Freq.(MHz)	Level	D_MKR	D_Freq.(MHz)	D_Level
0	2447	-21,2			
1	2429	-21,9			
2	2440	-25,6			
3	2408	-34,6			
4	2445	-27,4			

Fig. 6.10: Frecuencias capturadas por el Analizador de Espectros a 10 metros de distancia en el laboratorio

Los niveles de señal recibidos están dentro de los óptimos niveles de sensibilidad que el módulo MHX 2420 soporta[10].

Establecido el enlace entre módulos MHX2420, se realizaron pruebas de alcance de hasta 30 metros sin línea de vista[11] sin registrar pérdidas en la recepción de información. A partir de esta distancia se observaron datos errados en el remeto.

No se pudo continuar con las pruebas puesto que los módulos fueron averiados sin embargo las pruebas realizadas fueron suficientes para verificar que el CubeSat Kit está configurado correctamente. Pero otras pruebas deseables no se pudieron llevar a cabo, como por ejemplo realizar más pruebas en cuanto al alcance de los módems en diferentes ambientes.

[10] La sensibilidad mínima del módulo MHX2420 es de -108 dBm.
[11] Atravesando varios obstáculos y paredes.

CAPITULO 7

7. CONCLUSIONES Y RECOMENDACIONES

7.1 CONCLUSIONES

Cabe recalcar que el objetivo principal del presente proyecto de grado fue configurar e integrar el Picosatélite CubeSat Kit, basado en el RTOS (Sistema Operativo en Tiempo Real) SALVO usando el microcontrolador TI-MSP430, con módems MHX2420 en la banda de 2.4GHz. Este objetivo fue cumplido a cabalidad al desarrollar una aplicación de transmisión inalámbrica de texto plano en la banda antes mencionada usando la técnica de modulación Frequency Hopping Spread Spectrum (FHSS).

- Se evidenció que el factor clave en optimizar recursos de memoria fue el uso de SALVO, puesto que mediante este sistema operativo en tiempo real solo se usaron 9,6K de memoria FLASH del microcontrolador MSP430, el cual tiene un total de 55K de memoria FLASH, de esta forma solo se uso el 17.45% de los recursos que tenemos dejando el resto libre para el desarrollo de otras aplicaciones.
- En cuanto a los resultados obtenidos se pudo observar mediante un analizador de espectros los saltos en frecuencia que los módems MHX2420 dan en los canales en los cuales trabajan para transmitir y recibir información usando la técnica de modulación FHSS, realizando saltos de frecuencias aleatorios, según lo

configurado se daban cada 20 ms. Usando una potencia de transmisión de 30 dBm los niveles de señal medidos en el analizador de espectros son mayores a la sensibilidad mínima que el módulo MHX2420 debe recibir.

- Las pruebas de alcance no pudieron ser llevadas a cabo en su totalidad debido a la avería en los módulos MHX2420 y los tableros de desarrollo de Microhard Corp., pero en las pruebas que se llevaron a cabo se pudo constatar que hasta una distancia de aproximadamente 30 metros en un ambiente sin línea de vista, atravesando paredes y obstáculos, no se registraron pérdidas, a partir de esa distancia se registraron datos errados en la transmisión y recepción de información. Dado que las antenas que se utilizaron son omnidireccionales se concluye que para mejorar este alcance se debe usar otro tipo de antenas más directivas.
- Un factor clave en el éxito de desarrollo del presente proyecto fue el tener a disposición la arquitectura de hardware, sockets de conexión de dispositivos y periféricos, los tableros de desarrollo, ejemplos de código y un sistema de referencia. Las herramientas prediseñadas de hardware y software usadas en el presente proyecto disminuyeron la complejidad de implementación del sistema y desarrollo del proyecto en sí.
- Según las pruebas realizadas se pudo observar que el modo de operación de los módulos MHX2420 no pudo ser configurado de forma indistinta. Es decir, para este proyecto y esta aplicación desarrollada fue necesario configurar el módulo MHX2420 acoplado al tablero de desarrollo del CubeSat Kit en modo Esclavo, mientras que al módem MHX2420 integrado al tablero de desarrollo de Microhard Corp. se lo tuvo que configurar como Maestro, ya que en caso de configurar sus modos de operación al revés no se levantaba el enlace inalámbrico. Se asume que es debido a que el módulo MHX2420 del CubeSat Kit solo se enciende cuando va a transmitir información y se apaga luego de haberlo hecho, con lo que el tiempo que está encendido es limitado y el equipo que se encuentre configurado como Maestro debe estar encendido de forma constante.
- El proceso de instalación del sistema de desarrollo del CubeSat Kit es relativamente simple si se siguen los pasos descritos en el presente proyecto. Sin embargo, debido a la falta de información clara en un solo manual y la aparición

de problemas inesperados que requieren investigación y análisis lógico, el descubrir este proceso requirió mucho tiempo, paciencia y capacidad de deducción. Lo que hace a este documento muy valioso.

- El presente proyecto es el primer paso del DEEE y la ESPE en la incursión en investigación y desarrollo de sistemas satelitales de este tipo. Al cumplir con la primera fase del proyecto ESPE-CubeSat se espera fomentar al desarrollo de nuevas aplicaciones, que a largo plazo lleven a que la ESPE ponga en órbita un Picosatélite y ser pionera en el estudio del espacio usando estas herramientas.

7.2 RECOMENDACIONES

- A partir de los avances que se han llevado a cabo hasta el momento en el presente proyecto con el CubeSat Kit se pueden generar nuevas aplicaciones tales como el estudio de telemetría en el espacio, transmisión de video o fotografías desde el CubeSat Kit.
- Para optimizar tiempo y recursos al desarrollar nuevas aplicaciones, es necesario tomar como base las configuraciones ya generadas en este proyecto ya que en el cual se hace uso de la mayoría de herramientas que posee el CubeSat Kit.
- Las conexiones a llevar a cabo entre dispositivos son simples de realizar guiándonos en los manuales de instalación. Sin embargo se recomienda ser muy cuidadosos al manipular el hardware del sistema, puesto que en caso de hacerlo mal puede ocasionar daños irreversibles. Los dispositivos y elementos acoplados e integrados al CubeSat Kit son muy delicados por lo que requieren mayor cuidado, puesto que un simple golpe puede averiar o hacer fallar al sistema. Dos tableros de desarrollo y dos módems MHX2420 se averiaron debido a que un alumno realizó una manipulación errónea de los mismos.
- En cuanto a la instalación del software y drivers es importante hacerlo como se explica en los Anexos A y B para no tener problemas al momento de manipular e integrar tanto el software como el hardware.
- Se recomienda antes de iniciar con el desarrollo de nuevas aplicaciones se realice un estudio del presente proyecto que sirve como guía y base, ya que a

muestra paso a paso los procesos de instalación del software e integración del hardware. Seguido de los procesos de configuración de los equipos manipulados en el presente proyecto. Por otra parte es fundamental llevar a cabo una familiarización con las herramientas de desarrollo tanto de hardware como de software, mediante el uso de manuales que proveen sus distribuidores y el presente documento.

- El proceso de instalación del sistema de desarrollo del CubeSat Kit es relativamente simple si se siguen los pasos descritos en el presente proyecto. Sin embargo, debido a la falta de información clara en un solo manual y la aparición de problemas inesperados que requieren investigación y análisis lógico, el descubrir este proceso requirió mucho tiempo, paciencia y capacidad de deducción. Lo que hace a este documento muy valioso.

BIBLIOGRAFÍA

1. Pumpkin Inc, http://www.pumpkininc.com, 2012.
2. CubeSat Kit, http://www.cubesatkit.com, 2012.
3. CubeSat Organization, http://www.cubesat.org, 2012.
4. CubeSat Design Specification Rev. 12, California Polytechnic State University, http://www.cubesat.org/images/developers/cds_rev12.pdf, 2012.
5. CrossWorks for MSP430 Reference Manual, http://cdn.rowleydownload.co.uk/msp430/documentation/msp430_crossworks_reference_manual.pdf, 2012.
6. CubeSat Kit User Manual 3, Pumpkin Inc., Septiembre 2005, San Francisco
7. CubeSat Kit Development Board (DB) Hardware Revision: D, Pumpkin Inc., Septiembre 2009, Revisión B.
8. Building a Salvo Application with Rowley Associates CrossStudio for MSP430, Pumpkin Inc., Julio 2003.
9. Salvo Compiler Reference Manual CrossWorks for MSP430, Pumpkin Inc., Abril 2008.
10. Salvo User Manual, Pumpkin Inc., 2002, Version 4.2.2.
11. MSP430 Hardware Tools User's Guide, Texas Instruments, Mayo 2012.
12. MSP430 Datasheet, Texas Instruments, Marzo 2011.
13. MHX2420 Data Sheet, Microhard Corp, Canada.
14. MHX2420 Operating Manual, Microhard Systems Inc., Junio 2007, Canada.
15. MHX2420 User Note, Microhard Systems Inc., Junio 2007, Canada.

GLOSARIO DE TERMINOS

CIENCI: Centro de investigación científica ESPE.

FHSS: *Frequency Hopping Spread Spectrum*.

CSK: CubeSat Kit.

RTOS: *Real Time Operating System*.

TI: Texas Instruments.

CAL POLY: California Polytechnic State University.

P-POD: *Poly-PicoSatellite Orbital Deployer*.

PIRE: Potencia Isotrópica Radiada Equivalente.

RF: Radio Frecuencia.

FCC: *Federal Communications Commission.*

PPM: *Pluggable Proccesor Module.*

FEC: *Fordward Error Correction*.

CRC: *Cyclic Redundancy Check.*

DB: Decibelio.

BPS: Bits por Segundo.

PTP: Punto a Punto.

TDMA: *Time Division Multiple Access.*

RSSI: *Receive Signal Strenght.*

VDC: *Voltage* of Continuos Current.

GPS: *Global Positioning System.*

DCE: *Data Communications Equipment.*

DTE: *Data Terminal Equipment.*

ASCII: *American Standard Code for Information Interchange.*

TX: Transmisión.

RX: Recepción.

LED: *Light-EmittingDiode.*

WDT: *Watch Dog Timer.*

UART: *Universal Asynchromous Receiver-Transmitter.*