UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
ELÉCTRICA – TELECOMUNICACIONES

INTEGRACIÓN DE UN SATÉLITE CANSAT Y PROPUESTA DE SUS MEDIOS DE LANZAMIENTO PARA CAPACITAR RECURSOS HUMANOS A NIVEL UNIVERSITARIO

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA

PRESENTA:
ING. DIEGO NAVARRO LIEJA

TUTOR:
DR. ESAÚ VICENTE VIVAS

MÉXICO, D. F. ENERO DE 2015

Jurado Asignado

Presidente:	Dr. Salvador Landeros Ayala
Secretario:	Dr. Gutiérrez Castrejón Ramón
1er. Vocal:	Dr. Vicente Vivas Esau
2do. Vocal:	Dr. Martynyuk Oleksandr
3er. Vocal:	Dr. Matías Maruri José María

Agradecimientos

Este trabajo va dedicado a mi familia que es lo más importante en mi vida; a mi madre Martha Lieja, mi padre Ricardo Navarro, mi hermano Ricardo Navarro y Marisol Alcántara; siempre puedo contar con su apoyo y consejo para alcanzar mis metas, sin ellos nada de esto hubiera sido posible. Su amor y cariño me motivan a ser una mejor persona y a esforzarme cada día más.

Agradezco con mucho cariño a mis amigos y hermanos en quienes siempre puedo confiar y me han acompañado durante tantos años. Con mucho humor, tolerancia y ayuda mutua pero también con enojos y malos entendidos facilitan enormemente el camino recorrido.

Agradezco a mi tutor que me ha apoyado durante los últimos dos años, a mis profesores que me han enseñado todo lo que sé y a la UNAM que me ha dado tanto.

Resumen

En esta tesis se integró un satélite CanSat de tipo “Rover Come-Back”. Los satélites CanSat son satélites que no salen de la atmosfera terrestre del tamaño de una lata de refresco que se componen de subsistemas similares a los de un satélite real pero son de bajo costo y corto tiempo de construcción. El tipo “Rover Come-Back” se caracteriza por tener la capacidad de navegar autónomamente hasta un lugar predefinido o meta una vez que se encuentra en tierra.

El satélite CanSat desarrollado en esta tesis tiene integrado un sensor de presión y temperatura, GPS, acelerómetro, brújula digital, memoria extraíble, un sistema de comunicaciones, computadora de vuelo, motores y ruedas.

Se creó un sistema de comunicaciones a medida del CanSat que requiere bajo consumo de energía y capacidad de mandar una trama de datos a una estación terrena para dar seguimiento al CanSat en tiempo real.

Se creó software para configurar y controlar todos los sensores, memoria extraíble, receptor GPS, transmisor durante el vuelo de ascenso y descenso del CanSat y software de navegación para guiarlo hasta la meta una vez que se encuentra en tierra.

También se muestra de forma breve los métodos de lanzamiento más utilizados para competencias CanSat que pueden ser con cohete sonda, globo o drone.

También se propone una estrategia para crear una competencia nacional CanSat que tiene dos etapas, la primera educar en tecnología CanSat a un grupo de ingenieros de diferentes universidades y la segunda etapa requiere que los mismos ingenieros lleven los conocimientos adquiridos a sus universidades para capacitar estudiantes en tecnología CanSat e iniciar un concurso que involucre a estudiantes de diferentes instituciones educativas.

Contenido

Índice de Figuras

Capítulo 5: Introducción

Objetivos

El primer objetivo es la integración de un satélite CanSat en el Instituto de Ingeniería UNAM (IINGEN), es necesario incluir diferentes sensores, agregar un sistema de comunicaciones y crear ruedas que sean ligeras, sólidas y confiables, también desarrollar software operativo que controle el CanSat para que funcione de manera autónoma y finalmente integrarlo y validarlo.

El segundo objetivo es proponer una estrategia para establecer una competencia nacional CanSat en México con el propósito de implantar un mecanismo continuo de capacitación de estudiantes de ingeniería de nivel licenciatura y posgrado, que complemente el proceso de formación profesional de carreras de ingeniería.

Finalmente hacer una pequeña descripción de los métodos de lanzamiento más comunes utilizados en competencias CanSat y proponer un método adecuado para hacer lanzamientos CanSat en México.

1. Introducción

1.1 Antecedentes

México necesita mayor capital humano en el área espacial para impulsar su desarrollo.

Las instituciones de educación superior pueden jugar un papel clave para impulsar la Industria Aeroespacial nacional si se logra que los egresados cuenten con conocimientos sólidos, habilidades y capacidades que les permitan contribuir al desarrollo de nuevas tecnologías, participando en el diseño, construcción de partes y sistemas espaciales así como en su operación.

La Agencia Espacial Mexicana (AEM) tiene gran interés en el sector CanSat para ofrecer una capacitación nacional en el sector Aeroespacial y con miras a establecer una competencia Nacional CanSat que conlleve a que los mejores equipos de trabajo mexicanos puedan participar con sus desarrollos CanSat a nivel internacional [$_1$]. De igual forma, la red de ciencia y tecnología espacial (REDCYTE) de Conacyt inicia su apoyo al campo CanSat para capacitar recursos humanos.

Para crear una propuesta de una metodología para establecer una competencia nacional CanSat es útil contar con los conocimientos y experiencia para diseñar, integrar y validar un satélite CanSat.

1.2 Definición del problema

México tiene la necesidad de incrementar y mejorar día a día la cantidad y la calidad de sus recursos humanos en el campo Ingenieril, particularmente en el sector Aeroespacial que está contribuyendo a generar más y mejores empleos en nuestro país. Esto debe ser una prioridad para elevar el nivel de competitividad de nuestras instituciones educativas pero también de nuestra industria nacional.

1.3 Estado del arte

El término satélite hace referencia a un artefacto que es capaz de orbitar alrededor de un planeta o alrededor de otro satélite. Los satélites se dividen en dos tipos; los satélites artificiales y los satélites naturales. Los satélites artificiales son diseñados, integrados y lanzados al espacio. Después del lanzamiento no es posible recuperarlo para hacer cambios, correcciones o remplazos y en caso de que exista algún error el satélite podría perderse o disminuir sus funciones por lo tanto es de gran importancia eliminar las fallas antes del lanzamiento y dada la gran inversión de tiempo y costo que implica la creación de un satélite desde la planeación hasta el lanzamiento, no es viable construir otro satélite para remplazar el satélite defectuoso.

Es muy costoso el desarrollo de tecnología en la industria aeroespacial debido a que no permite errores, por lo tanto el capital humano con conocimiento en esta área es altamente valorado. Uno de los principales requerimientos para realizar un proyecto espacial es contar con personal calificado y con experiencia. Cualquier error en la planeación de la misión puede generar pérdidas de millones de dólares en un satélite, bajo estas condiciones nadie confiaría un proyecto espacial a personas sin experiencia y la experiencia se adquiere al trabajar directamente con el sistema que se quiere desarrollar.

Por otro lado es posible capacitar recursos humanos en el desarrollo de tecnología espacial creando satélites pequeños tipo Cubesat o CanSat. Con los avances tecnológicos y maduración de tecnologías contemporáneas ya resulta factible desde hace una década realizar proyectos satelitales pequeños en países en vías de desarrollo y en sus universidades. Tal es el caso de los satélites Cubesat y

CanSat que desde 1999 se emplean para entrenar recursos humanos y para validar nuevas tecnologías. Inicialmente se usaron en países desarrollados, sin embargo sus características de bajo costo y gran utilidad para incursionar en actividades espaciales con presupuestos reducidos (comparados con los requeridos para instalar satélites comerciales) han abierto las puertas en el campo espacial a muchas universidades de países en vías de desarrollo.

Sin embargo, actualmente existe un interés creciente en recintos universitarios de todo el mundo por incorporar los proyectos CanSat (satélites del tamaño de una lata de refresco) como medio de entrenamiento y adquisición de experiencia en diversas disciplinas de la Ingeniería. Los países desarrollados han madurado mucho sus técnicas y programas de entrenamiento CanSat, al grado de que lo emplean como medio de trabajo tanto en Universidades como en el bachillerato. Particularmente este tipo de programas ha sido muy exitoso pues los prototipos básicos que desarrollan equipos de alumnos están al alcance de muchos estudiantes.

Una vez que los estudiantes desarrollan un prototipo, lo pueden validar también de una forma muy sencilla: lanzándolos desde lo alto de edificios con paracaídas, lanzándolos desde globos aerostáticos, o lanzándolos desde cohetes sonda.

Aquellos estudiantes que participan en el lanzamiento de CanSats reciben un entrenamiento Aeroespacial de gran actualidad y calidad, solo comparable al que se obtiene al desarrollar proyectos espaciales o satelitales que implican costos mayores a los de un CanSat, grandes tiempos de desarrollo y gran infraestructura de trabajo. De ahí que sean tan atractivos de desarrollarse en universidades Mexicanas.

El concepto CanSat fue propuesto por el profesor Robert Twiggs de la Universidad de Stanford en 1999. CanSat es un satélite del tamaño de una lata de refresco. Es un concepto fundamental para introducir a los estudiantes de licenciatura al campo aeroespacial y provee una forma de que los estudiantes adquieran el conocimiento básico y conozcan algunos de los retos involucrados en el desarrollo de satélites. Los estudiantes serán responsables del diseño e integración de una pequeña carga útil electrónica que quepa en el interior de una lata de refresco.

Existen diversos medios de lanzamiento para un CanSat todos cumplen el propósito de elevar el CanSat a una altura aproximada y liberarlo para que comience a desempeñar sus funciones, los lanzamientos CanSat pueden ser por medio de cohetes sonda, globos aerostáticos o aeronaves multirotor.

El descenso a tierra del CanSat debe ser controlado, generalmente se logra por medio de un paracaídas. La adquisición de los datos generados por los sistemas

del CanSat y la recuperación del mismo permite a los estudiantes analizar las causas de éxito o fracaso.

Todo el concepto de CanSat es muy atractivo y enriquecedor para los estudiantes y contribuye a su aprendizaje en diversos campos de ingeniería. Dar a conocer este concepto a todo el mundo es uno de los objetivos principales de UNISEC (University Space Engineering Consortium) de Japon.

UNISEC es una organización sin fines de lucro que apoya a las universidades en actividades relacionadas con el desarrollo espacial como la creación de pequeños satélites y cohetes. [2]

Es precisamente UNISEC quien ha favorecido el entrenamiento de profesores universitarios mexicanos en programas CanSat, entre ellos el director de la presente tesis.

En EUA una competencia internacional llamada proyecto ARLISS (A Rocket Launch for international Students Satellites) es un esfuerzo colaborativo entre estudiantes, el programa de desarrollo de sistemas espaciales de la Universidad de Stanford y otras instituciones educativas y entusiastas de la cohetería de alta potencia de California para construir y lanzar CanSats. El grupo de cohetería de ARLISS provee los vehículos de lanzamiento que son capaces de liberar tres CanSats a 4 Km de altura lo que les da 15 min de caída, de esta forma se puede simular el tiempo de paso en el horizonte de un satélite de órbita baja; Robert Twiggs también fue fundador de ARLISS. [3]

Las misiones CanSat deben enfocarse a completar los objetivos de la misión, también el diseño y el desarrollo del CanSat debe producir un satélite capaz de alcanzar los objetivos propuestos con gran confiabilidad y robustez. Misiones básicas como de percepción remota, obtención de imágenes o adquisición de información utilizando diferentes MEMS son planeadas para principiantes en el proceso educativo de la tecnología CanSat.

Misiones más avanzadas como lanzar un CanSat desde un punto específico y lograr que el satélite se traslade a un punto final ya sea por aire o tierra, son reservadas para niveles de entrenamiento más avanzados en las competencias CanSat y se llaman competencias “ComeBack”.

Actualmente existe un interés creciente en recintos universitarios de todo el mundo por incorporar los proyectos CanSat (satélites del tamaño de una lata de refresco) como medio de entrenamiento y adquisición de experiencia en diversas disciplinas de la Ingeniería. Los países desarrollados han madurado mucho sus técnicas y programas de entrenamiento CanSat, al grado de que lo emplean como medio de

trabajo tanto en Universidades como en el bachillerato.[12] Particularmente este tipo de programas ha sido muy exitoso pues los prototipos básicos que desarrollan equipos de alumnos están al alcance de muchos estudiantes.

Con los avances tecnológicos y maduración de tecnologías contemporáneas ya resulta factible desde hace una década realizar proyectos satelitales pequeños en países en vías de desarrollo y en sus universidades. Tal es el caso de los satélites cubesat que desde 1999 se emplean para entrenar recursos humanos y para validar nuevas tecnologías y es el siguiente paso después de los CanSat en el desarrollo de tecnología satelital. Inicialmente se usaron en países desarrollados, sin embargo sus características de bajo costo y gran utilidad para incursionar en actividades espaciales con presupuestos reducidos (comparados con los requeridos para instalar satélites comerciales) han abierto las puertas en el campo espacial a muchas universidades de países en vías de desarrollo.

Aquellos estudiantes que participan en el lanzamiento de CanSats desde cohetes sonda, reciben un entrenamiento Aeroespacial de gran actualidad y calidad, solo comparable al que se obtiene al desarrollar proyectos espaciales o satelitales que implican costos mayores a los de un CanSat, grandes tiempos de desarrollo y gran infraestructura de trabajo. De ahí que sean tan atractivos de desarrollarse en universidades Mexicanas.

2. Integración de un satélite CanSat “Rover Come Back”

En este capítulo se describe el prototipo CanSat que se integró en esta tesis.

El CanSat está compuesto de 5 subsistemas diferentes que fueron diseñados específicamente para este CanSat, cada subsistema se compone de diversos elementos que le permiten realizar sus funciones.

2.1 Subsistemas del CanSat.

La computadora de vuelo (CV) es un DSP Texas Instruments TMS320F28027 de 32-bit a 60 MHz encargado de controlar la mayor parte de los elementos del CanSat. En su operación normal se encarga de encuestar los sensores GPS, de presión y temperatura, brújula electrónica y acelerómetro una vez por segundo, el lapso de tiempo de un segundo es arbitrario ya que los sensores de presión, temperatura y el acelerómetro pueden generar nuevos datos a más de 100 Hz, la brújula electrónica a 10 Hz y el GPS a 1Hz. Si se solicita información a los sensores antes de generar nuevos datos, los sensores envían la información almacenada en la memoria del sensor hasta que sea actualizada por una nueva medición.

Una vez que termina la comunicación con los sensores, la CV se encarga de guardar la información adquirida en una tarjeta de memoria micro SD que permite posteriormente recuperar la información almacenada y hacer un estudio de la misión utilizando los datos recabados.

La tarjeta de memoria esta formateada para utilizar el sistema de archivos FAT32 que es un estándar conocido y utilizado mundialmente que organiza los archivos y espacios de memoria y guarda un registro de cada uno de ellos.

El sistema de archivos FAT32 es muy complejo e innecesario para los requerimientos del CanSat, la programación del CanSat únicamente guarda una trama de datos con la información de los sensores en espacios de memoria continuos sin seguir el protocolo FAT32, por lo tanto una computadora no podrá leer los datos guardados en la tarjeta sin el programa adecuado.

Finalmente la CV crea una trama de datos con encabezado para enviarla al transmisor.

Después del retorno a tierra del CanSat, la CV controla la navegación del CanSat utilizando la información del GPS y de la brújula electrónica. Previo a la misión se carga en el programa del CanSat las coordenadas finales y de forma autónoma tiene que navegar hasta la meta.

Adicionalmente el microcontrolador ATmega88PA se utiliza para controlar principalmente el subsistema de potencia, está encargado de encender y apagar los dispositivos de acuerdo al estado de la misión para conservar energía.

Previamente y durante el vuelo se encienden todos los sensores y el transmisor mientras los motores, sistema de recuperación y corta hilo están apagados.

De regreso en tierra se apaga el acelerómetro, el sensor de presión y temperatura y el transmisor en caso de que la misión lo permita y enciende momentáneamente el corta hilo para separar el CanSat del paracaídas; posteriormente enciende el puente H para que las ruedas se puedan poner en funcionamiento.

Finalmente apaga todo y únicamente deja encendido el sistema de recuperación que debe tener la capacidad de mantenerse encendido por varias horas.

Adicionalmente monitorea el estado de la batería para cambiar el comportamiento del CanSat en caso de que la batería se esté agotando.

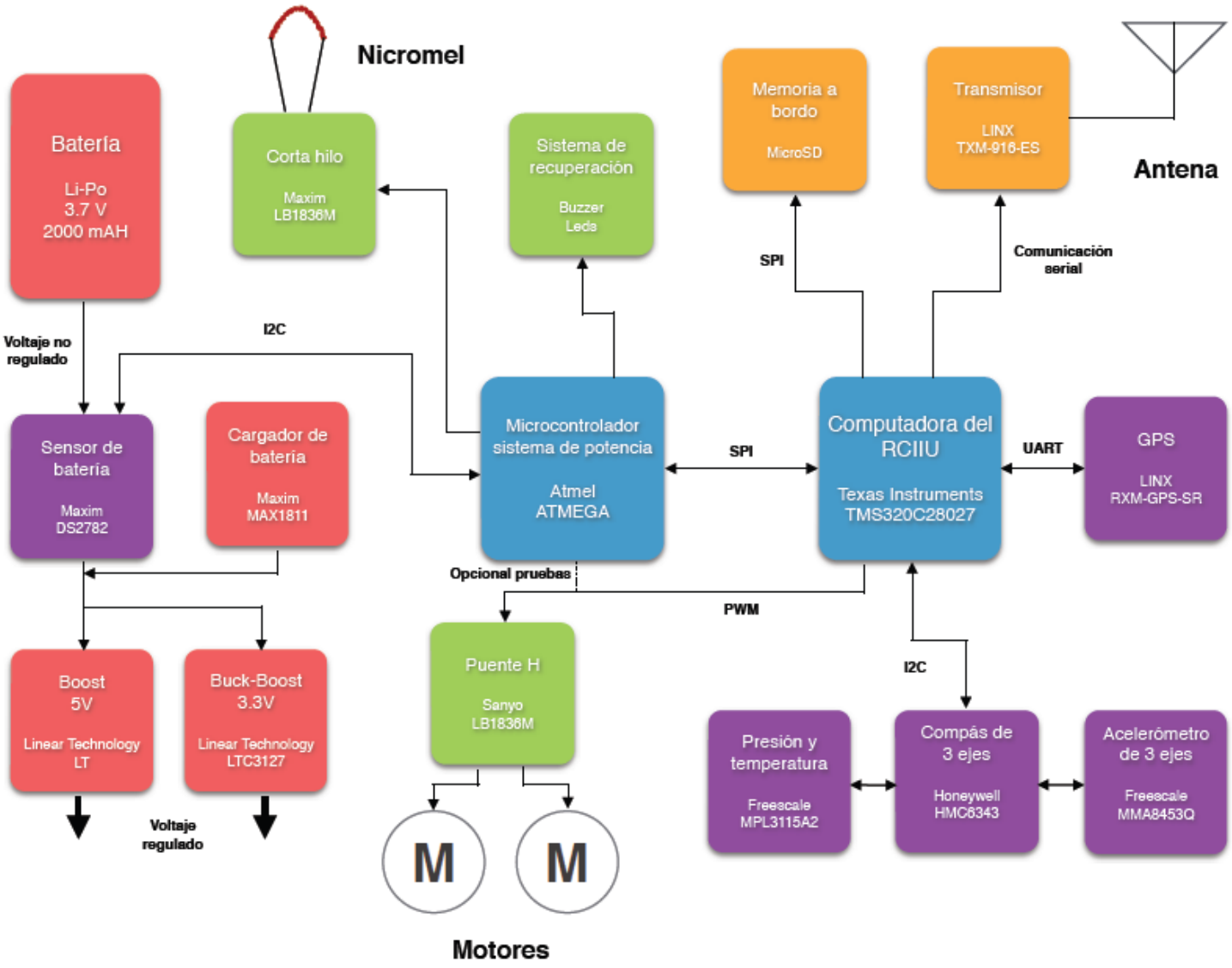


Figura 2.1 Diagrama del CanSat

Los subsistemas son:

- Potencia (rojo)
- Actuadores (verde)
- Sensores (morado)
- Control (azul)
- Transmisión y almacenamiento (amarillo)

2.1.1 Subsistema de Transmisión, almacenamiento y estación terrena

Almacena y transmite datos de las misiones del CanSat a la estación terrena. Los requerimientos del subsistema de transmisión y estación terrena depende de las especificaciones de la misión, algunas misiones otorgan puntos por la cantidad total de bits transmitidos que requiere alta velocidad de transmisión, otras misiones solamente requieren seguimiento del CanSat en tiempo real que no necesita altas velocidades de transmisión, incluso pueden solicitar comunicación bidireccional para controlar el CanSat por lo tanto es importante seleccionar los siguientes componentes de acuerdo a la misión, en este caso se utiliza para dar seguimiento en tiempo real al CanSat que requiere baja velocidad de transmisión y se envía la información una vez por segundo.

La potencia de transmisión debe ser suficiente para que la estación terrena pueda recibir datos en todo momento tomando en cuenta que el viento puede arrastrar el CanSat lejos de la misma por lo tanto se busca que la señal pueda recibirse a aproximadamente 5 Km, para este CanSat se calculó para 10 Km teniendo en cuenta que puede caer en algún lugar donde se atenúe la señal.

El subsistema de comunicaciones está formado por:

- Transmisor
- Amplificador
- Antena transmisora
- Antena receptora
- Receptor
- Memoria

2.1.1.1 Transmisor

El transmisor es el circuito integrado Linx TMX-916 – ES, es un módulo de un transmisor de canal único diseñado para transmisiones inalámbricas de información digital o analógica a una distancia máxima de 300 metros en exteriores con potencia de salida de 4dBm que es equivalente a 2.5mW. Utiliza modulación FM/FSK para proveer mayor seguridad contra el ruido en comparación con la modulación AM/ASK.

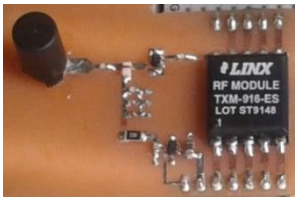


Figura 2.2 Circuito transmisor con antena.

Trabaja en la frecuencia de uso libre de 916 MHZ[15], tiene una desviación de 75 KHz a 3 v y utiliza un ancho de banda de 206 KHz transmitiendo a 56 Kbps.

La figura 4.3 muestra el diagrama del trasmisor. El transmisor modula el cristal con la señal digital en banda base presente en el pin de entrada de datos. La señal en banda base controla un VCO (oscilador controlado por voltaje) que forma parte del PLL (Phase-locked loop) que genera una señal con modulación FSK, después es amplificada, filtrada y enviada a la antena.

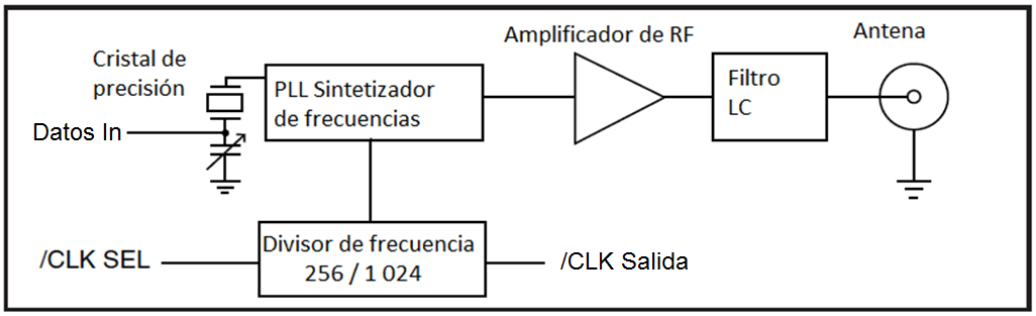


Figura 2.3 Diagrama del transmisor. [5]

2.1.1.2 Amplificador

La señal del transmisor es amplificada antes de ser enviada a la antena por un amplificador Sirenza SGA-4563 SiGe HBT MMIC [6] que entrega 15 dBm de potencia a la salida que equivale a 31.6 mW.

El circuito del amplificador es el siguiente:

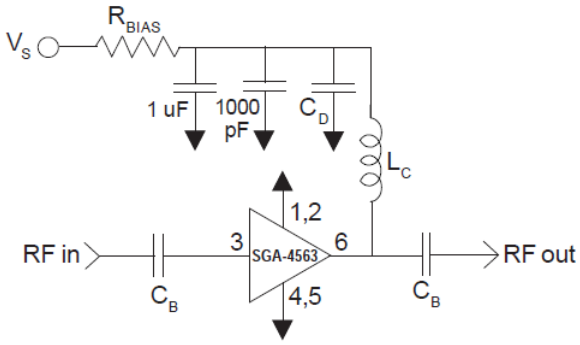


Figura 2.4 Diagrama del amplificador.

2.1.1.3 Antena Transmisora

La antena transmisora no puede ser muy grande debido a restricciones en las dimensiones del CanSat por lo tanto utilizamos una antena de ¼ de longitud de onda con ganancia de 1 dB, frecuencia central de 916 MHz y VSWR <2.

2.1.1.4 Antena Receptora

La antena receptora es una antena Log periódica que tiene 8.5 dB de ganancia en la frecuencia de 916 MHz con impedancia de 50 ohms y conector tipo N y VSWR <1.5.

2.1.1.5 Receptor

El receptor es de canal único para recepción inalámbrica digital o analógica, tiene configuración superheterodina con conversión a frecuencia intermedia a 10.7 MHz, su sensibilidad mínima es de -92 dBm a 9 600 baudios con BER de 10^{-5}.

Para unir los elementos es necesario que la línea de transmisión de microcinta tenga una impedancia de 50 ohms, la impedancia de la línea puede ajustarse al cambiar el grosor de la misma.

2.1.1.6 Cálculo de la distancia de transmisión

Utilizando la ecuación de Friis [7] podemos obtener un valor de referencia para la distancia máxima a la cual se puede transmitir y asegurar que la transmisión de datos hacia la estación terrena sea continua.

$$\frac{P_r}{P_t} = (1 - |\Gamma_t|^2) \cdot (1 - |\Gamma_r|^2) \cdot (\frac{\lambda}{4\pi R})^2 \cdot G_t \cdot G_r \qquad (1.1)$$

Dónde:

- Γ_t es el coeficiente de reflexión de la antena transmisora
- Γ_r es el coeficiente de reflexión de la antena receptora
- P_r es la potencia en el receptor
- P_t es la potencia en la salida del transmisor
- R es la distancia de transmisión
- G_t es la ganancia de la antena transmisora
- G_r es la ganancia de la antena receptora
- λ es la longitud de onda

Tomando en cuenta la sensibilidad mínima del receptor de -92 dBm obtenemos que la distancia máxima de transmisión es 16 Km, de esta forma aseguramos un enlace confiable entre el CanSat y la estación terrena aun cuando exista obstrucción en el camino. La distancia real a la que transmitirá el CanSat depende de diferentes factores como las especificaciones de la misión, la altura del lanzamiento y las condiciones meteorológicas; generalmente la distancia es menor a 5 Km.

2.1.1.7 Almacenamiento

Toda la información adquirida por los sensores es almacenada en una tarjeta de memoria MicroSD. La tarjeta puede ser removida para recuperar la información y analizarla.

2.1.2 Subsistema de potencia

Abastece al CanSat y sus subsistemas con diferentes voltajes, además de autorregularse. El subsistema de potencia está compuesto por:

- Baterías
- Convertidores DC-DC
- Cargador de Batería

2.1.2.1 Baterías

Las baterías son la fuente de energía del CanSat completo. Son baterías de Litio-polímero. Este tipo de baterías se caracterizan por su gran capacidad de carga y descarga eléctrica, tamaño compacto y prolongada vida útil.

Tienen un voltaje nominal de 3.7 V y son capaces de entregar 2000 mAh. Su peso aproximado es de 37g.

Sus características de peso y energía hacen que esta batería sea ideal para el CanSat, puede abastecer al CanSat de energía, incluyendo los dos motores para las ruedas que operan en el modo de navegación.

2.1.2.2 Convertidores Dc - Dc

Los convertidores son de marca Linear Technology modelo LTC3127 y LTC3122 tipo “buck-boost” de alta eficiencia (95%), se utilizan dos; los reguladores van conectados a la batería y proporcionara energía a todo el sistema, la batería entrega un voltaje de 3.7V y el regulador tipo “Buck” disminuye el voltaje a 3.3 V, esto se debe a que la mayoría de los componentes del sistema funcionan con 3.3V.

La salida de este regulador de 3.3 V se conecta al bus de alimentación donde la mayoría de los dispositivos están conectados. Otro regulador tipo “Boost” se utiliza para elevar el voltaje a 5 V que alimenta al amplificador de salida del transmisor.

Figura 2.5 Regulador Buck-Boost fabricado, con salida de 3.3V.

2.1.2.3 Cargador de batería

El circuito integrado para carga de batería marca Maxim Integrated modelo max1811, puede ser alimentado directamente desde un puerto USB o desde una fuente externa de hasta 6.5V. El cargador utiliza un FET interno para suministrar hasta 500 mA corriente de carga a la batería. La salida establece la corriente de carga a 100 mA ó 500mA.

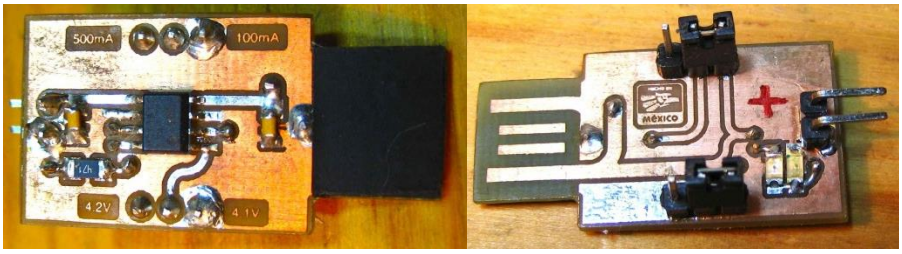


Figura 2.6 Cargador de batería fabricado, con opción de 500mA o 100mA y 4.2V o 4.1V.

2.1.3 Subsistema de Control

El subsistema de control administra los recursos del CanSat, procesa la información recibida y controla todos los subsistemas.

El subsistema de control está compuesto por:

- Computadora de vuelo
- Microcontrolador del sistema de potencia

2.1.3.1 Computadora de vuelo

La computadora de vuelo es un potente DSP de bajo costo marca Texas Instruments modelo TMS320F28027. Es un DSP en tiempo real de 32-bit a 60 MHz, con las siguientes características. [8]

Característica		28027 (60MHz)	
Empaquetado		38-Pin DA TSSOP	48-Pin PT LQFP
Ciclo de instrucciones		16.67 ns	
On-chip flash (16-bit word)		32K	
On-chip SARAM (16-bit word)		6K	
Código de seguridad para memoria flash/SARAM/OTP		Si	
Boot Rom (8K x 16)		Si	
One-time programable (OTP) ROM (16 bit Word)		1K	
Canales ePWM		8 (ePWM1/2/3/4)	
Entradas eCAP		1	
Temporizador Watchdog		Si	
12-Bit ADC	MSPS	4.6	
	Tiempo de conversión	216.67 ns	
	Canales	7	13
	Sensor de temperatura	Si	
	Dual Sample-and-Hold	Si	
32- Bit CPU timers		3	
Canales ePWM de alta resolución		4 (ePWM 1A/2A/3A/4A)	
Comparadores con DACs integrados		1	2
Inter integrated circuit (I^2C)		1	
Serial Peripherial Interface (SPI)		1	
Serial Communications Interface (SCI)		1	
I/O pins (compartidos)	Digital (GPIO)	20	22
	Analogo(AIO)	6	
Interrupciones externas		3	
Voltaje de alimentación		3.3v	

Este DSP es el responsable de controlar los subsistemas de sensores, transmisión y almacenamiento y una parte del subsistema de actuadores. Procesa los datos adquiridos de los sensores, guarda la información en memoria y envia la información al transmisor después de haber creado un paquete de datos. Se encarga de llevar a cabo la misión, utilizando la información de los sensores y da instrucciones a todos los demás subsistemas.

Para realizar las pruebas de validación y probar el software se utilizó la plataforma de desarrollo de Texas Instruments C2000 LaunchPad. Esta plataforma utiliza el mismo DSP antes mencionado.

Interactúa con el software IDE (Entorno de desarrollo integrado) CCS (Code Composer Studio) para ver la operación del DSP en tiempo real de esta forma se facilita la tarea de depuración de la programación; utilizando CCS se pueden cargar los programas al DSP fácil y rápidamente. No se requiere licencia para utilizar CCS en conjunto con esta plataforma de desarrollo.

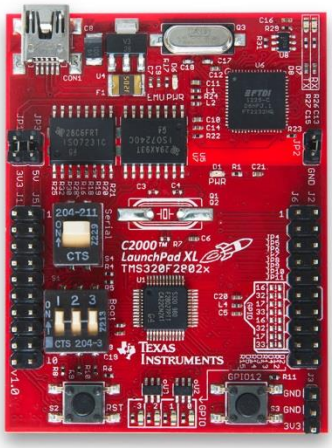


Figura 2.7 DSP TMS320F28027, Computadora de vuelo empleada en el CanSat.

2.1.3.2 Microcontrolador del sistema de potencia

La terminal “enable” de cada uno de los dispositivos de los subsistemas (Transmisor, amplificador, sensor de presión y temperatura, GPS, brújula, acelerómetro, corta hilo) está conectada al micro controlador que activa o desactiva los dispositivos de acuerdo al estado y las necesidades de la misión. La computadora de vuelo del CanSat envía ordenes al microcontrolador del sistema de potencia para cambiar los estados de actividad o reposo de los diferentes dispositivos y de esa forma habilitar o inhabilitar los subsistemas del CanSat.

El microcontrolador utilizado para este propósito es marca Atmel modelo ATmega88PA. Este modelo tiene 8 KBytes de memoria Flash, 512 Bytes de memoria EEPROM y 1 KByte de memoria RAM.

2.1.4 Subsistema de sensores

Se encarga de obtener diversos datos del entorno del CanSat para ser procesados por el CanSat. El subsistema de sensores está compuesto por:

- Sensor de batería
- Sensor de Presión y temperatura
- Compás de 3 ejes
- Acelerómetro de 3 ejes
- GPS

2.1.4.1 Sensor de carga de batería

Este sensor marca Maxim modelo DS2782 mide el voltaje y la corriente que fluye de la batería y es capaz de estimar el estado de carga de la batería; conociendo el consumo general del sistema se calculará la configuración óptima en los subsistemas para obtener el mayor rendimiento entre estados de carga.

2.1.4.2 Sensor de Presión y temperatura

Este sensor sirve para conocer el cambio en la presión y la temperatura de la atmosfera durante el vuelo de ascenso y descenso del CanSat, adicionalmente se puede calcular la altura aproximada a la que se encuentra el CanSat utilizando estos datos.

El sensor de presión y temperatura se comunica con el DSP usando el protocolo I2C a una frecuencia máxima de 400 KHz y puede realizar más de 100 mediciones por segundo.

El sensor de marca Freescale modelo Xtrinsic MPL115A2, emplea un sensor de presión MEMS para proveer mediciones precisas de presión y temperatura. Se comunica a través de un puerto I2C con el DSP.

Después de obtener los datos de las mediciones de presión y temperatura la información no puede ser utilizada, antes debe procesarse utilizando un algoritmo de compensación para obtener la presión absoluta compensada y la temperatura.

El sensor tiene la capacidad de detectar la presión en un rango de 50 Kpa a 115Kpa con un error de +-1Kpa y tiene una resolución de 0.15 Kpa.

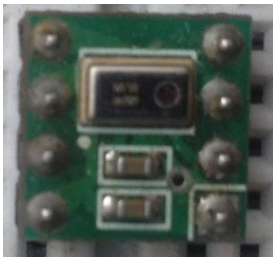


Figura 2.8 Sensor de presión y temperatura integrados al CanSat.

2.1.4.3 Acelerómetro de 3 ejes

Tener un acelerómetro abordo nos permite conocer en tiempo real y/o posteriormente el desplazamiento del CanSat en el espacio durante todo su trayecto desde el despegue hasta el regreso a la meta.

Conocer estos datos en tiempo real sirve para detectar eventos importantes durante la misión como el despegue, el inicio de la caída libre y el contacto con tierra, también permite identificar si se encuentra volcado de costado, o en una pendiente o si choca contra un objeto. El acelerómetro es capaz de identificar eventos como caída libre y choques, al hacerlo envía una interrupción a la computadora de vuelo.

Al poder identificar eventos importantes a partir del movimiento del CanSat es posible programar el CanSat para que reaccione adecuadamente ante situaciones previstas y disminuir la probabilidad de fallo de la misión.

Este acelerómetro de marca Freescale Semiconductor Xtrinsic MMA8453Q 3-Axis, tiene la capacidad de reconocer cuando se encuentra en caída libre, capacidad de orientación y detección de movimiento en tiempo real.

La obtención de datos puede variar entre 1.56 Hz y 800 Hz, la mayor exactitud del acelerómetro se obtiene con la frecuencia más baja de 1.56 Hz. Consume un máximo de 165 μA y este sensor se encuentra encendido durante todo el tiempo de operación del CanSat. Tiene comunicación I2C y está conectado directamente al bus I2C de la computadora de vuelo.

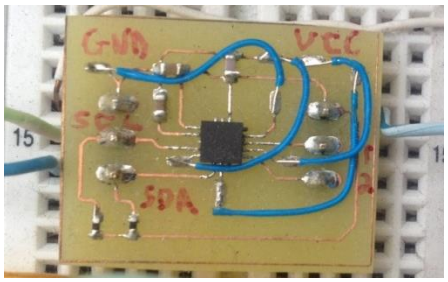


Figura 2.9 Acelerómetro integrado al CanSat.

2.1.4.4 GPS

Es indispensable integrar un receptor GPS al CanSat en misiones “come back” para que el CanSat pueda orientarse y saber también hacia donde se tiene que dirigir, es su principal herramienta de navegación.

La trama GPS es muy extensa sin embargo para guiar al CanSat hacia su destino únicamente necesita de cuatro campos: Latitud, indicador N/S, Longitud, indicador E/O, esto le permite saber al CanSat en donde se encuentra, hacia dónde dirigirse y a que distancia se encuentra el objetivo.

También recibe otra información que puede ser de utilidad como el tiempo, la altitud, estimación de la velocidad en tierra y si la información es válida o no en caso de que no esté recibiendo la señal de los satélites.

El GPS es de marca Linx Technologies y es de baja potencia consumiendo solamente 46 mW. Tiene comunicación UART y está conectado directamente al puerto serial de la computadora de vuelo.

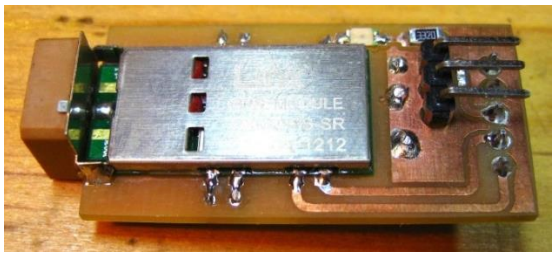


Figura 2.10 Receptor GPS integrado al CanSat.

2.1.4.5 Brújula de 3 ejes

El compás de 3 ejes también es un elemento necesario en un CanSat tipo “Rover Come Back” es completamente necesario para que el CanSat pueda saber en qué dirección exacta se está moviendo y hacia donde tiene que ir. Al igual que el GPS es una parte extremadamente importante del sistema de navegación.

Este pequeño circuito de marca Honeywell modelo HMC6343, tiene comunicación I2C y está conectado directamente al bus I2C de la computadora de vuelo.

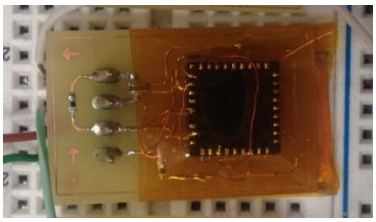


Figura 2.11 Brújula electrónica integrada al CanSat.

2.1.5 Subsistema de actuadores

El subsistema de actuadores consiste de transductores electromecánicos encargados de interactuar con el entorno del CanSat de manera física.

El subsistema de actuadores está compuesto por:

- Cortador de Hilo
- Puente H y motores de ruedas
- Sistema de recuperación

2.1.5.1 Cortador de Hilo

El cortador de hilo está compuesto por un circuito y un alambre de nicromo. El alambre de nicromo por un lado se encuentra haciendo un poco de presión sobre los hilos que sostienen el paracaídas y por otro lado se encuentra conectado al circuito alimentador que genera una corriente lo suficientemente alta para calentar el alambre de nicromo y este a su vez cortar los hilos que unen el paracaídas con el CanSat.

De esta forma el CanSat está libre para iniciar su navegación hacia la meta sin arrastrar el paracaídas y evitando el riesgo de que se atasque con algún obstaculo o se enrede con las ruedas.

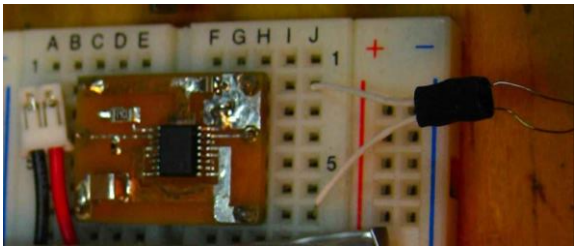


Figura 2.12 Circuito para cortar el hilo del paracaídas integrado al CanSat.

2.1.5.2 Puente H y motores

El Puente H es un circuito integrado que permite a un motor eléctrico de DC girar en ambos sentidos esto lo hace conmutando el sentido de la corriente.

Este circuito permite la marcha hacia adelante, hacia atrás, frenado y apagado. Consume un máximo de 800 mA y requiere un voltaje mínimo de operación de 2.5V.

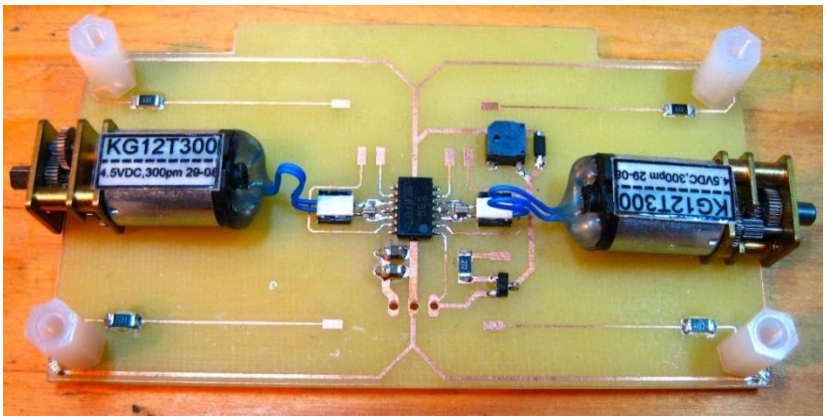


Figura 2.13 Puente H, motores y sistema de recuperación integrados al Cansat.

2.1.5.3 Sistema de recuperación

El sistema de recuperación está conformado por cuatro luces LED de color rojo que parpadean intermitentemente y se localizan en cada una de las esquinas del PCB de los motores. Además un Buzzer, que es una bocina de 5 x 5 mm, emite un zumbido a 75 dB. Esto sirve para localizar el CanSat al momento del aterrizaje o en caso de que no llegue a la meta y debe ser capaz de permanecer encendido durante varias horas.

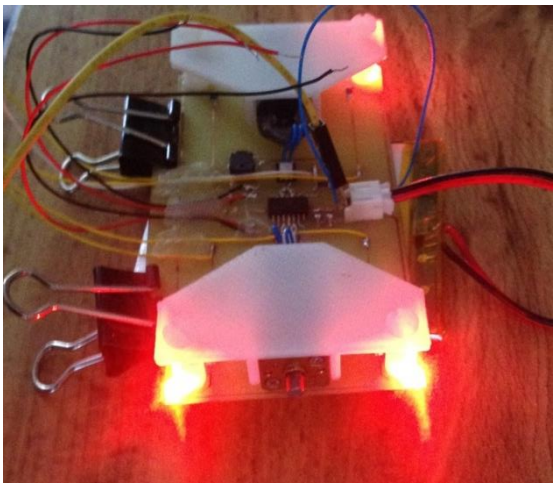


Figura 2.14 Sistema de recuperación del CanSat.

2.2 Pruebas de validación de los subsistemas

Es necesario realizar pruebas para validar la operación de los elementos que componen los subsistemas por diferentes razones.

1) Para probar que las conexiones del CanSat estén bien, es muy común encontrar que faltó conectar alguna pata de un CI, una conexión equivocada en la PCB o un punto mal soldado.
2) Para asegurar la interoperabilidad de los elementos o para comprobar que se obtiene el resultado esperado.
3) Para validar los subsistemas, se hicieron pruebas de cada uno de los elementos que componen a cada subsistema utilizando la plataforma de desarrollo C2000 LaunchPad de Texas Instruments.

Para utilizar esta plataforma de desarrollo es conveniente descargar el IDE Code Composer Studio (CCS) [9] que permite interactuar de manera fácil y rápida con el LaunchPad. El CCS no necesita licencia si se está utilizando en conjunto con el LaunchPad o cualquier microprocesador de la familia C2000.

En la imagen 4.16 se muestra el entorno de desarrollo CCS, la pantalla se divide en diferentes secciones: Debug, Expressions, el código que se está ejecutando y Console. En la sección Expressions que se encuentra en la parte superior derecha de la pantalla, podemos ver el valor de las variables del programa en tiempo real, esta sección es importante porque ahí se visualizan los resultados de las pruebas.

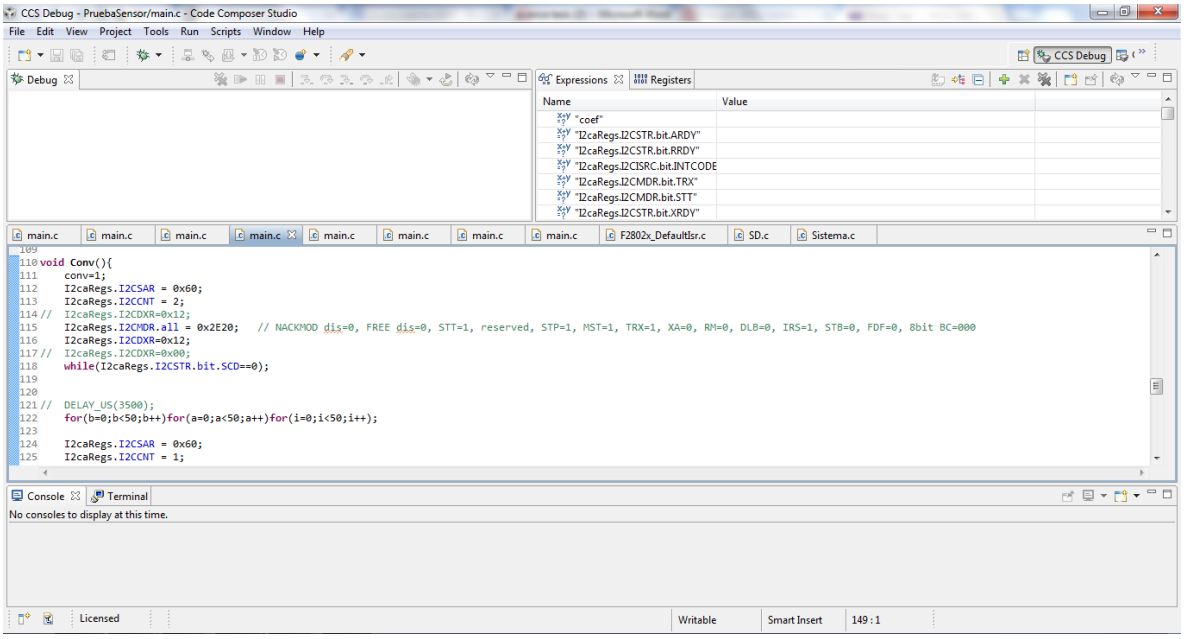


Figura 2.15 Entorno de desarrollo CCS.

Se creó software operativo para cada uno de los elementos desarrollado en CCS en lenguaje C. El programa se cargó al DSP y el dispositivo electrónico sujeto a prueba se conectó a los pines correspondientes del DSP.

Es posible ver los resultados de la ejecución del programa en tiempo real utilizando el depurador integrado en CCS.

Los elementos probados fueron los siguientes:

- Subsistema de sensores
    - GPS
    - Sensor de presión y temperatura
    - Acelerómetro de 3 ejes
    - Compas de 3 ejes
- Subsistema de actuadores
    - Puente H y motores
    - Sistema de recuperación (LEDs y buzzer)
- Subsistema de comunicaciones y almacenamiento
    - Transmisor
    - Receptor de la ET
    - Memoria a bordo
- Subsistema de potencia
    - Sensor de batería

2.2.1 Subsistema de sensores

2.2.1.1 GPS

El GPS se comunica con el DSP por un puerto SCI y está configurado por default para comunicarse a 9 600 baudios con 8 bits de datos, 1 bit de parada y sin bit de paridad. El GPS inicialmente envía información al DSP una vez por segundo y envía 8 mensajes diferentes cada uno con diferente propósito pero algunos campos se repiten en los diferentes mensajes.

El programa creado para validar el funcionamiento del GPS cambia la configuración de comunicación a 38 400 baudios enviando un mensaje al GPS, también le indica al GPS que deje de enviar los mensajes que no son necesarios, luego entra en un ciclo infinito esperando la recepción de datos.

El programa recibe un mensaje cada segundo con la información que el CanSat necesita para la navegación.

El mensaje tiene como encabezado “$GPRMC” que sirve para identificar el mensaje y todos los mensajes terminan con los caracteres <CR><LF>; a continuación se muestra una tabla con la información incluida en el mensaje [10]:

*$GPRMC,053740.000,A,2503.6319,N,12136.0099,E,2.69,79.65,100106,,,A*53*

Name	Example	Units	Description
Message ID	$GPRMC		RMC protocol header
UTC Time	53740		hhmmss.sss
Status	A		A=data valid or V=data not valid
Latitude	2503.632		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12136.01		dddmm.mmmm
E/W Indicator	E		E=east or W=west
Speed over ground	2.69	knots	TRUE
Course over ground	79.65	degrees	
Date	100106		ddmmyy
Magnetic Variation		degrees	Not Available, Null Field
Variation Sense			E=east or W=west (Not shown)
Mode	A		A=autonomous, D=DGPS, E=DR
Checksum	*53		
<CR> <LF>			End of message termination

Figura 2.16 Formato del mensaje GPRMC del GPS integrado al CanSat.

La siguiente tabla muestra la información recibida por el GPS:

$GPRMC, 020138.000,A,1918.5971,N,09913.5811,W,0.00,,190514,,,D,*69

ID del mensaje	$GPRMC
Hora UTC	020138.000
Estado	A
Latitud	1918.5971
N/S	N
Longitud	09913.5811
E/W	W
Velocidad	0.00
Curso	
Fecha	190514
Variación magnética	
Sentido de variación	
Modo	D
Checksum	*69
<CR><LF>	Fin del mensaje

Cada vez que el DSP recibe un mensaje del GPS, el programa separa la información útil del mensaje (Latitud, Longitud, N/S y E/W) y la guarda en dos variables.

Previamente se declaran dos variables dentro del programa con la Latitud y la Longitud de un punto dentro del Instituto de Ingeniería que representa el punto al que debe navegar el CanSat.

El programa utiliza los datos de Latitud y Longitud obtenidos del GPS para calcular una trayectoria recta hacia el punto al que tiene que llegar, la distancia se calcula utilizando el sistema WGS84 (Department of Defense World Geodetic System):

$$Dist(m) = 6378137 * ACos[Cos(LatA) * Cos(LatB) * Cos(LongB - LongA) + Sin(LatA) * Sin(LatB)] \quad (2.1)$$

Las latitudes y longitudes se encuentran en radianes.

También se calcula el azimuth entre el punto donde se localiza actualmente el CanSat y el punto al que tiene que llegar. El azimuth se calcula de la siguiente forma:

$$\tan \alpha = \frac{\sin(Long1 - Long2)}{cos(Lat1) \cdot tan\,(Lat2) - sin(\mathrm{Lat1}) \cdot cos(Long1 - Long2)} \quad (2.2)$$

El resultado del cálculo del azimut se muestra en el emulador junto con la distancia hasta el objetivo, las Latitudes y Longitudes en grados del CanSat y del objetivo.

(x)= azimuth	double	67.34634
(x)= distancia	double	5138.103
(x)= Long1	double	99.22637
(x)= Long2	double	99.18084
(x)= Lat2	double	19.32808
(x)= Lat1	double	19.31015

Figura 2.17 Resultado de la prueba GPS.

El resultado muestra que es necesario avanzar en línea recta con un azimut de 67.3° para llegar al punto que marcan las coordenadas especificadas en las variables Lat2 y Long2 y que se encuentra a aproximadamente 5,138 m en línea recta.

2.2.1.2 Sensor de presión y temperatura

Inicialmente es necesario obtener los coeficientes del sensor, los coeficientes a0, b1, b2, c12 se utilizan para compensar los errores inherentes al sensor en la obtención de medidas de presión y temperatura, el valor de los coeficientes es siempre el mismo en un mismo sensor pero diferente en cada sensor.

El segundo paso es solicitar al sensor que realice una medición, el sensor guarda los datos obtenidos en su memoria interna y el tercer paso es solicitar los datos obtenidos.

Los coeficientes obtenidos del sensor son los siguientes:

a0 = 2121.125
b1 = -2.507080078
b2 = -1.05090332
c12 = 0.000861406

Una vez obtenidos los valores de los coeficientes, presión y temperatura es necesario utilizar la ecuación de presión compensada para obtener el valor real de la presión en KPa.

La compensación de la presión se calcula con la siguiente ecuación:

$$Pcomp = a_0 + (b_1 + c_{12} \cdot T_{adc}) \cdot P_{adc} + b_2 \cdot T_{adc} \quad (3.1)$$

Dónde:

- Padc es la presión de 10 bits resultante del ADC del MPL115A
- Tadc es la temperatura de 10 bits resultante del ADC
- a0 es el coeficiente de desplazamiento de presión
- b1 es el coeficiente de sensibilidad de presión
- b2 es el coeficiente de desplazamiento de temperatura
- c12 es el coeficiente de sensibilidad de temperatura

Pcomp entrega un valor de 0 cuando la presión en el sensor es de 50 Kpa y entrega el máximo valor de la escala de 1023 cuando la presión en el sensor es de 115 Kpa.

La presión en Kpa se obtiene de la siguiente forma[17].

$$Presión\ (\text{Kpa}) = Pcomp \cdot \left[\frac{115-50}{1023}\right] + 50 \quad (3.2)$$

La altura se calcula utilizando los valores de presión y temperatura obtenidos del sensor en tiempo real, también utiliza constantes de presión a nivel del mar y una constante atmosférica. La altitud está dada en metros y fracciones de metro de acuerdo a la siguiente ecuación [18]:

$$h = 44330.77\ \{\ 1 - (\frac{p}{p_0})^{0.1902632}\} \tag{3.3}$$

Dónde:

- p0, es la presión a nivel del mar (101326 Pa)

Para obtener la temperatura real es necesario transformar el valor obtenido directamente del sensor, la temperatura es calculada de la siguiente forma:

$$Temperatura = \frac{(\ Tadc - 498.0\ )}{-\ 5.35 + 25} \tag{3.4}$$

La tabla siguiente muestra los resultados de la obtención de presión y temperatura y la presión compensada.

(x)= pres	double	76.30862
(x)= altura	double	2423.726
(x)= temperatura	double	21.82243

Figura 2.18 Resultado de la prueba de barómetro.

Dónde:

- pres es la presión compensada en Kpa
- altura está dada en (m) desde el nivel del mar
- temperatura compensada en °C

2.2.1.3 Acelerómetro de 3 ejes

El acelerómetro una vez que se enciende entra en estado de espera, en este momento es necesario configurarlo con las opciones deseadas como rango de detección que puede ser 2g, 4g u 8g y activar las interrupciones deseadas.

Posteriormente se envía un comando para activarlo para que comience su funcionamiento normal.

A continuación se presenta la respuesta del sensor al dejarlo en estado de reposo sobre cada uno de los 3 ejes:

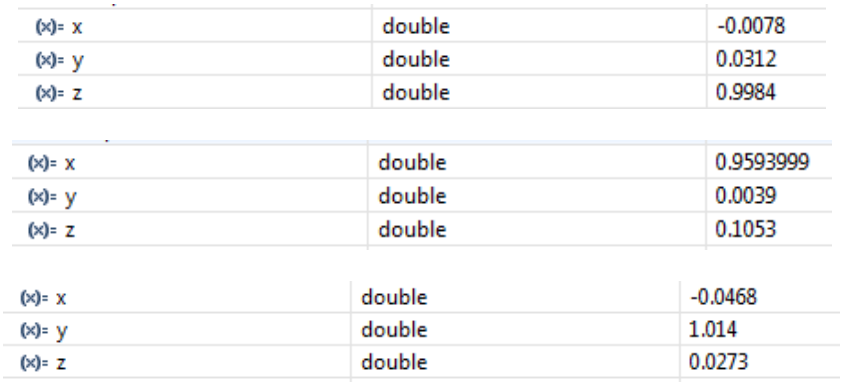


Figura 2.19 Resultado de la prueba del acelerómetro.

Cada variable x, y, z muestra las gravedades (g) a las que está sometido el sensor. Para llegar al valor final es necesario procesar la información recibida por el sensor.

Cada variable es un entero de 10 bits, se reciben dos bytes por cada variable (x,y,z), el primer byte de cada variable contiene los 8 bits más significativos y el segundo byte contiene los dos bits menos significativos.

El entero de 10 bits recibido representa un múltiplo de la unidad de medida (sensibilidad) del sensor, esta unidad de medida puede cambiar dependiendo del rango para el cual este programado el acelerómetro. Al multiplicar el entero de 10 bits por la sensibilidad mínima del sensor obtenemos el valor real en g.

La siguiente tabla muestra la sensibilidad para cada rango de detección:

Entero de 10 bits	Rango (sensibilidad)	Rango (sensibilidad)	Rango (sensibilidad)
	+- 2g (3.9 mg)	+- 4g (7.8 mg)	+- 8g (15.6 mg)
00 0000 0001	0.0039 g	0.0078 g	0.0156 g
00 0000 0000	0.0000 g	0.0000 g	0.0000 g
11 1111 1111	-0.0039 g	-0.0078 g	-0.0156 g

2.2.1.4 Brújula de 3 ejes

La brújula además de detectar la dirección de apuntamiento también puede detectar la inclinación en los tres ejes, para la aplicación del CanSat únicamente

estamos utilizando la dirección de apuntamiento, el dispositivo puede entregar una nueva medición hasta 10 veces por segundo.

Este sensor es muy preciso en sus mediciones, pero muy fácilmente se puede generar un error humano. La colocación del sensor en el PCB es muy importante ya que cualquier desvío en la alineación se refleja en la aplicación final por lo tanto es importante detectar el desvío y corregirlo en la programación al momento de entregar el resultado de apuntamiento.

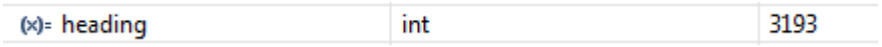


Figura 2.20 Resultado de la prueba de la brújula.

La variable “heading” indica la dirección de apuntamiento en decenas de grados con un rango entre 0° y 3599°.

Comprobando en tiempo real el apuntamiento medido por el sensor y la brújula electrónica de un teléfono móvil se obtiene una estimación del error de apuntamiento en el sensor de -24° aproximadamente, este error se puede corregir guardando un ángulo de corrección en el sensor [19].

2.2.2 Subsistema de actuadores

2.2.2.1 Puente H y motores

El comportamiento del motor está descrito por la siguiente tabla de verdad.

Entrada 1, 3	Entrada 2, 4	Salida 1, 3	Salida 2, 4	Modo
H	L	H	L	Adelante
L	H	L	H	Reversa
H	H	L	L	Freno
L	L	APAGADO	APAGADO	Espera

Para esta prueba se conectaron los motores a una bateria y el puente H a las terminales del DSP, se creó un programa que activa 4 puertos de uso general (GPIO) para controlar las 4 entradas que requiere el puente H. Dentro del programa los motores se pueden controlar en tiempo real cambiando el valor de los puertos GPIO, la fuerza que ejerce el motor para frenar es notable en cambio cuando se establece el modo de espera los motores quedan libres y continúan girando un poco.

Las ruedas están hechas de espuma de poliuretano lo que las hace muy ligeras y resistentes. El módulo de prueba y las ruedas son tan ligeros que las ruedas tienen poca tracción y se deslizan muy fácilmente por lo tanto en la versión final del CanSat las ruedas tendrán un recubrimiento plástico para evitar el deslizamiento.

Es necesario que el CanSat pueda girar por lo tanto se diseñó un programa que utiliza la información de los sensores como el GPS y la brújula para conocer cuántos grados necesita girar para alinearse en la dirección correcta. Utilizando la misma señal que utiliza el Buzzer se puede utilizar como temporizador para activar y desactivar los motores; al activarlos en direcciones contrarias, el CanSat girara durante un tiempo específico que depende de los grados que necesite girar.

El tiempo de giro se obtiene con la siguiente ecuación:

$$t = \frac{(azimuth - heading) \cdot C_c}{6 \cdot C_r \cdot rpm} \tag{4.1}$$

Dónde:

- t es el tiempo que tarda en girar hasta la posición deseada
- azimuth se obtiene con los datos del GPS
- heading la dirección de apuntamiento de la brújula
- C_c es la circunferencia del CanSat
- C_r es la circunferencia de las ruedas
- Rpm de los motores

2.2.2.2 LEDs y Buzzer

Los LEDs y el Buzzer se conectan a dos puertos PWM, el puerto de los LEDs se configuro con una señal triangular de 1 Hz que cambia el estado del puerto cada vez que se da un evento, cuando la señal llega al punto más alto, el puerto se configura con voltaje alto y cuando la señal pasa por cero el puerto cambia a señal baja, de esta forma los LEDs parpadean constantemente estando apagados medio segundo y encendidos medio segundo.

El buzzer funciona de manera similar sin embargo la frecuencia de operación del buzzer es de 4 KHz por lo tanto se configuró una señal triangular de 4 KHz que cambia de estado en el punto más alto y en cero.

2.2.3 Subsistema de comunicaciones y almacenamiento

2.2.3.1 Transmisor y receptor

El transmisor requiere una señal digital con valor máximo entre 3 V y 5 V y que la tasa de transmisión sea menor o igual a 56Kbps.

La señal es modulada en FSK y el voltaje de la señal define el desplazamiento máximo que puede ser de 75 KHz a 3 V o de 115 KHz a 5 V, como el DSP funciona a 3.3 V la señal que llega al transmisor también será de 3.3 V por lo tanto no es posible escoger el desplazamiento.

La velocidad de transmisión se configura en el puerto del DSP que conecta al transmisor, en este caso la comunicación se realiza por un puerto SCI (UART). Los CanSat no pueden tener comunicación bidireccional por lo tanto la comunicación dentro del CanSat es únicamente en un sentido. Aprovechando esta característica se utilizó el mismo conjunto de puertos SCI que utiliza el GPS, se configuro un puerto de transmisión y uno de recepción con protocolo SCI, el puerto de recepción se utiliza para recibir la trama de datos GPS y el puerto de transmisión se utiliza para mandar datos al transmisor. Esta decisión repercute en la configuración GPS debido a que la velocidad de transmisión tiene que ser igual a la de recepción por lo tanto el puerto SCI se configuró con la velocidad más cercana a 56 Kbps sin pasarse, resultando en una velocidad en transmisión y recepción de 38, 400 bps.

Para la prueba de transmisión se creó un programa que envía constantemente una trama de datos con una cadena de caracteres “S” con la leyenda “Hola Mundo!X” la X representa el número de iteración que se está enviando.

El transmisor y el receptor se probaron en conjunto, se conectó el receptor a otra plataforma de desarrollo “LaunchPad” y se configuró un puerto SCI con las mismas características que las configuradas para el transmisor que son:

- Un bit de parada
- Sin bit de paridad
- Protocolo “Idle-line”
- 8 bits por carácter
- 38, 400 baudios

En la imagen siguiente se puede ver el registro de recepción “ r ” en el cual se guardan los datos recibidos, se puede leer la leyenda “ Hola Mundo!72 ”.

La recepción en ese momento fue perfecta, el problema se presentó cuando la transmisión se volvió intermitente simulando el proceso de adquisición de datos

del CanSat. Todo el tiempo que tarda el DSP del CanSat en encuestar los sensores, crear la trama de datos, guardar los datos en la tarjeta SD y agregar un encabezado a la trama antes de enviarla al transmisor, además del tiempo restante que el CanSat queda en espera de la siguiente trama de datos, es tiempo en el que no se está utilizando el canal de comunicación y el receptor no detecta ninguna portadora. Mientras el receptor no detecta la portadora, la salida del receptor oscila y el DSP recibe datos aleatorios.

r	char[12]	0x00008806@Data
[0]	char	H
[1]	char	o
[2]	char	l
[3]	char	a
[4]	char	.
[5]	char	M
[6]	char	u
[7]	char	n
[8]	char	d
[9]	char	o
[10]	char	!
[11]	char	72 (Decimal)

Figura 2.21 Resultado de la prueba de Tx y Rx.

El programa creado para el receptor tiene la capacidad de reconocer el encabezado de una trama de datos y descartar todo lo demás; es posible que los datos aleatorios lleguen a coincidir con el inicio de trama y continúe recibiendo información apócrifa.

2.2.3.2 Memoria a bordo

El programa que controla la tarjeta de memoria SD [$_{11}$] fue difícil de crear debido a que la tarjeta sigue un protocolo complicado donde se tienen que seguir una serie de pasos para inicializar el sistema de la tarjeta.

El diagrama de estados para inicializar la tarjeta se presenta en la figura 4.24.

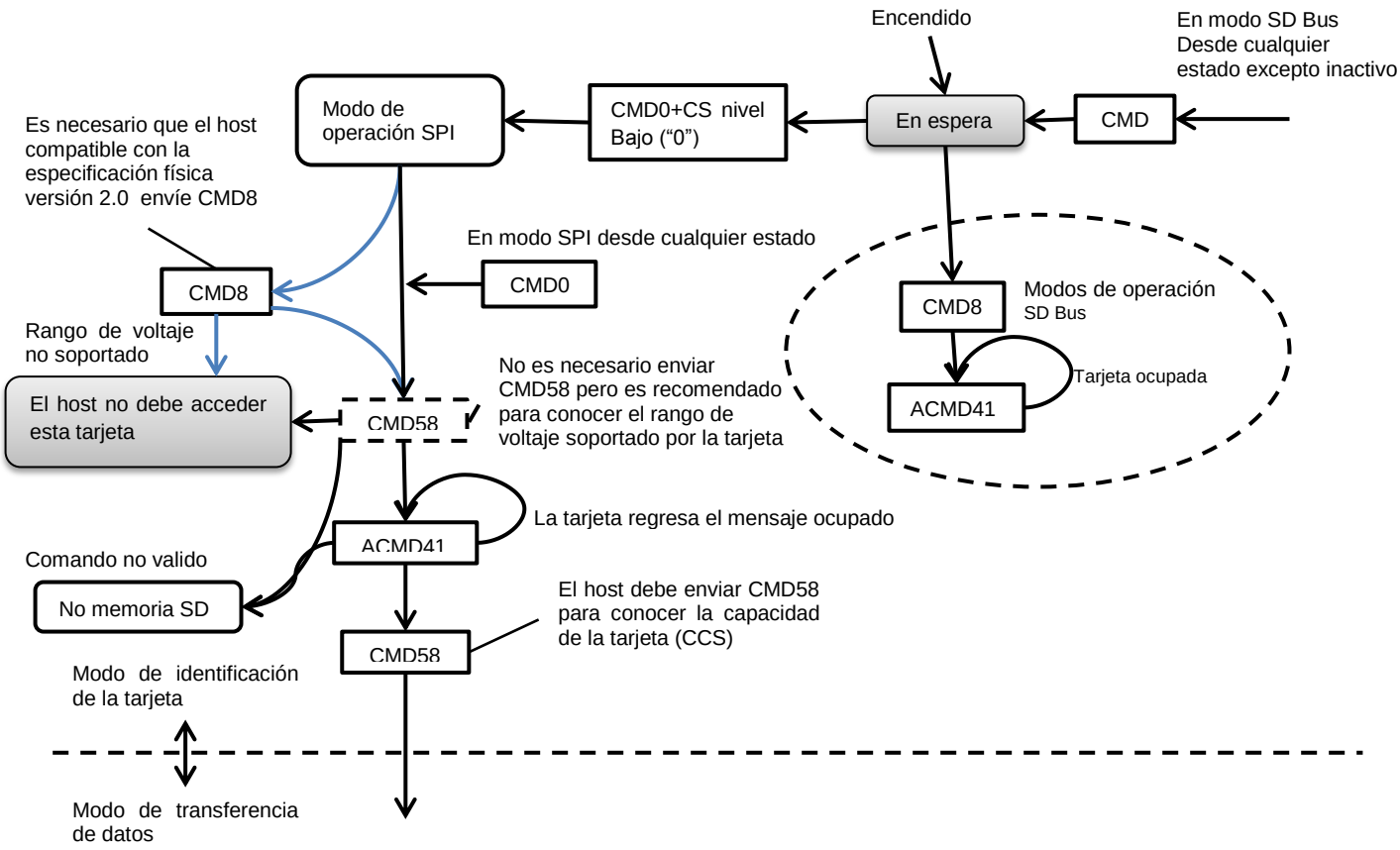


Figura 2.22 Diagrama de inicialización de la tarjeta SD.

Además algunos paquetes de datos tienen que incluir un campo CRC7 (Cyclic Redundancy Check) al final del paquete.

Cada vez que se envía un comando, la tarjeta envía una respuesta y existen 7 mensajes de respuesta con formatos diferentes y dos “tokens” de control; el programa de la tarjeta es capaz de diferenciar los mensajes y actuar de forma debida.

Una vez inicializada la tarjeta es conveniente enviar un comando que cambia el tamaño de bloque y lo ajustamos al tamaño más pequeño que es de 512 bytes, con el propósito de hacer más eficiente el uso de la tarjeta ya que la trama de datos es menor a 512 bytes y cada vez que se envía un comando de lectura o escritura se tiene que hacer en múltiplos del bloque especificado.

Una vez inicializada la tarjeta, la lectura y escritura es muy eficiente y se puede realizar más rápido que cualquier otro dispositivo, la velocidad más lenta es de 12.5 MB/s

El programa de prueba inicializa la tarjeta y la configura para un bloque de 512 Bytes, después escribe los datos de la variable “S” iniciando en un espacio de memoria escogido arbitrariamente y finalmente lee los datos a partir del espacio de memoria donde se comenzó a escribir y los guarda en la variable “r”.

r	char[515]	0x00008880@Data
[0 ... 99]		
(x)= [0]	char	.
(x)= [1]	char	P
(x)= [2]	char	r
(x)= [3]	char	u
(x)= [4]	char	e
(x)= [5]	char	b
(x)= [6]	char	a
(x)= [7]	char	.
(x)= [8]	char	o
(x)= [9]	char	f
(x)= [10]	char	i
(x)= [11]	char	c
(x)= [12]	char	i
(x)= [13]	char	a
(x)= [14]	char	l

Figura 2.23 Resultado de la prueba de la tarjeta SD.

2.2.4 Subsistema de potencia

2.2.4.1 Regulador “Boost” de 5 V

Se realizó una simulación del comportamiento del regulador “Buck-Boost” de 5 V, con una fuente de alimentación de 3.3 V.

Se hizo una prueba con carga dinámica para verificar que la red de compensación es estable en el intervalo de corrientes para la que fue diseñada.

La carga dinámica es representada por la línea roja en la figura 4.26, la carga cambia su consumo de corriente periódicamente y el dispositivo reacciona compensando el consumo de corriente en un tiempo muy corto, como consecuencia del aumento de corriente se genera un pequeño rizo en la señal de DC.

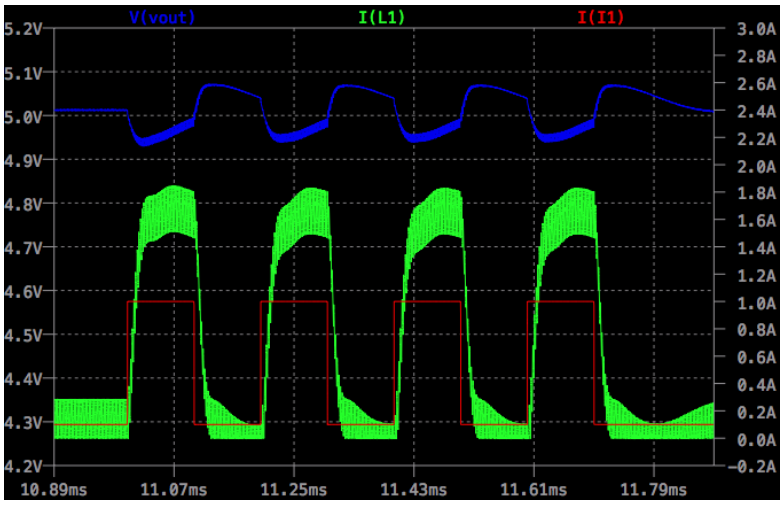


Figura 2.24 Simulación del regulador de 5.5 V.

Se realizó la simulación del regulador con el circuito mostrado en la figura 4.27, el regulador tarda 12ms en iniciar su operación y estabilizar el voltaje en 5 V.

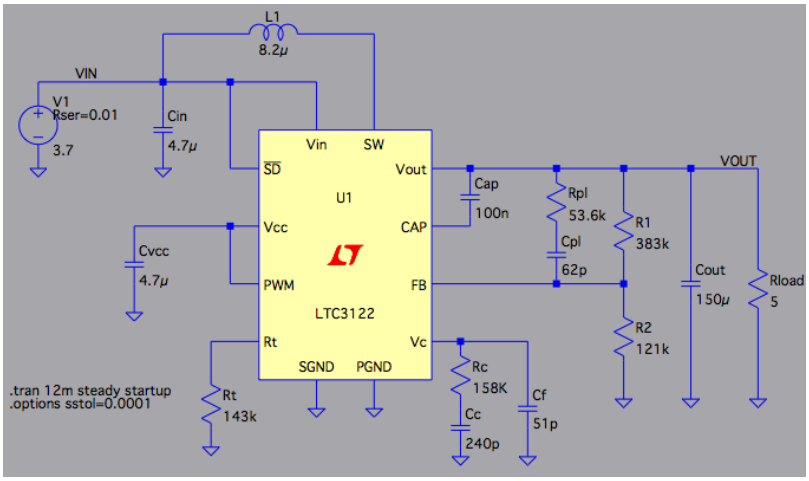


Figura 2.25 Diagrama del regulador de 5 V

De la simulación se obtiene el consumo de corriente, la disipación de potencia en cada elemento del circuito y la eficiencia se muestran en la tabla.

Ref.	I_{rms}	I_{pico}	Disipación
Cap	1 mA	3 mA	0 mW
Cc	0 mA	0 mA	0 mW
Cf	0 mA	0 mA	0 mW
Cin	15 mA	20 mA	0 mW

Cout	789 mA	1389 mA	6 mW
Cpl	0 mA	0 mA	0 mW
Cvcc	0 mA	0 mA	0 mW
L1	1504 mA	1675 mA	122 mW
R1	0 mA	0 mA	38 µW
R2	0 mA	0 mA	12 µW
Rc	0 mA	0 mA	0 µW
Rpl	0 mA	0 mA	0 µW
Rt	0 mA	0 mA	3 µW
U1	1504 mA	2111 mA	364 mW

Los valores que se obtuvieron fueron los siguientes:

Potencia de entrada: 5.51 W @ 3.69 V

Potencia de salida: 5.02 W @ 5.01 V

Eficiencia: 91.1 %

3. Revisión Preliminar de Diseño del CanSat rover come-back

Los documentos de Revisión Preliminar de Diseño (PDR) y de Revisión Crítica de Diseño (CDR) son documentos importantes en el proceso de planeación y desarrollo de todos los proyectos espaciales.

El PDR muestra el diseño original del CanSat; todas las ideas previas a la construcción del CanSat se integran en este diseño. Durante todo el proceso de creación del CanSat su diseño es evaluado y se hacen cambios en el diseño por diferentes razones y todos los cambios se reportan en la "Revisión Crítica de Diseño" donde se describe la versión final del CanSat.

3.1 Objetivos para una posible misión CanSat:

- Crear un CanSat con masa total menor a 350 gr con dimensiones máximas de 11.5 x 6.6 cm, medidas equivalentes a una lata de refresco.
- Integrar sensores de presión y temperatura, acelerómetro, compás digital y GPS para adquisición de datos de navegación en el transcurso de la misión.
- Guardar la información adquirida por los sensores en una tarjeta de memoria SD para analizar los resultados al término de la misión.
- Establecer comunicación con una estación terrena.
- Implementar un modo de ahorro de batería.
- Implementar un sistema de recuperación en tierra con luces y sonido.
- Mostrar en una pantalla en la estación información del CanSat como: trayectoria, datos de navegación y estado de la batería.

3.2 Subsistemas del CanSat

3.2.1 Estructura

El objetivo es crear una estructura ligera y resistente, la estructura está formada por el PCB que soporta todos los circuitos y cuenta con dos pequeños motores que mueven dos ruedas en los extremos del CanSat.

Es necesario crear ruedas resistentes, ligeras, baratas y confiables, con materiales fáciles de conseguir. El diseño original consta de una rueda central de plástico y rayos.

Los rayos miden 11 cm y están fijos a las ruedas, pueden doblarse contra el cuerpo del CanSat con ligas.

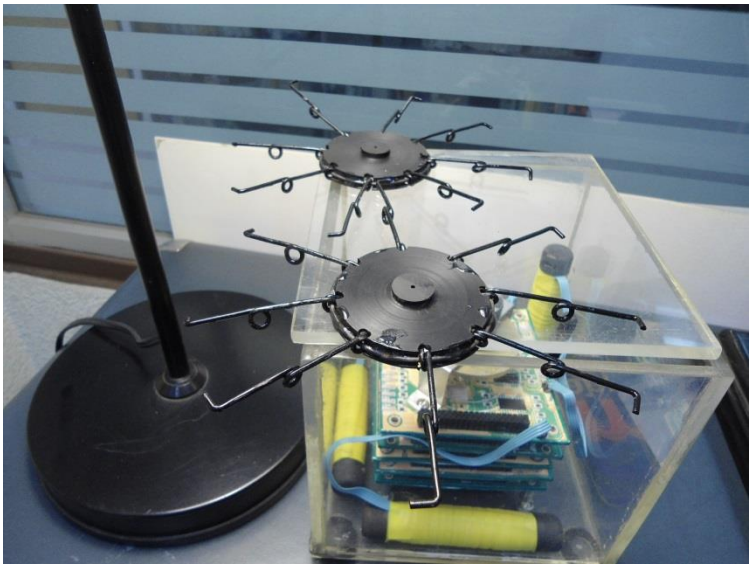


Figura 3.1 Primeras ruedas del CanSat.

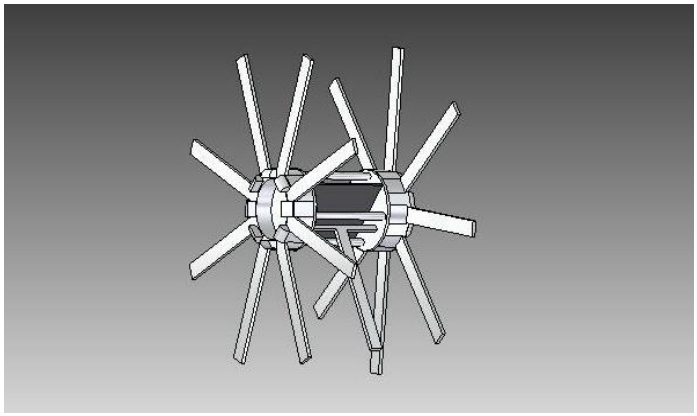


Figura 3.2 Diagrama de la estructura del CanSat

Antes del lanzamiento las ruedas se doblan para que el CanSat pueda guardarse en la bahía de carga del lanzador. Cuando cae a tierra las ruedas se abren y el radio de las ruedas aumenta, esto permite que el CanSat se eleve del suelo y pueda pasar sobre obstáculos pequeños.

Dos motores DC proveen de potencia a las ruedas en cada extremo del CanSat. Es posible agregar una tercera rueda en la parte trasera para aumentar la estabilidad, la tercera rueda consiste en un marco de metal con una pequeña rueda, el marco puede doblarse contra el cuerpo del CanSat.

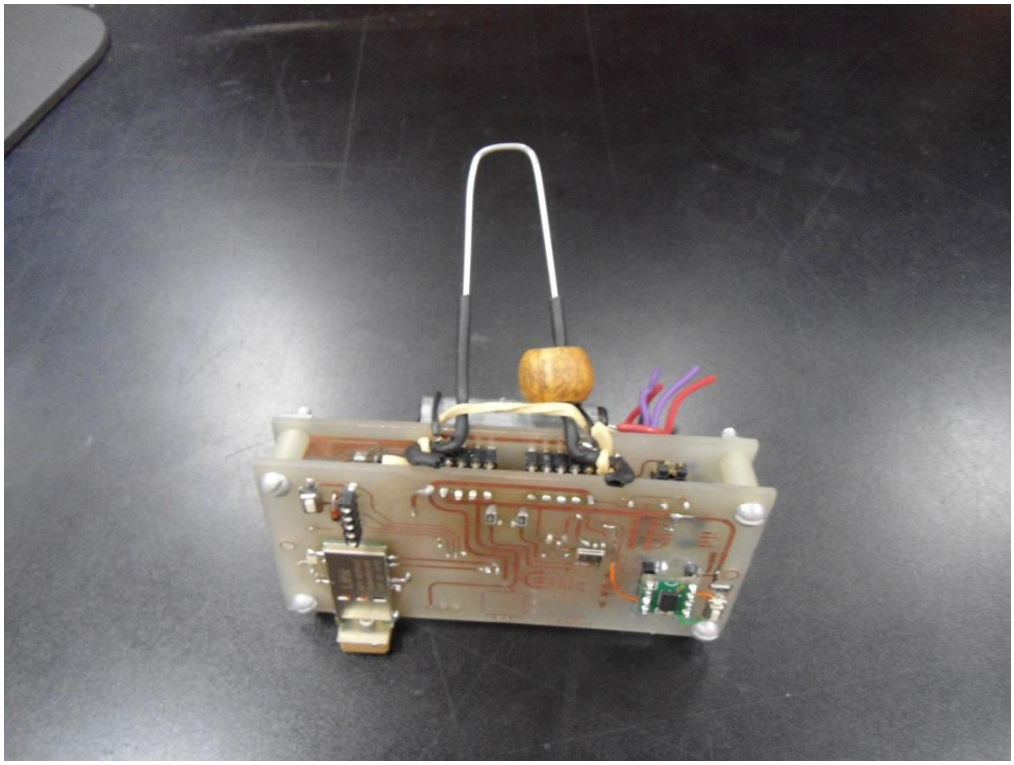


Figura 3.3 Prototipo de Estructura y PCBs del CanSat.

3.2.2 Subsistema de recuperación

El uso de un paracaídas asegura un aterrizaje suave y controlado. La tela de un paraguas es ideal para fabricar el paracaídas por ser un material ligero, fuerte, barato y tiene la forma adecuada.

El CanSat se une al paracaídas por medio de hilos de nylon anclados directamente al PCB. El paracaídas se dobla contra el cuerpo del CanSat y se despliega por gravedad al ser liberado del lanzador por lo tanto no se requiere de ningún mecanismo para desplegar el paracaídas.

3.2.3 Subsistema eléctrico

Diagrama de conexión de componentes

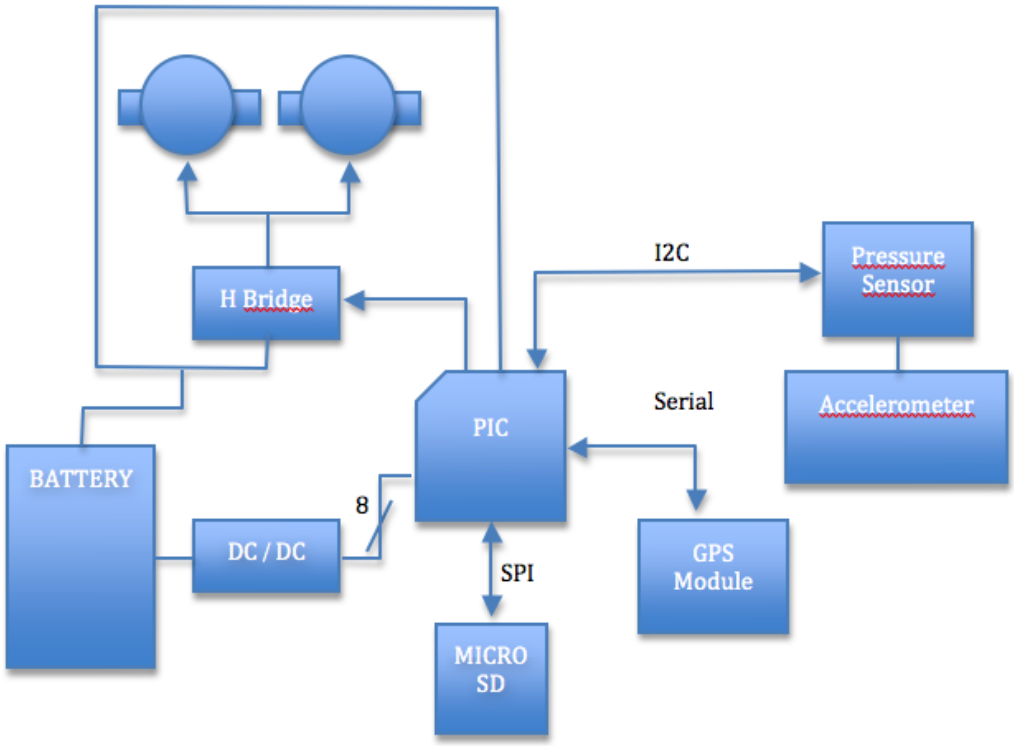


Figura 3.4 Primer diagrama del CanSat.

Principales componentes utilizados:

PIC18F4520-I/ML-ND	1	Microcontrolador
RXM-GPS-SR-B-ND	1	GPS
MPL115A2	1	Barómetro
MMA8453Q	1	Acelerómetro
Tarjeta Micro SD	1	Memoria
GH6123S-B	2	Motores Dc
1600 mAh	1	Batería

Otros componentes:

445-5178-1-ND	Cap 4.7u 6.3v
445-4113-1-ND	Cap 10u 6.3v
497-10150-1-ND	Level shifter
32-3354-1-ND	inductor

718-1403-1-ND	capacitor	
LTC3127EDD#PBF-ND	buck/boost	
PTN1206E3203BST1-ND	Res320k	1206
PTN1206E1823BST1-ND	Res182k	1206
PTN1206E4993BST1-ND	Res499k	1206
PTN1206E3242BST1-ND	Res32.4k	1206

3.2.4 Subsistema de comunicaciones

Esta versión de CanSat no tiene módulo de comunicaciones solamente incluye una ranura en el PCB para incluir el módulo en otro momento en caso de ser necesario.

3.3 Descripción de los componentes

PCBs

Los PCBs son la estructura principal del CanSat y sostienen todos los compnentes, tambien se encargan de llevar las señales eléctricas a todos los componentes. Los subsistemas están divididos en 2 PCBs que están diseñados para agregar más componentes de los que realmente se tienen como un sonar y un módulo de comunicación BlueTooth.

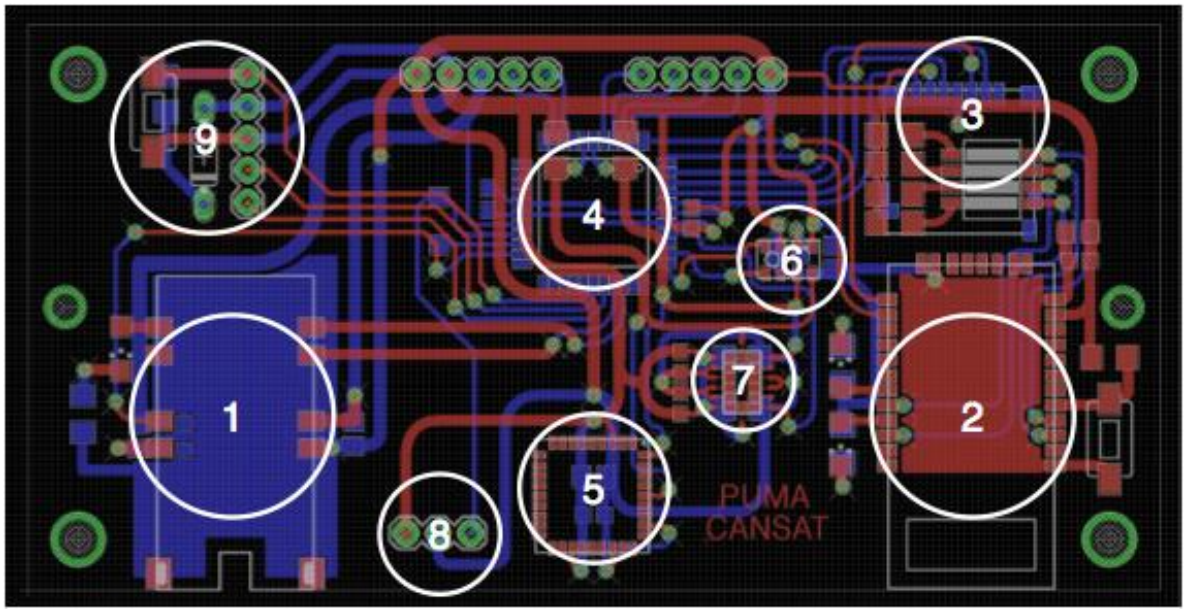


Figura 3.5 PCB superior del primer CanSat.

El PCB superior tiene los sensores, memoria y microcontrolador.

1. Modulo GPS Linx RXM-GPS-SR

Es necesario un módulo GPS para adquirir y encontrar las coordenadas de la meta y conocer la posición actual del CanSat. Este módulo tiene una antena frontal integrada, es muy pequeño, fácil de montar, preciso y fácil de usar.

2. Micro SD

Una tarjeta de memoria Micro SD es ideal para almacenar los datos obtenidos de los sensores por ser pequeña, con gran capacidad de almacenamiento y es removible. En esta tarjeta se almacenan todos los datos de los sensores y de navegación.

3. Microcontrolador PIC18F46J50

Este microcontrolador se encarga de la operación de todo el CanSat, del manejo y procesamiento de información, de encuestar los sensores. El CanSat requiere un microcontrolador con múltiples puertos de comunicaciones y diferentes protocolos.

Los protocolos de comunicación son los siguientes:

- I2C.- Comunicación con el acelerómetro de 3 ejes, sensor de presión y temperatura y compas de 3 ejes.
- USART para el GPS, Bluetooth y sonar.
- SPI para la tarjeta micro SD.

Es un MCU de 8 bits, baja potencia y tiene los puertos de comunicaciones necesarios, sus características son las siguientes:

- Frecuencia de operación 48 MHz
- Memoria 64 KB
- Puertos PWM 2
- Puertos USART 2
- Puertos Seriales(SPI,I2C) 2
- Timers 5
- ADC 13

4. Compas digital de 3 ejes HMC6343

Es una brújula digital que obtiene dirección e inclinación con un error de 2° en dirección y 1° en inclinación. Es parte fundamental del sistema de navegación del CanSat.

5. Sensor de presión y temperatura

Este sensor se usa para calcular la altitud del CanSat basado en la presión atmosférica, para que las líneas del paracaídas sean cortadas a una altura apropiada y garantizar un aterrizaje adecuado.

6. Acelerómetro

La información que se obtiene de este sensor sirve para determinar el estado de la misión, es posible saber si el CanSat está en la bahía de lanzamiento, si está en ascenso o descenso.

PCB Inferior

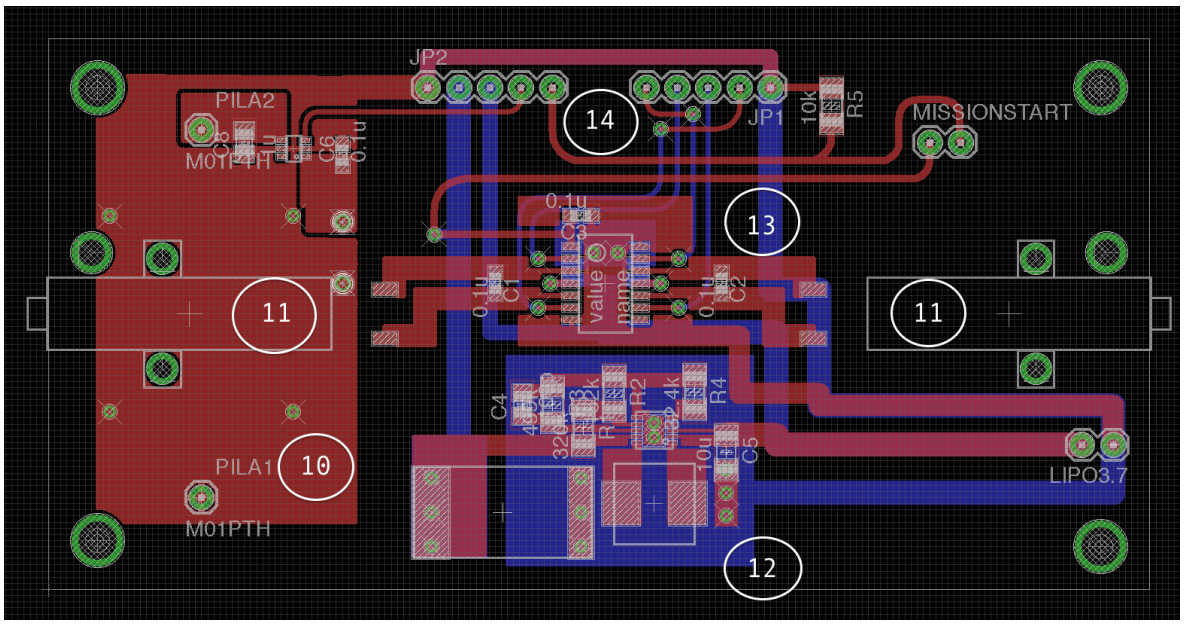


Figura 3.6 PCB inferior del primer CanSat.

10. Bateras de Li-ion

Se necesitan dos baterías de Li-polímero de 2000 mA/h. Estas baterías tienen gran capacidad y su tiempo de carga es corto; son capaces de proveer la energía necesaria a todos los subsistemas del CanSat.

11. Motores

Dos motores GH6123S-B mueven las ruedas y tienen las siguientes características:

- 6mm de diámetro
- Motor de engranes micro planetarios con eje en forma de D
- Proporción de engranes: 1:136
- Velocidad: 200 RPM
- Material: Fibra de vidrio reforzado con plástico
- Eje de bronce

12. Buck-Boost

Circuito convertidor de DC a DC, un convertidor “Buck” funciona de forma parecida a un regulador disminuyendo el voltaje a un nivel especifico pero tiene una mayor eficiencia, un convertidor “Boost” eleva el voltaje a un nivel especifico y tiene alta eficiencia, en el CanSat se utilizan dos, uno disminuye el voltaje a 3.3 V con el circuito integrado LTC3127 y otro lo aumenta a 5.0 V con el circuito integrado LTC3122.

13. Puente H

Un Circuito integrado que controla la dirección de giro de los motores.

3.4 Consumo de corriente estimada.

Componente	**Consumo máximo**
PIC18F4520	11 [µA]
GPS	60 [µA]
Convertidor	1 [A]
Barómetro	5 [µA]
Acelerómetro	165 [µA]

Motores (2)	80 [mA]
Circuito de alta potencia	0.5 [A]
TOTAL	**1.580246 A**

3.5 Estimación de la masa del CanSat

Subsistema Eléctrico

PIC18F46J50	1	Microcontrolador	1 gr
RXM-GPS-SR-B	1	GPS	2 gr
MPL115A2	1	Barómetro	0.5 gr
MMA8453Q	1	Acelerómetro	0.5 gr
Micro SD card	1	Memoria	0.5 gr
GH6123S-B	2	Motores DC	4.4 gr
800 mAh	2	Batería de celular	20 gr
950 mAh	1	Batería de Litio	9.2gr
	1	Otros	5 gr
		Aprox. Total	43.1 gr

Estructura

PCB	2	20 gr
Ruedas	2	80 gr
Otros	1	40 gr.
	Aprox. Total	140 gr

Subsistema de recuperación

Paracaídas	1	100 gr
Otros	1	20
	Aprox. Total	120 gr

Masa total aproximada del CanSat: 305 gr.

3.6 Software operativo

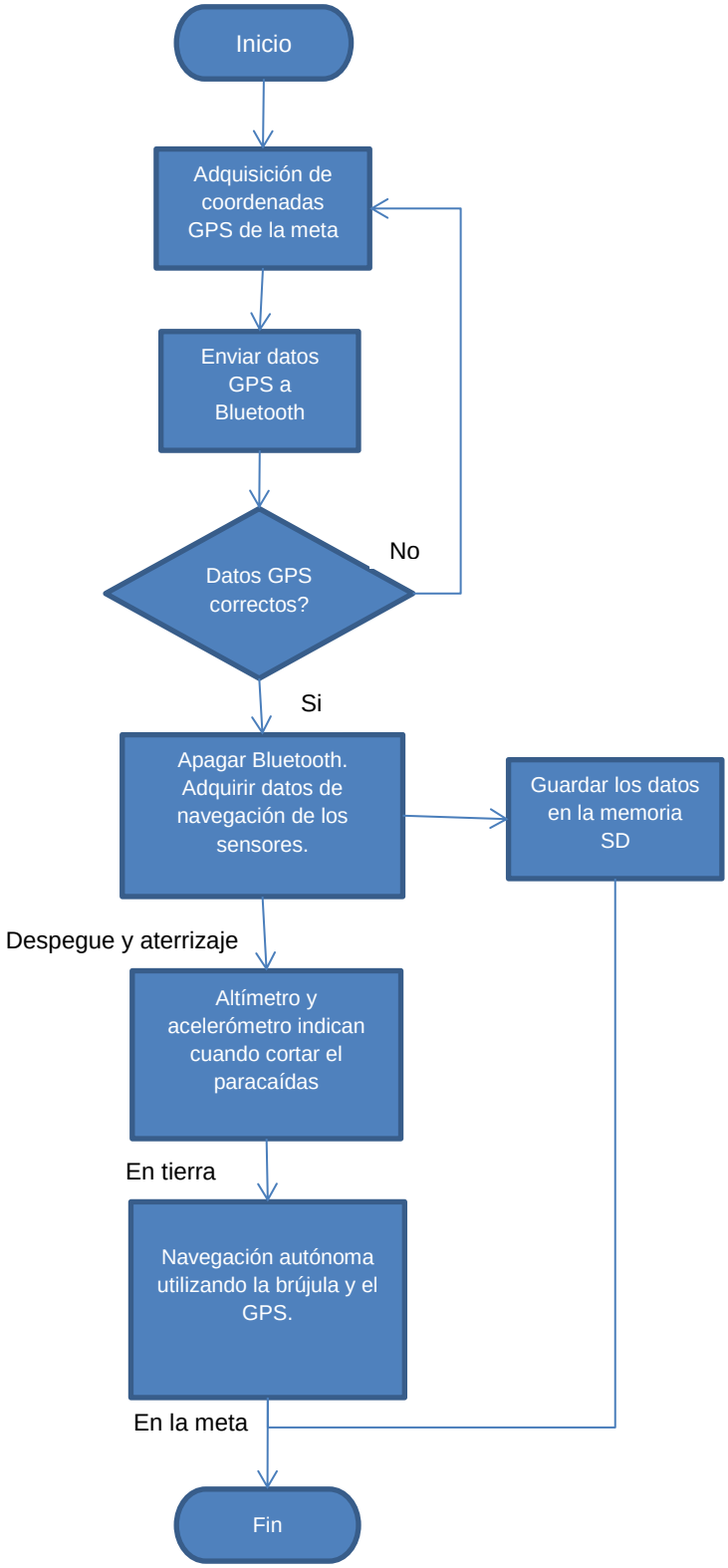


Figura 3.7 Primer software operativo del prototipo CanSat.

El primer diagrama de software operativo que se creo es muy sencillo, su único propósito es plantear el comportamiento básico del CanSat a lo largo de la misión.

Posteriormente se tomó la decisión de cambiar algunos elementos de Hardware como la computadora de vuelo por una más rápida y con mayores capacidades. Los cambios en hardware también generan cambios en software que se reflejan en el software operativo final.

El sistema toma información de navegación a un intervalo específico de tiempo y guarda los datos en una tarjeta de memoria micro SD, después de la misión es posible recuperar la tarjeta de memoria para analizar los datos adquiridos durante la misión:

- Posición
- Presión atmosférica
- Coordenadas GPS
- Datos de navegación
- Medidor de batería

4. Revisión Crítica de Diseño

4.1 Objetivos para una posible misión CanSat:

- Diseñar un CanSat suficientemente compacto y ligero que se asemeje a las dimensiones de una lata de refresco y pese menos de 350 gr.
- Disminuir el impacto de caída del CanSat para asegurar su correcto funcionamiento después del impacto.
- Incorporar sensores de presión y temperatura, GPS, brújula, acelerómetro, memoria extraíble y un sistema de comunicaciones.
- Adquirir y almacenar información de los sensores durante el vuelo de ascenso en el vehículo de lanzamiento y de descenso con el paracaídas.
- Lograr una comunicación exitosa con el CanSat y obtener datos en tiempo real.
- Lograr la navegación autónoma exitosa en el modo “rover-comeback”.
- Terminar la misión.

4.2 Operación del CanSat

La operación del CanSat se dividió en 6 etapas diferentes y se programaron 3 modos diferentes de operación.

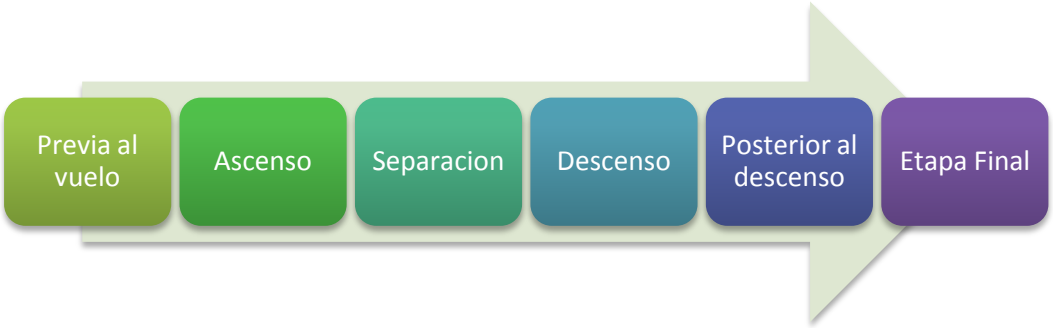


Figura 4.1 Etapas de la misión CanSat.

Etapa 1: Previa al vuelo

- Se realizan las últimas revisiones al CanSat y se programan las coordenadas de la meta.
- Se conecta la batería del CanSat y se inicializan los sistemas.
- Se coloca el CanSat en la bahía de carga del vehículo de lanzamiento.

Etapa 2: Ascenso

- El acelerómetro detecta el inicio del ascenso y lo registra.
- Envió de la información adquirida por los sensores de presión, temperatura, acelerómetro y GPS durante el vuelo de ascenso.

Etapa 3: Separación

- El acelerómetro detecta el cambio en el movimiento cuando llega al punto más alto y se registra la altura máxima aproximada por los sensores de presión y temperatura, u obtenida con la trama GPS.
- El paracaídas se despliega por acción de la gravedad.

Etapa 4: Descenso

- Continúa la transmisión de la información de los sensores.
- El acelerómetro detecta el impacto contra el suelo y lo registra.
- El cortador de hilo desprende el paracaídas del CanSat.

Etapa 5: Posterior al descenso

- Se apaga el transmisor y el sensor de presión y temperatura y se enciende la brújula electrónica.
- Inicia la navegación hasta el punto de llegada.

Etapa 6: Final

- Al llegar a la meta se apaga todo y se enciende el sistema de recuperación.

Los modos de operación cambian el funcionamiento del CanSat, son 3 modos diferentes que se activan conforme avanza el estado de la misión.

Cada modo activa y desactiva dispositivos del CanSat y puertos del DSP para hacer más eficiente el consumo de energía y el funcionamiento del CanSat, de

esta forma disminuye el riesgo de terminar la misión prematuramente a causa de falta de energía.

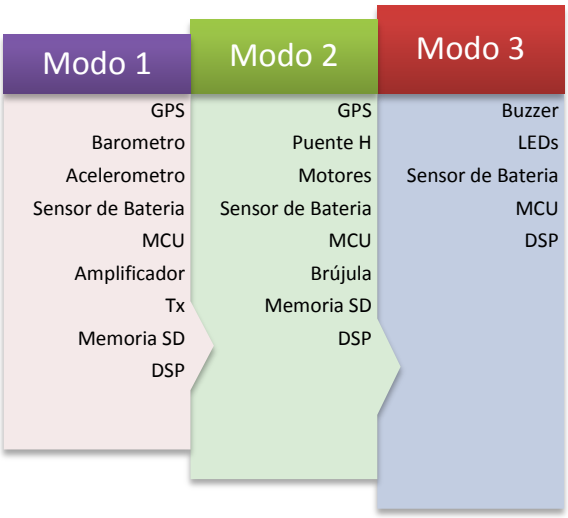


Figura 4.2 Modos de operación del CanSat.

El Modo 1 se encuentra activado durante las etapas 1 a 4, el Modo 2 se activa al iniciar la etapa 5 y el Modo 3 se activa al iniciar la etapa 6.

El sensor de batería, el controlador del subsistema de potencia, la computadora de vuelo y la memoria siempre están encendidos.

El primer modo de operación se activa al encender el CanSat y enciende el GPS, barómetro, acelerómetro y transmisor durante el vuelo de ascenso y descenso hasta llegar a tierra.

Una vez que el CanSat se encuentra en tierra el segundo modo de operación entra en funcionamiento, se encienden los motores y la brújula para iniciar la navegación autónoma. Se apaga el acelerómetro y el transmisor, en esta tesis se asume que el transmisor solo es necesario para transmitir información sobre el vuelo de ascenso y descenso, una vez en tierra la información de los sensores debería estabilizarse. Otra opción es que permanezca encendido el transmisor para rastrear la posición del CanSat en tiempo real.

El tercer modo de operación se inicia una vez que el CanSat llega a la meta o cuando la batería está demasiado baja para continuar navegando, se apagan todos los sensores y se inicia el subsistema de recuperación, se encienden luces LED color rojo que parpadean y un buzzer que emite un zumbido a 75 dB.

4.3 Presupuesto de potencia

El presupuesto de potencia calcula el consumo de corriente del CanSat cuando todos los dispositivos están encendidos, esto es solamente un supuesto.

	Min	Typ	Max	
GPS	-	31	49	mA
Puente H	-	22	35	mA
Motor (x2)	90	-	260	mA
Barómetro	-	5	6	µA
Acelerómetro	24	-	165	µA
Sensor de Bat	-	65	105	µA
MCU	-	4.1	9	mA
Brújula	3.5	4.5	5.5	mA
Amplificador	41	45	49	mA
Tx	5.5	7	8.5	mA
Memoria SD	-	100	100	mA
DSP	-	108	128	mA
Buzzer	-	100	-	mA
LED (x4)	-	70	-	mA
	925.525	1367.745	1600.776	mA

Es necesario mencionar que en ningún momento de la misión se van a encontrar encendidos todos los dispositivos del CanSat, cada modo de operación enciende y apaga los dispositivos necesarios por lo tanto el consumo de energía por modo de operación queda de la siguiente forma:

Modo de operación “Fase1”

	Typ	Max	
GPS	31	49	mA
Barómetro	5	6	µA
Acelerómetro	165	165	µA
Sensor de bat	65	105	µA
MCU	4.1	9	mA
Amplificador	45	49	mA
Tx	7	8.5	mA
Memoria SD	100	100	mA
DSP	108	128	mA
	295.335	334.876	mA

Modo de operación “Fase2”

	Typ	Max	
GPS	31	49	mA
Puente H	22	35	mA
Motor (x2)	260	260	mA
Sensor de bat	65	105	µA
MCU	4.1	9	mA
Brújula	4.5	5.5	mA
Memoria SD	100	100	mA
DSP	108	128	mA
	789.665	846.605	mA

Modo de Operación “Fase3”

	Typ	Max	
Buzzer	100	100	mA
LED (x4)	70	70	mA
Sensor de bat	65	105	µA
MCU	4.1	9	mA
DSP	108	128	mA
	492.165	517.105	mA

4.4 Diagrama de flujo de la misión

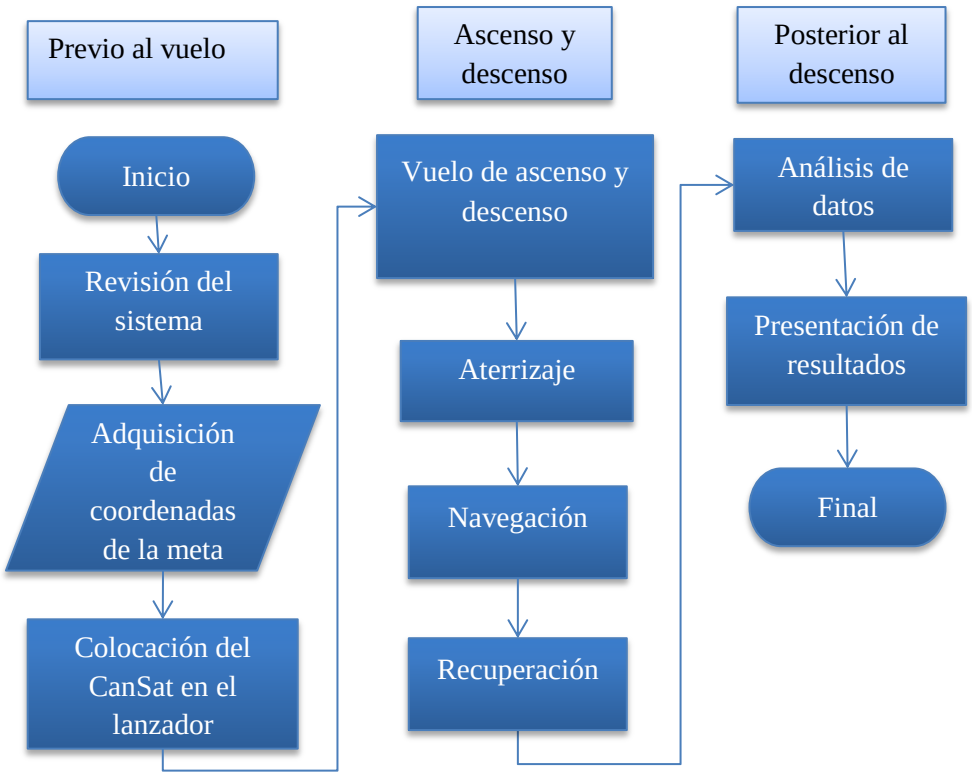


Figura 4.3 Diagrama de la misión CanSat.

4.5 Ruedas y estructura

La estructura del CanSat se compone por las PCBs mismas, el conjunto de PCBs, que en total son dos, tienen la rigidez necesaria para soportar el impacto contra el suelo al momento del descenso además de ser muy ligeras. Las PCB son de 1.6 mm de espesor de fibra de vidrio (FR4).

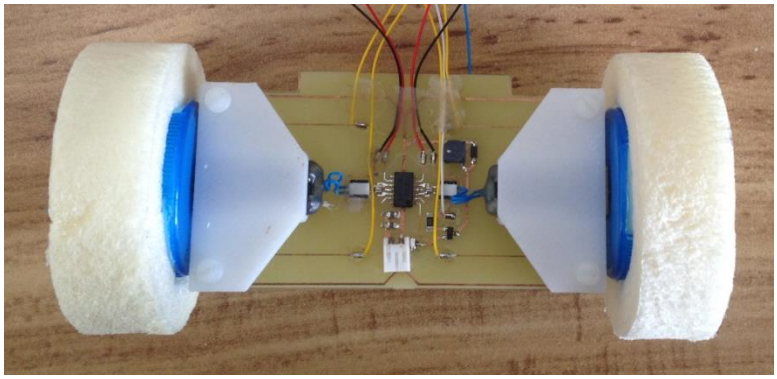


Figura 4.4 PCB inferior con ruedas.

La PCB inferior aloja el subsistema de potencia junto con los motores. Los motores están fijos a la tarjeta por un soporte anclado a las esquinas diseñado específicamente para este propósito, este soporte también se atornilla a las esquinas del PCB superior.

Los soportes además de sostener los motores agregan rigidez a la estructura completa.

Figura 4.5 Ruedas del CanSat.

Para hacer las ruedas se utilizó una lata de refresco vacia con la parte superior recortada como molde, luego se vertió la espuma de poliuretano dentro de la lata.

Cuando se seca la espuma, se corta la lata y obtenemos un cilindro de espuma que se corta en rebanadas de 2 cm. La llanta se une al eje del motor a través de una tapa de botella recortada para anclarse a la llanta con un agujero en el centro para acoplarse al eje.

Finalmente se recubre la cara de la llanta con una capa delgada de silicón para que la llanta no se deslice en el suelo.

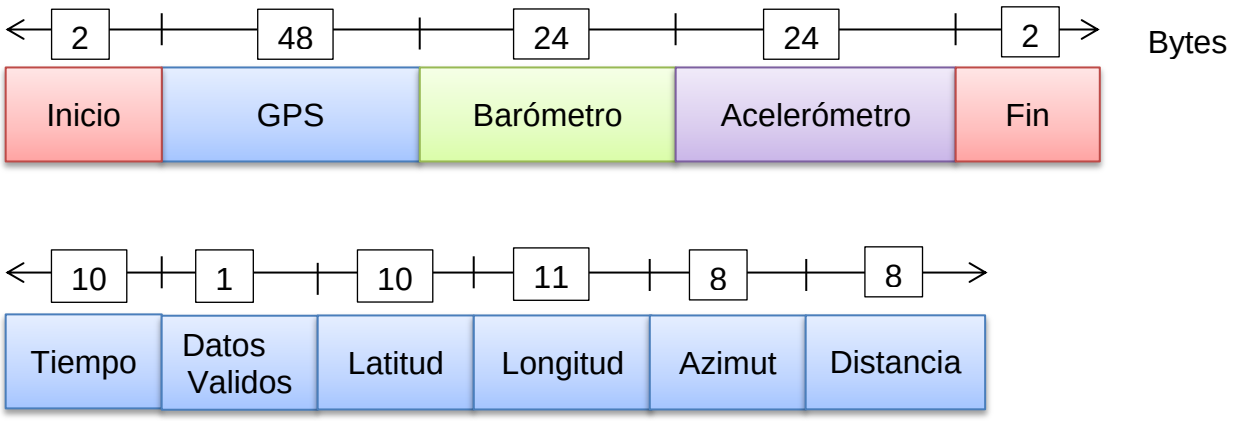
Los motores se cambiaron por motores más pequeños y ligeros modelo KG12T300 que tienen las siguientes características [20]:

Voltaje de operación	1.7 – 5.0 VDC
Velocidad sin carga	300 rpm
Corriente sin carga	90 mA
Corriente con carga	260 mA max
Torque	144 g – cm
Gear ratio	17:1
Longitud del eje	4.5 mm
Diametro	3.0 mm
Dimensiones	12mm x 10mm x 24mm
Peso	8.2 gr

4.6 Sistema de comunicaciones

Se cambió el sistema de comunicaciones Bluetooth que es de corto alcance por un sistema de comunicaciones digital de largo alcance con modulación FSK con frecuencia central en 916 MHz y potencia de transmisión de 31.6 mW o 15 dBm que enviará información a la estación terrena continuamente durante el vuelo.

La trama de datos que se envía consta de 100 Bytes y tarda 20.8ms a 38,400 bps:



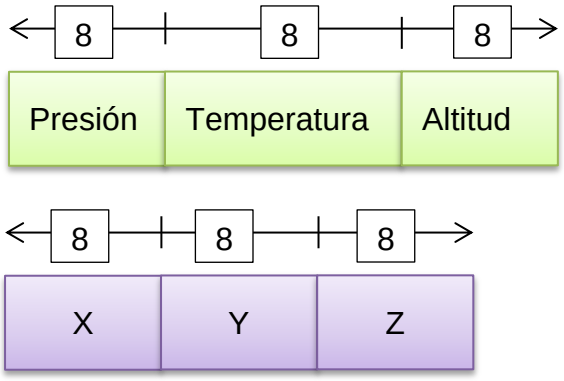


Figura 4.6 Trama de datos del CanSat.

4.7 Computadora de vuelo

La computadora de vuelo se cambió por el TMS320F28027 porque es más rápido y tiene la plataforma de desarrollo "LaunchPad" que facilita la creación de software. También se utilizó el micro controlador ATmega88PA para controlar la distribución de energía en el CanSat.

4.8 Costo

El costo del CanSat es de aproximadamente $ 4,000.00 desglosado en la siguiente tabla:

Artículo	Precio (MN)
TMS320F28027	91
ATmega88PA	32
Regulador "Buck-Boost" x 2	115
Antena ¼ λ	74
Tx	175
Rx	234
Antena Log Periódica	624
Sensor de batería	78
Conectores	72
Barómetro	35
Amplificador	29
Brújula	1,157
Memoria SD	52
GPS	598
Corta hilo	78
Cargador de batería	59
Acelerómetro	20
Otros	477
Total	**4,000**

4.8 Software operativo

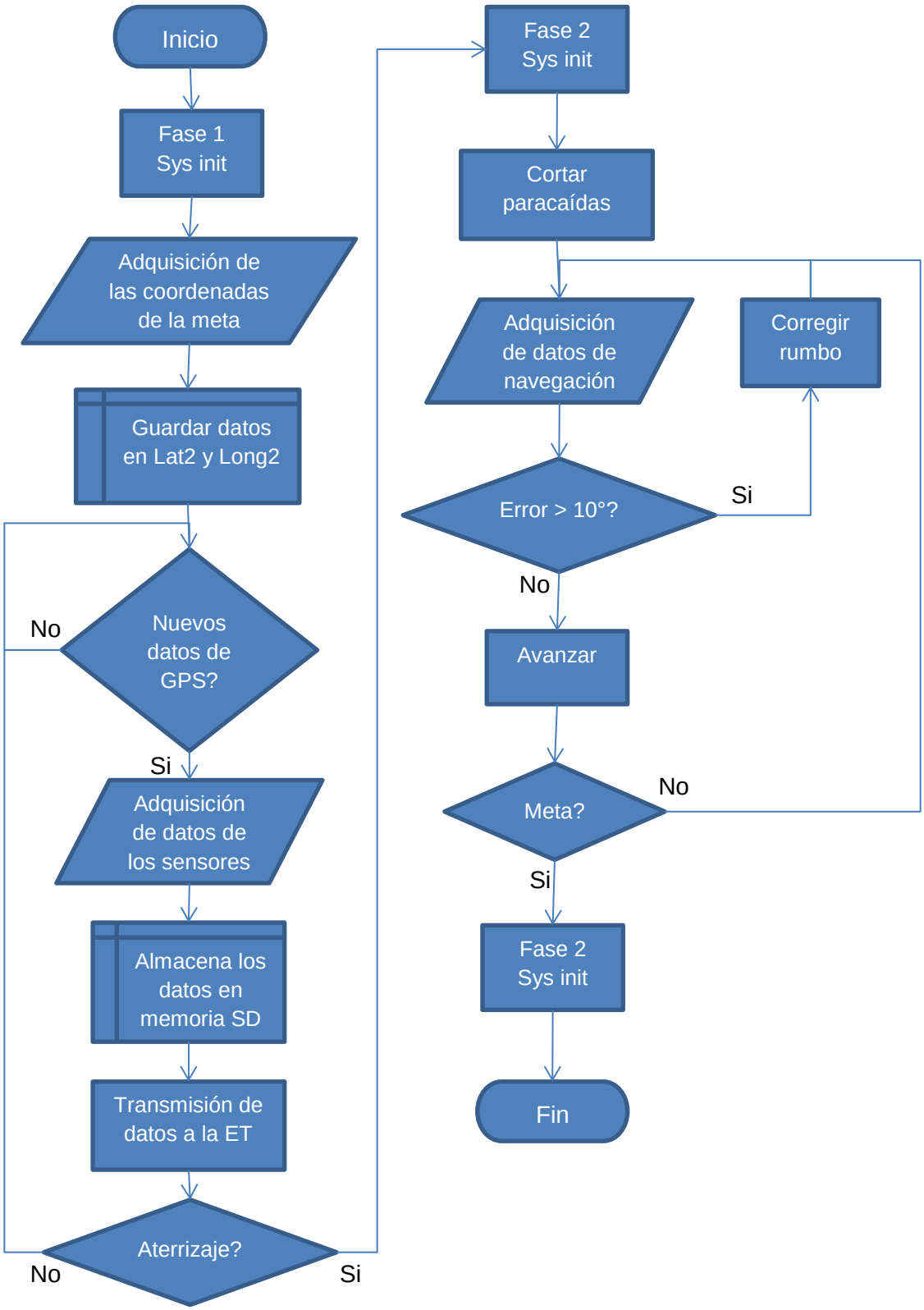


Figura 4.7 Diagrama del software operativo del CanSat.

5. Propuesta de los medios de lanzamiento para satélites CanSat

Una vez que los estudiantes desarrollan un prototipo CanSat, lo pueden validar también de una forma muy sencilla: lanzándolos desde lo alto de edificios con paracaídas, lanzándolos desde globos aerostáticos, lanzándolos desde cohetes sonda o desde un Drone.

5.1 Descripción del lanzamiento con un cohete sonda

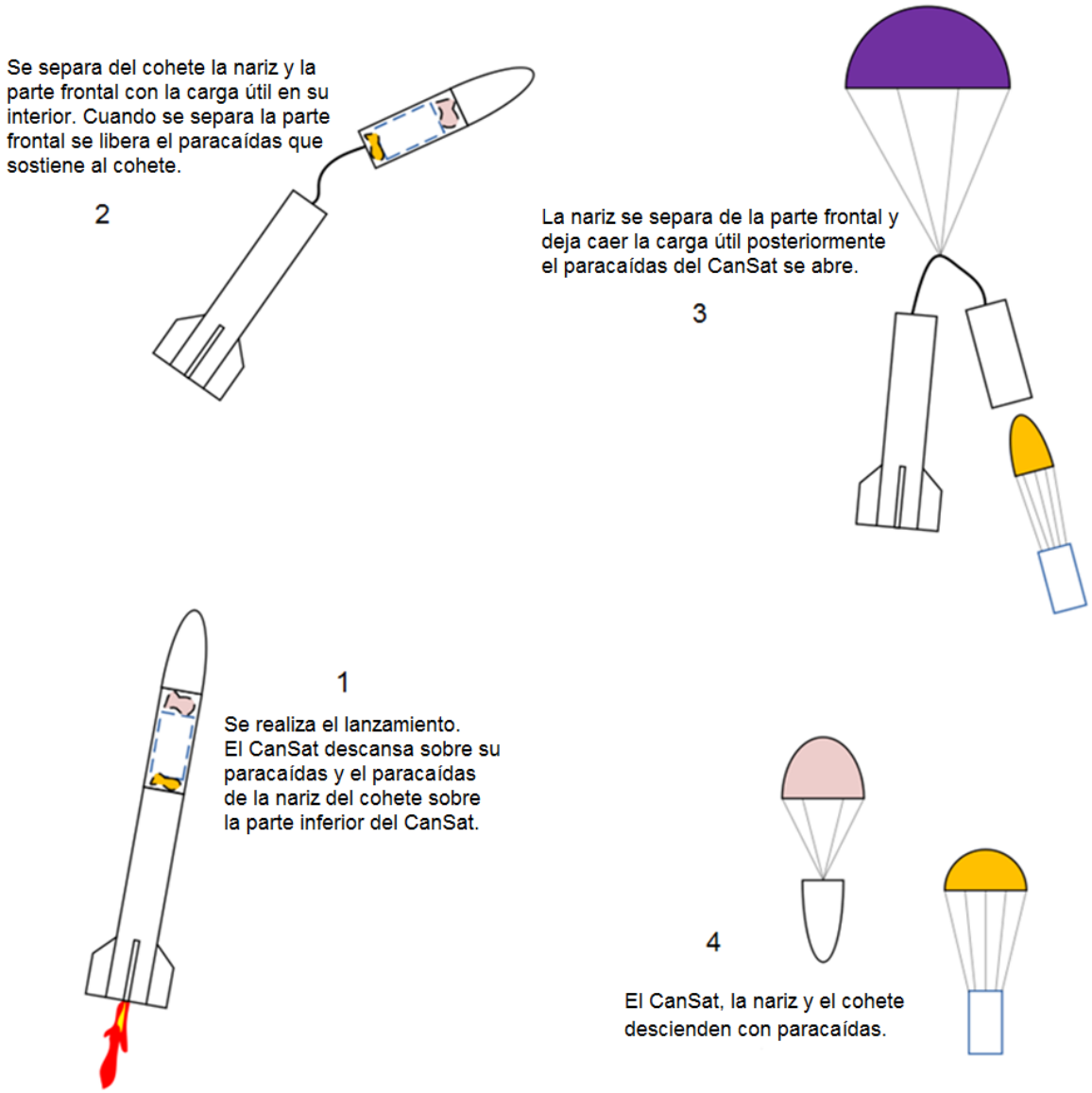


Figura 5.1. Esquema de lanzamiento de un CanSat con un cohete sonda.

Los cohetes con propósitos educativos se construyen con materiales de bajo costo, vuelan a velocidades relativamente bajas comparados con los cohetes reales y permiten a los estudiantes poner en práctica conocimientos teóricos en estructuras, sistema de despegue, propulsión, paracaídas, instrumentación científica y aerodinámica.

La figura 3.1 muestra un lanzamiento y su secuencia de separación; la secuencia de separación podría diferir dependiendo del modelo del cohete; dada la secuencia de separación de un cohete es importante tomar en cuenta la colocación del CanSat en su interior, si tomamos como ejemplo el caso mostrado en la figura 3.1, el CanSat debe colocarse en posición invertida descansando sobre el paracaídas, de tal forma que cuando la punta se separe y deje caer el CanSat el paracaídas quede en la parte superior y pueda desplegarse lo más rápido posible, de lo contrario el descenso podría ser diferente a lo planeado y ocasionar un funcionamiento imprevisto en la operación del CanSat.

5.2 Descripción del lanzamiento con un globo aerostático

Los globos son usados para observaciones científicas y para el desarrollo de tecnología espacial pudiendo alcanzar alturas de hasta 30 Km, en los vuelos CanSat se elevan a distancias cortas con el fin de controlar su posición y mantenerlos anclados al suelo con líneas de control.

Decenas de vuelos son realizados anualmente por las principales agencias espaciales para aplicaciones científicas y tecnológicas con cargas útiles de kilogramos y hasta toneladas.

Las ventajas de los globos científicos para fines didácticos son el corto tiempo de implementación, versatilidad para transportar cargas de gramos hasta varios kilos, tiempos de vuelo de minutos a horas, etc.

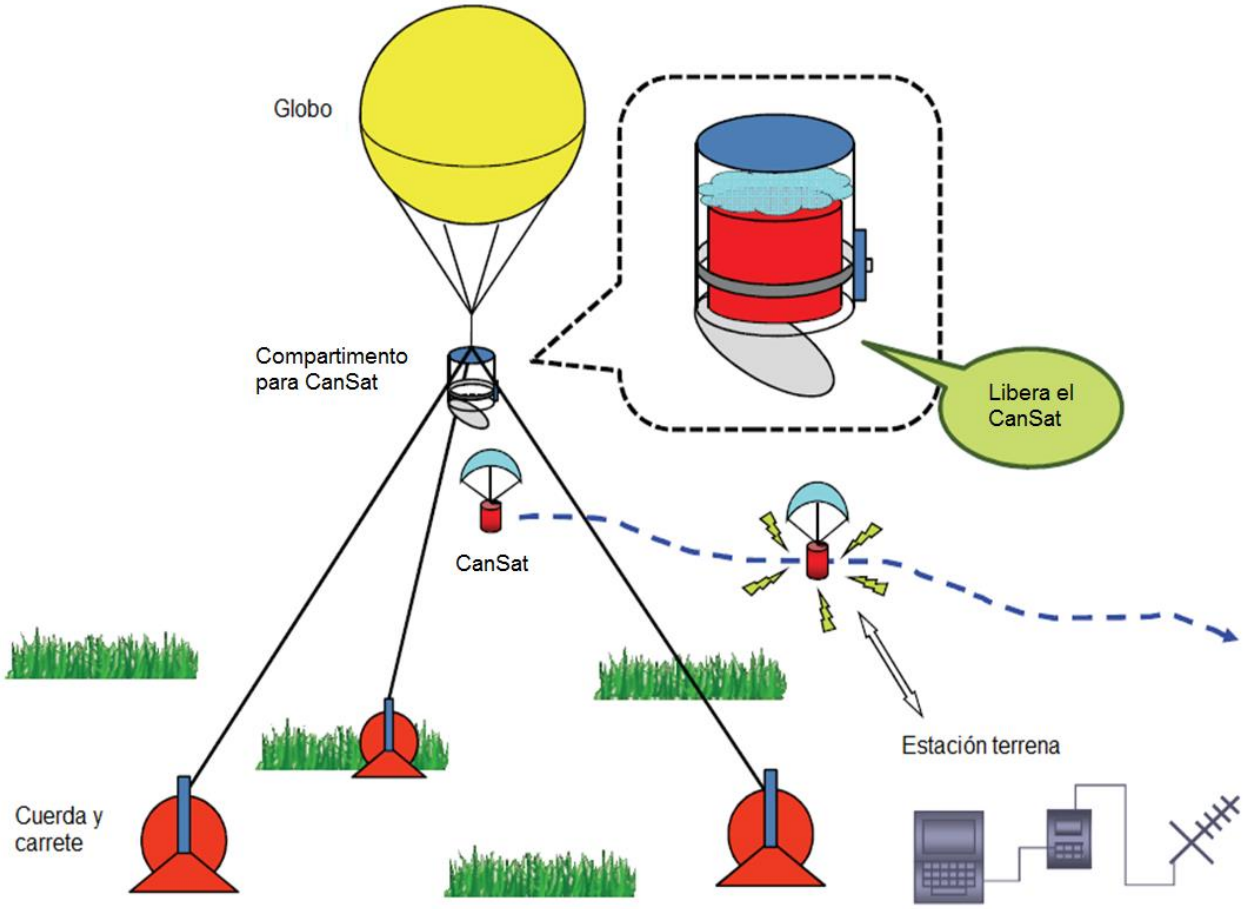


Figura 5.2. Esquema de lanzamiento de CanSat con globo.

La figura 5.2 muestra un típico lanzamiento de CanSat en globo, el sistema está compuesto por un globo que sujeta un compartimento para albergar el CanSat durante su ascenso, el globo a su vez está sujeto al suelo por cuerdas, es posible sujetarlo a una sola cuerda pero el globo se vuelve inestable en presencia de viento, al tener sujeción con dos o tres cuerdas es más sencillo tener control del ascenso del globo y su ubicación final al liberar el CanSat.

El globo puede ser de diferentes tamaños y materiales, los globos meteorológicos son globos de Neopreno o de caucho sintético, que es un polímero altamente resistente a humedad, bajas temperaturas, ozono y radiaciones ultravioleta, además de ser elástico. El globo debidamente inflado con Hidrógeno, llega a medir hasta 150 centímetros de diámetro máximo y es capaz de levantar un peso de 450 gramos. [13]

Los globos aerostáticos actuales están hechos de material sintético impermeable como nylon o poliéster, que resisten perfectamente a las altas temperaturas del aire caliente del interior.

Los globos que se utilizan como espectaculares están hechos de PVC, son más pesados, resistentes al clima y su tiempo de vida es mayor, la desventaja es que por ser más pesado se eleva menos que los otros dos tipos de globo.

5.3 Descripción del lanzamiento con un Drone.

Las aeronaves no tripuladas llamadas Drones han sido ampliamente utilizadas en la actualidad debido a su versatilidad, confiabilidad y facilidad de uso. Debido a su gran autonomía son utilizados en paquetería, vigilancia, filmografía y vuelos CanSat.

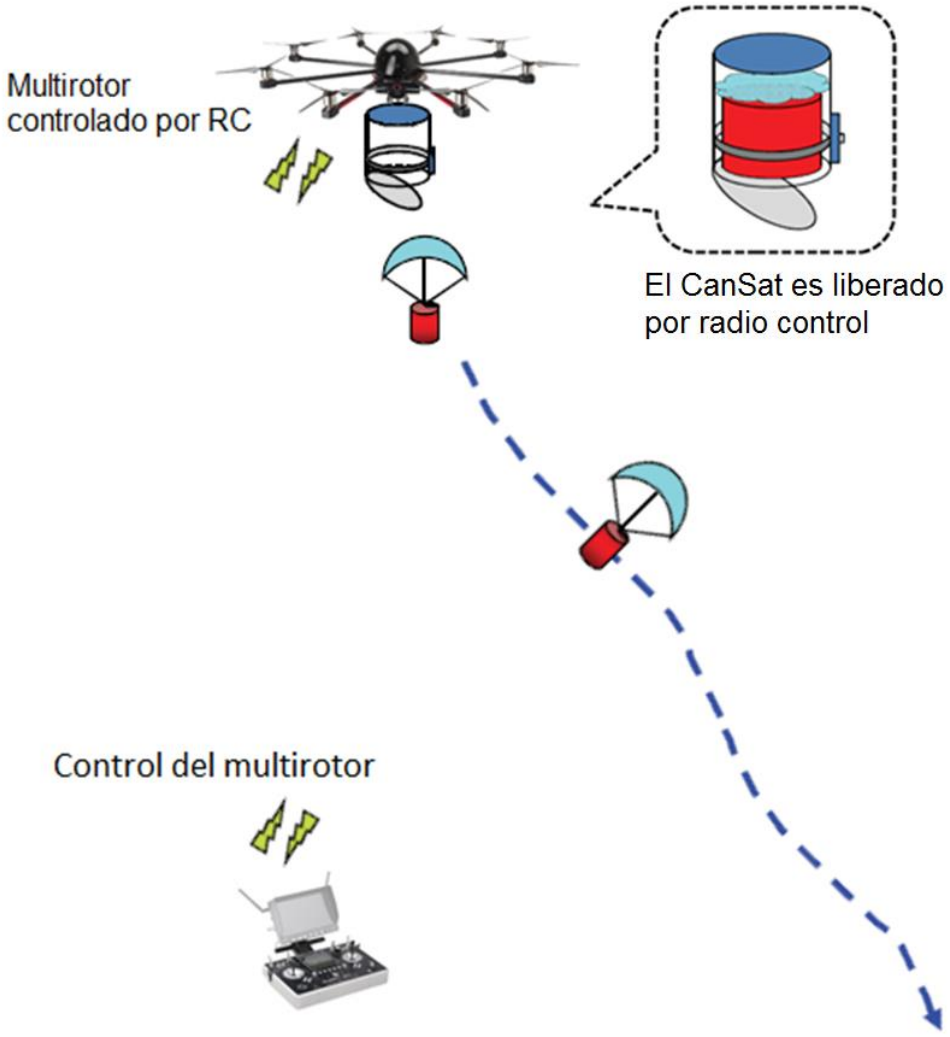


Figura 5.3 Esquema de lanzamiento de CanSat con drone.

Un Drone puede ser cualquier aeronave no tripulada que navegue autónomamente o controlada a distancia. Los Drones pequeños comúnmente son algún tipo helicóptero o multirotor. Un multirotor es una aeronave con más de dos rotores donde el control de la aeronave es alcanzado mediante la variación de la velocidad relativa de cada rotor para cambiar el empuje y el torque de cada uno.

Debido a la facilidad de construcción de los multirotores, estas aeronaves son frecuentemente utilizadas en aeronaves de radiocontrol que comúnmente tienen 4, 6 u 8 rotores y se utilizan ampliamente en proyectos de bajo presupuesto.

5.4 Recomendaciones de sistemas de lanzamiento CanSat

Los estudios en cohetería también pueden involucrarse en vuelos CanSat por esta razón es que muchas universidades del mundo utilizan este método como su primera opción. En caso de no contar con fácil acceso a la cohetería necesaria para lanzamientos CanSat, la mejor opción es la adquisición de un Drone.

El cohete es la mejor opción para vuelos de gran altura, es el método que puede llevar un CanSat no solo a mayor altura sino también en el menor tiempo; sin embargo, es necesario comprar combustible para cada lanzamiento, también es recomendable tener múltiples cohetes por precaución y para minimizar el tiempo entre lanzamientos, tambien se pueden conseguir cohetes que son capaces de llevar 6 CanSats a más de un kilómetro de altura. [14]

Los lanzamientos con multirotor son poco frecuentes pero su funcionamiento es muy eficiente y sencillo. La adquisición del equipo necesario puede ser costosa pero no requiere muchos complementos para utilizarse y puede alcanzar alturas de 2Km. Es muy compacto para transportar y los modelos economicos pueden costar $300 USD. [4]

Los globos pueden ser una forma más sencilla y menos costosa de realizar lanzamientos CanSat, sin embargo, la altura del lanzamiento podría estar limitada por varios factores; primero, no todos los globos vuelan a gran altura; segundo, los globos pueden dañarse, poncharse, romperse fácilmente; tercero, se requiere transportar suficientes tanques de helio al lugar del lanzamiento; cuarto, se requiere por lo menos una cuerda para anclar el globo al suelo de mayor longitud de la que se desea elevar el globo; quinto, es un sistema muy voluminoso para transportar.

6. Propuesta de una metodología para establecer una competencia nacional CanSat en México

Aquí se plantea una propuesta de proyecto semilla para iniciar un programa de capacitación a nivel licenciatura y posgrado mediante satélites pequeños CanSat.

Se propone realizar una estrategia que nos lleve a implantar un mecanismo continuo de capacitación de estudiantes de ingeniería (licenciatura y posgrado) de alto nivel, que integre tecnologías prioritarias que potencialicen y complementen el proceso de formación profesional de carreras de ingeniería (Aeroespacial, Aeronáutica, Eléctrica, Electrónica, Computación, Mecánica y Mecatrónica, entre otras).

También se persigue que este proceso de complementariedad educativa conlleve una forma de trabajo y de entrenamiento práctico sobre la marcha, incorporando a los estudiantes en el desarrollo de prototipos de alta complejidad, pero que al mismo tiempo sean accesibles (económicos) para que se tornen factibles y atractivos tanto para las instituciones como para los estudiantes. Se persigue también que esta experiencia sea completa y abarque trabajo en equipo desde el diseño, construcción, validación en laboratorio, pruebas experimentales en campo, participación en competencias nacionales e internacionales que permitan ofrecer una experiencia formativa y atractiva a la juventud, de tal forma que ésta acompañe a los estudiantes en toda su vida profesional.

Una parte muy importante de esta propuesta es implementar la fase inicial del programa de capacitación de un grupo de ingenieros de alto nivel en el campo Aeroespacial de la ingeniería que permita después en su segunda fase un programa de capacitación nacional de recursos humanos en tecnología CanSat. Que permita a su vez incrementar la cantidad de promotores CanSat en Universidades de diversas regiones del país que se incorporen a dicho programa.

El objetivo de esta propuesta es implementar la fase inicial del programa en el que se capacitará a un grupo de ingenieros, los cuales serán fundamentales para proseguir la segunda fase del programa que extenderá sus beneficios a Universidades de todo el país, iniciando con aquellas que tengan programas o actividades en Aeronáutica y Espacio, pero que será extensivo sin problemas a

Universidades que ofrezcan carreras de Ingeniería, en sus diversas carreras relacionadas con Tecnologías de la Información, Mecánica y similares.

La segunda fase de este proyecto llevará a entrenar a profesores de diversas universidades en tecnología CanSat y posteriormente en el mismo año se instalará la primera competencia Nacional de desarrolladores de satélites pequeños CanSat.

La segunda fase del proyecto (año siguiente) será un mecanismo formal y atractivo para la juventud ingenieril del país, que por un lado generará mejores recursos humanos en el sector Aeroespacial en México y por otro, contribuirá a ofrecer mejores recursos humanos a los programas de posgrado en el país (en campos como: Aeronáutica, Aeroespacial, Electrónica, Telecomunicaciones, Computación, Sistemas de Información Geográfica, Percepción Remota, Materiales, Mecánica, Robótica, Telemática, etcétera).

Es importante señalar que esta capacitación formará también recursos humanos de alta calidad que podrán incorporarse a sectores productivos de otras industrias, como: química, alimentos, automotriz, servicios, agroindustria, monitoreo atmosférico, etcétera.

6.1 Programa de capacitación a nivel licenciatura y posgrado en el campo aeroespacial con satélites CanSat

Las características de los CanSats los hacen ideales para dar una mejor calidad de enseñanza a estudiantes de Ingeniería en México, a costos sumamente bajos y con resultados de gran pertinencia nacional e internacional.

En este documento se presenta la propuesta para entrenar en equipo a un grupo de ingenieros (fase 1). Este equipo de trabajo desarrollará teórica y experimentalmente, prototipos de CanSats totalmente funcionales y calificados para competir internacionalmente. Con este proceso de trabajo y entrenamiento sobre la marcha generaremos al grupo de trabajo que nos permitirá después capacitar a profesores de Universidades de todo el país. Esta fase de trabajo será la fase consecutiva de este proyecto (fase 2). Una vez entrenados a profesores de Universidades, éstos repetirán el curso en sus universidades, dirigiéndolo a estudiantes tanto de Ingeniería como de posgrado. Con lo cual se extenderán ampliamente los resultados de este proyecto semilla.

Tanto en la fase 1 como en la fase 2 de este proyecto se tendrá el apoyo de una red que reúne a instituciones educativas en Japón, con quien el tutor de esta tesis tiene interacción y colaboración desde hace 3 años. De igual forma tiene contactos y apoyos de las Universidades de Wakayama y Tokyo en Japón y de Naciones

Unidas en materia de iniciativas de tecnología espacial básica. La red de Japón se llama UNISEC (University Space Engineering Consortium), quienes están invirtiendo muchos esfuerzos y dinero para promover estas tecnologías y formas de capacitación estudiantil por medio de proyectos espaciales, iniciando con los CANSAT. De igual forma hay otras redes muy activas en el campo CANSAT en otros países como EU, Francia, España y Turquía. Con algunas de ellas ya tenemos cierto contacto, pero precisamente por medio de este proyecto extenderemos aún más estos lazos de cooperación y generaremos nuevas interrelaciones que promuevan y vigoricen la iniciativa CANSAT en México.

Este último punto se alcanza al considerar que los mejores estudiantes nacionales podrán participar en competencias internacionales de CanSats, quienes promoverán la interacción con pares internacionales, y además, se incrementará la vinculación para formar los futuros líderes de nuestro país, así como se coadyuvará la participación Mexicana en otros proyectos o iniciativas Aeroespaciales internacionales.

La capacitación Nacional en CanSats que se ha indicado es extensiva a estudiantes de maestría de diversas áreas. Con estos estudiantes se tiene la ventaja de que los programas actuales de maestría consideran y promueven la movilidad, por lo cual el campo CANSAT les será también propicio para trabajar en equipos Mexicanos, pero también en equipos Internacionales para generar mayor vinculación y acentuar el trabajo y los beneficios de la redes tanto nacionales como internacionales.

6.2 Programa nacional de entrenamiento de recursos humanos en el campo aeroespacial

Bajo el escenario descrito en párrafos anteriores se impulsará en México un programa Nacional de capacitación de recursos humanos en el sector Aeroespacial. Un programa en donde expertos entrenarán teórica y prácticamente a profesores e investigadores del país. Para que éstos a su vez entrenen posteriormente a muchos estudiantes en sus Universidades. De esta forma, se procederá a crear entonces una competencia Nacional de proyectos CanSat, con estudiantes de todo el país, en donde a los ganadores se les apoyará para competir en el extranjero representando a México. Este programa deberá ser promovido por CONACYT, la Agencia Espacial Mexicana, la Sociedad Mexicana de Ciencia y Tecnología Aeroespacial (SOMECYTA), la UNAM, el IPN, etcétera, incluyendo además a empresas y asociaciones del sector privado.

Sin embargo, para llegar al anterior escenario, se requiere una masa crítica de expertos en Tecnología CanSat, un grupo bien preparado para impartir en México el programa de capacitación CanSat a profesores de Universidades con programas y/o intereses en el campo Aeroespacial.

Siendo uno de los expertos el tutor de esta tesis, quien fue entrenado en Febrero-Marzo de 2011 en Japón bajo el "Programa de Entrenamiento de Líderes en Tecnología CanSat" en un grupo de 12 personas de 10 países diferentes. En el curso se formaron 3 grupos de 4 personas y cada equipo desarrolló completamente 2 CanSats que fueron lanzados primeramente desde lo alto de edificios, posteriormente desde globos aerostáticos y después desde cohetes sonda. El programa es promovido por el gobierno Japonés y su objetivo es que las personas entrenadas regresen a sus países y repitan el entrenamiento que recibieron en Japón para promover las actividades espaciales a partir de satélites CanSat.

6.3 Atractivo financiero de capacitar recursos humanos con la tecnología CanSat

Un atractivo de los satélites CanSat, es que se pueden fabricar desde los 500 Usd hasta los 1,300 Usd en promedio. Lo cual los hace sumamente atractivos de desarrollar incluso para grupos de estudiantes entusiastas. Respecto a los lanzamientos en globos y cohetes sonda, éstos se pueden ofrecer gratuitamente a los estudiantes por parte de las Universidades con inversiones promedio de 4,000 Usd (para globos aerostáticos y cohetes sonda, ambos reutilizables) y consumibles de 200 Usd por lanzamiento. Lo cual lo hace sumamente atractivo para muchas Universidades Mexicanas.

6.4 Justificación de participación en competencia internacional de satélites CanSat

La estrategia que se expone en esta propuesta como fase 1 de trabajo consiste en capacitar una masa suficiente de expertos Mexicanos para que contribuyan después a impartir La Capacitación CanSat a nivel Nacional.

La fase 1 también busca ampliar la red de trabajo en satélites CanSat.

Como se ha señalado anteriormente, está participación de México a nivel internacional abrirá las puertas para extender nuestra red de contactos en el campo Aeroespacial, con las Universidades y los países participantes.

6.5 Plan para extender los beneficios a otras universidades de México con el apoyo de REDCYTE y otras instituciones

Respecto a la capacitación en Tecnología CanSat del equipo Mexicano que después capacitará a Universidades de todo México se propone realizarla por medio de su participación en una competencia internacional. Para tal propósito, se construirán 2 tipos de CanSat por parte de un Equipo Mexicano formado tanto por Asesores expertos como por Estudiantes de Ingeniería de la UNAM y del IPN quienes fabricarán y validarán los prototipos CanSat para competir.

Con este entrenamiento sobre la marcha se capacitará al grupo de ingenieros que posteriormente CAPACITARÁ a Universidades de México para extender los beneficios de la Tecnología CanSat.

Con el escenario expuesto anteriormente, esta propuesta en su fase inicial solo requiere recursos para:

- Capacitar a 12 personas en Tecnología CanSat (un equipo de trabajo con ingenieros de la UNAM y del IPN),
- Hacerlas participes de una competencia Internacional CanSat,
- Incrementar los lazos de interacción (red) con equipos de trabajo Aeroespacial de diversos países,
- Tener por vez primera la participación de Mexicanos en una Competencia CanSat internacional,

Tener la posibilidad incluso de ganar la competencia internacional con los primeros CanSats Mexicanos, que a su regreso capacitarán a Universidades Mexicanas para formar la primera competencia Nacional de CanSats.

6.6 Proyecto semilla para arrancar un programa de capacitación de recursos humanos en satélites pequeños CanSat con un equipo de trabajo

Para empezar un programa como el descrito, se debe preparar y entrenar al grupo semilla que después difundirá ampliamente tanto el programa como la competencia Nacional CanSat en Universidades de todo nuestro país.

Con tal objetivo, se propone capacitar en equipo a un grupo de ingenieros de la UNAM y del IPN, dentro del proyecto semilla, el cual persigue generar (en equipo) dos tipos diferentes de CanSats de alta tecnología. El grupo de trabajo estará compuesto por dos asesores con experiencia en el tema y por estudiantes de Ingeniería de carreras de Aeronáutica, Electrónica, Computación, Telecomunicaciones y Mecatrónica.

Para adquirir la capacitación completa que le de gran experiencia al grupo de trabajo, y para implantar después el plan Nacional de Capacitación en Satélites CanSat, se propone que el grupo desarrolle y participe en equipo con dos prototipos satelitales para una competencia Internacional CanSat.

6.7 Apoyos requeridos para arrancar el programa semilla de capacitación en tecnología CanSat

Para cristalizar el proyecto semilla de capacitación de recursos humanos en tecnología CanSat se requieren los siguientes apoyos:

- Componentes y materiales para desarrollar un CanSat tipo “rover come-back”, es decir, un satélite tipo robot con capacidad de aterrizaje después de lanzarse con un cohete sonda. Una vez en tierra el vehículo debe navegar por medios autónomos (ruedas e instrumentación electrónica) hasta llegar a una meta definida al inicio de la competencia. Este CanSat estará instrumentado con computadora, 2 motores para mover ruedas (desplazamiento izquierda, derecha, adelante y atrás), GPS, brújula digital, sonar para detectar obstáculos, baterías, sensores de presión atmosférica, acelerómetros, cámara de video, comunicación Blue-tooth para comunicación con dispositivos inteligentes y mucho software inteligente de

navegación, captura y procesamiento de datos. Todo esto se integrará en un CanSat que pesará 350 gramos que es el límite de la competencia.

- Componentes y materiales para desarrollar un CanSat tipo “Flying come-back”, el cual es un dispositivo capaz de planear grandes distancias, guiado y posicionado por GPS. Utilizará alas desplegables para alcanzar lo más cerca posible una diana colocada en el campo de lanzamiento. Tiene además todas las funciones básicas de un satélite (alimentación, comunicaciones, etc.).

- Movilidad de 2 equipos de trabajo para participar en una competencia internacional de CanSats.

6.8 Apoyo requerido para iniciar preliminarmente la segunda fase del programa de capacitación nacional en tecnología CanSat

Se propone que el programa Nacional de capacitación CanSat se realice como un proyecto consecutivo a esta propuesta. Para llevar adelante tal fase de trabajo habrá que contar con el apoyo de diversas entidades. La forma de realizar el programa será parecido al esquema que usan en Japón, en donde se deberá concentrar a las personas bajo entrenamiento durante al menos 2 o 3 semanas (deberán tener alojamiento y alimentos durante el curso y se trabajará también durante los sábados), en una institución que cuente con laboratorio de electrónica, taller mecánico, que cuente con vehículos para traslado a campo cercano, sala de reuniones y que esté cerca de campo abierto para realizar lanzamientos con globos y/o cohetes. Además, se deberán tener recursos para que cada asistente al programa pueda tener los materiales necesarios para desarrollar un CanSat (sensores, GPS, equipos de comunicaciones inalámbricos, protoboards, computadora en una sola tarjeta, cables, etc). Evidentemente que también se deberá contar con un globo aerostático y/o 4 cohetes sonda con número adecuado de motores de cohetes, más una pequeña torre de lanzamiento y equipo diverso.

Por otro lado es deseable que se prepare un curso teórico de antecedentes y entrenamiento previo para los participantes que recibirán la capacitación CanSat del próximo año. En este curso previo se hará énfasis en las partes que demandan más tiempo al desarrollar CanSats, entre ellas nociones de estructuras mecánicas, instrumentación electrónica, herramientas de desarrollo de software, así como aspectos de pruebas y validación operativa. Este curso preliminar permitirá que las

personas que se capacitarán el próximo año tengan mejores conocimientos para facilitar y alcanzar un mejor entrenamiento durante el curso.

6.9 Resultados esperados

Por medio de este proyecto semilla se obtendrán los siguientes resultados:

- La capacitación de un equipo de trabajo Mexicano (12 personas) en Tecnología CanSat.

- El equipo de trabajo (jóvenes Ingenieros asesorados por expertos, los proponentes) obtendrá entrenamiento, experiencia teórica y de campo en tecnología CanSat.

- El equipo obtendrá certificación internacional en tecnología CanSat por medio de su participación a nombre de México, CONACYT, UNAM e IPN en una competencia internacional CanSat.

- El equipo fabricará dos tipos prototipos CanSat completamente diferentes uno con capacidad de vuelo para llegar a una meta preestablecida y otro con capacidad de navegación en tierra para llegar a una meta predefinida. Ambos se llevarán a competir internacionalmente.

- El equipo de trabajo obtendrá vínculos de colaboración (red) internacional en el campo de satélites CanSats. Los vínculos Aeroespaciales se concretarán particularmente con los países y las instituciones que participarán en la competencia Internacional de CanSats.

- Este proyecto semilla fomentará la estrecha participación y capacitación de dos instituciones que son las dos principales instituciones educativas del país. Con ello se logrará una mayor vinculación de las mismas y el uso compartido de sus laboratorios y recursos. Posteriormente, la capacitación CanSat que adquieran, la compartirán por medio de las iniciativas ya señaladas con universidades de la red y también promoverán la integración de nuevas Universidades a la red.

- Como meta a mediano plazo, es decir, como continuación de este proyecto semilla (fase 2), se generará el programa de Capacitación nacional CanSat, la competencia nacional CanSat, y en paralelo se desarrollarán globos aerostáticos y se prepararán cohetes zonda, capaces de levantar dos o tres

CanSat a alturas suficientes para complementar la capacitación CanSat en México.

- Se realizará una documentación completa de calidad Internacional en el campo de satélites pequeños CanSat. Entre ellos documentación de:
    - Revisión Preliminar de Diseño.
    - Revisión Crítica de Diseño.
    - Reporte completo de Resultados de operación de los 2 CanSats referidos.

7. Conclusiones y trabajo futuro

Todos los objetivos planteados en la tesis se cumplieron satisfactoriamente, se terminó el CanSat del IINGEN, con todos los subsistemas y software operativo.

Se diseñó un sistema de comunicaciones que cumple con las necesidades del CanSat, es de muy baja potencia, utiliza únicamente 52 mA y con potencia de transmisión de 15 dBm, al compararlo con un transmisor XBee que utiliza 265 mA y con potencia de trasmisión de 0 dBm, son notorios los beneficios de utilizar un sistema de comunicaciones hecho a la medida, el modelo de XBee más parecido en potencia de trasmisión tiene un Transceptor con velocidad de 156 Kbps, transmite 17 dBm y consume 210 mA.

Se probaron y programaron los sensores de presión y temperatura, el acelerómetro, la brújula, el GPS, el transmisor y receptor, el sistema de recuperación (LEDs y Buzzer), el control de los motores y la navegación, la memoria y el DSP donde cada uno requiere su propia programación.

Se crearon nuevas ruedas ultra ligeras hechas con espuma de poliuretano y silicón. Ruedas más ligeras y más anchas le dan mayor movilidad al CanSat y con esto fue posible utilizar motores más pequeños y menos potentes sin disminuir la velocidad.

Se realizó una descripción detallada de los métodos de lanzamiento de CanSats, el tipo de lanzamiento más común es con cohete sonda donde se coloca el CanSat dentro del cohete y se puede lanzar a muchos kilómetros de altura. Esta opción es muy utilizada por que es menos problemático y voluminoso que los globos. Además la cohetería es estudiada en algunas universidades donde se realizan lanzamientos CanSat.

Otra opción menos utilizada por ser nueva es con un “drone” que eleve el CanSat hasta una altura aproximada y lo libere con un mecanismo controlado por RF, actualmente esta es la mejor opción si se desea adquirir un vehículo de lanzamiento específicamente para este propósito.

El método de lanzamiento con globo es el menos utilizado por ser voluminoso y difícilmente alcanza grandes alturas por lo tanto este método es mejor para que los estudiantes hagan pruebas con globos pequeños amarrados.

También se planteó la propuesta de un plan para capacitar recursos humanos a nivel licenciatura y posgrado donde se plantea capacitar un grupo semilla de expertos en tecnología CanSat de diferentes universidades para que ellos a su vez

extiendan el conocimiento a grupos de ingenieros dentro de las instituciones educativas del país. Posteriormente se organizará un concurso nacional donde los participantes habrán adquirido conocimientos de alta calidad respaldados por el proceso de diseño, construcción y operación de un CanSat, además es necesario aprender a hacer la documentación necesaria para el proyecto.

Como trabajo futuro es recomendable cambiar el sensor de presión y temperatura que se utilizó aquí por ser muy inexacto y poco confiable. Para aplicaciones de altimetría es necesario utilizar sensores más precisos como el MS5607 que es un poco más caro pero es más confiable o el MPL3115A2 que no es tan preciso pero tampoco es tan costoso.

También es recomendable hacer un circuito llamado “Squelch” que se integra en el receptor de la estación terrena y sirve para evitar que el receptor oscile y genere información aleatoria en ausencia de la portadora.

Referencias

[1] Agencia Espacial Mexicana. (2013). Proyecto Multiinstitucional de Formación de Capital Humano en el Campo Espacial. Mexico, DF.

[2] UNISEC-Global. (s.f.). *UNISEC-Global*. Obtenido de http://www.unisec-global.org/whatwedo.html

[3] ARLISS. (s.f.). *ARLISS A Rocket Launch for International Student Satellites*. Obtenido de http://www.arliss.org/

[4]Parrot SA. (s.f.). *Parrot MiniDrones*. Obtenido de http://www.parrot.com/es/

[5] LINX TECHNOLOGIES, INC. (2011). TXM-916-ES SERIES TRANSMITTER DATA GUIDE. Merlin, Oregon.

[6] Sirenza Microdevices, Inc. (s.f.). DC-2500 MHz, Cascadable.

[7] Balanis, C. A. (1997). *Antenna theory: analysis and design.* Wiley.

[8] Texas Instruments Incorporated. (2012). LAUNCHXL-F28027 C2000 Piccolo LaunchPad User's Guide.

[9] Texas Instruments Incorporated. (2011). Code Composer Studio V 5.0.

[10] LINX TECHNOLOGIES, INC. (s.f.). RXM-GPS-SR SERIES GPS RECEIVER MODULE DATA GUIDE.

[11] SD Group. (22 de Enero de 2013). SD Specifications Part 1 Physical Layer Simplified Specification Version 4.10.

[12] European Space Agency. (s.f.). *www.esa.int*. Obtenido de http://www.esa.int/Education/CanSat

[13] Rossbach de México, S.A. de C.V. (s.f.). *Rossbach de Mexico .* Recuperado el Junio de 2014, de http://www.rossbachmexico.com/globo.htm

[14] T-Minus Engineering B.V. (s.f.). *T-Minus Engineering B.V. Delft The Netherlands.* Obtenido de http://www.t-minus.nl/: http://www.t-minus.nl/products/cansat-launcher/

[15] Comisión Federal de Telecomunicaciones. (2012). Cuadro Nacional de Atribución de Frecuencias 2009. Mexico, DF.

[16] Rodríguez, A. A. (2004). *Geografia General.* Pearson Educación.

[17] Freescale Semiconductor. (s.f.). Miniature I2C Digital Barometer Document Number: MPL115A2.

[18] Portland State Aerospace Society. (2004). A Quick Derivation relating altitude to air. EUA, Oregon, Portland.

[19]Honeywell International Inc. (2006). Application Note – AN219 Digital Compass Reference Design with the SiRFstar2t GPS Chipset. Plymouth.

[20] IHS GlobalSpec. (s.f.). *Globalspec.* Recuperado el marzo de 2014, de http://datasheets.globalspec.com/ds/2353/RobotMarketplace/DAC1610A-B8B2-48CC-8104-EE06D427F16A

Anexos

Anexo 1: Código para configurar y controlar el receptor GPS

El código comentado en la rutina de interrupción de transmisión se descomenta cuando es necesario cambiar la configuración del GPS (tipos de mensajes, velocidad de transmisión, formato de los mensajes y frecuencia de los mensajes).

```
#include "DSP28x_Project.h"
#include <math.h>
#include <stdlib.h>
#include <string.h>

void   SCI_Init(void);
interrupt void sci_rxint_isr(void);
interrupt void sci_txint_isr(void);

extern void DSP28x_usDelay(Uint32 Count);

char NMEA[27]="$PSRF100,1,38400,8,1,0*3D\x0D\x0A";
char GGA[25]="$PSRF103,00,00,00,01*24\x0D\x0A";
char GLL[25]="$PSRF103,01,00,00,01*25\x0D\x0A";
char GSA[25]="$PSRF103,02,00,00,01*26\x0D\x0A";
char GSV[25]="$PSRF103,03,00,00,01*27\x0D\x0A";
char RMC[25]="$PSRF103,04,00,01,01*20\x0D\x0A";
char VTG[25]="$PSRF103,05,00,00,01*21\x0D\x0A";

char DatosGPS[72],temp[3],*lim=",";
int i=0, j=0, k=0, gga=0, gll=0, gsa=0, gsv=0, rmc=0,vtg=0,nmea=0,inicio=0,comp;
double azimuth,Long1,Lat1,pi=3.14159265359,distancia,Long3,Lat3;
double Long2=99.180840,Lat2=19.328077;

void main(void)
{
   InitSysCtrl();
   InitSciGpio();
   DINT;
   InitPieCtrl();
   IER = 0x0000;
   IFR = 0x0000;
   InitPieVectTable();
   EALLOW;
   PieVectTable.SCIRXINTA = &sci_rxint_isr;
   PieVectTable.SCITXINTA = &sci_txint_isr;
   EDIS;
   SCI_Init();
   PieCtrlRegs.PIEIER9.bit.INTx1 = 1;
   PieCtrlRegs.PIEIER9.bit.INTx2 = 1;
   IER |= M_INT9;
   EINT;
   for(k=0;k<72;k++)DatosGPS[k]=0;
```

```
/*
   SciaRegs.SCICTL1.bit.TXWAKE=1;
   SciaRegs.SCITXBUF=0x1;
*/
   for(;;)
   {
         strncpy(temp,&DatosGPS[20],2);
         temp[2]=NULL;
         Lat1=(strtod(&temp[0],NULL))+((strtod(&DatosGPS[22],&lim))/60.0);
         strncpy(temp,&DatosGPS[32],3);
         Long1=(strtod(&temp[0],NULL))+((strtod(&DatosGPS[35],&lim))/60.0);
         Long3=Long1-Long2;
         Lat3=Lat2-Lat1;

        azimuth=atan(sin((Long3)*pi/180.0)/((cos(Lat1*pi/180.0)*tan(Lat2*pi/180
        .0))-(sin(Lat1*pi/180.0)*cos((Long3)*pi/180.0))))*180.0/pi;
        distancia=6378137*acos(cos(Lat1)*cos(Lat2)*cos(Long2-
        Long1)+sin(Lat1)*sin(Lat2));
   }

}   // end of main

void SCI_Init(void)
{
      SciaRegs.SCICCR.all=0x07;       // One stop bit =0, odd parity =0, parity
disabled=0, loop back dis=0, idle-line=0, 8 bit data= 111
      SciaRegs.SCICTL1.all=0x23;      // reserved, receive error interrupt
dis=0, sw reset dis=1, reserved, txwake dis=0, sleep dis=0, txena=1, rxena=1
      SciaRegs.SCIHBAUD=0;
//    SciaRegs.SCILBAUD=194;                // 9600 baud
      SciaRegs.SCILBAUD=48;                 // 38400 baud
//    SciaRegs.SCILBAUD=31;                 // 57600 baud
      SciaRegs.SCICTL2.all=0xC3;            // Txrdy flag=1, txempty flag=1,
reserved5-2, Rx/Bk int ena=1, tx int ena=1
      SciaRegs.SCIRXST.bit.RXERROR=1; // Banderas de error en la recepcion.
}
interrupt void sci_rxint_isr(void)
{
      if((SciaRegs.SCIRXST.bit.FE==1)||(SciaRegs.SCIRXST.bit.OE==1)||(SciaRegs.
SCIRXST.bit.PE==1)||(SciaRegs.SCIRXST.bit.BRKDT==1))
      {
            asm("   ESTOP0");
            SciaRegs.SCICTL1.bit.SWRESET=0;
            PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;
            SciaRegs.SCICTL1.bit.SWRESET=1;
            return;
      }
      if(SciaRegs.SCIRXST.bit.RXRDY==1)
      DatosGPS[j]=SciaRegs.SCIRXBUF.bit.RXDT;
      if(DatosGPS[j]==0x24)
      {
            DatosGPS[0]=DatosGPS[j];
            j=0;
      }
      if(j==71)j=0;
```

```
	if(j==5)
	{
	if((DatosGPS[0]==0x24)&&(DatosGPS[1]==0x47)&&(DatosGPS[2]==0x50)&&(DatosG
PS[3]==0x52)&&(DatosGPS[4]==0x4D)&&(DatosGPS[5]==0x43))
		{	}
		else
		{
			j=0;
		}
	}
	if(DatosGPS[j]==0x0A)
	{
		for(k=j+1;k<72;k++)DatosGPS[k]=0;
		j=0;
	}
	j++;
	PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;
}
interrupt void sci_txint_isr(void)
{/*
	while(gga==0)
	{
		SciaRegs.SCITXBUF=GGA[i];
		i++;
		if(i==25)
		{
			gga=1;
			i=0;
		}
		PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;
		return;
	}
	while(gll==0)
	{
		SciaRegs.SCITXBUF=GLL[i];
		i++;
		if(i==25)
		{
			gll=1;
			i=0;
		}
		PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;
		return;
	}
	while(gsa==0)
	{
		SciaRegs.SCITXBUF=GSA[i];
		i++;
		if(i==25)
		{
			gsa=1;
			i=0;
		}
		PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;
		return;
```

```
	}
	while(gsv==0)
	{
		SciaRegs.SCITXBUF=GSV[i];
		i++;
		if(i==25)
		{
			gsv=1;
			i=0;
		}
		PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;
		return;
	}
	while(rmc==0)
	{
		SciaRegs.SCITXBUF=RMC[i];
		i++;
		if(i==25)
		{
			rmc=1;
			i=0;
		}
		PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;
		return;
	}
	while(vtg==0)
	{
		SciaRegs.SCITXBUF=VTG[i];
		i++;
		if(i==25)
		{
			vtg=1;
			i=0;
		}
		PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;
		return;
	}
	while(nmea==0)
	{
		SciaRegs.SCITXBUF=NMEA[i];
		i++;
		if(i==27)
		{
			nmea=1;
			i=0;
		}
		PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;
		return;
	}*/
}
```

Anexo 2: Código para configurar y controlar la tarjeta de memoria SD

Este código es largo y complicado porque la tarjeta requiere una secuencia de inicio para configurar la tarjeta con el protocolo de comunicación correcto y preparar la tarjeta para lectura y escritura.

```
#include <stdio.h>
#include "DSP28x_Project.h"

extern void DSP28x_usDelay(Uint32 Count);
void Enviar( char *chr, int cnt);
int SdInit(void);
int Set_Blk(void);
int Write_Blk(void);
int Read_Blk(void);
void SPI_Init(void);
interrupt void spi_rxint_isr(void);
 char CRC7( char * chr, int cnt);
 char Rbuf, error=0, c0=0, c8=0, c55=0, ac41=0, c16=0, c24=0, c17=0, d=0, rec=0,
Res[5], crc16[]="\x11\x11",blk=0, inicio[]="\xFE",
dummy[]="\xFF",relleno[]="\x00";
 char cmd0[]="\x40\x00\x00\x00\x00";
 char cmd8[]="\x48\x00\x00\x01\xAA";
 char cmd55[]="\x77\x00\x00\x00\x00";
 char acmd41[]="\x69\x00\x00\x00\x00";
 char cmd16[]="\x50\x00\x00\x02\x00";  //bloque de 512 (min)
 char cmd24[]="\x58\x00\x00\xA0\x00";  //inicio en la direccion 0xA000 dejando
libres los primeros 10 clusters = 40 KB
 char cmd17[]="\x51\x00\x00\xA0\x00";
 char s[]="Prueba oficial",r[515];
int i=0, j=0,a,init=0;
short data=0;
void main(void)
{
   InitSysCtrl();
   InitSpiGpio();
   DINT;
   InitPieCtrl();
   IER = 0x0000;
   IFR = 0x0000;
   InitPieVectTable();
   EALLOW;
   PieVectTable.SPIRXINTA = &spi_rxint_isr;
   GpioCtrlRegs.GPAMUX1.bit.GPIO6=0x00;
   GpioCtrlRegs.GPADIR.bit.GPIO6=1;
   GpioCtrlRegs.GPAPUD.bit.GPIO6=1;
   EDIS;
   SPI_Init();
   PieCtrlRegs.PIEIER6.bit.INTx1 = 1;
   IER |= M_INT6;
   EINT;
```

```
   cmd0[5]=CRC7(&cmd0[0],5);
   cmd8[5]=CRC7(&cmd8[0],5);
   cmd55[5]=CRC7(&cmd55[0],5);
   acmd41[5]=CRC7(&acmd41[0],5);
   cmd16[5]=CRC7(&cmd16[0],5);
   cmd17[5]=CRC7(&cmd17[0],5);
   cmd24[5]=CRC7(&cmd24[0],5);
   while(init<5){
         if(SdInit()!=1){
               init++;
         }
         else {
               init=6;
         }
   }
   while(blk<5){
         if(Set_Blk()!=1){
               blk++;
         }
         else{
               blk=6;
         }
   }

   if(Write_Blk()!=1)asm("   ESTOP0");

   if(Read_Blk()!=1)asm("   ESTOP0");

   asm("   ESTOP0");

}   // end of main
void SPI_Init(void)
{
      SpiaRegs.SPICCR.all=0x87; //SW reset =1 ; clk polarity= 0 ; reserved ;
loopback =0 ; 8 bits = 0111
      SpiaRegs.SPICTL.all=0x0F; //7-5 reserved ; OE int =0 ; clk phase = 1 ;
Master/slave =1 ; talk =1 ; int ena =1
      SpiaRegs.SPIBRR=0x0;
      SpiaRegs.SPIPRI.bit.TRIWIRE=0;
      GpioDataRegs.GPADAT.bit.GPIO6=1;
}
interrupt void spi_rxint_isr(void)
{
      Rbuf=SpiaRegs.SPIRXBUF;
      if((d==1)&&((Res[0]&0x1F)==0x05)&&(Rbuf==0xFF)){
            d=0;
            j=0;
      }     else if((d==1)&&(j>1))j--;
      if(Rbuf<0xFF){
            if(rec==1){
                  r[j]=Rbuf;
                  j++;
                  if(j==515){
                        j=0;
                        rec=0;
```

```
                }
                PieCtrlRegs.PIEACK.all = PIEACK_GROUP6;
                return;
            }
            Res[j]=Rbuf;
            j++;
        if(((c0==1)||(c55==1)||(c24==1)||(c17==1)||(ac41==1))&&(Res[0]<=1)){
                c0=0;
                c55=0;
                c24=0;
                c17=0;
                ac41=0;
                j=0;
            }else if((c0==1)||(c55==1)||(c24==1)||(c17==1))j=0;
            if(c8==1){
                if((j==5)&&(Res[0]<3)&&(Res[4]==0xAA)){
                    c8=0;
                    j=0;
                    error=0;
                }
            }
            if((Res[0]>0x03)&&(d==0)&&(ac41==0)&&(c55==0)){
                j=0;
                error=1;
            }
            if((c16==1)&&(Res[0]==0)){
                j=0;
                c16=0;
            }
        }
        PieCtrlRegs.PIEACK.all = PIEACK_GROUP6;
}
int SdInit(void)
{
        for(a=0;a<5;a++)Res[a]=0;
        error=0;
        for(a=0;a<10;a++)Enviar(&dummy[0],1);
        GpioDataRegs.GPADAT.bit.GPIO6=0;
        c0=1;
        Enviar(&cmd0[0],6);
        while(c0==1)Enviar(&dummy[0],1);
        if(error==1)return 0;
        for(a=0;a<5;a++)Res[a]=0;
        c8=1;
        Enviar(&cmd8[0],6);
        while(c8==1)Enviar(&dummy[0],1);
        if(error==1)return 0;
        if(Res[4]!=0xAA)return 0;
        if(Res[3]!=0x01)return 0;
        Res[0]=0x2;
        do{
            c55=1;
            Enviar(&cmd55[0],6);
            while(c55==1)Enviar(&dummy[0],1);
            if(error==1)return 0;
```

```
            ac41=1;
            Enviar(&acmd41[0],6);
            while(ac41==1)Enviar(&dummy[0],1);
      }while(Res[0]!=0);
      GpioDataRegs.GPADAT.bit.GPIO6=1;
      return 1;
}
 char CRC7( char * chr, int cnt)
{
      int k,a;
      char crc,Data;
      crc=0;
      for (a=0;a<cnt;a++) {
            Data=chr[a];
            for (k=0;k<8;k++) {
                  crc <<= 1;
                  if ((Data & 0x80)^(crc & 0x80))
                        crc ^=0x09;
                  Data <<= 1;
            }
      }
      crc = crc & 0x7F;
      crc=(crc<<1)|1;
      return(crc);
}
int Set_Blk(void)
{
      GpioDataRegs.GPADAT.bit.GPIO6=0;
      c16=1;
      Enviar(&cmd16[0],6);
      while(c16==1)Enviar(&dummy[0],1);
      GpioDataRegs.GPADAT.bit.GPIO6=1;
      if(error==1) return 0;
      return 1;
}
int Write_Blk(void){
      GpioDataRegs.GPADAT.bit.GPIO6=0;
      c24=1;
      Enviar(&cmd24[0],6);
      while(c24==1)Enviar(&dummy[0],1);
      if(error==1)return 0;
      d=1;
      Enviar(&inicio[0],1);
      Enviar(&s[0],14);
      for(a=0;a<501;a++)Enviar(&relleno[0],1);
      Enviar(&crc16[0],2);
      while(d==1)Enviar(&dummy[0],1);
      GpioDataRegs.GPADAT.bit.GPIO6=1;
      if(error==1) return 0;
      return 1;
}
int Read_Blk(void)
{
      GpioDataRegs.GPADAT.bit.GPIO6=0;
      c17=1;
```

```
	Enviar(&cmd17[0],6);
	while(c17==1)Enviar(&dummy[0],1);
	rec=1;
	while(rec==1)Enviar(&dummy[0],1);
	if(error==1) return 0;
	while(rec==1);
	GpioDataRegs.GPADAT.bit.GPIO6=1;
	return 1;
}
void Enviar( char *chr, int cnt)
{
	i=0;
	while(i<cnt) {
		if(SpiaRegs.SPISTS.bit.BUFFULL_FLAG==0)	{
			data=chr[i];
			data=data<<8;
			SpiaRegs.SPITXBUF=data;
			i++;
		}
	}
}
```

Anexo 3: Código para configurar y controlar los motores, buzzer y LEDs

```
#include "DSP28x_Project.h"
#include <stdlib.h>
void EPWM_Init(void);
void Sys_Init(void);
void epwm2_int_isr(void);
void timer0_int_isr(void);
void Girar(void);
int a,b,c,i, girar=10, g=0, ciclos;
void main(void) {
	Sys_Init();
	for(i=0;i<5;i++);
	GpioDataRegs.GPADAT.bit.GPIO1=0;
	for(i=0;i<5;i++);
	GpioDataRegs.GPADAT.bit.GPIO3=1;
	for(i=0;i<5;i++);
	GpioDataRegs.GPADAT.bit.GPIO4=0;
	for(i=0;i<5;i++);
	GpioDataRegs.GPADAT.bit.GPIO5=1;
	while(1){
		if((girar!=0)&&(g==0)){
			Girar();
		}
	}
}
void EPWM_Init(void){
	CpuTimer0Regs.TPR.bit.TDDR=0x34;	// 33,333 Hz
	CpuTimer0Regs.TPRH.bit.TDDRH=0x82;		// 33,333 Hz
	CpuTimer0Regs.TIM.half.LSW=0x0708;		// 1,800 Hz = 5 vueltas
```

```
       CpuTimer0Regs.PRD.all=20;
       CpuTimer0Regs.TCR.all=0x0810;          //Inter Flag clear=0, Inter En=1,
res=00, FREE/soft=10, res=0000, tim reload=0, stop=1, res=0000

       EPwm2Regs.TBPRD=0x177;
       EPwm2Regs.TBPHS.all=0;
       EPwm2Regs.TBCTR=0;
       EPwm2Regs.TBCTL.all=0xC682;     //Free/soft = 11 ; PHSDIR=0 ; CLKDIV=001
; HSPCLKDIV=101 ; SWFSYNC=0 ; SYNCOSEL=00 ; PRDLD=0 ; PHSEN=0 ; CTRMODE=10
       EPwm2Regs.AQCTLA.all=0x9;       //15-12 reserved ; CBD=00 ; CBU=00 ;
CAD=00 ; CAU=00 ; PRD=10 ; ZRO=01
       EPwm1Regs.TBPRD=0x4165;
       EPwm1Regs.TBPHS.all=0;
       EPwm1Regs.TBCTR=0;
       EPwm1Regs.TBCTL.all=0xDF82;     //Free/soft = 11 ; PHSDIR=0 ; CLKDIV=111
; HSPCLKDIV=111 ; SWFSYNC=0 ; SYNCOSEL=00 ; PRDLD=0 ; PHSEN=0 ; CTRMODE=10
       EPwm1Regs.AQCTLA.all=0x9;       //15-12 reserved ; CBD=00 ; CBU=00 ;
CAD=00 ; CAU=00 ; PRD=10 ; ZRO=01

}
void Sys_Init(void){
        InitSysCtrl();
        InitEPwmGpio();
        DINT;
        InitPieCtrl();
        IER = 0x0000;
        IFR = 0x0000;
        InitPieVectTable();
        EALLOW;
        PieVectTable.TINT0=&timer0_int_isr;
        EDIS;
        EALLOW;
        GpioCtrlRegs.GPAMUX1.bit.GPIO1=0x00;
        GpioCtrlRegs.GPAMUX1.bit.GPIO3=0x00;
        GpioCtrlRegs.GPAMUX1.bit.GPIO4=0x00;
        GpioCtrlRegs.GPAMUX1.bit.GPIO5=0x00;
        GpioCtrlRegs.GPADIR.bit.GPIO1=1;
        GpioCtrlRegs.GPADIR.bit.GPIO3=1;
        GpioCtrlRegs.GPADIR.bit.GPIO4=1;
        GpioCtrlRegs.GPADIR.bit.GPIO5=1;
        GpioCtrlRegs.GPAPUD.bit.GPIO1=1;
        GpioCtrlRegs.GPAPUD.bit.GPIO3=1;
        GpioCtrlRegs.GPAPUD.bit.GPIO4=1;
        GpioCtrlRegs.GPAPUD.bit.GPIO5=1;
        EDIS;
        EPWM_Init();
        PieCtrlRegs.PIEIER1.bit.INTx7 = 1;
        IER |= M_INT1;
        EINT;
}
void timer0_int_isr(void){
       for(i=0;i<5;i++);
       GpioDataRegs.GPADAT.bit.GPIO1=0;
       for(i=0;i<5;i++);
       GpioDataRegs.GPADAT.bit.GPIO3=0;
```

```
	for(i=0;i<5;i++);
	GpioDataRegs.GPADAT.bit.GPIO4=0;
	for(i=0;i<5;i++);
	GpioDataRegs.GPADAT.bit.GPIO5=0;
	CpuTimer0Regs.TCR.bit.TIF=1;
	g=0;
	girar=0;
	PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
}
void Girar(void){
	g=1;
	ciclos=abs(girar)+30;
	CpuTimer0Regs.TPR.bit.TDDR=0x34;	// 33,333 Hz
	CpuTimer0Regs.TPRH.bit.TDDRH=0x82;		// 33,333 Hz
	CpuTimer0Regs.PRD.all=ciclos;
	CpuTimer0Regs.TCR.bit.TRB=1;
	CpuTimer0Regs.TCR.all=0x8810;		//Inter Flag clear=1, Inter En=0,
res=00, FREE/soft=10, res=0000, tim reload=0, stop=1, res=0000
	if(girar>0){
		for(i=0;i<5;i++);
		GpioDataRegs.GPADAT.bit.GPIO1=0;
		for(i=0;i<5;i++);
		GpioDataRegs.GPADAT.bit.GPIO3=1;
		for(i=0;i<5;i++);
		GpioDataRegs.GPADAT.bit.GPIO4=1;
		for(i=0;i<5;i++);
		GpioDataRegs.GPADAT.bit.GPIO5=0;
		CpuTimer0Regs.TCR.all=0x8800;		//Inter Flag clear=1, Inter
En=0, res=00, FREE/soft=10, res=0000, tim reload=0, start=0, res=0000
	}
	else if(girar<0){
		for(i=0;i<5;i++);
		GpioDataRegs.GPADAT.bit.GPIO1=1;
		for(i=0;i<5;i++);
		GpioDataRegs.GPADAT.bit.GPIO3=0;
		for(i=0;i<5;i++);
		GpioDataRegs.GPADAT.bit.GPIO4=0;
		for(i=0;i<5;i++);
		GpioDataRegs.GPADAT.bit.GPIO5=1;
		CpuTimer0Regs.TCR.all=0x8800;		//Inter Flag clear=1, Inter
En=0, res=00, FREE/soft=10, res=0000, tim reload=0, start=0, res=0000
	}
	while(CpuTimer0Regs.TCR.bit.TIF==0);
	for(i=0;i<5;i++);
	GpioDataRegs.GPADAT.bit.GPIO1=0;
	for(i=0;i<5;i++);
	GpioDataRegs.GPADAT.bit.GPIO3=0;
	for(i=0;i<5;i++);
	GpioDataRegs.GPADAT.bit.GPIO4=0;
	for(i=0;i<5;i++);
	GpioDataRegs.GPADAT.bit.GPIO5=0;
	CpuTimer0Regs.TCR.bit.TSS=1;
	CpuTimer0Regs.TCR.bit.TIF=1;
	g=0;
	girar=0;
```

```
}
```

Anexo 4: Código para configurar y controlar el acelerómetro

```
#include "DSP28x_Project.h"
void   I2CA_Init(void);
void Acelerometro(void);
int Wake(void);
interrupt void i2c_int1a_isr(void);
extern void DSP28x_usDelay(Uint32 Count);
int a,b,i=0,j=0,acel=0,estado=0,stat=0,activar=0,rec[8];
int x1,y1,z1;
double x,y,z;
void main(void)
{
   InitSysCtrl();
   InitI2CGpio();
   DINT;
   InitPieCtrl();
   IER = 0x0000;
   IFR = 0x0000;
   InitPieVectTable();
   EALLOW;
   PieVectTable.I2CINT1A = &i2c_int1a_isr;
   EDIS;
   I2CA_Init();
   PieCtrlRegs.PIEIER8.bit.INTx1 = 1;
   IER |= M_INT8;
   EINT;
   for(i=0;i<8;i++)
   {
         rec[i]=0;
   }
   i=0;
   while(Wake()!=1);
   while(1)
   {
   Acelerometro();
   }
}   // end of main
void I2CA_Init(void)
{
   #if (CPU_FRQ_60MHZ)
     I2caRegs.I2CPSC.all = 5;    // Prescaler - need 7-12 Mhz on module clk d=5
   #endif
   I2caRegs.I2CCLKL = 45;// NOTE: must be non zero  60 Hz NOTE: must be non zero
   I2caRegs.I2CCLKH = 45;    // f= 60 MHz / [(IPSC+1)[(ICCL+d)+(ICCH+d)]]=100KHz
   I2caRegs.I2CIER.all = 0x18;      // Enable XRDY, RRDY interrupts
   I2caRegs.I2CMDR.all = 0x0020;    // Take I2C out of reset
}
void Acelerometro()
{
      acel=1;
```

```
	I2caRegs.I2CSAR = 0x1C;
	I2caRegs.I2CCNT = 1;
	I2caRegs.I2CDXR=0x01;
	I2caRegs.I2CMDR.all = 0x2620;   // NACKMOD dis=0, FREE dis=0, STT=1,
reserved, STP=0, MST=1, TRX=1, XA=0, RM=0, DLB=0, IRS=1, STB=0, FDF=0, 8bit
BC=000
	while(I2caRegs.I2CSTR.bit.ARDY==0);
	I2caRegs.I2CSAR = 0x1C;
	I2caRegs.I2CCNT = 6;
	I2caRegs.I2CMDR.all = 0x2C20;   // NACKMOD dis=0, FREE dis=0, STT=1,
reserved, STP=1, MST=1, TRX=0, XA=0, RM=0, DLB=0, IRS=1, STB=0, FDF=0, 8bit
BC=000
	while(I2caRegs.I2CSTR.bit.BB == 1);
	j=0;
	acel=0;
	if(rec[0]>=128){
		x1=((((rec[0]<<2)|(rec[1]>>6))^0x3FF)+1);
		x=x1*-0.0039;
	}
	else{
		x1=(rec[0]<<2)|(rec[1]>>6);
		x=x1*0.0039;
	}
	if(rec[2]>=128){
		y1=((((rec[2]<<2)|(rec[3]>>6))^0x3FF)+1);
		y=y1*-0.0039;
	}
	else{
		y1=(rec[2]<<2)|(rec[3]>>6);
		y=y1*0.0039;
	}
	if(rec[4]>=128){
		z1=((((rec[4]<<2)|(rec[5]>>6))^0x3FF)+1);
		z=z1*-0.0039;
	}
	else{
		z1=(rec[4]<<2)|(rec[5]>>6);
		z=z1*0.0039;
	}
}
int Wake()
{
	stat=1;
	I2caRegs.I2CSAR = 0x1C;
	I2caRegs.I2CCNT = 1;
	I2caRegs.I2CDXR=0x0B;
	I2caRegs.I2CMDR.all = 0x2620;   // NACKMOD dis=0, FREE dis=0, STT=1,
reserved, STP=0, MST=1, TRX=1, XA=0, RM=0, DLB=0, IRS=1, STB=0, FDF=0, 8bit
BC=000
	for(b=0;b<10;b++)for(a=0;a<10;a++)for(i=0;i<10;i++);
	while(I2caRegs.I2CSTR.bit.ARDY==0);
	I2caRegs.I2CSAR = 0x1C;
	I2caRegs.I2CCNT = 1;
```

```
	I2caRegs.I2CMDR.all = 0x2C20;   // NACKMOD dis=0, FREE dis=0, STT=1,
reserved, STP=1, MST=1, TRX=0, XA=0, RM=0, DLB=0, IRS=1, STB=0, FDF=0, 8bit
BC=000
	for(b=0;b<10;b++)for(a=0;a<10;a++)for(i=0;i<10;i++);
	while(I2caRegs.I2CSTR.bit.BB == 1);
	stat=0;
	if(estado==1)return 1;
	activar=1;
	I2caRegs.I2CSAR = 0x1C;
	I2caRegs.I2CCNT = 2;
	I2caRegs.I2CDXR=0x2A;
	I2caRegs.I2CMDR.all = 0x2E20;   // NACKMOD dis=0, FREE dis=0, STT=1,
reserved, STP=1, MST=1, TRX=1, XA=0, RM=0, DLB=0, IRS=1, STB=0, FDF=0, 8bit
BC=000
	for(b=0;b<10;b++)for(a=0;a<10;a++)for(i=0;i<10;i++);
	while(I2caRegs.I2CSTR.bit.BB == 1);
	activar=0;
	stat=1;
	I2caRegs.I2CSAR = 0x1C;
	I2caRegs.I2CCNT = 1;
	I2caRegs.I2CDXR=0x0B;
	I2caRegs.I2CMDR.all = 0x2620;   // NACKMOD dis=0, FREE dis=0, STT=1,
reserved, STP=0, MST=1, TRX=1, XA=0, RM=0, DLB=0, IRS=1, STB=0, FDF=0, 8bit
BC=000
	for(b=0;b<10;b++)for(a=0;a<10;a++)for(i=0;i<10;i++);
	while(I2caRegs.I2CSTR.bit.ARDY==0);
	I2caRegs.I2CSAR = 0x1C;
	I2caRegs.I2CCNT = 1;
	I2caRegs.I2CMDR.all = 0x2C20;   // NACKMOD dis=0, FREE dis=0, STT=1,
reserved, STP=1, MST=1, TRX=0, XA=0, RM=0, DLB=0, IRS=1, STB=0, FDF=0, 8bit
BC=000
	for(b=0;b<10;b++)for(a=0;a<10;a++)for(i=0;i<10;i++);
	while(I2caRegs.I2CSTR.bit.BB == 1);
	stat=0;
	if(estado==1)return 1;
	else return 0;
}
interrupt void i2c_int1a_isr(void)
{
	if(I2caRegs.I2CSTR.bit.XRDY==1)
	{
		if(activar==1)I2caRegs.I2CDXR=0x01;
	}
	if(I2caRegs.I2CSTR.bit.RRDY==1)
	{
		if(stat==1)estado=I2caRegs.I2CDRR;
		if(acel==1){
			rec[j]=I2caRegs.I2CDRR;
			j++;
		}
	}
	PieCtrlRegs.PIEACK.all = PIEACK_GROUP8;
}
```

Anexo 5: Código para configurar y controlar la brújula digital

```
#include "DSP28x_Project.h"
void I2CA_Init(void);
void Heading(void);
interrupt void i2c_int1a_isr(void);
extern void DSP28x_usDelay(Uint32 Count);
int a,b,i=0,j=0,head=0, heading=0,rec[8];
char   s[]="Hola Mundo!";
void main(void)
{
   InitSysCtrl();
   InitI2CGpio();
   DINT;
   InitPieCtrl();
   IER = 0x0000;
   IFR = 0x0000;
   InitPieVectTable();
   EALLOW;
   PieVectTable.I2CINT1A = &i2c_int1a_isr;
   EDIS;
   I2CA_Init();
   PieCtrlRegs.PIEIER8.bit.INTx1 = 1;
   IER |= M_INT8;
   EINT;
   for(i=0;i<8;i++)
   {
         rec[i]=0;
   }
   i=0;
   while(1){
      Heading();
   }
}   // end of main
void I2CA_Init(void)
{
   #if (CPU_FRQ_60MHZ)
     I2caRegs.I2CPSC.all = 5;     // Prescaler - need 7-12 Mhz on module clk d=5
   #endif
   I2caRegs.I2CCLKL = 45;// NOTE: must be non zero  60 Hz NOTE: must be non zero
   I2caRegs.I2CCLKH = 45;// f= 60 MHz / [ (IPSC+1)[(ICCL+d)+(ICCH+d)] ] =100 KHz
   I2caRegs.I2CIER.all = 0x18;      // Enable XRDY, RRDY interrupts
   I2caRegs.I2CMDR.all = 0x0020;    // Take I2C out of reset
                                    // Stop I2C when suspended
}
void Heading()
{
      head=1;
      I2caRegs.I2CSAR = 0x19;
      I2caRegs.I2CCNT = 1;
```

```
	I2caRegs.I2CDXR=0x50;
	I2caRegs.I2CMDR.all = 0x2620;   // NACKMOD dis=0, FREE dis=0, STT=1,
reserved, STP=0, MST=1, TRX=1, XA=0, RM=0, DLB=0, IRS=1, STB=0, FDF=0, 8bit
BC=000
	while(I2caRegs.I2CSTR.bit.ARDY==0);
	I2caRegs.I2CSAR = 0x19;
	I2caRegs.I2CCNT = 6;
	I2caRegs.I2CMDR.all = 0x2C20;   // NACKMOD dis=0, FREE dis=0, STT=1,
reserved, STP=1, MST=1, TRX=0, XA=0, RM=0, DLB=0, IRS=1, STB=0, FDF=0, 8bit
BC=000
	while(I2caRegs.I2CSTR.bit.BB == 1);
	head=0;
	heading=((rec[0]<<8)|rec[1]);
	j=0;
}

interrupt void i2c_int1a_isr(void)
{
	if(I2caRegs.I2CSTR.bit.XRDY==1)
	{
		if(head==1)I2caRegs.I2CDXR=0x00;
	}
	if(I2caRegs.I2CSTR.bit.RRDY==1)
	{
		if(head==1){
			rec[j]=I2caRegs.I2CDRR;
			j++;
		}
	}
	PieCtrlRegs.PIEACK.all = PIEACK_GROUP8;
}
```

Anexo 6: Código para configurar y controlar el sensor de presión y temperatura

```
#include "DSP28x_Project.h"
#include <math.h>
void I2CA_Init(void);
void Coefs(void);
void Conv(void);
interrupt void i2c_int1a_isr(void);
extern void DSP28x_usDelay(Uint32 Count);
unsigned int p,t,p0;
int a,b,i=0,j=0,k=0,coef=0,conv=0;
char  rec[8] ,s[]="Hola Mundo!", adc[4];
double a0=2121.125,b1=-2.507080078,b2=-
1.05090332,c12=0.000861406,pres,pres0,altura,altura1=0.0,altura2=0.0,altura3,w,x
,y,z,temperatura,c,d,e,f;
void main(void)
{
   InitSysCtrl();
   InitI2CGpio();
   DINT;
```

```
    InitPieCtrl();
    IER = 0x0000;
    IFR = 0x0000;
    InitPieVectTable();
    EALLOW;
    PieVectTable.I2CINT1A = &i2c_int1a_isr;
    EDIS;
    I2CA_Init();
    PieCtrlRegs.PIEIER8.bit.INTx1 = 1;
    IER |= M_INT8;
    EINT;
    for(i=0;i<8;i++)
    {
          rec[i]=0;
    }
    for(i=0;i<4;i++)
       {
          adc[i]=0;
       }

    i=0;
    //   Coefs();
    while(i<5){
          Conv();
          pres0=pres;
          p0=p;
          i++;
          if(altura<0)i--;
          if(altura>1)i--;
    }
    i=0;
    while(1){
          k=0;
          Conv();
    }
}   // end of main
void I2CA_Init(void)
{
    #if (CPU_FRQ_60MHZ)
      I2caRegs.I2CPSC.all = 5;      // Prescaler - need 7-12 Mhz on module clk d=5
    #endif
    I2caRegs.I2CCLKL = 45;// NOTE: must be non zero  60 Hz NOTE: must be non zero
    I2caRegs.I2CCLKH = 45;// f= 60 MHz / [ (IPSC+1)[(ICCL+d)+(ICCH+d)] ] = 100KHz
    I2caRegs.I2CIER.all = 0x18;      // Enable XRDY, RRDY interrupts
    I2caRegs.I2CMDR.all = 0x0020;    // Take I2C out of reset
                                     // Stop I2C when suspended

}
void Coefs()
{
       coef=1;
       I2caRegs.I2CSAR = 0x60;
       I2caRegs.I2CCNT = 1;
```

```
	I2caRegs.I2CMDR.all = 0x2620;   // NACKMOD dis=0, FREE dis=0, STT=1,
reserved, STP=0, MST=1, TRX=1, XA=0, RM=0, DLB=0, IRS=1, STB=0, FDF=0, 8bit
BC=000
	I2caRegs.I2CDXR=0x04;
	while(I2caRegs.I2CSTR.bit.ARDY==0);
	I2caRegs.I2CSAR = 0x60;
	I2caRegs.I2CCNT = 8;
	I2caRegs.I2CMDR.all = 0x2C20;   // NACKMOD dis=0, FREE dis=0, STT=1,
reserved, STP=1, MST=1, TRX=0, XA=0, RM=0, DLB=0, IRS=1, STB=0, FDF=0, 8bit
BC=000
	while(I2caRegs.I2CSTR.bit.BB == 1);
	coef=0;
}
void Conv(){
	conv=1;
	I2caRegs.I2CSAR = 0x60;
	I2caRegs.I2CCNT = 2;
	I2caRegs.I2CMDR.all = 0x2E20;   // NACKMOD dis=0, FREE dis=0, STT=1,
reserved, STP=1, MST=1, TRX=1, XA=0, RM=0, DLB=0, IRS=1, STB=0, FDF=0, 8bit
BC=000
	I2caRegs.I2CDXR=0x12;
	while(I2caRegs.I2CSTR.bit.SCD==0);
	for(b=0;b<50;b++)for(a=0;a<50;a++)for(i=0;i<50;i++);
	I2caRegs.I2CSAR = 0x60;
	I2caRegs.I2CCNT = 1;
	I2caRegs.I2CMDR.all = 0x2620;   // NACKMOD dis=0, FREE dis=0, STT=1,
reserved, STP=0, MST=1, TRX=1, XA=0, RM=0, DLB=0, IRS=1, STB=0, FDF=0, 8bit
BC=000
	while(I2caRegs.I2CSTR.bit.ARDY==0);
	I2caRegs.I2CSAR = 0x60;
	I2caRegs.I2CCNT = 4;
	I2caRegs.I2CMDR.all = 0x2C20;   // NACKMOD dis=0, FREE dis=0, STT=1,
reserved, STP=1, MST=1, TRX=0, XA=0, RM=0, DLB=0, IRS=1, STB=0, FDF=0, 8bit
BC=000
	while(I2caRegs.I2CSTR.bit.BB == 1);
	conv=0;
	p=(adc[0]<<2)|(adc[1]>>6);
	t=(adc[2]<<2)|(adc[3]>>6);
	d=b1+c12*t;
	d=d*p;
	e=b2*t;
	f=a0+d+e;
	c=65.0/1023.0;
	pres=f*c+50;

	temperatura=(t-498.0)/-5.35 +25.0;
	altura=0.99*(-
(temperatura+273.15)*(287.053)/(9.80665))*(log(pres/101.325));
}
interrupt void i2c_int1a_isr(void)
{
	if(I2caRegs.I2CSTR.bit.XRDY==1)
	{
		if(coef==1)I2caRegs.I2CDXR=0x12;
		if(conv==1)I2caRegs.I2CDXR=0x00;
```

```
	}
	if(I2caRegs.I2CSTR.bit.RRDY==1)
	{
		if(coef==1){
			rec[j]=I2caRegs.I2CDRR;
			j++;
		}
		if(conv==1){
			adc[k]=I2caRegs.I2CDRR;
			k++;
		}
	}
	PieCtrlRegs.PIEACK.all = PIEACK_GROUP8;
}
```

Anexo 7: Código para configurar y controlar el transmisor y receptor

```
#include "DSP28x_Project.h"
#include <stdlib.h>
#include <string.h>
void   SCI_Init(void);
void Sys_Init(void);
void Tx(void);
interrupt void sci_rxint_isr(void);
interrupt void sci_txint_isr(void);
char s[]="Hola Mundo! ", r[12];
int i=0,j=0,k=0,a,b,c;
void main(void)
{
	Sys_Init();
/*	while(1){
		while(SciaRegs.SCICTL1.bit.TXENA!=0);
//		if(i>11){
		i=0;
		Tx();
//		}
	}
*/
	while(1);
}   // end of main
void SCI_Init(void)
{
	SciaRegs.SCICCR.all=0x07;	// One stop bit =0, odd parity =0, parity
disabled=0, loop back dis=0, idle-line=0, 8 bit data= 111
	SciaRegs.SCICTL1.all=0x25;	// reserved, receive error interrupt
dis=0, sw reset dis=1, reserved, txwake dis=0, sleep dis=1, txena=0, rxena=1
	SciaRegs.SCIHBAUD=0;
//	SciaRegs.SCILBAUD=194;		// 9600 baud
	SciaRegs.SCILBAUD=48;		// 38400 baud
//	SciaRegs.SCILBAUD=31;		// 57600 baud
	SciaRegs.SCICTL2.all=0xC3;	// Txrdy flag=1, txempty flag=1,
reserved5-2, Rx/Bk int ena=1, tx int ena=1
	SciaRegs.SCIRXST.bit.RXERROR=1; // Banderas de error en la recepcion.
```

```
}
void Sys_Init(void)
{
	InitSysCtrl();
	InitSciGpio();
	DINT;
	InitPieCtrl();
	IER = 0x0000;
	IFR = 0x0000;
	InitPieVectTable();
	EALLOW;
	PieVectTable.SCIRXINTA = &sci_rxint_isr;
	PieVectTable.SCITXINTA = &sci_txint_isr;
	EDIS;
	SCI_Init();
	PieCtrlRegs.PIEIER9.bit.INTx1 = 1;
	PieCtrlRegs.PIEIER9.bit.INTx2 = 1;
	IER |= M_INT9;
	EINT;
	for(i=0;i<12;i++)r[i]=0;
}
void Tx(void)
{
	s[11]=k;
	SciaRegs.SCICTL1.bit.TXENA=1;
	SciaRegs.SCICTL1.bit.TXWAKE=1;
	SciaRegs.SCITXBUF=8;
	k++;
	if(k==255)k=0;
}
interrupt void sci_rxint_isr(void)
{
	if((SciaRegs.SCIRXST.bit.FE==1)||(SciaRegs.SCIRXST.bit.OE==1)||(SciaRegs.
SCIRXST.bit.PE==1)||(SciaRegs.SCIRXST.bit.BRKDT==1))
	{
		SciaRegs.SCICTL1.bit.SWRESET=0;
		PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;
		SciaRegs.SCICTL1.bit.SWRESET=1;
		SciaRegs.SCICTL1.bit.SLEEP=1;
		return;
	}
	if(SciaRegs.SCIRXST.bit.RXWAKE==1){
		if(SciaRegs.SCIRXBUF.bit.RXDT==8)SciaRegs.SCICTL1.bit.SLEEP=0;
		PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;
		return;
	}
	if(SciaRegs.SCIRXST.bit.RXRDY==1){
		r[j]=SciaRegs.SCIRXBUF.bit.RXDT;
		j++;
	}
	if(r[0]!=0x48)j=0;
	if(j==11)j=0;
	PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;
}
interrupt void sci_txint_isr(void)
```

```
{
	SciaRegs.SCITXBUF=s[i];
	i++;
	if(i==12){
		SciaRegs.SCICTL1.bit.TXENA=0;
	}
	PieCtrlRegs.PIEACK.all = PIEACK_GROUP9;}
```