



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

REQUERIMIENTOS, IMPLEMENTACIÓN Y VERIFICACIÓN DEL NANO-SATÉLITE
SUCHAI

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

TOMÁS IGNACIO OPAZO TORO

PROFESOR GUÍA:
MARCOS DÍAZ QUEZADA

MIEMBROS DE LA COMISIÓN:
CLAUDIO ESTEVEZ MONTERO
ALEX BECERRA SAAVEDRA

SANTIAGO DE CHILE
AGOSTO 2013

Resumen

Esta memoria se enmarca dentro del proyecto SUCHAI, el que busca diseñar, construir, integrar, programar, lanzar y operar un nano-satélite de estándar Cubesat con estudiantes de pregrado. En este contexto el objetivo de la memoria es formalizar, analizar y presentar: El proceso de diseño seguido hasta obtener los requerimientos. La arquitectura y operación de este. Y finalmente la verificación a las que fue sometido para comprobar su funcionamiento.

En la primera etapa se muestra cómo a partir de la misión y restricciones del proyecto se extraen los requerimientos y se reúnen en la llamada Matriz de Trazabilidad de Requerimientos (Requirements Traceability Verification Matrix - RTVM en inglés). En la segunda parte se detalla la arquitectura y funcionamiento del bus-SUCHAI, haciendo para ello una analogía con el modelo OSI. En la tercera y final se somete al satélite a pruebas parciales en el laboratorio, y a una prueba total en una cámara de alto vacío, buscando comprobar si se cumplen cada uno de los requisitos, usando para ello nuevamente la mencionada matriz.

Como conclusión de las tres etapas, se concluye que el SUCHAI aunque no satisface aún todos los requerimientos propuestos originalmente si lo hace con los más importantes. Lo que permite afirmar que se ha desarrollado un bus lo suficientemente modular, fiable y capaz para ser lanzado y operado en el espacio.

Por último, tres aportes del trabajo de memoria son destacados, las propuestas de mejoras para el bus-SUCHAI, la aplicación de herramientas de Ingeniería de Sistemas como la RTVM en un proyecto Cubesat y la propuesta de un proceso de diseño para desarrollos similares.

*Dedicado a mis padres,
a quiénes debo quien soy.
A mi amada Andrea por su
incondicional apoyo y cariño.
Y a mi querida hermana mayor.*

Agradecimientos

Agradecimientos a Andrés Espinosa y Marcos Flores por su disposición y colaboración en las pruebas de alto vacío. Agradecer también a Miguel Patiño, por su gratuita amabilidad y apoyo en herramientas del laboratorio de electrónica. Finalmente agradecer a Accel Abarca, Alex Becerra, Luis Herrera, Marcos Diaz, Carlos Gonzales, Juan Piña, Francisco Reyes y todos los demás integrantes del laboratorio del SUCHAI por hacer de aquél un excelente lugar.

Tabla de contenido

1. Introducción	1
1.1. Objetivos de la memoria	2
1.2. Estructura de la memoria	3
2. Marco teórico y contexto	4
2.1. Procesos de Diseño y sus herramientas	4
2.1.1. Proceso de diseño en Cascada	4
2.1.2. Proceso de Diseño en el área satelital	5
2.1.3. Matriz de Trazabilidad de Requerimientos	6
2.2. Arquitectura de un satélite	7
2.2.1. Estructura	7
2.2.2. Estados de operación	8
2.3. Estándar Cubesat	12
2.3.1. Cubesat Design Specifications	12
2.4. Estado del arte en proyectos Cubesat	13
2.4.1. Aspectos positivos	13
2.4.2. Aspectos negativos	13
2.4.3. Proyectos Cubesat estudiados	14
2.4.4. Compañías relacionadas	14
3. Proceso de diseño del SUCHAI	17
3.1. Antecedentes iniciales	17
3.1.1. Misión y Restricciones del proyecto	18
3.2. Componentes y su integración	19
3.2.1. Componentes adquiridos	19
3.2.2. Componentes construidos	19
3.2.3. Trabajo a realizar	20
3.2.4. Guías de diseño	21
3.3. Requerimientos de la arquitectura	21
3.3.1. Mecanismo de generación/recolección de requerimientos	22
3.3.2. Registro de resultados en la Matriz de Trazabilidad de Requerimientos	23
3.4. Discusión de la Etapa de Diseño del SUCHAI	23
4. Implementación del bus SUCHAI	27
4.1. Arquitectura del bus SUCHAI	27
4.1.1. Capa física	28

4.1.2. Capa de enlace	33
4.1.3. Capa de red	37
4.1.4. Capa de transporte	40
4.1.5. Resumen	44
4.2. Estados de operación del SUCHAI	46
4.2.1. Despliegue	46
4.2.2. Operación Nominal	50
4.3. Discusión de lo implementado	56
4.3.1. Limitaciones del SUCHAI	56
4.3.2. Lecciones aprendidas	57
4.3.3. Variantes a explorar	58
5. Pruebas al nano-satélite SUCHAI	61
5.1. Etapa de Verificación	61
5.1.1. Métodos de Verificación	61
5.1.2. Pruebas realizadas	62
5.1.3. Resultados	64
5.1.4. Resumen	67
6. Conclusión	69
6.1. Proceso de Diseño seguido y recomendado	69
6.2. Análisis de lo realizado y trabajos futuros	71
6.2.1. Transceiver	71
6.2.2. Kit Pumpkins	71
6.2.3. Bus de datos	72
6.2.4. Sistema de Archivos FAT	72
6.2.5. OBC	72
6.3. Resumen	72
Bibliografía	74
A. Glosario	79
B. Imágenes	82
C. Pruebas del SUCHAI	83

Índice de tablas

3.1. Componentes adquiridos para el satélite	19
3.2. Componentes construidos para el satélite	20
3.3. Componentes en el satélite	20
3.4. Componentes en la Estación Terrena	21
3.5. Guías de diseño del SUCHAI	21
4.1. Aclaración del doble uso de la palabra bus	28
4.2. Lista de componentes más importantes, función y pertenencia jerárquica	32
4.3. Componentes del SUCHAI y sus protocolos de comunicación	36
4.4. Comandos del SUCHAI y su descripción (R/W/N=Lectura/Escritura/Ninguna sobre alguna Variable de Estado)	39
4.5. Control de telemetría (frames) para el SUCHAI	55
5.1. Resumen de requisitos imprescindibles	65

Índice de figuras

1.1. Cubesat SUCHAI	2
2.1. Diagrama del diseño en cascada	5
2.2. Fases del ciclo de diseño de una misión. NASA	6
2.3. Fases del ciclo de diseño de una misión. ESA	6
2.4. Proceso de diseño. SMAD	7
2.5. Pumpkin's Cubesatkit (extraído desde www.pumkpkins.com)	15
2.6. Interconexión de placas en un bus PC104	16
3.1. Organigrama de la simulación de la vida SUCHAI	23
3.2. Extracto del documento RTVM del SUCHAI (mitad derecha)	24
3.3. Extracto del documento RTVM del SUCHAI (mitad izquierda)	25
3.4. Resumen del proceso de diseño del SUCHAI	26
4.1. Bus PC104 y pines controlados por el OBC (en gris)	30
4.2. Placa de Antenas	33
4.3. Extracto del documento para coordinación del bus PC104 del SUCHAI . . .	35
4.4. Arquitectura del bus SUCHAI	41
4.5. Resumen de la Arquitectura del bus SUCHAI	45
4.6. Diagrama de flujo de dep_silent_time	47
4.7. Diagrama de flujo de dep_init_Peripherals	47
4.8. Diagrama de flujo de dep_init_Repos	48
4.9. Diagrama de flujo de dep_deploy_antenna	48
4.10. Diagrama de flujo de dep_launch_tasks	49
4.11. Diagrama de la maquina de estado de payloads	52
4.12. Diagrama de la maquina de estado de payloads (arriba) y la de telemetría (abajo)	54
4.13. Ejemplo de telemetría recibida desde el SUCHAI (extracto)	56
5.1. Instalación en la cámara de vacío	63
5.2. Prueba en la cámara de vacío (1)	63
5.3. Prueba en la cámara de vacío (2)	64
5.4. Prueba en la cámara de vacío (3)	64
5.5. Momento del despliegue de antenas	65
5.6. Actualización de variables de estado	66
5.7. Ejecución de Payloads	66
5.8. Influencia de la TM bajo demanda sobre la ejecución de payloads	67

5.9. Decodificación (parcial) de TM	67
6.1. Resumen proceso de diseño recomendado	70
B.1. Documentación en html usando Doxygen del Código fuente el software SUCHAI	82

Capítulo 1

Introducción

La historia de los satélites comenzó el 4 de Octubre de 1957 cuando la en ese entonces Unión Soviética lanzó exitosamente el Sputnik. Desde esa fecha más de 6500 de ellos han orbitado el espacio con variados propósitos, desde demostración tecnológica hasta exploración espacial. Históricamente y hasta hoy en día, la gran mayoría de estos han sido concebidos para lograr comunicaciones a distancia o para la observación de la tierra, 58 % y 46 % respectivamente [42].

Aunque es difícil hacer clasificaciones en la actualidad estos artefactos son en promedio piezas de ingeniería sobre los 300 kg, de unos 3x3x4 metros de envergadura sino más, con un costo de construcción de al menos 50 millones de dólares y que demoran varios años en ser ensamblados y probados. Es por esta razón que el desarrollo espacial ha sido durante mucho tiempo un área onerosa en términos económicos y técnicos [42], [43].

En este contexto de alta complejidad y costos en construcción y lanzamiento, una de las iniciativas más interesantes que han surgido para permitir un acceso más fácil al espacio es la de los llamados Cubesat. Esta iniciativa es aún predominantemente académica, sin embargo es parte de un nuevo impulso hacia la miniaturización y abaratamiento de estos artefactos [33]. Ideado en 1999 entre la California Polytechnic State University y Stanford University, con el objetivo de "crear un diseño estándar de pico-satélites para reducir costos y tiempos de desarrollo, incrementar el acceso al espacio y mantener altas tasas de lanzamientos" [22]. Debido a sus características estos aparatos suelen ser de menor complejidad que los comerciales, sin embargo todo satélite ya sea grande, mediano o pequeño estará enfrentado a ambientes, situación y restricciones muy parecidas, las cuales debe ser capaz de sobrellevar. Dentro del mundo de los satélites existe distintas clasificaciones según peso y tamaño. Dentro de esta, una de las menores corresponde a un nano-satélite, es decir un satélite con un peso de entre 1 y 10 kg. Con un peso máximo de 1.33 kg un Cubesat es un tipo de nano-satélite.

1.1. Objetivos de la memoria

Esta memoria se enmarca dentro del proyecto SUCHAI, el que busca diseñar, construir, integrar, programar, lanzar y operar un nano-satélite de estándar Cubesat por parte de estudiantes. Este se inició formalmente en marzo del 2011 y a la fecha de Agosto de 2013 esta próximo a ser terminado y posteriormente lanzado.

Dada la inexistente experiencia a nivel universitario y nacional en este tipo de iniciativas, el objetivo implícito del proyecto consiste en aprender a construir, lanzar y operar un nano-satélite, y más específicamente uno de estándar Cubesat, contando con recursos y tiempos acotados. Con el proyecto se busca además, generar conocimiento en el área de construcción satelital en Chile al contar con un satélite capaz de superar las pruebas de certificación externas y dejar como legado un bus lo suficientemente genérico como para integrar payloads (experimentos) de baja a mediana complejidad.

Gran parte de lo hecho en el SUCHAI fue guiado, desde muy temprano y durante todo el desarrollo del proyecto por lineamientos generales como modularidad, escalabilidad, fácil integración y reutilización. Sin embargo una parte de lo realizado se racionalizó ex-post, en especial la etapa de diseño, donde los requerimientos y su proceso de recolección fueron formalizados como parte del presente escrito.

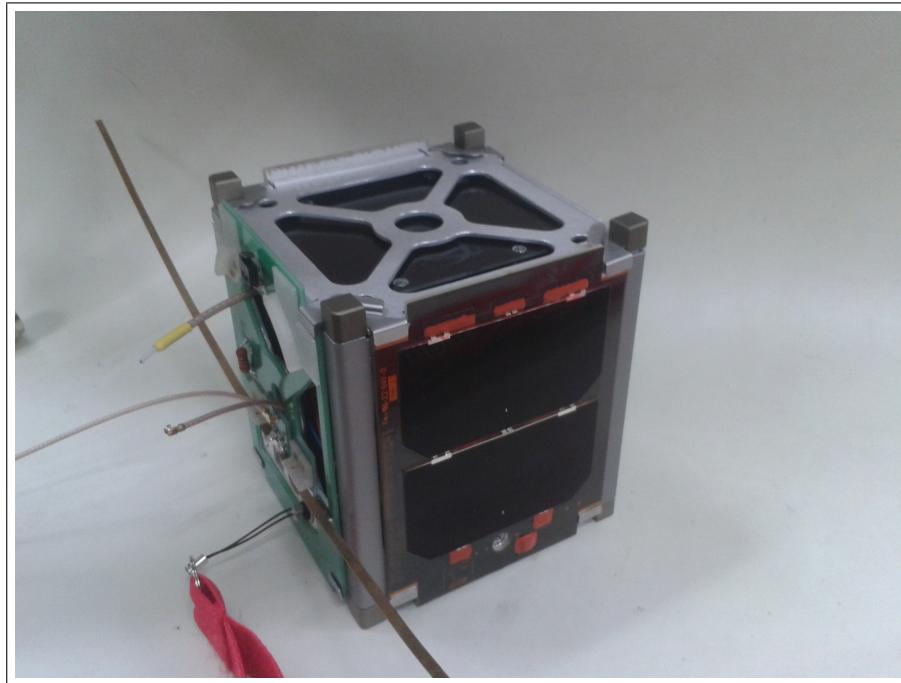


Figura 1.1: Cubesat SUCHAI

En concreto, el objetivo de esta memoria es **formalizar y justificar** el proceso que llevo al SUCHAI desde la declaración de su misión hasta su fase de pruebas, todo esto limitado al satélite mismo y no a la estación terrena u otros resultados. Esta memoria se enfoca en el **diseño, la arquitectura y las pruebas del bus-SUCHAI**, si el lector busca una descripción de cómo se llegó a la solución propuesta y/o como esta fue finalmente implementada, en especial la programación del computador a bordo (OBC) se sugiere leer también la memoria

de Carlos Gonzales [21].

Los objetivos principales de esta memoria son formalizar, analizar y presentar:

1. El proceso de diseño del SUCHAI extrayendo los requerimientos a partir del enunciado de la misión, sus restricciones y guías de diseño. Documentando finalmente los resultados en una Matriz de Trazabilidad de Requerimientos (RTVM).
2. Describir en detalle que es el bus-SUCHAI, sus capacidades y limitaciones, como fue implementado y que características posee.
3. Realizar pruebas al sistema y contrastar los resultados con los requerimientos de manera de comprobar si estos se cumplen o no. Esto implica incluir cada prueba realizada dentro del documento de la Matriz de Trazabilidad de Requerimientos de manera de saber que cada uno de ellos es cubierto.

1.2. Estructura de la memoria

En este capítulo se mencionó el mundo de los satélites, la iniciativa Cubesat y el desafío que se le presentó al equipo del SUCHAI de construir uno, así como los objetivos de la memoria. En el segundo capítulo, el marco teórico, se habla del estándar Cubesat, de distintos procesos de diseño que se encuentran en la literatura del área espacial, de la arquitectura más común en satélites, y finalmente de modelos de pruebas.

Luego le sigue el cuerpo de la memoria, siendo el tercer capítulo el del proceso de diseño, que describe los objetivos principales y secundarios del proyecto, y cómo se fueron tomando decisiones a lo largo de éste, hasta plantear los requisitos. Luego el cuarto capítulo que trata de la implementación, en él se describe la arquitectura detallando que la compone, cómo funciona, cuáles son sus capacidades y cuáles sus limitaciones. El último capítulo del cuerpo central es el quinto, está dedicado a las pruebas y en él se verifica y demuestra si lo implementado (la arquitectura y operación del SUCHAI) cumple efectivamente con los requisitos y objetivos establecidos en la etapa de diseño.

Finalmente el capítulo sexto correspondiente a la concluye respecto a si lo implementado satisface lo requerido, que trabajos quedan pendientes y que perspectivas de desarrollos posteriores hay.

Además de la Bilbiografía y los Apéndices se agrega un glosario para las siglas y palabras de contenido técnico, mas usadas en este memoria, ver apéndice A.

Capítulo 2

Marco teórico y contexto

2.1. Procesos de Diseño y sus herramientas

En distintas áreas de la ingeniería existen métodos o reglas generales, llamados **Patrones de Diseño, Ciclos de Diseño o Procesos de Diseño**. Son propuestas basadas en la experiencia acumulada por otros, y evolucionan continuamente. Una definición sugerida es « En ingeniería, el proceso de diseño es la formulación de un plan para ayudar a uno o más ingenieros a desarrollar un producto bajo determinados estándares de desempeño. Este proceso se compone de una serie de pasos, y partes de ellos pueden necesitar ser repetidos múltiples veces antes de llegar a la iteración final, con la que la producción del producto puede empezar» [1].

El modelo más conocido en Ingeniería [5] está orientado a proyectos con una estructura mandante-mandatado, consiste en sucesivas etapas de Investigación, Conceptualización, Estudio de Factibilidad, Establecimiento de Requerimientos, Diseño Preliminar o Ingeniería Básica, Diseño de detalle o Ingeniería de Detalle, Planificación de Producción y finalmente Producción. Sin embargo en áreas más específicas surgen variantes, en Programación de Software a este proceso se la llama **Ciclo de Vida del Software**, e incluye subtipos como: Modelo Cascada [2], Modelo Agil [3], Modelo Espiral [4]. Otra área es la de sistemas embebidos [7], que es la que más se aplica en el desarrollo de un Cubesat. Esta área es un punto de confluencia entre la Ingeniería Eléctrica y la Computacional, pues usa técnicas de hardware y software a la vez.

2.1.1. Proceso de diseño en Cascada

Proveniente del mundo del Software, el proceso de diseño en cascada consiste en cinco pasos secuenciales, a veces iterativos. Los pasos son: Requerimientos, Diseño, Implementación, Verificación y Mantenimiento. En Requerimientos se extraen los objetivos, funcionalidades y estándares que el sistema a diseñar debe cumplir. En Diseño se crea/decide la mejor arquitectura

tura que solucionará el problema. En la etapa de Implementación se construye y lleva a cabo lo diseñado. En Verificación se realiza la comprobación de que los requerimientos planteados en la primera etapa sean realmente satisfechos. Y finalmente la etapa de mantenimiento, en donde se llevan acabo todas las mejoras incrementales y correcciones que el producto final necesite.

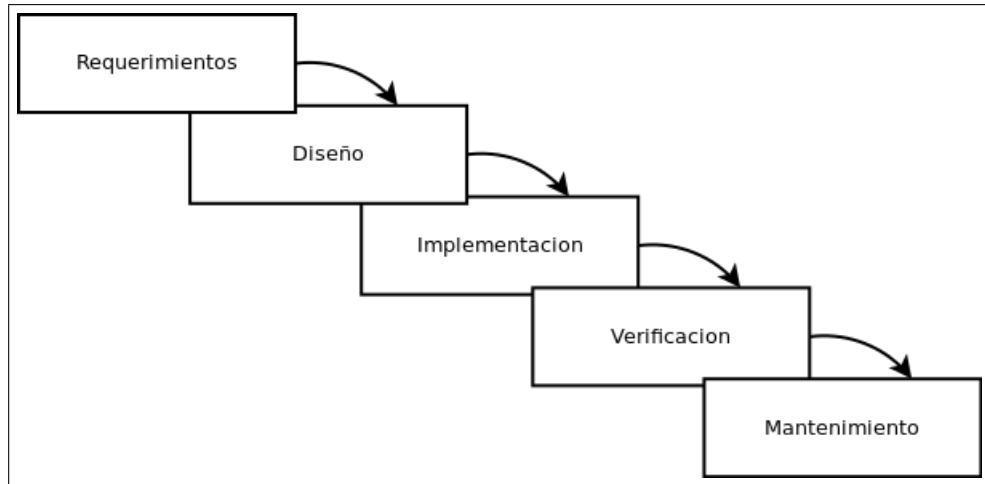


Figura 2.1: Diagrama del diseño en cascada

2.1.2. Proceso de Diseño en el área satelital

A pesar de que los procesos de diseños más conocidos son aquellos relacionados al área de software, el área satelital posee más de uno, de hecho cada agencia espacial tiene el suyo propio, por ejemplo NASA [35] en la figura 2.2, o ESA [34] en la figura 2.3. Estos procesos de diseño buscan analizar la factibilidad de la misión, luego de lo cual entran a una o varias etapas de estudio y luego de aprobadas pasan a la de construcción. La importancia de estos procesos radica en que permiten separar nítidamente, **para que** se quiere un satélite, del **como se hará** y finalmente del **quien lo hará**.

Proceso de diseño SMAD

Evitando seguir a priori el proceso de una agencia en particular, se presenta este proceso basado en el libro Satélite Mission Analisis Design [33]. Según este, el ciclo de diseño para un satélite tiene como objetivo, **determinar los requerimientos y luego diseñar el sistema que los cumpla al menor costo y riesgo (eficientemente)**. Ver figura 2.4.

El proceso de diseño, es una serie formal de pasos, en dónde a grandes rasgos, expertos analizan la utilidad de una determinada misión, proponen arquitecturas que la cumplan y las comparan. Si el proyecto es declarado factible y aprobado, se pasa a la etapa del diseño mismo de la arquitectura escogida. Finalmente ese diseño es construido y es en esta etapa en la que se realiza la ingeniería de detalle, en donde se traducen los objetivos generales y necesidades en especificaciones y requerimientos que son posteriormente verificados.

Pre-Fase A	
Estudio Conceptual	
Fase A	
Análisis Preliminar	
	Revisión de los Requerimientos de sistema
Fase B	Revisión del Diseño del Sistema
	Revisión imparcial
Definición	
	Revisión del Diseño Preliminar
Fase C/D	Revisión Crítica del Diseño
	Revisión del Estado de Preparación
	Revisión del Estado de Preparación para volar
Diseño y Desarrollo	
	Misión principal
Fase de Operación	Misión extendida
Operación de la misión y Análisis de datos	

Figura 2.2: Fases del ciclo de diseño de una misión. NASA

Actividades	Fase 0	Fase A	Fase B	Fase C	Fase D	Fase D/E Transición	Fase E	Fase F
Misión	x	x						
Requerimientos	x	x	x					
Definición			x	x				
Verificación				x	x	x		
Producción				x	x			
Utilización							x	
Desechado								x

Figura 2.3: Fases del ciclo de diseño de una misión. ESA

Cabe recordar que al igual que los procesos de diseño de cada agencia espacial, el SMAD está pensado para misiones de mayor alcance y costo, donde antes de tomar la decisión de construir un satélite se ha pasado por varios diseños preliminares y alternativas distintas para completar la misión.

2.1.3. Matriz de Trazabilidad de Requerimientos

Independiente del proceso de diseño escogido, en todos ellos existe una etapa de pruebas en donde se verifica el cumplimiento o no de los requisitos. Esto puede realizarse a través de una matriz de trazabilidad de requerimientos (del inglés Requirements Traceability Verification Matrix - RTVM) [6]. Esta herramienta, es una matriz en la que se puede seguir y registrar la vida de un requerimiento y a que otros entes está ligado, vale decir, es una relación, matemáticamente hablando, no necesariamente unívoca entre requerimientos y otros elementos. Principalmente se asocia cada requerimiento con los documentos que justifican su origen y el test que lo comprueba. En este sentido, un solo test puede cubrir más de un requerimiento, pero todo requerimiento debe estar cubierto por un test, y esta condición es la que una

Retroalimentación		Pasos	
--->	Definir objetivos	1	Definir objetivos generales y restricciones Estimar cuantitativamente las necesidades y requerimientos de la misión
		2	
--->	Caracterizar la misión	3	Definir alternativas conceptuales para la misión
		4	Definir alternativas de arquitecturas para la misión
		5	Identificar parámetros relevantes de cada arquitectura
		6	Caracterizar el concepto y arquitectura que tendrá la misión
<--	Evaluar la misión	7	Identificar requerimientos críticos
		8	Evaluar la utilidad de la misión
		9	Definir el concepto de la misión (concepto preliminar)
<--	Definir requerimientos	10	Definir arquitectura
		11	Asignar requerimientos a los elementos de la arquitectura

Figura 2.4: Proceso de diseño. SMAD

RTVM permite visualizar fácilmente.

2.2. Arquitectura de un satélite

A pesar de la gran variedad en formas y misiones que tiene muchos satélites, la gran mayoría de ellos tiene una estructura y funcionamiento más parecido de lo que podría pensarse [33]. En lo siguiente se hará una descripción del llamado **satélite típico** que es aquel que tiene esas características.

2.2.1. Estructura

Un satélite está diseñado para cumplir cierto propósito y el término para describir ese propósito es el de **misión**. Los equipos abordo que tienen directa relación con la misión se llaman **Payloads**, por ejemplo las cámaras para tomar fotos, sensores que detectan nubes, antenas repetidoras para comunicaciones, etc. A todo el resto de los equipos que facilitan y permiten cumplir la misión se le llama el **bus satelital**.

El satélite se comunica con una base en tierra llamada **Estación Terrena**, la cual envía **Telecomandos** que le indican al satélite qué acciones tomar, y este responde con **Telemetría**, la cual indica la respuesta a los telecomandos y entrega los datos recopilados por los diferentes payloads.

La comunicación se produce generalmente usando alguna codificación digital, como FSK, QPSK, GFSK etc. Las frecuencias utilizadas son desde cientos de MHz, hasta unas decenas de GHz, para ellos son necesarias antenas tanto en el satélite como en la estación terrena.

El equipo abordo del satélite encargado de la comunicación es llamado transmisor, radio, o más precisamente **transceptor** por ser receptor y transmisor a la vez (en inglés es llamado **transceiver** y es abreviado como TRX).

Como todo equipo, el satélite necesita de energía para operar, para ello la radiación electromagnética del sol es captada por paneles solares y transformada en energía eléctrica, que se almacena en baterías. Cuando el satélite queda a oscuras la energía almacenada en las baterías provee al sistema para seguir operando.

Cada uno de los componentes recién mencionados forma parte de los siete subsistemas que normalmente posee cada satélite [33]. Un resumen de la función de cada uno es la siguiente:

- Propulsión: Subsistema que proporciona de impulso para ajustar la órbita y orientación, además de manipular el momento angular, lo cual realiza usando propelentes.
- ADCS Attitude determination & control system: Es el subsistema encargado de determinar la orientación y la órbita del satélite, además de apuntar sus diferentes partes (como antenas hacia la tierra y paneles solares hacia el sol).
- Termal: Encargado de mantener el equipamiento a temperaturas adecuadas.
- Estructura y Mecanismos: Soporte físico para los equipos y sistemas. En él se incluyen las partes móviles, especialmente importante es el caso del despliegue de la(s) antena(s).
- Comunicación: Destinado a proveer la interfaz análoga-digital digital-análoga para telecomandos y telemetría.
- C&DH: Unidad encargada de recibir telecomandos desde el Subsistema de Comunicaciones y repartirlos a otros subsistemas, o ejecutarlos directamente. Procesa, almacena y da formato a los datos que serán posteriormente transmitidos a tierra. Generalmente contiene al Computador a Bordo (On Board Computer - OBC en inglés).
- Energía: Genera o recolecta, almacena, regula y distribuye energía eléctrica a los demás subsistemas.

2.2.2. Estados de operación

Desde antes su activación (encendido del sistema) hasta su muerte, ya sea accidental o programada, un satélite pasa por una serie de etapas. Estas tienen distintos períodos de duración y obviamente lo que se pretende es que la llamada etapa de **operación** es decir, la de funcionamiento normal, sea lo más extensa posible. La secuencia por la que todos los satélites pasan es:

- Integración al cohete y lanzamiento
- Despliegue / Puesta en servicio / Inserción en órbita

- Operación nominal
- Fin de vida útil (decommissioning)

Integración al cohete y lanzamiento

Los cohetes están compuestos de múltiples etapas, generalmente 2 o 3, siendo la ultima la encargada de liberar al satélite y dejarlo en la órbita correcta. El espacio físico en que se insertan es parte de la etapa superior del lanzador, y es llamada **cofia**, es generalmente de 4 o 5 metros de diámetro, lo que implica que tanto antenas como paneles solares deben ir plegados. A modo de ejemplo el satélite Alpha Bus [45], tiene una envergadura de 14 metros con sus paneles solares desplegados, esto implica contar con mecanismos especiales para ello.

Despliegue / Puesta en servicio / Inserción en órbita

Luego de liberado por la etapa superior o el mecanismo de despliegue, el satélite vive su primer momento de vida independiente, hasta ese entonces, este tenia la posibilidad de recibir energía desde las baterías de la etapa superior del cohete. Sin embargo desde este momento está a su suerte. Esta segunda etapa de la vida de un satélite, se extiende desde su despliegue por parte del cohete hasta su entrada en **operación nominal** o **entrada en servicio**.

Cuando los satélites no cuentan con un subsistema de propulsión (2.2) no pueden modificar su órbita, y quedan en la órbita en que la etapa superior los dejó. Ahora bien, cuando cuentan con un subsistema de propulsión, pueden cambiarla autónomamente. Esto lo realizan casi siempre los satélites geoestacionarios, que son dejados por el lanzador en una órbita intermedia llamada de **órbita de transferencia**. Encendiendo luego sus motores hasta llegar a una órbita geosíncrona [33] (de casi 36.000 km).

En esta etapa, también deben inicializar cada uno de sus componentes, verificar el correcto funcionamiento de los subsistemas y payloads. Así mismo deben desplegar sus paneles solares y antenas y establecer comunicación con la estación terrena. **Todas estas comprobaciones y acciones para pasar a un estado operacional pueden ser hechas de manera autónoma por el satélite o desde tierra, a través de telecomandos.** Un resumen de ellas es el siguiente.

1. Desplegar paneles solares y dado el caso apuntarlos al sol
2. Desplegar antenas y dado el caso apuntarlas a la tierra
3. Iniciar correcciones de orientación del satélite (attitude control)
4. Comprobar estado de subsistemas
5. Comprobar estado de payloads
6. Desplegar e inicializar payloads

7. Iniciar maniobras orbitales en caso de ser necesarias

Operación nominal

Esta siguiente etapa es en la que el satélite hace aquello para lo que fue ideado. En esta situación el satélite puede actuar **Bajo Demanda** o de modo **Autónomo**, sin embargo el modo en que terminan actuando es una combinación de ambas. No es posible que un satélite opere completamente en modo Bajo Demanda, pues nada sucedería con uno al que tenga que indicarse desde tierra por telecomandos, que debe desplegar sus antenas para poder establecer comunicación con esa misma estación. En el otro extremo, un satélite 100 % autónomo podría ser lo deseable, pero su programación debe ser extremadamente robusta y es más compleja de desarrollar y verificar que un sistema más pasivo [13]. Finalmente, el ente que determina el porcentaje de esa mezcla es el llamado **software de vuelo**, en donde se implementa que tanta capacidad Bajo Demanda y de Autonomía tendrá el satélite.

Modo Bajo Demanda (On Demand):

En este modo, la estación terrena obtiene los parámetros orbitales del satélite obtenidos, por ejemplo, a través del NORAD [40] y/o del lanzador. Luego y por medio de un software predictor de órbita se determina la **trayectoria y duración de la pasada**, se procede a **seguir** (tracking en inglés) al satélite apuntando la antena de la estación terrena hacia él. Luego, el equipo en tierra, envía telecomandos por medio del enlace de comunicaciones de subida del bus, el satélite las recibe y responde con telemetría por medio de un enlace de comunicaciones de bajada del bus. En algunos casos, los payload pueden contar con un enlace de bajada exclusivo de payload, que son generalmente de alta velocidad para descargar muchos datos.

Modo Autónomo:

En este modo, el satélite reacciona ante eventos que se producen en una secuencia no definible a priori. Este tipo de comportamiento se puede programar como un agregado a la operación On Demand y consiste en mayor o menor cantidad de situaciones en las que el aparato responde automáticamente sin necesidad de supervisión ni control [13]. Se podrían definir dos tipos de eventos de importancia que deben desencadenar respuestas.

Una fuente de estos eventos son los producidos por la influencia del ambiente de la órbita sobre los componentes del equipo. Ejemplos de estos son niveles críticos de carga en las baterías, lo que demanda apagar u operar en modo de ahorro o hibernación a subsistemas dispensables, como payloads, ADCS, termales y eventualmente comunicaciones. Otros podrían ser el reinicio accidental del computador a bordo (aquel que maneja el C&DH).

La otra fuente de eventos de importancia son los fenómenos relacionados con la misión de satélite. Si este es uno de comunicaciones, la llegada de una señal a transmitir debe desencadenar la activación del payload adecuado y su ejecución. Si el satélite fuese uno de detección de incendios forestales, debiese capturar imágenes y registrar con sensores el hecho, y además reportarlo mediante telemetría a la estación de tierra más cercana.

Fin de vida útil (decommissioning)

La última etapa de vida de un satélite se considera aquella a partir de la cual el satélite no puede prestar, o bien se elige que no prestará más el servicio para el cual fue concebido. La primera apunta a un caso de **fallo** y la segunda a una decisión planificada desde la estación terrena, a esta decisión se le llama **desmantelamiento** o **desguace** del satélite, proveniente de la palabra técnica **decommissioning** (en inglés).

Fallo:

El fin de vida útil por fallo es aquel en que el satélite ya no puede seguir efectuando su misión **producto de un evento inesperado y/o repentino**, frente al cual no se preparó un plan de acción. Frente a estos eventos, el equipo en tierra pierde parte o todo el control sobre el aparato y este queda a merced de las condiciones de la órbita. Los eventos asociados al espacio más conocidos y sus fallos asociados son los siguientes:

- **Radiación cósmica:** Partículas de alta energía impactan contra el computador a bordo (C&DH) u otro componente electrónico, destruyéndolo o dejándolo en un estado en que no puede efectuar más operaciones (los primeros satélites se vieron severamente afectados por este fenómeno [9]).
- **Colisiones:** Destrucción de parte del satélite por impacto de otro objeto. Estos fragmentos pueden ser naturales, o restos de otros satélites (el primer caso fue el registrado en 2009 entre el Kosmos 2251 y el Iridium 33 [38]).
- **Errores de programación** Debidos a estados no previstos ni verificados con anterioridad (la fallida sonda Fobos-Grunt habría fracasado en encender sus motores para cambiar de órbita debido a un fallo de programación, la sonda terminó estrellándose frente a las costas del sur de Chile [39]).

Luego de ocurridos estos, el satélite deja de prestar las mismas funcionalidades, pudiendo quedar **parcialmente operativo**, o **no-operativo**, en cuyo caso pasa a formar parte de la creciente **basura espacial**. Por ejemplo, un satélite en órbitas bajas puede demorar 25 años en decaer producto del roce atmosférico y reentrar a la tierra incinerándose en el camino.

Fin de vida programada

Este tipo de sucesos ocurre en general con satélites geoestacionarios. Estos, ocupan espacio en un anillo sobre el ecuador de 35,786 km de radio y debido a que esta órbita es limitada, su uso ha sido sujeto a regulación. Por lo cual, hoy en día, los satélites que se encuentran en esta órbita y que estén próximo al fin de su vida útil, **son alejados a una órbita superior**, pues es energéticamente más económico que hacerlos volver y caer sobre la tierra, y **dejados en estado de actividad mínima** (haciendo una analogía con estado de hibernación de un PC) [10]. El fin de vida útil de estos satélites es estimada por el fabricante, y los factores principales para su cálculo son los siguientes:

- **Baterías:** El ciclo de carga y descarga es limitado, por tanto llegado el momento, la batería, por más energía que capten los paneles solares, no podrá retenerla.
- **Propulsión:** Debido a que estos satélites requieren un alto grado de precisión en su

orientación hacia la tierra (ADCS), necesitaran de mecanismos de ajuste. Si estos son a base de propelentes el fin de estos supondrá la perdida de capacidad de orientación (también llamado **control de actitud**).

Y aunque no lo fuese, un satélite geoestacionario, necesita reajustar su órbita (esta se degrada ya no por rastros de atmósfera como en LEO, sino por influencia solar, entre otros factores [33]), y eso lo permite el sistema de propulsión y la capacidad su estanque,

- **Deterioro por temperatura** Debido a la ausencia de atmósfera, toda la disipación de energía térmica es por radiación, lo que implica temperaturas muy elevadas en los circuitos, en comparación a lo que les sucedería en tierra. Por ellos, y a pesar de que gran parte del esfuerzo de diseño de satélites apunte a ello, la capacidad de disipación decae con el tiempo, por lo que tarde o temprano los componentes entraran a operar en zonas fuera de rango [10].

2.3. Estándar Cubesat

Un Cubesat es una clase de satélite pequeño, técnicamente es un nano-satélite, utilizado con fines de investigación. En su tamaño estándar llamado **1U** es un cubo de 10cm de lado, es decir 1 litro de volumen, con una masa máxima de 1.33 kg. Existen variantes llamadas 2U y 3U donde uno de los lados pasa a ser de 20 cm y 30 cm respectivamente [23]. El estándar nació en 1999 entre la Universidad Politécnica de California (Cal Poly) y la Universidad de Stanford de manera de ayudar a las universidades en sus iniciativas satelitales. Originalmente propuesto por el profesor Jordi Puig-Suari de Cal Poly y Bob Twiggs de la Universidad de Stanford, el proyecto partió con el objetivo de "enseñar a estudiantes las capacidades para diseñar, construir, verificar y operar un satélite con capacidades parecidas a las del Sputnik" [23]. El estándar Cubesat fue pensado con fines académicos, de modo de "crear un diseño estándar de pico-satélites para reducir costos y tiempos de desarrollo, incrementar el acceso al espacio y mantener altas tasas de lanzamientos" [22]. El primer Cubesat fue lanzado el 2003 y al 2012 se han lanzado 75, principalmente proyectos universitarios y de demostración tecnológica [23].

2.3.1. Cubesat Design Specifications

El documento técnico que recoge los diferentes requerimientos que un satélite debe cumplir están recogidos en el documento CDS [22]. Alguno de los más importantes son:

- Debe ser un cubo 10 cm por cada lado.
- Debe pesar menos de 1.33 kg.
- La energía química acumulada no debe ser superior a los 100 Watt/hora.
- Debe tener rieles para su contacto con el dispositivo que los libera una vez en órbita

(llamado P- POD).

- Todo artefacto no debe exceder los 6,5 mm a la superficie del satélite.
- Todos los sistemas de despliegue deben hacerse después 30 minutos del lanzamiento desde el P-POD.

2.4. Estado del arte en proyectos Cubesat

Para esta sección se han analizado los casos de seis desarrollos, el KYSAT, AAUSAT, HAUSAT, ICUBE, PicPot y el PolySat [25], [26], [27], [28], [29], [30]. Además de lo anterior se ha extraído información desde informes de la comunidad Cubesat, [31].

2.4.1. Aspectos positivos

El impacto de los Cubesat ha sido muy importante y variado, el principal propósito al que han contribuido son el de demostración tecnológica y de conceptos [31] haciendo de los satélites pequeños (nano y pico satélites) una alternativa real para múltiples objetivos [33], lo que ha llevado la industria a concebir misiones que sigan el principio de "Smaller, Cheaper, Faster, Better" o en español, "Más pequeño, más barato, más rápido y mejor". Por otro lado, la utilización de productos comerciales para fines espaciales (llamado COTS Commercial Off The Shelf) ha sido otro de los aportes. Sin embargo hay componentes comerciales que debido al tamaño y potencia energética limitada de los Cubesat se han debido desarrollar específicamente, tales son las radios para comunicaciones entre el satélite y la tierra, los sistemas de trackeo y posicionamiento, y los de propulsión. En ellos existe un gran impulso y son proyectos de vanguardia [31].

En resumen los Cubesat han demostrado ser una alternativa real y efectiva (dentro de sus limitaciones) para acceder al espacio con plazos y costos muy por debajo del promedio de otros desarrollos satelitales.

2.4.2. Aspectos negativos

Sin embargo no todo en los Cubesat es positivo, muchos de los desarrollos Cubesat son inorgánicos en su arquitectura pues no integran coherentemente los subsistemas. Esto es porque, aunque el desarrollo de ellos está separado en módulos (subsistemas termal, mecánico, energía, comunicaciones, command and data handler, posicionamiento y propulsión) no existe una manera ordenada de intercambiar información/comandos entre ellos, es decir, carecen de un bus de intercambio de información. Esto sucede principalmente debido a que los Computadores a Bordo en Cubesat son microcontroladores, que para ser versátiles incorporan

distintos protocolos de comunicación. Esto hace menos uniforme la comunicación interna del satélite y peor aún, genera una centralización en la arquitectura pues todo los subsistemas dependen del OBC y la comunicación entre dos ellos es solo posible a través del OBC.

Lo anterior llevado a la práctica significa que para comunicarse con un subsistema X se elija un protocolo A, pero para comunicarse con el subsistema Y haya que usar el protocolo B pues el microcontrolador no dispone de más puertos para el protocolo B. Véase los casos de [25], [26] y [27].

En otro aspecto, los subsistemas de Propulsión, termal y ADCS generalmente no existen en un nano-satélites, esto limita el tipo de payloads que podría portar el satélite como por ejemplo payloads que necesiten apuntar siempre hacia la tierra, o que requieran de correcciones en su órbita, o que necesiten medir con extrema precisión lo que implica controlar la temperatura. Sin embargo aún hay una gran cantidad de misiones que es posible y son interesantes de concretar.

2.4.3. Proyectos Cubesat estudiados

De entre los proyectos Cubesat estudiados para esta memoria destaca que la mayoría de sus software de control están basados en operaciones según estados. Como un estado inicial, comunicación, fallo, payloads, entre otros [25], [27], [26]. Por otro lado, la mayoría, o casi todos los proyectos de Cubesats incorporan estudiantes de Magisters y/o estudiantes de Doctorado. Algunos cuentan incluso con apoyo de agencias espaciales locales, además de estudiantes de pregrado [31].

2.4.4. Compañías relacionadas

Con el surgimiento de cada vez mas proyectos Cubesat en distintas parte del mundo, han surgido a la par desarrollos comerciales dedicados a proveer servicios y productos para estos satélites. Ejemplos de ello son paneles solares, equipos de comunicación, sistemas de posicionamiento y mecanismos y estructuras compatibles todos con el estándar. De entre las más conocidas están por ejemplo Clyde Space, empresa con base en Glasgow, Escocia son especialistas en sistemas de energía, baterías y paneles solares de alta eficiencia para pequeños y micro satélites. Gomspace, empresa nacida al alero de un desarrollo Cubesat de la Universidad de Aalborg en Dinamarca, especializada en transmisores, subsistemas y software. Mención aparte merece la compañía Pumpkins que desarrolló tempranamente un Kit para construir Cubesats, este se ha convertido con el tiempo en el estándar de facto para la mayoría de las misiones [37].

Se detalla en especial el producto ofrecido por Pumpkins por dos razones, la primera debido a que es la usada por el SUCHAI, la segunda es por ser la compañía que estableció el bus PC/104 (más adelante detallado) para la interconexión de plazas, patrón que todas las demás compañías han seguido hasta hoy.

Kit de Pumpkins

En sus propias palabras cuentan: *En el año 2000, estábamos asesorando estudiantes en la Universidad de Standfor y Universidad de Santa Clara, respecto a cómo usar nuestro software para sistemas embebidos, llamado Salvo RTOS para microcontroladores destinados a misiones en micro-satélites. Con el pasar del tiempo y una vez que el estándar Cubesat para pico-satélites gano adeptos, nos fuimos convenciendo que un kit para Cubesats con componentes comerciales podría mejorar ostensiblemente las probabilidades de cada misión de lograr un lanzamiento a tiempo y una misión exitosa. Fue así como el Kit Cubesat nació. Adicionalmente, nos percatamos que si ofrecíamos un ambiente de desarrollo con software y hardware integrados, los usuarios podrían desarrollar los componentes específicos de su misión sin tener que trabajar respecto a los requerimientos físicos del Cubesat.*

El kit de Pumpkins se compone de.



Figura 2.5: Pumpkin's Cubesatkit (extraído desde www.pumpkins.com)

- **Estructura:** El esqueleto es hecho de aluminio galvanizado, y con estrictas medidas de modo de satisfacer los 10 cm por cada lado. Sobre este cubo de metal se monta la placa madre, y se conectan los switches. Por fuera y sobre los lados de este cubo se instalan los paneles solares y antenas.
- **Switches:** Son exigencias del estándar Cubesat, y existen de dos tipos:
 - El primero es el llamado de **switch de despliegue**, su objetivo es el de mantener cortadas las líneas que van desde las baterías (cargadas antes del lanzamiento) hacia el resto de los sistemas. Una vez que el satélite se libera del cohete, el switch se suelta y empieza a fluir energía hacia el resto del sistema.
 - El segundo tipo, el llamado **switch RBF**, sigla de Remove Before Flight, es usado con el mismo objetivo (evitar que el Cubesat se energice cuando no debe) pero en la etapa de integración con el cohete, una vez terminado el proceso se retira este switch y el satélite queda armado y listo para, una vez liberado al espacio, iniciar su vida en órbita.
- **PPM:** Esta es la placa que contiene al microprocesador del satélite, los modelos

disponibles son: PIC24FJ256GA110 o dsPIC33FJ256GP710 de Microchip, C8051F120 de Silicon Labs, o los procesadores MSP430F(1612/1611/2618) de TI [36]. Además contiene osciladores y una memoria Flash concebida para reprogramar al satélite en órbita.

- **MB:** Es la sigla de Mother Board o placa madre, esta placa emula lo que sucede en un PC siendo la interfaz entre el procesador (PPM) y el resto del las tarjetas y/o sistemas. Para ofrecer mayor variedad, la compañía mantiene una única MB sobre la cual pueden montarse diversos procesadores. La MB además contiene un conector USB, un host para una tarjeta SD, y un RTC (real time clock) que es un dispositivo con una pila independiente que mantiene la hora con una precisión de 2 minutos por semana.
- **Bus PC/104:** Para poder contar con un satélite funcional se requieren de otros subsistemas (Comunicaciones, Energía y los Payloads), la interfaz elegida es el estándar PC/104, nuevamente emulando lo que sucede en un PC con los buses de comunicación como PCI o AGP. Este estándar industrial es un bus de intercomunicación montable verticalmente, lo que elimina la necesidad de cables para interconectar dos placas (ver figura 2.6), sin embargo, solo provee una interfaz física y no un sistema de arbitraje o de intercambio de información como PCI o AGP.

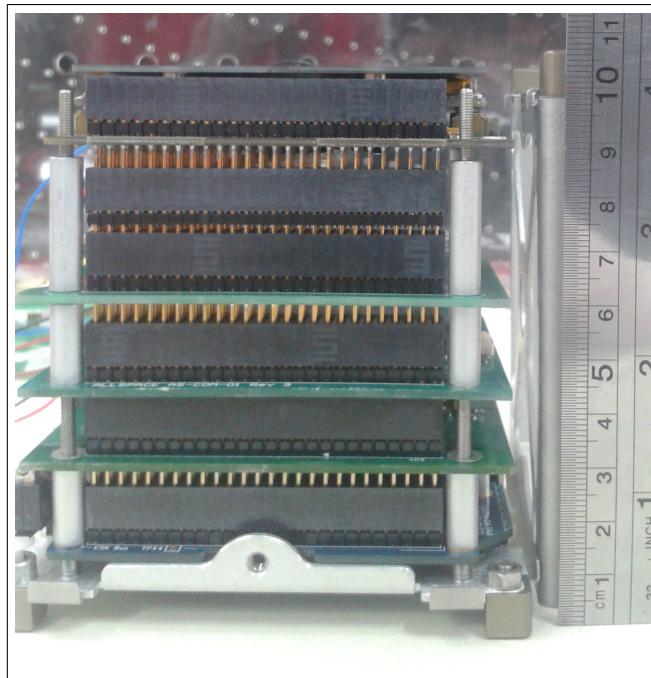


Figura 2.6: Interconexión de placas en un bus PC104

Capítulo 3

Proceso de diseño del SUCHAI

3.1. Antecedentes iniciales

El documento que formalizó y dio inicio al desarrollo del SUCHAI fue ”Satellite of University of Chile for Aerospace Investigation (SUCHAI)” [24]. En él se propone la construcción de un satélite de estándar Cubesat, y se mencionan los objetivos y restricciones estimadas. Sin embargo, estas no se reunieron en un solo enunciado, algunas de las frase que podrían conformar una declaración de intenciones son:

- Este proyecto propone el diseño, construcción, lanzamiento y operación de un pico-satélite en una ventana de tiempo de 2.5 años.
- El costo del proyecto es de alrededor de 170.000 dólares.
- Los principales beneficios esperados dentro del proyecto SUCHAI son:
 1. Entrenar estudiantes en un proyecto multidisciplinario real
 2. Estudiar la factibilidad de lanzar un satélite principalmente diseñado en Chile, el cual permita a la Universidad de Chile tomar el liderazgo dentro del país del área espacial, la cual es ampliamente conocida como una impulsora de tecnologías.
- Uno de los paradigmas actuales en tecnología satelital está apuntando hacia misiones que sigan el principio **Smaller, cheaper, faster, and better** y hacer más por menos. [...] Esto es posible con el rápido avance en la miniaturización de la electrónica [...].
- Los objetivos del proyecto pueden ser clasificados en dos tipos
 1. Educacionales: Seguir desarrollando el know-how existente de los estudiantes, en el trabajo en grandes proyectos multidisciplinarios.
 2. Científico: Explorar la posibilidad de usar Cubesats para investigaciones científicas, con posibles aplicaciones en las necesidades del país.

3.1.1. Misión y Restricciones del proyecto

El llamado **enunciado de la misión** que guió el proyecto SUCHAI fue el siguiente.

- Misión SUCHAI: El proyecto SUCHAI tiene como objetivos, diseñar, construir, integrar, programar, lanzar y operar un nano-satélite de estándar Cubesat por parte de estudiantes.

Además se dieron restricciones en cuanto al costo, el tiempo de duración del proyecto y a su alcance. Las restricciones más importantes a cumplir son.

- Restricción 1: Completar la Misión SUCHAI en una ventana de tiempo de 2.5 años
- Restricción 2: Completar la Misión SUCHAI con un presupuesto del orden de 170.000 dólares
- Restricción 3: Completar la Misión SUCHAI principalmente con estudiantes de la Universidad de Chile

Para este proyecto se decidió, en parte, rehacer el camino que centros extranjeros con experiencia en nano-satélites (como CalPoly) han hecho. Además, ya que ni en la Universidad de Chile, ni en Chile, existía conocimiento en el tema de construcción de satélites, todas las decisiones tomadas fueron siempre siguiendo la lógica de "Mejor esfuerzo", conocida en inglés técnicamente como **Best-Effort** [17], y muchas de ellas fueron tomadas en el camino. El objetivo general en la práctica, equivalía al de **Aprender a diseñar, construir, integrar, programar, lanzar y operar un nano-satélite que funcionara razonablemente, que fuese de estándar Cubesat y hecho por estudiantes**.

Para el caso del SUCHAI, y contrario a un proceso de diseño como el SMAD, los requerimientos no son expresados en términos de la precisión de sus Payloads, de hecho estos no son importantes por lo que hagan, solo son importante en cuanto permitirán medir la capacidad que tenga o no tenga el bus-SUCHAI para hacerlos funcionar y enviar su telemetría. En el caso de este proyecto, el objetivo primordial es la calidad del bus-SUCHAI. Lo anterior es una conclusión directa de la misión del SUCHAI, puesto que si se analiza el enunciado de la misión (mission statement en inglés) este no menciona algún fenómeno a medir, o algún servicio a proveer. No indica si la telemetría descargada será siquiera utilizada por alguien aparte del equipo SUCHAI. Esto se debe, a que el objetivo es el satélite en si mismo.

Subsistema	Descripción	Compañía
Estructura	Cubo y switch de despliegue	kit Pumpkins
C&DH	Microprocesador PIC24F, host para tarjetas SD, reloj RTC, y salidas para comunicación serial	kit Pumpkins
Comunicaciones	Transceiver para el satélite, no incluye la antena, construida por el equipo-SUCHAI	Allspace
Energía	Recolección de energía por paneles solares, almacenamiento en una batería y distribución con reguladores	Clyde Space

Tabla 3.1: Componentes adquiridos para el satélite

3.2. Componentes y su integración

3.2.1. Componentes adquiridos

El marco dentro del cual se encuentra el proyecto es acotado, ese marco es el de **proyectos Cubesat**, esto permitió reducir el número de posibilidades a analizar. Y ya que el estándar Cubesat es uno predominantemente académico (a diferencia de proyectos de nano-satélites como OSCAR de la AMSAT), se podía entonces tener como referencia proyectos desarrollados por otras universidades. Y ya que la mayor parte de estos, y sobre todo los primerizos, usan el kit de Pumpkins [37], se decidió, muy acertadamente hacer lo mismo para el caso del SUCHAI. Luego de tomada esta decisión se fijó otro grado de libertad, pues incluir el kit implica seguir una serie de patrones de diseño y uso, especialmente que el resto de los subsistemas **deben ser PC/104 compatibles**. Así también se redujo el trabajo a realizar a una escala capaz de realizarse por el número limitado de personas del equipo.

La siguiente decisión respecto al diseño del proyecto fue la de adquirir a compañías del rubro Cubesat dos componentes críticos, los subsistemas de Comunicaciones y de Energía. Esta decisión se tomó considerando los objetivos específicos a cumplir, ver Restricción 1, Restricción 2 y Restricción 3. Por otro lado, y al igual que lo analizado al momento de adquirir el kit de Pumpkins, tanto la escala de personas trabajando en el proyecto, los plazos y los recursos económicos no hacían factible otra decisión, teniendo en cuenta el trabajo que más adelante se detallará. Los componentes adquiridos fueron los del cuadro 3.2.

3.2.2. Componentes construidos

Con los equipos adquiridos se contaba con el subsistema de energía, comunicaciones y parcialmente el de estructuras. Sin embargo estos no son suficientes pues quedan aún compo-

Subsistema	Descripción	Placa de Ubicación
Estructura	Antena dipolo y su Balun	Placa Antena
Estructura	Mecanismo de Despliegue	Placa Antena
C&DH	Almacenaje persistente de datos	Placa Payloads
Payload	Experimentos varios	Placa Payloads

Tabla 3.2: Componentes construidos para el satélite

Componente	Subsistema	Diseñar y Construir	Desarrollar Software
Cubo (kit Pumpkins)	Estructuras	No	No
PPM y MB (kit Pumpkins)	C&DH	No	Si
Transceiver (Allspace)	Comunicaciones	No	Si
Placa antena (Diseño propio)	Comunicaciones	Si	Si
EPS (Clyde Space)	Energía	No	Si
Placa payload (Diseño propio)	Payload	Si	Si

Tabla 3.3: Componentes en el satélite

nentes por construir, tal es el caso de la antena y su mecanismo de despliegue, el cual debía ser cuidadosamente controlado por el software de vuelo. Este componente debió ser diseñado, construido y probado por el equipo del SUCHAI. Completando lo anterior se tienen todas las partes indispensables. Adicionalmente, para mejorar las prestaciones del C&DH y su software de vuelo se decidió añadir una memoria permanente para el almacenaje de datos, estos contendrían variables de la condición del aparato. Al ser persistente la memoria los datos no se perderían luego de un reinicio del Computador a Bordo.

3.2.3. Trabajo a realizar

Al equipo del SUCHAI se le planteó la siguiente situación. Se tenía por un lado un conjunto de hardware adquirido que debía ser integrado y por otro lado placas que debían ser diseñadas y construidas (placas de Despliegue de Antenas y Payloads). Para ambos casos, y para la mayoría de los componentes adquiridos, se debió desarrollar software (ver cuadro 3.3 y 3.4), este debía permitir controlarlos de manera autónoma y de manera remota (desde la estación terrena). Finalmente se debía verificar el conjunto completo, para asegurar que se pasarían las pruebas de certificación Cubesat, ver [22].

Componente	Diseñar y Construir	Desarrollar Software
Programa decodificación de Telemetría (Diseño propio)	No	Si
TNC estación terrena (Allspace)	No	No
Transceiver estación terrena (ICOM)	No	No
Antena estación terrena (Yaezu)	No	No

Tabla 3.4: Componentes en la Estación Terrena

Guías de diseño
<p>Se debe primero desarrollar un bus-SUCHAI funcional, que sea lo más modular posible, apuntando a tener la capacidad de cambiar un subsistema (como el TRX de Allspace, o la EPS de ClydeSpace por el de otra compañía, o uno desarrollado por la universidad) y realizar la menor cantidad posible de cambios en el software. Solo luego de ello, montar los Payloads al bus SUCHAI, tales como experimentos en ingrávitez, sensores de temperatura, magnéticos o cualquier otro (dentro de cierto rango de compatibilidad más tarde descrito) de la manera más genérica posible de modo de poder usarlo como bus-SUCHAI y/o ampliarlo para futuras misiones.</p>

Tabla 3.5: Guías de diseño del SUCHAI

3.2.4. Guías de diseño

Una vez aclarado los componentes adquiridos y los construidos, lo que resta es definir qué tan bien se pretende hacer este primer satélite. Pero antes de pasar a una etapa de recolección de requerimientos, se conversaron y fijaron directrices en torno a las cuales el proyecto se desarrollaría, siendo las principales las de **modularidad, escalabilidad, fácil integración y reutilización**, estas se plasmaron en el siguiente enunciado 3.5.

3.3. Requerimientos de la arquitectura

La sección anterior (ver 3.2) es el punto de partida y la base alrededor de la cual el SUCHAI fue diseñado y luego implementado, sin embargo esto no permite ni desarrollar ni realizar las pruebas de manera certera ni objetiva. Existe una forma recomendada para contrastar si el trabajo realizado era suficiente o no para asegurar el buen funcionamiento del SUCHAI y son los llamados **requerimientos**.

Existían eso sí, una serie de funciones conocidas que el satélite debía cumplir y otras que

se quería que cumpliese, estas eran:

1. Enviar y obtener una señal de Beacon por parte del satélite.
2. Enviar y obtener telemetría de estado por parte del satélite
3. Ejecutar payloads obtener telemetría de payloads
4. Controlar la operación del satélite usando telecomandos y recibiendo la telemetría de vuelta cuando esta fuese requerida
5. Poder establecer una conexión cerrada de telecomandos-telemetría
6. Ser modular, genérico, escalable, reusable
7. Permitir integrar Payloads fácilmente

A pesar del listado anterior, cuando en un proceso de diseño se habla de requerimientos se habla de algo definido y concreto. Estos, son el conjunto de enunciados bajos y para los cuales se diseña e implementa una determinada solución a un problema, y más aún, es en base a la verificación de estos que se mide el grado de éxito de la solución desarrollada [16]. Por eso solo luego de establecer que subsistemas se adquirirían y cuales no, de determinar que trabajo era necesario realizar tanto en hardware como software, y en especial solo después de tener claras las funcionalidades esperadas se formalizaron estos.

3.3.1. Mecanismo de generación/recolección de requerimientos

Existen criterios para aceptar formalmente un requerimiento como tal, y estos son **atómicos**, **no suntuariedad** y **verificabilidad**. **atómicos** en el sentido de que cada requerimiento cubre solo un aspecto, pues es contra ellos que se realizará la verificación, y estos deben ser siempre lo más específicos posibles para ser concluyentes. **No suntuarios** pues detrás de cada uno debe haber una razón y necesidad real para justificarlo, de lo contrario el producto se complejiza sin necesidad. **Verificables** pues como se dijo, cada uno es puesto a prueba en el proceso de verificación, y un requerimiento no verificable no puede ser declarado como satisfecho o insatisfecho.

Para generar correctamente los requerimientos se siguió un proceso en que se establecía lo que se esperaba del satélite y a partir de esto extraerlos iterativamente. Este método de trabajo consistió en el **ejercicio teórico de simular la vida del satélite** y en vista de las acciones/reacciones necesarias del SUCHAI extraer los requerimientos. Esto, desde su certificación para ser lanzado hasta su operación nominal, considerando diversos **escenarios**, como aquellos de batería baja o degradación de la misma, períodos de incomunicación con la tierra, temperatura baja o alta, etcetera. Un extracto de este proceso se presenta en la figura 3.1.

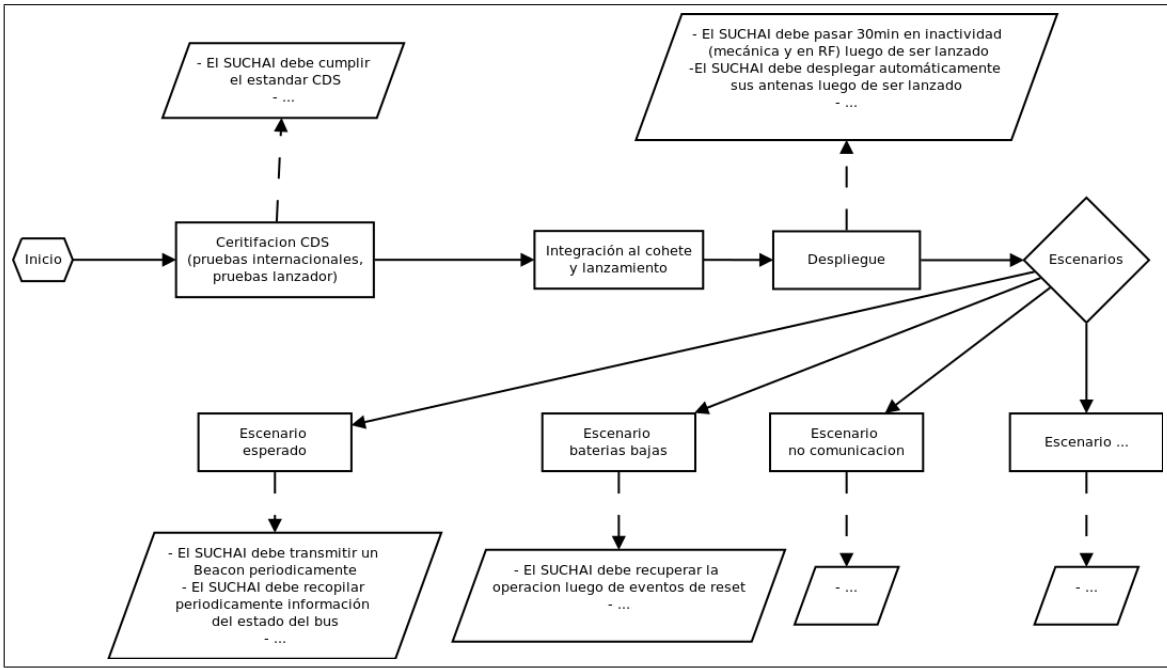


Figura 3.1: Organigrama de la simulación de la vida SUCHAI

3.3.2. Registro de resultados en la Matriz de Trazabilidad de Requerimientos

El resultado del proceso antes descrito fue recogido en la Matriz de Trazabilidad de Requerimientos del SUCHAI, también llamada RTVM del SUCHAI. Este documento cumple más de un propósito a la vez. Por un lado recoge los requerimientos y permite clasificarlos en áreas (subsistema de Comunicaciones, tarea de Despliegue, etc) e importancia (imprescindible, no imprescindible). Por otro lado interconecta cada requerimiento con su respectiva implementación en la Arquitectura del bus SUCHAI. Finalmente lo interconecta con el test que lo verificará. La RTVM del SUCHAI es de suma utilidad, pues permite entender cómo los requerimientos (moldeados a su vez por las restricciones y la misión) se transforman en hardware o software en la implementación. Y de manera inversa, ver qué modulo de la implementación (tanto hardware como software) es responsable de cumplir qué requerimiento(s).

3.4. Discusión de la Etapa de Diseño del SUCHAI

Una de las mayores recomendaciones a sugerir si se pretende realizar un nuevo satélite, es la realización de un proceso formal y ojala guiado por algun procesos de diseño de alguna agencia espacial, eso si salvaguardando las evidentes diferencias en escala que harán absolutamente prescindibles a algunos pasos y más breves a otros. Otra alternativa es seguir el propuesto al final de la presente memoria (ver figura 6.1), que se cree es más adecuado a la escala de un Cubesat.

Implementación		Testeo y Verificación			Documentos adicionales	Comentarios
Componentes implicados	Modulos de Software implicados	Link del documento de la prueba	Nombre de la prueba	Resultado		
Todos (debido al outgassing)	Ninguno					
Todos, pero solo mecanicamente	Ninguno					
Ninguno	Listener (taskDeployment)	https://docs.google.com/document/d/1ySPzsharing	Prueba de Despliegue Antenas			
Placa antenas	Listener (taskDeployment)	https://docs.google.com/document/d/1ySPzsharing	Prueba de Despliegue Antenas	Satisfactorio		
Placa antenas	Listener (taskDeployment)	https://docs.google.com/document/d/1ySPzsharing	Prueba de Despliegue Antenas			
Placa EPS, Paneles solares	Ninguno					
Placa TRX	Listener (taskCommunication)	https://docs.google.com/document/d/1QfmCPtUmiQ-J_YcEH1VY4i30ZmhHPJA/edit?usp=sharing	Prueba Beacon			
OBC, memEE PROM, RTC, memSD, Placa TRX, Placa EPS, Payloads	Listener (taskCommunication) (taskHousekeeping)	https://docs.google.com/document/d/1QfmCPtUmiQ-J_YcEH1VY4i30ZmhHPJA/edit?usp=sharing	Prueba Beacon			
Placa TRX	Listener (taskDeployment)		Link Budget			
Placa TRX	Listener (taskDeployment)		Link Budget			
ICOM	Ninguno		Link Budget			

Figura 3.2: Extracto del documento RTVM del SUCHAI (mitad derecha)

En resumen y respecto a lo realizado finalmente, el proceso de diseño del SUCHAI paso desde el **enunciado de la misión** y la especificación de sus **restricciones** hacia el añadido de **guías de diseño**, luego de lo cual se identificaron las funcionalidades a cumplir y se extrajeron finalmente los **requerimientos** que el satélite debe cumplir, ver figura 3.4.

RTVM del SUCHAI		Diseño		Imp	
ID del Requerimiento	Tags del Requerimiento	Requerimientos	Imprescindible	Requisito	Subsistemas implicados
Diseñar, construir, integrar, programar, lanzar y operar un nano-satélite de estándar Cubesat por parte de estudiantes					HW: Bus SUCHAI=C&DH(O Switch Despliegue, Placa Solares)+Com(TRX, calibre Command Path)
1	Estandares	El SUCHAI debe cumplir el estándar CDS	Imprescindible	Activo	Todos (Mec, C&DH, Com, Energia, Payloads)
2	Estandares	El SUCHAI debe cumplir las pruebas del lanzador	Imprescindible	Activo	Todos (Mec, C&DH, Com, Energia, Payloads)
3	Deployment	El SUCHAI debe pasar 30min en inactividad (mecánica y en RF) luego de ser lanzado	Imprescindible	Activo	C&DH
4	Deployment	El SUCHAI debe desplegar automáticamente sus antenas luego de ser lanzado	Imprescindible	Activo	Mec, C&DH
5	Deployment	El SUCHAI debe detectar el despliegue o no de las antenas y reiniciar si es necesario		Activo	Mec, C&DH
6	Power Budget	El SUCHAI debe tener energía para funcionar por al menos 6 meses		Activo	Energia
7	Beacon	El SUCHAI debe transmitir un Beacon periódicamente	Imprescindible	Activo	Com
8	Beacon	El Beacon del SUCHAI debe contener información relevante del bus		Activo	Todos (Mec, C&DH, Com, Energia, Payloads)
9	Link Budget	El Beacon debe ser escuchado en la Est Terrena	Imprescindible	Activo	Com
10	Link Budget	La TM debe ser escuchada en la Est Terrena	Imprescindible	Activo	Com
11	Link Budget	Los TC de la Est Terrena deben ser escuchados por el SUCHAI	Imprescindible	Activo	Est Terrena

Figura 3.3: Extracto del documento RTVM del SUCHAI (mitad izquierda)

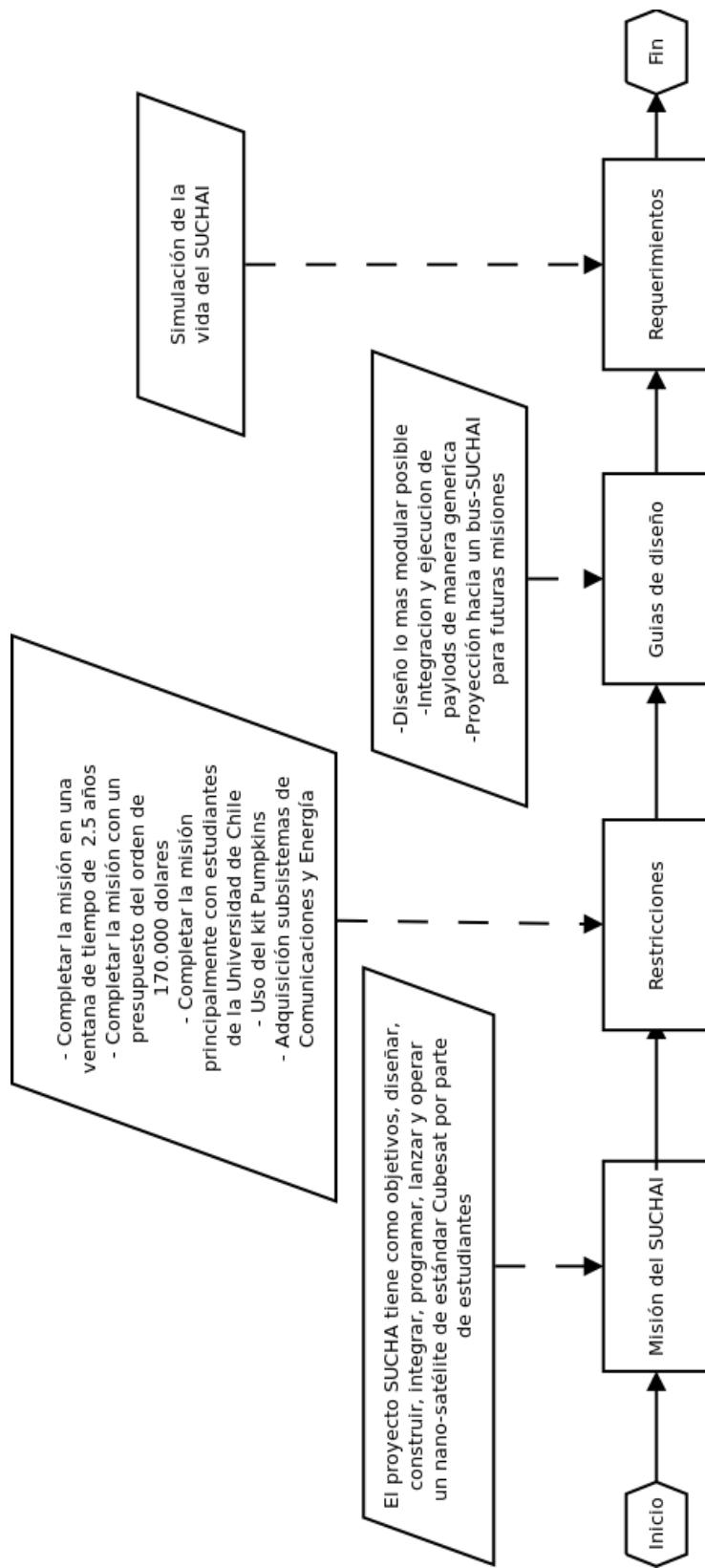


Figura 3.4: Resumen del proceso de diseño del SUCHAI

Capítulo 4

Implementación del bus SUCHAI

4.1. Arquitectura del bus SUCHAI

El Cubesat SUCHAI es un proyecto complejo, fusiona elementos de sistemas embebidos y software, con elementos de comunicación por radiofrecuencia, ingeniería mecánica e ingeniería eléctrica. Para describir de mejor manera lo desarrollado, se hará una analogía con el modelo OSI. La analogía es clara en capas inferiores, pero no en las altas. Siendo generosos se puede ver un parecido entre una comunicación host to host del modelo OSI, con el caso de comunicación entre subsistemas del satélite. Sin embargo la comparación no es fiel en todo sentido y no pretende serlo, pero se espera permita explicar de mejor manera los múltiples aspectos del bus SUCHAI.

Bus SUCHAI y Bus PC/104
Vale la pena antes que todo, aclarar una fuente de confusión respecto a la palabra bus, pues tiene distintos significado según el área.
El primero es en el ámbito satelital, cuando se habla del bus SUCHAI en el caso de esta memoria, se pretende hacer referencia al sistema completo (todos los subsistemas, las frecuencias de transmisión y su codificación, las antenas, paneles solares, baterías, tarjetas de memorias, comandos internos, protocolos de telemetría y telecomandos, etc), pero exceptuando los Payloads. Se pretende que el bus sea un proveedor de servicios, y que los payloads sean los usuarios de ellos, de esta forma se crea un ente servidor y uno o mas entes cliente . El objetivo es reutilizar el servidor para múltiples misiones, de modo que pueda ser estandarizado y sean los clientes o payloads los que varían y son diseñados a medida para cada misión.
El segundo significado de la palabra bus es uno mucho más técnico y preciso, usado en el área de electrónica, un bus es un conjunto de vías de comunicación , que permiten a múltiples dispositivos intercambiar información masiva y concurrentemente . Ejemplos de estos buses son comunes en PCI o AGP en el caso de computadores de escritorio, los buses MIL-STD-1553 y OBDH/RS485 en el caso de satélite comerciales y en el caso del SUCHAI el bus PC/104. Ya que la base de intercambio de información entre los subsistemas de un satélite es siempre a través de un bus, con el tiempo se le asigno ese nombre al conjunto completo y de ahí nace la confusión.

Tabla 4.1: Aclaración del doble uso de la palabra bus

4.1.1. Capa física

Modelo OSI: La capa física define las especificaciones físicas y eléctricas para los dispositivos. En particular, define la relación entre el dispositivo y el medio físico de transmisión, como cables de cobre o fibra óptica. Esto incluye el layout de pines, voltajes, impedancia de las líneas, especificaciones de los cables, especificaciones de tiempo de señales, repetidores, adaptadores de red, etc [19].

SUCHAI: El SUCHAI, con 1.33 kg de peso máximo que podrá alcanzar es clasificado como un nano-satélite, específicamente uno de tipo Cubesat, tiene un diseño modular con cuatro subsistemas desarrollado usando el kit Pumpkins como base, esto es, entorno al bus PC/104. Se escogió el Kit de Pumpkins como subsistema mecánico y de C&DH. La placa llamada EPS de ISIS conforma el subsistema de Energía (batería y reguladores), el que incluye paneles solares. Y el transmisor o TRX de Allspace en comunicaciones, que no incluye la antena. No se cuenta en el SUCHAI con un subsistema Termal, ni de Propulsión ni de ACDS.

Por parte del equipo, se desarrollaron dos placas. Una es llamada **Placa de Antenas** que contiene a los dos dipolos que la conforman (la antena), y al mecanismo para su despliegue y comprobación. La otra placa es la llamada **Placa de Payload** que contiene los componentes de control del despliegue de las antena, memorias EEPROM y todo el hardware de los diferentes experimentos (Payloads).

Conceptualmente la memoria EEPROM destinada a servir al C&DH, no debería estar en la Placa de Payloads pues **para hacer más modular al satélite cada placa debiese ser solo un único subsistema**. Sin embargo debió ser reubicada allí debido a que la placa de C&DH es parte del Kit Pumpkins y esta se recibió obviamente ya terminada y construida.

Placas y su interconexión

Desde la base del Cubesat, la primera placa es parte del kit y es la llamada MB, sobre ella se encuentra el PPM en el cual va el OBC. Desde la MB nace el bus PC/104, este bus permite interconectarlos a todos y entre todos sin necesidad de cables, lo hace por medio de un sistema hembra-macho de pines. Esto le da cierta rigidez estructural al conjunto, pero para evitar desconexiones por la vibración del cohete en el lanzamiento, se añaden tornillos que a la vez permiten fijar la altura de separación entre placas, ver figura 2.6. Cada placa mide aproximadamente 9 cm x 9 cm de área.

Bus PC/104

En el caso del SUCHAI el OBC elegido es el microprocesador PIC24FJ256GA110 de Microchip, por lo que todos los voltajes de sus pines serán con niveles entre 0 y 3.3V. Por diseño del Kit los pines del OBC están conectados hacia componentes del PPM, la MB y el bus PC/104. De los 100 pines con que cuenta el PIC24, 19 de ellos son **pines de sistema** es decir pines para el funcionamiento del PIC24 mismo, y no para su operación. En la placa PPM, 10 pines no están conectados y por tanto no pueden ser aprovechados, 4 corresponden al protocolo de depuración (debugging) JTAG y tampoco son aprovechados, de los restantes, 5 están destinados a la memoria Flash de la placa la que no fue usada para el SUCHAI.

Por lo tanto solo 62 pines pasan a la MB, y de estos, 4 son destinados a control de hardware, específicamente: para el encendido/apagado del USB, para la alimentación o no del Transceiver de Pumpkins (no utilizado en el SUCHAI), para el paso o no de información hacia ese hipotético Transceiver y por último para el encendido/apagado de la tarjeta SD. Un quinto pin queda en el MB por razones no entendidas aún, es el llamado DCD_MHX que es parte de control de flujo del protocolo RS-232.

Finalmente solo 57 pines del OBC llegan al bus. Por lo tanto solo 57 de los 104 pines del bus son controlados por el OBC. El resto de los $104 - 57 = 47$ pines del bus que no son manejados por el OBC son de variada índole y pueden subdividirse en 3 subgrupos, aquellos de **alimentación**, los de **uso reservado** y los **libres**. El detalle de los pines del bus PC/104 se puede apreciar en la figura 4.1.

En el subgrupo de alimentación los pines más importantes son: DGND que es la tierra del bus y por lo tanto de todo el satélite. VBAT, que está pensado para estar directamente conectado a la batería del subsistema de Energía. +5V_SYS linea de 5V proveniente del subsistema de Energía. VCC_SYS linea de 3.3V proveniente del subsistema de Energía. Y

H1 (IZQUIERDA)		H2 (DERECHA)		
Nº	Nº	Nº	Nº	
1	SDA3/CN05/PMD7/REF7 (Pin 3)	SCL1/CN84/PMD6/REF6 (Pin 4)	2	1
3	SDA2/CN36/RAS3 (Pin 53)	SCL2/CN35/RA2 (Pin 50)	4	3
5	RPT2/CN11/PMW/RD4 (Pin 51)	RPT4/2/CN57/RD1/2 (Pin 79)	6	5
7	RPT22/CN52/PMBE/RD3 (Pin 78)	RTCC/RP2/CN53/RD6 (Pin 65)	8	7
9	RPT12/CN75/RF12 (Pin 40)	RPT31/CN70/RF13 (Pin 39)	10	9
11	RPT43/CN20/RD14 (Pin 47)	RPT5/CN21/RD15 (Pin 46)	12	11
13	C1IND/RP21/CN80/PMA3/RG6 (Pin 10)	C1INC/RP26/CN81/PMA4/RG7 (Pin 11)	14	13
15	C2IND/RP19/CN10/PMA3/RG8 (Pin 12)	C2INC/RP27/CN11/PMA2/RG9 (Pin 14)	16	15
17	RPT10/CN17/PMA9/RF4 (Pin 49)	RPT17/CN18/PMA0/RF5 (Pin 50)	18	17
19	RPT30/CN70/RF2 (Pin 52)	RPT10/CN71/RF3 (Pin 51)	20	19
21	RPT4/CN54/RD9 (Pin 69)	RPT44/CN75/RF7 (pin 54)	22	21
23	RPT15/CN74/RF0 (Pin 53)	CN6/PMD5/RE5 (Pin 3)	24	23
25	Fault [Active Low]	PMA0/VREF+C/N42/RA10 (Pin 29)	26	25
27	Sense	VREFI	28	27
29	Reset [Active Low]	VREF+C/N41/PMA7/RA9 (Pin 26)	30	29
31	OFF_VCC	5V_USB	32	31
33	FWR_MHX	RST_MHX [Active Low]	34	33
35	CTS_MHX [Active Low]	RTS_MHX [Active Low]	36	35
37	DSR_MHX [Active Low]	DTR_MHX [Active Low]	38	37
39	TxD_MHX	RXD_MHX	40	39
41	SDA1/CN84/RG3 (Pin 56)	VBACKUP	42	41
43	SCL1/CN83/RG2 (Pin 57)	RSVDO	44	43
45	RSVD1	RSVDO2	46	45
47	USER0	USER1	48	47
49	USER2	USER3	50	49
51	USER4	USER5	52	51

An15/REF0/RP29/CN12/PMA0/RB15 (Pin 44) An14/CTPLS/RP14/CN32/PMA1/RB14 (Pin 43)
 An12/CTED2/CN30/PMA1/RB12 (Pin 41) An10/CVERFC/CN20/PMA3/RB10 (Pin 34)
 AN5/RP6/CN26/RB8 (Pin 32) PGEC2/AN0/RP0/CN24/RB6 (Pin 26)
 PGEC1/AN1/RP1/CN3/RB1 (Pin 24) AN5/CINAC/CN9/RB3 (Pin 22)
 PGEC3/AN2/CN8/RB4 (Pin 21) PGEC3/AN3/C1NEURP25/CN6/RB4 (Pin 20)
 RP13.5/CN86/REG (Pin 16) ASC12/RP13.5/CN41/RA14 (Pin 66)
 CN70/RG1 (Pin 89) CN65/RP1 (Pin 66)
 5V_SYS VCC_SYS
 DGND DGND
 AGND AGND
 S0 S1
 S2 S3
 S4 S5
 VBATT USER6
 USER7 USER9
 USER10 USER11

Figura 4.1: Bus PC104 y pines controlados por el OBC (en gris)

finalmente VBACKUP linea de 3V proveniente de una pila ubicada en la MB, está conectada al RTC o reloj de tiempo real de la misma placa, se emplea para mantener una referencia temporal absoluta.

En el subgrupo de uso reservado los pines más importantes son. Pines de Kill-Switch, destinados a cerrar/abrir un circuito para alimentar o no al satélite entero. Estos pines son 6 y se llaman S0, S1 .. S5 y la idea es que sean empleados por la placa del subsistema de energía. Ese es exactamente lo que sucede en el caso de la placa EPS de ISIS. Otro pin importante es OFF_VCC destinado a apagar desde otra placa, la energía del PPM, que a su vez la provee a la MB. Esto podría ser deseable para que otra placa tomase control del satélite, sin embargo no fue usado.

En el último subgrupo, todos tienen en común no estar conectados ni al OBC, ni al PPM ni al MB. Pueden ser empleados para comunicación entre placas, sin embargo esto no es deseable, pues rompe la noción de bus, al no poder **todos los componentes** recibir **todas las señales** que pasen por algún punto del bus. Ninguno fue utilizado

Conexiones fuera del PC/104

Idealmente no deberían haber conexiones fuera del bus, lamentablemente, es inevitable que hayan excepciones, estas son las siguientes.

La EPS adquirida incluía seis paneles solares, uno por cada cara. Estos son montados a las caras exteriores del cubo y son fijados por medio de ganchos o trabas. Desde cada panel solar nace un cable que va a la placa de la EPS, placa que contiene a la batería y al sistema de regulación.

El TRX además de la placa que contiene al sistema de modulación/demodulación y codificación/decodificación, incluye un conector coaxial, desde el que nace un cable que llega hasta el conector de la placa de antenas.

El sistema de despliegue de antenas diseñado para el SUCHAI precisa de cables que vayan desde la placa de payload hasta la placa de antenas. Esta última está ubicada en una cara exterior del cubo, uno de los 6 paneles solares es reemplazado por esta placa, ver figura 4.2.

Componentes principales

La lista de componentes más importantes, su función y la placa y/o subsistema al que corresponden se presenta en el cuadro 4.2.

Subsistema	Placa	Subgrupo	Componente	Función
C&DH	MB	PPM	OBC	Contener y ejecutar el Software de vuelo del SUCHAI
C&DH	MB	Hw MB	Mem SD	Proveer de almacenamiento seguro y masivo al software de vuelo SUCHAI
C&DH	MB	Hw MB	MAX232 y FTDI	Conversores UART a RS232 y USB respectivamente, usados para Consola del software SUCHAI
C&DH	Placa Payload	Hw OBC	Mem EEPROM	Proveer de almacenamiento seguro y rápido para las Variables de Estado del software SUCHAI
Com	Placa Transceiver		TRX	Encargado de codificar/decodificar, modular/demodular telecomandos/telemetría desde y hacia tierra, además de transmitir Beacon, todo bajo demanda del software SUCHAI.
Com	Placa Antenas	Hw Com	Balun	Provoca un desfase entre la señal que llega a una y otra antena, de manera que se comporte como un dipolo.
Com	Placa Antenas		Dipolos	Laminas de cobre-berilio ajustadas a 437 MHz.
Estruct y Mec	Placa Payload	Hw Antena	Switch de corriente	Switches que permiten/cierran el paso de corriente para el despliegue de antenas.
Estruct y Mec	Placa Antenas		Switch genérico	Comprobación del despliegue de antenas.
Estruct y Mec	Placa Antenas		Cuñas plásticas	Piezas plásticas pasivas que fijan la antena en su posición de plegado y que permiten su despliegue una vez dada la orden desde el software SUCHAI.
Energía	Placa EPS		Batería	Almacena energía desde los paneles solares, provee energía hacia el resto del satélite.
Energía	Placa EPS		Controlador	Microprocesador que controla la carga/descarga de la batería, paneles solares y la regulación de las líneas de 3V y 5V.
Energía	Paneles Solares		Celdas Solares	Se usarán 5 celdas para captar la energía del sol, convertirla en eléctrica y llevarla a la placa EPS.

Tabla 4.2: Lista de componentes más importantes, función y pertenencia jerárquica

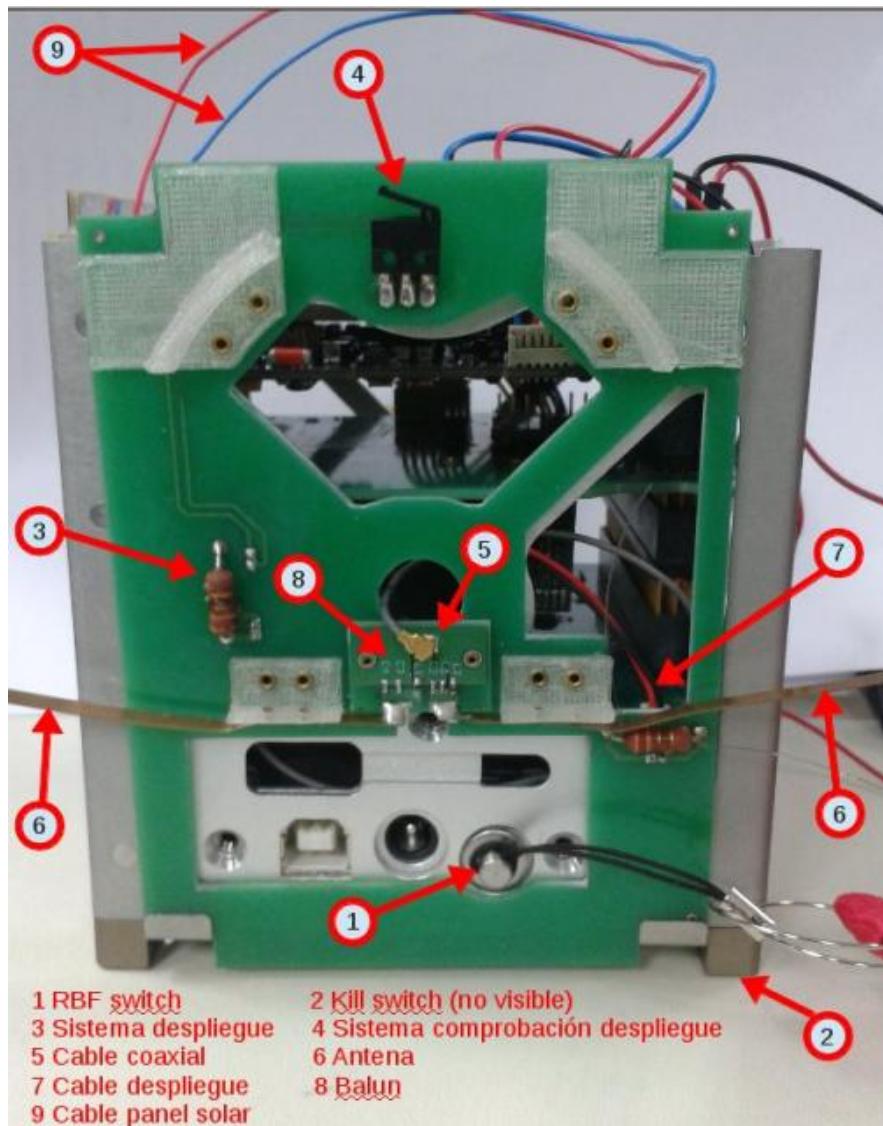


Figura 4.2: Placa de Antenas

4.1.2. Capa de enlace

Modelo OSI: La capa de enlace brinda las características funcionales y de procedimientos para la transferencia de datos entre entidades de la red, y detecta y posiblemente corrige errores que puedan ocurrir en la capa física [19].

SUCHAI: Se explicó que la comunicación entre placas, y por tanto entre subsistemas, se da a través del bus PC/104. También se explicó que la conexión entre el C&DH y el bus PC/104 se da a través de los pines del OBC. Si se analizan la arquitectura interna de este, se verá que además de la CPU cuenta con módulos llamados **periféricos** con los que puede intercambiar información con el exterior y pines de propósito general (de entrada o salida binaria 1-lógico=3.3V y 0-lógico=0V). Existen tres tipos de periféricos, **I2C**, **SPI** y **UART**, dedicados a protocolos de comunicación estándar. Esto fuerza a que todos los demás protoco-

los de comunicación de los demás subsistemas sean compatibles con los del OBC, algo lógico de todos modos.

Diseño del bus PC/104 para el SUCHAI

Desde el punto de vista de los subsistemas, las placas de Energía y Comunicaciones adquiridas tienen destinados para su uso una serie de pines, y vienen diseñados por hardware de esa manera. Por tanto al asignar los pines del bus, se debía considerar la **compatibilidad** entre ellos, para así **evitar colisiones**. Posteriormente, todo subsistema y/o Payload que se agregase, debía ser sobre aquellos pines libres.

Para coordinar tal proceso se empleó un documento, en el cual se hizo una biyección entre cada pin del OBC, cada pin del bus y la función a cumplir. Así por ejemplo, para el manejo de experimentos que irán en la **Placa de Payload**, se decidió usar el pin H1.1 y H1.2 del bus correspondiente a los pines 3 y 4 del OBC como el protocolo de comunicación I2C, ver figura 4.3.

Protocolo I2C, caso especial

La coordinación evita usar pines incompatibles con el fin deseado (por ejemplo usar un pin de SPI como UART), y a la vez destina cada pin a un componente y una función. Esto es imprescindible para los protocolos de comunicación SPI y UART, pues ambos son unívocos, es decir, solo sirven para comunicar dos dispositivos y no más. Por eso un caso especial es el del protocolo I2C pues este es en estricto rigor un tipo de bus. En el se pueden **multiplexar dispositivos** que se comuniquen usando los mismo pines, para tal efecto existen las llamadas **direcciones**, que son números de 7 bit que identifican el chip físico y existe un **maestro**, que es el dispositivo que controla en cierto instante el bus I2C y muchos **esclavos** que son todos los demás puntos conectados. El maestro inicia las transacciones con los esclavos, de los cuales solo responde el aludido (según la dirección única de 7 bits).

Drivers de protocolos

Desde el punto de vista de programación del OBC, se implementaron los llamados **drivers** que son funciones genéricas que permiten la transmisión de datos entre dispositivos usando algún protocolo. La parte de más bajo nivel del software de estos drivers **siempre** es específico del OBC elegido, pues los periféricos tienen registros de control y buffers de escritura lectura con direcciones de memoria dependientes del modelo exacto de microprocesador. De todos modos, y para facilitar la eventual migración del bus SUCHAI hacia un procesador distinto, el objetivo es que la firma se mantenga (el nombre, valor de retorno y argumentos de la función) y la forma de utilizarla sea independiente de dispositivo al que le transmite o desde el

Nº	I/O	Estado	Nº
1	I/O	IO.23 (SDA2)	IO.22 (SCL2)
		Datos - Segundo bus I2C (Payloads - temp, gyro) SDA3/CN65/PMD//RE7 (Pin 3)	Reloj - Segundo bus I2C (Payloads - temp, gyro) SCL3/CN64/PMD6/RE6 (Pin 4)
		IO.21 (SDA1)	IO.20 (SCL1)
3	I/O	MAGN_SW (Magnetotorquer) SDA2/CN36/RA3 (Pin 59)	CAM_SW (Camara - Payload) SCL2/CN35/RA2 (Pin 58)
		IO.19 (GPIO)	IO.18 (GPIO)
5	I/O	Active Low reset input (GPS - Payload) RP25/CN13/PMW/RD4 (Pin 81)	Hold (Camara - Payload) RP42/CN57/RD12 (Pin 79)
		IO.17 (GPIO)	IO.16 (GPIO)
7	I/O	INT1 (Gyro 3 axis - Payload) RP22/CN52/PMBE/RD3 (Pin 78)	RTCC/RP2/CN53/RD8 (Pin 68)
		IO.15 (-UCTS1)	IO.14 (-URTS1)
9	-		SW_LP (Despliegue Langmuir Probe) RP31/CN76/RF13 (Pin 39)
			IO.12 (-URTS0)
			IO.13 (-UCTS0)

Figura 4.3: Extracto del documento para coordinación del bus PC104 del SUCHAI

Componente	Protocolo de comunicación	Periférico del OBC involucrado
Consola Debug	UART	UART1
Mem SD	SPI	SPI1
Mem EEPROM	I2C	I2C1 (bus del sistema)
EPS	I2C	I2C1 (bus del sistema)
TRX	I2C	I2C1 (bus del sistema)
TRX	UART	UART2
Payloads	I2C	I2C3 (bus de payloads)

Tabla 4.3: Componentes del SUCHAI y sus protocolos de comunicación

que reciben. El detalles de que componentes usaban que protocolos se presenta en la tabla 4.3

Drivers de componentes

El caso del TRX es un excelente ejemplo para explicar la necesidad de un driver de segundo nivel. En el caso del TRX hay dos protocolos en uso, uno es I2C y el otro UART, el principio de funcionamiento es que los datos de telecomandos y telemetría sean transmitidos por UART, y las ordenes de configuración (frecuencia de uso, tasa de transmisión, mensaje en el Beacon, etc) para el TRX sean enviados por I2C. Esto revela la necesidad de crear **drivers de componentes** que manejen el comportamiento de estos de manera comprensiva. La manera de construirlos es usando los drivers de protocolos. Cada subsistema tiene un driver único de cómo se maneja, pero dos drivers de componentes pueden usar el mismo driver de protocolo.

Dentro del software SUCHAI, cada vez que es necesaria la comunicación con otro componente del satélite se hace uso de estos drivers y no de los de protocolo.

4.1.3. Capa de red

Modelo OSI: La capa de red brinda las características funcionales y de procedimientos para la transferencia de datos de largo variable desde un host de origen en una red, hasta el host de destino en otra red (en contraste con la capa de enlace que conecta hosts dentro de la misma red). La capa de red realiza funciones de ruteo, y puede también realizar fragmentación y desfragmentación, y reportar errores de envío [19].

SUCHAI: Si seguimos la analogía en que una conexión OBC-componenteY es una red (implementada con I2C por ejemplo), y que la conexión OBC-ComponenteY es otra (con protocolo UART). Entonces la entidad que permite interconectarlas es una que reside en la capa de red. En el caso del SUCHAI tal entidad es un **comando**. Los comandos son las unidades fundamentales que determinan el comportamiento del satélite, esto significa que son la resolución mínima con la que se ejercen acciones en este. Los comandos realizan operaciones complejas o sencillas, pero siempre acotadas. **Existe un comando para cada acción, y un comando solo hace una determinada acción.** El que funciones en el driver de algún componente efectivamente integren dos o más protocolos de comunicación no las hace comandos, pues son genéricos y no realizan una tarea específica, que es lo que se espera de un comando. Además, su objetivo solo concierne a un componente o subsistema.

Si se respeta esta arquitectura, cosa que ocurre en el software SUCHAI, solo a través de un comando se afecta el comportamiento de uno o más componentes y/o subsistemas. Finalmente, de todas las funciones potencialmente ejecutables (empleables) en el software SUCHAI, **solo un comando puede ser llamado desde capas superiores**. Es decir la siguiente capa solo puede ejecutar comandos. Coincidentemente esto también es parte de lo que el modelo OSI exige.

Funcionamiento basados en Comandos

La mayoría de los satélites tienen un funcionamiento basado en comandos [33] y **la pieza central en esta arquitectura es el C&DH**. El subsistema de C&DH es el encargado de recibirlos (decodificados por el subsistema de Comunicación) y repartirlos a otros subsistemas, o ejecutarlos directamente. Recolecta los datos de Payloads y de salud de satélite, los procesa y almacena para que puedan ser posteriormente transmitidos a tierra. Y como en la mayoría de los satélites, incluyendo los comerciales, todo lo anterior es hecho a través de una serie coherente de comandos, los cuales pueden ser enviados desde tierra y ejecutados según un itinerario o generados por algún sistema inteligente en el satélite.

Este subsistema **es el único elemento activo** pues los demás, aunque posean procesadores y software propio (caso de la Energía y Comunicaciones en el SUCHAI), no actúan sobre otros subsistemas, muchas veces ni sobre ellos mismos. Por ejemplo, comúnmente sucede, que si la batería está en niveles críticos de carga el subsistema de Energía no envía ordenes a otros subsistemas para que entren en modo de ahorro. Esta situación **debe ser**

detectada por el C&DH quién **debe reaccionar** con los comandos apropiados, o por el control de tierra.

Arquitectonicamente hablando, y aunque con variaciones según la implementación final, los componentes del C&DH son los siguientes:

- Itinerario: También llamado Flight Plan, almacena una lista de comandos a ser ejecutados en cierto momento/lugar determinado.
- Command handler: Encargado de recibir los comandos, ya sea desde el subsistema de Comunicaciones, desde el Supervisor o desde el Itinerario. Está a su vez compuesto por:
 - Dispatcher: Envía los comandos a los respectivos subsistemas para que estos los ejecuten.
 - Executer: Ejecuta directamente los comandos (precisa de poder de computo y comunicación con otros subsistemas para influir sobre ellos)
- Data handler: Encargado de almacenar datos de salud de satélite (estado de todos los subsistemas) y de payloads.
- Formatter: Encargado de dar formato a la información desde el Data handler y enviarla de cierta manera preestablecida hacia el subsistema de comunicaciones al momento de ser enviados a tierra.
- Supervisor: Encargado de generar comandos para monitorear la salud del satélite, y si así está previsto, generar comandos adecuados ante las situaciones detectadas.

Implementación en el SUCHAI

El Hardware empleado por el C&DH y sus características principales son:

- **OBC:** Encargado de ejecutar el software de vuelo SUCHAI y de comunicarse con el resto de los subsistemas por medio del bus PC/104. Posee 256 KB de memoria programable, y el tamaño del software de vuelo SUCHAI no puede ser mayor que este valor. Tiene una RAM de 16 KB, lo que también limita el poder de procesamiento. La velocidad de operaciones es de 16 MHz.
- **memoria EEPROM:** Memoria permanente, brinda funciones de almacenamiento para un máximo de 127 variables de estado del satélite. Estas variables resumen y controlan los aspectos críticos de operación y salud del SUCHAI
- **memoria SD** Memoria permanente de 2 GB, brinda funciones de almacenamiento para los payloads. Es empleada para almacenamiento masivo de datos, como experimentos, telecomandos recibidos y plan de vuelo (itinerario con acciones a ejecutar).

Área	Código	Nombre	Descripción	Argumento	R/W/N
DRP	0x5024	drp dep write test	Prueba para escribir en la memoria flash	N	N
DRP	0x5025	drp dep update	Lee los datos de memoria externa y actualiza el repositorio	1=Verboso, 0=Austero	W
DRP	0x5026	drp dep write deployed	Actualiza el valor del flag DEPLOYED en mem externa	N	N
DRP	0x5027	drp fpl set index	Actualiza el valor del índice de modificación del f. plan	Índice	W

Tabla 4.4: Comandos del SUCHAI y su descripción (R/W/N=Lectura/Escritura/Ninguna sobre alguna Variable de Estado)

- **Reloj RTC:** Reloj de tiempo real, con una pila de respaldo independiente. Permite mantener una noción de tiempo absoluto, y no una relativa al momento en que se resetea el OBC.

El realizar y mantener una lista de comandos y sus efectos es una práctica indispensable para asegurar que cada aspecto controlable e importante del satélite esté cubierto por un comando, es además recomendada como un paso del proceso de diseño [33]. En el caso del SUCHAI, aquel detalle se encuentra en la documentación del código fuente (ver apéndice B), un extracto de ella se aprecia en la tabla 4.4.

4.1.4. Capa de transporte

Modelo OSI: La capa de transporte permite la transferencia transparente de datos entre usuarios finales, brindando un servicio de transferencia de datos confiable hacia la capa superior. La capa de transporte maneja la confiabilidad de un determinado enlace, a través del control de flujo, fragmentación/desfragmentación y control de errores del mismo [19].

SUCHAI: Ya que los comandos pertenecen por analogía a la capa de red, entonces la o las unidades que envían grupos de comandos son las de la capa de transporte. El paralelismo, aunque ya no tan claro como en capas inferiores, está en que acá se implementan los encargados del manejo de tareas críticas en el satélite, usando comandos como sus unidades mínimas de acción. Esta capa provee un servicio **transparente** a la capa superior, como por ejemplo, Itinerario, salud del satélite, envío de telemetría, etc.

En el caso del SUCHAI, sobre la capa de comandos, se implementó una arquitectura de software basada en una de tipo **Command Pattern**, cuya implementación es ampliamente detallada en [21]. En esta arquitectura existen seis entes principales, uno o más **Listeners**, un **Dispatcher**, un **Executer**, un **Command Repository**, un **Data Repository** y un **Status Repository**, ver figura 4.4.

Listeners

Los Listeners son entes con alta cohesión interna, están destinados a prestar un servicio específico. Los Listener que existen en el SUCHAI son:

- **taskConsole:** Consola serial, que permite la comunicación entre un PC escritorio y el SUCHAI, se emplea para propósitos de Debug, de modo de ejecutar Comandos y ver los resultados.
- **taskHousekeeping:** Encargado de generar los comandos de actualización de las variables de estado y reaccionar ante situaciones críticas. Un ejemplo clásico es el estado de carga de las baterías, que al llegar a un cierto nivel gatilla la ejecución del comando de ahorro, previniendo la transmisión de Telemetría y/o Beacon, las cuales consumen energía.
- **taskCommunication:** Encargado de ejecutar el tipo de comunicación que tendrá el SUCHAI, dependiendo del **modo de operación** del satélite (que es una variable de estado). Existen dos modos implementados actualmente:
 - **Modo TC:** Se produce ante la llegada de telecomandos al TRX. Si se detecta aquella situación, los telecomandos son rescatados y ejecutados uno a uno.
 - **Modo RSSI:** Debido a que con el TRX usado en el SUCHAI (de la compañía

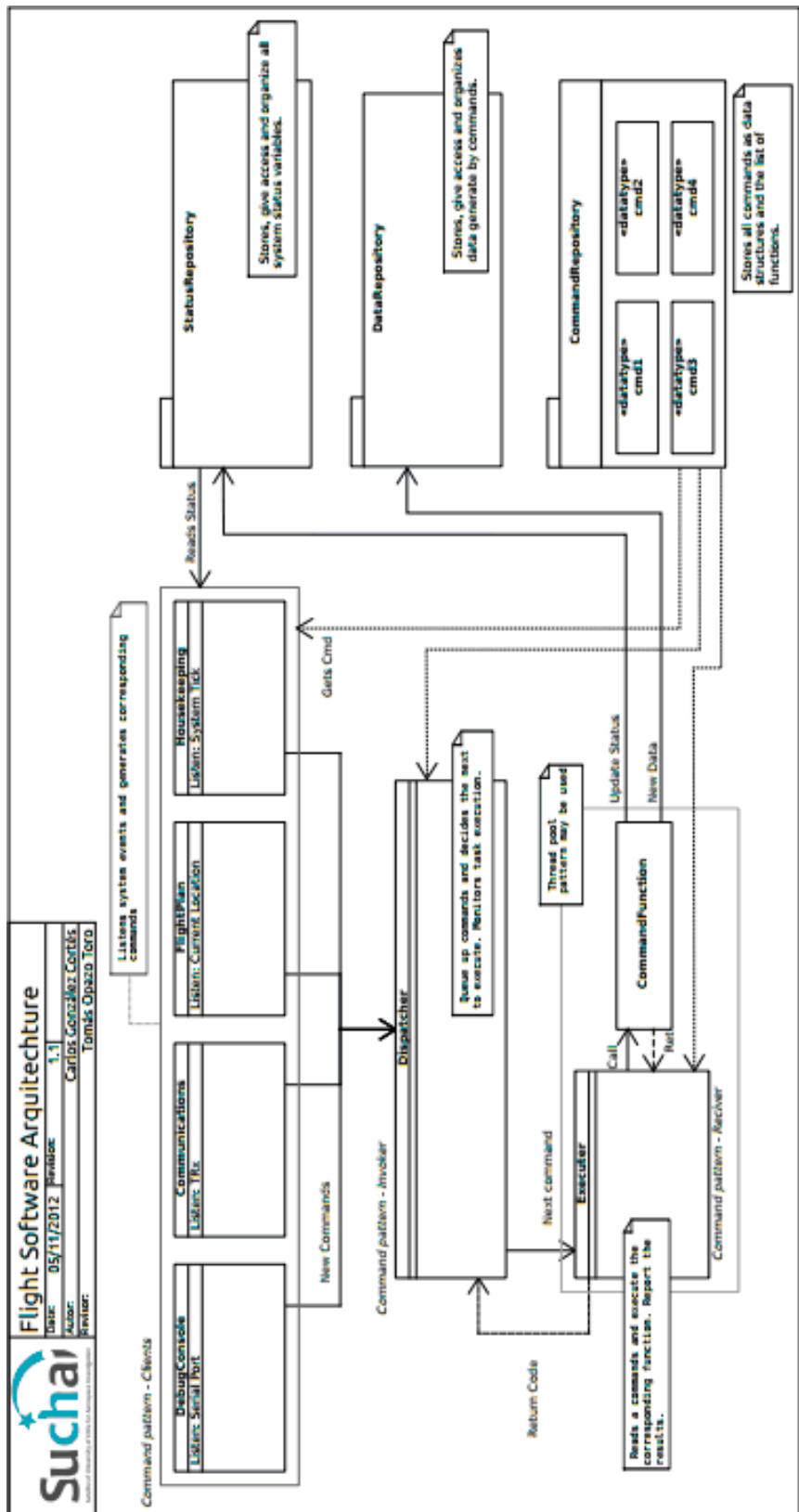


Figura 4.4: Arquitectura del bus SUCHAI

Allspace), no se ha logrado recibir telecomandos de manera robusta, se debió implementar este método, que corresponde a la detección de un carrier en la frecuencia de comunicación **con un patrón determinado**, como por ejemplo, que este activo por 30 segundos, y menos de 60. Luego de lo cual se transmite toda la telemetría predeterminada (la recepción de esta última si ha sido robusta).

- **taskFligthPlan:** Encargado de ejecutar el itinerario precargado de comandos para la inicialización, ejecución y post procesamiento de payloads. Existen en dos variantes. La primera, llamada taskFligthPlan y con resolución de un minuto, que ejecuta el comando asociado a la hora actual, puede ser reconfigurado a través de telecomandos. Y el segundo tipo, llamado taskFligthPlan2 que ejecuta comandos de payloads hasta cada 4 segundos, pero que no puede ser reconfigurado.

En el SUCHAI, estos Listeners son tareas periódicas y que repiten su actividad cada cierto intervalo. Ellos tienen acceso al Command Repository pero **no ejecutan directamente los comandos** sino que solo los encolan (envían) al Dispatcher, esto es muy restrictivo, pues además **no pueden ejecutar (llamar) funciones**. Para permitir una lógica de control sobre que comandos encolar (y así evitar que sean simples listas inmutables de comandos), los Listeners **tienen acceso de lectura al Status Repository**. Dicho de otro modo, pueden leer el valor de las variables de estado y decidir que comando encolar o no en base a ellas. Como ya se menciono, **los comandos tienen la plena libertad de modificar cualquier aspecto del satélite, incluyendo las variables de estado, con lo que con este lazo cerrado se pueden implementar maquinas de estado y algoritmos de control en los listeners**. De esa forma se asegura que cualquier maquina de estado implementada en un Listener tenga que tener todos sus estados en el Status Repository.

Status Repository

Este es el lugar en donde se mantienen las variables de estado del satélite. Descrita anteriormente, corresponde físicamente a la memoria EEPROM y está destinada **únicamente a este fin**. Al ser un componente externo y no dependiente de electricidad para conservar sus datos aunque el OBC del SUCHAI se resetee múltiples veces o incluso se apague por completo los valores de las variables de estado son conservados, así toda maquina de estado podrá seguir funcionando desde el lugar y con la configuración en que quedó por ultima vez.

Dispatcher

El SUCHAI se encarga de sus múltiples tareas a través de Listener concurrentes, encargados de generar los comandos respectivos. Ahora bien, estos comandos no son ejecutados por los Listeners, sino encolados (enviados) a un Dispatcher. En estricto rigor, y como ya se había mencionado antes, es el Command Handler el encargado de decidir si retransmitir el comando recibido al subsistema de destino, o enviarlo al Executer del C&DH para ejecutarlo directamente.

En la arquitectura del SUCHAI no existen la posibilidad de que los subsistemas ejecuten comandos, esto es debido a dos razones. La primera es que el grado de complejidad de los subsistemas es lo suficientemente bajo aún como para que solo sea necesario configurar registros para ponerlos en operación. La segunda y más importante, es que los subsistemas adquiridos en el SUCHAI (y en general todos los desarrollos comerciales para Cubesats) son desarrollos aislados, llamados en inglés "Custom Design". En ellos, la única estandarización es de cara al bus PC/104. Cada fabricante cuida de no crear colisiones entre pines usado por placas de otras compañías conocidas, pero no hay ninguna estandarización hacia capas superiores. Por ello, los protocolos de comunicación son distintos entre distintos subsistemas (I2C en la EPS, I2C y UART en el TRX), y la forma de configurarlos y modificar su comportamiento también es muy distinta.

Es por las anteriores razones que no existe un Dispatcher como tal en el SUCHAI, sin embargo, el termino es **reutilizado** para describir a la tarea encargada de recibir los comandos producidos por los Listener y enviarlos al Executer.

Permisos de ejecución SysReq

En el SUCHAI el Dispatcher cumple otra importante tarea, que es decidir si un determinado comando puede o no ser enviado al Executer (y con ello ejecutado). El objetivo es discernir si el satélite cuenta con los Requerimientos mínimos para usar ese comando. Cada comando tiene asociado un valor llamado **SysReq**, por "Requerimientos del Sistema", actualmente este SysReq equivale a un estado de carga de la batería tal, que ejecutar ese comando no ponga en peligro a la operación normal del satélite. Es decir, **SysReq es el nivel mínimo de energía con el que el comando asociado se puede ejecutar**. Si el nivel actual de energía (que es una variable de estado), no permite ejecutar el comando, este es rechazado y no enviado a Executer. Es a través de el manejo de SysReq que se implementa el **Presupuesto Energético** en el satélite.

Executer

Es la ultima pieza del eslabón de ejecución de un comando. El Executer se encarga de ejecutar el comando, es decir, llamar a la función correspondiente con los argumentos correspondientes. En el SUCHAI existe solo un Dispatcher que recibe los comandos desde múltiples Listener, y existe solo un Executer que recibe los comandos desde el Dispatcher. El que exista solo un Executer implica que solo un comando a la vez puede ser ejecutado, lo que hace más determinista el comportamiento del sistema, pues **una vez encolados la ejecución de comandos es secuencial y atómica**.

Dado que es el encargado de la ejecución de los comandos, el Executer debe tener el suficiente poder de computo en memoria RAM para llevar a cabo su trabajo, es la entidad más demandante en recursos computacionales, pues los comandos tienen acceso a todas las

funciones de los drivers, y pueden operar sobre cualquier componente en el satélite.

Data Repository

Visto anteriormente, pero desde una capa inferior, el Data Repository es el ente que contiene al conjunto de funciones de acceso y almacenamiento de la memoria SD del C&DH. Todos los datos de payloads son almacenados en este repositorio para ser posteriormente enviados por telemetría. La cola de telecomandos recibidos por el TRX desde tierra es almacenada en él, al igual que el Flight Plan. Dado que no se implemento un sistema de archivos FAT o similar, se creo un sistema de gestión rudimentario para separar y acceder a arreglos o buffers de datos de manera de separar y asignar la memoria SD para diferentes usuarios.

Command Repository

Esta estructura es la encargada de almacenar los comandos del SUCHAI. Cada comando tiene asociado: una **función**, un **número único identificatorio**, y un **nivel de requerimiento energético (SysReq)**. De esta manera los Listener acceden solo al numero identificatorio, llamado simplemente **ID** del comando, y bajo ese ID lo encolan al Dispatcher. La **relación ID-función es absolutamente univoca**, sin embargo esto no tiene que ser necesariamente así para SysReq.

El Command Repository no existe físicamente fuera del OBC, si no que es una estructura en memoria RAM del software SUCHAI y que contiene buffers con los comandos del SUCHAI. Cada vez que se reinicie el OBC la estructura es inicializada nuevamente, esto significa que se asocian de nuevo los tres componentes de un comando.

4.1.5. Resumen

En la figura 4.5 se muestra un resumen de la arquitectura del SUCHAI, donde se puede ver que modulo o componente corresponde a que capa del modelo OSI en la analogía de las secciones anteriores.

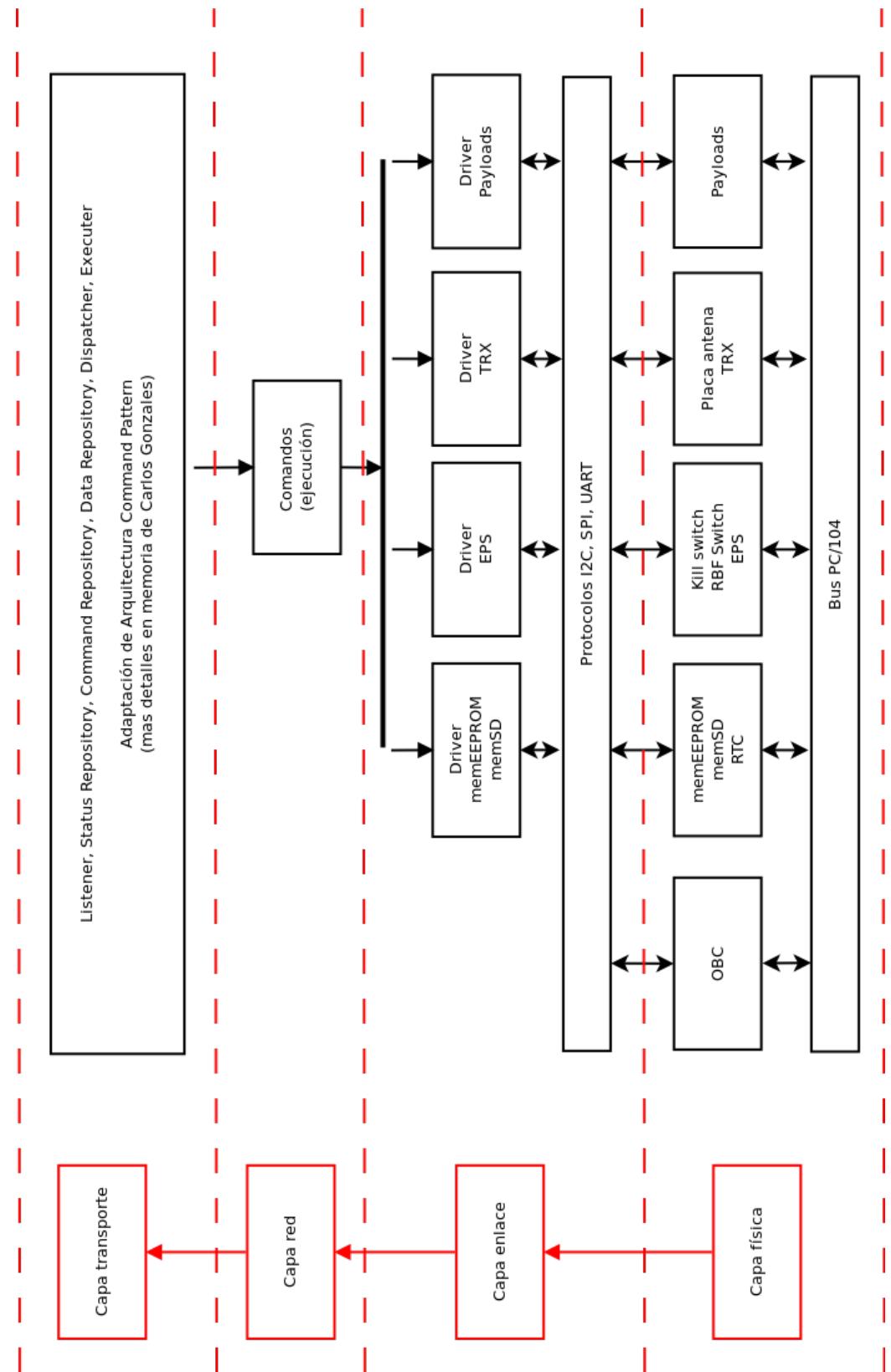


Figura 4.5: Resumen de la Arquitectura del bus SUCHAI

4.2. Estados de operación del SUCHAI

4.2.1. Despliegue

En este caso, y al igual que la gran mayoría de los satélites, si no todos, se requiere de un proceso de despliegue e inicialización de sus sistemas. En el SUCHAI esta tarea es llevada a cabo por el **taskDeployment**, esta es un precursor de la arquitectura de software descrita anteriormente (ver figura 4.4). Cada vez que el OBC se reinicia (y con ello el software SUCHAI) la primera tarea que se ejecuta es esta. Solo después de finalizado con éxito la ejecución de taskDeployment se puede dar paso a la estructura de Listeners, Dispatcher, Executer y Repositorios antes mencionados.

La ejecución de esta tarea **debe** ser estrictamente secuencial, pues la inicialización de por ejemplo, el repositorio de Datos es posible solo después de la inicialización de la tarjeta SD. Es por tanto natural presentar las decisiones y acciones de la tarea de despliegue en la forma de **diagramas de flujo**, ver figura 4.6, 4.7, 4.8, 4.9 y 4.10.

Los pasos principales en esta etapa en el SUCHAI son ejecutados por las funciones mostradas a continuación:

1. dep_silent_time: Inactividad por 30 min (requerimiento de CDS [22])
2. dep_init_Peripherals: Inicialización de componentes externos (memorias externas, subsistemas, etc)
3. dep_init_Repos: Inicialización de Repositorios (Status, Data y Command)
4. dep_deploy_antenna: Despliegue de antenas
5. dep_launch_tasks: Inicio operación normal software SUCHAI

dep_silent_time

Debido a que los Cubesat son lanzados en grupos, y para no dañar físicamente ni eléctricamente a los vecinos, en el CDS [22] se estableció el requerimiento de que todo despliegue de partes móviles y/o radiación electromagnética fuese al menos 30 minutos después de que el cohete los libere. Esto permite que se encuentre suficientemente separados unos de otros para no interferirse.

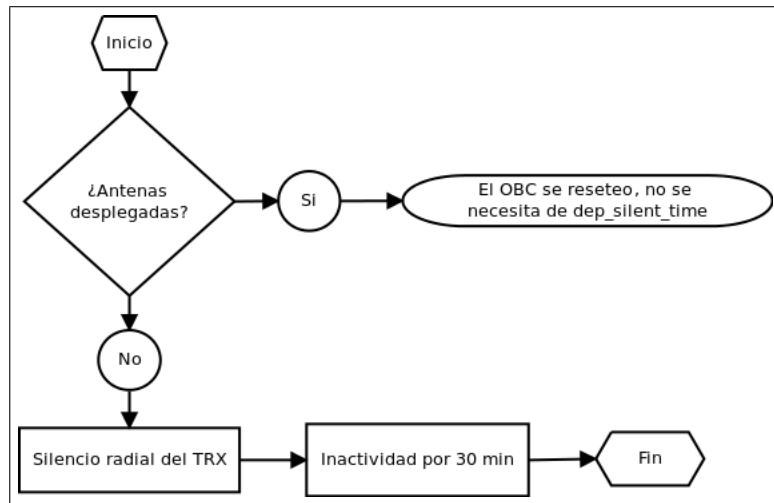


Figura 4.6: Diagrama de flujo de dep_silent_time

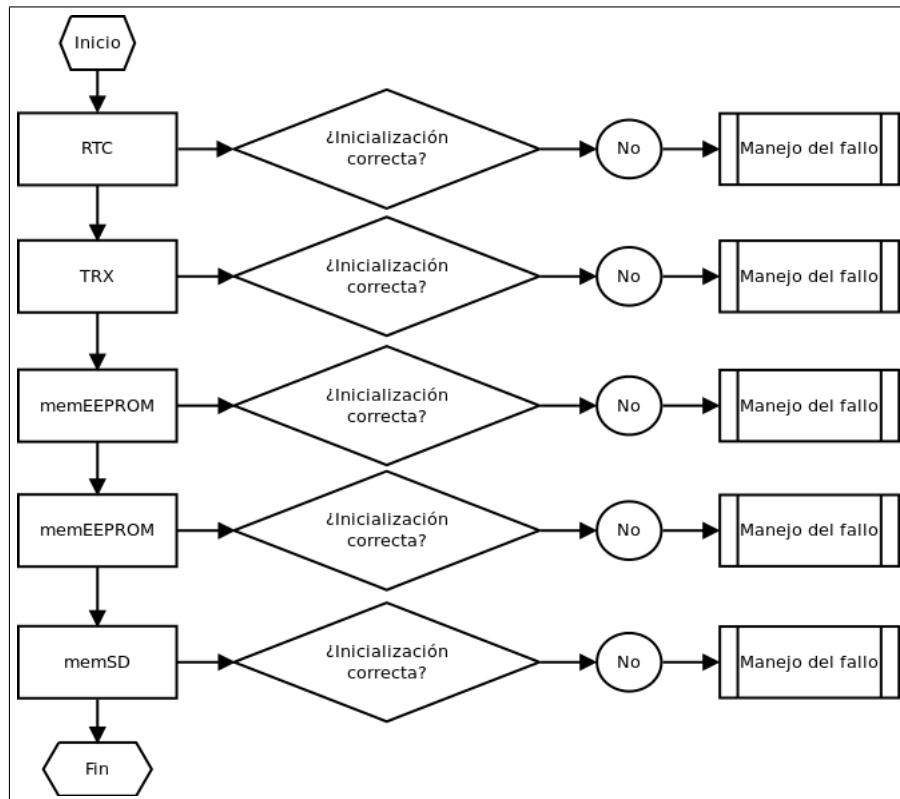


Figura 4.7: Diagrama de flujo de dep_init_Peripherals

dep_init_Peripherals

En esta etapa se realizan las inicializaciones de **todo el hardware del bus SUCHAI que sea manejado por el OBC**. Todos los subsistemas y sus componentes son configurados en sus valores por defecto (por ejemplo el RTC, memEEPROM y memSD del C&DH). Vale recalcar que los Payloads no son parte del bus y no caben en esta categoría, son inicializados en una etapa posterior, y solo cuando son enviados los comandos de ejecución de ellos por

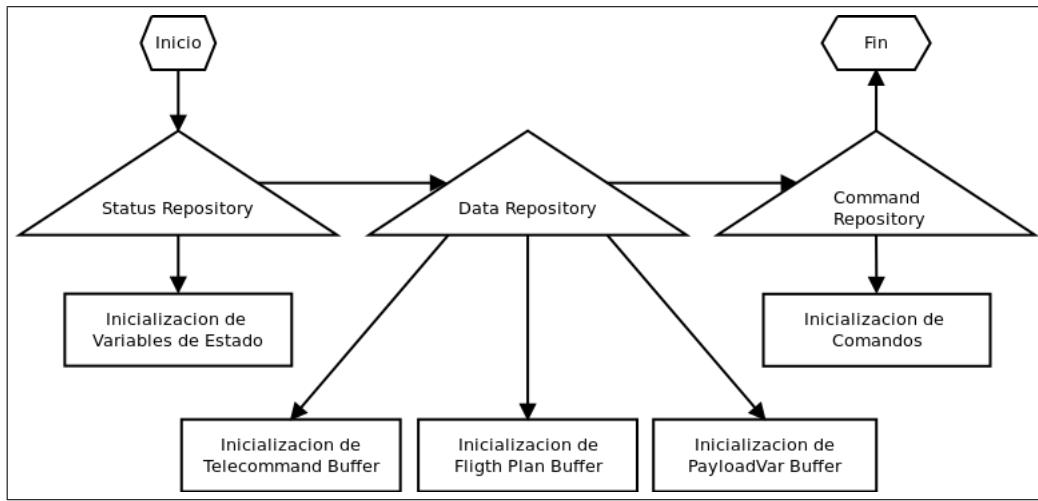


Figura 4.8: Diagrama de flujo de dep_init_Repos

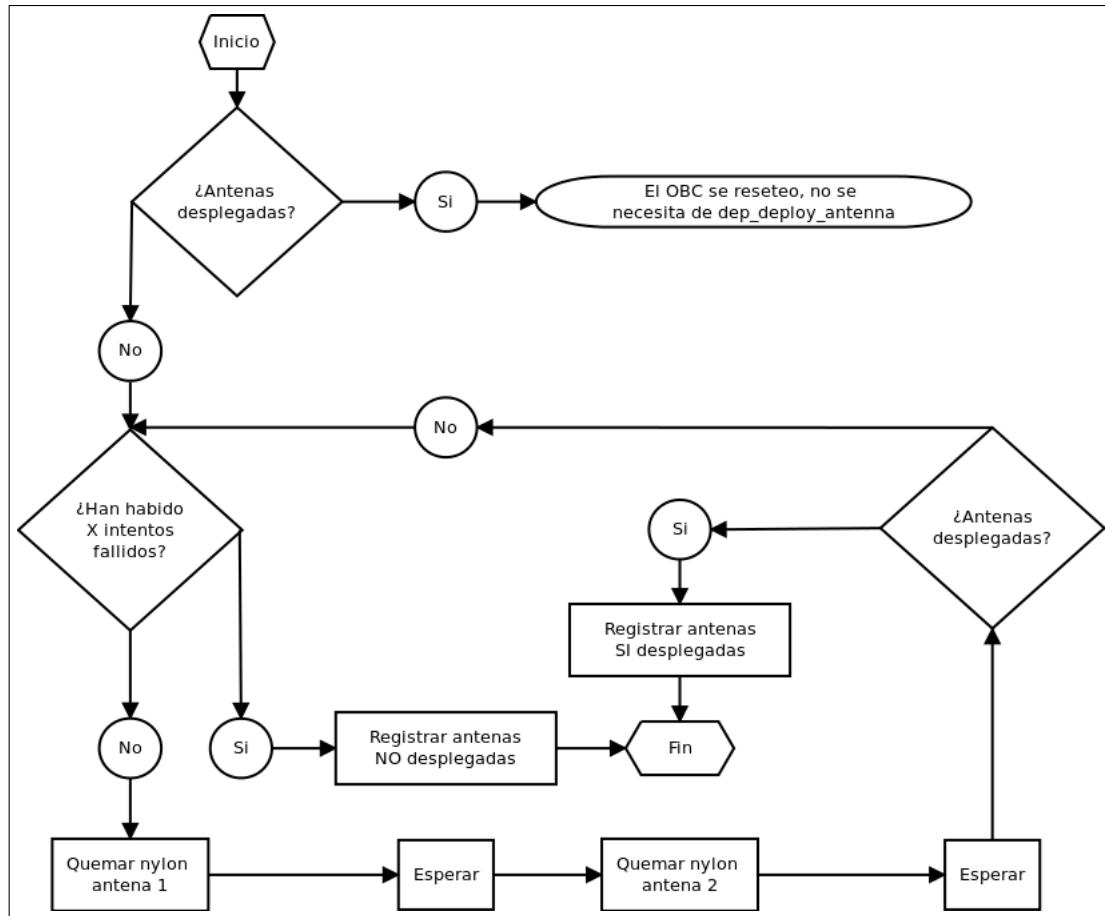


Figura 4.9: Diagrama de flujo de dep_deploy_antenna

Flight Plan.

Ya que la inicialización es sobre hardware externo al OBC y que esta puede fallar, se deben implementar y ejecutar rutinas en caso de fallo. La decisión de cuáles serán cubiertos y hasta qué grado es una decisión de diseño, pues influye en el costo y tiempo de la programación.

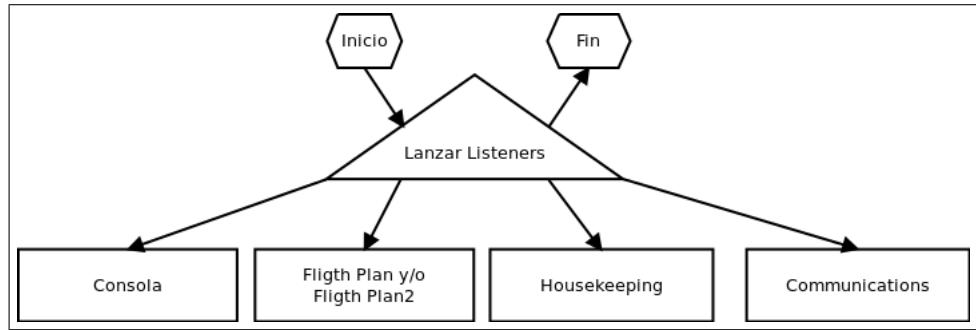


Figura 4.10: Diagrama de flujo de `dep_launch_tasks`

Es en estos casos de fallas de inicialización que conceptos como **redundancia, seguridad, resiliencia, adaptación y reconfiguración** de software y sistemas entran al juego [14].

`dep_init_Repos`

En esta etapa, las estructuras **internas** del software son inicializadas. Tales son el Status, Data y Command Repositories. Estas estructuras en caso de estar contenidas en hardware externo solo pueden ser correctamente inicializadas **después** de `dep_init_Peripherals`, por esta razón nuevamente se recalca que la secuencialidad en esta etapa es crucial, similar a lo que sucede, guardando las proporciones, en el arranque o **bootstrapping** de los PC.

En el caso del Status Repository el comportamiento por defecto es dejar intactas las variables, salvo el caso de aquellas que necesitan ser modificadas al haber un Reinicio del OBC, como la variable **numero de reset**, o la variable **horas sin reset**. Hay otras como la variable **despliegue correcto** que evidentemente no pueden ser modificadas en esta situación.

En el caso del Data Repository el comportamiento es dejar intactos los arreglos de datos (buffers) de Flight Plan y Payload, pero vaciar el de Telecomandos, pues pudo haber habido un comando inexistente o alguna situación que haya causado el reinicio anormal.

En el caso de command Repository la inicialización es siempre la misma, pues la lista de comandos es por ahora solo modificable **en tiempo de compilación**, es decir que luego del despegue el satélite no puede agregar nuevos comandos y por tanto la lista es inmutable.

`dep_deploy_antenna`

En esta etapa las antenas son desplegadas y luego del proceso se actualiza la variable de estado **antenas desplegadas** indicando si el proceso fue exitoso o no. Adicionalmente se guarda la fecha y hora del despliegue y el numero de veces que se intentó.

dep_launch_tasks

Una vez terminadas las inicializaciones de hardware y software, se procede a lanzar las tareas de operación nominal, o normal. Luego de ocurrido esto la tarea taskDeployment ya no es necesaria (lo es solo luego de cada reinicio) y es eliminada. Es solo a partir de este punto que se cuenta completamente con la adaptación de la Arquitectura de Command Pattern.

4.2.2. Operación Nominal

En la operación nominal del SUCHAI, el comportamiento está dado por la superposición de varios Listeners que ejecutan funciones específicas. Más explícitamente, los cuatro Listeners involucrados son **taskCommunications**, **taskConsole**, **taskHousekeeping** y **taskFlightPlan/taskFlightPlan2**. En todo momento el satélite se encarga de actualizar sus variables de estado y de chequear su estado de salud, reaccionando ante eventos preprogramados. El software también se encarga de ver que cada payload se ejecute, y que luego se guarden sus datos en el Data Repository. Finalmente, el satélite está siempre atento a recibir telecomandos desde tierra o en su defecto, la señal RSSI y enviar en respuesta la telemetría adecuada.

Estando en vuelo, el Listener de taskConsole no es necesario pues evidentemente no habrá un computador conectado al satélite para trabajar en el depurándolo y/o enviándole comandos. Respecto a los Listeners que quedan, Housekeeping se encarga de actualizar solo las variables de estado que le son explícitamente encargadas, pues las otras son solo modificadas por el Listener (incluyendo taskDeployment) del que son parte. Esto sucede así porque tanto taskCommunication como taskFlightPlan(2) son máquinas de estado finita o FSM, y sus variables de estado son justamente aquellas que taskHousekeeping no interfiere. En este sentido el SUCHAI en su operación nominal funciona en base a dos máquinas de estado, una de comunicaciones y otra de manejo de payloads implementadas ambas a través de Listener y del Status Repository.

Execute Before Flight

Ya que muchas de las acciones que se llevan a cabo son producto de las mencionadas máquinas de estado, se le debe indicar al software cual es su estado inicial. Por ejemplo, el despliegue de antenas de la figura 4.9, muestra claramente que una vez que se haya realizado el despliegue exitoso, ese valor es almacenado en el Status Repository y no es modificado posteriormente, por tanto la única forma de que intente nuevamente el despliegue es borrando ese valor (modificar la variable en cada reinicio sería incorrecto). La misma situación sucede con los payloads y otras variables de estado, que al no ser modificadas en el reinicio del OBC (porque no deben serlo) quedan inalteradas.

Por lo anteriormente dicho, antes de la integración del SUCHAI con el cohete se debe ejecutar el comando **executeBeforeFlight**. Este comando inicializa las variables de estado del satélite, de modo que este sepa al despertar **que es la primera vez que se enciende**. Posteriormente siempre tendrá noción de cuantas veces se ha reiniciado. Además de las variables de estado, arreglos (buffers) como los de FlightPlan y el de Telemandos deben ser precargados, en el primer caso llenado con el itinerario por defecto, en el segundo vaciado por completo.

FSM de Payloads y Comunicaciones

Asociado a cada payload, existe una variable de estado llamada **dat_pay_xxx_perform**, donde xxx es el nombre del payload (por ejemplo, sensor de temperatura, GPS, cámara, giroscopio, etc). Esta variable de estado indica si el payload debe estar en ejecución o no. En caso positivo, FlightPlan genera los comandos adecuados para que se ejecuten.

En el SUCHAI y con el objetivo de lograr un **funcionamiento estándar e independiente respecto a los payloads**, se ideo una FSM (maquina de estado finita, o autómata finito determinista) que posee cuatro estados, y es la misma para **todos los payloads**, ver figura 4.11. Desde FlightPlan se ejecuta esta maquina de estado, para cada payload en el SUCHAI. Como toda maquina de estados, posee entradas salidas, un reloj y estados internos (memoria). La entrada corresponde a la variable de estado **dat_pay_xxx_perform**, la salida corresponde a comandos enviados por la FSM a taskDispatcher (a través de taskFlightPlan), los estados de la maquina son cuatro, y cada estado tiene unívocamente asociado un solo comando (o ninguno). El reloj está dado por la frecuencia con que se llame desde FlightPlan a esta maquina de estados, y como cada estado tiene asociado un y solo un comando, cada llamada solo genera una acción por actualización de la FSM.

pre_init

En el primer estado, llamado **pre_init**, en el que **parten y permanecen** todos los payloads que no tengan activo su variable de estado **dat_pay_xxx_perform** correspondiente, en este estado no se envía ningún comando, y su objetivo es dejarlo en inactividad total.

init

Los que si tengan activa la variable **dat_pay_xxx_perform** pasan al segundo estado, llamado **init**, en él se envía el comando que inicializa los arreglos (buffers) de Data Repository del payload correspondiente y **se determina la cantidad de muestras que podrá guardar el experimento** (ver figura 4.11).

take

El estado init es transitorio y por él solo se pasa una vez, el siguiente estado es el llamado **take**, en el, se toman las muestras, adquieren sensores, realizan experimentos etc, de modo que se obtengan los datos, estos son guardados en el arreglo (buffer) asociado a cada payload.

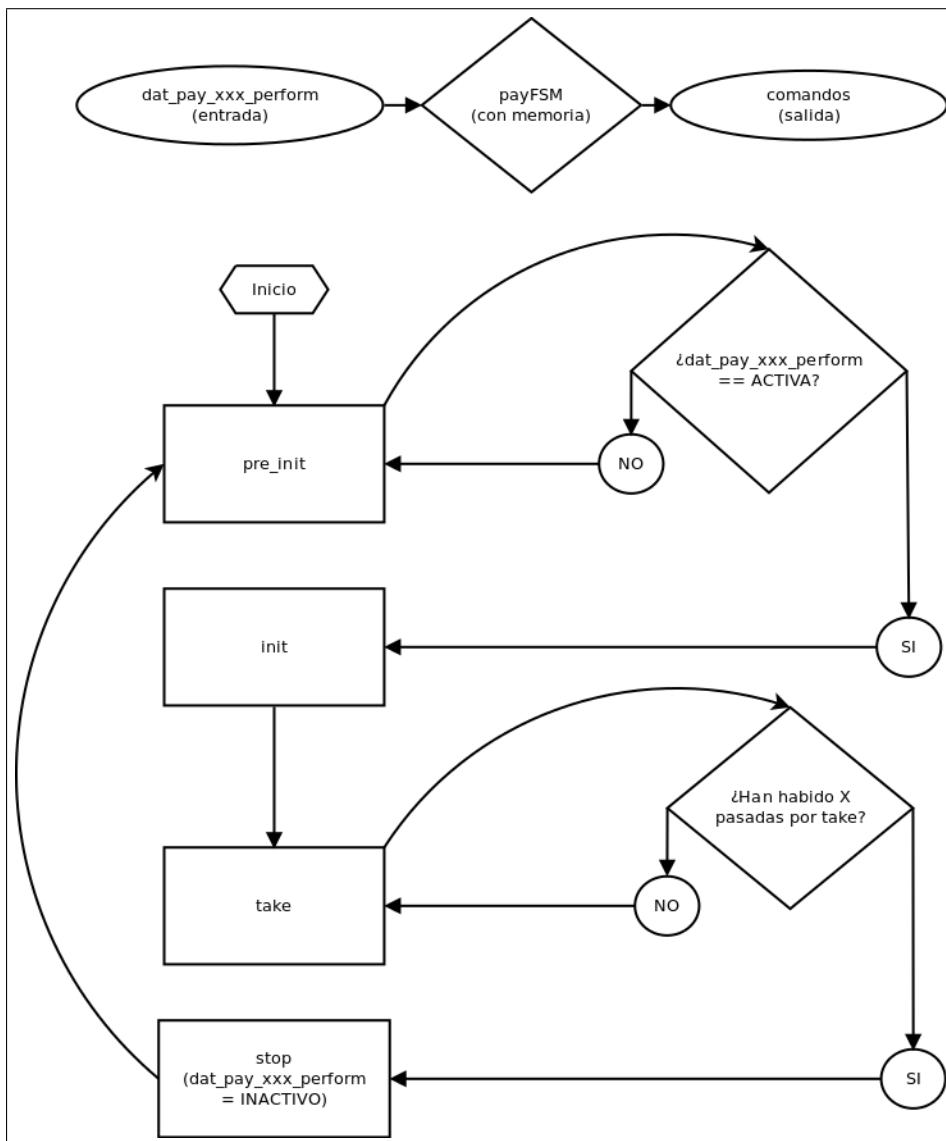


Figura 4.11: Diagrama de la maquina de estado de payloads

Esta operación se repite X-veces (ver figura 4.11) luego de lo cual el buffer se encuentra lleno. Cuando se detecta esta condición la maquina de estados pasa al estado stop.

Vale recalcar que la FSM solo es llamada en intervalos discretos por el taskFlightPlan, por lo que si el comando asociado a **take** toma Y datos por cada llamada, entonces el estado **take** sera llamado $\frac{X}{Y} = Z$ veces antes de pasar al siguiente. Dicho de otro modo, si el comando asociado a **take** toma Y datos por ronda, y se necesitan Z rondas, el arreglo (buffer) debe ser de largo $ZY = X$.

stop

En esta ultima etapa se detiene al payload, se apagan sus sistemas y además se realiza todo el post-procesamiento necesario para dejar en el mismo arreglo (buffer) asociado, solo la información estrictamente necesaria de ser transmitida a tierra. Este estado es único, al igual

que **init**, por lo que es solo un punto de transito para pasar nuevamente al estado **pre_init**. En el estado **stop** se desactiva la variable `dat_pay_xxx_perform` correspondiente, de modo que al llegar nuevamente al primer estado quede sin volver a ejecutarse. Esta condición se reactivara **solo cuando la telemetría del payload haya sido enviada a tierra**, esta operación la realiza no la FSM de payloads, sino que `taskCommunications` luego de haber armado y enviado la telemetría. Solo después de haber enviado la telemetría y reactivado la variable `dat_pay_xxx_perform` correspondiente, el payload se volverá a ejecutar.

El funcionamiento se implemento de la forma antes expuesta por dos motivos. El primero es debido a la falta de recepción segura de telecomandos desde tierra, por lo que la información de los payloads no podía ser solicitada Bajo Demanda (On Demand), por lo que un esquema simple era necesario. La segunda razón que refuerza el modelo implementado es que la información de la totalidad de los payloads de un Cubesat en la actualidad **no son en tiempo real**. Por lo que la información es siempre cuantizada y enviada a tierra en paquetes. Entonces, para cada experimento se puede calcular un largo de paquete mínimo que haga coherente a la información contenida en el. Ese largo es el mismo que el destinado al arreglo (buffer) asociado.

A modo de ejemplo, dado el experimento de sensores de temperatura, se determina que la unidad mínima coherente de información es una órbita completa a la tierra. Luego se calcula que si una órbita durase 4 horas, y si se toman datos cada 1 minuto se tendrán 240 datos necesarios para cubrir la órbita. Entonces se asigna ese número al arreglo (buffer) del sensor de temperatura. Entonces una vez iniciada la FSM de payloads, se capturarán cada 1 minuto las mediciones, y luego de 4 horas (1 órbita) se habrá llenado el arreglo (buffer). De esta manera simple se puede tener datos coherentes y solo volver a ejecutar el experimento cuando esa información haya sido descargada, pues de nada sirve ejecutar y producir información que no pueda ser recuperada. La interrelación entre la FSM de payloads y la telemetría de estos se puede ver en la figura 4.12.

Formato de la Telemetría

La telemetría en el SUCHAI es toda aquella información que el Subsistema de Comunicaciones envía (y que se espera, la estación terrena decodifique). Debido a las características específicas del TRX a bordo, la transmisión se realiza en unidades discretas y constantes llamadas **frames** de telemetría. Estos frames consisten en 128 bytes, lográndose en pruebas de laboratorio, tasas de transmisión cercanas a un frame por segundo ($0,125[\frac{kB}{seg}]$). Considerando una ventana de tiempo de 10 minutos (duración promedio estimada) en que hay contacto con el satélite, se pueden descargar.

$$1\left[\frac{frames}{seg}\right] \cdot (10 \cdot 60)[seg] = 6000[frames] \Rightarrow 750\left[\frac{kBytes}{pasada}\right] \quad (4.1)$$

A partir del cálculo de 4.1 podemos ver que tanto el ancho de banda ($0,125[\frac{kB}{seg}]$) como el

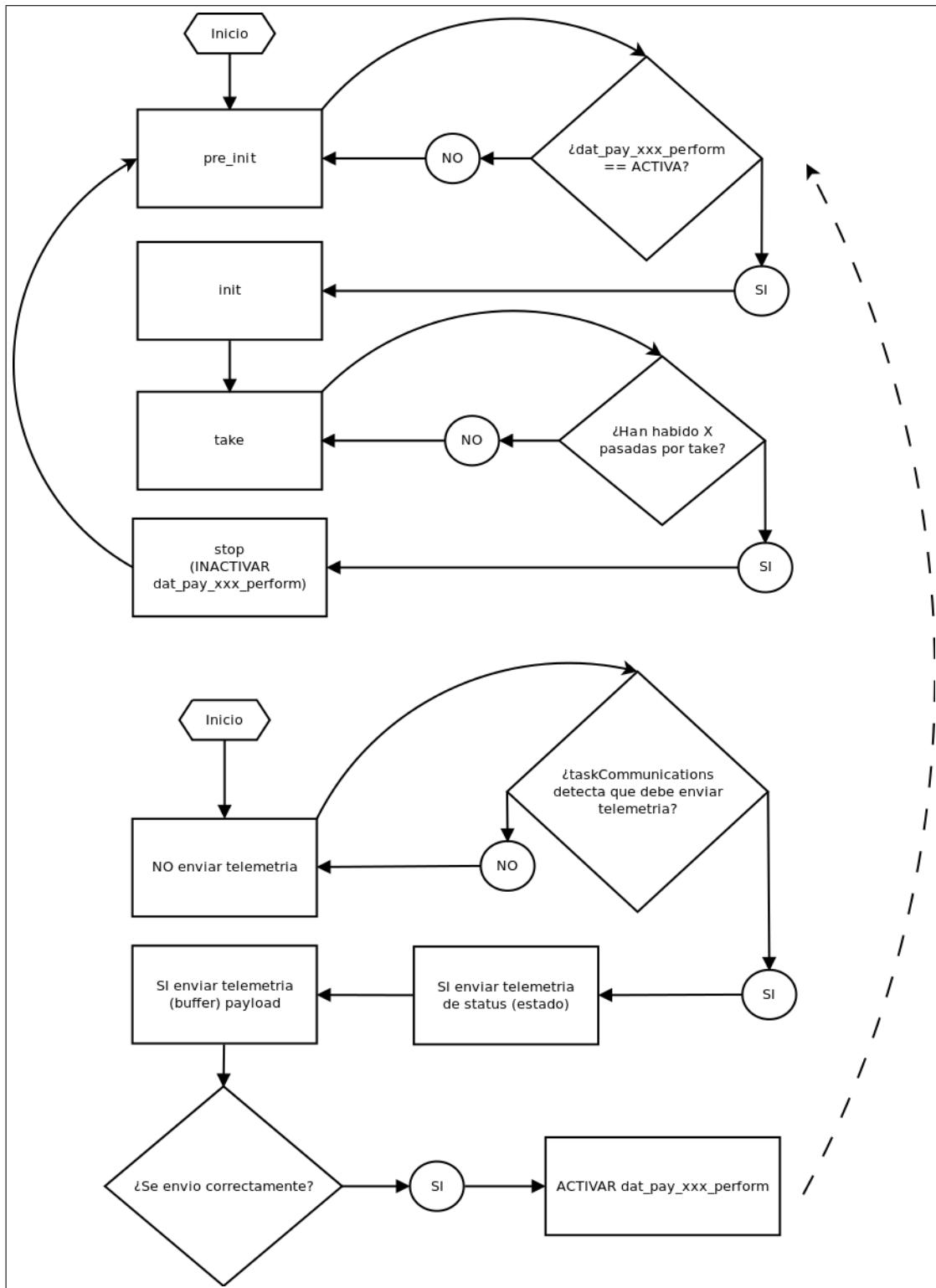


Figura 4.12: Diagrama de la maquina de estado de payloads (arriba) y la de telemetría (abajo)

producto final ($750[\frac{kB}{pasada}]$) son muy pequeños, de hecho el ancho de banda es incluso inferior a un módem telefónico ($44[kbps]$).

Byte	Sección	Tipo	Descripción
1,2	Control de frame	Tipo de frame	Tipo de frame: (1) Inicio; (2) Fin; (3) Continuación; (4) Simple
3,4	Control de frame	Numero de frame	Para telemetría de múltiples frames, cada frame se identifica con un numero correlativo
5,6	Telemetría	Código de TM	Identifica el tipo de telemetría que contiene el frame
7 .. 128	Telemetría	Datos	Datos, agrupados como int de 2 BYTES con formato BIGENDIAN: [7 D1H] [8 D1L] [9 D2H] [10 D2L] ...

Tabla 4.5: Control de telemetría (frames) para el SUCHAI

El producto de la compañía Allspace, provee de un sistema transparente para el usuario. Cada frame enviado por el TRX a bordo del satélite es, o bien descartado completamente producto de errores de comunicación (errores de bit en la decodificación), o bien aceptado completamente (recepción sin errores). El contenido de cada frame es de libre uso, en el caso del SUCHAI se decidió implementar un sistema de control de frames simple, con una carga (Overhead) de $\frac{6}{128} \Rightarrow 4,68\%$.

En el cuadro 4.5 se puede observar la estructura de control, consistente en dos secciones principales. La primera es llamada **control de frame**, y se emplea para distinguir si el frame recibido es único o parte de un paquete mayor que debió ser dividido en múltiples frames para ser enviado. En el caso de las transmisiones satelitales del SUCHAI no se detectó en prueba alguna la necesidad de preverlos llegar fuera de orden, sin embargo, aunque este caso se diese el control de frame implementado permite reordenarlos. Esto ya que cada paquete enviado lleva un numero que los ordena correlativamente.

La segunda sección corresponde a la llamada **sección de telemetría** en donde se transmiten los datos, encabezado previamente por un identificador del tipo. El identificador corresponde al generador de la información, y es de dos tipos: **Telemetría de estado** o **Telemetría de payloads**. Teniendo cada payload un identificador único.

En el ejemplo de la figura 4.13, se puede observar que el primer frame corresponde a telemetría de estado, esta información cabe en menos de 128 bytes, por lo que su tipo es de **inicio**. Luego, su identificador o **código de TM** es 6 y diferente al de los demás frames, que son de otro tipo. En la quinta columna se ve que su tipo es de **continuación** y esto es confirmado por el **numero de frame**, que parte en 0 y crece en cada envío desde el satélite (envíos consecutivos). Se puede observar cómo las columnas 5, 6 y 7 (frames 2, 3 y 4) corresponden a un mismo tipo (mismo **código de TM**), en específico corresponden al payload sensor de temperatura, que envía los valores captados en su arreglo (buffer).

dat_CubesatVar	PayloadBuff	Frame1 (hex)	Frame2 (hex)	Frame3 (hex)	Frame4 (hex)
		100	300	300	300
0=pues cubesatVar cabe en 1 frame		0	0	1	2
6=ID TM cubesatVar	pay_i= ID TM	6	5	5	5
dat_CubesatVar[0]=0x1	MaxPayIndx	1	39	165	015C
dat_CubesatVar[1]=0x4	NextPayIndx	4	40	015F	0
dat_CubesatVar[2]=0x0	Buffer	0	166	015C	164
dat_CubesatVar[3]=0x0		0	160	0	160
dat_CubesatVar[4]=0x71		71	160	164	015C
dat_CubesatVar[5]=0x1		1	0	160	0
dat_CubesatVar[6]=0x3		3	165	015C	164
dat_CubesatVar[7]=0x0		0	160	0	160
dat_CubesatVar[8]=0x1		1	015F	164	015C
dat_CubesatVar[9]=0x0		0	0	160	0
dat_CubesatVar[10]=0x0		0	165	015C	164
dat_CubesatVar[11]=0x1		1	160	0	160
dat_CubesatVar[12]=0x1		1	015F	164	015C
dat_CubesatVar[13]=0xD		000D	0	160	0
dat_CubesatVar[14]=0x5		5	165	015C	164
dat_CubesatVar[15]=0x3		3	160	0	160
dat_CubesatVar[16]=0xF		000F	015F	164	FFFE
dat_CubesatVar[17]=0x17		17	0	160	FFFE
dat_CubesatVar[18]=0xD		000D	165	015C	FFFF

Figura 4.13: Ejemplo de telemetría recibida desde el SUCHAI (extracto)

4.3. Discusión de lo implementado

La implementación del SUCHAI detallada en este capítulo tiene variantes que se podrían haber realizado de manera distinta, por ello se mencionaran los aspectos que seria interesante explorar y se darán las razones de porque se hicieron finalmente de la manera en que están.

4.3.1. Limitaciones del SUCHAI

Telecomandos

Existe una tercera maquina de estados que no fue presentada en las secciones anteriores, y que es parte de taskCommunications, corresponde a la FSM de Telecomandos, no se presento porque a pesar de estar implementada en el software, no pudo ser probada debido a los errores de recepción del TRX de Allspace y por tanto no es ejecutada por el OBC en una operación nominal, errores que no ocurrirían solo al equipo del SUCHAI, ver [20].

La posibilidad de contar con telecomandos abre la puerta a una gran gamma de posibilidades, incluso reprogramar el OBC, ejecutar comandos específicos (como intuitivamente lo sugiere la palabras tele-comandos), acceder y modificar el Status Repository, agregar nuevos comandos, etc, en definitiva tener mayor control sobre la operación del SUCHAI, y no estar limitados a las FSM implementadas al momento de lanzarse. Sin embargo las características

de ejecutar payloads, almacenar sus resultados y transmitirlos On Demand **si** son cubiertos por la implementación actual a través del RSSI.

Si se llegase a reemplazar el TRX de Allspace por uno robusto en el envío y recepción de TM y TC, luego de su integración, se debería pasar directamente a la etapa de pruebas de la FSM de telecomandos, pues ya está implementada.

FSM de Payloads

Aunque ya se menciono en parte, es importante aclarar que los Payloads que el SUCHAI puede ejecutar son aquellos que:

1. Se pueden integrar al bus PC/104 del SUCHAI (ver 4.1.1)
2. No necesitan transmitir telemetría en tiempo real
3. Pueden ser ejecutados en rondas discretas de tiempo (se realiza TDM en los payloads)
4. Su información relevante puede ser almacenarse en menos de 128 kB (arreglo a ser transmitido como TM)
5. Demanda para su ejecución una capacidad de procesamiento (básicamente RAM) acorde con lo disponible por el OBC del SUCHAI

4.3.2. Lecciones aprendidas

Adquisición y pruebas preliminares

La principal enseñanza en el desarrollo SUCHAI es la constatación de que en el área de Cubesat el subsistema más critico es el de Comunicaciones, esto debido a que en los otros subsistemas importantes (Energía y C&DH) existen más alternativas probadas en el espacio. Los TRX son generalmente uno de los componentes más caros, junto con los paneles solares. Y a la vez son uno de los menos probados, el TRX de Allspace hasta la fecha no lo ha sido en el espacio, y a través del contacto con el equipo del Cubesat "Capitán Beto" se constato que al parecer es un problema extendido, pues aquel equipo adquirió un TRX de la compañía AstroDev, que tampoco tuvo resultados satisfactorios una vez en vuelo. Es por tanto de vital importancia asegurar un link de comunicaciones, y se recomienda por tanto o bien desarrollar un subsistema propio o adquirir uno que ya este suficientemente probado en el espacio.

Organización del trabajo

Otro aspecto complicado para el proyecto, fue no contar con una base más amplia de estudiantes trabajando al menos 6 horas semanales y de manera continua en el laboratorio. Esto hizo atrasar algunas tareas principales y no poder cubrir otras secundarias. La mayor parte del tiempo se contó con 4 estudiantes con la dedicación más arriba mencionada, recibiendo solo aportes esporádicos de los demás. Al mismo tiempo, en casi todas las experiencias de otras universidad en Cubesats existen estudiantes de magíster o bien trabajando en el proyecto directamente o asesorándolo, esto es algo que también habría ayudado al SUCHAI, Por otro lado fue imprescindible haber contado con un ingeniero de proyecto para realizar todas las labores de coordinación y otras de desarrollo como la Placa de Payloads y la de Antenas.

4.3.3. Variantes a explorar

Mayor capacidad RAM

Uno de las principales limitaciones del OBC usado (PIC24) no es su memoria de programa (memoria Flash de 256 KB), sino su limitada memoria RAM (16 KB), lo que dificultó y limitó en varias ocasiones la programación y la ejecución del software SUCHAI. Por ejemplo, a cada tarea (cada Listener) se le debió asignar un espacio de memoria RAM justo y preciso, pues apenas alcanzaba para ejecutarlos a todos concurrentemente (para más detalles se recomienda ver la memoria de Carlos Gonzales). Por la razón antes dicha es que mantener el Command Repository en una memoria externa (memSD por ejemplo) en vez de la RAM es algo deseable, sin embargo no se hizo por falta de tiempo.

En el mismo sentido reemplazar el PIC24 por otro de mayor memoria RAM es algo con ventajas y desventajas. Las ventajas son evitar los problemas recién mencionados, las desventajas seria tener que volver a programar drivers para los protocolos de comunicación y en espacial si hubiese que hacer cambios para poder correr el sistema operativo (FreeRTOS) en él. El encontrar un OBC con mejores capacidades y lo más compatible con FreeRTOS es la mejor opción.

Registro de comandos procesados por Dispatcher

Tener un registro con los comandos ejecutados y no ejecutados por SysReq es una tarea responsabilidad de Dispatcher. Si se hubiese implementado se tendría un registro en la memSD de cada comando ejecutado, los parámetros con los que fue llamado y los valores que retornó, esto permitiría saber exactamente que comando estaba ejecutándose al momento de producirse un error. Otro aspecto interesante sería mezclar esta funcionalidad con un sistema FAT en la memSD, esto permitiría acceder al registro como a un archivo de texto cualquiera. Esta característica no se implementó por falta de tiempo.

Sistema FAT para la memSD

Agregar un sistema de archivos de tipo FAT al almacenaje de la memSD en vez de uno basado en memorias físicas (arreglos para payloads) hubiese permitido estructurar la memoria de almacenaje de manera clásica, como carpetas y archivos dentro de ellas. Y más importante permitiría un añadido transparente de información a cada archivo. Y finalmente hubiese permitido acceder a la información desde un común computador de escritorio. Habría permitido por ejemplo configurar el satélite escribiendo un archivo especial. Esta característica se trato de implementar y quedo en una variante del repositorio, pero debido a problemas de memoria RAM (las librerías FAT son en general pesadas, para estándares de sistemas embebidos) no se pudo continuar, sin embargo es parte de los trabajos futuros a desarrollar por parte del equipo SUCHAI.

Múltiples Executers

Otra variante no explorada es el tener más de un Executer, lo que no fue probado por motivos de memoria RAM. Esta característica estaría controlada por el Dispatcher, que por ejemplo al recibir cierto tipo de comandos podría establecer un sistema donde se les asegurase una ejecución prioritaria. Esto no se implemento por disponibilidad de RAM y tiempo para su programación.

Bus de datos

El SUCHAI adolece en buena parte del mismo problema que otros Cubesat, y es no poseer un bus de datos **único** para el intercambio de información y comandos. Para el intercambio de información se uso I2C, UART y SPI. Además, ya que algunos de estos protocolos (SPI y UART) solo permiten dos nodos y tienen una jerarquía maestro-esclavo, se genera una centralización en la arquitectura del satélite (red tipo estrella) pues todo los subsistemas dependen del OBC y la comunicación entre dos subsistemas es solo posible a través de este. El trabajo futuro en este caso es estandarizar el protocolo del bus de datos, uno como I2C es una alternativa razonable. Al respecto hay dos alternativas, o bien seleccionar solo componentes con el protocolo soportado. La segunda es contar con conversores de protocolos como aquellos de UART-I2C o bien incluir pequeños microcontroladores como PIC18 para tal propósito.

Protocolo CSP

Una variante muy interesante, que no fue implementada en el SUCHAI por haberla conocido cuando el trabajo estaba ya terminado es el de CSP (Cubesat Space Protocol), que es un protocolo equivalente a TCP/IP pero para sistemas embebidos, está pensado para emular el comportamiento de internet en el envío de mensajes entre host (subsistemas) sobre una red, viendo a la estación terrena simplemente como otro punto de esta. Esta implementada

en I2C, UART+KISS, CCSDS y otros protocolos de la capa de enlace. El TRX de Gomspace actualmente trabaja usando CSP, por lo que de ser adquirido para el SUCHAI habría que ampliar el protocolo a los demás subsistemas, lo que necesitaría de algún interfaz. Este protocolo es parte de los trabajos futuros que el equipo del SUCHAI pretende realizar [32].

Capítulo 5

Pruebas al nano-satélite SUCHAI

En el transcurso de un proyecto y en el paso de una etapa a otra existen peligros potenciales que deben ser evitados, tres de ellos se recogen a continuación y de entre ellos se pondrá especial atención en el tercero, el cual acapará todo el foco del presente capítulo.

1. Interrogante 1 ¿ Es la arquitectura/solución propuesta en la etapa de diseño la más eficaz y eficiente para cumplir los requerimientos del proyecto?.
2. Interrogante 2 ¿ Es lo construido realmente la implementación de la arquitectura diseñada?
3. Interrogante 3 ¿ Cumple lo implementado con los requerimientos del proyecto?

5.1. Etapa de Verificación

Para esta etapa se decidió usar como guía la Matriz de Trazabilidad de Requerimientos descrita en el capítulo 3, ver figuras 3.3 y 3.2. En esta, cada requerimiento es verificado para decidir si es satisfecho o no. Para realizar estas pruebas existen diferentes métodos y categorías y en la literatura del área hay más de una versión y taxonomía de ellos. Por lo anterior se explicitarán los empleados para el SUCHAI y su significado.

5.1.1. Métodos de Verificación

1. Inspección: Se entiende que una Inspección consiste en poner a prueba el requerimientos usando alguno de los cinco sentidos del encargado. Ejemplo de lo anterior es la comprobación visual del despliegue de antenas.
2. Test: En un Test se realiza la comprobación a través de la operación directa del aparato a prueba, es una demostración empírica y planificada llevada a cabo de manera elaborada y generalmente usando implementación especial. Por ejemplo la medición de voltajes

usando un multímetro, durante la operación nominal del satélite.

3. Análisis: En una Análisis se emplean modelos matemáticos, simulaciones, algoritmos, gráficos u otros principios científicos para concluir el estado de cumplimiento de un requerimiento. Un caso es el cálculo de enlace o el presupuesto energético, ambos son modelos matemáticos. Generalmente se aplica a aspectos difíciles o imposibles de medir con un test o inspección.

Para cada requerimiento se generó un documento que detalla el procedimiento y el resultado final. Asociado a este documento está la metodología empleada y el nombre de la prueba en la que se basa el documento, ver 3.2.

5.1.2. Pruebas realizadas

Tanto las pruebas parciales como la prueba total se realizaron usando la versión "v1.0" del SUCHAI, que especifica una configuración tanto en hardware como software. Los detalles de cada prueba intermedia, las fechas, procedimiento, los equipos de medición necesarios y en qué consiste la versión "v1.0" se encuentran en el documento "Programa de Pruebas del SUCHAI" adjunto en el apéndice C.

Pruebas parciales

También llamadas como de laboratorio, fueron todas aquellas pruebas intermedias llevadas a cabo durante el desarrollo del satélite o aquellas que tenían como objetivo verificar aspectos que no serían posibles de medir en la prueba de operación. Esto sucedió con el caso las pruebas de carga de baterías, telemetría y con las de reiteración de despliegue tras una falla inicial.

Prueba Operacional

También llamada Prueba Total fue la parte más importante y final de la verificación de la RTVM. En ella se sometió al SUCHAI a condiciones de vacío como las que enfrentará en el espacio, empleando para ello equipos, dependencias y personal del departamento de Física de la Facultad de Ciencias Física y Matemáticas de la Universidad de Chile. Con lo que se logró una simulación realista de lo que sucederá en órbita. Su objetivo era simular la operación desde su lanzamiento hasta varias horas en órbitas, llevar a cabo cargas y no-cargas de baterías según predicciones de radiación solar como simulación de paneles solares y controlar la emisión de Beacon y de Telemetría en los momentos que se requerían (Beacon periódicamente, Telemetría en cada pasada del satélite por sobre Santiago). Lamentablemente no todos estos objetivos pudieron ser satisfechos, sin embargo eso no impidió poder concluir respecto a los requerimientos verificados, ya que otras pruebas de laboratorio los confirmaban. Las imágenes del satélite en la cámara de vacío se pueden ver en las figuras 5.1, 5.2, 5.3 y 5.4

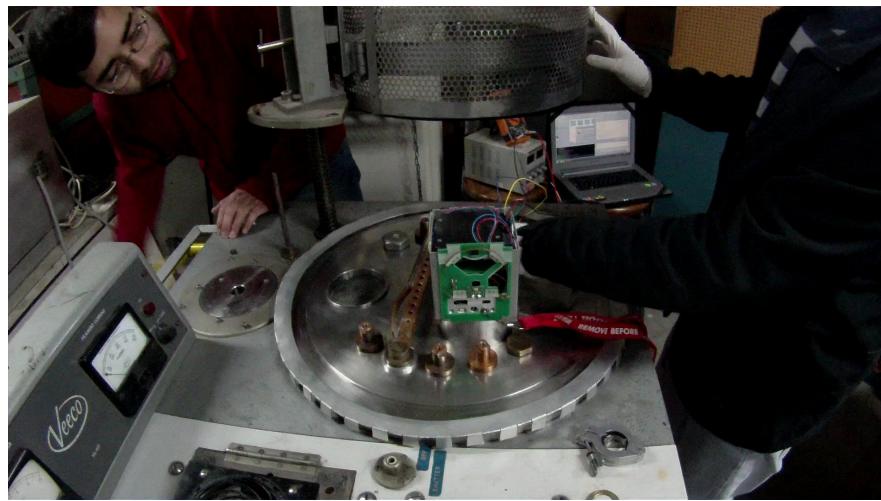


Figura 5.1: Instalación en la cámara de vacío



1 Cubesat SUCHAI
3 Medidor voltaje Batería EPS
5 Cúpula cámara de vacío

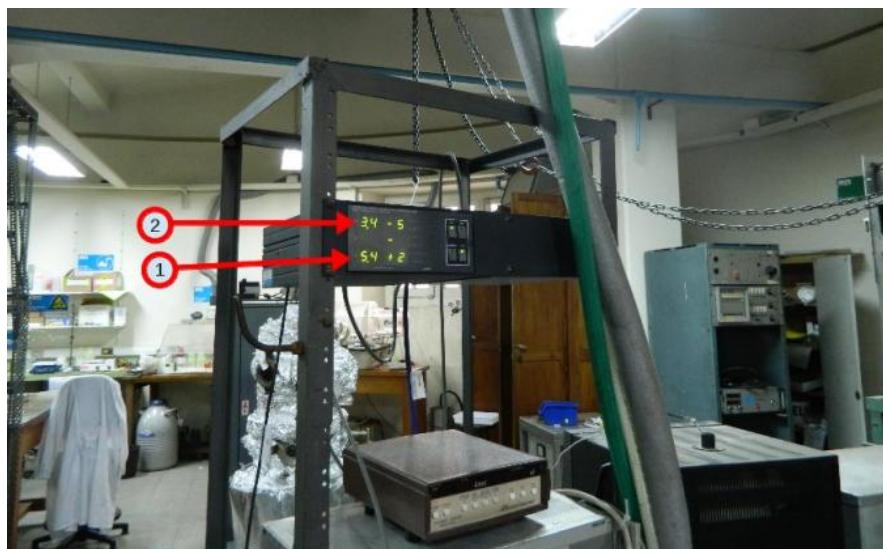
2 Fuente Poder (simula Paneles Solares)
4 Registro consola del SUCHAI
6 Bombas cámara de vacío

Figura 5.2: Prueba en la cámara de vacío (1)



1 Cámara para registro pruebas 2 Registro escrito de la prueba

Figura 5.3: Prueba en la cámara de vacío (2)



1 Medidor vacío [mBar] 2 Medidor alto vacío [mBar]

Figura 5.4: Prueba en la cámara de vacío (3)

5.1.3. Resultados

El detalle de cada una de las verificaciones y las pruebas es un conjunto de archivos asociados todos ellos a través de la RTVM, se ubican bajo la carpeta raíz "Pruebas Oficiales" y se puede obtener según indica el apéndice C. Ya que no tiene sentido reescribirlos en este

Función a cumplir	Requerimientos asociados	Resultado
Silencio radial y despliegue de antenas	3, 4 y 5	Satisfechos, se comprobó en la prueba operacional, ver momento del despliegue en figura 5.5
Actualización variables de estado	12	Satisfecho, se comprobó en prueba de laboratorio, ver figura 5.6
Control de Paylaods	20	Satisfecho, se comprobó en la prueba operacional, ver figura 5.7
TM bajo demanda	22	Satisfecho, se comprobó en las pruebas de laboratorio y operacional, ver figura 5.8
Recepción y decodificación de TM	23, 24 y 25	Satisfecho, se comprobó en prueba de laboratorio, ver figura 5.9

Tabla 5.1: Resumen de requisitos imprescindibles

capítulo, se detallan en la tabla 5.1 aquellos resultados más representativos.

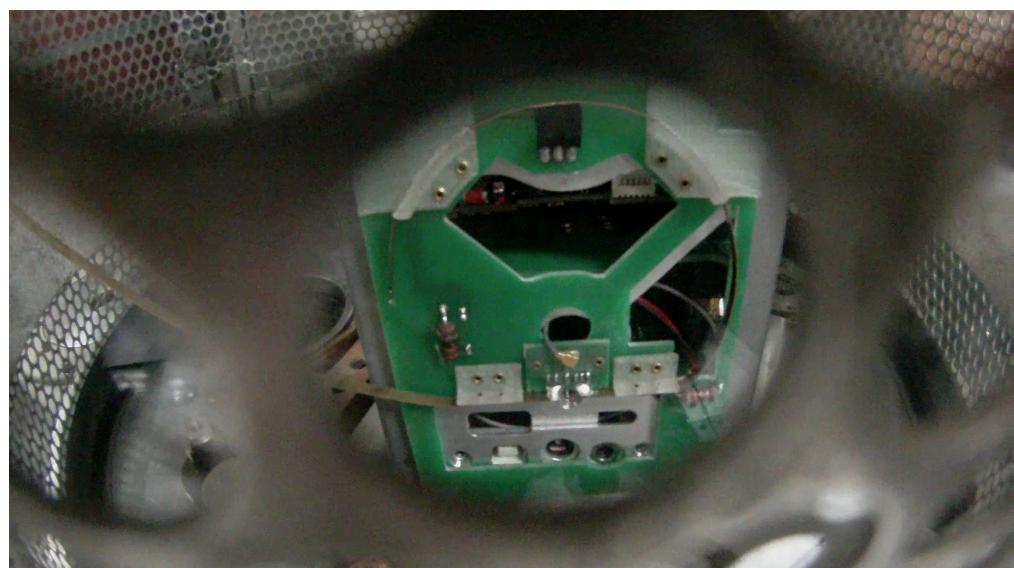


Figura 5.5: Momento del despliegue de antenas

```

===== WELCOME TO THE SUCHAI CONSOLE - PRESS ANY KEY TO START =====
>>>[Communications] Started^M
eps_bat0_voltage= 248^M
eps_bat0_current= 87^M
eps_bus5V_current= 1023^M
[Dispatcher] Orig: 0x1104 | Cmd: 0x5000 | Param: 0^M
eps_bus3V_current= 1016^M
eps_bus_battery_current= 1023^M
eps_bat0_temp= 586^M
eps_panel_pwr= 0^M
eps_status= -16384^M
eps_soc= 4^M
eps_socss= 4^M
eps_state_flag= 0^M
-----
eps_bat0_voltage= 248^M
eps_bat0_current= 87^M
eps_bus5V_current= 1023^M
[Dispatcher] Orig: 0x1104 | Cmd: 0x5000 | Param: 0^M
eps_bus3V_current= 1017^M
eps_bus_battery_current= 1023^M
eps_bat0_temp= 586^M
eps_panel_pwr= 0^M
eps_status= -16384^M
eps_soc= 4^M
eps_socss= 4^M
eps_state_flag= 0^M
-----
eps_bat0_voltage= 250^M
eps_bat0_current= 87^M
eps_bus5V_current= 1023^M
[Dispatcher] Orig: 0x1104 | Cmd: 0x5000 | Param: 0^M
eps_bus3V_current= 1016^M
eps_bus_battery_current= 1023^M
eps_bat0_temp= 586^M
eps_panel_pwr= 0^M
eps_status= -16384^M

```

1 Variables de estado de la EPS
2 Actualización de las variables

Figura 5.6: Actualización de variables de estado

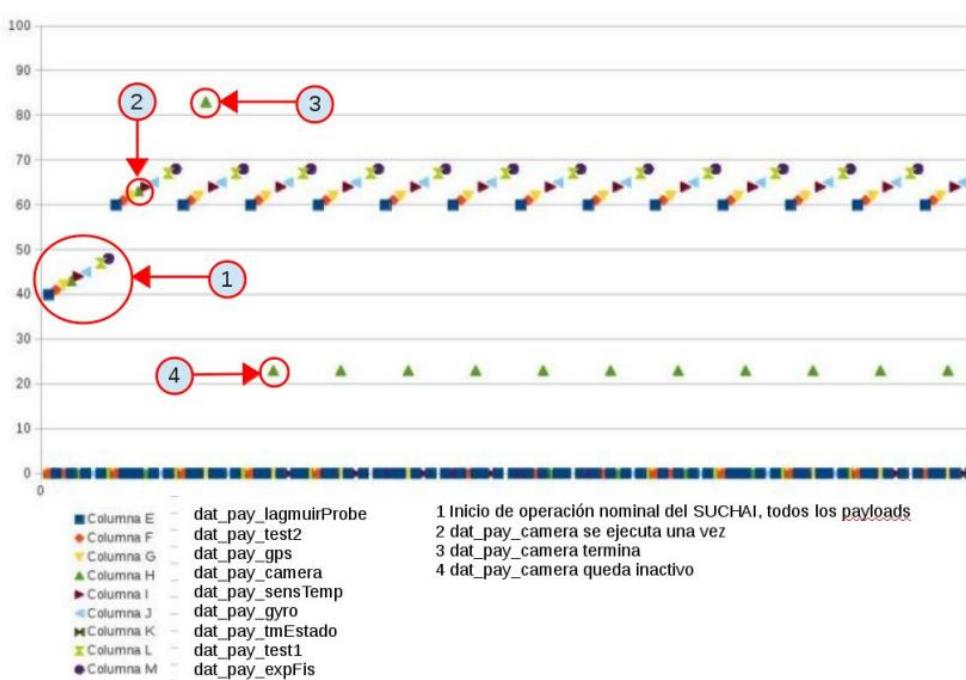


Figura 5.7: Ejecución de Payloads

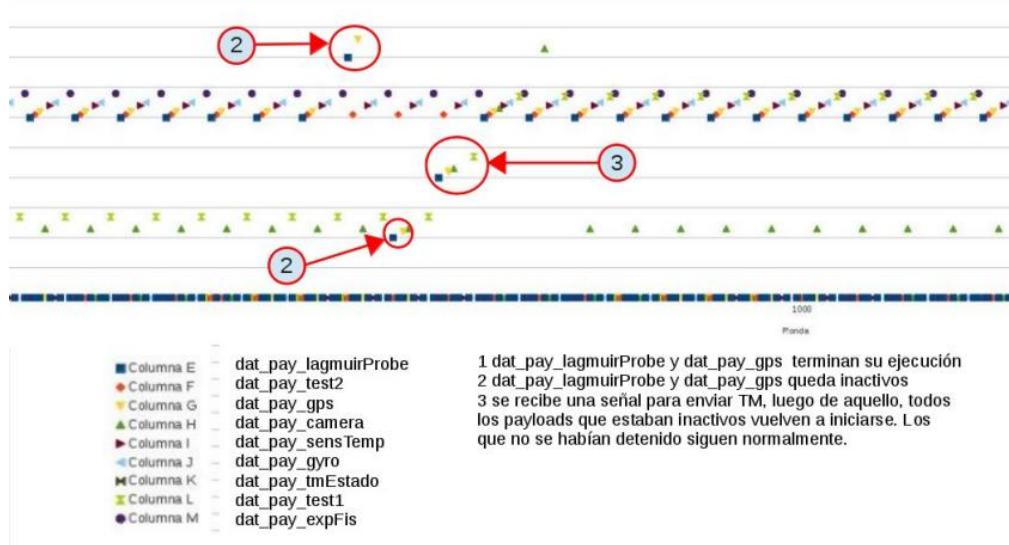


Figura 5.8: Influencia de la TM bajo demanda sobre la ejecución de payloads

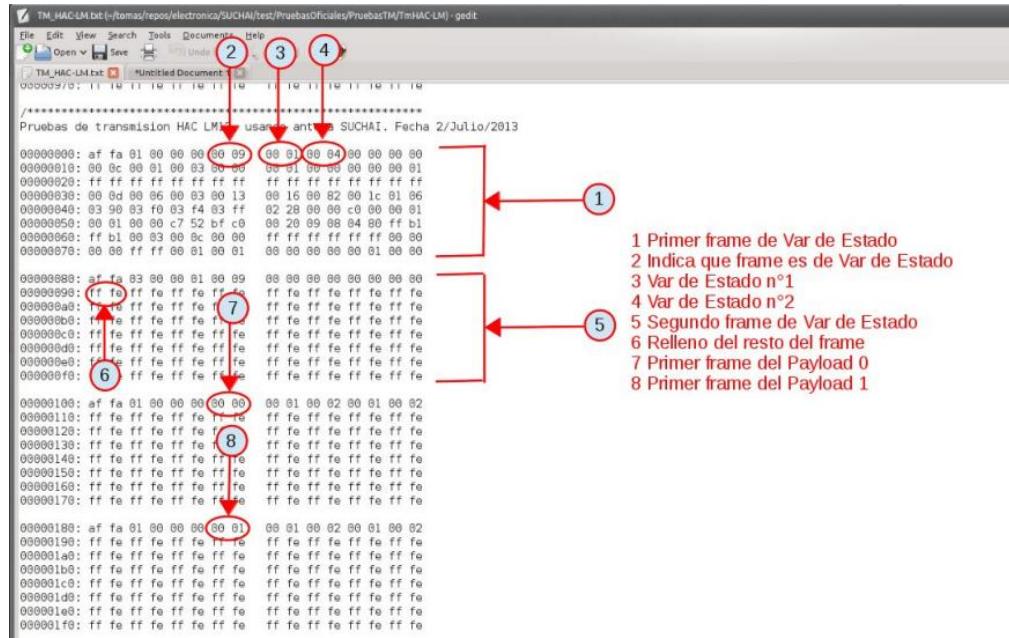


Figura 5.9: Decodificación (parcial) de TM

5.1.4. Resumen

De los 27 requerimientos originales, dos de ellos (26 y 27) fueron declarados **"No-activos"**, debido a que por problemas con el Transmisor de Allspace, no ha sido posible recibir Telecomandos en el satélite. De los 25 restantes, dos han sido declarado como **"Insatisfechos"**, seis como **"Pendientes"** y finalmente 17 han sido declarados como **"Satisfechos"**.

Dos de los declarados como Pendientes solo pueden ser verificados después de las pruebas de certificación internacional que aun no se realizan, los otros cuatro no han sido revisados

por tiempo, y por ser requerimientos no-operacionales, que debiesen ser sometidos a revisión por terceros dado su grado de subjetividad (modularidad, escalabilidad, etc). De los dos declarados como Insatisfechos, uno lo es debido limitaciones de espacio para diseñar una segunda placa de C&DH, lo que forzó en dejar hardware de ese subsistema en la placa de payloads, el segundo requisito insatisfecho es el de variables de estado en el contenido del Beacon, eso es netamente debido a la mala operación del Transmisor de Allspace, que no es capaz ni de recibir Telecomandos, ni de configurar adecuadamente los contenidos a transmitir.

De los 25 requerimientos Activos, 13 de ellos equivalentes al 52%, fueron declarados "**Imprescindibles**", de los cuales todos han sido declarados como Satisfechos, salvo los dos que, como ya se mencionó, solo son decidibles luego de las pruebas internacionales. Se puede entonces afirmar con seguridad que reemplazando el subsistema de comunicaciones, se tendría un requisito mas como Satisficho, y podrían reactivarse dos más, con lo que el satélite aumentaría drásticamente sus prestaciones.

Capítulo 6

Conclusión

Lo realizado en esta memoria fue determinar para el Cubesat SUCHAI, los requerimientos necesarios e indispensables de implementar (funciones, estándares, operaciones, comportamientos, etc), luego detallar la arquitectura implementada y que prometía satisfacerlas, para finalmente comprobar si esto era efectivamente así. Por ello saber/explicar cómo se llegó a una determinada solución no es el objetivo de esta memoria, sino que dada una, determinar si cumple o no con lo establecido en el diseño de la misión. En este sentido, los hitos más importantes fueron:

1. Diseño: La especificación de la **misión** y las **restricciones**.
2. Diseño: La determinación de **requerimientos**.
3. Pruebas: Las realización de **pruebas parciales**.
4. Pruebas: Las realización de una **prueba operacional**.
5. Pruebas: Verificación de requerimientos y documentación de estas.

El elemento que cohesiona estos hitos es la mencionada RTVM del SUCHAI 5, que relaciona la etapa inicial con la de Pruebas. Es a través de esta matriz, que fácilmente se puede ver que todo requerimiento es comprobado por alguna prueba. Este proceso es el principal aporte de la memoria, el que fue detallado a través de ella y que se resume a continuación.

6.1. Proceso de Diseño seguido y recomendado

Existe un fuerte símil entre las etapas de Requerimientos, Diseño, Implementación, Verificación y Mantenimiento del Proceso de Diseño en Cascada con lo hecho en caso del SUCHAI, **aunque no con los mismo nombres**. Siendo el primero, tercero y cuarto abordados en esta memoria con los nombres de Etapa de Diseño, Arquitectura y Pruebas respectivamente

(ver capítulos 3, 4, 5). El segundo paso, llamado de Diseño en el Modelo en Cascada, se llevo a cabo en parte en la referida memoria de Carlos Gonzales [21], en ella se explica cómo y por que se llego a la arquitectura del SUCHAI. En la presente memoria se detalló el que se debía realizar (Requerimientos), que se realizó (Implementación) y cómo se comprobó (Verificación). La etapa de Mantenimiento quedaría supeditada a la de Operación del satélite, y solo existiría en el caso de que el satélite pueda ser reprogramado.

Queda como sugerencia para futuras misiones el realizar un proceso de diseño similar adoptando para ello los nombres del Modelo en Cascada para evitar confusiones como las ocurridas aquí. Al construir un nuevo satélite de la escala de un Cubesat se recomienda realizar una etapa de Requerimientos donde estos queden expresados en una RTVM, luego una etapa de Diseño en donde se propongan alternativas (no más de las que se tenga capacidad de analizar y comparar) y se elija una de ellas, llevarla a la etapa de Implementación y finalmente demostrar en la etapa de Verificación si cumplen o no con los requerimientos, usando para ello nuevamente la RTVM. Existen otras alternativas como los modelos SMAD [33] o de agencias espaciales como la ESA o la NASA, sin embargo dada la complejidad, administración y duración de estos se estima sería mejor contar con menos etapas pero mas explicitas y claras.

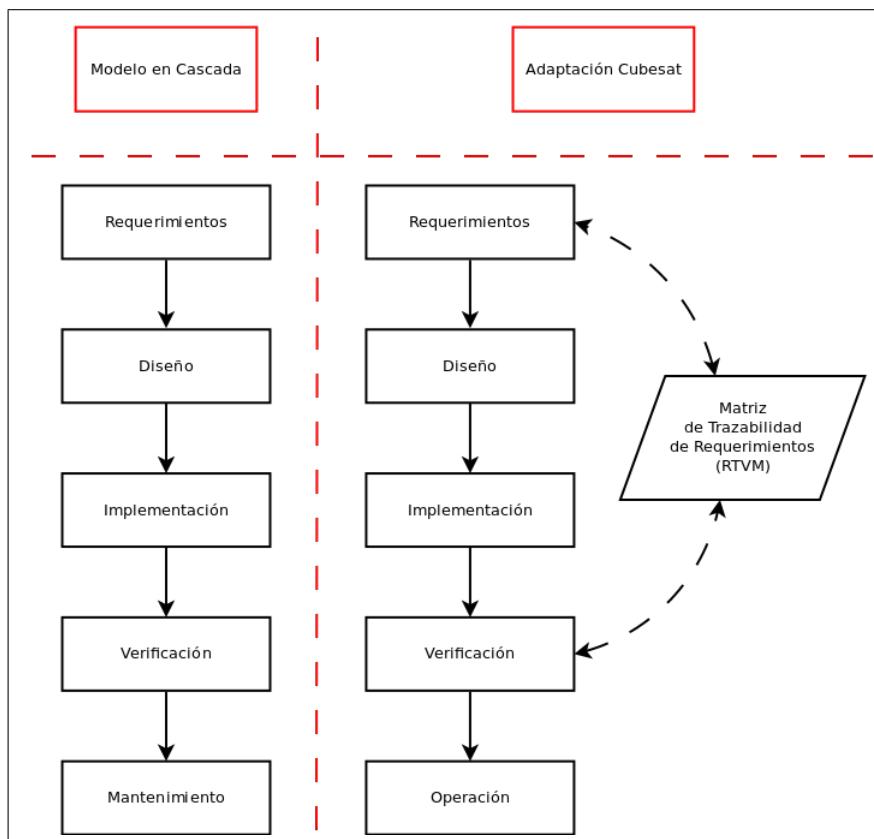


Figura 6.1: Resumen proceso de diseño recomendado

6.2. Análisis de lo realizado y trabajos futuros

Por las razones dadas a lo largo de esta memoria se estimó que la elección del software de control fue un buen acierto del equipo del SUCHAI, pues se aparta de la opción tradicional vista en otros desarrollos Cubesat en otras universidades del mundo, donde se emplean maquinas de estado para cada una de las acciones de un satélite como inicialización, tomar datos de payloads, comunicarse o entrar en modo ahorro de energía. En el SUCHAI en cambio existe una arquitectura más homogénea en su funcionamiento que la hace mas predecible, pues todas las acciones son tomadas de la misma forma, independientemente si se trate de comunicación con la tierra o ejecutar payloads.

6.2.1. Transceiver

A pesar de que el SUCHAI, con las características detalladas en esta memoria, cumple correctamente con el funcionamiento esperado tal y como se vio en las pruebas operacionales, hubo aspectos que degradaron su desempeño y lo hicieron estar por debajo de su potencial. Estos problemas se resumen básicamente en la imposibilidad de contar con Telecomandos con el Transceiver adquirido a Allspace, lo que limita la flexibilidad del satélite y sus acciones. Esto provoco un retraso considerable en el desarrollo del proyecto, y obligo al equipo del SUCHAI a establecer un método subóptimo, por medio de RSSI, para lograr el requisito de Telemetría On Demand.

Como trabajos futuros se recomienda fuertemente el reemplazo del Transceiver de Allspace por algunas más robustas y que permita contar Telecomandos hacia el Cubesat.

6.2.2. Kit Pumpkins

El kit Pumpkins demostró ser un excelente punto de partida para desarrollos Cubesat, y debido a que su costo es razonable se recomienda usarlo como base **mecánica** para un SUCHAI-2, esto es usar el esqueleto de aluminio, rieles, y switches, para así reducir las áreas de trabajo y tener mayor certeza de aprobar la certificación CDS en cuanto a dimensiones.

Sin embargo las placa MB y PPM son definitivamente reducibles a una sola más pequeña y más específica para el bus SUCHAI, se recomienda muy fuertemente desarrollar una placa propia en torno al OBC del SUCHAI (PIC24), incluyendo todo el hardware del C&DH, (RTC, memorias no volátiles -EEPROM-, y dispositivos de almacenamiento de datos -Tarjeta SD-, Switches para despliegue de antenas, etc..). De esta manera se lograría tener una relación unívoca entre tarjeta física y subsistemas del satélite.

6.2.3. Bus de datos

Una debilidad de muchos Cubesat y lamentablemente del SUCHAI también es que utilizan más de un protocolo para el intercambio de datos a bajo nivel (capa de Enlace). En el mundo de los PC y de los satélites comerciales hace largo tiempo se ha convergido a una arquitectura de bus de datos no jerárquica y uniforme. En este sentido se recomienda hacer lo mismo y homogeneizar los protocolos a uno solo, siendo los recomendados CAN o bien I2C, en ambos casos es muy probable que haya que implementar un sistema de conversión desde y hacia otros protocolos con CAN o I2C respectivamente. Se recomienda también implementar dos buses, uno del sistema y otro para los payloads.

De la mano con la unificación de un protocolo de transferencia de datos y comandos está la integración de el protocolo CSP (de capa de Red y Transporte). De esta forma cada subsistema pasa a ser un host, y el Transceiver pasa a ser un router, y desde la estación terrena (que también seria un host) se puede establecer comunicación con cualquier nodo dentro del satélite y no únicamente con el OBC como muchas veces sucede.

6.2.4. Sistema de Archivos FAT

Actualmente el almacenamiento de datos para Payloads se realiza en una tarjeta SD, actualmente se accede a los datos por direccionamiento físico. Esto podría mejorar considerablemente si se implementa un sistema FAT que permitiese leer y escribir la información del satélite desde un computador común y corriente. Esto haría más estándar y fácil la programación y debug del satélite.

6.2.5. OBC

Todo el software SUCHAI pudo ser desarrollado y verificado en un microcontrolador PIC24, todo este software pudo ser grabado y luego ejecutado, sin embargo se detectó escasez de memoria RAM y se recomendaría escoger un modelo de mayor capacidad en ese sentido. La memoria de programa demostró ser suficiente en todo momento.

6.3. Resumen

El desarrollo de un Cubesat ha demostrado ser una tarea que requiere de un grupo humano especializado y activo, es un área que precisa de la integración de varias ramas de la ingeniería. Energía, comunicaciones a distancia y procesamiento es una triada poderosa pero difícil de integrar. El SUCHAI demostró su capacidad en operaciones de vacío muy similares a las que tendrá en órbita y fue capaz de funcionar correctamente, hay tareas que aun quedan pendientes, pero un buen punto inicial ha sido logrado, se espera que su lanzamiento y

operación así lo siga confirmando.

Bibliografía

- [1] Wikipedia. Definición de un proceso de diseño. http://en.wikipedia.org/wiki/Engineering_design_process. Accesada en Julio del 2013. Basado en: Ertas, A. and Jones, J. (1996). The Engineering Design Process. 2nd ed. New York, N.Y., John Wiley and Sons, Inc.
- [2] Wikipedia. Definición de diseño en cascada (Waterfall Model). http://en.wikipedia.org/wiki/Waterfall_model. Accesada en Julio del 2013.
- [3] Wikipedia. Definición de Agile Programming. http://en.wikipedia.org/wiki/Agile_software_development. Accesada en Julio del 2013.
- [4] Wikipedia. Definicion de Spiral Model. http://en.wikipedia.org/wiki/Spiral_model. Accesada en Julio del 2013.
- [5] Wikipedia. Definición de un proceso de diseño (en Ingenieria). http://en.wikipedia.org/wiki/Engineering_design_process. Accesada en Julio del 2013.
- [6] Ramesh, B. and Jarke, M. Software Engineering, IEEE Transactions on. Toward reference models for requirements traceability. 2001, volume 27, number 1. doi=10.1109/32.895989. ISSN 0098-5589.
- [7] Wikipedia. Definición de Sistemas integrados (Embedded Systems). http://en.wikipedia.org/wiki/Embedded_system. Accesada en Julio del 2013.
- [8] Leonardo M. Reyneri, Claudio Sansoe, Claudio Passerone, Stefano Speretta, Maurizio Tranchero, Marco Borri, and Dante Del Corso. Design Solutions for Modular Satellite Architectures. 2010. Aerospace Technologies Advancements, Thawar T. Arif (Ed.), ISBN: 978-953-7619-96-1, InTech, DOI: 10.5772/6933.<http://www.intechopen.com/books/aerospace-technologies-advancements/design-solutions-for-modular-satellite-architectures>.
- [9] Mayo, J. S., Mann, H., Witt, F. J., Peck, D. S., Gummel, H. K. and Brown, W. L. The Command System Malfunction of the Telstar Satellite. Bell System Technical Journal. 1963. Bell System Technical Journal, 42: 1631-1657. doi: 10.1002/j.1538-7305.1963.tb04044.x
- [10] Anthony D Galvan. Decommissioning A Geosynchronous Satellite. June 2011. Thesis

for the Degree Bachelor of Science in Aerospace Engineering.

- [11] Thomas A. Henzinger and Joseph Sifakis. The Embedded Systems Design Challenge. Proceedings of the 14th International Symposium on Formal Methods (FM), Lecture Notes in Computer Science. 2006. Pages 1–15. Publisher, Springer.
- [12] Christopher J. McNutt, Robert Vick, Harry Whiting, James Lyke. Modular Nanosatellites - Plug-and-Play (PnP) CubeSat. 2009. 7th Responsive Space Conference. April 27-30, 2009. Los Angeles, CA.
- [13] Louise Dennis and Michael Fisher and Alexei Lisitsa and Nicholas Lincoln and Sandor Veres. Satellite Control Using Rational Agent Programming. IEEE Intelligent Systems. Volume 25, number 3, issn 1541-1672. 2010. <http://doi.ieeecomputersociety.org/10.1109/MIS.2010.88>. IEEE Computer Society. Los Alamitos, CA, USA.
- [14] Jaime Esper. Modular, Adaptive, Reconfigurable Systems: Technology for Sustainable, Reliable, Effective, and Affordable Space Exploration. 2005. AIP Conf. Proc. 746, pp. 1033-1043. <http://dx.doi.org/10.1063/1.1867227>.
- [15] Wikipedia. Definición de Quality of Service (QoS). http://en.wikipedia.org/wiki/Quality_of_service. Accesada en Julio del 2013
- [16] Wikipedia. Definición de Requerimientos (en el contexto de diseño). https://en.wikipedia.org/wiki/Requirements_analysis. Accesada en Julio del 2013.
- [17] Wikipedia. Definición de Best-Effort. http://en.wikipedia.org/wiki/Best-effort_delivery. Accesada en Julio del 2013.
- [18] DIGITAL CONNECTIONS COUNCIL OF THE COMMITTEE FOR ECONOMIC DEVELOPMENT. OPEN STANDARDS, OPEN SOURCE, AND OPEN INNOVATION: Harnessing the Benefits of Openness. 2006. Reporte online, <http://www.ced.org/reports/single/open-standards-open-source-and-open-innovation>. Vista en Agosto del 2013)",
- [19] Wikipedia. Definición del Modelo OSI http://en.wikipedia.org/wiki/OSI_model. Accesada en Julio del 2013.
- [20] Ivan M. Bland. Receive Sensitivity of the PolySat Communication System. March 2010. Vista en Agosto del 2013. <http://digitalcommons.calpoly.edu/theses/265/>.
- [21] Memoria de Carlos Gonzales: Diseño e implementación del software de vuelo para un nano-satélite tipo Cubesat. 2013. Tesis de grado.
- [22] Cubesat project. Cubesat Design Specification (CDS). 2013. Vista en Agosto del 2013. <http://www.cubesat.org/index.php/documents/developers>
- [23] Wikipedia. Definición de Cubesats. 2013. Vista en Agosto del 2013. <http://en.wikipedia.org/wiki/CubeSat>.

- [24] Marcos Diaz. Satellite of University of CHlie for Aerospace Invertigation (SUCHAI). 2010. Documento de formalización de la misión SUCHAI.
- [25] Hankuk Aviation University. HAUSAT Cubesat. 2004. Vista en Agosto del 2013. http://space.skyrocket.de/doc_sdat/hausat-1.htm.
- [26] Institute of Space Technology (IST). ICUBE Cubesat. Vista en Agosto del 2013. <http://www.ist.edu.pk/about/breakthrough-research-projects/icube>.
- [27] Kentucky University. KYSAT Cubesat. Vista en Agosto del 2013. <http://ssl.engineering.uky.edu/missions/orbital/kysat1/about-kysat-1/>.
- [28] Aalborg University. AAUSAT3. Vista en Agosto del 2013. <http://www.space.aau.dk/aausat3/>.
- [29] Cal Poly University. PolySat. Vista en Agosto del 2013. <http://polysat.calpoly.edu/>.
- [30] Architecture of a Small Low-Cost Satellite D. Del Corso, C. Passerone, L. M. Reyneri, C. Sanso e, M. Borri, S. Speretta, M. Tranchero Politecnico di Torino, Dipartimento di Elettronica corso Duca degli Abruzzi, 24, 10129 Torino, Italy. 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007) 0-7695-2978-X/07. 2007
- [31] A survey and assessment of the capabilities of Cubesats for Earth observation. Acta Astronautica. 2012 .issn 0094-5765. <http://www.sciencedirect.com/science/article/pii/S0094576511003742>. Daniel Selva and David Krejci.
- [32] AAUSAT project. Definición de Cubesat Space Protocol (CSP). Wikipedia, Vista en Agosto del 2013. http://en.wikipedia.org/wiki/Cubesat_Space_Protocol.
- [33] Space Mission Analysis and Design. Wertz, J.R. and Larson, W.J. isbn 9780792359012. lccn 2005283477. series Space Technology Library. <http://books.google.cl/books?id=veyGEAKFbiYC>. 1999. Springer.
- [34] European Space Agency. Space Project Management. Project planning and implementation. ECSS-M-30B Draft 14. 6 July 2007.
- [35] NASA. Project Life Cycle Phases. Vista en Agosto del 2013. <http://www2.jpl.nasa.gov/basics/bsf7-1.php>.
- [36] Pumpkins. Cubesat Kit. Vista en Agosto del 2013. <http://www.cubesatkit.com/>.
- [37] Pumpkins. Cubesat Kit. Vista en Agosto del 2013. <http://www.cubesatkit.com/content/space.html>.
- [38] Wikipedia. Colision entre satelites. Vista en Agosto del 2013. http://en.wikipedia.org/wiki/2009_satellite_collision.

- [39] Wikipedia. Fallo de la mision FobosGrunt. Vista en Agosto del 2013. <http://www.space.com/162-programming-error-botched-russian-rocket-launch-report.html>.
- [40] Celstrak. Norad, orbital elements. Vista en Agosto del 2013. <http://www.celestrak.com/NORAD/elements/>.
- [41] Anatoly Zak. Russian Space Web (pagina de divulgación). Vista en Agosto del 2013. <http://www.russianspaceweb.com>.
- [42] Satellite Industry Agency. Futron Corporation. State of satellite industry report. 2012. Vista en Agosto del 2013. <http://www.sia.org>.
- [43] Globalcomsatphone. Satellite Costs. Vista en Agosto del 2013. <http://www.globalcomsatphone.com/hughesnet/satellite/costs.html>.
- [44] European Space Agency. International Space Station velocity information. Vista en Agosto del 2013. http://www.esa.int/Our_Activities/Human_Spaceflight/International_Space_Station/ISS_International_Space_Station.
- [45] ESA. Alpha Bus, Europe's solution for the high-power satcom market. Bulletin 142. May 2010.
- [46] Fuerza Aerea de Chile. Fasat Charlie. Vista en Agosto del 2013. <http://www.saf.cl/proyectos.php?cod=34>.

Apéndices

Apéndice A

Glosario

Glosario		
Sigla/Palabra	Significado	Descripción
C&DH	Command and Data Handler	Subsistema de un satélite (ver 2.2 para más detalles)
ADCS	Attitude determination and control system	Subsistema de un satélite (ver 2.2 para más detalles)
EPS	Energy Power System	Placa y subsistema de Energía, producto de la empresa Clyde Space. Contiene una batería y reguladores de voltaje, entre otros.
TRX	Transceiver	Subsistema transmisor y receptor, recibe y envía Telecomandos y Telemetría respectivamente.
TC	Telecomandos	Comando enviado desde la estación terrena hacia el satélite
TM	Telemetría	Comando enviado desde el satélite hacia la estación terrena
RTVM	Requirement Traceability and Verification Matrix	Matriz de Trazabilidad de Requerimientos
SMAD	Space Mission Analysis and Design	Libro de referencia en procesos de diseño de satélites, hace referencia también al proceso de diseño que propone.
CDS	Cubesat Design Specifications	Documento que detalla el estándar oficial Cubesat
PPM	Pluggable Processor Module	Placa montable sobre la MB, contiene al OBC
MB	Mother Board	Placa parte del C&DH del Kit Pumpkins, sobre ella se sitúa el PPM, contiene además componentes como el RTC, memSD entre otros

Glosario (continuación)		
Sigla/Palabra	Significado	Descripción
RTC	Real Time Clock	Componente alimentado independientemente con una pila, esta encargado de mantener una referencia de tiempo absoluta al satélite
memSD	Memoria Secure Digital	Memoria no volátil de almacenamiento masivo, generalmente de 2GB
memEEPROM	Memoria Electrically Erasable Permanent Read Only Memory	Memoria no volátil, de rápido acceso pero limitada capacidad, de un par de kB
PC/104		Bus de datos utilizado por el Kit Pumpkins
OBC	On Board Computer	Computador central a bordo de un satélite, casi siempre es parte del C&DH
PIC24		Abreviación del microprocesador PIC24FJ256GA110 de 16 bit de la compañía Microchip
Mem	Memoria	Abreviación de la palabra Memoria
Com	Comunicaciones	Abreviación de la palabra Comunicaciones
Hw	Hardware	Abreviación de la palabra Hardware
Sw	Software	Abreviación de la palabra Software
Mec	Mecánico	Abreviación de la palabra Mecánico
CDS	Cubesat Design Specifications	Documento que detalla el estándar oficial Cubesat
I2C		Protocolo de comunicación usado en circuitos integrados
SPI		Protocolo de comunicación usado en circuitos integrados
UART		Protocolo de comunicación usado en circuitos integrados
CDS	Cubesat Design Specifications	Documento que detalla el estándar oficial Cubesat
OSI	Open System Interconnections	Modelo teórico para comunicación entre sistemas
RSSI	Received Signal Strength Indicator	Indicador de la potencia de la señal recibida por un TRX
FSM	Finite State Machine	Maquina de Estados Finitos o Autómata Finito Determinista
FAT	File Allocation Table	Sistema de archivos para medios de almacenamiento masivo
CSP	Cubesat Space Protocol	Protocolo que emula la entrega de paquetes entre hosts en redes TCP/IP
SysReq	System Requirements	En el SUCHAI, nivel de energía del satélite
Payload	Carga Pagada (traducción)	Experimento o equipo anexo al bus satelital destinado a sensar o registrar un fenómeno de interés

Glosario (continuación)		
Sigla/Palabra	Significado	Descripción
Memoria RAM	Random Acces Memory	Memoria de trabajo para el sistema operativo, los programas y la mayoría del software. Es allí donde se cargan todas las instrucciones que ejecutan el procesador y otras unidades de cómputo.
GPS	Global Positioning System	Sistema de posicionamiento global
TDM	Time Division Multiplexing	Técnica de división temporal de utilización de un recurso (señal) para que más de un usuario haga usufructo de él
FreeRTOS		Sistema Operativo libre, de tipo "preemptive scheduling" para sistemas embebidos
QoS	Quality of Service	Concepto relacionado a varios aspectos de telefonía y redes de computación que permiten el transporte de datos bajo requerimientos especiales.
NASA	National Aeronautical and Space Administration	Agencia espacial estadounidense
ESA	Europoean Space Agency	Agencia espacial europea

Apéndice B

Imágenes

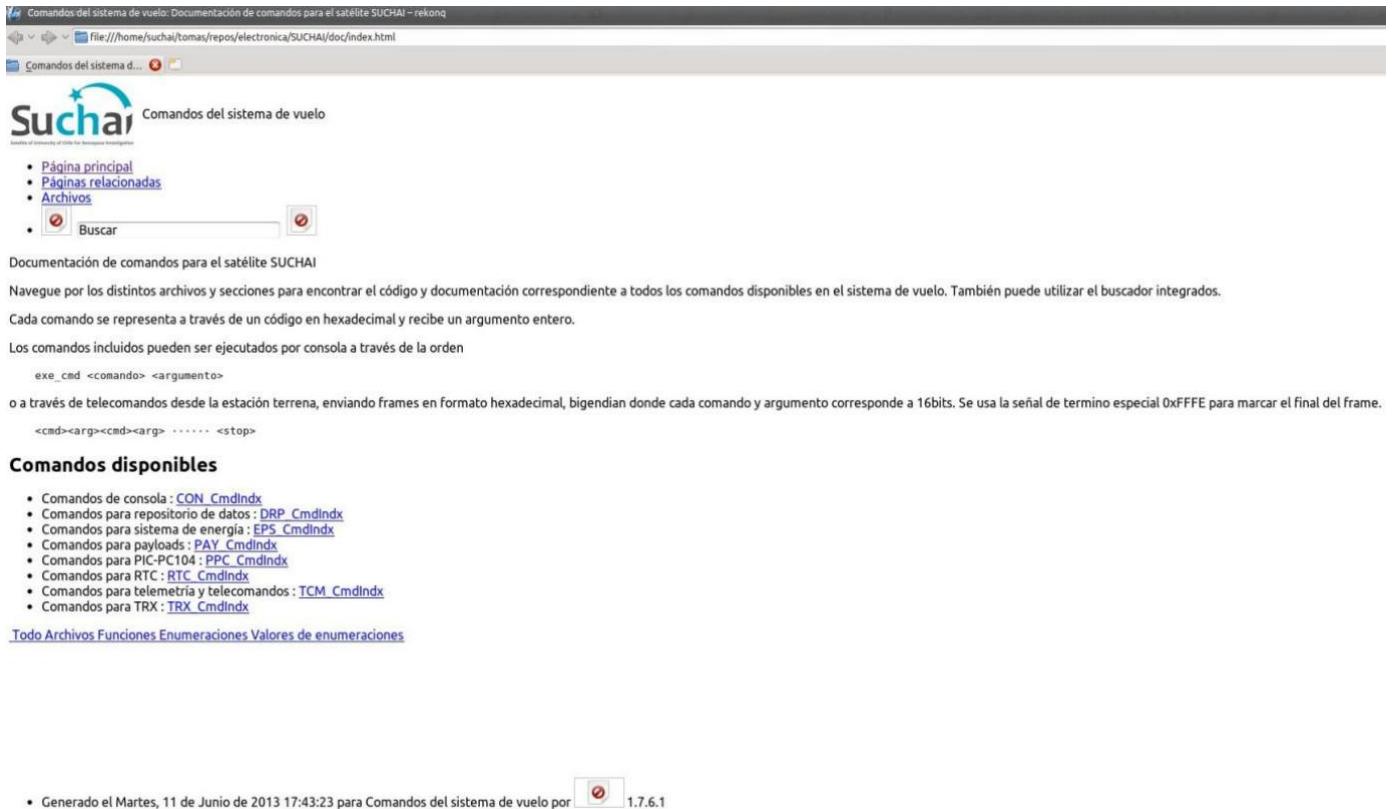


Figura B.1: Documentación en html usando Doxygen del Código fuente el software SUCHAI

Apéndice C

Pruebas del SUCHAI

Todos los documentos asociados a las pruebas del SUCHAI se encuentran en el repositorio svn del servidor del SUCHAI. Se encuentran bajo la carpeta "**PruebasOficiales**". Esta incluye:

- Documento: "**EstandarOficialPruebasVerif.docx**" Donde se explican los procedimientos para hacer consistente y valida de la verificación del SUCHAI
- Documento: "**Programa de pruebas del SUCHAI.docx**" Donde se detallan las características del SUCHAI v1.0 y de las pruebas de laboratorio y de operación realizadas.
- Documento: "**Carta Gantt.xlsx**" Planificación realizada para la prueba operacional.
- Documento: "**RTVM del SUCHAI.xlsx**" Es la RTVM oficial del proyecto SUCHAI
- Documentos: "**verifReq1.docx - verifReq27.docx**" Donde se detalla como se verificó cada requerimiento, la prueba en que si hizo y su resultado
- Archivo comprimido: "**PowwerBudget_v10.rar**" Power Budget del SUCHAI
- Subcarpetas: "**PruebaCamaraVacio02Ago2013**" Carpeta con todos los detalles de la prueba operacional
- Subcarpetas: "**CertificacionInternacional**" Carpeta con todos los detalles de esta prueba parcial de laboratorio
- Subcarpetas: "**PruebaBeacon**" Carpeta con todos los detalles de esta prueba parcial de laboratorio
- Subcarpetas: "**PruebaCargaEPS**" Carpeta con todos los detalles de esta prueba parcial de laboratorio
- Subcarpetas: "**PruebaDespAnt**" Carpeta con todos los detalles de esta prueba parcial de laboratorio

- Subcarpetas: "**PruebaEPS₁1Jul13**" Carpeta con todos los detalles de esta prueba parcial del laboratorio Subcarpetas:
- Subcarpetas: "**PruebasTM**" Carpeta con todos los detalles de esta prueba parcial de laboratorio
- Subcarpetas: "**PruebasVarias**" Carpeta con todos los detalles de esta prueba parcial de laboratorio

A cada miembro de la comisión se le hace entrega de un CD o Usb con la mencionada carpeta, de no ser así o de requerir otra por favor contactar via mail a Tomas Opazo: tomas.opazo.t@gmail.com u otro miembro del proyecto SUCHAI con acceso al repositorio.