Carrera
Ingeniería Civil Aeroespacial
Universidad de Concepción

Desarrollo de un sistema de apoyo a la educación en ingeniería de sistemas utilizando satélites para aulas

Diego Alejandro González González

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título profesional de Ingeniero Civil Aeroespacial

Profesor guía:

Dr. Alejandro López Telgie

12 de Enero, 2026

Concepción, Chile

“In the time of your life, live—so that in that good time there shall be no ugliness or death for yourself or for any life your life touches (...)”

- *William Saroyan, The time of your life*

Esta memoria de título está dedicada a todas las vidas que han tocado la mía, y que han contribuido a quien soy.

Las palabras me fallan, y las hojas me faltan, para poder agradecer a toda la gente que se lo merece y que me gustaría incluir, en ese orden. Me veo empujado a dos extremos irreconciliables: la necesidad de ser completo y la de ser conciso. Sospecho que no hay forma rigurosa de satisfacer ambas.

Quiero agradecer en primer lugar a mi familia, y en particular:

A mi madre, Jimena, por ser un apoyo y compañía constante durante los 7 años que ha tomado el desarrollo de esta carrera. A mi hermano, Nadia Vicente, por ser siempre una fuente de inspiración y orgullo que me ha mantenido siempre al frente del juego. A mi padre, Fernando, por ser un apoyo constante e incondicional, además de siempre velar por mi bienestar a la distancia. A mis abuelos, tíos y primos; Carlos, Flora, Fernando, Adriana, Pablo, Pamela, Eleonor, Claudia, Renato, Martin, Amanda, Oscar, y a una verdadera multitud a quienes no alcanzaré a nombrar, pero cuyo cariño y apoyo ha permeado mi ser desde el inicio.

Quiero también agradecer a la familia encontrada: Tomás, Caecilia, Tatiana, Nicolás y Nicolás 2, por estar siempre conmigo, por entender, y por mucho, mucho más. Gracias por tanto y perdón por tan poco.

A mis compañeros de memoria, Nicolás 3 y 4, Bertila, Iván, Amanda y Lucas, por hacer de esta etapa mucho más llevadera gracias al compañerismo, amistad y dedicación de todos. Y a todos mis compañeros y compañeras, amigos y conocidos, de cuyo apoyo, buen ánimo y cuya disposición desde el día uno de mi carrera ha sido nada menos que de primera categoría.

A los profesores y profesoras que me han formado:

Mi profesor guía, Alejandro, por ser al mismo tiempo un motor por la excelencia y un abogado por la eficiencia. Alguien que me ha demostrado que entre soñar y crear no hay nada salvo praxis.

Los profesores que me han acompañado y guiado estos años: Frank Tinapp, Fabiola Maureira, Bernardo Hernández, Pablo Cornejo, y a todos los profesores que han aportado en la educación recibida año a año.

A mis amigos que han seguido conmigo durante todos estos años: Gabriel, Camila, Jesús, Ale y a muchos más que no alcanzo a nombrar.

A todas las brillantes y gentiles almas de Asclepios: Léonie, Vero, Matias, Elena, Emma, Loïc, Jessica, Simon, Ilaria, y las docenas de fantásticas personas que fueron parte de esta experiencia.

Y a todas las almas que han tocado la mía en la infinidad de circunstancias que forman una vida, y a todos quienes en algún momento de necesidad hayan extendido una mano. Alguien recién salvado de ser ahogado quizás no dé las gracias en el momento, pero la gratitud será una constante el resto de su vida.

Resumen

Esta memoria de título presenta las fases de diseño conceptual y en detalle, implementación, integración y validación de un kit educativo tipo CubeSat orientado a apoyar la enseñanza de ingeniería de sistemas espaciales, con énfasis en el ciclo AIT/AIV, la integración de subsistemas y la gestión de complejidad bajo restricciones reales. El proyecto se enmarca en el crecimiento del entorno espacial chileno y en la necesidad de formar capital humano capaz de enfrentar sistemas interdisciplinarios desde etapas relativamente tempranas de formación profesional.

El trabajo adopta un enfoque iterativo y pragmático, donde el sistema es considerado más como una plataforma educativa que como un satélite funcional. El diseño integra subsistemas de potencia (EPS), computador a bordo (OBC), sensores, comunicaciones y estructura, priorizando la modularidad, la seguridad eléctrica e informática y la trazabilidad del proceso de integración. En el software de vuelo se implementa una arquitectura minimalista basada en CircuitPython, con manejo explícito de excepciones, modos de debug y desarrollo, y tolerancia a fallas de sensores, permitiendo operar el sistema aun en configuraciones parciales o incompletas e incluso resistir fallas en el circuito. Se diseñan e implementan ciclos básicos de Command and Data Handling y Communications, utilizando WLAN y MQTT como interfaces mínimas de telemetría y comando, dejando planteada una arquitectura robusta para futuras iteraciones.

La integración de sensores contempla buses I2C, OneWire y señales analógicas, incluyendo sensores de potencia, temperatura y luminosidad. Si bien no se logra integrar completamente un payload óptico ni una IMU debido a limitaciones técnicas, de librerías y a incidentes durante la integración, estas falencias se analizan explícitamente y se utilizan para modelar decisiones de diseño más seguras en futuras implementaciones. El proyecto enfatiza la importancia de identificar puntos de detención responsables cuando la implementación completa introduce riesgos injustificados.

Como validación, se diseña y ejecuta una experiencia educativa piloto en la forma de un laboratorio práctico grupal, centrada en aceptación, integración y verificación de subsistemas, permitiendo a los estudiantes interactuar directamente con el kit, las guías de apoyo y la infraestructura de comunicaciones. La experiencia demuestra la utilidad del sistema como herramienta de apoyo a la educación, revelando además oportunidades ergonómicas de mejora en materiales de apoyo, tiempos y organización del trabajo colaborativo.

Finalmente, se evalúa la factibilidad logística y económica de producir el sistema en lotes pequeños (5 a 10 kits), demostrando que es posible fabricar, preparar e integrar los kits en plazos inferiores a un mes y con costos unitarios acotados. Las conclusiones reconocen tanto los éxitos del proceso como las falencias reiteradas, pero destacando el enfoque en seguridad, la necesidad de equipos multidisciplinarios y el valor educativo de exponer a los estudiantes a sistemas reales, incompletos y propensos a falla.

Abstract

This thesis presents the conceptual and detailed design, implementation, integration, and validation phases of a CubeSat-type educational kit designed to support the teaching of space systems engineering, with an emphasis on the AIT/AIV cycle, subsystem integration, and complexity management under real-world constraints. The project is framed within the growth of the Chilean space sector and the need to develop human capital capable of addressing interdisciplinary systems from relatively early stages of professional training.

The work adopts an iterative and pragmatic approach, where the system is considered more as an educational platform than as a functional satellite. The design integrates power (EPS), on-board computer (OBC), sensor, communications, and structural subsystems, prioritizing modularity, electrical and IT security, and traceability of the integration process. The flight software implements a minimalist architecture based on CircuitPython, with explicit exception handling, debug and development modes, and sensor fault tolerance, allowing the system to operate even in partial or incomplete configurations and even withstand circuit failures. Basic Command and Data Handling and Communications cycles are designed and implemented, using WLAN and MQTT as the minimum telemetry and command interfaces, establishing a robust architecture for future iterations.

Sensor integration includes I2C and OneWire buses, as well as analog signals, including power, temperature, and light sensors. While full integration of an optical payload and an IMU is not achieved due to technical limitations, library constraints, and integration issues, these shortcomings are explicitly analyzed and used to inform safer design decisions in future implementations. The project emphasizes the importance of identifying responsible stopping points when full implementation introduces unjustified risks.

As a validation measure, a pilot educational experience is designed and implemented in the form of a group hands-on lab, focused on the acceptance, integration, and verification of subsystems, allowing students to interact directly with the kit, supporting guides, and communications infrastructure. Experience demonstrates the system's usefulness as an educational support tool, also revealing ergonomic opportunities for improvement in support materials, timing, and the organization of collaborative work.

Finally, the logistical and economic feasibility of producing the system in small batches (5 to 10 kits) is evaluated, demonstrating that it is possible to manufacture, prepare, and integrate the kits in less than a month and with limited unit costs. The conclusions acknowledge both the successes of the process and the recurring shortcomings, but highlight the focus on safety, the need for multidisciplinary teams, and the educational value of exposing students to real, incomplete, and failure-prone systems.

Tabla de contenidos

Glosario

AIT	*: Assembly, Integration, and Testing*
AIV	*: Assembly, Integration, and Verification*
BTB	*: Board to Board*
C4i	*: Centro para la Industria 4.0*
CDH	*: Command & Data Handling Subsystem*
CDIO	*: Conceive, Design, Implement, and Operate.*
CDS	*: CubeSat Design Specification*
COM	*: COMmunication Serial Port*
ConOps	*: Concept of Operations*
COTS	*: Commercial Off-The-Shelf*
CSRM	*: CubeSat System Reference Model*
CUBEEK	*: CUBEsat Educational Kit*
ECSS	*: European Cooperation for Space Standardisation*
EGS	*: Electric Ground Support*
EPS	*: Electrical Power System*
ESE	*: Experiential Systems Engineering*
FPGA	*: Field Programmable Grid Array.*
GPIO	*: General Purpose Input / Output*
I2C	*: Inter-Integrated Circuit*
INCOSE	*: International Council on Systems Engineering*
ISISPACE	*: Innovative Solutions In Space*
LEO	*: Low Earth Orbit*
MarCO	*: Mars Cube One*
MBSE	*: Model-Based Systems Engineering*
MQTT	*: Message Queuing Telemetry Transport*
OBC	*: On-Board Computer*
PCB	*: Printed Circuit Board*
PLA	*: Poly-Lactic Acid*
SEU	*: Single EventUpset*
SPEL	*: Space and Planetary Exploration Laboratory*
SPI	*: Serial Peripheral Interface*
SSE	*: Space Systems Engineering*
TCS	*: Thermal Control System*
UART	*: Universal Asynchronous Receiver-Transmitter*
UNISEC	*: University Space Engineering Consortium*
WLAN	*: Wireless Local Area Network*

1 Introducción

1.1 Contexto

En la última década, el sector espacial chileno ha mostrado un crecimiento evidente tanto en los ámbitos académico, público y privado. Ejemplos claros de esto son el desarrollo de los nanosatélites SUCHAI por parte del SPEL de la Universidad de Chile, el Sistema Nacional Satelital (SNSat) impulsado por la Fuerza Aérea de Chile, y el satélite Lemu Nge, primer proyecto de este tipo desarrollado por una empresa privada en el país. Estos avances reflejan un interés real en el desarrollo de sistemas satelitales en el país.

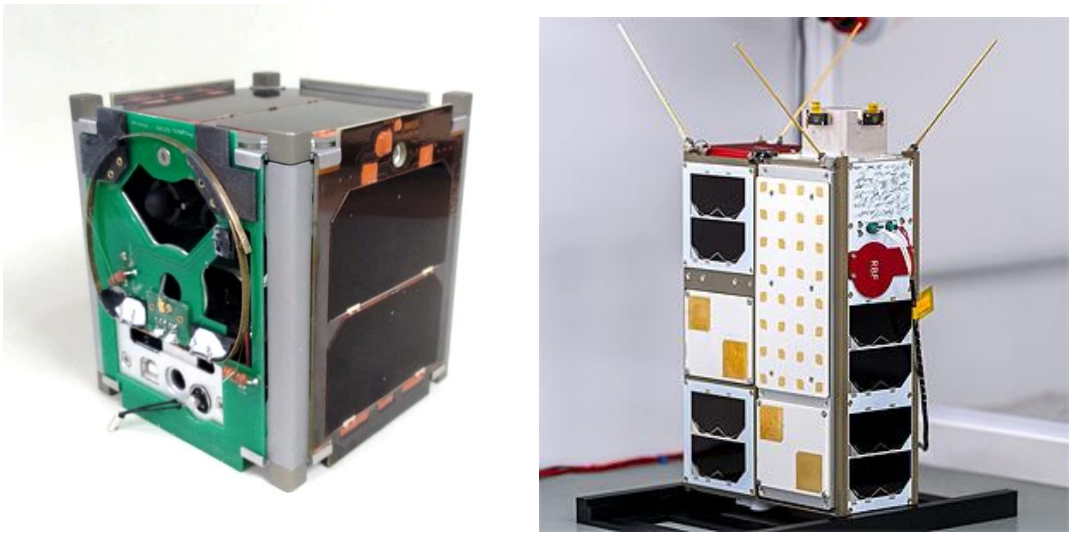


Figura 1: Satélites SUCHAI 1 (izquierda) y Lemu Nge (Derecha).

A medida que los proyectos aumentan en cantidad y complejidad, no solo se requieren equipos más grandes y mejor coordinados para poder suplir las demandas de estos. Sumado a esto, la complejidad técnica y altas demandas de sofisticación tecnológica de los sistemas espaciales más avanzados impone una necesidad creciente por profesionales altamente especializados. La llamada Pirámide de Capital Humano Especializado (Figura 1) ilustra este problema: mientras más grandes, avanzados y costosos son los satélites por desarrollar, más reducido es el grupo de profesionales capaces de llevarlos a cabo, culminando en que los proyectos más ambiciosos no podrán ser desarrollados a

menos que se cuente con un equipo de primer nivel, un “Dream Team”, equivalente a la selección nacional en ámbito deportivo.

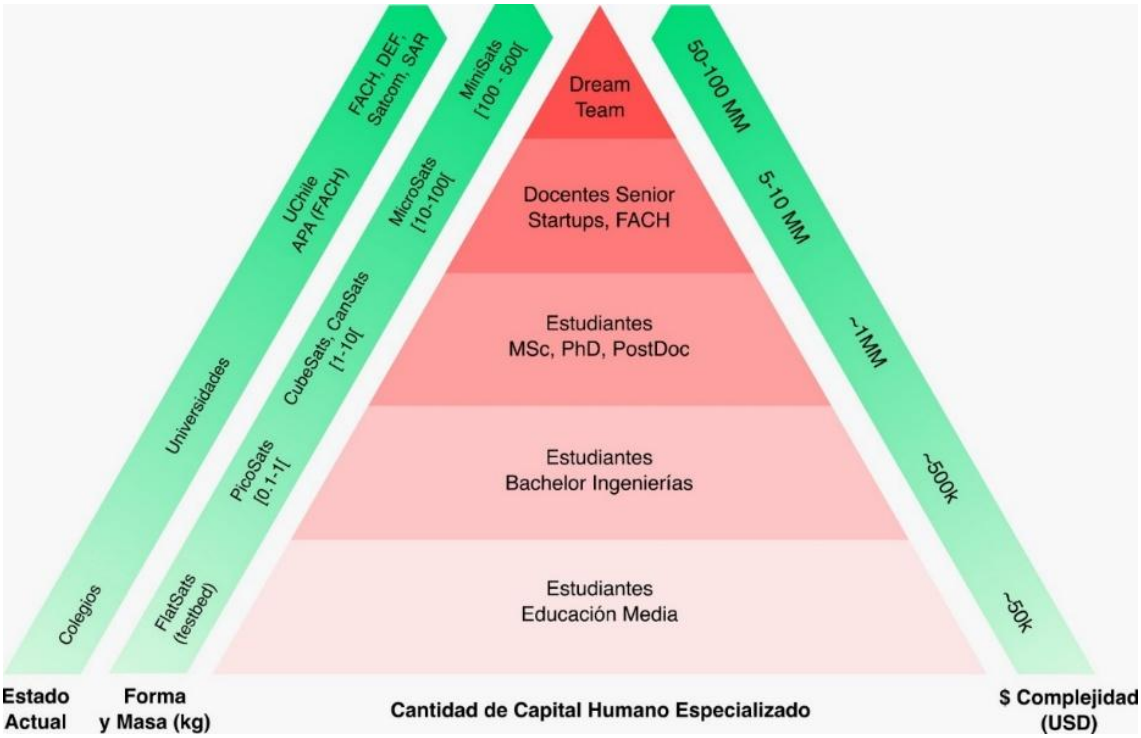


Figura 2: Pirámide de Capital Humano Especializado que ilustra el estado actual del sector de desarrollo satelital en Chile, y las capacidades teóricamente alcanzables en cada nivel. Fuente: López, A. No publicado.

1.1.1 Space Systems Engineering

1.1.1.1 Complejidad y especialización

Los sistemas espaciales, como los sistemas satelitales, son eminentemente complejos. Las limitantes y peculiaridades del entorno operacional en que se encuentran requieren un desarrollo muy meticuloso y altamente especializado. Fenómenos como la exposición al vacío, la microgravedad, o las diferentes fuentes de radiación presentes fuera de la protección de la atmósfera general efectos sobre los materiales y componentes de cualquier sistema expuesto a estos entornos. Un ejemplo de estos fenómenos son los *Single Event Upset* también llamadas “bit-flips”, donde una partícula ionizada parte del efecto de la radiación cósmica puede cambiar estados binarios al interactuar con componentes electrónicos (Binder et al., 1975). Otro fenómeno notable es el *Cold Welding*, donde superficies de metal se pueden unir molecularmente al contacto directo, esto debido a la ausencia de

una capa de oxido en la superficie que normalmente evitaría el contacto directo entre la matriz metálica de ambas partes (Merstallinger et al., 2009).

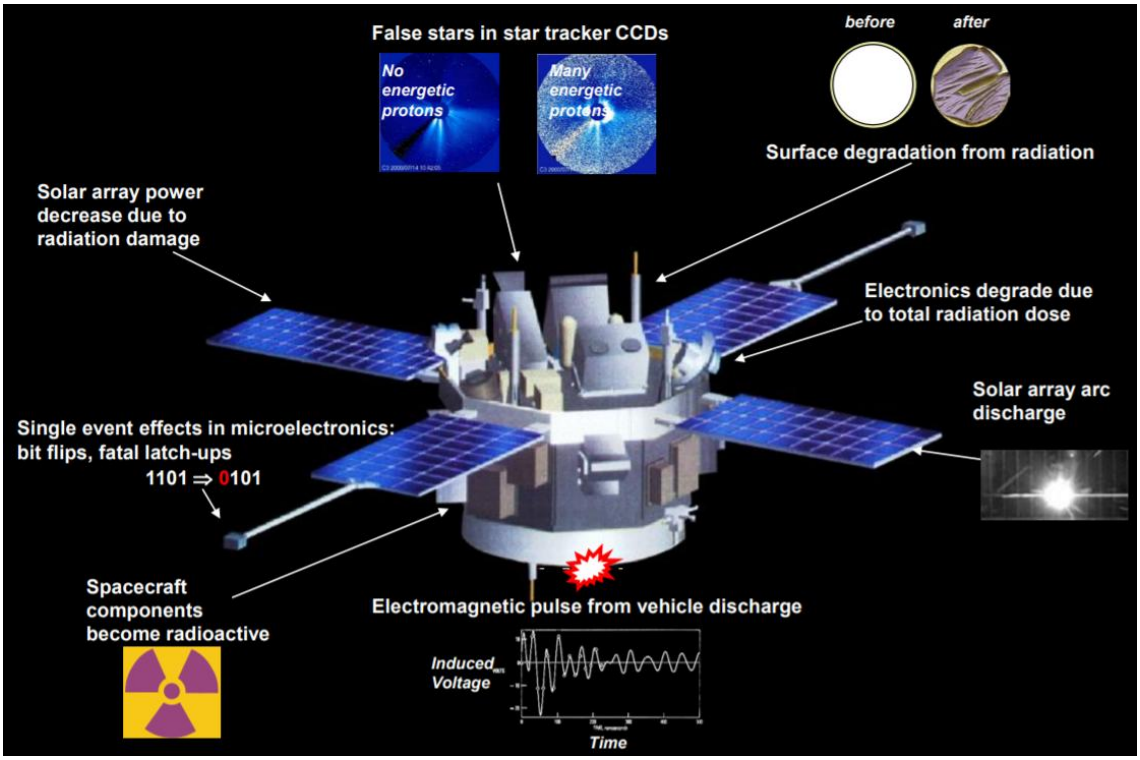


Figura 3: Representación de distintos riesgos ambientales a los cuales está sujeto un sistema espacial. Obtenido de (Friedlander, 2018), figura no original.

De lo anterior, es fácil observar cómo la complejidad de un sistema espacial puede escalar rápidamente en la medida que se continúa el análisis en detalle. Esto significa, a grandes rasgos, la presencia de dos tendencias de enfoque en diseño opuestas, pero complementarias. Por un lado, la necesidad de especialización profunda en áreas específicas del sistema (p.ej. electrónica), permitiendo que un profesional de más alto nivel pueda evaluar, diseñar e implementar un sistema que satisfaga las necesidades particulares de su propia disciplina. En este caso también existe un riesgo de sobre especializar un equipo en un área en particular, o exceder la capacidad esperada del sistema en esta

área, mientras que otras podrían ser dejadas de lado o no estar a la par con la competencia en otras áreas. Esto se ilustra, a grandes rasgos, en la Figura 4.

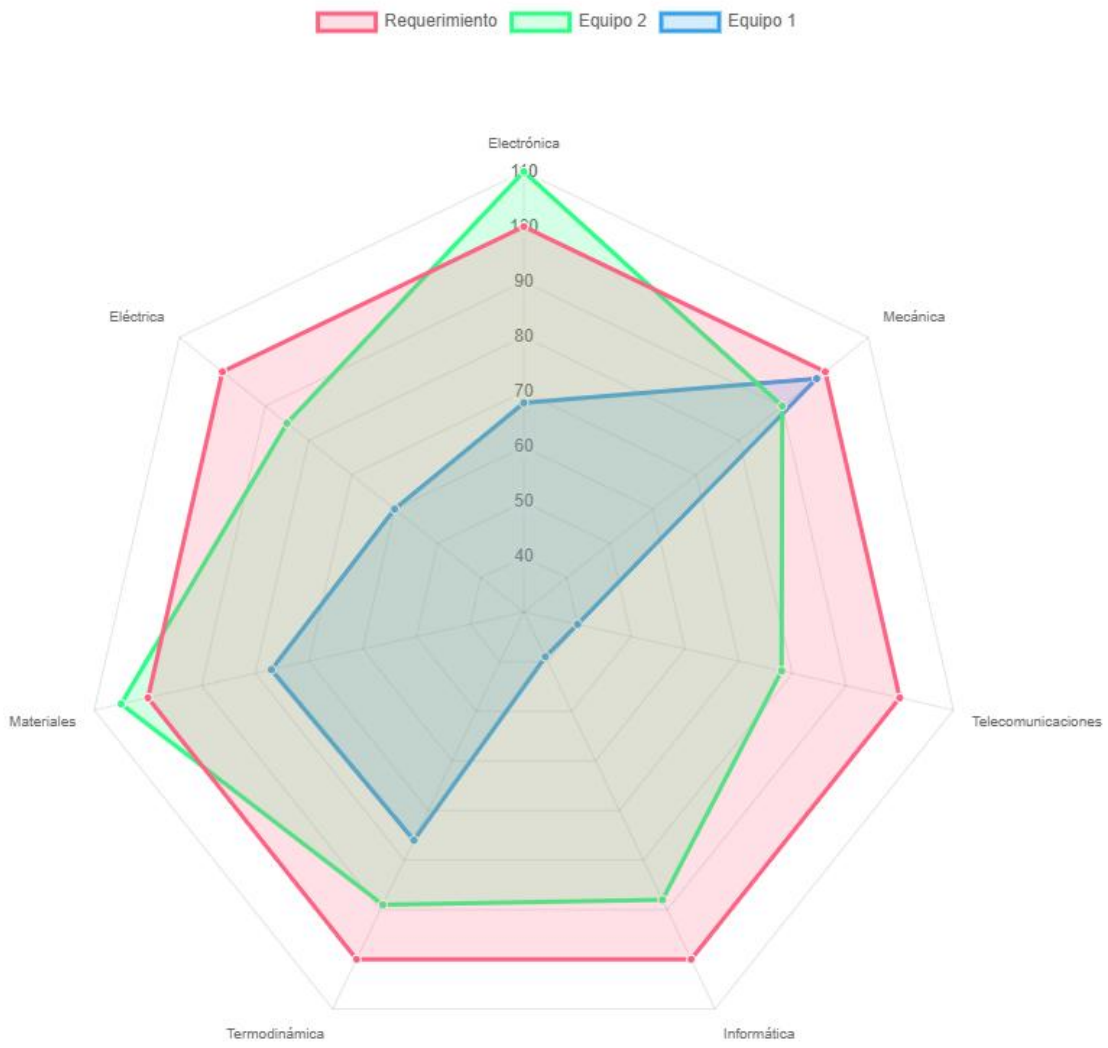


Figura 4: Esquema de radar ilustrando la competencia de dos equipos en diversas disciplinas.[1]

En la figura anterior, el hipotético equipo 1 está sobre especializado en un área (mecánica), pudiendo cumplir con gran parte de las expectativas del sistema en esta área, pero siendo deficiente en áreas como telecomunicaciones o informática. El equipo 2 está más balanceado, pero supera la expectativa en ciertas áreas (electrónica y materiales). En teoría, la diferencia entre las capacidades y el requerimiento siempre implicarían un gasto extra de recursos para suplir (en caso de no ser suficiente), o un "overengineering" del área donde se es más competente. Esta última es menos conflictiva, pues el riesgo de un fallo en esta área sería menor, pero el tiempo y recursos dedicados a sobredimensionar

[1] Los valores representados en la figura son completamente arbitrarios. Elaborado utilizando herramienta web: https://www.chartjs.org/docs/latest/charts/radar.html.

la solución en esta área podrían ser mejor invertidos en áreas con competencias deficientes respecto a lo requerido.

Uno de los riesgos de la sobre especialización en un área es el de minimizar las necesidades de otras áreas. Esto se ve ilustrado (en forma de caricatura) en la Figura 5.

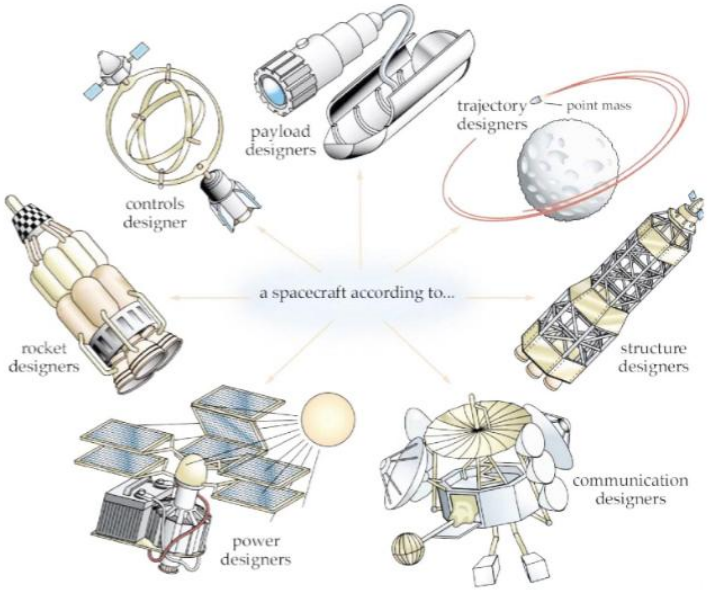


Figura 5: Caricatura ilustrando diferentes visiones de un sistema espacial de acuerdo con los diseñadores de diferentes subsistemas. Obtenido del material de la clase "Spacecraft Subsystems" del curso "Fundamentos de la Ingeniería de Sistemas Espaciales", López, A. No publicado.

Esto ocurre principalmente debido a que siempre se puede seguir aumentando la complejidad de un sistema, lo mencionado en los primeros párrafos de esta sección hace eco con, parafraseando la Ley de Tesler (Yablonski, 2024), "La complejidad no se reduce, solo se maneja", pero también con la noción de que la complejidad siempre puede aumentar (Chechile, 2021).Un corolario a ambas podría ser que en la medida que la competencia en un área aumenta, el horizonte de complejidad dentro del área también aumenta.

En este sentido, los satélites tipo CubeSat ofrecen una plataforma especialmente adecuada para la enseñanza, particularmente en el marco del New Space (Aboaf et al., 2020; Schilling, 2024). Se trata de sistemas estandarizados, de relativa simplicidad, pero con aplicaciones reales en la industria. Esto permite que los estudiantes se familiaricen con arquitecturas satelitales concretas y con la integración de subsistemas, sin necesidad de recurrir a proyectos de alto costo o riesgo. Adaptados con fines educativos, los CubeSats permiten introducir prácticas de SSE de forma didáctica, a la vez que abren la posibilidad de diseñar y fabricar soluciones localmente utilizando componentes comerciales (COTS) y procesos accesibles como impresión 3D o fabricación de circuitos electrónicos.

Volviendo a la idea de manejar la complejidad, esta labor corresponde a la segunda mitad de las tendencias de enfoque, la generalización. De la misma forma que todos los músicos son solistas sin un director de orquesta, todos los ingenieros son especializados sin un director de proyecto. Una sinfonía no surge (normalmente) por una organización descentralizada emergente, sino que siguiendo a un director capaz de hacer que todos los músicos cumplan con una parte del todo. En el caso de un

equipo ingeniería trabajando en un sistema complejo, cada ingeniero especialista actúa en conjunto para facilitar la emergencia del sistema, y el director de orquesta de esta analogía sería un ingeniero de sistemas (Systems Engineer, o en el caso de dominio específico, Space Systems Engineer).

1.1.1.2 Descomposición funcional de un sistema espacial

Según la *European Cooperation for Space Standarization* (o por su sigla ECSS), el órgano europeo para la estandarización espacial, un sistema espacial se puede descomponer en primera instancia en tres dominios mayores o segmentos, con un cuarto *bonus* correspondiente a todo lo que es soporte (*Support Segment*) (ECSS Secretariat, 2012). Estos serían: *Space Segment* (espacial), *Ground Segment* (terrestre) y *Launch Segment* (lanzador). La descomposición completa se observa en la Figura 6.

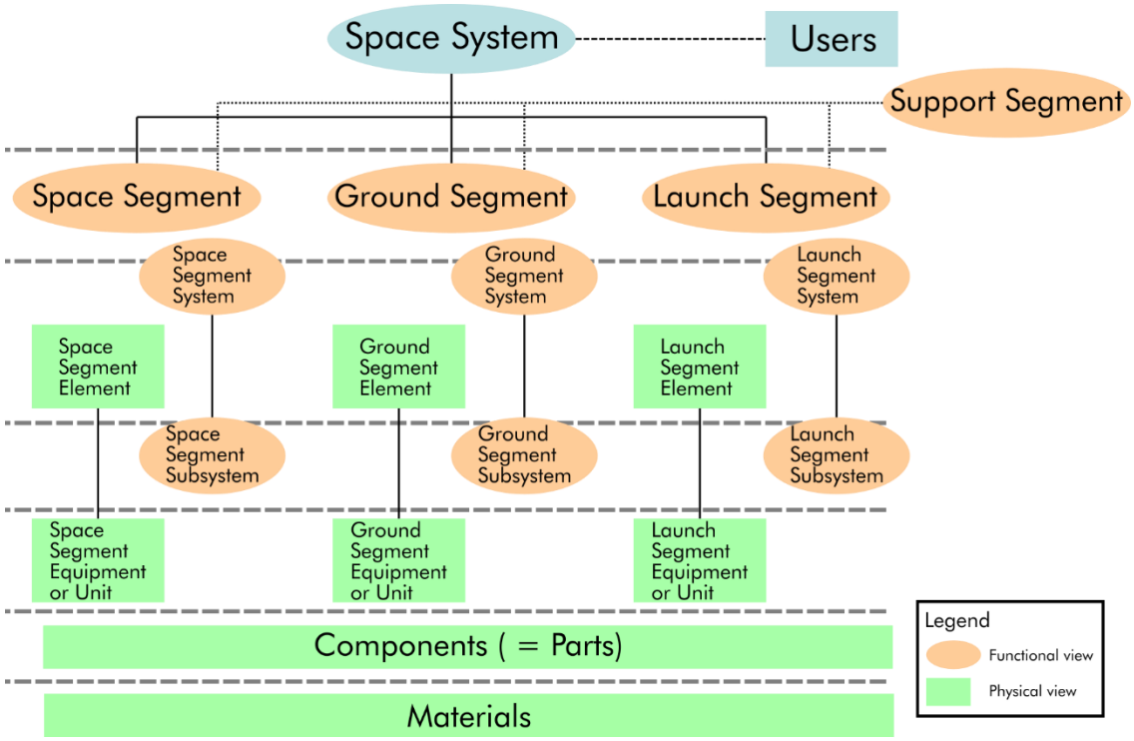


Figura 6: Diagrama ilustrando la descomposición jerárquica de un sistema espacial (ECSS).

1.1.2 Ciclo de vida y procesos de integración (AIT/AIV)

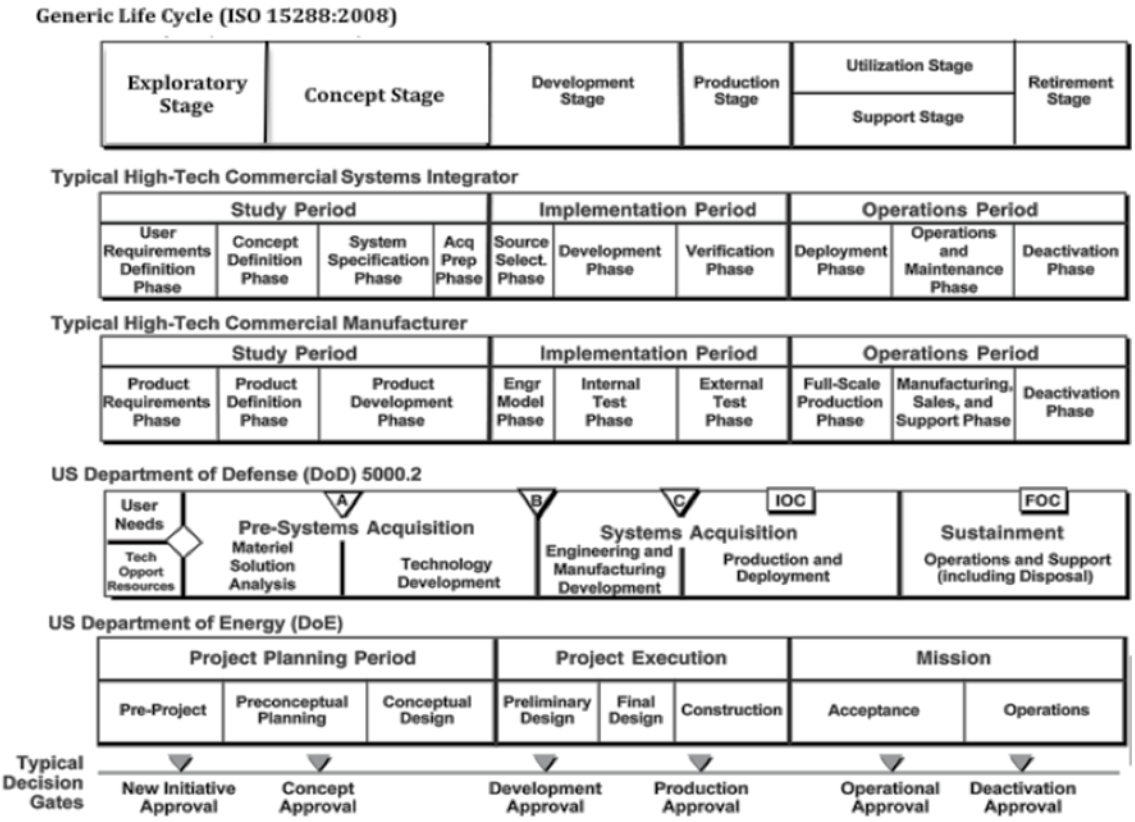


Figura 7: Modelos de ciclos de vida típicos de sistemas. (INCOSE, 2023)

Una de los procesos críticos en la ingeniería de sistemas corresponde a los procesos de integración, debido a que usualmente en estos se pueden presentar errores o fallas que puedan retrasar un proyecto. Es por esto que los procesos de ensamble, integración y verificación toman más relevancia. La capacidad de detectar errores antes de que su corrección represente un retroceso o atraso significativo del proceso. Esto también es particularmente relevante al trabajar con un gran número de satélites siguiendo la lógica “lean”, donde los ciclos de desarrollo son mucho más acotados y la confiabilidad se puede ver comprometida (Faure et al., 2017).

Bajo esta lógica, durante el año 2022 (publicado en 2023) se buscó desarrollar una herramienta de apoyo a la enseñanza de Systems Engineering, en la memoria de título de Bastián Villarroel: el HAISE-Sat (Figura 2) (Villarroel, 2023). Este fue posteriormente evaluado en el marco del proyecto de ingeniería desarrollado durante el semestre pasado (2025-1), donde se concluyó que, al contrastar las necesidades y requerimientos actuales de distintos stakeholders de la Facultad de Ingeniería, no cumplía con algunos aspectos críticos. Entre ellos, la capacidad de controlar y monitorear la telemetría del satélite sin depender de un computador, o la posibilidad de conectar múltiples satélites a un mismo sistema. A partir de ese diagnóstico, se elaboró un planteamiento conceptual de una herramienta que

sí cumpliera con dichas necesidades, llegando incluso a un prototipo funcional al cierre del proyecto, mostrado en la Figura 8 (González, 2025).

Es precisamente desde este punto que se plantea ahora continuar el desarrollo, iterando sobre el prototipo existente para ajustarlo a las normas y especificaciones estandarizadas del CubeSat Design Specification (CDS).

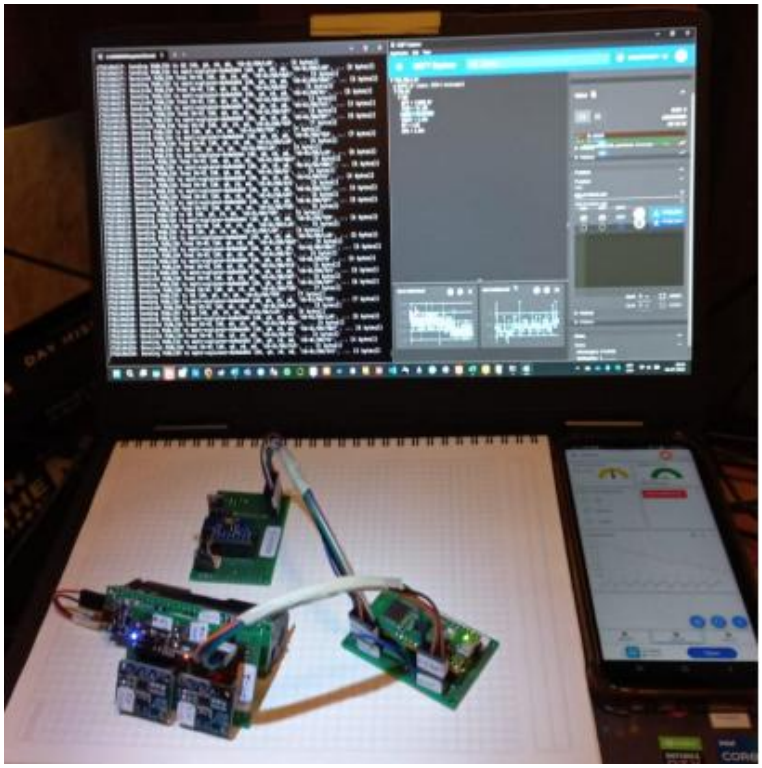


Figura 8: Prototipo de satélite educativo desarrollado en PIA. (González, 2025)

1.2 Hipótesis (Condiciones de diseño)

Esta memoria de título plantea que la implementación de un sistema educativo basado en una experiencia práctica enfocada en el ciclo de Armado-Integración-Verificación (AIV) en base a satélites educativos con factor de forma CubeSat se puede desarrollar siguiendo las siguientes consideraciones de diseño:

C1. El satélite seguirá el estándar CubeSat CDS 14.1 o posterior. (The CubeSat Program, 2022)
C2. El satélite seguirá el estándar PC/104 para sus tarjetas de circuito impresas (PCB), de acuerdo con el estándar de GomSpace.
C3. El costo recurrente por unidad será inferior a 200 $USD.
C4. El costo no recurrente de la experiencia deberá ser inferior a 300 $USD.
C5. El satélite utilizará la Raspberry Pi Pico W como computador a bordo (OBC).
C6. La experiencia deberá ser desarrollada en menos de 2 horas de clase (90 minutos) de principio a fin.
C7. El satélite deberá ser compatible con payload en desarrollo por MT de Nicolás Valderrama.

1.3 Objetivos

1.3.1 Objetivo general

Implementar una experiencia educativa de armado, integración y verificación (AIV) en base a satélites educacionales en formato CubeSat para aplicaciones en cursos de Systems Engineering.

1.3.2 Objetivos específicos

1. Desarrollar un prototipo funcional de la experiencia contemplando un kit funcional de satélite, equipamiento de apoyo y el flujo de la experiencia.
2. Implementar un modelo digital que describa la arquitectura del sistema en una herramienta aplicable basado en el CSRM. (OMG, 2023)
3. Evaluar el diseño del prototipo para producción en lotes de 5 a 10 unidades.

1.4 Metodología

Para cumplir con los objetivos planteados se propone el siguiente procedimiento:

Objetivo 1: Desarrollar un prototipo funcional del satélite educativo: Se realizará un diseño detallado siguiendo metodologías de Systems Engineering según INCOSE (INCOSE, 2023) a partir del ConOps y de las especificaciones entregadas en el PIA previamente realizado (González, 2025). El diseño se llevará a una primera versión de prototipo por medio de software CAD para elementos estructurales, y posteriormente realizando el armado utilizando componentes COTS, el uso de una Raspberry Pi Pico W, elementos fabricados como marcos impresos en 3D y PCBs, y la integración de un payload óptico. Se aplicarán procesos iterativos de diseño, ensamblaje y verificación para garantizar el correcto desarrollo del prototipo.

Se emplearán métodos de Systems Engineering, didáctica profesional y ergonomía cognitiva para diseñar una experiencia de aprendizaje efectiva. Esta buscará llevar la complejidad hacia el diseñador del sistema, de manera que el estudiante se concentre en la comprensión de los conceptos y no en los detalles técnicos menores o en distracciones innecesarias. La experiencia abarcará una introducción teórica, actividades prácticas de ensamblado, integración y verificación, una evaluación de salida y un cierre.

La experiencia será implementada en un piloto con estudiantes y evaluada en base a indicadores de éxito definidos previamente. Se considerarán tanto métricas técnicas (funcionamiento del satélite educativo) como pedagógicas (aprendizajes alcanzados, carga cognitiva percibida, satisfacción de los participantes). Estas métricas serán consultadas con los stakeholders, tanto en su planteamiento como en su medición.

El diseño e integración de un payload óptico sencillo se desarrollará en paralelo al prototipo general, verificando compatibilidad eléctrica, mecánica y didáctica. Este será seleccionado a partir de una evaluación de alternativas y buscará ser compatible con otros payloads, como el desarrollado por MT de Nicolás Valderrama.

Objetivo 2: Implementar un modelo digital que describa la arquitectura del sistema en una herramienta aplicable basado en el CSRM: El prototipo se modelará en una plataforma que permita implementar un modelo de sistema MBSE (Violet Labs). Se establecerán requerimientos, interfaces y métodos de verificación/validación. Se adoptarán los lineamientos del CSRM y la CDS 14.1 para asegurar compatibilidad y replicabilidad del diseño.

Objetivo 3: Evaluar el diseño del prototipo para producción en lotes de 5 a 10 unidades. A partir del diseño preliminar, se evaluarán costos, tiempos y recursos necesarios para producir un lote de satélites educativos. Esto implicará elaborar presupuestos, agendas de trabajo y estimaciones de manufactura, considerando tanto los subsistemas como los procesos de ensamblaje, integración y verificación.

1.4.1 Declaración respecto al uso de herramientas de inteligencia artificial (IA)

Con el propósito de favorecer la transparencia y el uso ético de herramientas de IA generativa de acuerdo con recomendaciones para publicaciones (COPE Council, 2024; Elsevier, 2025; GYAP, 2025), se incluye la siguiente declaración:

Durante el desarrollo de la presente memoria de título se utilizaron herramientas de IA generativa, particularmente ChatGPT de OpenAI, y Gemini de Google, principalmente como una herramienta de apoyo a la investigación en disciplinas complementarias, la investigación de alternativas de diseño, y la verificación de fuentes de investigación, así como la elaboración de scripts en Python para la implementación rápida de software. Esta información fue verificada donde fuera posible. Los contenidos vertidos sobre este documento son de ***responsabilidad única del autor, y no se declara coautor a ninguna herramienta de IA generativa*** *incluida o no en esta declaración.*

Adicionalmente se declara la inclusión de una única imagen generada por IA, siendo esta la incluida en la sección del Concepto de Operaciones (Figura 15). El motivo para incluir esta figura es para mejorar la claridad del Concepto de Operaciones del sistema desarrollado, permitiendo presentar una figura más refinada con un tiempo de implementación menor. Esta misma figura fue retocada manualmente después de ser generada para corregir errores menores en su diseño.

2 Estado del arte

Si bien la teoría respecto al uso de kits educativos para el apoyo en la enseñanza de disciplinas técnicas no es una invención moderna, las técnicas y metodologías utilizadas en la implementación de estos han evolucionado desde la implementación de los primeros kits de laboratorios, particularmente en ciencias puras. Respecto a la enseñanza en ingeniería, la tendencia actual busca aplicar enfoques más integrativos y sistemáticos a la resolución de problemas y diseño de soluciones, principalmente con el enfoque de que lo aprendido y aplicado en clases no se limite a enfoques teóricos, sino que refleje y fomente habilidades relevantes para la práctica profesional de la ingeniería moderna. (Crawley et al., 2007).

Por otro lado, cuando el enfoque está en entregar una formación directamente relacionada con los principios de la ingeniería de sistemas, otros métodos se han aplicado para facilitar la enseñanza de métodos y herramientas de la ingeniería de sistemas de una forma más visceral. Un ejemplo es el caso

del enfoque del Experiential Systems Engineering (ESE) (Golkar, 2020), donde igualmente se aplican principios del método CDIO.

Algo en lo que las fuentes encontradas suelen estar de acuerdo es en el valor agregado que representa una experiencia práctica al momento de aplicar, reforzar y consolidar los conocimientos teóricos (Aboaf et al., 2020; Golkar, 2020; KISPE, 2023; Loveless, 2018; Muller & Bonnema, 2013; Schilling, 2024). En efecto, esto se basa en los principios establecidos por el mismo método CDIO (Crawley et al., 2007), el cual a su vez se relaciona con principios del aprendizaje experiencial (Kolb, 1984).

Una de las fases críticas del ciclo de vida de un sistema espacial corresponde al proceso de integración, donde se ensamblan componentes a un subsistema, o subsistemas e instrumentos se ensamblan en un sistema, asegurándose que cada unidad individual funcione correctamente al estar interconectada (Pisacane, 2005). Uno de los motivos por los cuales este proceso es importante es que poder identificar errores o fallas en componentes o instrumentos en la medida que son integrados, evita que estos errores puedan propagarse, o impliquen un proceso de desensamble en caso de ser identificados más adelante en el proceso de integración, lo que implica retrasos y costos mayores correctivos. En ciclos rápidos de desarrollo, se ha mostrado que los procesos de verificación y testeo durante procesos de integración permite una identificación de errores y fallas con anterioridad (Faure et al., 2017).

2.1 Estado de la práctica

En la facultad de ingeniería de la Universidad de Concepción, el método de enseñanza de ingeniería ha integrado elementos descritos anteriormente, siendo un ejemplo notable la implementación progresiva de la metodología CDIO en diferentes cursos (Salas et al., 2024). Aun así, estos cursos utilizan las fases completas del ciclo CDIO para ser aplicadas en resolución de problemas genéricos en ingeniería, sin un enfoque especial en una de las cuatro áreas. Más aun, las asignaturas se limitan al plazo de un semestre académico, teniendo un tiempo limitado para profundizar en procesos más detallados como lo serían los ciclos AIT o V&V.

Respecto a kits educativos aplicados en Chile, hoy en día los proyectos gubernamentales se enfocan principalmente en los CanSats (FACH, 2023), satélites miniaturizados cuyo diseño consiste en que puedan ser introducidos dentro del volumen de una lata de bebidas (350 ml). Estos proyectos están orientados principalmente a estudiantes secundarios en Chile (ver Figura 9), mientras que el diseño de los CanSat está orientado originalmente a ser un desafío para estudiantes como una primera introducción a desafíos de la ingeniería (Gansmoe et al., 2015).

Figura 9: Imagen promocional del programa piloto de educación espacial escolar.

Estos proyectos, aunque valiosos para el fomento de las disciplinas STEM y el desarrollo de habilidades técnicas relacionadas con sistemas espaciales, carecen de un vínculo directo entre la introducción y familiarización de conceptos, y las aplicaciones actualmente vigentes en la idustria espacial, lo que es relevante al momento de entrenar profesionales o estudiantes superiores en procesos complejos de ingeniería, o en el uso de estándares vigentes.

Históricamente, uno de los kits que más se han acercado a las necesidades y estándares industriales en los procesos educativos ha sido el EyasSat (EyasSat LLC, 2011), un kit mucho más robusto que integra procesos más avanzados del ciclo de ingeniería, tal como pruebas de aceptación y ciclo AIT/AIV, dividido en múltiples experiencias de laboratorio, con un enfoque en la integración de subsistemas. EyasSat fue desarrollado hace más de 20 años (2004), y aunque su diseño representa una aproximación razonable para su tiempo, múltiples avances han ocurrido desde su implementación original que han ido dejando obsoletos varios estándares como el factor de forma, restando un poco el valor de la experiencia hoy en día.

Con la implementación masiva del estándar CubeSat como una plataforma versátil no solo en contextos educativos, sino que en aplicaciones de uso comercial e investigación (Bouzoukis et al., 2025), cuyo uso solo ha ido aumentando desde el 2013 (Figura 10), demostrando ser una plataforma capaz de soportar misiones fuera de la región LEO como demostró el Mars Cube One (MarCO), como parte de la misión Mars InSight (NASA, 2018).

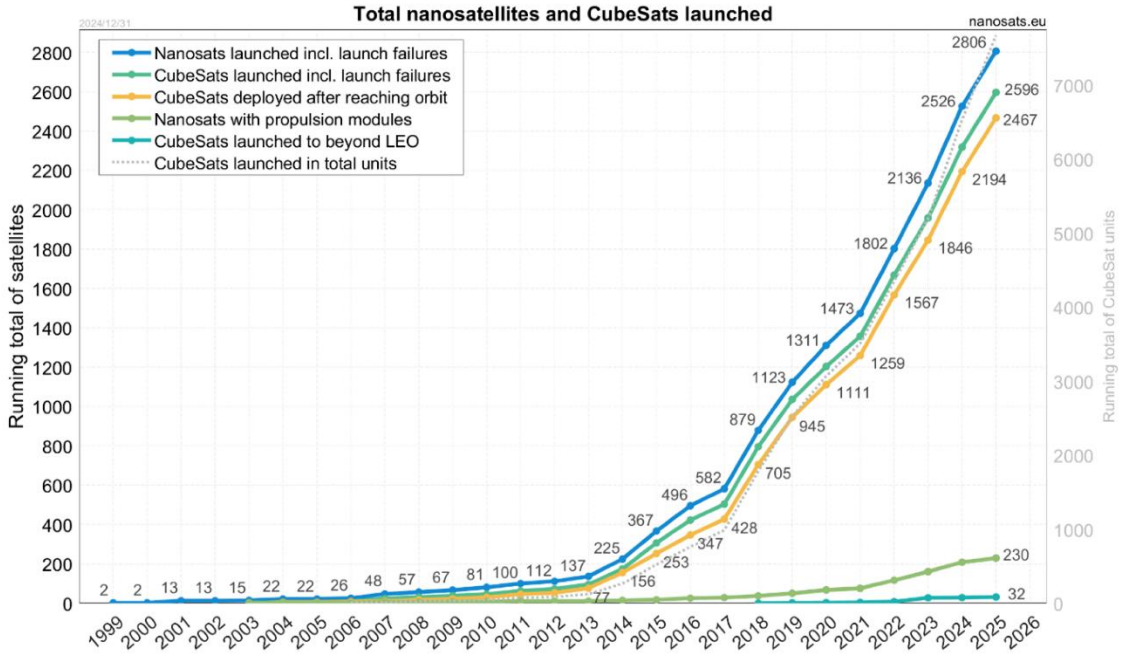


Figura 10:Número total de nanosatélites y CubeSats lanzados entre 1998 y 2025 (Kulu, 2025)

En respuesta a todo lo anterior, el uso de CubeSats como herramientas de apoyo a la educación en ingeniería, en particular respecto al aprendizaje experiencial, resulta una oportunidad invaluable para que estudiantes no solo practiquen técnicas asociadas a al ingeniería de sistemas, sino que también permitan familiarizar a los estudiantes con tecnologías vigentes y versátiles que bien podrían ser implementadas en una variedad de aplicaciones en la industria. Así, nuevos kits que sigan estos estándares son atractivos, y asociaciones como Orion Space han desarrollado alternativas modernas que buscan replicar estos factores de forma como el CUBEsat Educational Kit (CUBEEK), cuyo bajo costo y orientación hacia la educación permite una alternativa más fiel a los factores de forma presentados anteriormente (Orion Space, 2024). Una limitante que presenta este kit es que busca atender el mismo segmento de estudiantes equivalentes a la enseñanza media, por lo que algunas características no se ajustan a definiciones más estrictas de envelopes e interfaces (Figura 11).

Figura 11: Cubesat Educational Kit (CUBEEK). Orion Space, Nepal.

Una alternativa que conserva mejor las interfaces y el factor de forma de CubeSats actuales, sin perder el enfoque en educación, corresponde al EssentialSAT (Figura 12), un kit de apoyo que integra el proceso AIT/AIV, junto con simulaciones de estación terrestre, EGS, y control (KISPE, 2023).

Figura 12: Demostración del EssentialSAT durante un evento de TSTI en colaboración con ESA/ESTEC.Fuente: LinkedIn[2].

[2] Teaching Science & Technology, Inc. (TSTI), LinkedIn.

Una limitante significativa del EssentialSAT es su elevado costo, el cual puede ascender a más de 12,900 $USD por kit[3]. Considerando que el costo de implementación del sistema presentado en este informe se busca limitar a $200 USD por kit, el EssentialSAT es significativamente más costoso.

En 2022-2023, en la Universidad de Concepción se buscó desarrollar un kit equivalente siguiendo limitantes similares a las de este proyecto (Villarroel, 2023), consiguiendo un prototipo inicial mostrado en la Figura 13. Durante el PIA desarrollado como proyecto precursor a esta MT, se realizó un análisis de este, determinando que, aunque representa avances en algunos frentes, particularmente en la implementación de estructura y hardware, las principales limitantes de este kit estaban en la implementación del software y la experiencia educativa en sí (González, 2025).|

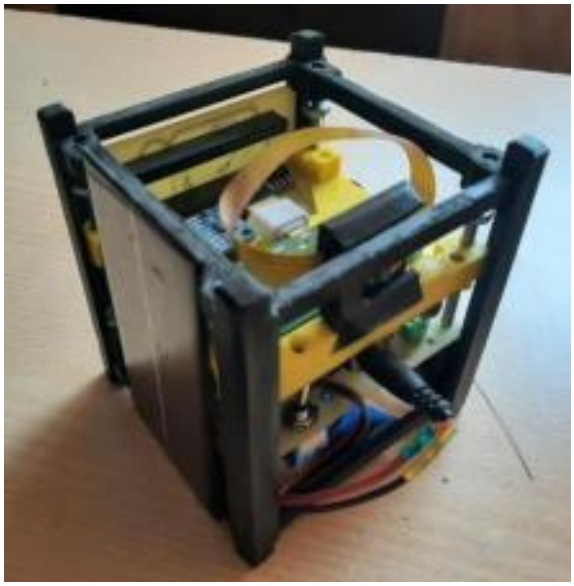


Figura 13: HAISE-Sat desarrollado en MT (Villarroel, 2023).

[3] La página de la tienda de KISPE presenta el valor en £9,700 por kit. A la fecha de este informe, este valor asciende a $12,908 USD. https://www.kispe.shop/product/essentialsat-standard-kit

3 Diseño conceptual

3.1 Definiciones generales

A partir de lo desarrollado anteriormente durante el Proyecto de Ingeniería Aeroespacial (González, 2025), se tiene un planteamiento general de la arquitectura del sistema a desarrollar. Este será descrito a continuación.

3.1.1 Definiciones generales, stakeholders y requerimientos

Según INCOSE, un stakeholder se definen como cualquier individuo, grupo u organización con interés en, o preocupaciones relativas a un sistema. Se identifican tres stakeholders principales que representan distintos ámbitos de interés y responsabilidad:

- El Prof. Alejandro López, como tutor y mandante, quien prioriza la disponibilidad de una herramienta asequible, fácil de operar y desplegar, alineada con las necesidades formativas en ingeniería de sistemas para diversos niveles educativos.
- El Prof. Frank Tinapp, cuyo enfoque está en asegurar que el sistema permita una integración efectiva en distintos niveles de enseñanza y promueva aprendizajes significativos, particularmente en el contexto de estudiantes de primer año de ingeniería.
- Nicolás Valderrama, estudiante responsable del desarrollo de una FPGA, cuyo interés consiste en asegurar que las interfaces y arquitectura del sistema permitan la integración modular y escalable de este componente.

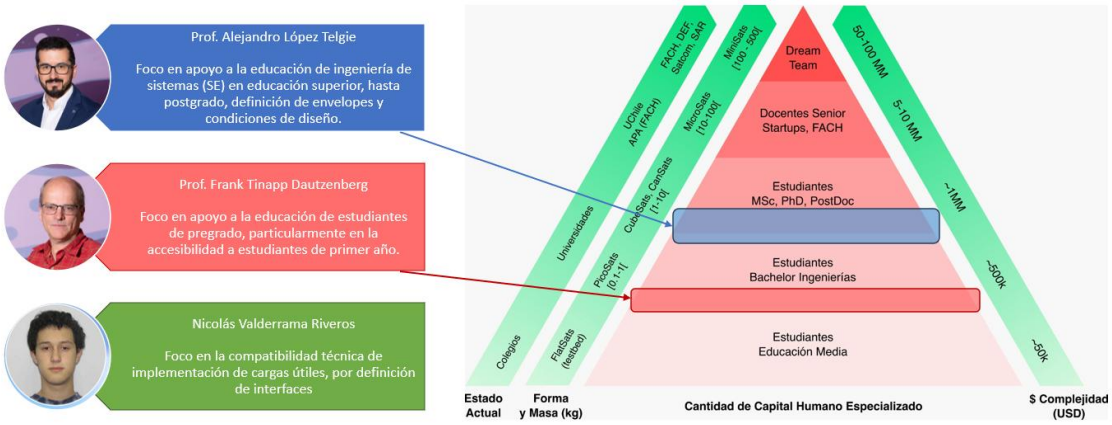


Figura 14: Resumen de stakeholders del sistema durante el proceso de diseño conceptual del sistema.

Estos roles definen requerimientos funcionales, pedagógicos y técnicos que se integran en el diseño del sistema. Para la recopilación de necesidades y la formulación de requerimientos de sistema, se realizaron breves entrevistas con los mencionados Stakeholders. A partir de estas, se obtuvo el siguiente conjunto de necesidades:

Alejandro López:

- **N-AL01:** La experiencia necesita poder soportar la aplicación de técnicas del proceso de integración de un sistema (AIT/AIV)

- **N-AL02:** La experiencia necesita reflejar características típicas de sistemas espaciales en la industria, como el factor de forma CubeSat, uso de bus entre placas de la forma PC/104, etc.
- **N-AL03:** Necesita que los estudiantes puedan conectarse al satélite utilizando sus teléfonos celulares, sin depender de un computador por equipo
- **N-AL04:** Necesita que la experiencia pueda ser realizada en clases. Considera limitación de tiempo, espacio, cantidad de estudiantes, e infraestructura presente. Estimación de tiempo de clases: 2 horas.
- **N-AL05:** Necesita que el satélite tenga una autonomía que garantice su operabilidad por la duración de la experiencia, sin necesidad de cargar o conectar a un computador.
- **N-AL06:** Necesita que la experiencia tenga un costo menor de 200USD por kit para su adquisición.
- **N-AL07:** Necesita que la experiencia se ajuste a una clase de 30 estudiantes, considerando trabajo en equipo de 3 integrantes, con un kit por equipo, más uno adicional para el expositor
- **N-AL08:** Necesita que el costo de mantenimiento de la experiencia completa (11 kits) sea menor a 100USD por semestre.

Frank Tinapp:

- **N-FT01:** Necesita que sea una experiencia práctica que acerque a los estudiantes a lo que significa ser un ingeniero aeroespacial.
- **N-FT02:** Necesita que los conocimientos requeridos por estudiantes para poder operar el sistema y participar en la experiencia sean comunes y entregados a nivel secundario, o que puedan ser explicados rápidamente en la misma sesión de hasta 3 horas.
- **N-FT03:** Necesita que los estudiantes puedan operar el sistema, realizar modificaciones menores no relacionadas con conocimientos de electrónica ni programación, pero que puedan resolver desafíos específicos por medio del método CDIO.
- **N-FT04:** Necesita que la experiencia se vincule a contenidos entregados por otras asignaturas del mismo nivel, así los estudiantes pueden integrar sus conocimientos desde diferentes áreas.
- **N-FT05:** Necesita que los estudiantes puedan conectarse al satélite utilizando sus teléfonos celulares.
- **N-FT06:** Necesita que el tiempo mínimo asignado a una experiencia sea de 3 sesiones, de máximo 3 horas cada una.
- **N-FT07:** Necesita que el tiempo máximo dedicado a un proyecto en torno a la experiencia sea de no más de 15 sesiones.
- **N-FT08:** Necesita que la experiencia permita la lectura de al menos un dato que represente una medida escalar (no vectorial) de fácil compresión.

Nicolás Valderrama:

- **N-NV01:** Necesita que el bus de potencia y datos entre subsistemas sea compatible con la FPGA.
- **N-NV02:** Necesita que el bus entre subsistemas y las interfaces mecánicas sigan el estándar PC/104 según estándar de-facto usado por GomSpace.

A partir de este conjunto de necesidades, se extrajeron el siguiente conjunto de requerimientos:

- **R-01:** La experiencia debe tener una arquitectura que permita la descomposición en elementos, subsistemas, y componentes o partes. (Desde **N-AL01**)
- **R-02:** La experiencia debe estar diseñada de forma que se aplique de forma práctica el ciclo de Ensamble-Integración-Testeo/Verificación, dentro de un plazo de mínimo 3 sesiones y máximo 15, aplicando conceptos y técnicas del método CDIO. (Desde **N-AL01**, **N-FT01**, **N-FT03** y **N-FT07**).
- **R-03:** La experiencia debe cumplir con estándares de la industria comunes en la actualidad para CubeSats, incluyendo el estándar CubeSat (CDS y CSRM). (Desde **N-AL02**)
- **R-04:** La experiencia debe permitir que los estudiantes puedan interactuar con el satélite por medio de sus teléfonos celulares, recibiendo datos y enviando comandos desde este. (Desde **N-AL03** y **N-FT07**).
- **R-05:** La experiencia debe costar máximo 200 USD para la implementación por cada kit (Desde **N-AL06**), y máximo 100 USD por el mantenimiento de todos los kits por semestre de operación (Desde **N-AL08**).
- **R-06:** La experiencia debe considerar al menos un kit de satélite por cada 3 estudiantes considerando un curso de 30, más un kit adicional para el expositor. (Desde **N-AL07**)
- **R-07:** La experiencia debe incluir la mínima cantidad de electrónica y programación, casi a nivel operativo, o completamente guiado en caso de ser necesario. (Desde **N-FT02**)

3.1.2 Concepto de Operaciones (ConOps)

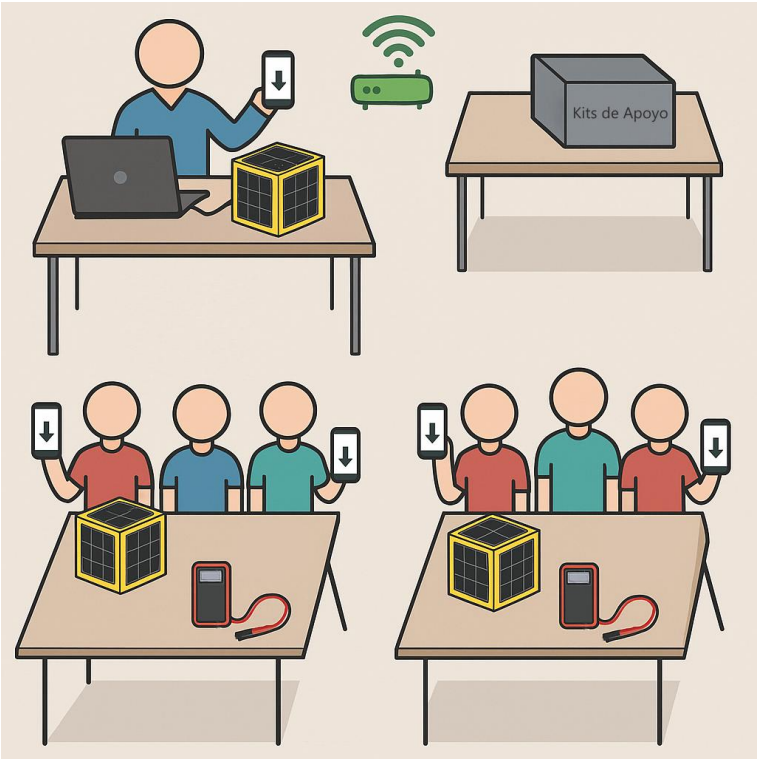


Figura 15: Representación simplificada del Concepto de Operaciones (ConOps) del sistema de apoyo propuesto en PIA anterior.

La operación del kit se realiza durante una clase presencial de máximo 3 horas pedagógicas de duración, en una sala equipada con mesas de trabajo y una red WiFi. Los estudiantes se organizan en

equipos de tres personas, cada uno con acceso a un kit satelital educativo (cubo amarillo en la figura), un multímetro digital (rombo rojo con cables) y una balanza digital gramera (rectángulo morado).

Durante la sesión, cada equipo lleva a cabo el ensamblaje de componentes, pruebas individuales y posteriores verificaciones tras la integración al sistema completo. Para esto, además de las herramientas disponibles en cada mesa grupal, se facilitarán kits de testeo comunes (elemento gris en la figura) que permitan realizar pruebas con equipo más especializado, pero que no necesariamente es usado en todo momento, como verificación de continuidad en cables o pines del sistema. Finalmente, se realiza una validación funcional del kit mediante la medición de una serie de parámetros clave.

Las métricas principales incluyen verificación de envelopes físicos, como masa y volumen, y funcionalidad del sistema mediante interpretación de telemetría (downlink) y ejecución de comandos (uplink). Esta interacción se realiza a través de los teléfonos celulares personales de los estudiantes, los cuales se conectan a los satélites mediante una aplicación con interfaz gráfica.

El profesor cuenta con un computador con conexión a la red WiFi del aula y la capacidad de conexión física a los kits mediante cable USB-micro USB. Esto le permite preparar, diagnosticar y corregir directamente el código del software de vuelo en cada kit satelital, además de facilitar la operación general antes y durante la actividad.

3.2 Definición de interfaces

Dado que una de las condiciones de diseño impuestas es la de permitir la integración de una carga útil desarrollada en una memoria de título en paralelo por Nicolás Valderrama (mencionado anteriormente como un stakeholder del sistema en el párrafo 3.1.1, y como condición de diseño C7), se han definido una serie de interfaces necesarias para coordinar una integración adecuada. A grandes rasgos estas se dividen en tres categorías: mecánicas, electrónicas y lógicas.

Cabe mencionar que estas interfaces también se proyectan para integraciones futuras, concordantes con las condiciones de diseño C1 y C2, además de facilitar la operación del equipo de apoyo electrónico (EGS).

Para facilitar la coordinación con MT de Nicolás Valderrama, se han realizado dos estándares de interfaces conjuntas: Interfaces mecánicas y estándares de configuración, y las interfaces lógicas.

El estándar de diseño, según la configuración, se define en la Figura 16:

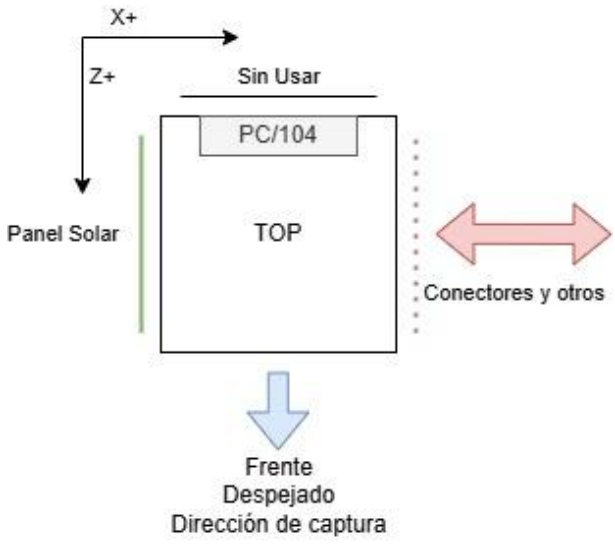


Figura 16: Definiciones de diseño para ejes de referencia, uso de caras y disposición general del PCB.

3.2.1 Interfaces mecánicas

Hay dos especificaciones que principalmente definen la mayoría de las interfaces mecánicas del sistema, el CubeSat Design Specification o CDS (The CubeSat Program, 2022), y el estándar PC/104, particularmente la versión implementada por GomSpace. A grandes rasgos, el CDS define las limitaciones e interfaces externas o envelopes del satélite, siendo una de las principales razones para esto el uso extendido de los P-POD o Poly-PicoSatellite Orbital Deployer, un dispositivo que permite transportar y desplegar múltiples unidades de CubeSats de forma estandarizada.

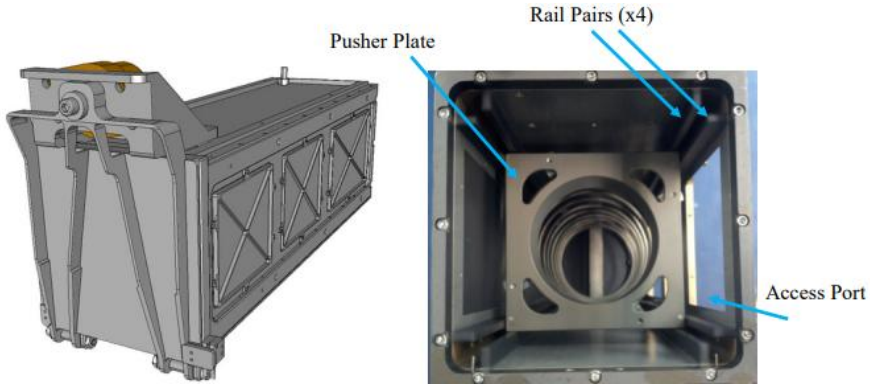


Figura 17: Un modelo de P-POD asociado al estándar CDS 14.1. Es relevante mencionar que este no es el único modelo utilizado, pero todos respetan las interfaces definidas aquí. (The CubeSat Program, 2022)

Los CubeSats se definen según el volumen utilizado. Este se encuentra estandarizado, y el principio fundamental es que cada unidad o U corresponde a 1 L de volumen, siendo un cubo regular de lado 10 cm (Figura 18). En la práctica, y de acuerdo con el CDS, las dimensiones definidas por el diseño no son tan regulares, dejando solo la sección transversal vertical (plano XY) como un cuadrado regular de lado 10 cm, pero con un alto en el eje Z de 11.35 cm, considerando que los rieles de las cuatro esquinas sobresalen verticalmente en ambos extremos. El otro driver que define la configuración del CubeSat es la masa de este, definido como 2 kg por cada 1U. Así, un CubeSat de 3U tiene una masa máxima de 6 kg, y una altura máxima de 34.05 cm.

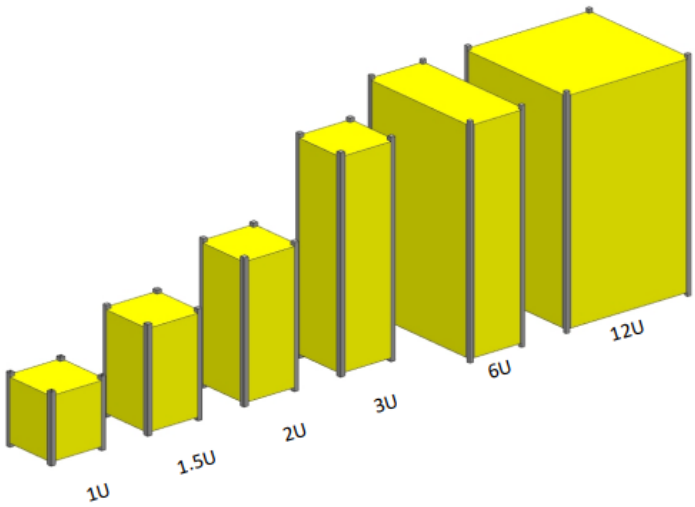


Figura 18: Familia de CubeSats según el CDS 14.1.

Respecto a las especificaciones del bus PC/104, este corresponde a una especificación de interfaces de comunicación entre PCBs. Esta especificación es ampliamente utilizada en la industria, siendo de la misma familia que el conector más reconocido PCI y PCI-Express, habitualmente presentes en electrónica computacional.

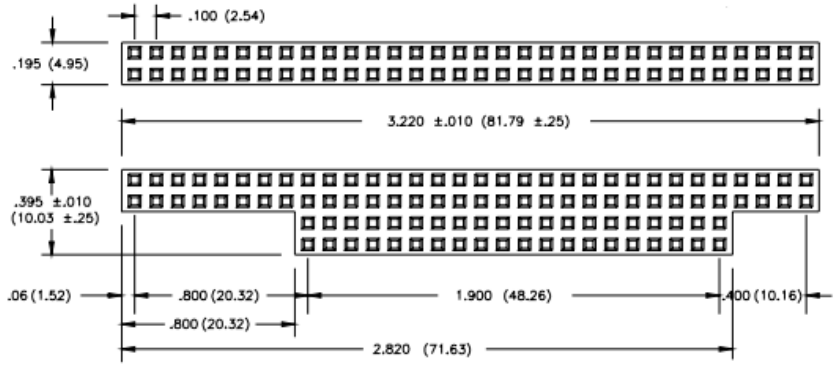


Figura 19: Conectores definidos por el estándar PC/104 de 8-bit (arriba) y 16-bit (abajo).

En los casos observados en la Figura 19, los conectores son directamente los definidos por la especificación del PC/104 para comunicación de 8 y 16 bits, siendo la de 16 bits la que contiene los 104 pines de conexión que dan nombre al estándar. Detalles de las dimensiones entregadas por estándar PC/104 para el diseño de los PCBs se encuentran en el **¡Error! No se encuentra el origen de la referencia.**.

El principio fundamental del estándar PC/104 es el "apilamiento" o *stacking* de los PCBs utilizando una única línea de conectores continua entre estos, como una columna vertebral. Esto se ilustra en la Figura 20. Este principio es fundamentalmente diferente al otro estándar usualmente utilizado, el UNISEC o el *Backplane Board*, que actúa más como una placa madre tradicional en un computador de escritorio.

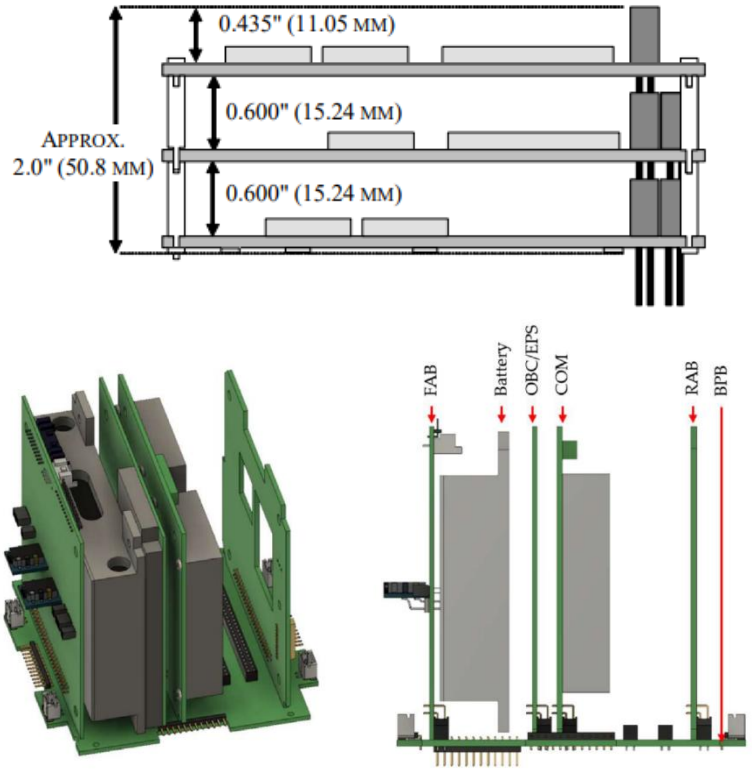


Figura 20: Dimensiones y principio de apilamiento de PCBs en el estándar PC/104 (Arriba). UNISEC Backplane Board (Abajo).

El estándar PC/104 encontró un nicho dentro de las aplicaciones satelitales, y especialmente en CubeSats, siendo uno de los métodos más usados para conectar tarjetas de circuitos en nanosatélites. Algunas empresas como GomSpace han desarrollado variaciones al estándar original del PC/104. Mientras que los conectores tradicionales del PC/104 utilizan el estándar asimétrico ISA de 16bits con 104 pines de conexión (con dos hileras de 32 pines cada una, y las otras dos de 20) (Figura 19), el estándar GomSpace utiliza también 104 pines de conexión apilables entre PCBs, pero utilizando 4 hileras simétricas de 26 pines cada una.

Dentro del estándar PC/104, y concordante con los estándares utilizados en diseño de CubeSats, existen 4 puntos de anclaje mecánicos, habitualmente en la forma de separadores o *spacers*. Estos se pueden apreciar como agujeros cercanos a las esquinas del PCB. Es importante notar que estos no son simétricos.

3.2.2 Interfaces lógicas

Interfaces lógicas no sólo definen qué elementos físicos representan interfaces entre subsistemas y elementos, pero también incluyen qué métodos y protocolos se utilizan en cada caso. En la Figura 21 se puede apreciar un esquema simplificado de los componentes, elementos y las interfaces lógicas requeridas.

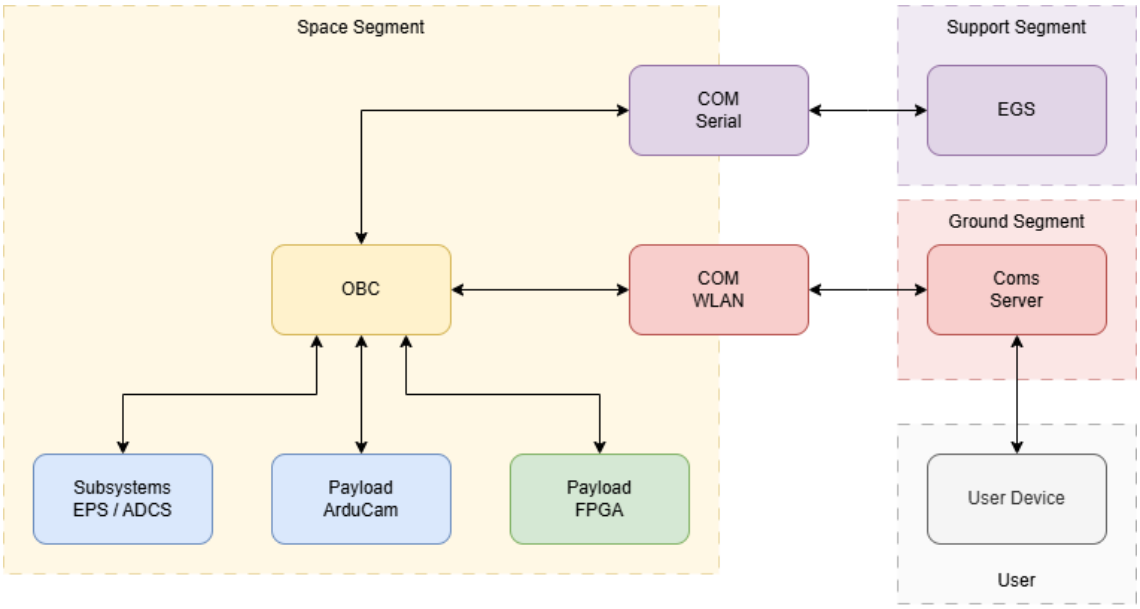


Figura 21: Arquitectura de sistema con flujos de información e interfaces lógicas.

Internamente, dentro del mismo CubeSat, se utiliza el Bus PC/104 como interfaz lógica, pero esto demanda una asignación de pines que se coordine no solo entre los subsistemas del bus, sino que también con el payload desarrollado por Nicolás Valderrama, y con cualquier dispositivo periférico que se integre al sistema (dígase, el EGS). Con este fin se ha desarrollado una planilla colaborativa para disponer de todos los pines asignados dentro del sistema (Figura 22).[4]

H1				H2			
1	JART0 TX (NV	2		1		2	
3	JART0 RX (NV	4		3		4	
5		6		5		6	
7		8		7		8	
9		10		9		10	
11		12		11		12	
13		14		13		14	
15		16		15		16	
17		18		17		18	
19		20		19		20	
21		22		21		22	
23	OW-DS18-O	24		23		24	
25	A-TEMT-O	26		25	5V-O	26	5V-O
27		28		27	3.3V-O	28	3.3V-O
29		30		29	GND-O	30	GND-O
31		32	5V-I (BAT)	31	AGND-O	32	
33		34		33		34	
35		36		35		36	
37		38		37		38	
39		40		39		40	
41	I2C_SDA-I/O	42		41	I2C_SDA-I/O	42	
43	I2C_SCL-I/O	44		43	I2C_SCL-I/O	44	
45		46		45	V_BAT-O	46	CAM-SCK
47	LUPO	48	LUPO	47		48	CAM-MOSI
49	LUPO	50	LUPO	49		50	CAM-MISO
51	LUPO	52	LUPO	51		52	CAM-CS

Figura 22: Pin layout para el bus PC/104 compartido.

[4] El documento fuente de este layout puede ser encontrado aquí: PIN Layout.xlsx.

3.2.2.1 Nota sobre I2C y comunicación serial entre microcontroladores

Para evitar conflictos de contención entre múltiples controladores, conflictos de dirección I2C, y en general simplificar el uso de este protocolo, se propone definir redes diferenciadas I2C entre el BUS y payload. Una alternativa para esto es definir el protocolo de comunicación serial UART como el principal medio de comunicación inter-microcontroladores, definiendo al OBC como el maestro, único receptor de los subsistemas, y emisor universal. De esta forma, el pin Rx del OBC se conecta al pin H1-1 del bus, y el pin Tx al H1-3. Al conectar un dispositivo que desee comunicarse de forma serial con el OBC, deberá conectar sus propios pines Tx a H1-1 y Rx a H1-3, asegurándose de instalar un diodo de conmutación (1n4148, sugerido) entre el Tx y H1-1, con el fin de evitar una señal invertida.

4 Hardware

4.1 Subsistemas electrónicos

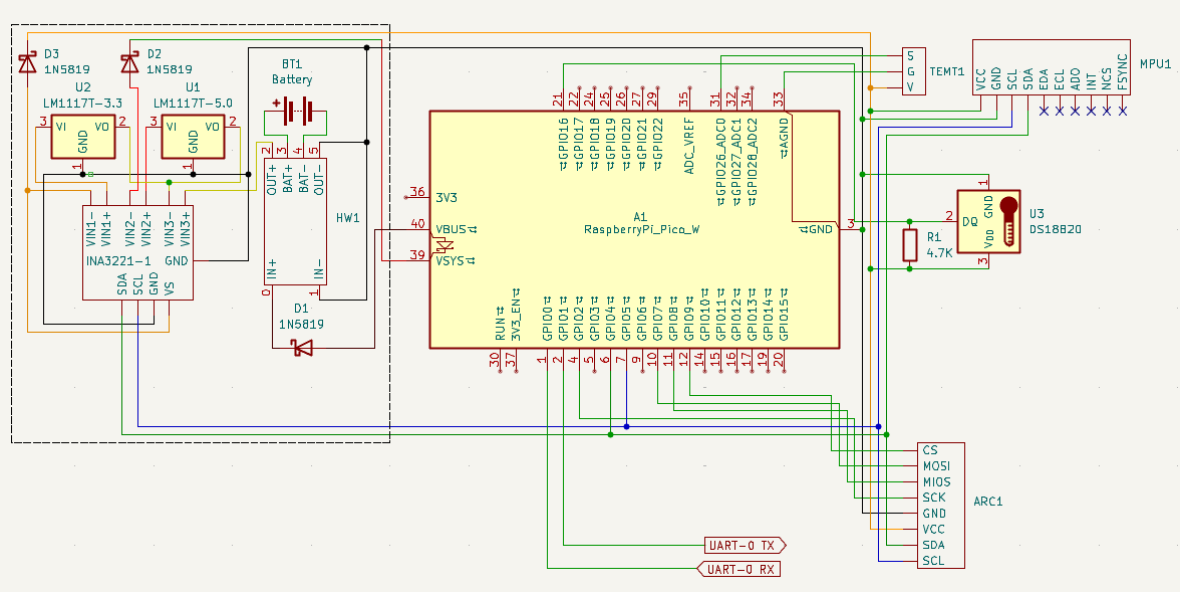


Figura 23: Diagrama de circuito general del bus, considerando todos los componentes de los 3 subsistemas electrónicos, pero sin mostrar el bus PC/104.

4.1.1 On-Board Computer (OBC)

La función principal de este subsistema es procesar y distribuir comandos y datos. También se le puede llamar el *Command & Data Handling Subsystem* o CDH. Además, debido a que la Raspberry Pi Pico W posee un módulo integrado wifi, y tomando lecciones aprendidas de proyectos anteriores respecto a la complejidad que representa integrar un módulo de comunicaciones, se ha asignado la función de comunicaciones a este componente.

Como fue mencionado anteriormente, el componente principal de este subsistema corresponde a la Raspberry Pi Pico W (Figura 24), un microcontrolador flexible, y que combina bajo costo, masa y consumo energético con una capacidad razonable de procesamiento, almacenamiento y comunicaciones.

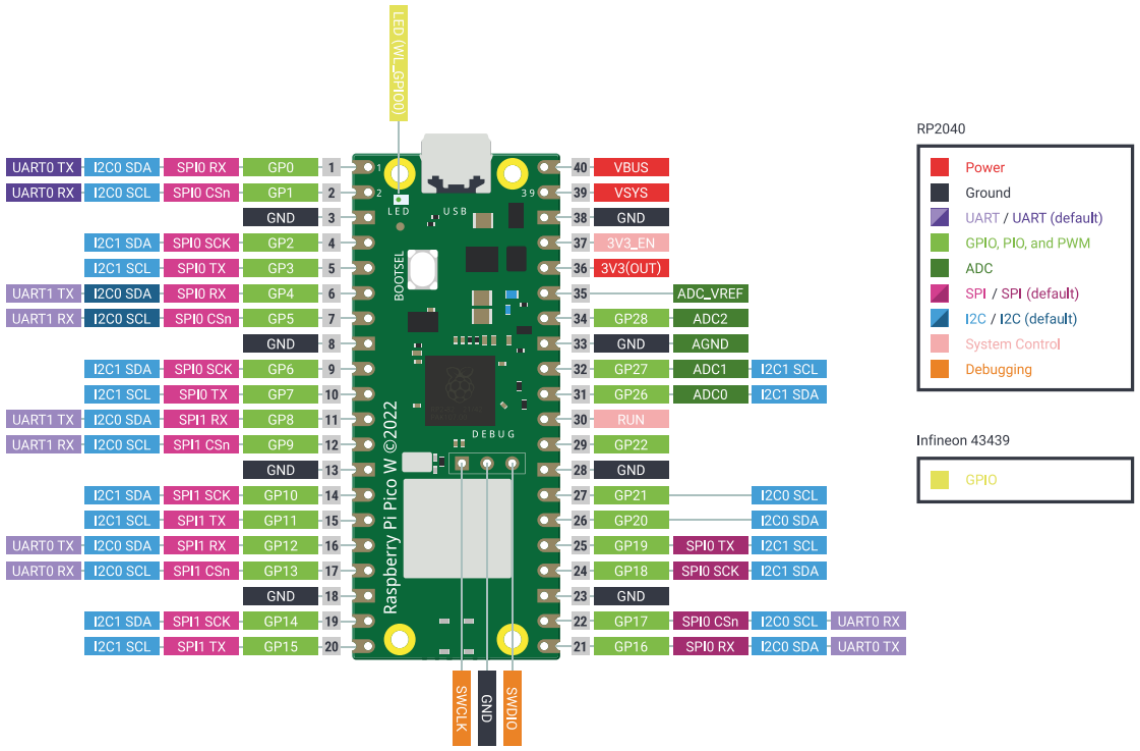


Figura 24: Pinout de la Raspberry Pi Pico W, con sus respectivas funciones soportadas por cada pin.

El OBC controlará todos los componentes conectados a este, ya sea por medio de protocolos seriales SPI o UART, o I2C. La idea es centralizar todos los procesos del satélite en este componente, permitiendo un mayor control de estos, y así facilitar y priorizar los protocolos de comunicación internos y externos. El detalle del software será descrito en el párrafo 5.1.1.

4.1.1.1 Alimentación, conexión serial y carga al EPS

Para poder realizar pruebas, alteraciones al software de vuelo o configuración avanzada, es necesario conectar la Raspberry a un computador a través de un cable micro-USB. Este permite comunicación serial con el ordenador, permitiendo acceder a la memoria flash, pero también provee alimentación de 5V a la Raspberry.

La alimentación por medio del cable USB está conectada a los pines de potencia 40 (VBUS) y 39 (VSYS), donde existe un circuito interno de protección entre el pin 40 y 39, incluyendo un diodo. Considerando que el diseño del EPS considera un controlador de carga para la batería que admite alimentación de 5V-1A, una idea propuesta es la de integrar la función de alimentación de la Raspberry y el circuito de carga de la bateria, pero solo cuando el conector USB esté en uso. La propuesta sencilla contempla conectar la salida de 5V desde el EPS al pin 39 (el cual deberá estar protegido por un diodo), alimentando solamente la Raspberry y sin salir al pin 40. En caso de estar alimentado por USB, la Raspberry proveerá una salida de 5V por el pin 40 hacia la entrada de 5V del EPS, permitiendo la carga de la batería en paralelo.

Otra ventaja de este método, es que permitiría que el EPS esté activo durante las pruebas realizadas por medio de esta conexión, permitiendo acceder a los sensores internos del EPS (como la INA3221).

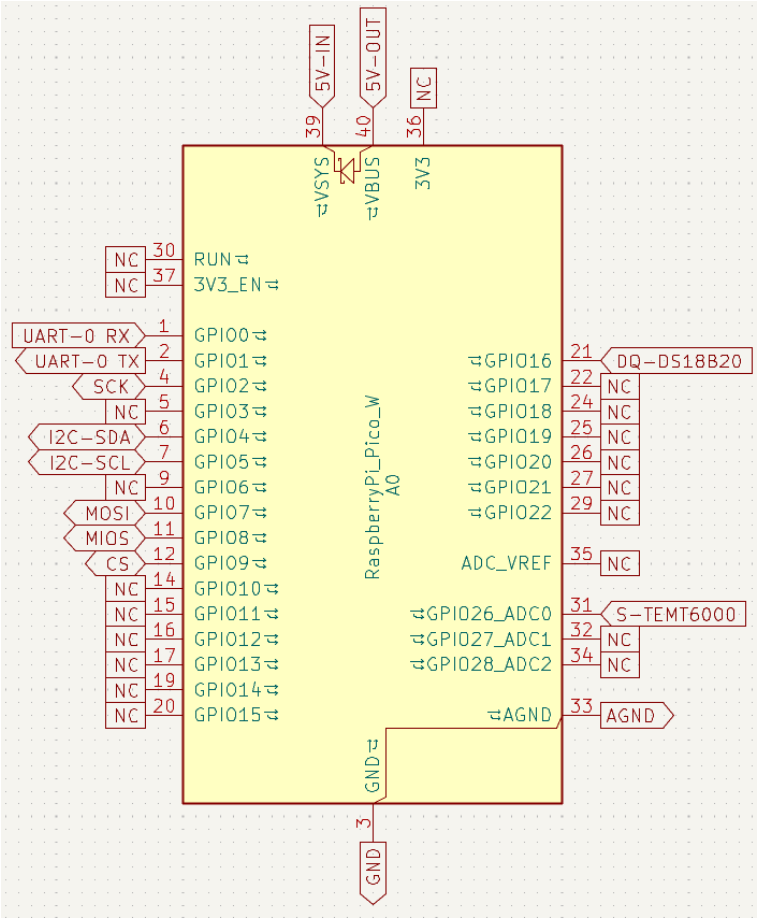


Figura 25: Diagrama de conexiones para el OBC con etiquetas de red.

Posibles desventajas de este método incluyen un riesgo de inestabilidad en la potencia entregada a la Raspberry cuando se use la batería, ya que la Raspberry estaría más vulnerable al estar alimentada desde el pin 39, en lugar del 40. Esto podría mitigarse usando un circuito de protección que debería ser diseñado, pero requeriría al menos dos tipos de capacitores, un diodo adicional y una pequeña resistencia.

La alternativa segura es simplemente conectar la salida de 5V del EPS al pin 40 por medio de un diodo, pero se debe estudiar la protección al circuito del EPS, y el acceso a los sensores, ya que no podrían ser alimentados por medio de esta conexión.

4.1.1.2 Problema detectado: peak de consumo wifi

Debido a un peak de consumo producto del uso del módulo wifi integrado, pruebas con este componente en una protoboard con alimentación por medio de los pines 39 o 40 resultan riesgosos. La Raspberry consume, en condiciones normales sin usar el módulo wifi, aproximadamente 100 mA. Al activar el módulo wifi, este consumo genera una alteración transiente en el sistema que puede exceder los 300 mA. Debido a limitaciones con los circuitos de conexión internos de una protoboard, estos pueden quemarse al superar los 200 mA, causando intermitencias y cortes en la operación nominal del OBC. Se cree que esto podría estar causando errores en la integración y testeo de otros componentes.

Debido a limitantes de tiempo y falta de pruebas en detalle, no se presentan datos y mediciones del proceso. Estas serán incluidas en futuras entregas. Se espera poder realizar pruebas adicionales sobre este microcontrolador sin depender de una protoboard, pero de momento ha sido una limitante.

Conclusión: Para hacer pruebas funcionales de componentes que requieran el uso del módulo wifi no se podrá hacer uso de una protoboard.

4.1.2 EPS

La función principal del EPS es la de generar, almacenar, acondicionar y distribuir la energía eléctrica necesaria para que el sistema opere. A nivel conceptual, este subsistema no presenta mayores cambios respecto a lo elaborado en PIA, pero en ejecución se han realizado algunas modificaciones, y análisis más detallado de componentes utilizados ha revelado algunos puntos de preocupación que deberían ser atendidos.

El diseño original del EPS se muestra en la Figura 26, donde se aprecian los componentes principales del subsistema.

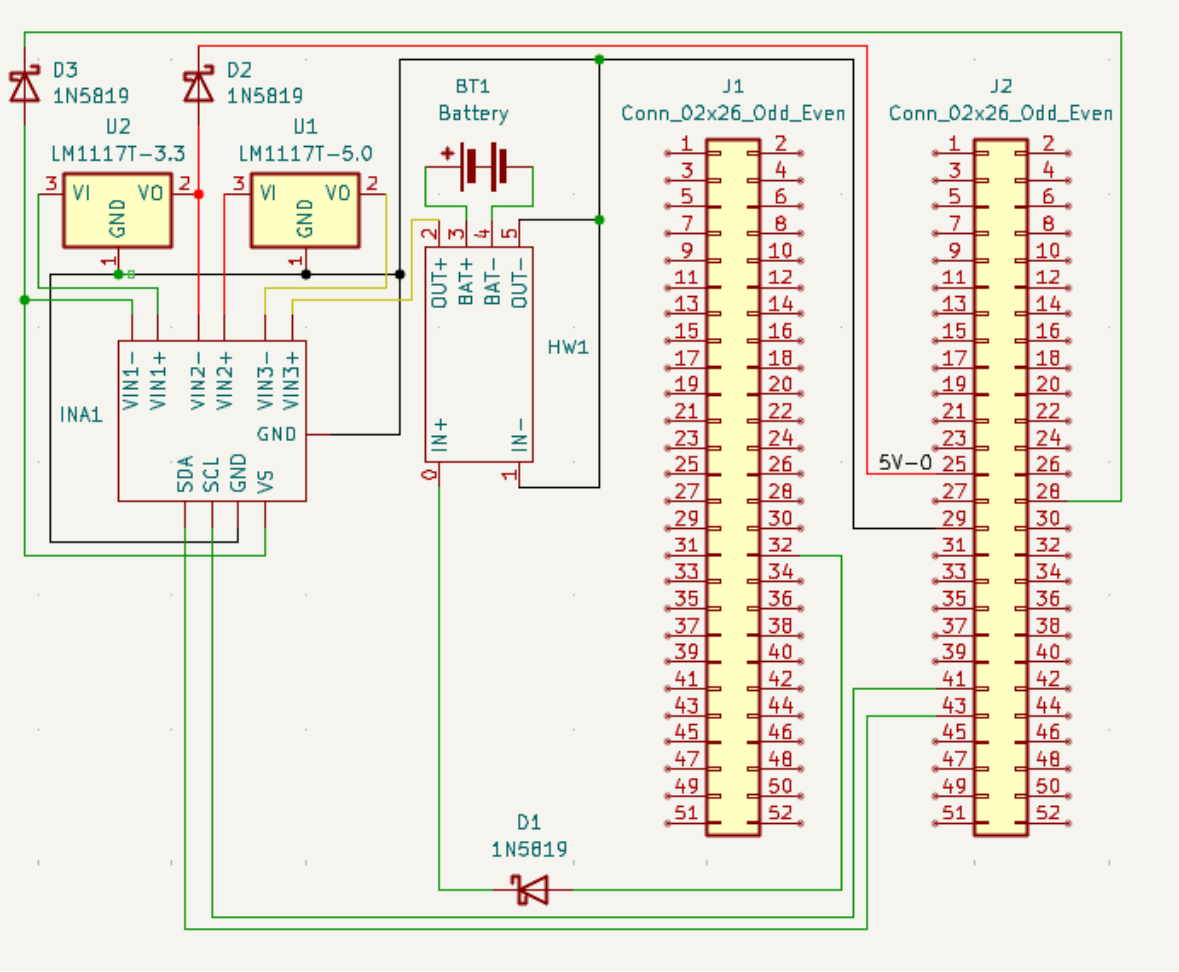


Figura 26: Primera versión del diagrama de circuito para el EPS.

La batería utilizada para este equipo es una batería de litio-ión 18650, con una capacidad estimada de 2000 mAh.

4.1.2.1 Regulador de carga de la batería

El módulo HW-373 es un regulador de carga para baterías de Li-Ion 18650, basado en el componente TP4056. Este incluye un circuito de protección para evitar sobrecarga y descarga, además de regular la carga y descarga de la batería, y admite conectar la batería en paralelo a la carga del equipo entregando un voltaje no regulado. Para cargar la batería dispone de dos opciones, se puede conectar a través de un circuito a una fuente regulada de 5V hasta 1A, o a través de un conector USB-C de 5V y hasta 1A. Crucialmente, este no fue diseñado para permitir la carga a través de paneles solares, requiriendo una corriente regulada de entrada.

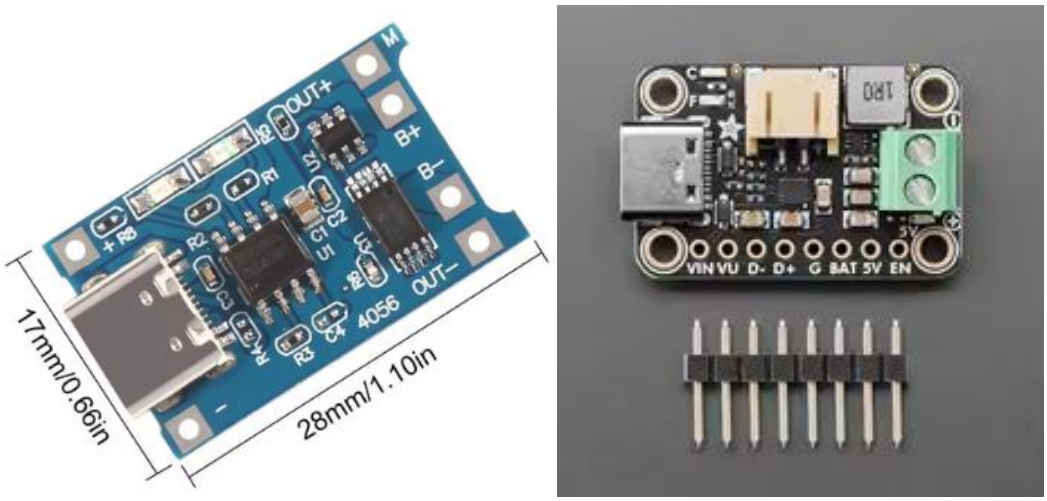


Figura 27: Módulo de carga de baterías HW-373 (izquierda), y módulo de carga Adafruit BQ25185 (derecha).

La alternativa más eficaz para permitir carga a través de un circuito, además de soportar carga a través de paneles solares, es el módulo Adafruit BQ25185. El problema es que el costo por unidad de este módulo es de, por lo menos, $7,000 CLP + Envío (aprox. $4,500 CLP), comparado con el valor del HW-373 que se reduce a un marginal $280 CLP, sin necesidad de pagar el envío. Siendo que es más de 40 veces más caro, no se consideró para esta implementación.

4.1.2.2 Panel Solar

Como resultado de la selección del módulo de carga, el panel solar no será incorporado en esta implementación del sistema, al menos no conectado eléctricamente al módulo de carga, ni al circuito de potencia. Se está considerando la implementación en un soporte estructural, además de incluir puntos de medición para obtener lecturas del voltaje y corriente generados.

4.1.2.3 Puntos de medición de corriente

Una de las características solicitadas para el diseño de este subsistema fue la de permitir la medición de voltajes y corrientes en distintos puntos para el desarrollo de la experiencia. En un principio, esto fue interpretado como leer la carga de las barras considerando corriente y voltaje, y por medio de un multímetro digital. Adicionalmente fue integrado un sensor de corriente y voltaje INA3221 para medir 3 canales: La salida de la batería, la salida de 5V del convertidor, y la salida de 3.3V del regulador.

Para permitir puntos de lectura de voltaje se propuso implementar puertos hembra de conectores banana en las tres líneas, además de un puerto común a tierra.

Como alternativa, se propone instalar sensores de corriente y voltaje INA219 para monitorear el consumo específico de potencia por cada placa de subsistema. Esto representaría utilizar 3 módulos INA219, los cuales se montarían físicamente en cada una de las placas, y estarían conectadas al bus I2C central. Estas permitirían entregar mediciones precisas del consumo de cada subsistema de forma constante, sin necesidad de realizar mediciones directamente a través de un multímetro.

4.1.3 Sensores y Payload

En total se presentan 3 sensores y una carga útil integrada definida como una cámara digital ArduCAM-2MP-Plus (Figura 30). Los tres sensores usados son:

1. Sensor de temperatura digital DALLAS DS18B20: Comunicado a través de protocolo OneWire.
2. Sensor de luminosidad analógico TEMT-6000: Corresponde a un fototransistor integrado con su divisor de voltaje, entregando una señal analógica.
3. Sensor MPU 9250: Módulo combinado de un IMU y un sensor giroscópico, comunicado a través de bus I2C.

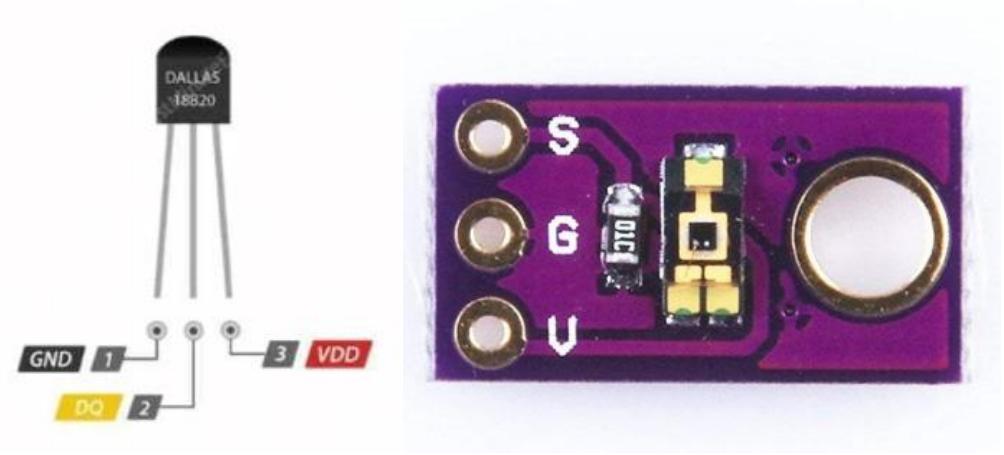


Figura 28: Sensores de temperatura DS18B20 (izquierda) y TEMT6000 (derecha).

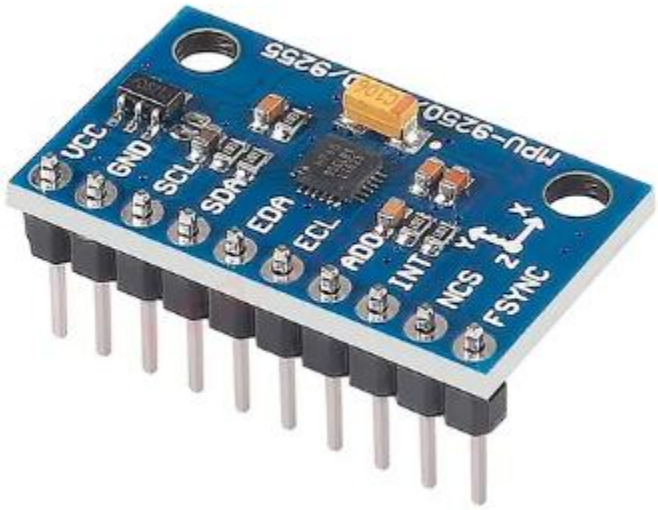


Figura 29: Sensor IMU/MPU 9250

Respecto a la cámara digital, este es un módulo integrado que incluye la cámara OV2640 de 2MP, un microcontrolador dedicado, y la capacidad de comunicarse por medio de los protocolos I2C y SPI para control y transmisión de datos. La principal ventaja de este módulo es que permite reducir la cantidad de pines necesarios para el control de la cámara de 18 (de la OV2640), a tan solo 8.

Una limitante de este módulo es que los controladores disponibles para su uso están solo disponibles en CircuitPython, y no en MicroPython, lo que representa un cambio respecto a la implementación del prototipo realizado en el PIA, donde se utilizó MicroPython.

Figura 30: Módulo ArduCAM-2MP-Plus, con OV2640 integrada.

4.2 Estructura

Originalmente se había propuesto realizar un CubeSat de 3 unidades, pero debido al reducido tamaño total del sistema, el resultado sería que 2 de estas unidades quedarían completamente vacías. Como resultado, se ha replanteado la configuración de la estructura para utilizar 1U en lugar de 3U.

Figura 31: Estructura de CubeSat provista por ISIS para 3U (Izquierda) y 1U (Derecha).

La estructura sigue el estándar CubeSat según ISIS, cuyos modelos estructurales están disponibles libremente en modelos 3D para trabajar en software CAD.

En lugar de utilizar estructuras de aluminio, busca trabajar con estructuras fáciles y comparativamente más baratas de iterar y prototipar por medio de impresión 3D en PLA. Con este fin se realizó una impresión 3D de la estructura de 1U de ISIS, comprobando que, aunque en apariencia se vea frágil, la estructura resulta sorprendentemente resistente. Un detalle para considerar para en el diseño, es que al migrar de una estructura metálica a una plástica los tornillos de apoyo no disponen de un hilo directo donde entrar, por lo que se considera el uso de insertos de bronce roscados. Además, en lugar de usar tornillos M2.5 para este soporte se plantea migrar todas las roscas a M3, y utilizar tornillos de este calibre para los apoyos. Esto, debido a que las barras de apoyo para las placas de subsistemas utilizan el diámetro M3, y se encuentran roscadas en los extremos.

Estas varillas estructurales se pueden conseguir como varillas de empuje, normalmente utilizadas como varillas de control en proyectos de modelismo aficionado. La ventaja que representan respecto a los espárragos tradicionales es que, al no estar roscadas en toda la extensión de la varilla, evitan que otros elementos estructurales como las placas de circuitos y los soportes estructurales no se atasquen y dañen al ser montados. Esto es particularmente útil cuando se trata de kits educativos cuyo principal uso implica el montaje y desmontaje reiterado de los elementos estructurales.

Adicionalmente, para permitir un espaciado concordante con el estándar PC/104, se prevé la fabricación de *spacers* en PLA, que simplemente son cilindros perforados a través del eje central, con una altura aproximada de 15.2 mm. Estos se instalarán en las esquinas de las placas de circuito, en las varillas estructurales.

5 Diseño preliminar de software y equipo de soporte

5.1 Software y Comunicaciones

5.1.1 Software de Vuelo

Dentro del sistema desarrollado hay varios elementos que requieren un software para operar correctamente. Entre los más críticos se encuentra el OBC, consistente principalmente de una Raspberry Pi Pico W, cuya función principal es la de entregar comandos, procesar datos, y permitir la comunicación con una plataforma externa. Entre las plataformas externas se definen una estación terrestre, consistente en un servidor de comunicaciones, las terminales de usuario, y el equipamiento de apoyo en tierra para facilitar los procesos de verificación. Un esquema simplificado de esta arquitectura se puede observar en la Figura 21.

A partir de esta definición, el punto crítico de diseño de software es el OBC. A este software se le denomina el *software de vuelo*. Algunas definiciones importantes respecto al software vienen dadas, en primer lugar, por la designación del hardware. Raspberry Pi Pico es un *micro**controlador***, a diferencia de otras distribuciones de Raspberry que pueden ser definidas como *micro**computadores***, la función de diseño del primero es permitir el control de otros dispositivos por medio de una arquitectura simplificada. El principal beneficio, y de especial interés para este sistema, es que esto permite un uso más optimizado de recursos por medio del *firmware* utilizado. En vez de implementar un sistema operativo (OS) más generalista, Pico utiliza una distribución de Python llamada *MicroPython*, o una variación de este conocida como *CircuitPython*. La elección entre estos dos depende principalmente de la capacidad que tengan para gestionar librerias, controladores y la versatilidad de estos.

Al trabajar con Raspberry Pi, los distintos scripts de software se almacenan en la memoria flash del controlador. Esto permite que, al momento de energizar el componente, estos scripts se ejecuten automáticamente. Raspberry no ejecuta todos los scripts en cualquier orden, sino que sigue la siguiente lógica: Al encender, si existe boot.py, este se ejecuta primero. Cuando termine boot.py, o en caso de que no exista, intentará ejecutar main.py (Figura 32). Esto permite tener dos scripts independientes, pero que pueden “heredar” información. Así, variables, funciones y clases de boot.py pueden ser importadas en main.py.

Figura 32: Flujo de software al energizar Raspberry Pi.

El software principal deberá cumplir 3 funciones generales:

1. Recibir y procesar comandos desde la estación terrestre
2. Realizar lectura y procesamiento de datos de sensores
3. Transmitir los datos a la estación terrestre

Debido a que, por limitaciones del software y las capacidades propias del procesador, no se puede hacer una implementación rápida y limpia de procesamiento multihilo[5], sólo se podrá ejecutar una de estas funciones cada vez. El diseño del software se concentra en establecer modos o fases de operación, con el fin de evitar problemas y conflictos. De esta forma, dentro de un paso de tiempo o ciclo se define que el OBC:

- Comenzará la primera fase "escuchando" si recibe comandos destinados a este equipo, y en caso de recibir uno, priorizará cumplir el comando antes de pasar a la siguiente fase.
- La segunda fase realizará una lectura de los sensores designados como activos dentro de la configuración del sistema. Algunos sensores requieren más tiempo o recursos de procesado, por lo que no todos podrán ser leídos al mismo tiempo. Los más críticos serían: La cámara digital, el sensor MPU, y el sensor DS18B20.
- La tercera fase realizará la transmisión de datos. Dependiendo de la cantidad y tamaño de estos se deberá establecer dentro del protocolo de modo cuál será el método, pero si es telemetería básica el protocolo base MQTT bastaría.

La previsión es que habrá al menos 3 modos de operación:

1. Standby: No se transmite información, pero se recopilará información designada como crítica.
2. Normal: Recopila datos de telemetría de todos los sensores regulares, salvo la cámara digital, y se transmiten a tierra cada ciclo.
3. Captura: El ciclo tomará una captura de imagen con la cámara y la enviará a tierra. El tiempo de este ciclo debe ser definido, al igual que el método de transmisión. No está claro si el protocolo MQTT es suficientemente robusto como para soportar transmitir imágenes. Durante este modo el OBC tratará de recopilar datos críticos durante el ciclo, pero no los transmitirá inmediatamente.

Con el fin de permitir recopilar más información respecto el funcionamiento interno del equipo, se propone implementar un sistema de registro (logs) que permita almacenar información de debug durante cada ciclo.

5.2 Electric Ground Support Equipment (EGS)

El EGS cumple dos funciones principales:

1. Facilitar los procesos de aceptación y verificación respecto interfaces lógicas del bus.
2. Simplificar y facilitar el proceso de conexión a la red wifi de la sala de clases, sin necesidad de tener que reprogramar cada uno de los OBC.

Para cumplir estas funciones, el EGS necesita poder conectarse al bus PC/104, leer datos desde el OBC y otros subsistemas, y ser capaz de interactuar con los OBC de forma inalámbrica.

[5] El multihilo o *multithreading* es una técnica que permite ejecutar operaciones o "hilos" en paralelo, tomando provecho de múltiples núcleos de procesador, o a través de software especializado. La mayoría de los sistemas operativos soporta esta característica y permite mejorar la experiencia de usuario, pero en el caso de un microcontrolador esta capacidad no está soportada de forma nativa y su implementación está lejos de ser trivial.

Debido a la versatilidad ofrecida, la capacidad de interactuar con un módulo wifi, y el relativo bajo costo que representa, una Raspberry Pi Zero W con un circuito y conector PC/104 dedicado sería ideal para esta función.

Figura 33: Raspberry Pi Zero W.

Esta se puede programar con al menos dos scripts, el primero permitiría leer directamente los puertos de comunicación serial (UART/SPI), I2C, y similares del bus, entregando datos de telemetría y salud directamente a un computador, display, LED o cualquier otro medio de verificación visual para comprobar lecturas dentro de rangos aceptables.

El segundo script utilizaría las funciones del módulo integrado wifi para generar una conexión de descubrimiento UDP visible a todos los dispositivos cercanos, entregando información sobre la red wifi utilizada, credenciales de acceso, y la IP del broker MQTT. De esta forma, cuando los OBC se enciendan, primero buscarán la red del EGS, recibirán la información de la conexión wifi, y desde ahí procederán a conectarse regularmente al MQTT.

6 Implementación del prototipo

El proceso de implementación comienza por el diseño de los componentes fundacionales. Considerando que dos de los requerimientos más restrictivos respecto al diseño del sistema corresponden tanto a la especificación del CDS, como el uso del estándar PC/104 (Gomspace), estos fueron los primeros en ser tomados en consideración. En otras palabras, el proceso de diseño siguió más o menos el mismo orden lógico que un proceso de integración: de adentro hacia afuera y de abajo hacia arriba. De esta forma, lo primero en ser diseñado fue la placa de circuito impresa (PCB) básica de todos los subsistemas al interior del CubeSat.

6.1 Diseño de PCBs

El diseño básico del PCB fue realizado en conjunto con Nicolás Valderrama, tomando inspiración directa de una PCB utilizada por Gomspace, en particular respecto a la posición de los conectores Board-to-Board o BTB, las perforaciones utilizadas por las varillas de guiado, y la forma general del PCB (Figura 34). Para la realización de este diseño se utilizó el módulo de diseño de componentes electrónicos incluido en el software Autodesk Fusion.

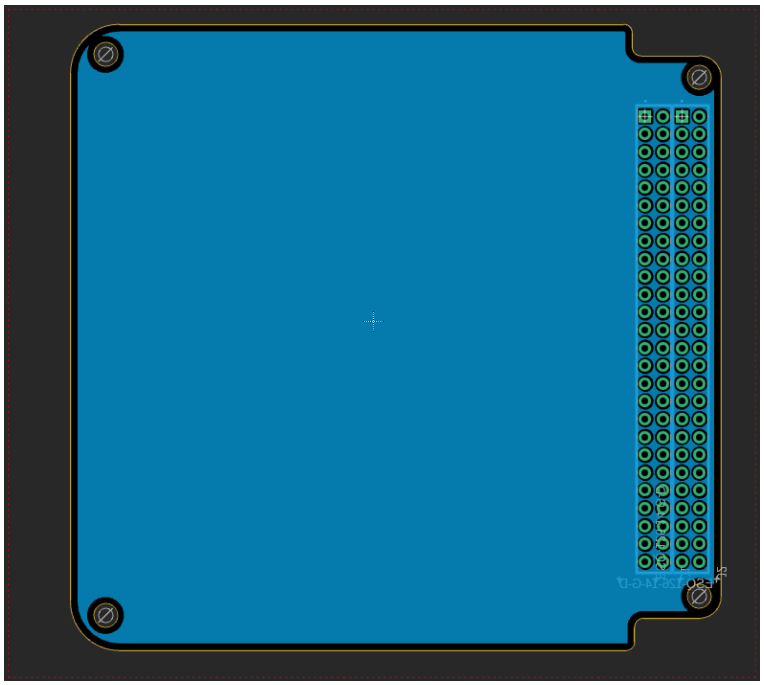


Figura 34: PCB básico y en blanco diseñado para el prototipo.

Una complicación mayor encontrada durante el desarrollo de este proyecto fue la notable inexperiencia en diseño de circuitos, uso de la herramienta de diseño de PCBs de Fusion (y de otros softwares afines) y el desarrollo poco ortodoxo de los componentes electrónicos diseñados. Un ejemplo de esto corresponde a conocimiento respecto de las reglas de diseño de circuitos. Estos definen varias directivas respecto a la forma en que se implementan distintos elementos dentro de un PCB, tales como el ancho de los canales de cobre (las rutas), los diámetros de perforaciones o cortes, reglas respecto a las vías, y quizás más crítico aun, la distancia mínima entre elementos de la PCB.

6.2 Diseño y fabricación del EPS

Las funciones del EPS se definen anteriormente como: almacenar, acondicionar, distribuir y generar energía eléctrica para el sistema. Tomando en cuenta estas funciones, el procedimiento comenzó por el diseño de la batería. A partir de las lecciones aprendidas durante el PIA, efectivamente siendo la iteración de este proyecto. La idea general de esta iteración es la de mantener gran parte de la arquitectura de EPS del proyecto anterior, pero intentando iterar sobre las falencias percibidas de la implementación anterior. Estas serían:

1. Implementar la iteración de prototipo siguiendo el factor de forma CubeSat y PC/104, buscando una compatibilidad con equipamiento de GomSpace, para asegurar entrenamiento en factor de forma más fidedigno posible a las prácticas vigentes en la industria. Esto se realizaría por medio del uso de una PCB en especificación, lo que en términos más prácticos significa la inclusión de conectores BTB en una configuración determinada, la disposición de estos en la PCB, y los puntos de interfaz mecánica entre la PCB y los elementos estructurales del PC/104 y CubeSat.
2. Cumplir la función de generar energía del EPS, de acuerdo con la descomposición funcional del sistema. Esto se buscaría implementar por medio de actualizar componentes en el circuito

de potencial del EPS previo, con el fin de permitir un generador solar integrado en la línea de potencia principal.

3. Considerar la ubicación de puntos de lectura para mediciones eléctricas durante la experiencia, que sean fáciles de indicar, y de acceder durante el proceso de integración y verificación.
4. Simplificar la lectura de potencia para el microcontrolador. Anteriormente se habían utilizado 3 sensores INA-219 para tomar lectura de potencia en las 3 líneas de potencia de interés: La línea de la batería, la barra de 5V y la barra de 3.3V. Esto permite verificar que el voltaje sea el correcto en cada canal, además de leer la corriente (A) que permite calcular el consumo de cada una de las barras, y de la batería, lo que permite verificar sus funciones internamente.
5. Permitir un medio de encender y apagar el EPS una vez ensamblado, para fines de seguridad y conveniencia. Además, evitar que ningún elemento quede soldado que pueda requerir recambio por algún defecto, o que requiera pruebas en aislamiento como debugs o evaluación independiente.
6. Mantener medidas de seguridad en torno a la batería, además de problemas por backflow de corriente en componentes que puedan ser sensibles. Esto por medio de ser más cauteloso al integrar elementos adicionales, incluir el uso de rectificadores de corriente, y realizar una conexión más limpia entre componentes. Finalmente, buscar un soporte de batería que permita una remoción más rápida de la batería manteniendo un agarre seguro.

Además, el diagrama general de circuito (para el EPS) durante la iteración anterior es el siguiente:

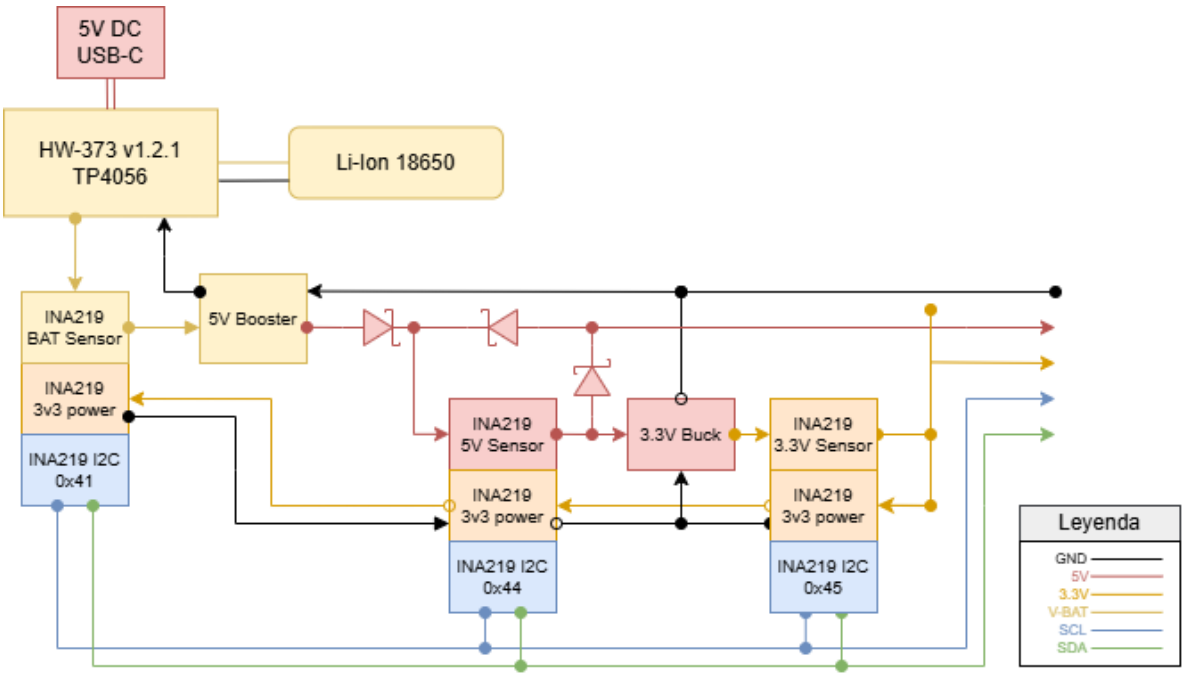


Figura 35: Diagrama de interfaces eléctricas y lógicas del EPS desarrollado en PIA

De esta forma, lo que se busca implementar es reemplazar los 3 sensores INA-219 por un único sensor INA-3221 de 3 canales, simplificando la integración mecánica del subsistema (al ser ensamblado durante una experiencia), además de liberar 2 direcciones I2C de este bus que podrían ser usadas por

otros sensores INA-219 en otras PCBs del sistema. También se busca hacer más segura la barra de 5V por medio de un enfoque en la interfaz de alimentación del OBC por el EPS, debido principalmente a la posibilidad de que la corriente se invierta en caso de proveer alimentación o una conexión externa al OBC, pudiendo causar problemas por corriente invertidas en componentes, incluyendo la batería. Otros dos enfoques funcionales que se consideraron fue la capacidad de alternar estados de encendido y apagado, y evaluar el posible recambio del convertidor de 5V anterior (Figura 36), el cual había levantado preocupaciones por ser poco confiable durante el proceso de integración pasado, ligado a problemas de sobrecalentamiento y sobrecarga, además de requerir la configuración manual del regulador por medio de retirar dos resistencias soldadas al componente. El resultado fue la primera iteración de diseño (Figura 37).

Figura 36: Regulador de voltaje configurable usado para la barra de 5V del prototipo desarrollado en PIA. Las resistencias indicadas como A y B se deben retirar de acuerdo con la configuración deseada. Para 5V de salida, se deben quitar ambas resistencias.

6.2.1 Primera iteración

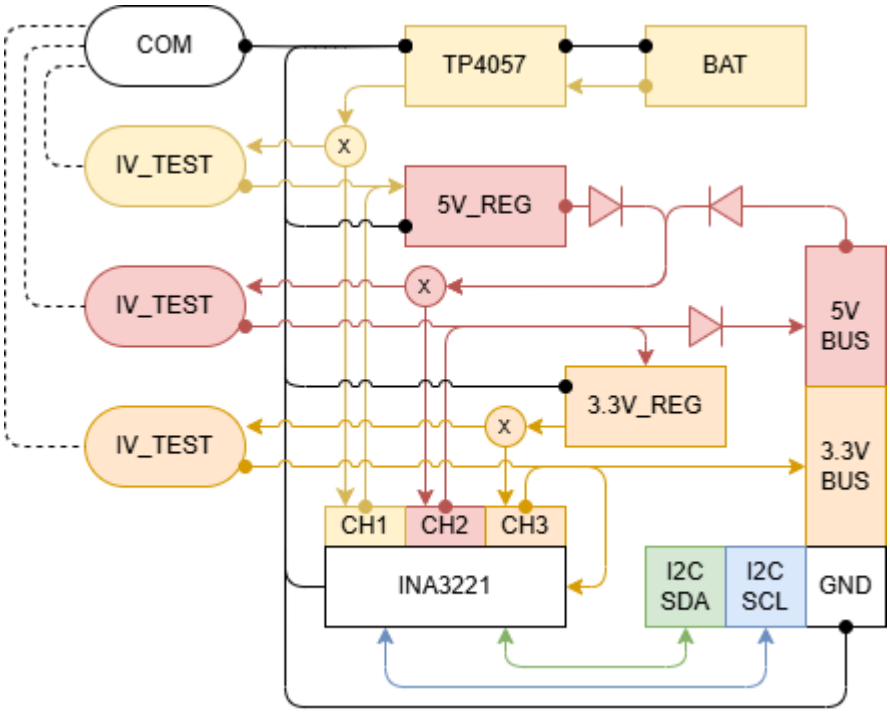


Figura 37: Diagrama de primera iteración de diseño sobre el EPS

En un principio, el diseño del EPS se enfocó en permitir una lectura de puntos de medición eléctrica de interés por medio de un multímetro digital durante la operación del satélite. Para facilitar esto, se buscó una alternativa para presentar puntos de lectura externos facilitados por conectores físicos presentes en la estructura del satélite, y en la forma de conectores de banana hembra conectados sobre el PCB mismo. Esto permitía una lectura desde el exterior, incluso en situaciones donde el EPS estuviera completamente integrado al resto del CubeSat, y que funcione en conjunto con un sensor INA-3221 capaz de medir los 3 canales a la vez. Respecto a la parte física del circuito, el diseño de este se enfocó en primer lugar en asegurar que el espacio disponible en el satélite permitiera una adecuada instalación. En otras palabras, no diseñar eléctricamente con componentes que no quepan dentro del PCB/CubeSat. Esto se consiguió diseñando un modelo 3D del ensamble del EPS, asumiendo las limitaciones impuestas por los factores de forma requeridos.

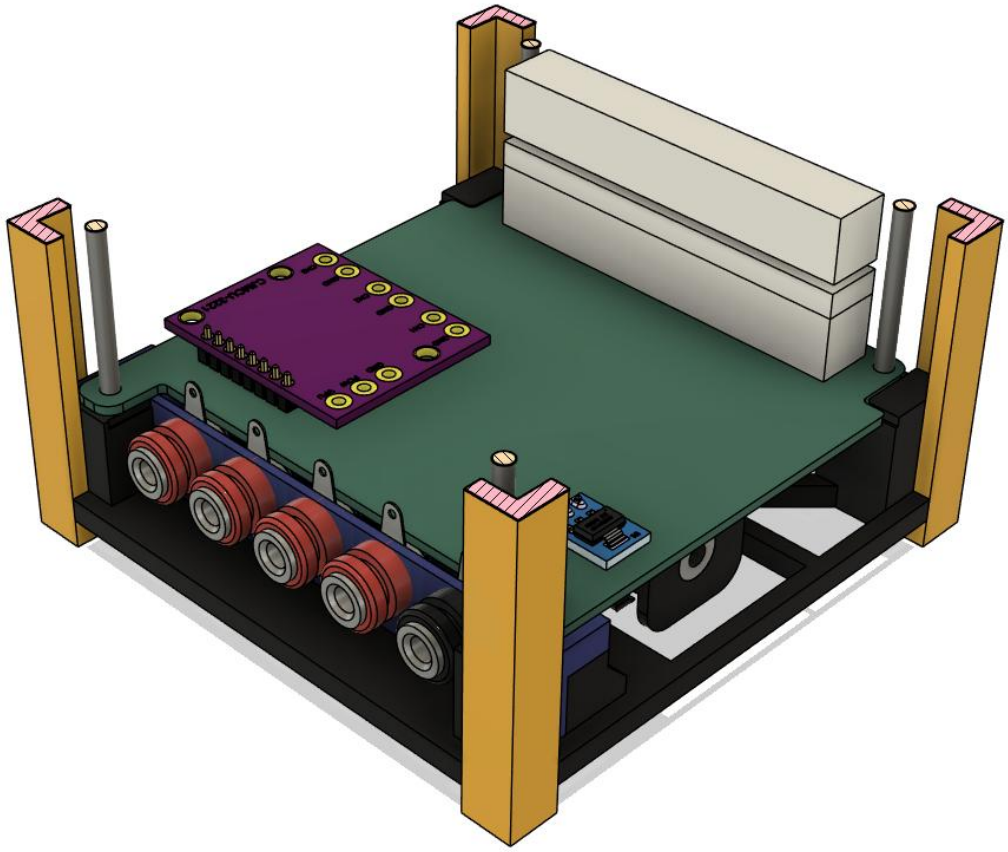


Figura 38: Modelo de primera iteración del EPS en Fusion.

En la Figura 38 se aprecia una aproximación en 3D de los componentes principales del EPS y su interacción con partes del subsistema estructural. Algunos componentes fueron obviados de la representación debido a que no eran significativamente grandes como para representar un riesgo creíble de exceder el espacio disponible. Se puede apreciar la implementación de 5 conectores banana hembra hacia el frente del conjunto. La instalación segura de estos se debía suplementar con una pieza auxiliar que permita un soporte para la manipulación de los conectores. En este punto del diseño, aun no se consideró el método de fabricación del PCB, pero la impresión por ambos lados, y por ende la instalación de componentes por ambos lados, eran las suposiciones de diseño vigentes.

A partir de este desarrollo se tomaron las siguientes consideraciones:

1. Se obtuvo la aclaración de que la medición física del EPS no era una necesidad tan urgente como se asumía en un principio, siendo la verdadera necesidad la de poder rastrear en tiempo real el consumo de poder del sistema. Esta función ya estaba cubierta durante el diseño por el sensor INA-3221, el cual supera este estándar con la capacidad de lectura continua y la posibilidad de realizar un registro automatizado en el OBC. Lo que nos lleva a la siguiente:
2. La remoción de los puntos de lectura físicos debido a que no son prioritarios en este momento.
3. La necesidad de presentar una orden de trabajo para el PCB con prioridad respecto al resto del diseño. Esto debido a que los tiempos de fabricación de PCB pueden ser largos y serían un cuello de botella al proyecto. Por lo tanto:

4. La prioridad de presentar un diseño de PCB rápidamente, requiriendo claridad respecto al diseño del circuito eléctrico, el diseño físico y confirmación de que sean funcionales en la práctica.
5. Tener más preocupación respecto al diseño estructural. En conjunto con el diseño de EPS, el diseño de los otros dos subsistemas principales se realizaba en paralelo, por lo que las nuevas indicaciones para el diseño estructural fueron incorporadas en la siguiente iteración.
6. La posibilidad de realizar una fabricación alternativa de PCB utilizando la fresadora CNC del LTA. Esto limitaría el diseño a unas reglas de diseño más restrictivas y a tan solo una capa de cobre en el PCB, pero un ciclo de trabajo mucho mas rápido para un prototipo.

6.2.2 Rediseño continuo

Para alcanzar la siguiente iteración se debieron avanzar en varias áreas simultáneamente. El rediseño del EPS fue una de las autodefinidas prioridades respecto a las lecciones aprendidas del PIA. Estas fueron evaluadas en iteraciones rápidas y coordinadas con cambios de diseño en otros elementos y subsistemas, especialmente el sistema estructural y OBC.

El primer paso fue determinar exactamente qué componentes formarían parte del EPS. Los elementos que considerar fueron el regulador de 5V, el módulo de carga de baterías y la incorporación de un panel solar funcional. Comenzando con estos últimos dos, la carga de baterías de litio con celdas fotovoltaicas no es una solución trivial, y requiere de ciertas consideraciones de diseño como medidas de seguridad.

Comenzando con el soporte de baterías, la solución durante la iteración pasada del EPS fue el uso de un soporte de plástico para una única batería 18650 mostrada en la Figura 39, izquierda. La instalación de baterías en este soporte es sencilla, pero poder retirarlas resulta moderadamente incómodo por culpa de las aletas de soporte a ambos lados. Además, la conexión por medio de cables resulta ser incómoda y poco elegante al ser integrada en un PCB. Por esto se optó por el soporte de baterías único mostrado a la derecha de la misma figura, la que permite desde una única ranura, sin mayor esfuerzo mecánico poder insertar y retirar la batería, además de disponer de terminales metálicas a ambos extremos en lugar de conexiones por cable, las cuales son más fáciles de integrar directamente al PCB.

Figura 39: Soportes para baterías 18650. Izquierda: Soporte de plástico con aletas y conexión por cable. Derecha: Soporte en esquema abierto y con contactos en las terminales.

6.2.2.1 Carga solar de baterías

Generalmente se recomienda el uso de un módulo dedicado que regule la carga y descarga de la batería. El módulo regulador HW-373, basado en el chip TP-4056, permite una entrada de carga por conexión al PCB, que teóricamente podría usarse para cargar la batería con una fuente externa. El problema es que este chip no regula la entrada de un panel solar, y debido a limitantes de diseño por tamaño, instalación y conveniencia, los paneles solares no podrían cumplir con el voltaje mínimo requerido por el chip para permitir la carga de las baterías, y además podrían bloquear la descarga de esta hacia el resto del sistema.

Una solución que se buscó fue la integración de un chip regulador de carga solar. Para esto existen dos alternativas principales:

1. Utilizar un módulo dedicado para regular la tensión de carga solar antes de entrar al módulo de carga TP-4056. Esto permitiría una tensión regulada, lo que permitiría un control más limpio de la carga solar. En este caso, la configuración de los componentes sería en serie desde el panel solar, al regulador solar, luego al regulador de carga y finalmente a la batería o al resto del sistema. Esta alternativa resulta ser inestable debido a que los reguladores solares normalmente están diseñados para operar únicamente como regulador de carga solar, habitualmente considerando múltiples celdas fotovoltaicas y baterías, pero sin permitir compatibilidad entre otros métodos de carga alternativos.
2. Utilizar un módulo integrado que combine las capacidades de carga solar, además de dos métodos de carga adicionales: por cable directo (USB-C), y por alimentación desde el OBC. Algunas alternativas existen, principalmente en torno al módulo TP-4057, pero estos componentes suelen ser más costosos de implementar que una solución simple en torno al módulo TP-4056.

De esta forma, la mejor alternativa evaluada fue la del módulo de carga universal Adafruit BQ-24074 (Figura 40), el cual permite los tres métodos de carga de la batería, y entrega una tensión limitada hasta un tope de 4.4V, lo que permite la implementación segura de reguladores de 3.3V y booster de 5V, además de permitir una corriente máxima de carga y descarga de hasta 1.5A (contra el 1.0A del TP-4056 del HW-373). Las desventajas de este componente son el mayor costo de compra (disponible únicamente a través de la página de Adafruit por $14.95, sin contar envío), y la necesidad de conectar los paneles por medio de un Jack DC de 2.1mm. En general, las ventajas de las medidas de seguridad del componente, además de la flexibilidad que permite en su operación, la hacen una alternativa particularmente buena.

A partir de una discusión con el profesor guía, se optó por no complicar el mecanismo de carga del EPS, sobre todo considerando que la idea de los paneles solares es simplemente ser demostrativa y no necesariamente funcional para cargar las baterías (las baterías se buscan cargar con antelación a la experiencia, y se dispone de cargadores externos para este fin). Esto, sumado al costo adicional respecto a la HW-373, y el costo de rediseño del EPS llevaron a la decisión de continuar usando el módulo HW-373 sin carga solar.

El cambio que se aplicó consiste en simplemente acotar los métodos de carga para fines de este prototipo, aunque conceptualmente se plantea para la siguiente iteración.

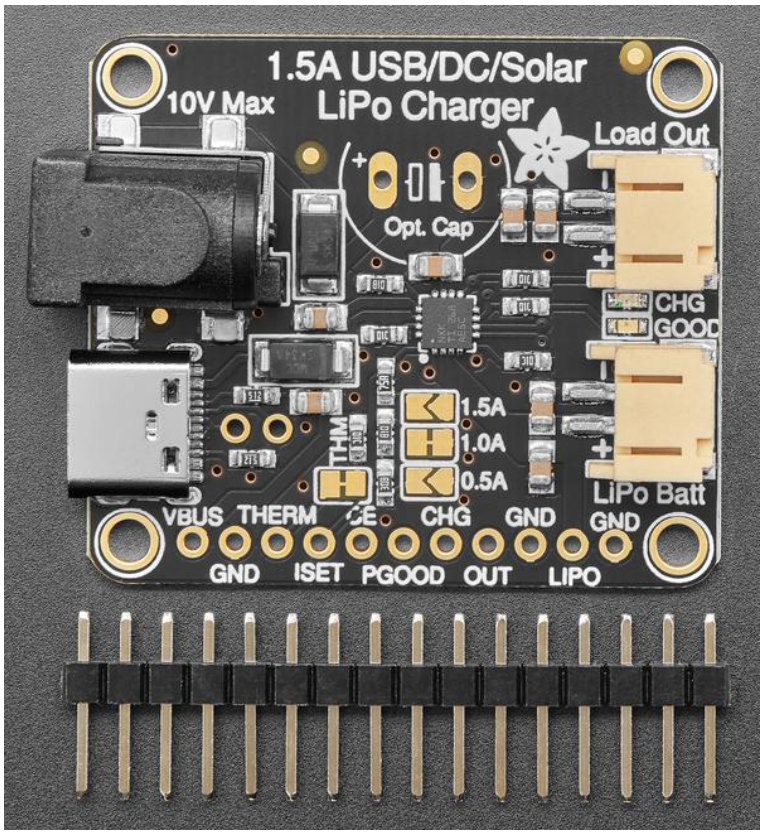


Figura 40: Módulo de carga universal Adafruit BQ-24074.

6.2.2.2 Stack de regulación para las barras de potencia

El EPS debe entregar dos barras de tensión regulada para el uso por otros subsistema del CubeSat, esto debido a la limitante del OBC de que debe ser alimentado con una fuente de 5V, pero los pines GPIO (de control digital o analógico) sólo pueden recibir hasta 3.3V, por lo tanto se optó por realizar una distinción de uso para las barras de 5V para únicamente el OBC o módulos microcontroladores Raspberry Pi, y la barra de 3.3V para sensores y otros componentes de uso general.

Siguiendo con la idea de buscar un reemplazo para el regulador de voltaje de 5V usado en la iteración de prototipo anterior (Figura 36), se evaluó la compra de un regulador más simple, menos costoso y fácil de implementar en el diseño. De esta manera se encontró el regulador de 5V mostrado en la Figura 41.

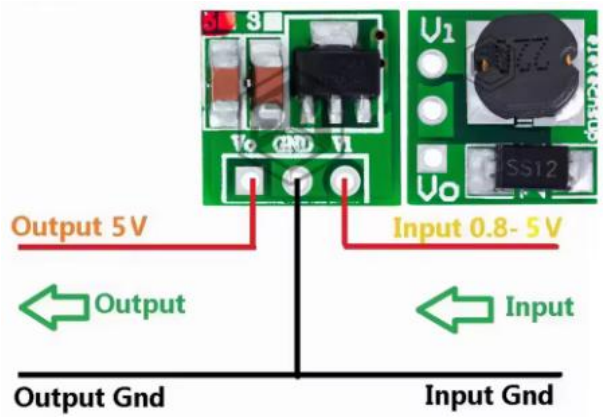


Figura 41: Regulador booster de 5V utilizado.

Durante el proceso de diseño se identificó una limitante a este componente en la forma de la capacidad de corriente máxima para la conversión de tensión, además de la limitante operacional de que debe tener una tensión de entrada mínima de 3V, pero con una corriente máxima de 380 a 480 mA. Esto representa la mitad de la capacidad del regulador de 5V utilizado anteriormente, pero debido a que no se esperaba aumentar considerablemente la carga sobre el EPS, se decidió implementar una arquitectura donde ambas barras utilizaran un booster de 5V como primer paso del stack (esquema de stack paralelos, Figura 42). En el caso de la barra de 3.3V, esto se justifica debido a que el regulador usado en esta barra es un reductor o Buck de 3.3V, lo que significa que la tensión mínima de entrada es de 3.3V.

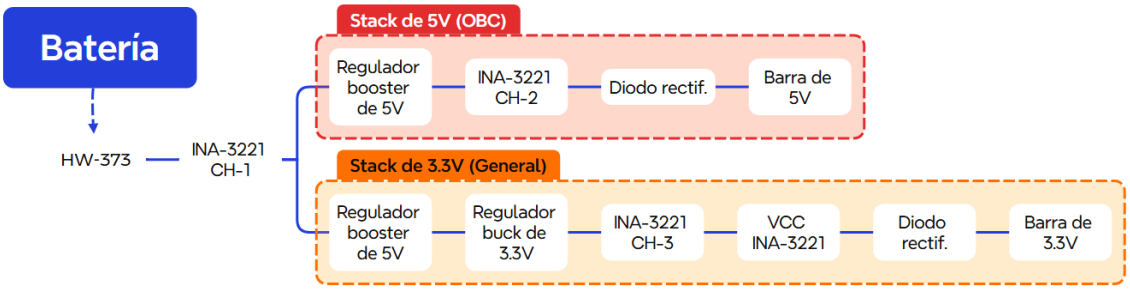


Figura 42: Esquema de stack paralelos para ambas barras de alimentación

Debido a que la batería 18650 cambia su tensión efectiva entre 4.2V y 2.5V durante el proceso de descarga (aunque la tensión nominal sea de 3.7V), esto causa que cuando la batería se descargue por debajo del umbral de los 3.3V, el regulador de esta barra solo podría entregar el mismo voltaje de entrada. Esto se ilustra en la Figura 43, donde se aprecia la tensión de entrada y salida del regulador.

Como un detalle importante, el regulador de bajada de tensión genera calor para reducir la tensión de salida. A mayor tensión de entrada (y corriente), mayor generación de calor del componente. Esto puede causar un accidente si no se incluye un método para disipar el calor de forma segura, lo que junto al calor generado por los reguladores de 5V y el regulador de carga, podrían justificar la implementación de un subsistema de control térmico (TCS).

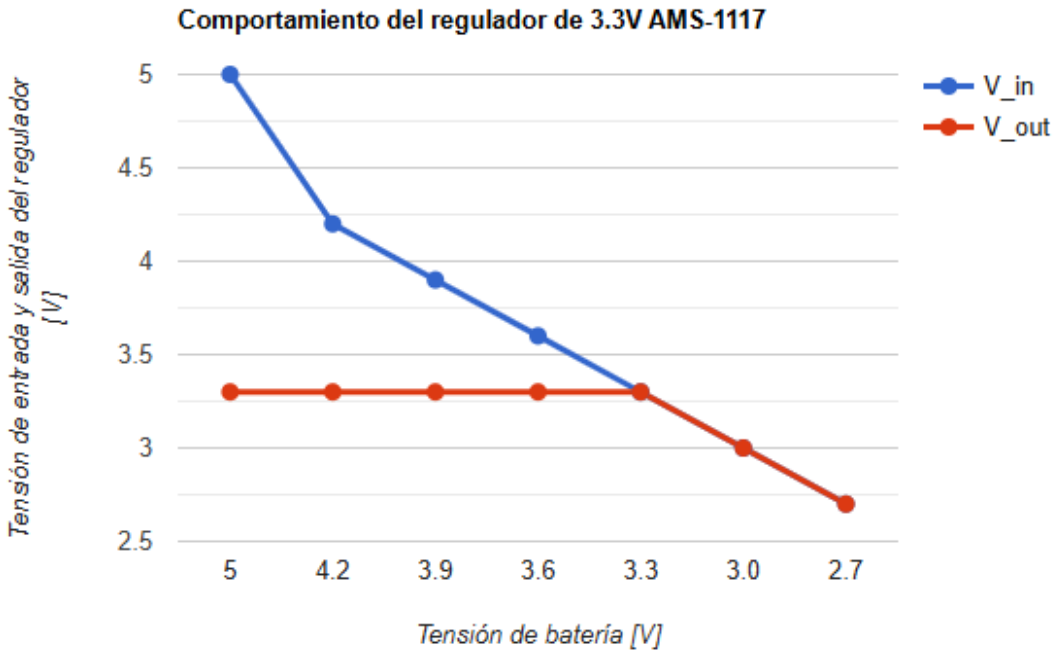


Figura 43: Comportamiento del regulador de 3.3V

Finalmente, la implementación de estos dos stack se protege de ser retroalimentado por el resto del sistema utilizando diodos rectificadores Schottky al final de cada stack, justo antes de la conexión al bus BTB. Los diodos Schottky son rectificadores de corriente utilizados para forzar el flujo de corriente en una sola dirección, muy similares a una válvula unidireccional en una analogía hidráulica. Un diagrama de referencia se puede apreciar en la Figura 44. El modelo más común de encontrar corresponde al diodo 1n5819 cuya tensión de diseño es de hasta 40V con una caída de voltaje de 0.6V a 0.9V (la tensión mínima que debe presentar el diodo para permitir una corriente en la dirección de transmisión). Mientras que el diodo con el que se diseñó corresponde al 1n5817, cuya tensión de diseño es de 20V, con una caída de tensión de entre 0.45V y 0.75V. La decisión de diseñar con este componente es debido a que el techo operacional de la tensión es menor, al igual que la caída de voltaje mínima requerida. En términos operativos, durante el desarrollo de esta iteración se determinó que las diferencias entre estos dos componentes no son lo suficientemente significativas como para limitar el diseño al 1n5817, sino que el 1n5819 puede ser utilizado indistintamente.

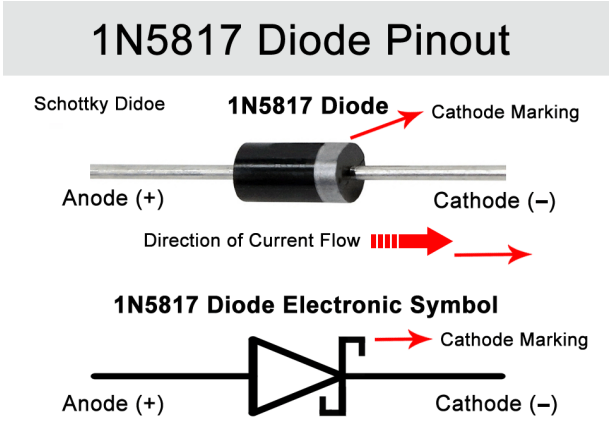


Figura 44: Diodo Rectificado Schottky utilizado como elemento de protección del EPS.

6.2.2.3 Diseño e implementación de PCB

Para el diseño del PCB se comenzó el diseño del diagrama de circuito, considerando todos los elementos ya definidos. Para este fin se utilizó el software KiCad como una primera aproximación de diseño. Esto debido a que la creación de librerías o componentes personalizados es mucho más rápida que en Fusion, y lo que se buscaba con mayor prioridad era tener una idea clara de la distribución de componentes y conexiones eléctricas. Una vez esto fuera completado se pasaría a la implementación de este circuito final a Fusion y el diseño del PCB. El esquema iterado fue el resultante en la Figura 46. Notablemente en esta iteración aun falta el segundo convertidor de 5V para la barra de 3.3V, y el interruptor para la alimentación. Además, este diseño aun contempla la conexión a la línea de carga del HW-373 con una conexión al pin de alimentación del OBC.

Este diseño fue llevado a Fusion para la siguiente iteración, llegando al diseño directo del PCB. Durante esta iteración se intentó realizar el diseño del PCB intentando acomodar las restricciones o reglas de diseño en el caso de fabricar con una fresa CNC el PCB. El método usado por la fresa CNC para fabricar el PCB es que a partir de una placa virgen de fibra con una capa de cobre, la fresa cortaría la primera capa de la tarjeta, como se puede apreciar en la Figura 45.

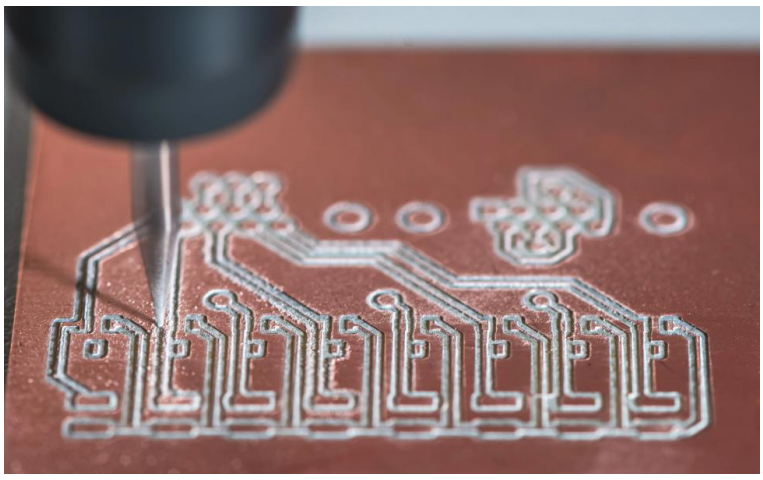


Figura 45: Proceso de fresado CNC de tarjetas de circuitos impresos

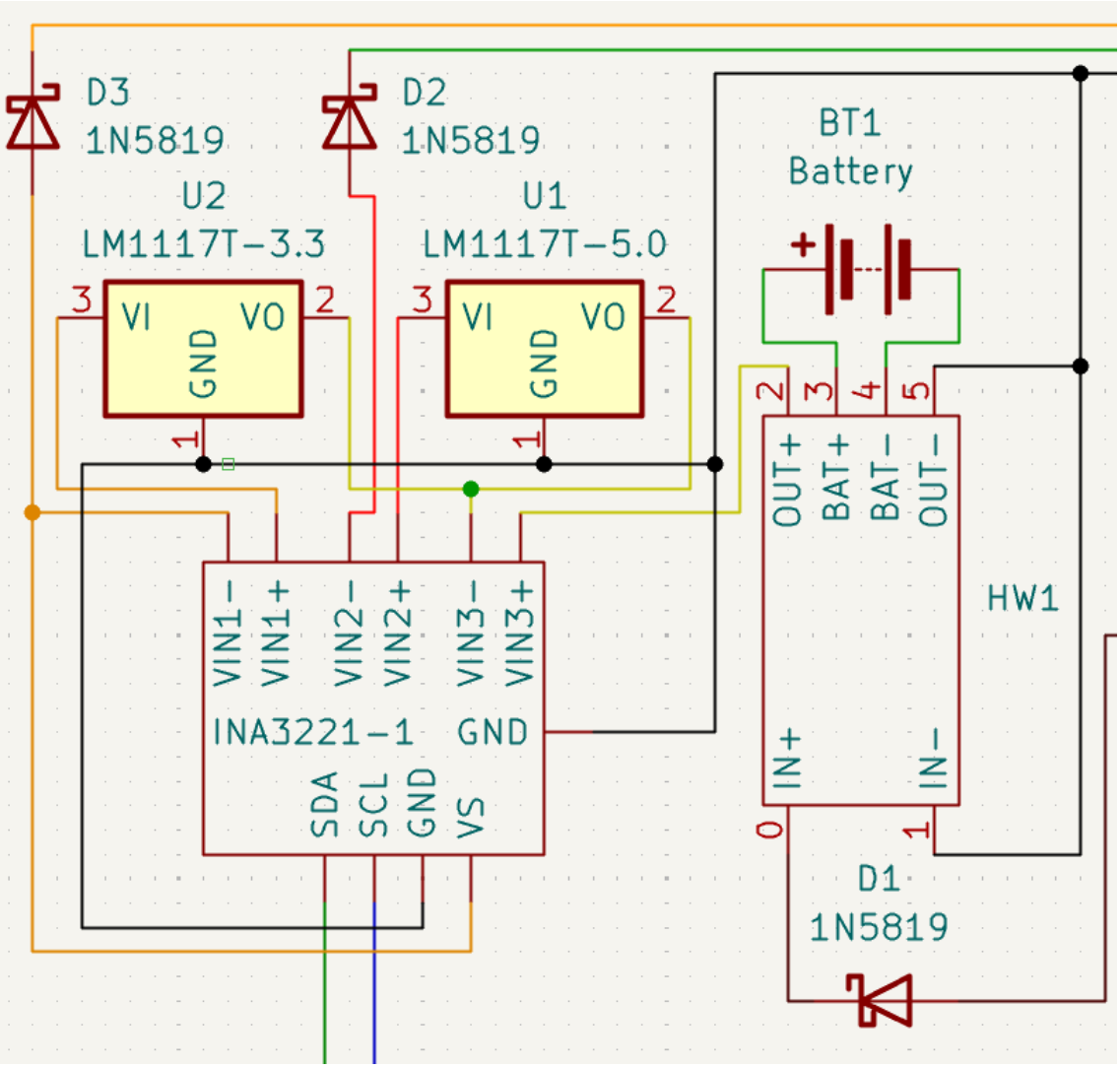


Figura 46: Esquema eléctrico del EPS en la iteración intermedia, en KiCad.

La principal restricción del método de fresado CNC para la fabricación del PCB es el diámetro de la fresa de 0.5 mm, lo que restringe significativamente el espacio disponible para realizar perforaciones de pines (pad). Esto es particularmente restrictivo en el sector del conector BTB del PC/104, donde la separación entre pads es de 2.54 mm. Esto incluye el ancho del conector (aproximadamente 1 mm), el área de contacto del pad, o sea, donde debe haber una "ruta" envolviendo el pad de mínimo 0.3mm de ancho (sumando 1.6 mm de diámetro total), y el área de exclusión de la ruta, por donde la fresa pasaría, de 0.5mm de ancho (superando los 2.6 mm de diámetro). En conclusión, no hay espacio suficiente para permitir la instalación de estos elementos.

Es por este motivo que la última iteración de diseño consideró la fabricación en el C4i, donde la posibilidad de fabricación permitiría la utilización de 2 capas de circuito, uso de vías, y menores restricciones en el diseño respecto a la alternativa de fabricación por CNC.

Una acotación realizada por el responsable del C4i a cargo de la fabricación del PCB fue que hacer pasar rutas de circuitos entre medio de pads, especialmente en la zona del PC/104, aunque sea técnicamente posible, no es deseable por el riesgo de causar arcos entre rutas y pads. Esto incidió

directamente en la decisión de redefinir el pineo del conector PC/104, llevando todas las conexiones relevantes a la primera fila del conector (H1 impar), en lugar de usar predominantemente la tercera fila (H2 impar). Así, desde la Figura 22 se realizó la modificación presente en la Figura 47.

H1				H2			
1	UART0 TX (NV	2		1		2	
3	UART0 RX (NV	4		3		4	
5		6		5		6	
7		8		7		8	
9		10		9		10	
11		12		11		12	
13		14		13		14	
15		16		15		16	
17		18		17		18	
19		20		19		20	
21		22		21		22	
23		24		23		24	
25	5V-O	26		25		26	
27	3.3V-O	28		27		28	
29	GND-O	30		29		30	
31		32		31		32	
33		34		33		34	
35		36		35		36	
37		38		37		38	
39		40		39		40	
41	I2C_SDA-I/O	42		41		42	
43	I2C_SCL-I/O	44		43		44	
45		46		45		46	
47	LUPO	48	LUPO	47		48	
49	LUPO	50	LUPO	49		50	
51	LUPO	52	LUPO	51		52	

Figura 47: Pinout final del conector PC/104 en la última iteración del prototipo.

Con todas las principales definiciones de diseño, se pasaría a la implementación final del EPS.

6.2.3 Implementación final

El diseño final del EPS se realizó considerando inmediatamente el diseño el PCB para ser fabricado en el C4i. Por esto, se diseñó en Fusion, teniendo precaución de crear las librerías de componentes electrónicos necesarias dentro de este software. Una preocupación significativa, y que efectivamente causó problemas, es la vinculación de pines físicos y electrónicos (en el diseño del PCB y el esquema eléctrico, respectivamente), debido a que por cómo se montarían los elementos en el PCB, estos podrían quedar invertidos. Esto ocurrió en 3 componentes, 1 de ellos siendo los reguladores de 5V, que afortunadamente eran suficientemente compactos además de tener una disposición de pines simétrica (entrada – tierra – salida), lo que permitió poder simplemente dar vuelta los componentes en el ensamble final.

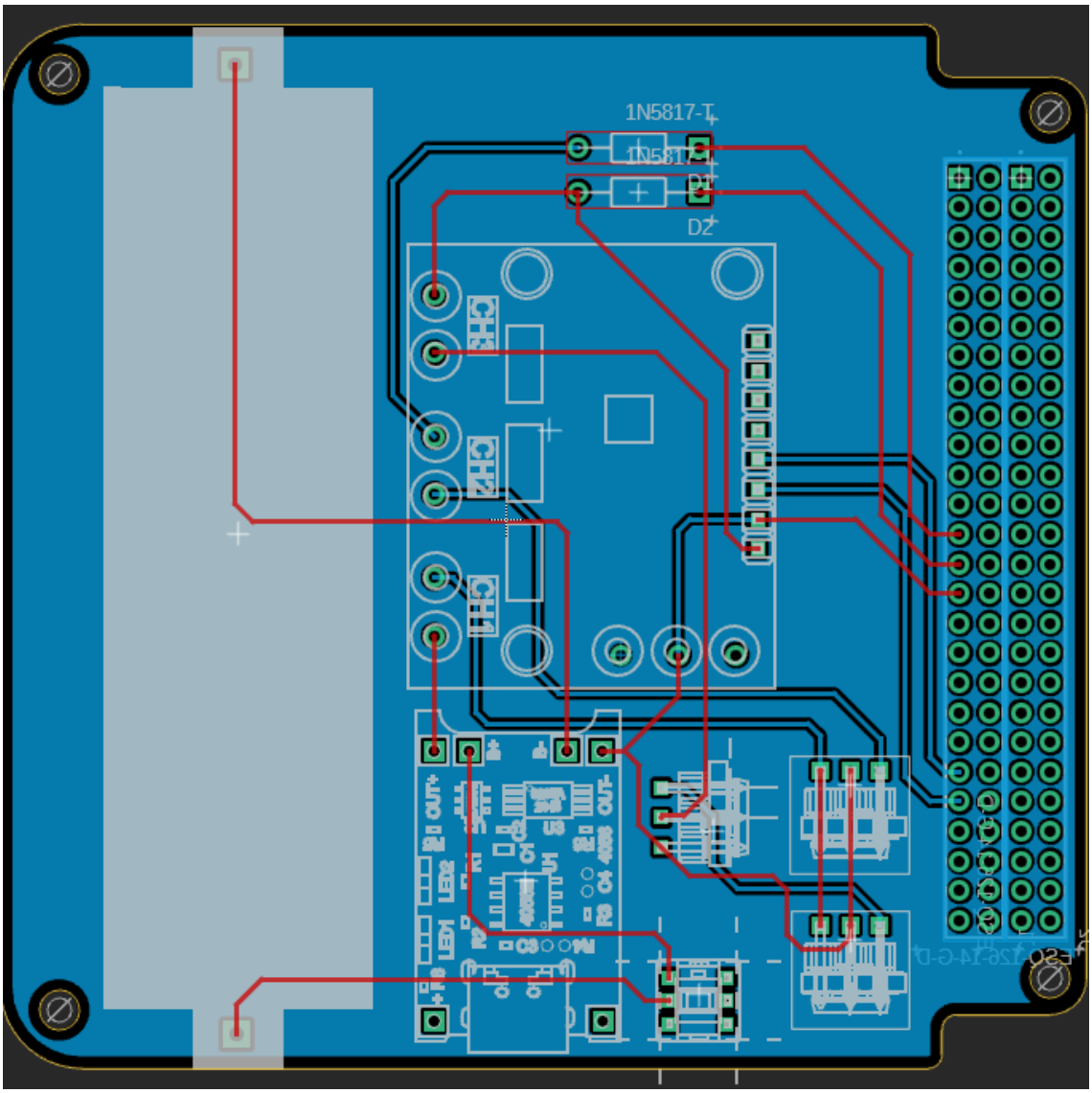


Figura 48: Diseño de PCB para el EPS, en Fusion.

Los componentes más restrictivos respecto a su ubicación en el PCB fueron, en decreciente orden de complejidad:

1. El porta baterías, cuyo tamaño significativo simplemente forzó a ser ubicado de forma transversal al eje frontal del satélite (de lado a lado). Esto es un punto importante a considerar para futuras iteraciones, debido a que la batería es: 1. Frágil y un riesgo de seguridad en caso de ser dañada, 2. Significativamente pesada, afectando el balance del CubeSat, y 3. Un componente crítico del sistema.
2. Módulo de sensor INA-3221, debido a que requiere tener conexiones para la lectura del voltaje de salida de la batería, y ambas barras de alimentación, y además debe ser alimentada, y poseer una conexión al bus I2C principal.
3. El regulador de carga HW-373, cuyo conector de carga USB-C debe estar orientado en dirección X+, de acuerdo con lo definido en la Figura 16.
4. Interruptor de potencia, que debe ser accesible desde afuera con facilidad.

Todo esto fue integrado en el diseño del PCB (Figura 48). Este diseño fue presentado al C4i para fabricación, donde algunos errores o diferencias de procedimiento causaron una fabricación sub-óptima del PCB respecto a las necesidades del diseño (Figura 49).

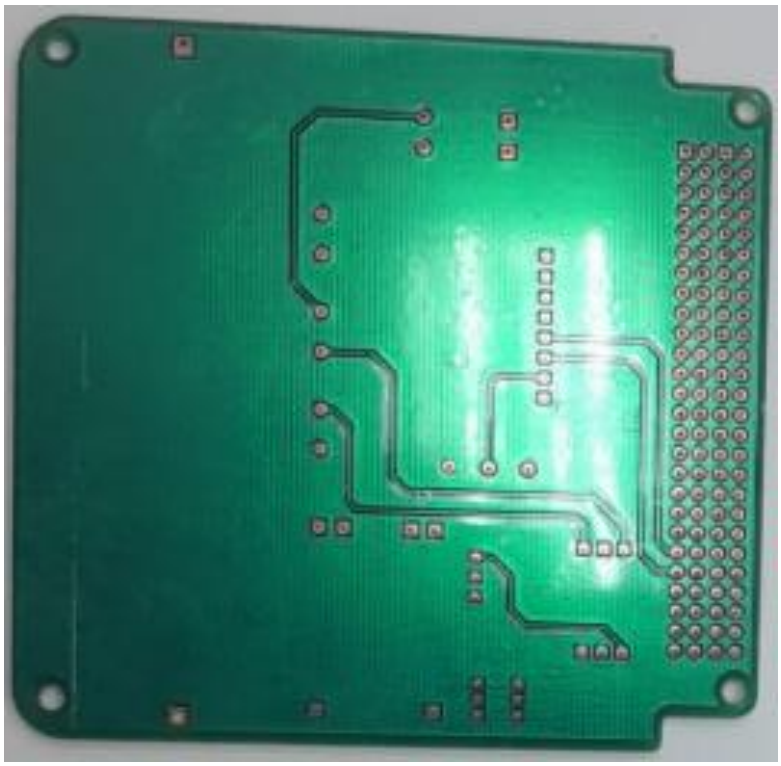


Figura 49: PCB del EPS recién fabricado

Uno de los errores más comunes fue que el tamaño de los agujeros o pad para la instalación de componentes era menor al ancho de los pin-header utilizados para la conexión. Para solucionar este problema, se consideró usar pin-header hembra en lugar de soldar directamente los componentes al PCB. Esto cumpliría dos propósitos:

1. Los conectores de soldadura para los pin-header hembra son más delgados que sus análogos macho, haciendo la integración de componentes más fácil.
2. Permitir la posibilidad de retirar e intercambiar componentes en caso de presentar fallos o querer hacer pruebas en aislamiento.

Algunas consecuencias de esta decisión serían las siguientes:

1. La experiencia AIT podría considerar la aceptación e integración de componentes del subsistema en lugar de ser un subsistema monolítico.
2. La altura de los subsistemas en el stack del CubeSat sería mayor. Esto no sería particularmente problemático en el EPS debido a que la batería ya aporta una altura significativa y superior a la de los componentes + conectores.
3. Algunos componentes no soldados son mas susceptibles a soltarse del ensamble al ser movidos, agitados o girados.

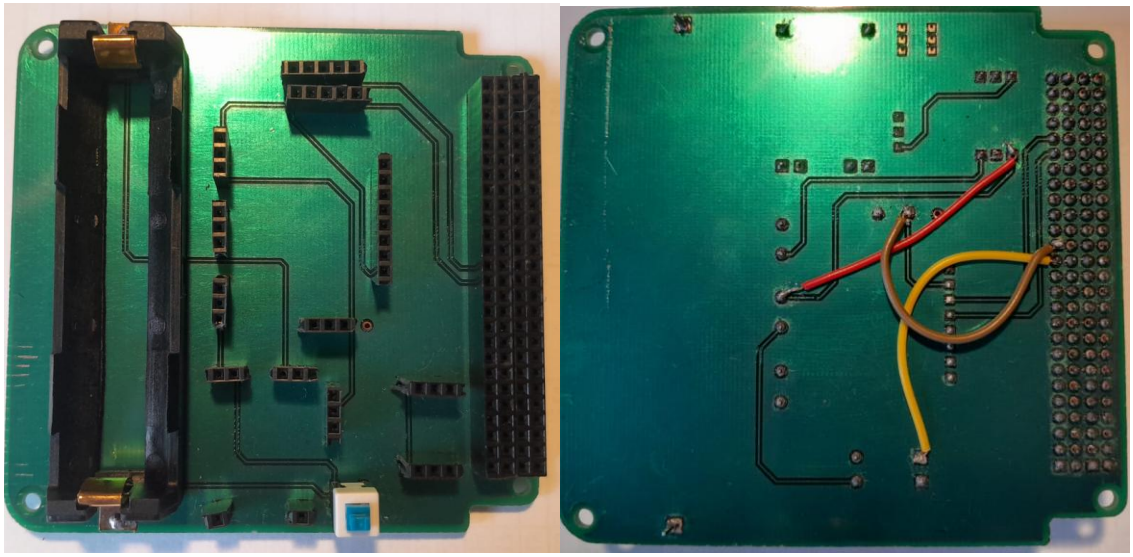


Figura 50: PCB del EPS con elementos de superficie soldados. Izquierda: vista superior. Derecha, vista inferior.

El otro problema o defecto de fabricación encontrado fue la discontinuidad de algunas rutas o conexiones esperadas. Una de estas fue la de tierra o GND. Esta es usada por defecto como una capa completa por el fabricante. Eso resulta en que el EPS tuviera esta capa faltante, ya que en Fusion no se definió ninguna capa como la capa de tierra sola (cada red de conexiones debe tener un identificador), causando que entre el bus BTB y el resto del circuito no existiera una conexión. Esto fue solucionado usando cables auxiliares que fueron soldados al reverso del PCB, los cuales se pueden apreciar en la Figura 50 (derecha).

Otras dos rutas fueron encontradas como discontinuas, una de ellas siendo la línea de 5V entre la INA-3221, y el regulador de la barra de 5V (cable rojo). Este defecto es visible a simple vista como puede apreciarse en la Figura 51.

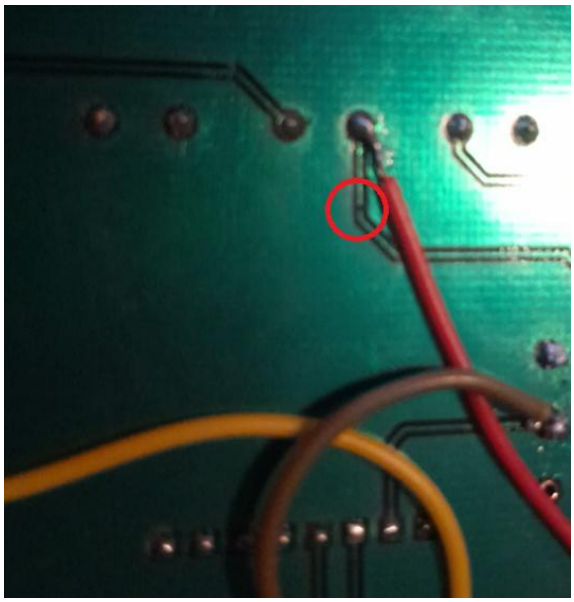


Figura 51: Defecto de ruta discontinua en PCB del EPS.

Finalmente, el EPS completamente integrado puede apreciarse en la Figura 52:

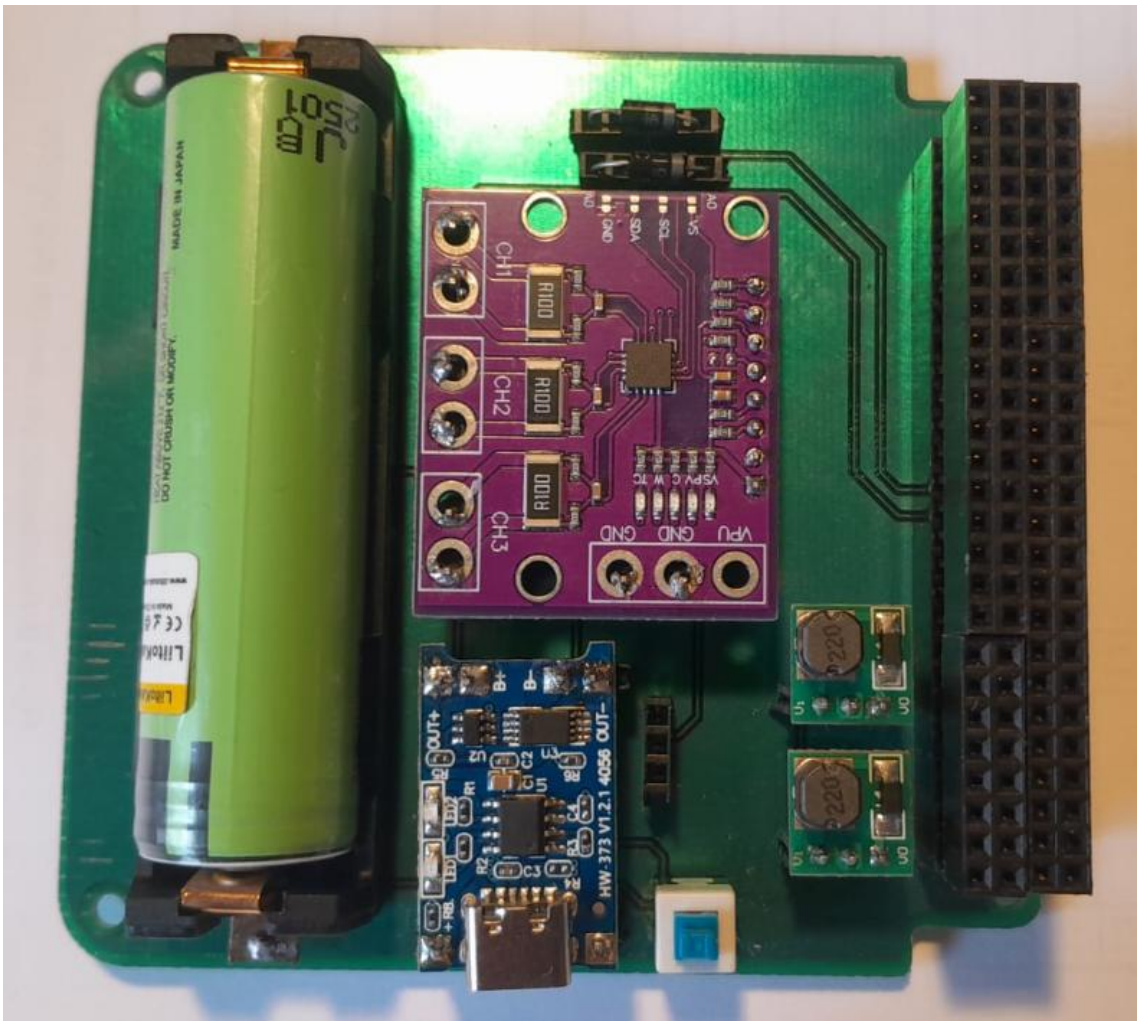


Figura 52: EPS completamente integrado

6.3 Diseño y fabricación de la estructura

La función principal de la estructura es proveer un soporte físico (interfaces mecánicas o de fuerza) para los elementos físicos del resto del CubeSat. Esto, siguiendo las especificaciones de los estándares PC/104 (GomSpace) y CDS. La principal limitante respecto a la estructura viene dada según las indicaciones ya mostradas en los capítulos 3.3.1, 4.2 y Anexo A. Originalmente el diseño de la estructura siguió la forma del PCB presentado en el Anexo A, además de intentar seguir las especificaciones del CDS desde un principio en temas de la disposición de las “patas” y el cuerpo. Esto puede verse reflejado en la Figura 38, donde el PCB del EPS es simplemente una forma primitiva del diseño final. Otra consecuencia de esta aproximación es la robustez de las partes estructurales como la base mecánica del EPS.

Para continuar el desarrollo de una forma más lógica y de acuerdo con las necesidades de la experiencia como de las especificaciones entregadas, se consideraron los siguientes principios de diseño:

1. La estructura debe reducir la cantidad de partes libres donde sea posible.
2. La estructura debe reducir la cantidad de partes únicas o diferentes entre sí.
3. La estructura debe permitir un ensamble rápido y conveniente.
4. El proceso de ensamble de la estructura debe depender de la menor cantidad de herramientas posible, y reducir la complejidad.
5. En general, se busca generar una estructura liviana, pero resistente.
6. Debe seguir las especificaciones del CDS. Se presenta la estructura de Isispace como un ejemplo de diseño.

Para simplificar la explicación respecto al proceso de diseño, este se divide en dos secciones:

1. Elementos de sujeción (o fasteners) tales como tornillos, tuercas, pasadores, etc.
2. Elementos estructurales o marcos.

6.3.1 Elementos de sujeción

Estos representan principalmente las interfaces mecánicas entre elementos físicos. Hay dos categorías de estas interfaces: 1. Varillas de soporte y 2. Tornillos en uniones semipermanentes.

Las varillas de soporte son una adaptación de los separadores de tarjetas del PC/104 o standoffs (Figura 53).

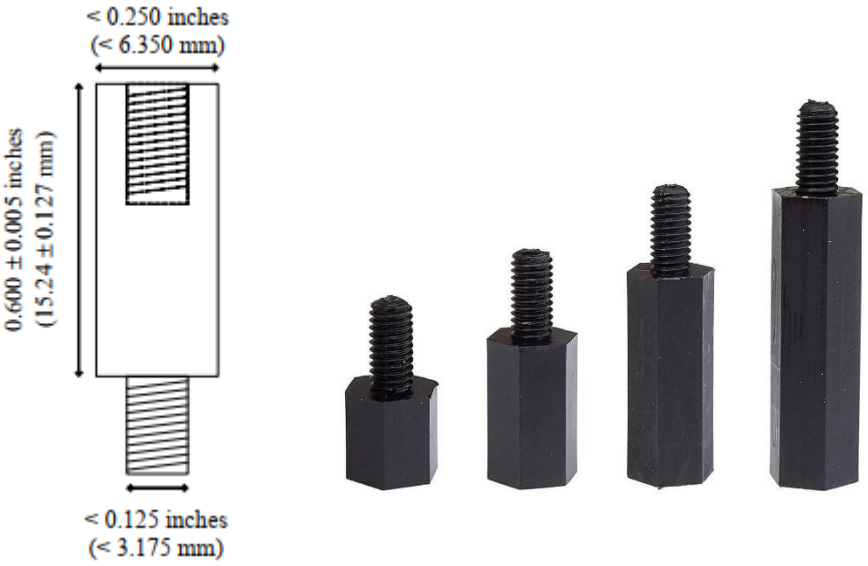


Figura 53: Spacers o standoffs. Izquierda: Según la especificación PC/104. Derecha, ejemplo de standoffs de distintas longitudes.

Estos materiales presentan algunos inconvenientes respecto a la implementación de la estructura, siendo uno de los mayores, la disponibilidad de estos y la incertidumbre respecto a la longitud necesaria en específico. Algunos kits de distribución incluyen un surtido de diferentes diámetros y longitudes, pero por lo general son cantidades limitadas de cada uno.

La otra complicación es la longitud total del ensamble de standoffs, la cual debe coincidir con el espacio disponible dentro del marco del CubeSat.

La primera alternativa evaluada fue la de diseñar spacers para ser impresos en 3D en PLA, lo que presentó el problema de cómo hacer un hilo efectivo en una parte impresa, sobre todo considerando que el diámetro de los agujeros en los PCB donde deberían estar los standoffs son de 3mm. En resumen, esta solución es inviable y poco práctica para los alcances de este proyecto.

La siguiente alternativa consiste en tomar el mismo principio básico, imprimir en 3D separadores, pero en este caso simplemente serán pasadores con una varilla central que se extienda a lo largo de la altura del espacio interior del CubeSat. De esta forma se plantea el uso de una varilla central que restringiría el movimiento planar X/Z, acompañado de un elemento separador que restrinja la separación en Y, según los ejes definidos en la Figura 16.

En la primera iteración de diseño se consideró usar un espárrago, o varilla completamente roscada, que recorriera las 3 unidades del CubeSat (Cuando en la primera mitad del proyecto aun se consideraba la fabricación de un CubeSat de 3U en lugar de 1U). Esta solución traería complicaciones en múltiples formas:

1. Implementación en el diseño: Al ser una varilla roscada, esta requeriría de una rosca donde ser sujeta. Hay dos alternativas en este caso: A) Utilizar tuercas en conjunto con la varilla, permitiendo perforaciones lisas en los elementos estructurales en lugar de requerir perforaciones roscadas o B) Utilizar insertos roscados para estructuras de PLA.
2. Consideraciones prácticas por el ensamble: La presencia de la rosca genera una interferencia con las perforaciones de los PCB, lo que a corto plazo aumenta el riesgo de que un PCB quede atascado en una posición oblicua, y en general sea un poco más incomoda de ensamblar. Pero a largo plazo también podría causar daño en los PCB mismos al erosionar las paredes de la perforación. Esto último adicionalmente acarrea el riesgo de entrar en contacto eléctrico con la placa base de cobre en el PCB, habitualmente usada como la red de tierra (GND), según lo mencionado anteriormente en el capítulo 6.2.3.
3. Problemas para la adquisición: Para permitir la incorporación de una varilla de soporte a lo largo de las 3U del CubeSat, esta requeriría de una longitud superior a los 300 mm. Si bien no es un elemento imposible de encontrar, sí es uno que múltiples pernerías indicaron no trabajar por la combinación entre la longitud elevada y un diámetro muy bajo.

Aun así, esta iteración logró definir dos cosas: 1. La presencia de un hilo es favorable para facilitar la integración con otros elementos estructurales y 2. La presencia del hilo es desfavorable respecto a su interacción con los PCB. Por lo tanto, **es deseable que las varillas de soporte presenten un hilo en los extremos, pero no en el centro.**

Una parte encontrada en el mercado que convenientemente cumple con estas especificaciones son las varillas de empuje utilizadas en modelos de barcos y aviones (Figura 54). Estas usualmente son roscadas en los extremos para permitir el ensamble de extremos de sujeción, pero que en este caso permitirían atornillar directamente las varillas al marco de la estructura si esta incluye un hilo hembra.

Figura 54: Varillas de empuje roscadas únicamente en los extremos.

En paralelo, la decisión de usar elementos estructurales impresos en 3D fue tomada, cementando la decisión de usar insertos roscados de PLA en los puntos de anclaje de las varillas. Otra consecuencia de esta decisión fue la de establecer el diámetro de rosca M3 como un estándar para todos los elementos roscados de sujeción en la estructura. Con esto se tomó la decisión de comprar varillas de empuje de 30 cm, roscadas por ambos extremos.

Otra decisión de diseño que afectó el diseño final de las varillas de soporte fue la resolución de adoptar únicamente una configuración de 1U en lugar de las 3U, reduciendo la longitud total de las varillas. Como las varillas solo realmente necesitan ser atornilladas al marco inferior[6], las varillas fueron cortadas para la nueva longitud requerida, aproximadamente 8.7 cm de largo. Dos varillas cortas se pueden obtener de una sola varilla de empuje larga con este método.

Finalmente, respecto a los tornillos e insertos roscados, estos siguen todos la lógica del estándar M3. Esto causó un rediseño del marco estructural para permitir la inclusión de estos tornillos más grandes. Además, se definió que todos los tornillos tendrán una cabeza compatible con un destornillador de cruz.

[6] Implementar una rosca en el extremo superior requería incluir una tuerca u otro elemento libre respecto al marco. Como no existe un riesgo de que la varilla se deslice fuera del marco, ni que el marco superior dependa de la varilla para conservar la estructura, esta puede quedar libre.

Figura 55: Izquierda: Inserto roscado usado en PLA. Derecha: Instrucciones de diseño para la incorporación

En un principio se intentó verificar las recomendaciones de diseño (Figura 55, derecha) por medio de realizar una prueba con una matriz de puntos para la prueba (Figura 56). A partir de esta prueba se definió que el ancho mínimo para que la estructura soporte el inserto es de 1mm, y con un diámetro de perforación de 4mm.

Figura 56: Probeta de pruebas para insertos roscados en PLA, con una matriz de perforaciones para probar diámetro de perforación recomendado y ancho de pared mínimo.

Si bien esta configuración de diámetro de agujero-ancho de pared sería efectiva para mantener el inserto en su lugar, y aguantaría el proceso de instalación, más adelante causaría problemas por la resistencia del material, causando la ruptura de 3 puntos de sujeción del marco. Un defecto de diseño encontrado posterior a la fabricación es que el ancho de la pared en las sujeciones horizontales es menor al requerido, siendo este de 0.9 mm en su parte más angosta.

Figura 57: Aplicación exitosa de insertos roscados en la estructura del CubeSat.

En la Figura 57 se puede apreciar la instalación de insertos en puntos de sujeción donde han soportado las cargas mecánicas sin problemas, mientras que en la Figura 58 se presenta uno de los 3 puntos de falla donde la pared superior de la perforación falló por exceso de cargas.

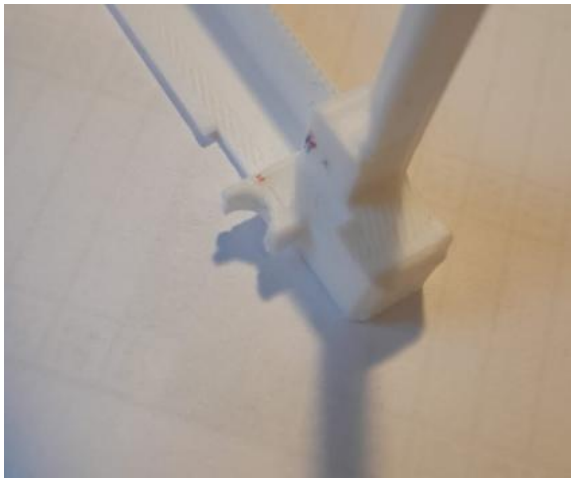


Figura 58: Punto de sujeción con inserto de PLA quebrado.

Otro elemento que no está normalmente considerado dentro de la estructura, sino que en realidad es más bien dependiente de cada PCB, son los conectores BTB del PC/104. Aun asi, conviene evaluarlos en conjunto con las especificaciones estructurales debido a que también tienen una injerencia sobre la altura entre PCBs.

Normalmente la especificación del PC/104 incluye una especificación para esta altura (Ver Anexo A), la cual depende de la longitud de los pines macho (o las “patas largas” del PCB), y la profundidad de los conectores hembra por la parte superior. Para permitir un contacto correcto entre estos, la longitud de ambos está definida.

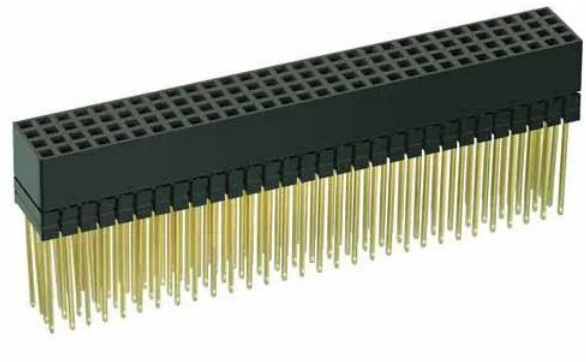


Figura 59: Conector Stackthrough PC/104 de 4x26 pines.

Lamentablemente, si bien estos conectores (Figura 59) existen en el mercado y están disponibles por distribuidores como Mouser Chile, el costo unitario de estos es prohibitivo[7]. Por esta razón se optó por hacer un "ensamble equivalente" de conectores, usando conectores más accesibles de tipo 2x8 y 2x18 (Figura 60).

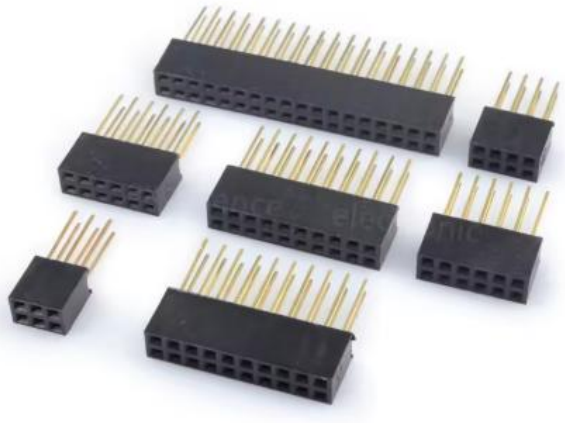


Figura 60: Conectores no conformes a PC/104, pero agrupables para ser equivalentes.

Este conector Stackthrough sería particularmente relevante para la PCB que se instale encima de otras. Debido a que el orden de integración del sistema busca ser: EPS -> OBC -> Estructura, se define que el EPS será el subsistema que se ubique en la parte inferior (Figura 64). En el caso del EPS, no es necesario que los conectores macho sobresalgan o se extiendan hacia abajo, por lo que se usan conectores hembra de doble fila cortados para cumplir con los 26 pin de longitud.

Un problema con esto es que la altura de los conectores usados es menor que la especificación PC/104, resultando en que la separación entre PCBs no se conforme con las definiciones mostradas en el Anexo A. Considerando que los spacers de PLA diseñados para la primera iteración usaron estas medidas como referencia, el resultado es que estos fueron sobredimensionados respecto a las

[7] Mouser Chile dispone de un conector de especificación PC/104, marca Harwin, pero el costo unitario comienza en 24,595 $CLP. *[Mouser Chile]*

necesidades de la estructura y por lo tanto no fueron incluidos en el ensamble final del prototipo, pendiente de rediseño para cambiar la altura.

6.3.2 Elementos estructurales

Los elementos estructurales o colectivamente el marco estructural del CubeSat fueron modificados directamente a partir de la plantilla presentada por Isispace. Originalmente se trabajó sobre la plantilla de 3U, pero esta fue recortada y resultó ser idéntica para una de 1U (Figura 61).

Figura 61: Plantilla estructural de ISISPACE para un CubeSat de 1U.

Originalmente se pensó en realizar los elementos estructurales en chapa metálica plegada, lo que permitiría fabricar múltiples marcos idénticos en masa, con capacidades disponibles en el taller mecánico del edificio tecnológico mecánico de la universidad (TM), pero debido a la complejidad de algunos elementos que requerirían maquinado especial, la complejidad agregada de balancear métodos poco ortodoxos de fabricación en un prototipo, y riesgos debido a la conducción eléctrica de la estructura, se desistió de esta idea.

Una preocupación durante el diseño de estos elementos era la capacidad de la impresora 3D de fabricarlos, especialmente considerando los agujeros que no pueden quedar “verticales” en la cama de la impresora y serían impresos longitudinalmente.

La primera prueba consideró la impresión 3D de la estructura para 1U con modificaciones mínimas (Figura 62), resultando ser sorprendentemente fiel al modelo digital y cuyos agujeros resultaros ser aceptables posterior a la impresión. El tiempo total de impresión fue poco más de 6 horas y solo hizo falta un intento para obtener una impresión satisfactoria.

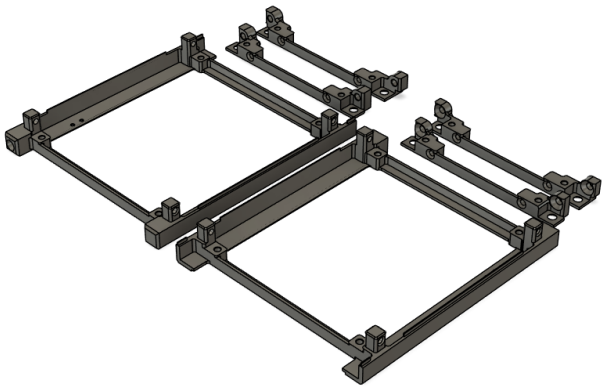


Figura 62: Partes estructurales impresas en 3D para la primera prueba de impresión.

La siguiente iteración del diseño consideraría la integración con los elementos de sujeción. Estos principalmente requieren la incorporación de 12 insertos roscados en los elementos del marco. Para esto, la principal modificación fue la de agrandar todos los agujeros que determinan interfaces físicas utilizadas, siendo estos las uniones entre los marcos laterales, los conectores frontales y posteriores, y los puntos de sujeción de las varillas de apoyo. Todos estos mantendrían el centro de la perforación sin cambios, pero aumentarían su diámetro a 4 mm para permitir la instalación del inserto.

Los otros cambios consisten en la eliminación de agujeros no usados en este prototipo, como los relevantes para el kill-switch, y los agujeros de apoyo para los paneles exteriores. Como todos estos elementos serían impresos en 3D en conjunto con los spacers de PLA diseñados anteriormente. Todos estos serían impresos en conjunto desde el mismo archivo STL (Figura 63).

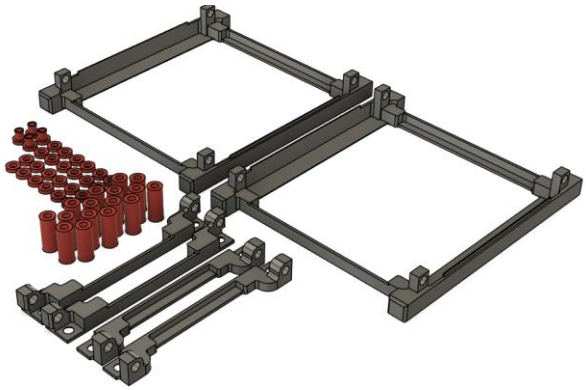


Figura 63: Conjunto de elementos impresos para la iteración final.

6.3.3 Implementación final

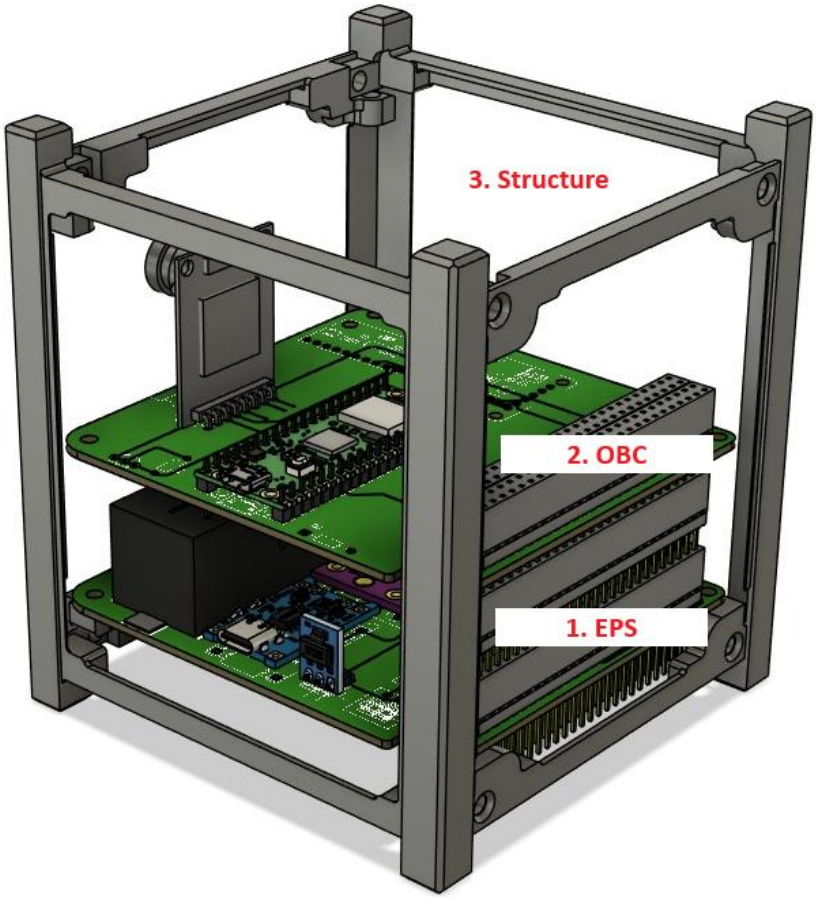


Figura 64: Configuración de subsistemas en CubeSat e implementación final de la estructura.

La implementación final de la estructura siguió los pasos anteriormente descritos en el diseño de esta. El resultado es que la estructura final está compuesta únicamente de 7 tipos de partes o materiales:

1. 2x Marcos laterales identicos.
2. 1x Conector frontal inferior (con insertos)
3. 1x Conector frontal superior (sin insertos)
4. 1x Conector posterior inferior (con insertos)
5. 1x Conector posterior superior (sin insertos)
6. 4x Varillas de soporte
7. 8x Tornillos M3x6 con cabeza de cruz.

Estos, en conjunto con los spacers impresos en 3D y los PCB recién fabricados se presentaron en el primer ensamble:

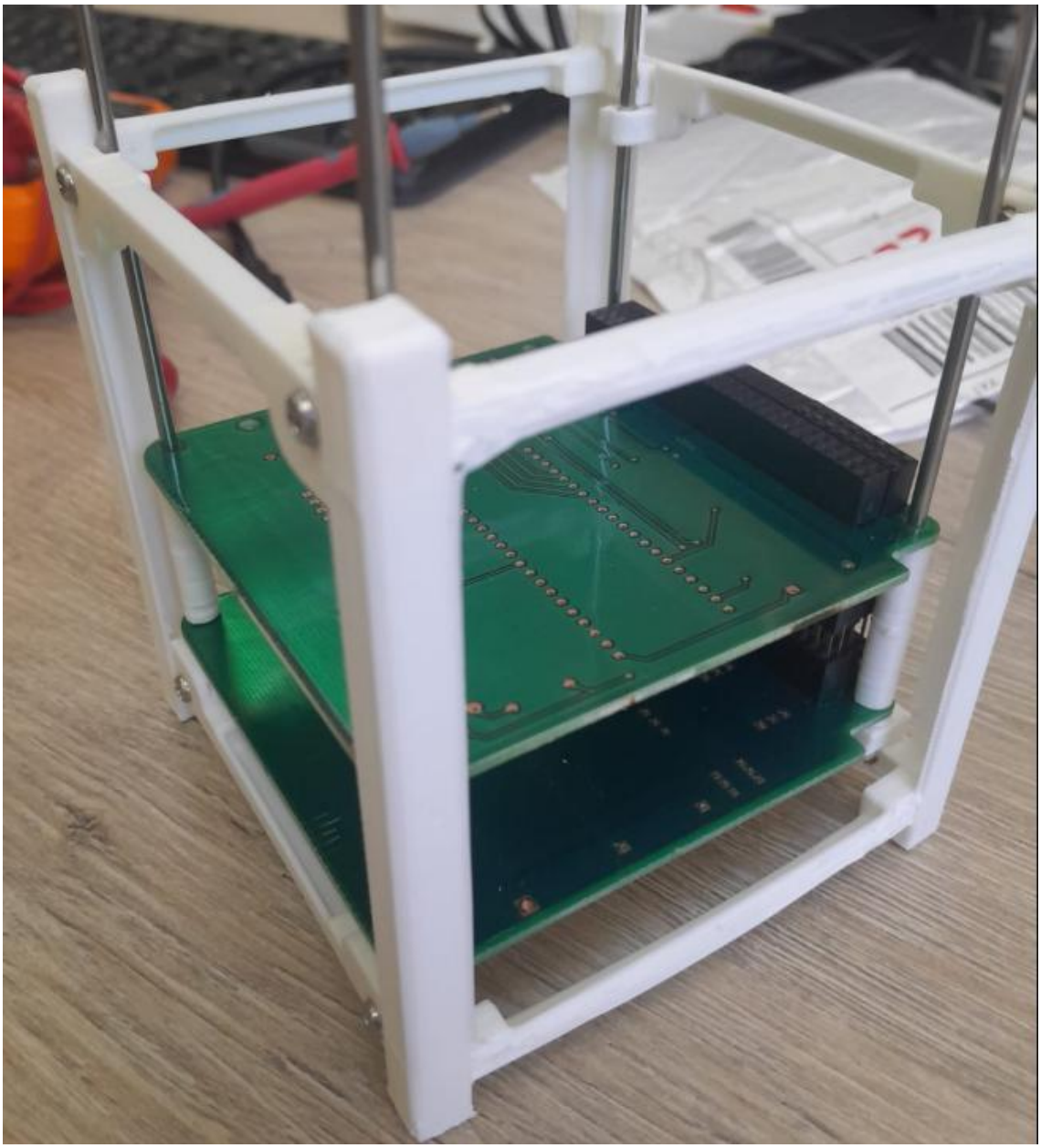


Figura 65: Primer ensamble estructural con PCBs. En esta fotografía aun no se han cortado las varillas a la longitud final.

Gracias a esto, se identificó la discrepancia de altura entre los spacers impresos y la altura de los conectores BTB. La distancia entre el EPS y el OBC requiere que se instale un conjunto de conectores adicional sobre los conectores del EPS, lo que permite generar un puente entre los conectores de ambos subsistemas.

Finalmente, luego de recortar las varillas de soporte a la longitud final según lo indicado en el punto 6.3.1 (Figura 66), y de realizar los insertos de PLA en las 12 posiciones necesarias, se consigue una integración estructural final.

Figura 66: Varillas recortadas a la longitud final.

6.4 Diseño de detalle e implementación del OBC y funciones asociadas

Lo llamado OBC en esta iteración del prototipo es una amalgama de funciones llevadas al mismo ensamble de un único PCB. Estas funciones son:

1. Gestión de comandos y datos (Command and Data Handling)
2. Comunicaciones
3. Determinación de actitud
4. Medición de datos escalares (de temperatura y luminosidad)
5. Captura de imágenes

El OBC, propiamente tal es el On-Board Computer, el cual fue definido como la Raspberry Pi Pico W, de acuerdo con lo mencionado en el capítulo 4.1.1. Este componente no requiere de un rediseño físico, pero sí está sujeto a decisiones de diseño relacionadas con otros componentes del subsistema.

En orden cronológico de diseño, estos serían:

1. Captura de imágenes, módulo de cámara Arducam 2MP.
2. Sensores de potencia INA-3221 del EPS y INA-219 dedicado al OBC.
3. Sensores escalares de temperatura (DS-18b20) y luminosidad (TEMT-6000).
4. Sensor IMU MPU-92/65.

Además de considerar las integraciones de un payload externo desarrollado por Nicolás Valderrama en la forma de, originalmente, una FPGA, y posteriormente reemplazado por otra Raspberry Pi Pico, y la integración del EGS, el cual también presenta una amalgama de funciones entre comunicaciones y verificación de interfaces lógicas del stack de conectores PC/104.

Estos fueron descritos en la sección 4.1, y en general no presentan mayores cambios respecto a su integración.

6.4.1 Diseño e implementación del PCB

En retrospectiva, el diseño del PCB para este ensamble es poco elegante, ya que la aproximación a resolver la incorporación física de los elementos fue de hacer calzar todos los componentes dentro del espacio disponible en el PCB, siguiendo con las directrices de diseño planteadas en la Figura 16. El diseño del diagrama general de circuito, y del PCB siguió los mismos pasos que en el caso del EPS, con la inclusión adicional de un bus UART que se plantea como un mecanismo de comunicación serial entre microcontroladores conectados al bus PC/104, en particular a los microcontroladores presentes en el prototipo de Nicolás Valderrama y del EGS.

El esquema de circuito fue diseñado en Fusion, utilizando bibliotecas personalizadas para los siguientes componentes:

- Raspberry Pi Pico W
- TEMT-6000
- DS-18b20 + Resistencia de 4.7kΩ
- ArduCam 2MP
- MPU-9250
- INA-219

El resultado es el esquema de circuitos (por redes) mostrado en la Figura 67.

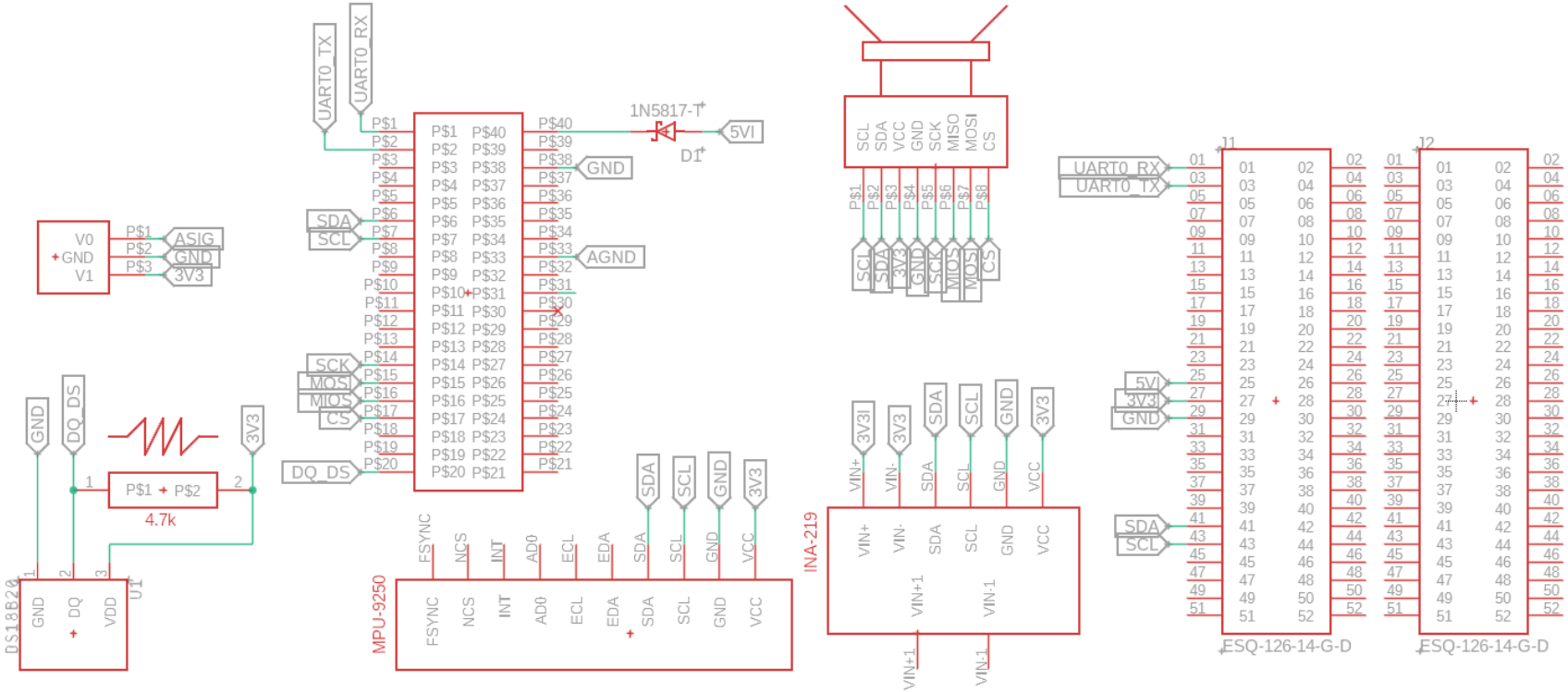


Figura 67: Diagrama general de circuito para el ensamble OBC.

A partir de este diagrama se realizó el diseño del PCB siguiendo el mismo estándar que el EPS.

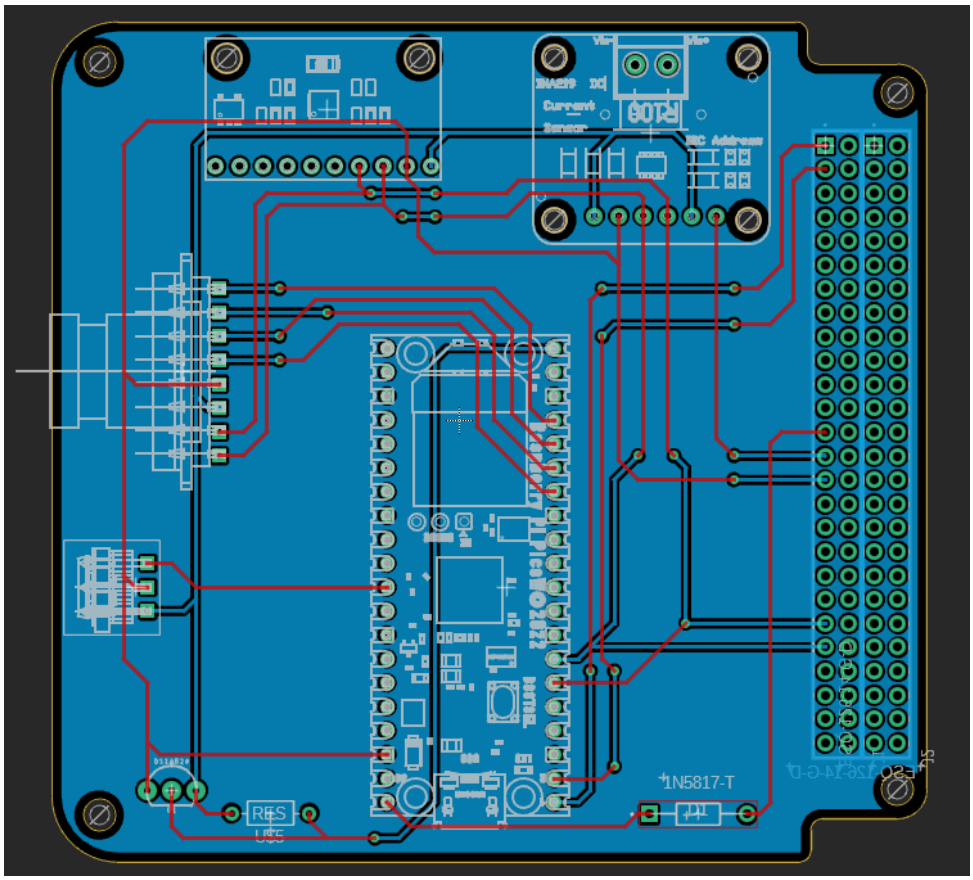


Figura 68: Diseño del PCB para el ensamble del OBC.

Una corrección que hacer a este modelo, es que la IMU MPU-92/65 está invertida respecto a los pines de conexión. En el ensamble final, esta quedaría apuntando hacia “adentro” del CubeSat, en lugar de cómo está presentada en la Figura 68. Esta configuración no interfiere mecánicamente con la cámara ArduCam, pero sí requiere que el orden de integración priorice la Cámara antes que la IMU. La interferencia se puede apreciar en la Figura 69.

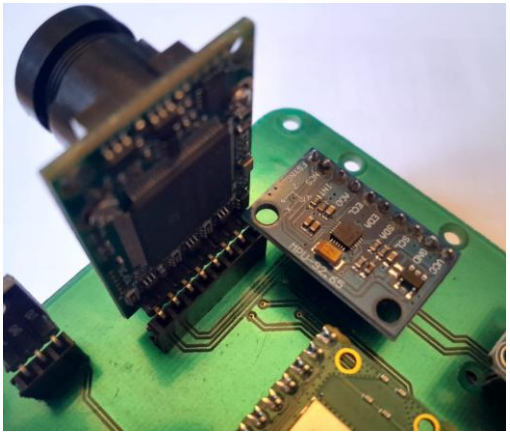


Figura 69: Interferencia del módulo MPU-92/65 con la ArduCam

6.4.1.1 Revisión al diseño de acuerdo con observaciones de comisión evaluadora

Debido a la imposibilidad de implementar la cámara ArduCam Mini 2MP por complicaciones relacionadas con la disponibilidad de controladores y la falta de soporte técnico por parte del fabricante, se decidió realizar un cambio al diseño del PCB considerando la integración de un módulo de cámara OV2640 de 18 pines. Este se realizó siguiendo el PinOut presente en el documento de datos técnicos del componente (MPJA, 2019), y realizando una modificación al PCB en Fusion.

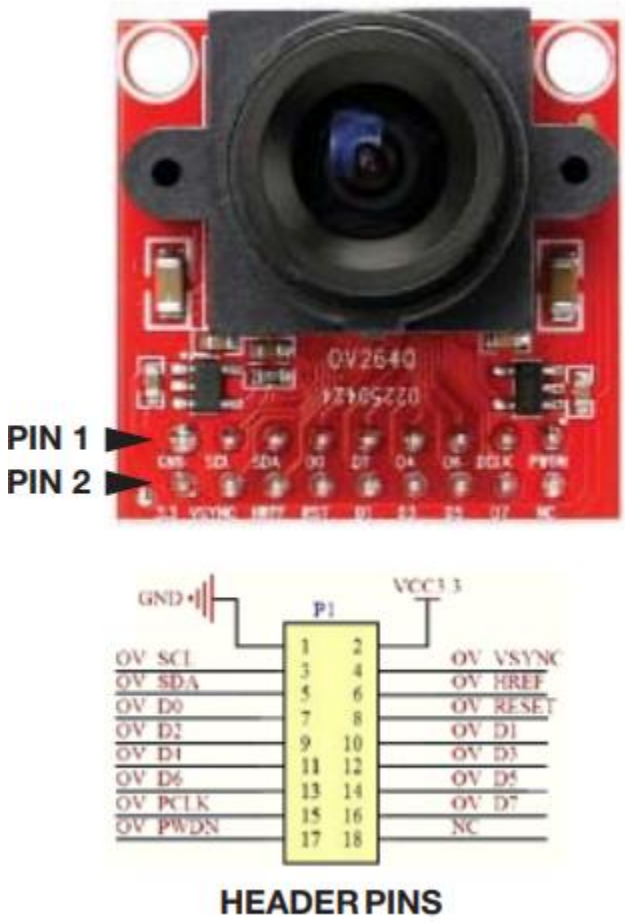


Figura 70: Pinout de la cámara OV2640 integrada en la última iteración del PCB para el OBC

Al hacer la integración en Fusion, se realizó una nueva biblioteca de componente con un proxy o modelo aproximado del componente. Para este fin se asume que la integración de este componente será análoga a los otros componentes, es decir, a través de un conector hembra directamente soldado en la PCB, mientras que el componente tendrá soldado Pin Header machos (y en este caso en L de doble fila, Figura 72), con los cuales se conectará al OBC sin necesidad de soldar permanentemente la cámara al PCB. De esta manera, lo único que hay que diseñar en el PCB son los pad para conectar el Pin Header Hembra, y las rutas de conexión para el circuito.

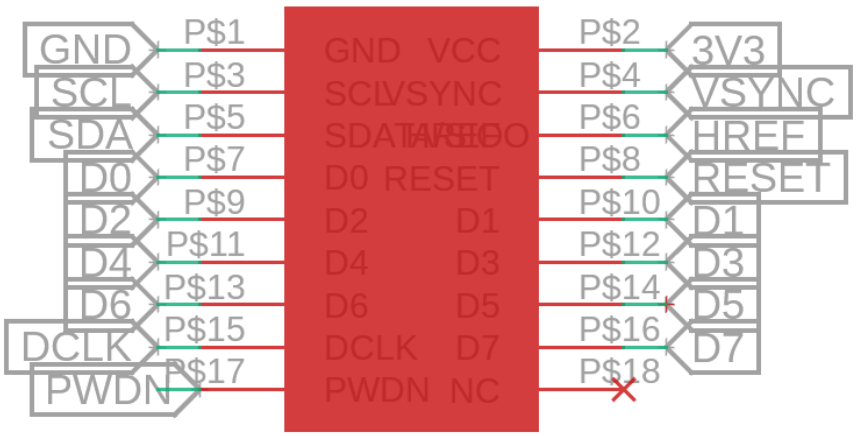


Figura 71: Biblioteca "proxy" de la cámara OV-2640 con conexiones definidas para el OBC.

Figura 72: Pin Header Macho de doble fila y en curva.

De esta forma, la conexión al PCB y al resto del circuito, en particular al microcontrolador, se puede asegurar usando la configuración de pines como muestra la tabla:

Tabla 1: Pinout del componente OV-2640 a la Raspberry Pi Pico W.

Pin [OV-2640-1]	**Conectado a:**	**Pin [OV-2640-2]**	**Conectado a:**
1: G**ND**	GND – pin 38	2: 3V3	3V3 – pin 36
3: SCL	SCL - pin 7	4: VSYNC	GPIO – pin 21
5: SDA	SDA - pin 6	6: HREF	GPIO – pin 22
7: D0	GPIO – pin 9	8: RESET	GPIO – pin 4
9: D2	GPIO – pin 11	10: D1	GPIO – pin 10
11: D4	GPIO – pin 14	12: D3	GPIO – pin 12
13: D6	GPIO – pin 16	14: D5	GPIO – pin 15
15: DCLK	GPIO – pin 25	16: D7	GPIO – pin 17
17: PWDN	GPIO – pin 24	18: NC	NC (No Conectado)

El resultado es el siguiente PCB modelado:

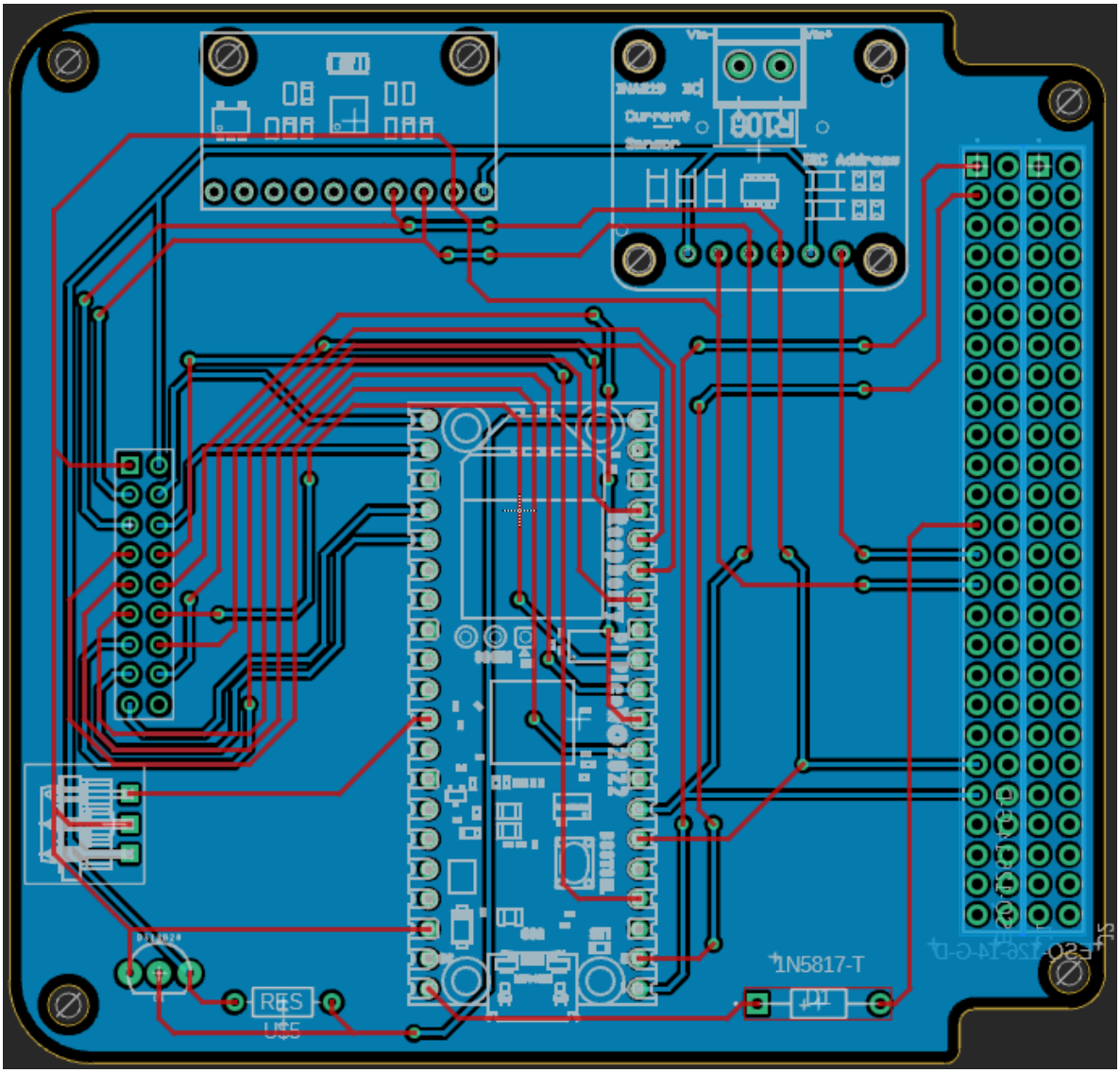


Figura 73: Iteración corregida del PCB para el OBC, considerando el cambio de la cámara por una OV-2640

6.4.2 Implementación de Software

La implementación del software de vuelo sigue los lineamientos de interfaces lógicas explicados en la sección 3.2.2, y siguiendo los principios de diseño de la sección 5.1. En resumen, el OBC Raspberry Pi Pico W, a través del microcontrolador RP-2040, ejecuta un firmware en la forma de un script diseñado para operar con CircuitPython, una distribución de MicroPython desarrollada por Adafruit, con el propósito de simplificar la implementación de librerías para componentes comunes.

Para poder cargar el firmware al OBC se debe conectar la Raspberry Pi Pico W por medio de un cable micro USB a un computador con un entorno integrado de desarrollo (IDE) compatible con CircuitPython. Para fines de esta memoria de título, el IDE seleccionado fue Thonny, el que permite no solo instalar el firmware desde el dispositivo virgen, sino que también modificar el código y la estructura de archivos de la memoria flash integrada del OBC por medio de una conexión serial COM al IDE.

La versión de CircuitPython utilizada es la v.10.0.3, instalada automáticamente por Thonny como la más actual a la fecha 17 de Octubre del 2025.

CircuitPython permite la implementación de librerías especializadas para gestionar diferentes funciones o componentes conectados al microcontrolador, además de permitir control de hardware. Adafruit dispone de distribución de bundles o paquetes de librerías estandarizadas y aceptadas por CircuitPython, dependiendo de su versión. Estos pueden encontrarse disponibles en la página de CircuitPython.

Estas librerías se instalan en la Raspberry por medio de copiar las carpetas o archivos. mpy directamente en el directorio /lib del almacenamiento interno.

La estructura de archivos es la mostrada en la Figura 74, donde los nombres que comienzan con un slash (/) representan carpetas, mientras que los símbolos (< y >) delimitan descripciones cortas de cada una:

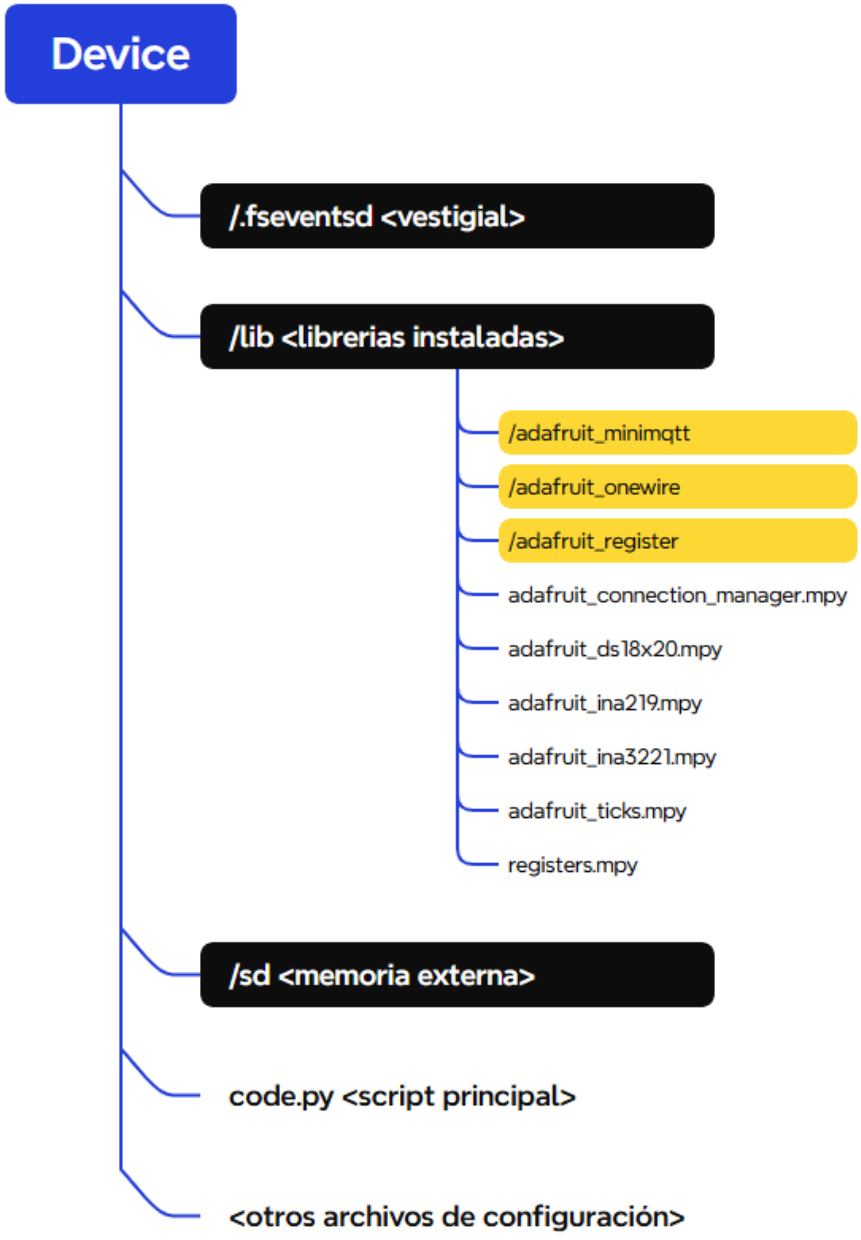


Figura 74: Estructura de archivos en el dispositivo Raspberry Pi Pico W.

Nota: Esta estructura de archivos representa la versión final que se desarrolló para este prototipo. Dos librerías faltantes serían las necesarias para operar los componentes IMU (MPU-92/65) y la cámara (ArduCam 2MP). La omisión de estos dos componentes será explicada más adelante.

6.4.2.1 Implementación general del software de control y comunicaciones

La arquitectura de control y comunicaciones del OBC sigue la misma estructura básica mostrada en la Figura 21. Expandido para incluir la gestión de tiempo (Scheduler) del controlador, y los distintos modos de operación. Estos últimos son:

1. Configuración: Modificar o definir variables de configuración.

2. Recepción: de comandos y procesamiento de estos, como cambios en la configuración o respuesta inmediata a solicitudes en caso de comunicación bidireccional.
3. Lectura: de datos de sensores y dispositivos de captura (cámara), dependiendo del modo de operación.
4. Procesamiento: En caso de ser necesario, procesar datos para ejecutar funciones autónomas o generar el paquete de transmisión.
5. Trasmisión: particularmente a los elementos de comunicación en “tierra”.
6. Espera: hasta el final del ciclo de tiempo definido.

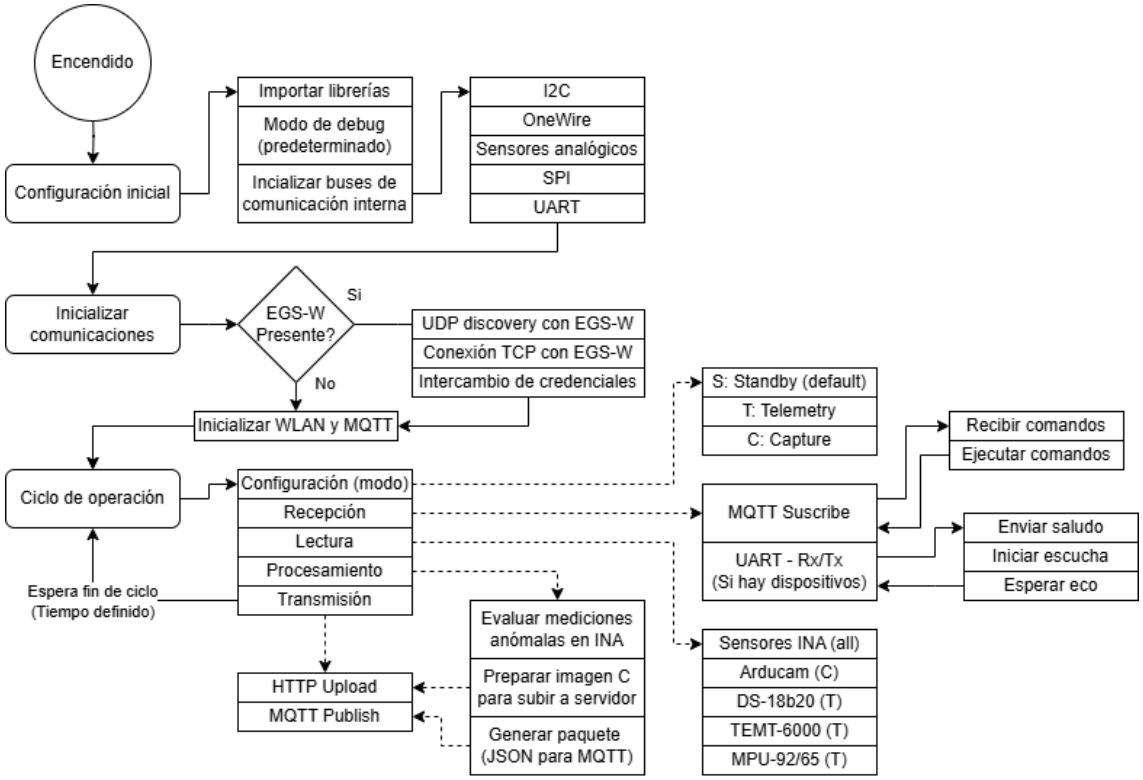


Figura 75: Diagrama general de software de vuelo

Esta arquitectura de software representa una implementación minimalista de los ciclos de Command and Data Handling, y de Communications. Además es importante mencionar que el funcionamiento correcto de este subsistema depende de la una coreografia coordinada con otros elementos del sistema, vease un EGS con capacidades inalámbricas (EGS-W), un servidor o broker MQTT, una red de área local inalámbrica (WLAN), y de un servidor remoto para almacenamiento de archivos mayores (HTTP). Para estos elementos se plantean diferentes métodos de implementación, siendo de la variedad “robusta” o ideal, y “minimalista” o efectivamente implementada.

Una de las características fundacionales del diseño de este sistema corresponde a la capacidad modular de sensores y componentes en los buses de sensores. Esta capacidad fue reforzada por medio de un modo de debug para permitir un diagnóstico rápido de un script, y de una estructura de software que permitiera reportar errores arrojados por sensores ausentes sin detener el código. Una implementación muy simple del registro de debug es el siguiente:

```
DEBUG = True  # Por defecto al programar

def debug(msg):

    if DEBUG:

        print("[DEBUG]", msg)
```

Esta función simplemente detecta si es que el script está corriendo en modo de debug (DEBUG = True, editable a False para desactivar), y si es el caso, presentar un mensaje predefinido. Este mensaje puede ser el texto de una excepción (un error).

Una implementación más robusta de esta función podría utilizar archivos locales en la memoria flash de Raspberry para dejar un registro aun no teniendo acceso a la consola de desarrollo. La misma idea de usar archivos auxiliares para almacenar datos y variables independientemente de la instancia específica del script corriendo se plantea para almacenar variables de configuración, tal como el estado de debug. Este archivo sería leído inmediatamente luego de encender, como parte de la configuración inicial (Figura 75). Todas las funciones que buscan hacer uso de archivos en la memoria flash de la Raspberry fracasaron al intentar ser implementadas y por lo tanto no fueron incluidas en la iteración final.

6.4.2.2 Implementación particular de la cámara ArduCam

La cámara Arducam fue seleccionada debido a dos motivos: La necesidad de tener un Payload óptico en el sistema para enriquecer la experiencia educativa, y la complejidad de integrar un componente como la OV2640 en aislamiento, debido a que la mayoría de los circuitos de demostración en los que se presenta posee un total de hasta 18 pines de conexión. Eso, sumado a los otros sensores presentes en el sistema, representa un cantidad significativa de conexiones que gestionar, y que podrían saturar la cantidad limitada de pines de conexión de la Raspberry Pi Pico W.

En un principio, la Arducam representó una mejora respecto al componente OV2640 solo, debido a que la Arducam integra varios componentes de post-procesado, permitiendo que la interfaz al sistema solo sean 8 pines (6 lógicos: 2 del I2C y 4 SPI), mucho más fácil de integrar mecánica y electrónicamente al sistema.

La complejidad de la integración surge al momento de realizar el software, donde los buses I2C y SPI deben ser configurados siguiendo las limitantes intrínsecas al sistema CircuitPython, donde no se puede simplemente usar cualquier combinación de pines I2C o SPI, sino que un set completo de los predefinidos (Como muestra el diagrama de pinout de la Pico W en la Figura 24).

El problema mayor fue que la librería requerida para la integración en el software de vuelo es desarrollada por ArduCam mismo, y para peor, no han actualizado las librerías para utilizar con Python. La alternativa ofrecida por el foro de soporte de Arducam es cambiar la implementación para utilizar C en lugar de Python. En este punto del proyecto, este cambio no es razonable, debido a que: 1. No poseo experiencia en programar con C, 2. El IDE utilizado (Thonny) no soporta C, 3. El desarrollo en C es significativamente más engorroso que Python, requiriendo un enfoque activo en la gestión de memoria (Python hace esto de forma automática), y una sintaxis menos flexible.

Una sección de las conclusiones incluye una reflexión respecto a la futura implementación de este componente.

6.4.2.3 Implementación particular de sensores asociados al bus I2C

Los sensores asociados al bus de comunicaciones I2C incluyen, además de la ArduCam, ambos sensores INA (219 y 3221) de potencia, y la IMU MPU-92/65.

Los sensores INA, en sus versiones de 1 canal (219) y 3 canales (3221), son utilizados para realizar medidas de consumo eléctrico y verificación de tensión en el ensamble del OBC y EPS respectivamente. Estos se comunican directamente a través del bus I2C para entregar datos de lectura instantáneos (estos sensores realizan una medición continua, pero solo transmiten los datos al ser requeridos por el maestro del bus I2C). Ambos sensores presentan la dirección 0x40 por defecto en el bus, lo que requiere una alteración a las resistencias de configuración de dirección. Dado que la placa INA-3221 utilizada no presenta esta alternativa (a diferencia del componente ofrecido por Adafruit), la INA-219 debería ser modificada. Las direcciones disponibles son 0x41, 0x44 y 0x45, y deben ser coordinadas en un documento de interfaces.

El script utilizado para inicializar el bus I2C fue el siguiente:

```
# ----- SENSOR INIT ------
# bus i2c


i2c = None
mpu9250 = None
arducam = None
try:
    i2c = busio.I2C(board.GP5, board.GP4)   # SCL=GP5, SDA=GP4 on Pico W
    debug("I2C bus initialized.")
except Exception as e:
    debug(f"I2C bus error: {e}")
    i2c = None
    debug("MPU-9250 skipped: I2C pull-ups not present.")
    mpu9250 = None # No fue implementada
    debug("ArduCam skipped: I2C pull-ups not present.")
    arducam = None # No fue implementada
debug("Initializing sensors...")
```

Las librerías requeridas para el uso del bus I2C son busio (BUS-I/O), además de las librerías específicas para cada componente:

```
import busio

from adafruit_ina3221 import INA3221 # 3 canales (bat/3v3/5v)

from adafruit_ina219 import INA219 # 3v3 del OBC (ADCS)
```

En particular el ciclo try-except permite manejar excepciones en la ejecución del código sin detener el proceso. En este caso, si existe un error en la inicialización del bus I2C, simplemente se genera un aviso en la consola (si debug está activado), y define los sensores del bus como no existentes.

La inicialización de los sensores es idéntica ara el caso de la INA 219 e INA3221, esta última se muestra a continuación:

```
# --- INA3221 ---

ina3221 = None # Crear variable para la dirección

if i2c == None:

    debug("I2C not initialized: INA3221 cannot work...")

else:

    try:

        ina3221 = INA3221(i2c)  # Por defecto 0x40, pero esta función la busca igual

        ina3221_s = True

        debug("INA3221 initialized.")

    except Exception as e:

        debug(f"INA3221 error: {e}")

        ina3221 = None
```

En este caso se aprecian dos elementos de control de errores: En caso de que el bus I2C no pudiera inicializarse, el script omite la búsqueda de sensores que dependan de este bus. Además, en caso de sufrir un error durante la inicialización del sensor INA-3221, la excepción sería mostrada como un mensaje del debug, y no detendría la ejecución del programa.

La última parte de la implementación de las INAs, es la función de lectura. Esta se define antes del ciclo principal (main loop) del proceso, y se activa simplemente llamando a la función cuando se requiera.

```
# Función de lectura de las INAs

def read_inas():

    data = {}

    # INA3221: 3 canales, índices: 0–2

    if ina3221: # Solo si la INA3221 fue inicializada

        for ch in range(3):

            try:

                ch_obj = ina3221[ch]

                # Parámetros eléctricos crudos

                data[f"ina3221_ch{ch+1}_bus_v"] = ch_obj.bus_voltage # Respecto a GND

                data[f"ina3221_ch{ch+1}_shunt_v"] = ch_obj.shunt_voltage # Entre in y out

                data[f"ina3221_ch{ch+1}_mA"] = ch_obj.current

            except Exception as e:

                debug(f"INA3221 ch{ch+1} read error: {e}")

    # INA219: Canal único

    if ina219:

        try:

            data["ina219_v"] = ina219.bus_voltage

            data["ina219_sv"] = ina219.shunt_voltage

            data["ina219_mA"] = ina219.current

        except Exception as e:

            debug(f"INA219 read error: {e}")

    return data
```

El resultado de esta función es entregado como una matriz de datos llamada “data”, la cual entrega un conjunto de lecturas crudas de los sensores. Voltajes:

$$V_{bus} = V_{shunt} + V_{load}$$

Donde bus es la diferencia de potencial entre GND y VCC, shunt es la caida de tensión a ambos lados de la resistencia de prueba interna, y load corresponde a la tensión de salida del sensor.

La corriente se mide directamente con un resultado en mA.

La integración del módulo MPU 92/65 no se alcanzó a concretar a tiempo para la implementación de esta iteración del prototipo. En general esto fue debido a dos factores: 1. Que no se consideraba este módulo como uno de los prioritarios para integrar, siendo la Arducam, sensores INA y sensores escalares más importantes desde el punto de vista de la experiencia educativa, y 2. Un error eléctrico crítico durante la fase de integración causó daño sobre el módulo utilizado. El diagnostico de este incidente se encuentra en la sección de resultados.

6.4.2.4 Implementación particular de sensores no asociados al bus I2C

Los sensores que no dependen del bus I2C son ambos sensores escalares: El sensor de temperatura DS-18B20, que corresponde a un transistor que depende del bus OneWire, y el sensor de luminosidad TEMT-6000, el cual es un fototransistor de señal analógica. Estos dos sensores son referenciados colectivamente como “Sensores de telemetría” dentro del código, haciendo alusión a que este es el modo de operación en el cual son activamente referenciados.

Para el software, las librerias requeridas para su integración son las siguientes:

```
import adafruit_onewire.bus # Bus OneWire

import analogio # I/O analógico

# Sensor de temperatura

import adafruit_ds18x20
```

Para inicializar los sensores, estos deben registrar el pin GPIO de la Raspberry que van a utilizar como bus de comunicaciones o datos, indicando la designación interna de GPIO, no la numeración física del Pin. (GPIO-0 o GP15, por ejemplo).

Para el caso de la inicialización del sensor de temperatura (DS-18B20), el script de inicialización se presenta a continuación (página siguiente):

```
# ----- DS18B20 -----
debug("Initializing DS18B20...")
try:
    #GP15 es el pin asignado al bus OneWire.
    ow_pin = board.GP15
    ow_bus = adafruit_onewire.bus.OneWireBus(ow_pin)
    devices = ow_bus.scan()
    if len(devices) == 0:
        debug("No DS18B20 detected.")
        ds18 = None
    else:
        # Para múltiples DS18B20s, se pueden enumerar con una lista.
        ds18 = adafruit_ds18x20.DS18X20(ow_bus, devices[0])
        debug("DS18B20 initialized.")
except Exception as e:
    debug(f"DS18B20 error: {e}")
    ds18 = None
```

Este script define el pin GP-15 (General Purpose) como el bus OneWire requerido por el sensor usado. Al igual que en la implementación del bus I2C, se hace uso de las funciones de debug y el ciclo try-except para evitar errores que paralicen la ejecución del script completo.

Esta función también permite la flexibilidad de integrar múltiples sensores DS-18B20 en paralelo, lo que en este momento no es necesario pero permite hacer pruebas con múltiples sensores si es necesario, particularmente en una protoboard.

Para la inicialización del fototransistor TEMT-6000 se sigue un script similar, aunque con algunas diferencias derivadas de su naturaleza como un transistor de señal analógica en lugar de ser un sensor independiente. El script de inicialización se presenta en la página siguiente:

```
# ----- TEMT6000 -----
debug("Initializing TEMT6000...")
try:
    temt = analogio.AnalogIn(board.GP26)  # ADC0
    debug("TEMT6000 initialized.")
except Exception as e:
    debug(f"TEMT6000 error: {e}")
    temt = None
if temt:
    try:
        raw = temt.value           # 0–65535: valores mínimo y máximo de lectura
        v = temt.value * 3.3 / 65535
        debug(f"TEMT6000: raw={raw}, voltage={v:.3f} V")
    except Exception as e:
        debug(f"TEMT6000 read error: {e}")
```

En este caso el Pin GP26 corresponde al primero de 3 pines con capacidad de lectura analógica de la Raspberry, todos los otros GP son de naturaleza digital. Esto signfica que quien hace la lectura del sensor no es el módulo TEMT-6000 directamente, sino que la propia Raspberry al registrar la tensión recibida en el GP26. Este valor corresponde a un valor de 16 bits (entre 0 y 65535), y representa todos los puntos “continuos” entre los valores mínimo y máximo de tensión (0V a 3.3V). Por este motivo, la lectura “crudo” del sensor de luminosidad está dado por el producto entre el valor de lectura y la tensión máxima dividido sobre el valor máximo de 16 bits.

En este caso, el script realiza una lectura de prueba apenas se inicializa “el sensor”, para confirmar por debug en caso de una anomalía.

Finalmente, la lectura de ambos sensores es implementada en una única función combinada: la función de lectura de telemetría (read_telemetry_sensors). Esta se presenta en el siguiente bloque de código:

```
# Funcion de telemetría
def read_telemetry_sensors():
    data = {}
    if ds18:
        try:
            data["temp_c"] = ds18.temperature
        except Exception as e:
            debug(f"DS18B20 read error: {e}")
    if temt:
        try:
            raw = temt.value
            data["temt_raw"] = raw
            data["temt_v"] = raw * 3.3 / 65535
        except Exception as e:
            debug(f"TEMT6000 read error: {e}")
    return data
```

Esta función entrega la lectura de los sensores de telemetría de la misma manera que la lectura de los sensores INA. Los valores entregados por esta función están en [°C] para el sensor de temperatura (asumiendo que el divisor de voltaje esté implementado correctamente), y como un valor de 16 bits crudo para la TEMT-6000.

Implementación particular de interfaces de comunicación

Las interfaces de comunicación implementadas dentro del software de vuelo dependen significativamente de qué elementos e infraestructura digital está disponible. Siguiendo la lógica de las propuestas “robustas” y “mínimas”, esta sección explicará la implementación mínima al momento de presentar estar iteración del prototipo, además de explicar en términos conceptuales la implementación robusta.

La implementación mínima incluye dos interfaces de comunicación interconectadas: la WLAN y el MQTT. WLAN provee el canal común de comunicaciones (radial), mientras que MQTT representa el protocolo específico utilizado para transmitir telemetría. Una analogía útil en este caso sería comparar WLAN a un cable entre dos terminales eléctricas, mientras que MQTT sería equivalente a un protocolo como OneWire o I2C.

Para hacer uso de estas dos características en el software de vuelo, hace falta importar las siguientes librerías:

```
import os

import wifi

import socketpool

from adafruit_minimqtt.adafruit_minimqtt import MQTT

# MQTT es el protocolo de telemetría rápida
```

Hay dos variedades de librerias MQTT: mini y robust. Para los propósitos de este proyecto, con la librería mini basta y sobra, ya que la alternativa está orientada a funcionar como servidor en lugar de ser solo como cliente. No se quiere que el OBC actúe como servidor o broker del MQTT.

Debido a la mayor extensión de los bloques de código usados para el manejo de los protocolos de comunicación, estos han sido trasladados al Anexo B. Esto incluye el script para el ciclo operativo principal (main loop).

Respecto a las otras características no implementadas, pero que forman parte de la propuesta robusta de este subsistema son

6.5 Diseño e implementación en detalle de la infraestructura de soporte

Como infraestructura de soporte, este capítulo pretende presentar el proceso de diseño, desarrollo y la implementación final de los elementos de infraestructura IT (WLAN, servidores y otros servicios digitales), y el EGS, considerando elementos físicos y lógicos requeridos para las implementaciones mínimas y robustas del sistema.

Lo mínimo requerido para que el kit CubeSat pueda realizar las funciones principales de comunicación para recepción de comandos y telemetría son una red Wi-Fi, y un dispositivo con OS Windows, de acuerdo a las condiciones de diseño prácticamente impuestas durante la realización de esta memoria de título (computador personal con este sistema operativo). La utilización de otros OS es posible, pero no ha sido probado para este sistema.

La función principal de este dispositivo auxiliar es de actuar como servidor de la red MQTT de comunicaciones dedicada especialmente a la telemetría del kit, de acuerdo a lo estipulado en el capítulo 5.

Para esta implementación se utiliza el broker (servidor MQTT) Mosquitto, cuyas ventajas incluyen ser de código abierto y que simplemente puede instalarse usando archivo .exe. Este se puede descargar desde la página oficial y seguir las instrucciones del instalador.

La recomendación es instalar Mosquitto en la ruta <C:\"Program Files (x86)"\mosquitto>, lo que permite ejecutar el script de ejecución presente sin cambios.

Una vez instalado, la carpeta de instalación incluirá un archivo de configuración llamado "mosquitto.conf". Al abrirlo con un editor de texto se deberá cambiar la configuración agregando el puerto designado al broker MQTT (por defecto 1883), agregando o editando la opción justo bajo "General Configurations", y para simplificar el proceso de conexión de dispositivos se modifica la configuración "allow_anonymous true" en la misma sección.

Al terminar, el archivo de configuración se verá así:

```
# Config file for mosquitto
#
# See mosquitto.conf(5) for more information.
#
# Default values are shown, uncomment to change.
#
# Use the # character to indicate a comment, but only if it is the
# very first character on the line.

# =================================================================
# General configuration
# =================================================================

listener 1883

# Use per listener security settings.
#
# It is recommended this option be set before any other options.
#
# If this option is set to true, then all authentication and access control
# options are controlled on a per listener basis. The following options are
# affected:
#
# acl_file
allow_anonymous true
# allow_zero_length_clientid
# auto_id_prefix
# password_file
# plugin
# plugin_opt_*
# psk_file
```

Figura 76: Captura del archivo de configuración del broker MQTT mosquitto.

Para activar el servicio del broker, este requiere iniciarse a través del símbolo de sistema o la consola de windows. Para este propósito se escribió un script simple que se puede ejecutar como un archivo batch o .bat.

```
ipconfig

cd C:\"Program Files (x86)"\mosquitto

mosquitto -v -c mosquitto.conf

@pause
```

Este script permite:

1. Desplegar la información de IP del broker usando el comando “ipconfig”. El broker estará alojado y será alcanzable por medio de la IPV4 presentada en la consola al ejecutar este script.
2. Ejecutar el archivo de configuraciones de mosquitto, lo que permite que efectivamente corra desde el puerto 1883 con las configuraciones seleccionadas.

Si los pasos anteriores se realizaron correctamente, la consola debería mostrar algo como esto:

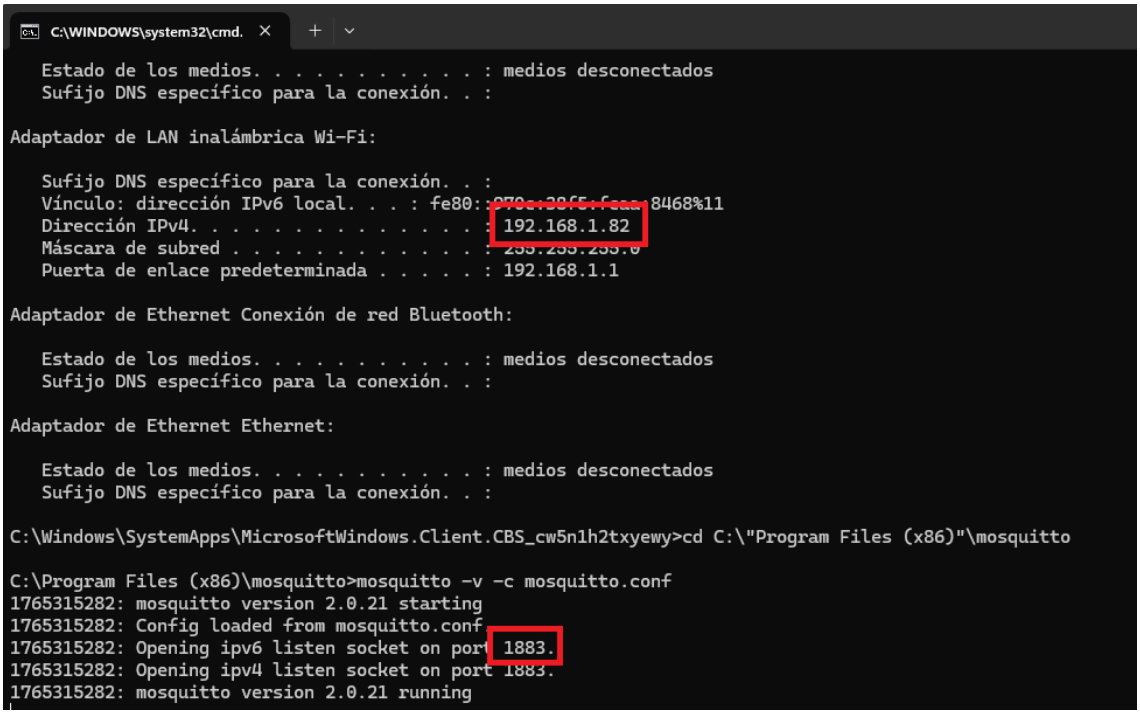


Figura 77: Consola al ejecutar el script de inicialización del broker mosquitto. Notar la IPV4 y puerto utilizados.

Cualquier dispositivo que desee conectarse a la red MQTT se deberá conectar a la red WLAN alojada, y utilizar la IPV4 junto al puerto alojado. Al entrar, estará en condiciones de recibir telemetría suscribiendóse a “device/telemetry”, y podrá enviar comandos publicando a “device/commands”.

Desde telemetry se recibirá un archivo en formato JSON, con datos crudos de la telemetría. Este archivo es legible en un formato de texto, pero al ser integrado a un dashboard donde los datos se quieran graficar, se deberá desempacar el JSON dentro de la aplicación usada como dashboard.

El broker solo actúa como un intermediario en todo este proceso, no envía ni recibe comandos, y solo almacena el último mensaje recibido en cada canal. Los paquetes enviados y recibidos tienen un límite teórico de hasta 256 MB, lo que habitualmente es limitado por el broker por defecto a un tamaño mucho mas bajo, en torno a los 64 KB.

Para conectarse como cliente al broker MQTT se plantean dos métodos:

1. Para dispositivos móviles (Android): Descargar la aplicación IoT MQTT Panel desde la Play Store. Al iniciar la aplicación se deberá establecer la conexión a la IP y puerto que aloja el broker, y posteriormente la aplicación requerirá crear un dashboard para visualizar datos dentro de esta conexión. Esta aplicación soporta múltiples dashboard para la misma conexión, y varias conexiones MQTT simultaneas
2. Para dispositivos de escritorio con Windows, se recomienda el programa MQTT Explorer, que permite conectarse de forma similar a la aplicación IoT MQTT Panel, pero que adicionalmente permite formatear paquetes de datos de forma libre, rastrear todos los tópicos en uso por la red, y graficar los datos directamente en la misma ventana.

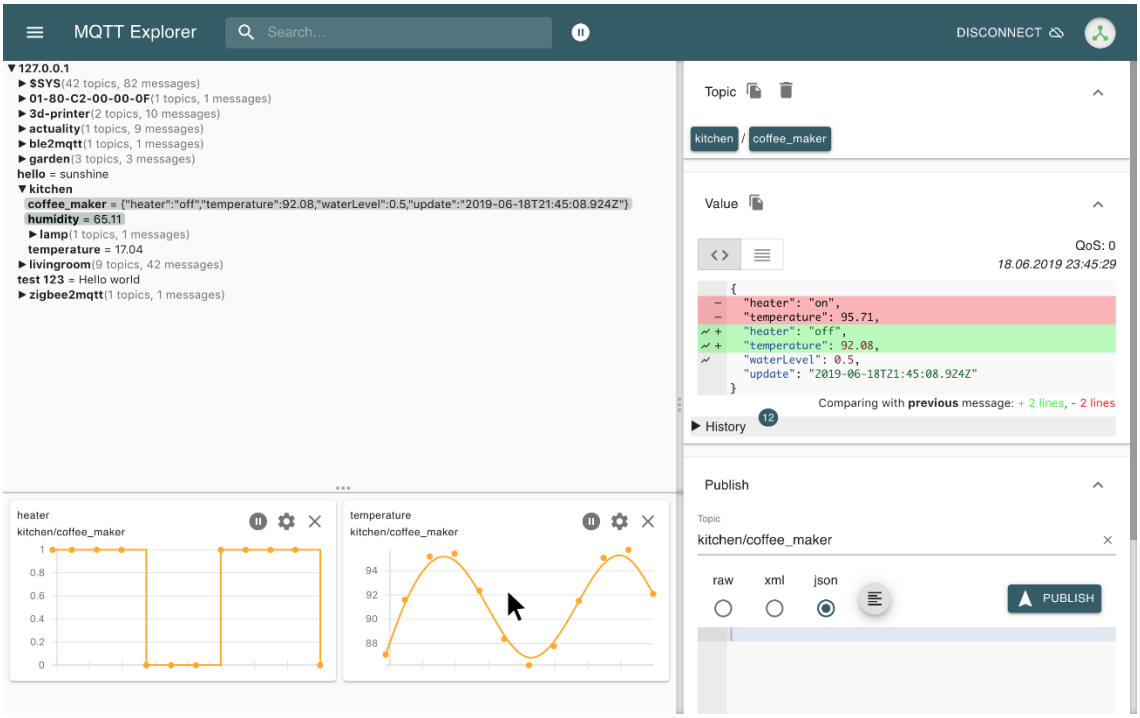


Figura 78: Software MQTT Explorer

7 Prototipo de experiencia

La última fase para la validación de esta iteración del prototipo requiere de la ejecución de una experiencia educativa que permita a los estudiantes interactuar directamente con la herramienta de apoyo. Esto es importante de realizar, debido a que el proyecto está planteado como una experiencia educativa que se apoya con el despliegue de satélites educativos, y no el satélite educativo en sí mismo.

Además, se considera la última etapa de la iteración, definida como un microcosmos del ciclo de vida de un proyecto, comenzada desde el planteamiento conceptual hasta la validación en un entorno operativo de pruebas.

7.1 Generalidades y diseño

La experiencia educativa diseñada se define en primera instancia como un laboratorio práctico del ciclo AIT/AIV. Este debe estar autocontenido, permitir realizar un ciclo que contemple:

1. Aceptación de componentes: No todos los componentes entregados estarán en buenas condiciones o serán los adecuados. Es tarea de los estudiantes realizar pruebas de aceptación contrastando los componentes recibidos con sus características técnicas entregadas, y con las especificaciones de armado del kit.
2. Ensamble, integración y verificación: El ensamblado del kit se realizará siguiendo especificaciones de armado, mientras que el proceso de integración se realizará siguiendo la lógica de la descomposición funcional del sistema. Un kit armado no es lo mismo que un kit integrado, y la experiencia deberá reforzar esta distinción. El proceso para asegurar que el ensamble cumple con integración es la verificación.

Adicionalmente, el tiempo mínimo en que la experiencia debe ser contenida será de 90 minutos. Este debe incluir una presentación del kit, la experiencia y qué esperar, y un cierre de la experiencia.

El diseño de la experiencia consideró, en primer lugar, los objetivos de aprendizaje de la experiencia. Para esta iteración, estos son los siguientes:

1. Presentación del concepto de integración como el proceso a través del cual un conjunto de elementos con funciones granulares se consolidan y permiten la emergencia de una función de nivel superior.
2. La importancia del proceso de aceptación y verificación en cada etapa del proceso.
3. La trazabilidad del proceso AIT.

Adicionalmente, se consideró implementar una variedad de materiales de apoyo para permitir múltiples métodos de aprendizaje, además de aumentar la cantidad de oportunidades para el aprendiza de conceptos. De esta manera se consideró un método de “tridente”, donde la experiencia se sustenta sobre 3 materiales principales:

1. El material de trabajo mismo, el kit de trabajo que incluye los componentes del CubeSat, herramientas de trabajo y de medición directamente relacionados con la experiencia.

Figura 79: Kit completo entregado para el equipo de trabajo durante la experiencia.

2. Una guía de apoyo a disposición de los estudiantes, mostrando una guía paso a paso de los procedimientos a realizar, incluyendo información aclaratoria respecto a instrucciones entregadas, información relevante respecto a componentes o herramientas utilizadas, y material de apoyo como diagramas y planos anotados. Esta se entregó en dos formatos: en una guía física impresa, y una guía digital en PDF (Incluida en el Anexo D)
3. Una presentación en PowerPoint a realizar concurrentemente durante la experiencia. (Incluida en el Anexo E)

Adicionalmente se consideró la aplicación de una encuesta / evaluación respecto a la experiencia al final de esta, pero debido a la longitud del formulario, los estudiantes no alcanzaron a completarla durante el tiempo asignado a la experiencia (90 minutos).

7.2 Experiencia Piloto

Considerando restricciones de diseño e implementación, además de complicaciones emergentes durante la fase de integración del prototipo, la experiencia piloto se planteó como una experiencia acotada en torno a la integración y aceptación del EPS, con la integración física (no lógica) del OBC como un objetivo secundario si el tiempo disponible fuera suficiente.

La experiencia misma fue calendarizada para el Martes 2 de Diciembre del 2025. Esta fue realizada en la sala de clases del segundo piso del LTA, con una participación de un máximo de 4 estudiantes y un mínimo de 3 (uno de los estudiantes debió retirarse temprano para realizar un laboratorio). Esta fue grabada a través de una reunión de Teams, alcanzando a cubrir los temas principales, la integración exitosa del EPS, e incluso la integración parcial del OBC y la estructura. Durante esta experiencia fue que los soportes de la estructura fallaron durante la manipulación de esta.

El tiempo de integración del EPS fue de aproximadamente 40 minutos, incluyendo la aceptación de componentes, verificación de funciones y la verificación del subsistema completo. Los estudiantes expresaron comodidad al poder trabajar con la guía de apoyo, pero en múltiples ocasiones realizaron una integración de los componentes antes de realizar la aceptación, o mediciones para los envelopes (en especial, de masa). Para mitigar esto, la guía requería que en cada etapa de la experiencia los estudiantes registraran valores asociados a la aceptación de los componentes, asegurándose de que la integración se hiciera en orden y de forma segura.

Otros puntos de mejora hechos notar por los estudiantes incluyen:

1. Una elección más apropiada de mesas para el trabajo en grupo. Durante el desarrollo de la experiencia, uno de los estudiantes presentó dificultades para integrarse la flujo de trabajo. Como medidas de adaptación, el estudiante perdía la concentración enfocandose en su teléfono celular, pero eventualmente tomó la iniciativa de realizar un ensamblado en paralelo de la estructura del kit, y eventualmente con la entrega de la guía en PDF, se pudo integrar mejor a la experiencia.
2. Falta una descripción de lo que hace un microcontrolador como la Raspberry Pi Pico W.
3. La inclusión de modelos 3D y animaciones digitales del ensamble son vistas favorablemente por los estudiantes.

Figura 80: Estudiantes durante la experiencia de integración.

8 Evaluación de producción en escala

Para la proyección de producción en escala se toman las siguientes consideraciones:

1. El diseño evaluado corresponde a la iteración actual del prototipo, incluyendo las limitaciones de aplicabilidad expuestos en el Capítulo 6, tales como la limitación de usar la cámara Arducam, o implementaciones robustas de otras características.
2. El ensamble y la fabricación de los kits serán realizados en el Laboratorio de Técnicas Aeroespaciales (LTA) de la UdeC, en Concepción por estudiantes (sin sueldo).
3. Se evalúan batch de fabricación de 5 y 10 kits cada uno.

A partir de esto, se define el siguiente flujo de trabajo:

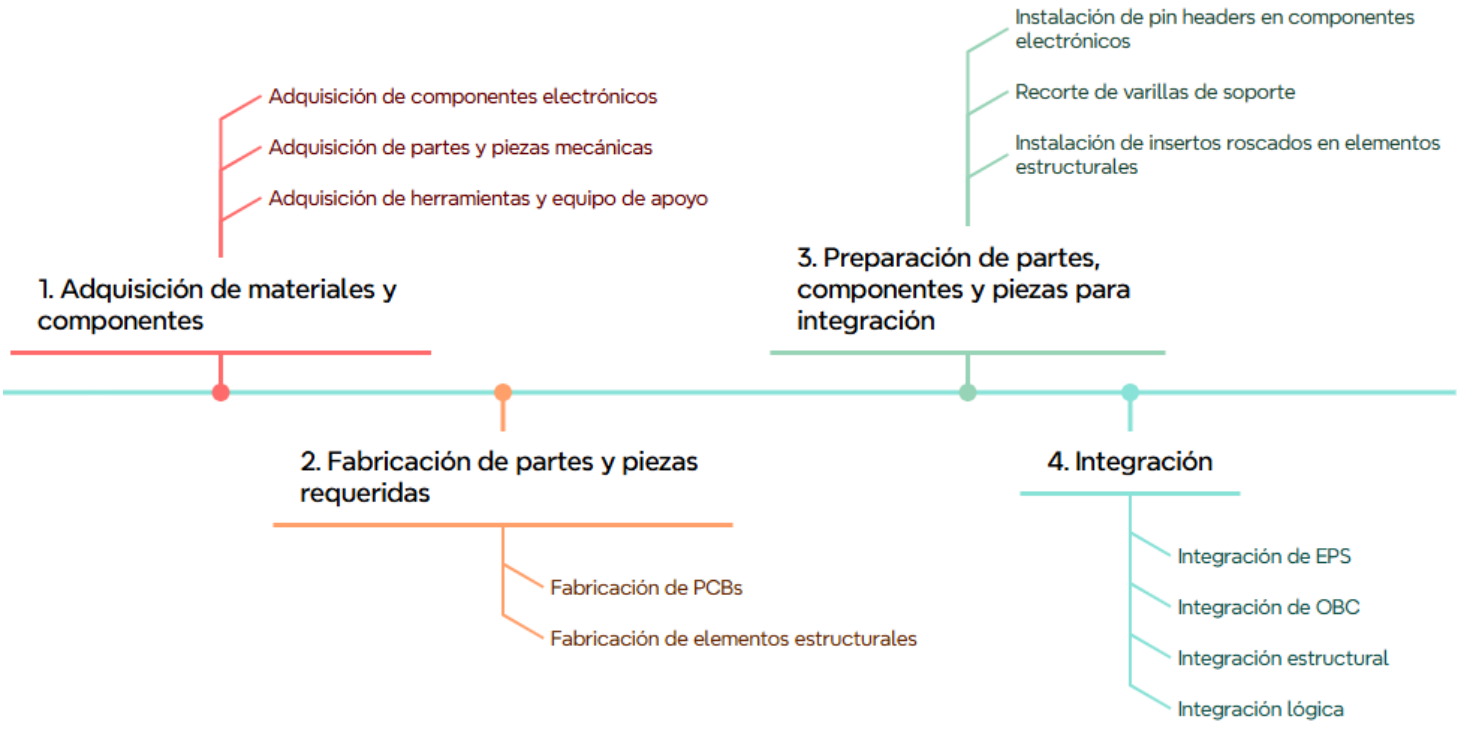


Figura 81: Flujo de producción para prototipo en escala

De esta forma se definen 4 fases de producción:

1. **Adquisición:**

Componentes y partes que se van a comprar. Los drivers principales son el costo y tiempo de entrega. Aquí, la propuesta es simplemente comprarlos desde AliExpress, el cual tiene un costo consistente y comparativamente más bajo que otras alternativas como Mercado Libre o directamente en tiendas especializadas de electrónica, componentes mecánicos o incluso utilería general como cajas plásticas. La excepción a esta preferencia es cuando el componente no está disponible en AliExpress, como es el caso de la ArduCam (que debe comprarse por Amazon).

La lista completa de componentes y piezas que comprar es (por kit):

CubeSat Components and parts:

- 1x Regulador de carga HW-373
- 2x Reguladores DC-DC de 5V
- 1x Reductor de voltaje de 3.3V AMS 1117
- 1x Raspberry Pi Pico W
- 1x ArduCam Mini 2MP Plus. (Amazon)
- 1x 18650 Li-Ion Battery
- 1x 18650 Li-Ion Battery Holder
- 1x INA-219 Power Sensor
- 1x INA-3221 Power Sensor

- 1x MPU-92/65 IMU
- 1x DS-18B20 Temperature Sensor
- 1x TEMT-6000 Phototransistor
- 1x 4.7kΩ Power Resistor
- 3x 1N5817 Schottky Diodes
- 4x 8x2 Stackthrough connector
- 4x 18x2 Stackthrough connector
- 2x 26x2 (or longer) female board connector.
- 1x Switch – push button with 6 connectors
- 4x 40x1 female pinheader
- 12x M3x5.4 Threaded inserts
- 2x Pushrods (at least 180 mm long)
- 8x M3x6 screws

Support Equipment:

- 1x Balanza digital gramera
- 1x Multímetro digital
- 1x Destornillador de cruz corto
- 1x Termómetro digital
- 2x Cajas organizadoras de componentes
- 1x Caja de plástico de almacenamiento

Insumos adicionales:

- Estaño de soldadura

La estimación de costos de esta etapa está considerada en el Anexo C.
Costo unitario: $USD 102.00, costo para 5 kits: $USD 510.01, para 10: $USD 1012.00

El tiempo estimado de entregas para esta etapa es de 15 días calendarios.

2. **Fabricación:**

En particular de los elementos estructurales del marco y los PCBs. El driver principal en este caso son los costos de fabricación y los tiempos de entrega. Los costos de fabricación del marco se calculan considerando que la fabricación de estos elementos se hará usando las impresoras 3D del LTA, mientras que los PCB se considera mandar a fabricar al extranjero. Si bien, en la práctica se puede mandar a fabricar los PCB al C4i, de la misma manera que se fabricaron para la realización de este prototipo, nos fue indicado que la mejor alternativa para fabricar en masa es mandar a hacerlo a través de una página como PCBWay, debido a que los tiempos y costos de fabricación en el extranjero, además de las técnicas disponibles en estos servicios, son más favorables que lo que podrían realizar en el C4i.

El costo de fabricación estimado por PCB Way para la un batch de 10 unidades (10 EPS + 10 OBC) es de $USD 79.54, con el costo de envío de $USD 26.91 ya incluido, y aproximadamente 3 días hábiles para fabricar. Se considera 5 días de envío, con hasta 10 días totales entre el momento de compra y la entrega final.

Finalmente, el costo de fabricación de los elementos estructurales impresos en 3D (con PLA), se estima en $USD 1.14 para un total de 38 g de filamento, y un tiempo de impresión de aproximadamente 5 horas, por kit. Para un total de 5 y 10 kits, el costo aumenta a la estratosférica suma de $USD 5.70 para 5 kits, y $USD 11.4 para 10 kits, con un mínimo de 3 días de fabricación para 5 kits, y 5 para 10.

3. **Preparación**

Consiste en 3 principales labores:

- Recortar las varillas de soporte (Pushrods) a la longitud final, asegurando que solo uno de los extremos presente un hilo. Tiempo total: 15 minutos.
- Realizar soldadura de pinheader y elementos fijos a los PCB y componentes electrónicos: Pinheader macho para los componentes, y hembra para los PCB. Tiempo total: hasta 1 día de trabajo si no hay defectos de fabricación.
- Preparar los elementos estructurales, separándolos de la cama de la impresora 3D, limpiando retazos y desechos de la impresión, e instalando los insertos roscados. Tiempo total: Hasta 30 minutos.

Tiempo total aproximado: 1 día por kit, máximo. Si la soldadura se hace por alguien con experiencia, podría tomar hasta 2 horas como mínimo. Lo más razonable sería estimar 4 horas por kit, si lo hace una sola persona, lo que significan 5 días hábiles de trabajo para una jornada de 8 horas.

4. **Integración**

Particularmente para verificar la correcta instalación de los componentes. En estricto rigor, esta etapa puede ser parte de la experiencia AIT misma, pero se recomienda realizarla al menos una vez antes de hacer la experiencia para asegurarse que los componentes, subsistemas y elementos funcionen correctamente. Es por esto que la integración del Kit completo, asumiendo que no haya errores, y que se pueda instalar el software directamente desde el repositorio sin problemas, sería de alrededor de 2 horas cronológicas.

8.1 Costos y tiempos total estimados

Para las 4 fases de producción, se estima un tiempo mínimo de preparación (para 10 kits) de:

1 día de compra y órdenes de fabricación

15 días aproximado de tiempo de espera (incluye los 5 días mínimo de impresión 3D con una sola impresora, 10 días máximos para la entrega de los PCB)

5 días hábiles (hasta 7 días totales) de preparación y uno a dos días de integración si lo hace una sola persona.

En total: 25 días desde el inicio de la producción podrían dar como resultado 10 kits.

Y el costo total de este proceso, sin considerar HH ni costos de mantención de herramientas ni costo de servicios (como la electricidad), sería de:

$USD 119.15 por unidad, sumando $USD 595.75 para 5 kits, y 1191.50 para 10.

Respecto a la metodología para realizar estos cálculos, la mayoría de los componentes de la primera fase se considera adquirir por Aliexpress. Esta estimación consideró los costos sin oferta, pero de la categoría Choice, lo que elimina el costo de envío cuando el total de compra excede cierto umbral. El único elemento que no es Choice es la TEMT-6000, al cual se calculó el costo sumado al costo de envío. Los costos están obtenidos en $CLP, con la conversión a la fecha 9 de Diciembre del 2025 siendo $CLP 925.93 a $USD 1.00.

9 Modelo Digital

Un modelo digital, en el contexto de la ingeniería de sistemas, se asocia a la práctica de Model Based Systems Engineering (MBSE) o Ingeniería de Sistemas Basada en Modelos. El principal objetivo de esta práctica es generar un modelo digital del sistema de interés donde sus integrantes puedan modificar propiedades o parámetros y evaluar en tiempo real el impacto de estos sobre el sistema completo. Un ejemplo de una decisión arbitraria que podría representarse en un modelo digital sería la decisión entre dos componentes de carga útil óptica. Supongamos que el modelo digital ya está desarrollado, y se busca evaluar como la cámara A más barata, pero más pesada y menos eficiente, se desempeña comparada con una cámara B más costosa, pero superior en desempeño. Un modelo digital podría rastrear los márgenes en tiempo real, y evaluar ambas alternativas de una forma informada en base a un banco de datos actualizados en tiempo real.

En otras palabras, el usuario o ingeniero proyectista sólo debe seleccionar entre la alternativa A y B, y un dashboard le presenta la variación en rendimiento del sistema en tiempo real. Sin el modelo digital, los cálculos deben hacerse manualmente o a través de múltiples software externos, donde la transcripción de datos o parámetros puede verse afectada.

Para realizar un modelo digital conforme a las buenas prácticas de MBSE, este debe contener 3 elementos: Herramienta, Metodología y Lenguaje. Para desarrollar este modelo, la herramienta seleccionada es Violet Labs. Esta corresponde a una herramienta de reporte y procesamiento integrable con terceras aplicaciones a través de integraciones, además de proveer una infraestructura básica de modelado de sistemas.

Una nota importante, es que el acceso a la plataforma Violet Labs está mediado por la creación previa de credenciales de acceso (usuario y contraseña, asociadas a un correo). En caso de querer acceder o hacer uso de este modelo, se debe invitar y gestionar la creación de las cuentas de los participantes de forma anticipada.

9.1 Misión (Program)

Violet Labs rastrea proyectos, misiones o sistemas en la forma de "Programas" o *Programs* dentro de la aplicación web. Estos permiten determinar membresía, autoría y arquitectura de un sistema. Para este desarrollo, este ha sido guardado como un programa con el título "UdeC 2025-2 MT DGG – HAISE-Sat 2"[8].

Dentro de la ventana de edición del programa se puede integrar la arquitectura del sistema, la cual será utilizada como guía para la integración de parámetros, requerimientos y scripts. En esta primera versión del modelo digital, este programa incluye una jerarquía limitada al primer segmento de la misión, el Kit CubeSat mismo. Más aun, para ajustarse con la escala reducida de la experiencia piloto y no sobrecargar la plataforma con conjuntos de parámetros, el subsistema de potencia eléctrica o EPS ha sido el enfoque principal del modelado. Este incluye 5 ensambles, y un total de 12 componentes individuales, todos aportando un valor de masa y costo. Una representación de la arquitectura cargada en Violet Labs se demuestra en la Figura 82.

[8] Si como lector posee acceso a la plataforma Violet Labs, este modelo puede encontrarse en la página udec.violetlabs.com

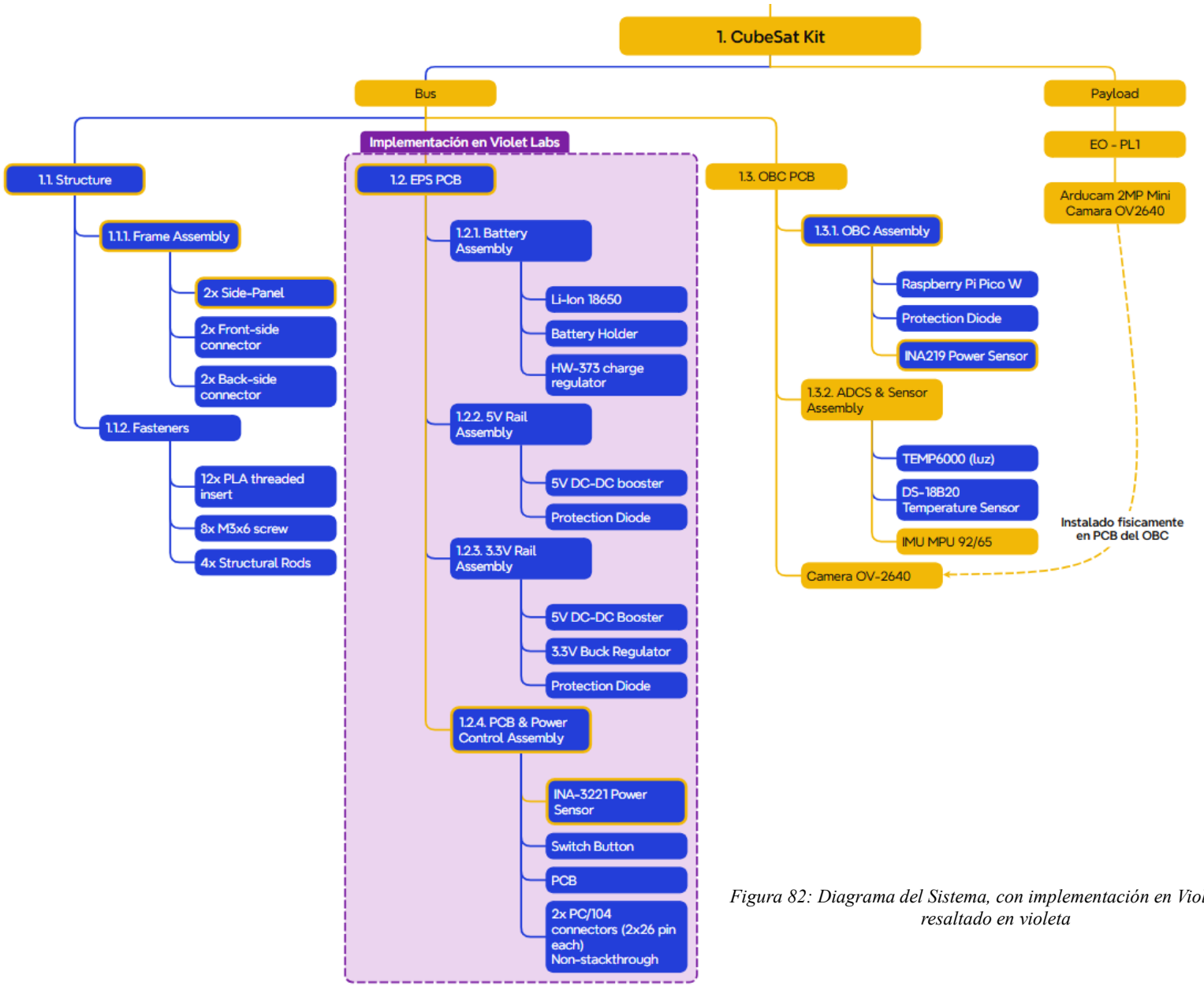


Figura 82: Diagrama del Sistema, con implementación en Violet Labs resaltado en violeta

9.2 Requirements

Violet Labs también permite la integración de requerimientos de sistema, tal como los presentados en la Sección 3.1.1. de este documento. De esta forma, los 7 principales requerimientos de la misión han sido cargados en el modelo digital como requerimientos de alto nivel. Adicionalmente, algunos de estos requerimientos han sido descompuesto y se han generado requerimientos “anidados” o hijos, los cuales se han asignado a segmentos. Sucesivas descomposiciones de requerimientos pueden ser asignadas a otras partes de la arquitectura del sistema. Eventualmente, estos requerimientos pueden ser vinculados para ser verificados o validados automáticamente en la medida que sus parámetros asociados se actualicen automáticamente.

9.3 Parameters & Scripts

Las últimas dos características de interés dentro de la plataforma de Violet Labs, y las cuales representan el corazón operativo del modelo digital, corresponden a los conjuntos de parámetros y los scripts.

Los conjuntos de parámetros permiten definir entidades llamadas parámetros como propiedades de un sistema. Estos parámetros pueden ser propiedades numéricas, físicas o cualitativas, a las que se les puede asignar un valor (numérico o literal) y una unidad de medida. Un ejemplo, sería la masa de un componente, cuya unidad de medida puede ser gramos (g).

Los parámetros asociados al modelo digital de este sistema están guardados bajo la carpeta “2025 MT-DGG Sat-2”.

Debido a limitaciones con la función de anidado de parámetros, cada conjunto de parámetros está guardado al mismo “nivel” dentro de la carpeta. Para facilitar el orden dentro de esta carpeta, se ha implementado una nomenclatura que incluye el identificador numérico de cada componente, elemento y subsistema modelado, lo que asegura que al menos exista un orden en la lista de parámetros.

Los parámetros rastreados más relevantes en la primera versión de este modelo son la masa y el costo. Todos los componentes del EPS tienen estas propiedades asignadas.

Para poder manejar estas propiedades, y realizar comprobaciones, cálculos y operaciones más complejas, se utiliza la función de scripts de Violet Labs. Esta permite el uso de scripts de código en Python (o Julia) para generar salidas. Los scripts funcionan a través de una integración con JupyterLab, lo que permite trabajar con celdas de entrada o salida. Las celdas de entrada poseen código, mientras las celdas de salida reciben la salida de consola.

Uno puede importar entidades o parámetros desde Violet Labs al script usando la función “Violet[“var”]”, donde “var” es el nombre de la variable. Al hacer esto, Violet requerirá que se vincule un parámetro o entidad de entrada, por ejemplo, la masa de un componente.

Posteriormente, el script se ejecutará, y la celda de salida presentará cualquier output del código como si fuera una terminal de consola. Es importante señalar que para vincular la salida de una celda de script como un parámetro de salida, solo un parámetro puede ser vinculado por celda, ya que Violet toma el output completo de una celda como el valor de entrada al parámetro de salida.

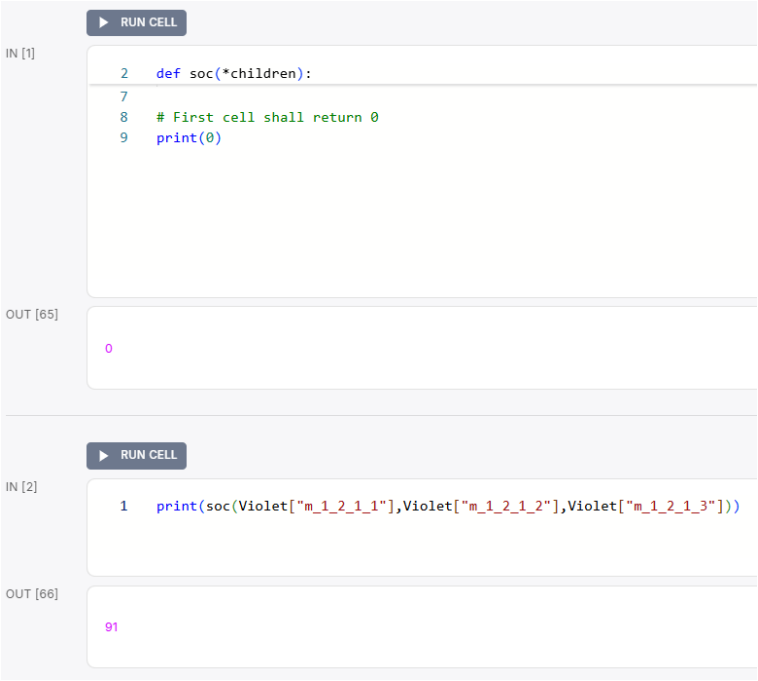


Figura 83: Captura de un script de "suma de hijos" en el modelo digital de Violet Labs.

Respecto a la idea de “Suma de Hijos”, esto representa un concepto de herencia de parámetros en jerarquía de sistemas. La masa del objeto “padre” corresponde a la suma de las masas de todos sus objetos “hijos”. Un ejemplo graficado de esto se aprecia en la Figura 84.

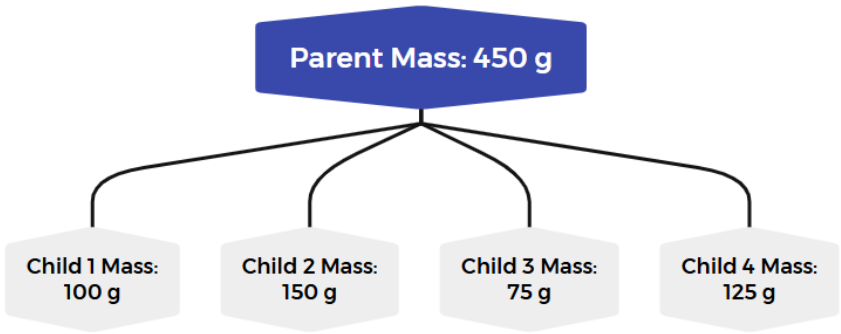


Figura 84: Representación gráfica de una suma de hijos.

10 Conclusiones

Por un lado, la ejecución de una iteración completa de diseño, implementación, integración y validación del prototipo de experiencia se considera un éxito respecto a los entregables. Por otro lado, es imposible no notar el fallo reiterado al intentar implementar funciones que han fallado en distintas iteraciones desde el inicio de este proyecto. Una de estas siendo la incapacidad de hacer funcionar un payload óptico, en un principio por una electrónica engorrosa dada la complejidad de integrar un componente poco amistoso como son las cámaras OV2640, pero en esta ocasión por la complejidad de intentar integrar un componente que digitalmente es mucho más engorroso de integrar con el sistema operativo utilizado. Otras falencias son la no integración de un EGS al sistema, a pesar de contar con el diseño conceptual de este.

Respecto a los mayores éxitos de este proceso, se considera la fabricación de un kit de satélite completo y cuya funcionalidad global de ser una herramienta de apoyo a la educación de ingeniería de sistemas ha sido demostrada en la experiencia piloto. Aun así, sería ideal poder realizar más iteraciones de este prototipo para validar completamente el diseño y su aplicabilidad en aulas.

Otro punto fuerte del diseño de esta iteración fue el enfoque en medidas de seguridad, especialmente en los subsistemas EPS y OBC. El primero se sometió a prueba cuando un error en la fase de integración resultó en la instalación incorrecta del módulo DS-18B20, causando un cortocircuito. En condiciones sin proteger, esto podría haber causado un accidente con la batería, o la destrucción de componentes reguladores en el EPS. En este caso, el limitador de corriente integrado en la HW-373 evitó que la corriente total del cortocircuito fuera crítica, con el resultado de que incluso la DS-18B20 erróneamente instalada resultase ilesa a pesar de haberse sobrecalentado. La otra medida que resultó exitosa fue la implementación de un modo de desarrollo o debug y la gestión de excepciones en el software de vuelo en la OBC, lo que simplificó los ciclos de prueba una vez que estos fueron condensados en la fase de integración.

La implementación de un modelo digital del sistema ha sido desafiante, en gran parte debido a las dificultades operativas y restricciones inherentes de la herramienta utilizada. Violet Labs permite realizar todo lo solicitado, en teoría, pero debido a que aun es una herramienta en desarrollo cuyo principal objetivo no es realizar modelado MBSE directamente, sino que facilitar la integración entre otros programas que operan dentro de la lógica MBSE, algunas de las fases del diseño del modelo resultan ser engorrosas o directamente contraintuitivas. Sin embargo, el resultado de esto ha sido una primera versión del modelo digital que eventualmente podría apoyar el desarrollo de la actividad propuesta en la experiencia piloto. Esta, aunque limitada, permite presentar conceptos como la “suma de hijos” o la validación de envelopes y presupuestos de diseño.

Finalmente, se solicitó que se evaluara la factibilidad logística y económica de desarrollar este sistema de apoyo a la educación con un presupuesto acotado y en lotes de producción de entre 5 y 10 kits cada una. Esto se consiguió, cumpliendo holgadamente con las limitantes de presupuesto impuestas, y consiguiendo un tiempo de implementación que no resulta ser excesivo (menos de un mes).

Algunas reflexiones finales respecto a todo lo anterior serían que la factibilidad y la utilidad de esta herramienta no son una fantasía. La diferencia en los costos de implementación actuales podría ser utilizada para mejorar sin problemas las falencias encontradas.

Otro punto importante que debe ser mencionado, y que a nivel personal ha sido relevante, es la prevalencia de problemas de dominio específico presentes en esta implementación. En particular han sido desafiantes los problemas al diseñar e integrar elementos electrónicos e informáticos, lo que sumado a la plétora de problemas de diseño encontrados, y al enfoque generalista asumido como parte de la aplicación de ingeniería de sistemas, indican que el desarrollo continuado de este proyecto podría beneficiarse enormemente de la formación de un equipo de trabajo multidisciplinario, especialmente la inclusión de un ingeniero electrónico y un ingeniero informático.

11 Referencias

Aboaf, A. P., Harrod, E. S., Zola, M., Prakash, A., Palo, S. E., Marshall, R., Pilinski, M. D., Rainville, N., Dahir, A., Nataraja, V., Schawb, B., Gardell, A., & Warshaw, L. (2020). A Methodology for Successful University Graduate CubeSat Programs. *34th Annual AIAA/USU Conference on Small Satellites*.

Binder, D., Smith, E. C., & Holman, A. B. (1975). Satellite Anomalies from Galactic Cosmic Rays. *IEEE Transactions on Nuclear Science*, *22*(6), 2675–2680. https://doi.org/10.1109/TNS.1975.4328188

Bouzoukis, K.-P., Moraitis, G., Kostopoulos, V., & Lappas, V. (2025). An Overview of CubeSat Missions and Applications. *Aerospace*, *12*(6), 550. https://doi.org/10.3390/aerospace12060550

Chechile, I. (2021). *La Ciencia Dura: Pensamiento sistémico, mitos y verdades sobre estudiar Ingeniería*.

COPE Council. (2024). *Authorship and AI tools*. https://doi.org/10.24318/cCVRZBms

Crawley, E. F., Malmqvist, J., Östlund, S., & Brodeur, D. R. (2007). *The CDIO Approach - Rethinking Engineering Education*.

ECSS Secretariat. (2012). *ECSS System - Glossary of terms*.

Elsevier. (2025, noviembre 12). *Generative AI Policies for Journals*. https://revistas.udec.cl/index.php/gyap/PoliticaAI

EyasSat LLC. (2011). *EyasSat User's Manual (5.0)*. www.EyasSat.com

FACH. (2023, diciembre 18). *CanSat ALFA 2023: Escolares Chilenos Lanzan 10 Satélites Educativos | Fuerza Aérea de Chile*. https://fach.mil.cl/finaliza-el-programa-de-educacion-espacial-escolar-cansat-alfa-2023

Faure, P., Tanaka, A., & Cho, M. (2017). Toward lean satellites reliability improvement using HORYU-IV project as case study. *Acta Astronautica*, *133*, 33–49. https://doi.org/10.1016/j.actaastro.2016.12.030

Friedlander, D. (2018, mayo 21). *COTS in space: the radiation barrier*. Passive Components Blog.

Gansmoe, T., Mathisen, S. V., Grande, J., Dalsgaard, J. F., & Rossing, N. K. (2015). *The CanSat Book* (2.12).

Golkar, A. (2020). Experiential Systems Engineering Education Concept Using Stratospheric Balloon Missions. *IEEE Systems Journal*, *14*(2), 1558–1567. https://doi.org/10.1109/JSYST.2019.2917823

González, D. (2025). *Desarrollo de un sistema de apoyo a la educación en ingeniería de sistemas en base a satélites educacionales*. Universidad de Concepción.

GYAP. (2025). *Política de uso de Inteligencia Artificial (IA)*. https://revistas.udec.cl/index.php/gyap/PoliticaAI

INCOSE. (2023). *INCOSE Systems Engineering Handbook, 5th Edition*.

KISPE. (2023). *Essential Sat Explained | John Paffett Managing Director | KISPE | Ignite Space*. https://www.youtube.com/watch?v=rNLXLSaJ1TI

Kolb, D. . (1984). *Experiential learning*. Prentice Hall.

Kulu, E. (2025, abril 30). *Nanosats Database. Figures*. https://www.nanosats.eu/#figures

Loveless, T. D. (2018). *Space Systems Engineering in Undergraduate Education*.

Merstallinger, A., Sales, M., Semerand, E., & Dunn, B. D. (2009). Assessment of Cold Welding between Separable Contact Surfaces due to Impact and Fretting under Vacuum. *ESA STM*, *279*. https://esamultimedia.esa.int/multimedia/publications/STM-279/STM-279.pdf

MPJA. (2019). *35647-MP*. https://www.mpja.com/download/35647mpdata.pdf

Muller, G., & Bonnema, G. M. (2013). *Teaching Systems Engineering to Undergraduates; Experiences and Considerations*.

NASA. (2018). *Mars InSight Launch Press Kit*. https://www.jpl.nasa.gov/news/press_kits/insight/launch/download/mars_insight_launch_press kit.pdf

OMG. (2023). *CubeSat System Reference Model Profile, v1.0*. https://www.omg.org/spec/CSRM/

Orion Space. (2024). *CUBEEK Cubesat Educational Kit*. Tindie.com.

PC/104 Specification Version 2.6. (2008).

Pisacane, V. L. (2005). *Fundamentals of Space Systems* (2nd ed.).

Salas, A., Farias, O., Tinapp, F., Vasquez, F., & Leaman, F. (2024). Experiencia de asignatura integradora aplicando metodo-logía CDIO a estudiantes de Ingeniería Mecánica Experience of a integrative subject applying CDIO meth-odology to Mechanical Engineering students. En *Revista Iberoamericana de Ingeniería Mecánica* (Vol. 28, Número 2).

Schilling, K. (2024). Hands-On Education for Smart, Small, Self-Organizing Satellite Systems in "New Space". *IFAC-PapersOnLine*, *58*(16), 169–174. https://doi.org/10.1016/J.IFACOL.2024.08.481

The CubeSat Program. (2022). *CubeSat Design Specification Rev. 14.1*. Cal Poly.

Villarroel, B. (2023). *Integración de un CubeSat bus diseñado como herramienta de aprendizaje en ingeniería de sistemas espaciales HAISE-Sat.*

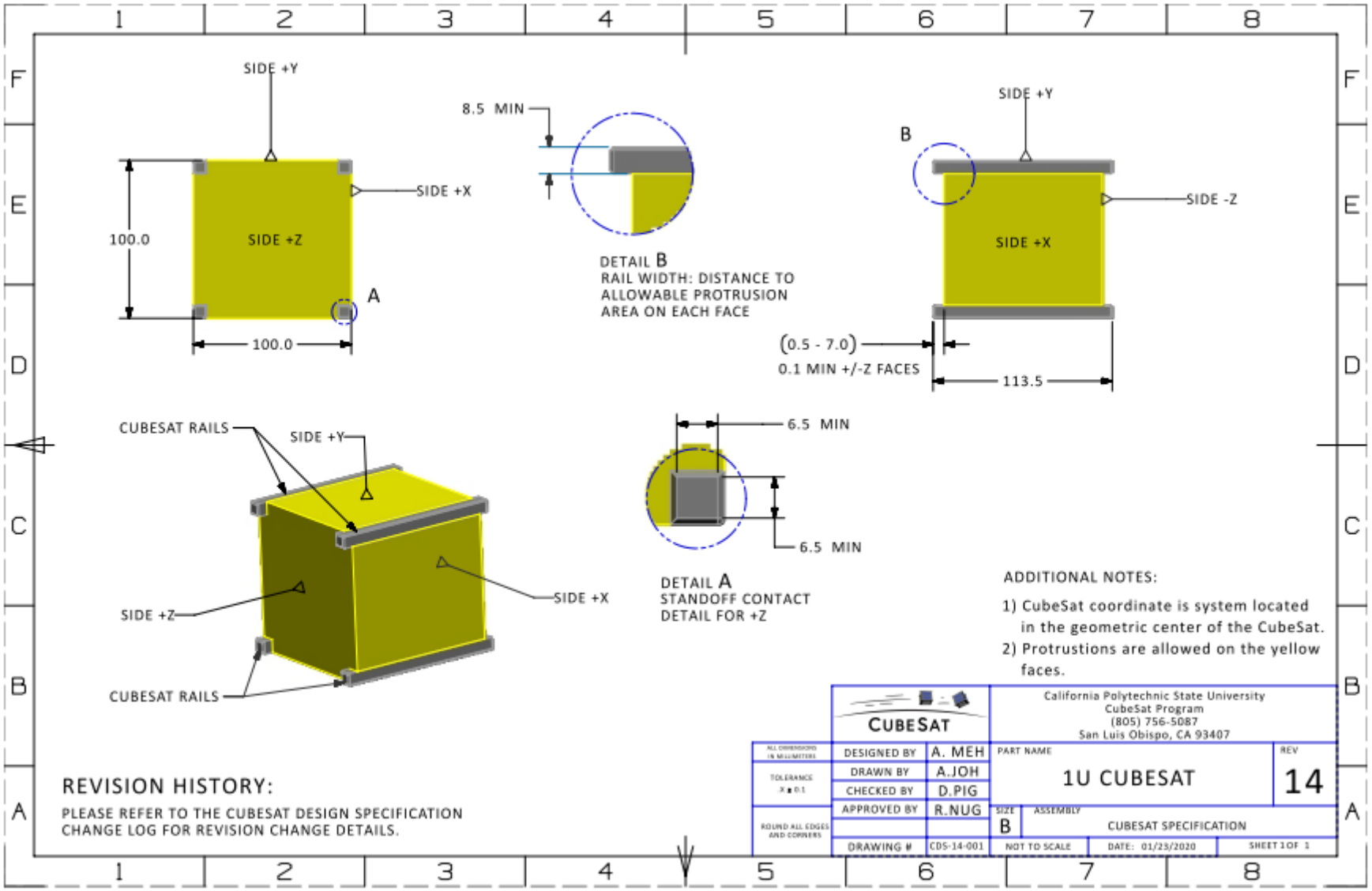
Yablonski, J. (2024, febrero 12). *Tesler's Law*. Laws of UX. https://lawsofux.com/articles/2024/teslers-law

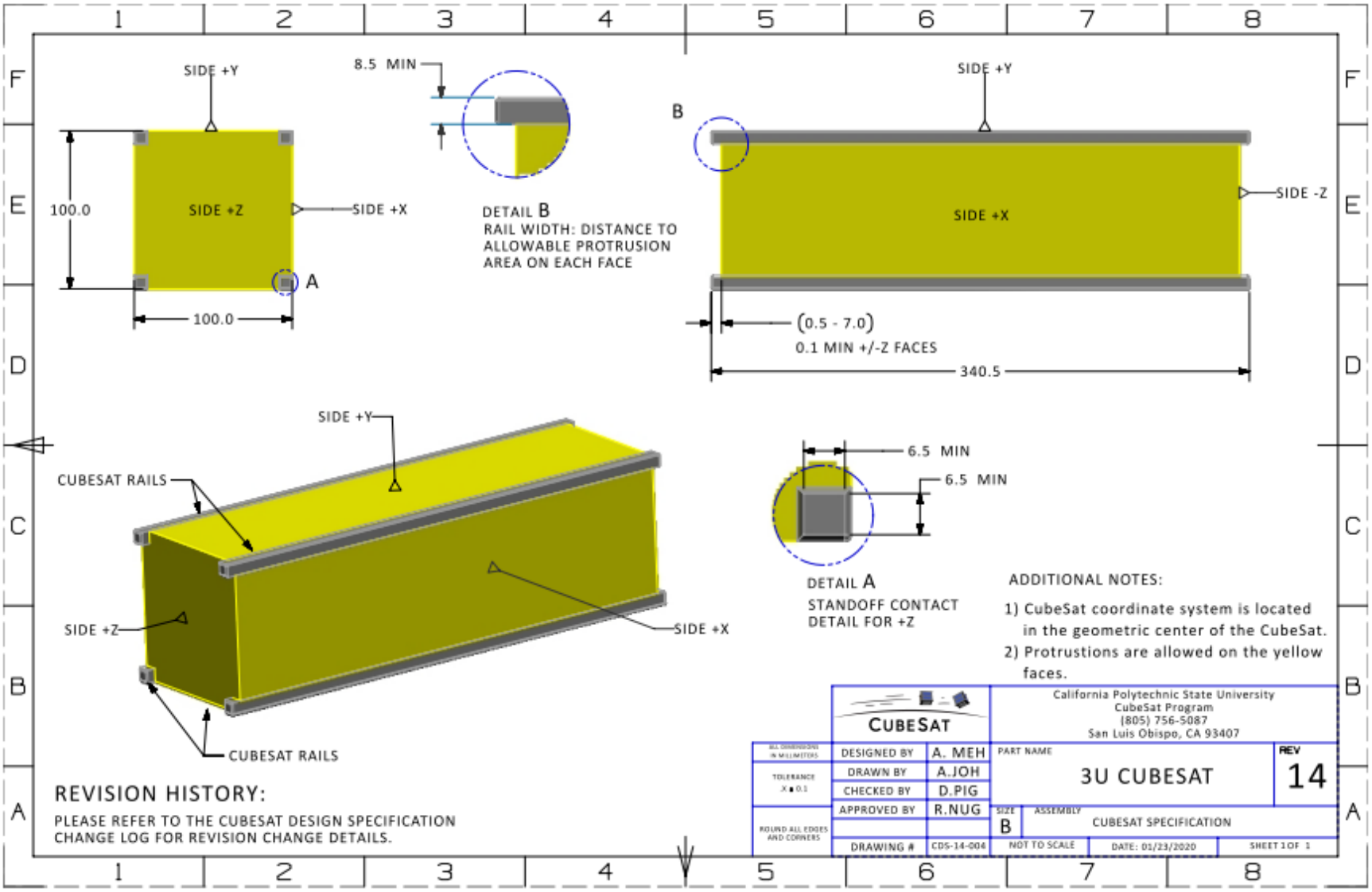
Anexo A: Planos y especificaciones mecánicas

Este Anexo incluye planos y diagramas esquemáticos de componentes mecánicos relacionados al sistema, incluyendo especificaciones estandarizadas.

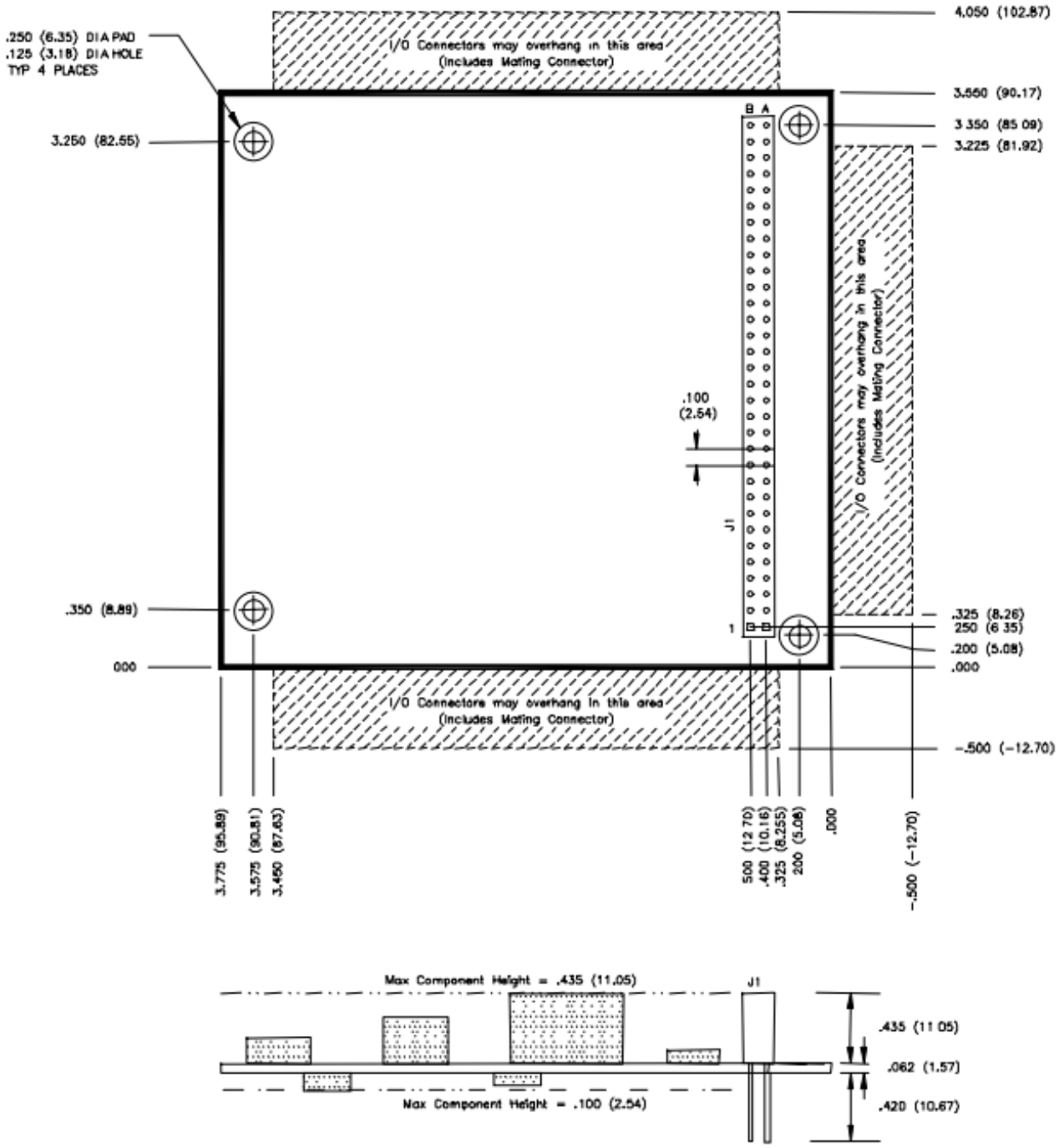
- Especificaciones mecánicas del CubeSat Design Specification, CDS.[9]
- Especificaciones mecánicas del estándar PC/104.[10]

[9] (The CubeSat Program, 2022)
[10] (PC/104 Specification Version 2.6, 2008)

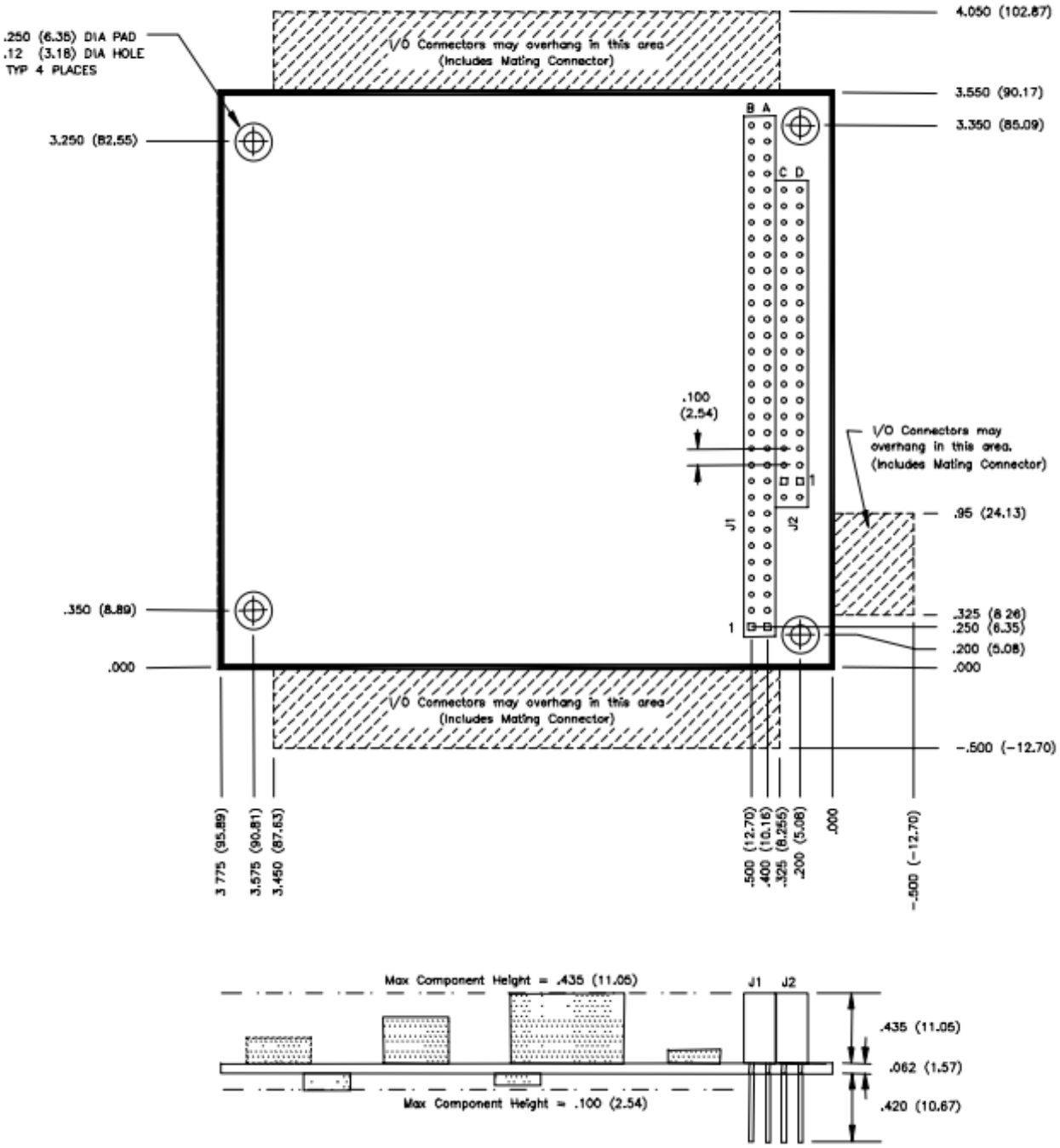

SIDE +Y
SIDE +X
SIDE +Z
100.0
100.0
A
8.5 MIN
DETAIL B
RAIL WIDTH: DISTANCE TO
ALLOWABLE PROTRUSION
AREA ON EACH FACE
SIDE +Y
B
SIDE -Z
SIDE +X
(0.5 - 7.0)
0.1 MIN +/-Z FACES
113.5
CUBESAT RAILS
SIDE +Y
SIDE +Z
SIDE +X
CUBESAT RAILS
6.5 MIN
6.5 MIN
DETAIL A
STANDOFF CONTACT
DETAIL FOR +Z
ADDITIONAL NOTES:
1) CubeSat coordinate is system located in the geometric center of the CubeSat.
2) Protrustions are allowed on the yellow faces.
REVISION HISTORY:
PLEASE REFER TO THE CUBESAT DESIGN SPECIFICATION
CHANGE LOG FOR REVISION CHANGE DETAILS.
CUBESAT
California Polytechnic State University
CubeSat Program
(805) 756-5087
San Luis Obispo, CA 93407
DESIGNED BY A. MEH
DRAWN BY A.JOH
CHECKED BY D.PIG
APPROVED BY R.NUG
PART NAME
1U CUBESAT
REV
14
SIZE B
ASSEMBLY
CUBESAT SPECIFICATION
DRAWING # CDS-14-001
NOT TO SCALE
DATE: 01/23/2020
SHEET 1 OF 1


SIDE +Y
8.5 MIN
SIDE +Z
SIDE +X
100.0
100.0
A
DETAIL B
RAIL WIDTH: DISTANCE TO
ALLOWABLE PROTRUSION
AREA ON EACH FACE
B
SIDE +Y
SIDE +X
SIDE -Z
(0.5 - 7.0)
0.1 MIN +/-Z FACES
340.5
SIDE +Y
CUBESAT RAILS
SIDE +Z
SIDE +X
CUBESAT RAILS
6.5 MIN
6.5 MIN
DETAIL A
STANDOFF CONTACT
DETAIL FOR +Z
ADDITIONAL NOTES:
1) CubeSat coordinate system is located in the geometric center of the CubeSat.
2) Protrustions are allowed on the yellow faces.
REVISION HISTORY:
PLEASE REFER TO THE CUBESAT DESIGN SPECIFICATION
CHANGE LOG FOR REVISION CHANGE DETAILS.
CUBESAT
California Polytechnic State University
CubeSat Program
(805) 756-5087
San Luis Obispo, CA 93407
DESIGNED BY A. MEH
DRAWN BY A.JOH
CHECKED BY D.PIG
APPROVED BY R.NUG
DRAWING # CDS-14-004
PART NAME
3U CUBESAT
REV
14
SIZE B
ASSEMBLY
CUBESAT SPECIFICATION
NOT TO SCALE
DATE: 01/23/2020
SHEET 1 OF 1

Dimensions are in inches / (millimeters)



Dimensions are in inches / (millimeters)



Anexo B: Scripts mayores de código implementado

Este anexo incluye snippets de código utilizados para:

- Configurar e inicializar los clientes WLAN y MQTT del OBC.
- Definir función de transmisión de telemetría a través de MQTT
- Establecer el ciclo del software principal

B.1: Configuración de redes WLAN y MQTT

```
# ---------- CONFIG ----------

WIFI_SSID = "---"

WIFI_PASSWORD = "---"

MQTT_BROKER = "---"

MQTT_PORT = 1883 # por defecto en brokers MQTT

# ---------- SETUP WIFI + MQTT ----------

# Inicialización de wifi

def connect_wifi():

    debug("Connecting to WiFi...")

    wifi.radio.connect(WIFI_SSID, WIFI_PASSWORD)

    debug("WiFi connected...")

# Nota, esta implementación corre el riesgo de quedarse atascada eternamente en
intentar conectarse a una red

# Inizialización de MQTT

def setup_mqtt():

    pool = socketpool.SocketPool(wifi.radio)

    mqtt_client = MQTT(

        broker=MQTT_BROKER,

        port=MQTT_PORT,

        socket_pool=pool,

        keep_alive=60

    )

    debug("MQTT client ready.")

    return mqtt_client

# Modos de operación

MODE_STANDBY = 0

MODE_TELEMETRY = 1

MODE_CAPTURE = 2
```

B.2: Definición de función gestora de comandos (recepción) del MQTT

```
# Gestor de comandos por MQTT
def handle_command(cmd):
    cmd = cmd.strip().upper()
    global current_mode
    if cmd == "STANDBY":
        current_mode = MODE_STANDBY
        debug("Mode set to STANDBY.")
    elif cmd == "TELEMETRY":
        current_mode = MODE_TELEMETRY
        debug("Mode set to TELEMETRY.")
    elif cmd == "CAPTURE":
        current_mode = MODE_CAPTURE
        debug("Mode set to CAPTURE.")
    else:
        debug(f"Unknown command received: {cmd}")
```

Nota: Esta iteración del OBC tan solo está programada para recibir 3 comandos diferentes, los cuales solo cambian el modo de operación del CubeSat.

B.3: Definición de función de publicación (transmisión) del MQTT.

```
# Función de publicación o transmisión de telemetría
def publish_packet(packet: dict):
    try:
        mqtt.publish("device/telemetry", str(packet))
    except Exception as e:
        debug(f"MQTT publish error: {e}")
```

B.4: Ciclo operativo principal (main loop) del software de vuelo. (Parte 1)

```
# ----- Main Loop -----

connect_wifi()

mqtt = setup_mqtt()

mqtt.on_message = lambda client, topic, msg: handle_command(msg)

debug("Connecting MQTT...")

mqtt.connect()

mqtt.subscribe("device/commands")

debug("MQTT connected and subscribed.")

last_cycle = time.monotonic()

current_mode = MODE_STANDBY

while True:

    now = time.monotonic()

    elapsed = now - last_cycle

    # ----- 1. Recibir  -----

    try:

        mqtt.loop()

    except Exception as e:

        debug(f"MQTT loop error: {e}")
```

B.5: Continuación:

```
    # ----- 2. Gestión de comandos y lectura -----
    if current_mode == 0:
        cycle_time = 1.0  # segundos
        data = read_inas()
    elif current_mode == 1:
        cycle_time = 1.0
        data = {}
        data.update(read_inas())
        data.update(read_telemetry_sensors())
    elif current_mode == 2:
        cycle_time = 4.0
        data = read_inas()
        perform_capture_cycle()
        # Volver a standby automáticamente
        current_mode = MODE_STANDBY
        debug("Capture complete. Returning to STANDBY.")
    # ----- 3. Procesar (crear JSON) -----
    packet = {
        "mode": current_mode,
        "uptime": time.monotonic(),
        "data": data
    }
    # ----- 4. Transmitir -----
    debug(f"Publishing packet: {packet}")
    publish_packet(packet)
    # ----- 5. Esperar -----
    if elapsed < cycle_time:
        time.sleep(cycle_time - elapsed)
    last_cycle = time.monotonic()
```

Anexo C: Detalle de estimación de costos de producción

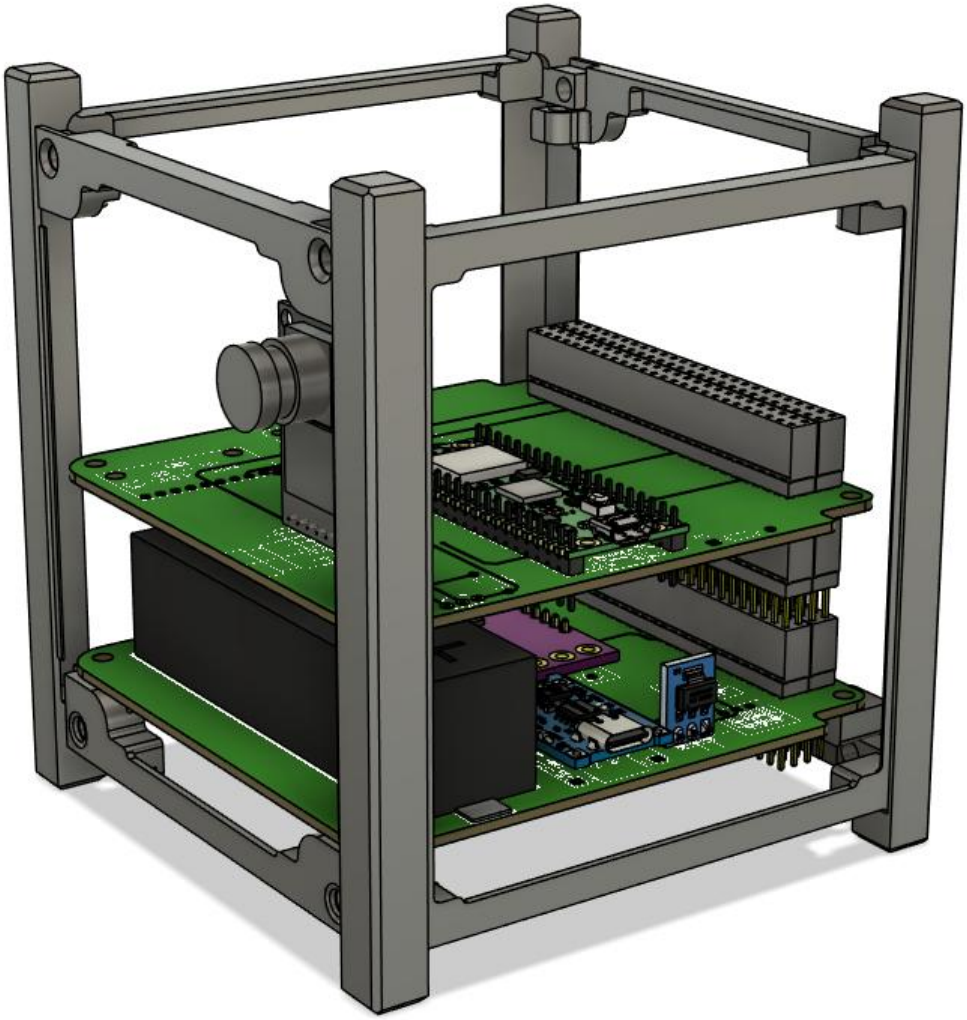
Item	$CLP/bundle	$USD/bundle	u/bundle	$CLP/u	$USD/u	u/kit	Link	USD / 5 kit	USD / 10 kit	CLP / 5 kit	CLP / 10 kit
CubeSat Components and parts:											
- 1x Regulador de carga HW-373	$ 8.090	$ 8,74	10	$ 809	$ 0,87	1	Link	$ 4,37	$ 8,74	$ 4.045	$ 8.090
- 2x Reguladores DC-DC de 5V	$ 2.036	$ 2,20	10	$ 204	$ 0,22	2	Link	$ 2,20	$ 4,40	$ 2.036	$ 4.072
- 1x Reductor de voltaje de 3.3V AMS 1117	$ 4.592	$ 4,96	10	$ 459	$ 0,50	1	Link	$ 2,48	$ 4,96	$ 2.296	$ 4.592
- 1x Raspberry Pi Pico W	$ 12.800	$ 13,82	1	$ 12.800	$ 13,82	1	Link	$ 69,12	$ 138,24	$ 64.000	$ 128.000
- 1x ArduCam Mini 2MP Plus. (Amazon)	$ 24.065	$ 25,99	1	$ 24.065	$ 25,99	1	Link	$ 129,95	$ 259,90	$ 120.325	$ 240.649
- 1x 18650 Li-Ion Battery	$ 48.212	$ 52,07	10	$ 4.821	$ 5,21	1	Link	$ 26,03	$ 52,07	$ 24.106	$ 48.212
- 1x 18650 Li-Ion Battery Holder	$ 1.876	$ 2,03	1	$ 1.876	$ 2,03	1	Link	$ 10,13	$ 20,26	$ 9.380	$ 18.760
- 1x INA-219 Power Sensor	$ 9.089	$ 9,82	10	$ 909	$ 0,98	1	Link	$ 4,91	$ 9,82	$ 4.545	$ 9.089
- 1x INA-3221 Power Sensor	$ 1.817	$ 1,96	1	$ 1.817	$ 1,96	1	Link	$ 9,81	$ 19,62	$ 9.085	$ 18.170
- 1x MPU-92/65 IMU	$ 24.400	$ 26,35	10	$ 2.440	$ 2,64	1	Link	$ 13,18	$ 26,35	$ 12.200	$ 24.400
- 1x DS-18B20 Temperature Sensor	$ 7.619	$ 8,23	10	$ 762	$ 0,82	1	Link	$ 4,11	$ 8,23	$ 3.810	$ 7.619
- 1x TEMT-6000 Phototransistor	$ 1.479	$ 1,60	1	$ 1.479	$ 1,60	1	Link	$ 7,99	$ 15,98	$ 7.397	$ 14.794
- 1x 4.7kΩ Power Resistor	$ 3.235	$ 3,49	50	$ 65	$ 0,07	1	Link	$ 0,35	$ 0,70	$ 324	$ 647
- 3x 1N5817 Schottky Diodes	$ 1.132	$ 1,22	50	$ 23	$ 0,02	3	Link	$ 0,37	$ 0,73	$ 340	$ 679
- 4x 8x2 Stackthrough connector	$ 3.967	$ 4,28	10	$ 397	$ 0,43	4	Link	$ 8,57	$ 17,14	$ 7.934	$ 15.868
- 4x 18x2 Stackthrough connector	$ 4.975	$ 5,37	10	$ 498	$ 0,54	4	Link	$ 10,75	$ 21,49	$ 9.950	$ 19.900
- 2x 26x2 (or longer) female board connector.	$ 6.000	$ 6,48	10	$ 600	$ 0,65	2	Link	$ 6,48	$ 12,96	$ 6.000	$ 12.000
- 1x Switch – push button with 6 connectors	$ 836	$ 0,90	10	$ 84	$ 0,09	1	Link	$ 0,45	$ 0,90	$ 418	$ 836
- 4x 40x1 female pinheader	$ 2.350	$ 2,54	20	$ 118	$ 0,13	4	Link	$ 2,54	$ 5,08	$ 2.350	$ 4.700
- 12x M3x5.4 Threaded inserts	$ 3.767	$ 4,07	100	$ 38	$ 0,04	12	Link	$ 2,44	$ 4,88	$ 2.260	$ 4.520
- 2x Pushrods (at least 180 mm long)	$ 6.842	$ 7,39	10	$ 684	$ 0,74	2	Link	$ 7,39	$ 14,78	$ 6.842	$ 13.684
- 8x M3x6 screws	$ 1.810	$ 1,95	50	$ 36	$ 0,04	8	Link	$ 1,56	$ 3,13	$ 1.448	$ 2.896
Support Equipment:											
- 1x Balanza digital gramera	$ 10.000	$ 10,80	1	$ 10.000	$ 10,80	1	Link	$ 54,00	$ 108,00	$ 50.000	$ 100.000
- 1x Multímetro digital	$ 8.389	$ 9,06	1	$ 8.389	$ 9,06	1	Link	$ 45,30	$ 90,60	$ 41.945	$ 83.890
- 1x Destornillador de cruz corto	$ 1.788	$ 1,93	1	$ 1.788	$ 1,93	1	Link	$ 9,66	$ 19,31	$ 8.940	$ 17.880
- 1x Termómetro digital	$ 945	$ 1,02	1	$ 945	$ 1,02	1	Link	$ 5,10	$ 10,21	$ 4.725	$ 9.450
- 2x Cajas organizadoras de componentes	$ 2.511	$ 2,71	1	$ 2.511	$ 2,71	2	Link	$ 27,12	$ 54,24	$ 25.110	$ 50.220
- 1x Caja de plástico de almacenamiento	$ 6.600	$ 7,13	1	$ 6.600	$ 7,13	1	Link	$ 35,64	$ 71,28	$ 33.000	$ 66.000
Insumos adicionales:											
- Estaño de soldadura	$ 7.421	$ 8,01	1	$ 7.421	$ 8,01	0	Link	$ 8,01	$ 8,01	$ 7.421	$ 7.421
	$ 218.643,32	$ 236,13				Unitario		USD / 5 kit	USD / 10 kit	CLP / 5 kit	CLP / 10 kit
	925,93	USD = x CLP			Total	$ 102,00		$ 510,01	$ 1.012,00	$ 472.230	$ 937.039
EPS + OBC PCB	$ 73.648	$ 79,54	10	$ 7.365	$ 7,95	2	Link	$ 79,54	$ 159,08	$ 73.648	$ 147.297
					TOTAL+PCB	$ 117,91		$ 589,55	$ 1.171,08	$ 545.878	$ 1.084.336

Raspberry Pi Zero W
EGS PCB

Anexo D: Guía de apoyo a la experiencia educativa

Guía de experiencia AIT/AIV

Manual del Estudiante



Diego Alejandro González G. (MT)

12 de Enero, 2026

V0.2 – EPS

Declaración: Trabajo en progreso

Esta guía representa un trabajo en progreso, hay información relevante para la experiencia que aún no ha sido agregada. El principal propósito de esta guía es evaluar la recepción y utilidad de esta al momento de realizar la actividad con el fin de elevar la autonomía de los estudiantes al realizar las actividades de la experiencia.

Al final de la experiencia se deberá hacer entrega de los formularios de entrada y salida presentados en esta guía.

Información general de la experiencia - Integración EPS

Tiempo máximo para la experiencia: 90 minutos

Contenido teórico:

- Concepto de Integración
- Ciclos AIT/AIV
- Aceptación.

Actividades prácticas:

- Aceptación de componentes de EPS.
- Integración del EPS.
- Ensamblado de la estructura y OBC mínimo.
- Verificación del EPS al estar integrado.

Requisitos:

- Electrónica industrial / electrotecnia
- Mediciones
- Familiaridad con conceptos de sistemas espaciales

En caso de dudas, consultar al profesor / tutor a cargo de la experiencia o al autor a través del correo digonzalez2019@udec.cl o alternativamente dagg450@gmail.com.

Control de versiones

V0.1 02.12.2025 – Avance 3 MT.

- Documento creado.
- Contenidos: Actividad práctica de integración EPS. Formulario de entrega integrado en la actividad dentro de la guía misma. Anexo no marcado con planos complementarios del CDS, PC/104 y PinOut del PC/104.

V0.2 12.01.2026 – Corrección de observaciones MT.

- Agregado control de versiones.
- Agregada información de experiencia y datos de contacto.
- Agregada Tabla de Contenido.
- Agregados los capítulos:
    - Introducción.
        - Descripción del kit y lista de materiales.
        - Introducción a Sistemas Espaciales.
        - Cronograma de la Experiencia.
        - Referencia a modelo MBSE.
- Agregado Formulario de Evaluación.
- Corregidas referencias al formulario en el cuerpo de la actividad.

Tabla de contenido

1. Introducción

Esta experiencia fue diseñada con el fin de entregar una oportunidad de aprendizaje experiencial respecto a algunos de los conceptos más fundamentales de la ingeniería de sistemas espaciales, integrados en estándares de diseño vigentes y probados en la industria y la academia, orientado especialmente a estudiantes de ingeniería de sistemas en la educación superior o directamente para profesionales.

Esta guía está pensada para actuar como apoyo a los materiales de la actividad, incluyendo el kit CubeSat, y la presentación del profesor.

NOTA: Esta guía es una referencia rápida para consultar información o procedimientos específicos, pero no reemplaza la presentación sincrónica del profesor. Seguir las instrucciones del profesor entregadas durante la presentación de la actividad.

El Kit CubeSat

Figura 1: Kit CubeSat y contenidos de la caja entregada por cada grupo.

El kit CubeSat que acompaña esta experiencia permite poner en práctica los conceptos estudiados de la ingeniería de sistemas. El kit completo incluye todos los contenidos de la caja de materiales entregada, además de esta guía. Los contenidos de la caja deberían ser:

REQUERIDOS:

- 1x Caja de materiales grande
- 2x Organizadores de componentes con:
    - 1x Microcontrolador de prototipo, Raspberry Pi Pico W.
    - 1x Batería Li-Ion 18650.
    - 2x Convertidores Booster DC-DC de 5V.
    - 1x Convertidor Buck de 3.3V.
    - 1x Regulador de carga HW-373.
    - 3x Diodos 1N5817 (o 1N5819).
    - 1x Resistencia de 4.7kΩ.
    - 8x Tornillos M3x6.
    - 1x Sensor INA3221.
    - 1x Sensor de luminosidad TEMT-6000.
- 2x Placas de circuito impresos
    - 1x EPS
    - 1x OBC
- 2x Marcos mayores de PLA.
- Un conjunto de marcos menores de PLA:
    - Marcos con separación larga y corta, con y sin inserto roscado (1 cada uno, 4 en total)
- 4x Barras de soporte M3x86, roscadas en un extremo.
- 1x Cable micro-USB a USB.
- 1x Multímetro digital con dos sondas.
- 1x Destornillador de cruz.
- 1x Balanza digital
- 1x Guía de Experiencia (Este documento).

NO REQUERIDOS:

- 1x Kit de EGS:
    - 1x PCB de EGS.
    - 1x Raspberry Pi Zero W.
    - 1x Diodo switch.
- 1x Cinta métrica.

Que no están cubiertos en esta guía.

NOTA: Al recibir el kit, revisar que este incluya todos los componentes requeridos presentados en esta sección. Esto constituye la primera parte de la actividad y es parte del proceso de aceptación del Kit.

Sistemas Espaciales

Según la *European Cooperation for Space Standarization* (o por su sigla ECSS), el órgano europeo para la estandarización espacial, un sistema espacial se puede descomponer en primera instancia en tres dominios mayores o segmentos, con un cuarto *bonus* correspondiente a todo lo que es soporte (*Support Segment*) (ECSS Secretariat, 2012). Estos serían: *Space Segment* (espacial), *Ground Segment* (terrestre) y *Launch Segment* (lanzador). La descomposición completa se observa en la figura adjunta:

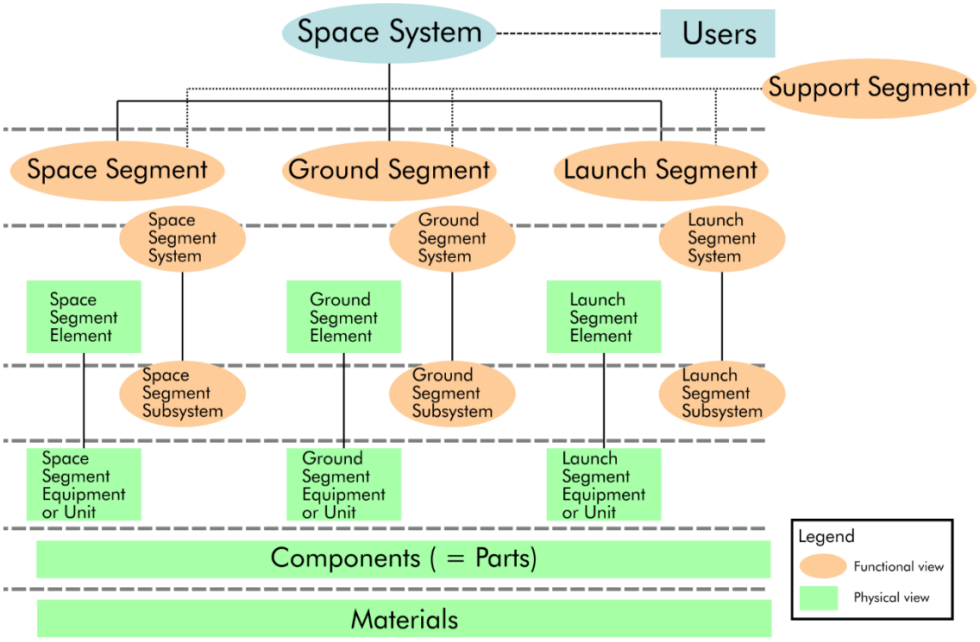


Figura 2: Descomposición de sistema espacial

De esta forma, al hablar de un componente nos referimos a un objeto físico que forma parte de un ensamble (equipamiento) o un subsistema. Adicionalmente, para la descomposición de subsistemas dentro de un CubeSat se utiliza el CubeSat System Reference Model, o CSRM. Este define los principales subsistemas que son típicos en el desarrollo de este tipo de sistemas espaciales.

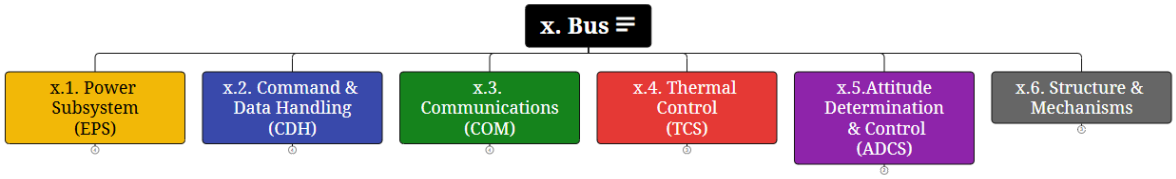


Figura 3: Descomposición de subsistemas de acuerdo con el CSRM.

Fases de la experiencia

La experiencia se divide en 4 etapas dentro de los 90 minutos disponibles. Estos pueden variar en tiempo y contenido dependiendo de las necesidades específicas de la experiencia o clase realizada, pero a continuación se presenta una sugerencia:

1. Introducción: Aproximadamente 5 minutos. Se presenta el objetivo de la experiencia y las herramientas y métodos principales.
2. Presentación teórica: Aproximadamente 5 a 10 minutos. Se presentan los conceptos fundamentales a ser practicados en esta experiencia.
3. Actividad de Integración:
    a. Preparación y aceptación de componentes: Entre 5 a 10 minutos. Se revisan y aceptan los kits de CubeSat para la realización de la actividad.
    b. Integración del EPS: Aproximadamente 20 a 30 minutos. Se realiza la integración física y funcional del EPS.
    c. Verificación del EPS: Aproximadamente 5 minutos. Se finaliza el EPS y se verifican interfaces y envelopes.
    d. Opcional: Integración con OBC y estructura: Aproximadamente 15 minutos. Se ensambla una implementación mínima del OBC con el sensor de luminosidad, el microcontrolador Raspberry Pi Pico W, y un diodo de protección adicionar requerido por la Raspberry. PRECAUCIÓN: El sensor de luminosidad TEMT-6000 es un transistor, la instalación errónea de este causará un corto circuito en la barra de 3.3V, lo que dañará varios componentes de la barra además de la TEMT-6000. Proceder con precaución.
4. Cierre y evaluación: Aproximadamente 15 minutos. Se realiza un cierre de la experiencia, donde se presentará una conclusión y se hará entrega de los formularios de evaluación de la experiencia.

Trabajo MBSE

Adicionalmente, se puede realizar un trabajo de verificación en conjunto con el modelo de sistema cargado en Violet Labs. Este se puede encontrar en el espacio de trabajo de la Universidad de Concepción, donde existen algunos requerimientos ya cargados, además de un conjunto de parámetros y un script para vincular la suma de hijos. Esto no corresponde a un trabajo que esté implementado en esta versión del documento, pero es perfectamente realizable.

2. Actividad práctica: Integración de EPS

Para la integración del EPS seguimos la lógica de “seguir el flujo de corriente”. Comenzando desde la batería se realiza la integración de elementos siguiendo la línea de corriente (positiva). La idea es poder verificar que los componentes instalados efectivamente se comporten como se espera, esta corresponde a la prueba de aceptación del componente.

Antes de comenzar, se sugiere medir la masa de la placa de circuitos del EPS sin ningún componente removible instalado.

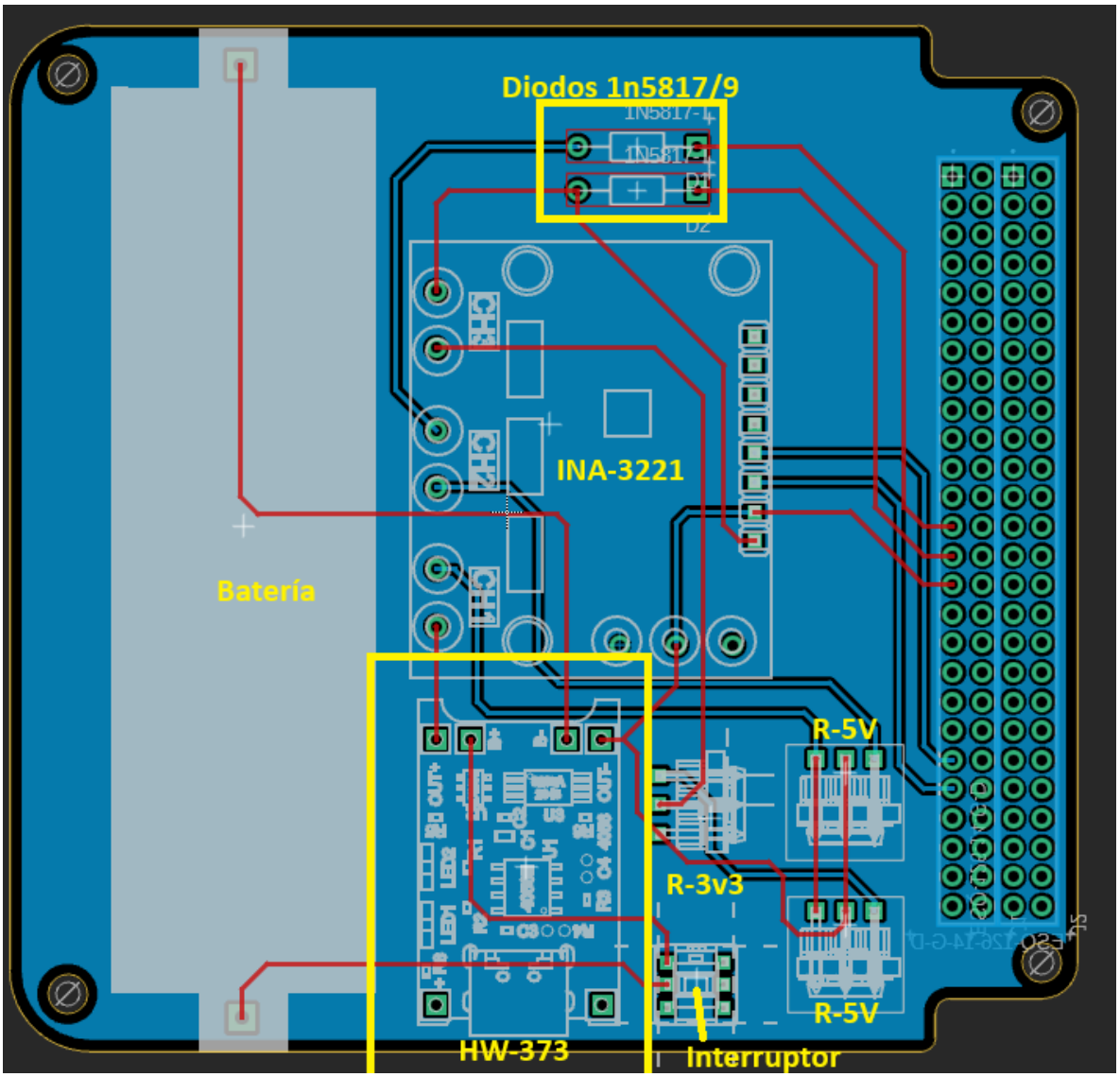


Figura 4: Diagrama general de la placa de circuitos del EPS.

2.1. Batería

El kit de satélite utiliza una única batería de Li-Ion 18650, cuyo voltaje nominal es de 3.7V. En la realidad, las baterías entregan un voltaje variable entre 4.2V y 2.5V, cuando están limitados por un regulador de carga/descarga como es este caso.

Figura 5: Batería Li-Ion 18650 de referencia. Notar la polaridad de los extremos.

Para verificar que la batería esté funcionando correctamente se debe:

1. Utilizar un multímetro digital (Ref. Figura 6: Funciones de un), conectar las terminales al **Común (COM, negro)**, y la terminal de uso **general (V, Ω, mA, Rojo)**.
2. Encender el multímetro y colocar el selector en la posición de lectura de voltaje: **DCV (Voltaje en Corriente Continua).**
   NOTA: El multímetro utiliza diferentes escalas para presentar las mediciones. En el caso de la batería, el orden de magnitud es mayor a 2V (2000 mV), por lo que se debe colocar el selector en la posición de 20V.
3. Hacer contacto con las sondas del multímetro a los extremos de la batería. El terminal negro va al polo negativo, y el rojo al positivo.
4. Registrar el voltaje medido.
5. Registrar la masa de la batería.

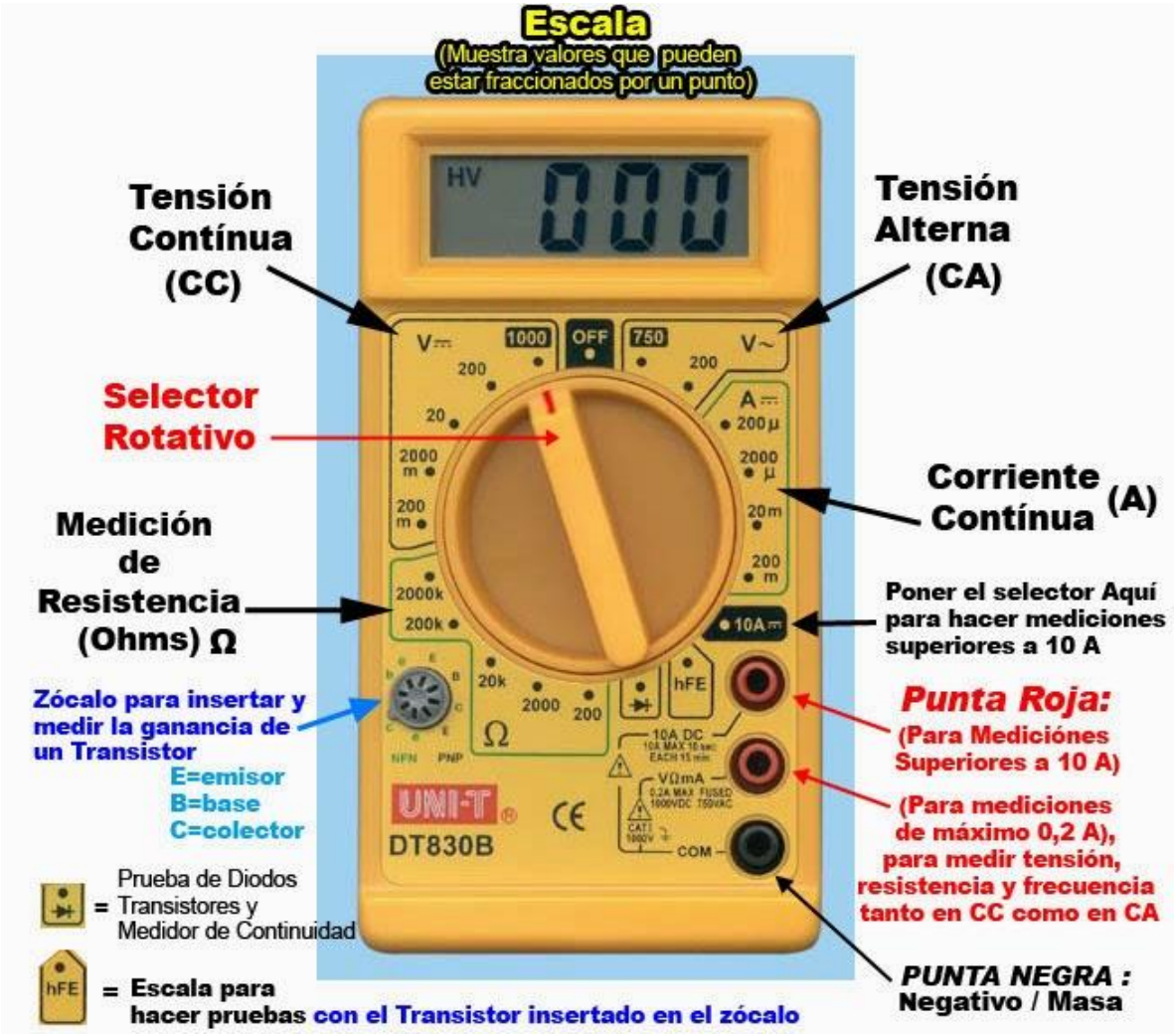


Figura 6: Funciones de un multímetro digital.

Para la integración al EPS, se debe:

1. Asegurar que no hayan otros componentes instalados en la placa, con la excepción del interruptor.
2. El interruptor debe estar en la posición de apagado (abajo). Se puede verificar pulsandolo. Figura 7.
3. Verificar la correcta polaridad de la batería respecto al portabaterías. En caso de tener dudas, el polo positivo va hacia el borde donde se encuentra el interruptor.

   PRECAUCIÓN: Debido a que los terminales de la batería se conectan a la tarjeta de circuito directamente, y que estos terminales sobresalen por la parte inferior, es muy importante no trabajar sobre ninguna superficie que pueda ser conductor (como una mesa de metal). Ante la duda, mejor consultar antes de instalar la batería.

4. Instalar la batería, insertando primero el extremo negativo de esta, y luego el positivo.
5. Una vez instalada, verificar que el porta baterías esté recibiendo la corriente correctamente. Para esto, medir el voltaje en las terminales que conectan el porta baterías a la placa de circuito.

Figura 7: Referencia rápida para el estado del interruptor.

2.2. Regulador de carga

La batería se conecta al resto del circuito a través de un regulador de carga, que la protege de varias complicaciones que podrían emerger, como una sobre carga, retroalimentación o sobredescarga. Este corresponde a un módulo HW-373. Registrar masa.

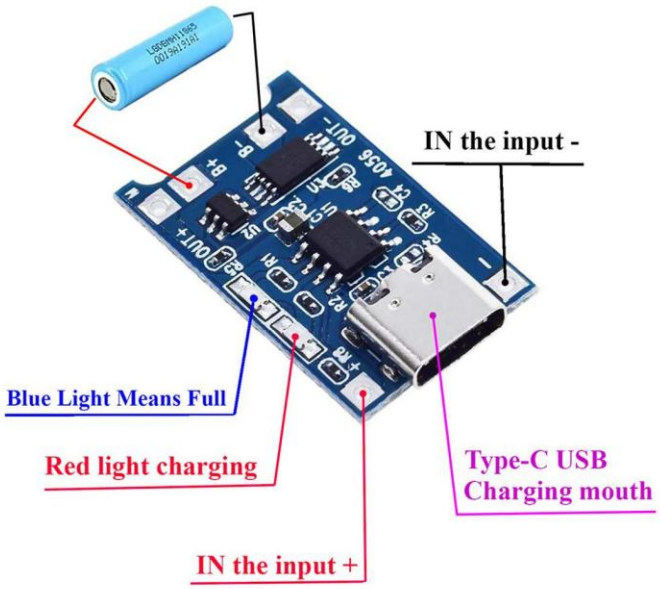


Figura 8: Módulo HW-373

Para realizar la integración de este componente, seguir estos pasos:

1. Asegurarse que la batería esté instalada en la polaridad correcta y que el interruptor del EPS esté en la posición de apagado.
2. Instalar con cuidado el regulador HW-373 en la posición designada en la placa. (Figura 1)
3. Encender el EPS y verificar el voltaje de salida entre las terminales OUT+ y OUT-. Este no debería variar significativamente respecto al voltaje de la batería (la máxima diferencia admisible es 0.1V).
   Registrar voltaje.

2.3. Sensor INA-3221

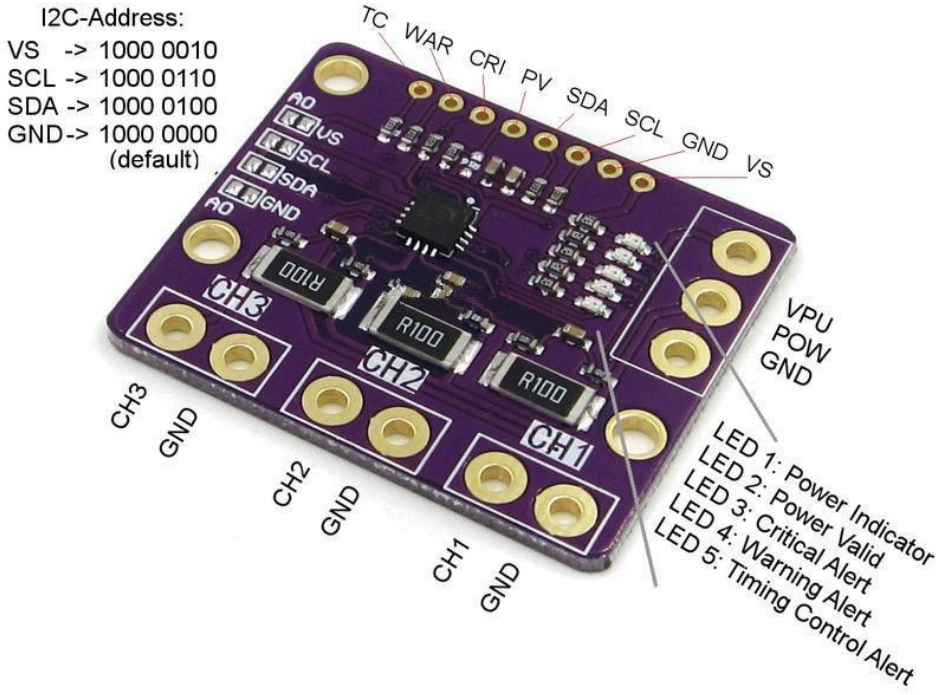


Figura 9: Sensor INA-3221

Buscar el sensor INA-3221. Este corresponde a un sensor de corriente y tensión de 3 canales. Esto significa que puede tomar mediciones de la corriente y tensión en 3 circuitos diferentes, lo que nos permite medir el estado de los circuitos de la batería (3.7V), y ambas barras de alimentación (3.3V y 5V). Esta va montada al centro del PCB, junto a la batería y solo tiene una posición en la que encajará correctamente, con los canales en dirección a la batería.

Registrar masa de la INA-3221.

La tensión entre las terminales de cada canal de medición de la INA-3221 (CH1, CH2, CH3, respecto a sus terminales negativas), debe ser 0V. Además, la tensión entre cada canal y la tierra [GND] debe ser:

- CH1: 3.7V
- CH2: 5V
- CH3: 3.3V

Una forma de verificar que la INA-3221 está funcionando es por medio de los LED integrados (rojo y verde). Si están encendidos, la INA-3221 está conectada. Igualmente, verificar las terminales.

2.4. Reguladores de las barras

Las barras se refieren a la alimentación de las dos fuentes de poder del EPS: la línea de 5V y de 3.3V. Estas fueron diseñadas para operar en paralelo, siguiendo la lógica siguiente:

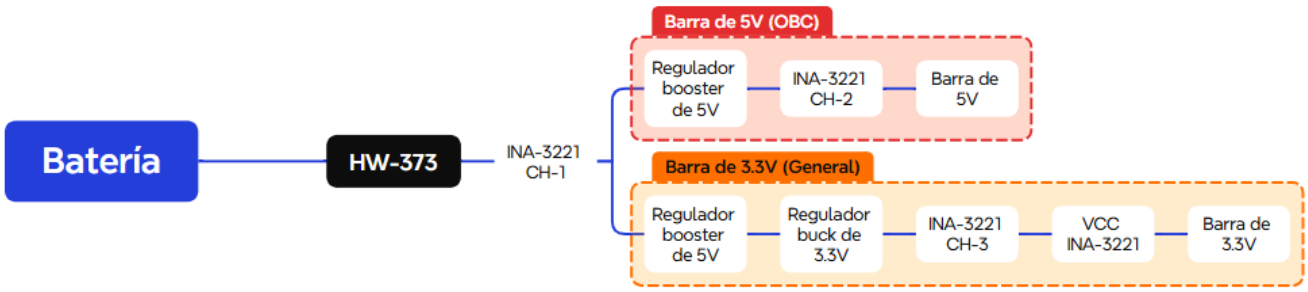


Figura 10: Diagrama de alimentación de las barras de 3.3V y 5V.

De esta forma, en ambas barras se usan los mismos reguladores de 5V. Estos son los siguientes:

Figura 11: Regulador Booster DC-DC de 5V.

Estos reguladores tienen la extraña peculiaridad de que el voltaje de entrada es por V1, y la salida es VO, con el pin central siendo tierra. Una forma de recordar en qué dirección deben instalarse es que siempre deben apuntar hacia “Adentro” del PCB, con VO en dirección del bus PC/104.

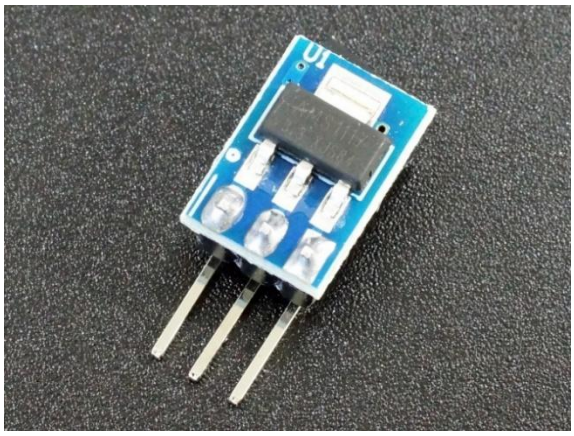


Figura 12: Regulador AMS-1117 de 3.3V

Finalmente se tienen los reguladores de 3.3V, que reducen el voltaje de entrada de 5V a 3.3V. Estos son del modelo AMS-1117.

Registrar la masa de todos los reguladores.

2.5. Diodos de protección y aceptación del EPS

Lo último que falta es instalar los dos diodos Schottky para proteger el circuito regulador. Estos se instalan al costado del sensor INA-3221.

Los diodos para instalar son los **1N5817**. Estos generan una caida de voltaje entre los extremos, por lo que es necesario medir esa diferencia para asegurar que la barra mantenga una tensión aceptable para cada barra. Registrar masa de cada uno.

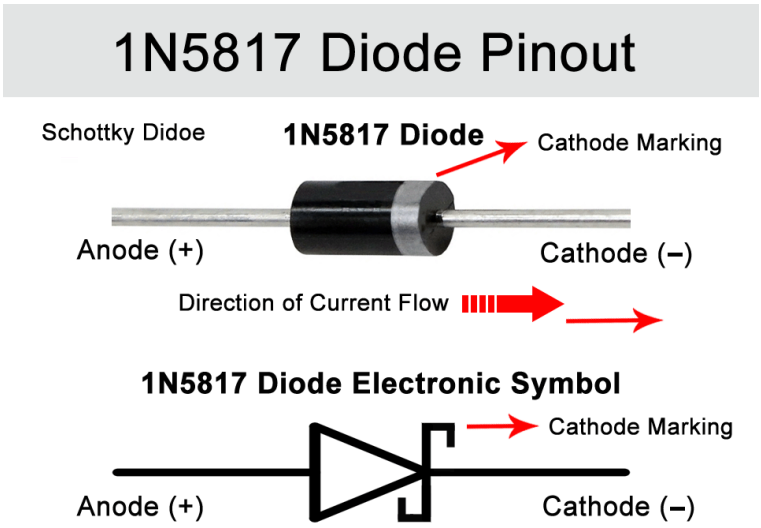


Figura 13: Descripción y uso de un diodo

Finalmente, realizar la prueba de aceptación del EPS. Esta consiste en tomar la masa total del subsistema completamente integrado, verificar las barras de potencia (3.3V y 5V), y verificar que exista continuidad en la línea de tierra (GND).

Apunte suplementario: Electrónica

Para facilitar la referencia rápida de conceptos de electrónica relevantes para la experiencia, se presenta el siguiente apunte. Estos utilizan el modelo hidrodinámico como una analogía que compara los conceptos eléctricos al flujo de fluidos en sistemas de tuberías. Es importante mencionar que el modelo hidrodinámico representa una aproximación de los fenómenos y no es una descripción acertada de los fenómenos descritos.

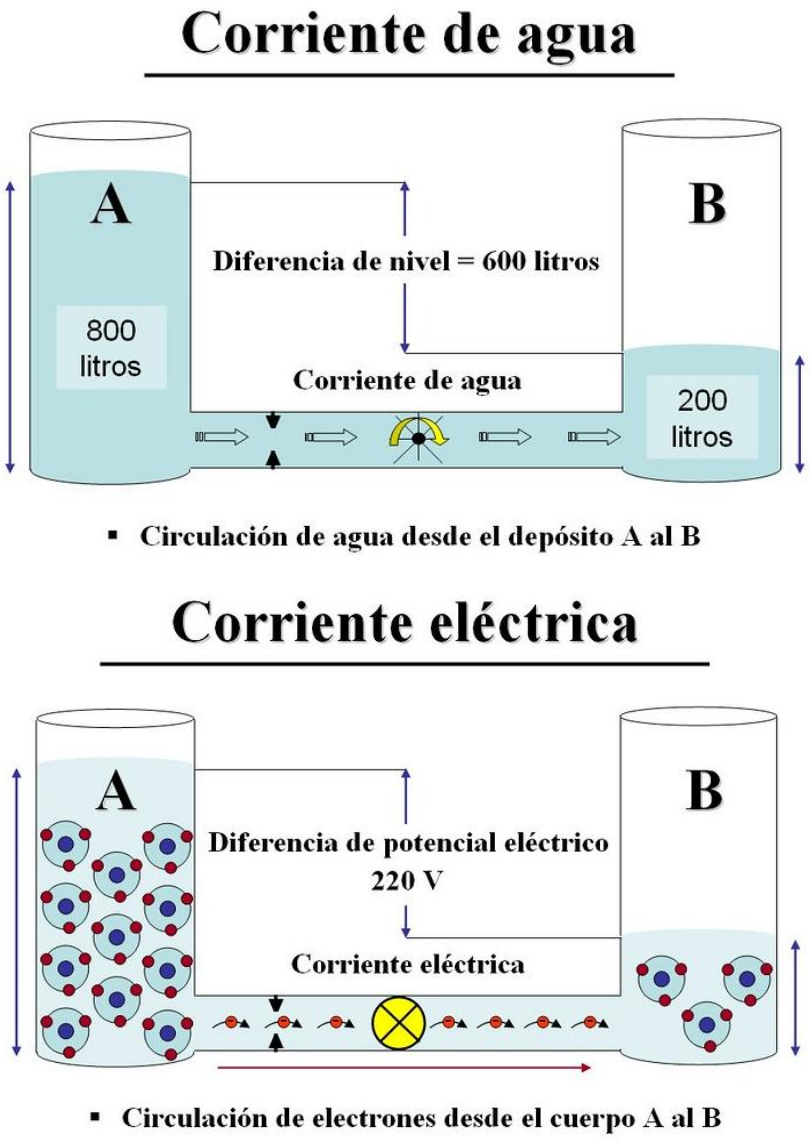


Figura 14: Analogía Hidrodinámica de la Electricidad

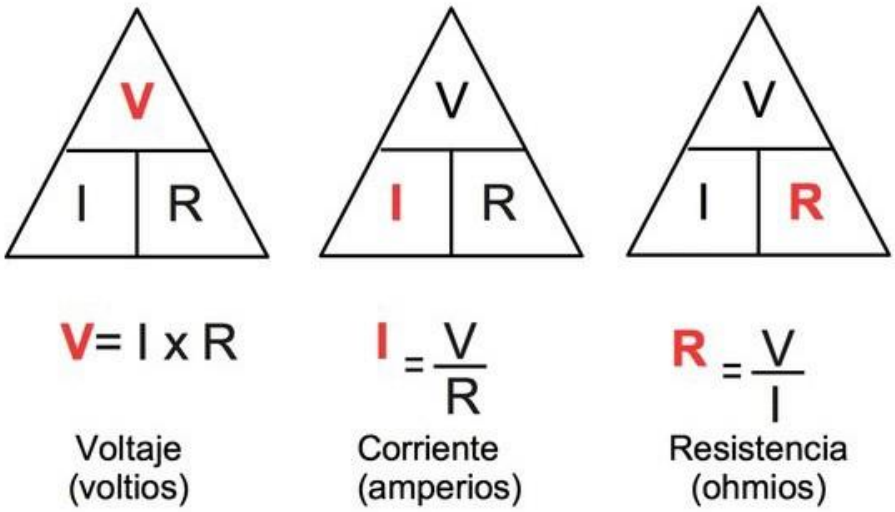
Tensión, corriente y resistencia

La tensión eléctrica (V), también llamada **voltaje** o diferencia de potencial, representan la energía disponible que impulsa electrones a través de un material conductor. Se mide en Voltios o Volt [V] con un voltímetro o multímetro con esta función. En el modelo hidrodinámico corresponde a la diferencia de altura entre las columnas de agua A y B.

La corriente eléctrica (I) corresponde al flujo de partículas cargadas (como electrones) a través de un material conductor. Este requiere una diferencia de potencial para ser efectiva. Su unidad de medida es el **Amperio [A]** y se mide con un amperímetro o multímetro con esta función. En el modelo hidrodinámico, la corriente eléctrica se relaciona directamente con el flujo de agua entre las columnas A y B.

La resistencia eléctrica (R) es la oposición a la corriente eléctrica que un material impone. Generalmente los materiales con baja resistencia se llaman conductores, mientras que los materiales con baja resistencia son aislantes. Su unidad de medida son los **Ohmios u Ohm [Ω]**, y se mide con un ohmímetro o un multímetro con esta función. Un ejemplo en el modelo hidrodinámico es el uso de una válvula de paso que reduzca el flujo de agua por una tubería. Asimismo, la tubería también ejerce una resistencia al flujo por medio de fuerzas viscosas.

Ley de Ohm

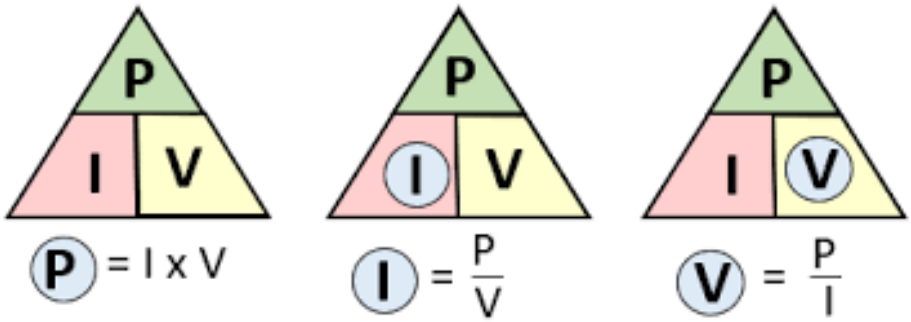


La Ley de Ohm describe la relación entre la tensión, corriente y resistencia eléctrica de un circuito, y a grandes rasgos indica que si conocemos dos de estas, podemos calcular siempre la tercera.

Potencia eléctrica

La potencia eléctrica (P) representa la velocidad a la que la energía eléctrica se consume o genera en un circuito. Su unidad es el Watt [W] o Vatio, y al igual que en la mecánica, la potencia puede expresarse como energía por unidad de tiempo [J/s].

En circuitos eléctricos, la potencia puede calcularse a través de la Ley de Watt, la cual, análoga a la Ley de Ohm, relaciona la Corriente, Tensión y Potencia Eléctrica.



Circuitos

Un circuito electrónico es un sistema de componentes interconectados (resistencias, transistores, condensadores, etc.) conectados por conductores que permiten el flujo controlado de corriente eléctrica para realizar funciones específicas, a diferencia de un circuito eléctrico simple que solo usa elementos básicos como cables y ampolletas o interruptores, un circuito electrónico añade componentes activos o semiconductores que permiten procesar información o amplificar señales para aplicaciones más avanzadas.

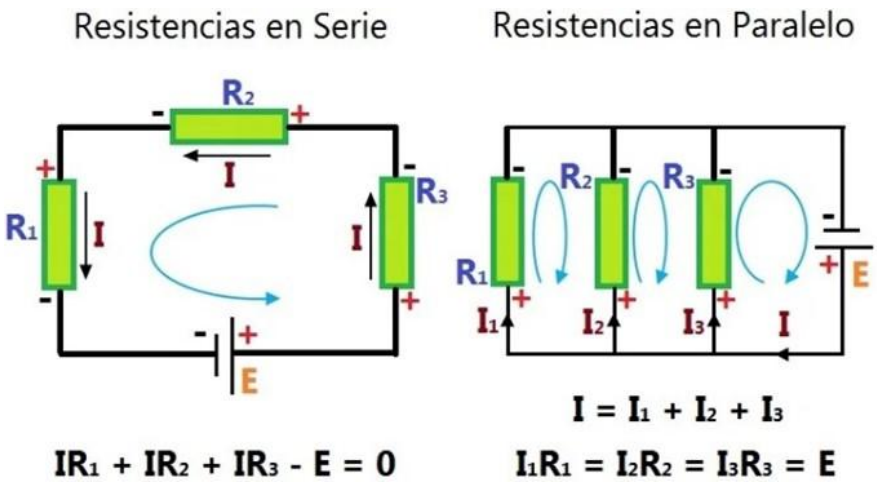


Figura 15: Leyes de Kirchhoff

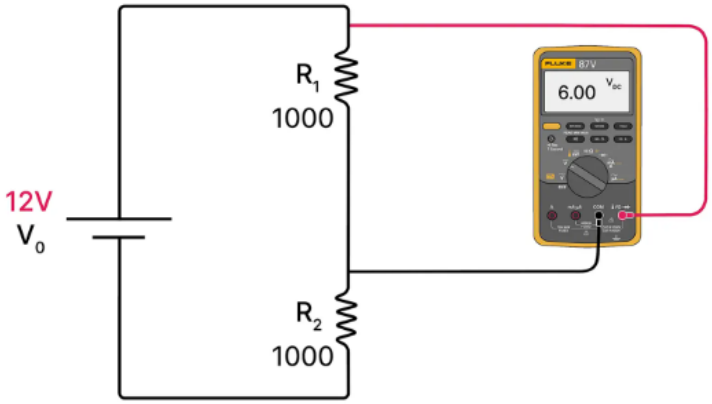
Las Leyes de Kirchhoff son dos principios fundamentales para analizar circuitos eléctricos: **La ley de corrientes** dice que la suma de corrientes que entran a un nodo es igual a la suma de las que salen, y **la ley de voltajes** establece que la suma de voltajes en cualquier malla o bucle cerrado es cero, aplicando la conservación de la energía.

Diodos

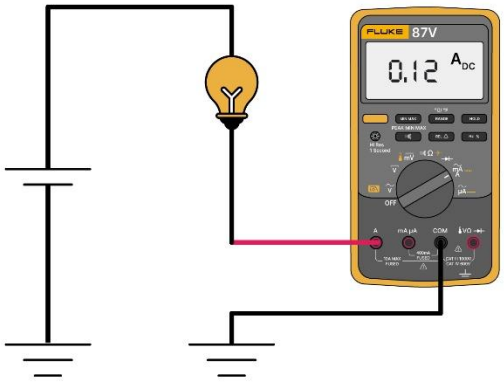
Los diodos, en particular los diodos Schottky, son un tipo de semiconductor que solo permite el flujo de corriente eléctrica en una sola dirección. Son análogos a una válvula unidireccional en un modelo hidrodinámico. Los diodos Schottky tienen un voltaje de diseño. En el caso de los 1N5817, este es de 20V, mientras que un 1N5819 es de 40V. Ambos operan bastante más abajo de sus voltajes máximos en esta experiencia.

Mediciones básicas con multímetro

Para utlizar un multímetro digital en mediciones de tensión, resistencia o corriente se debe tener ciertas precauciones. En el caso de la lectura de Resistencia o Tensión, esta lectura siempre se hace formando un circuito paralelo con el multímetro:

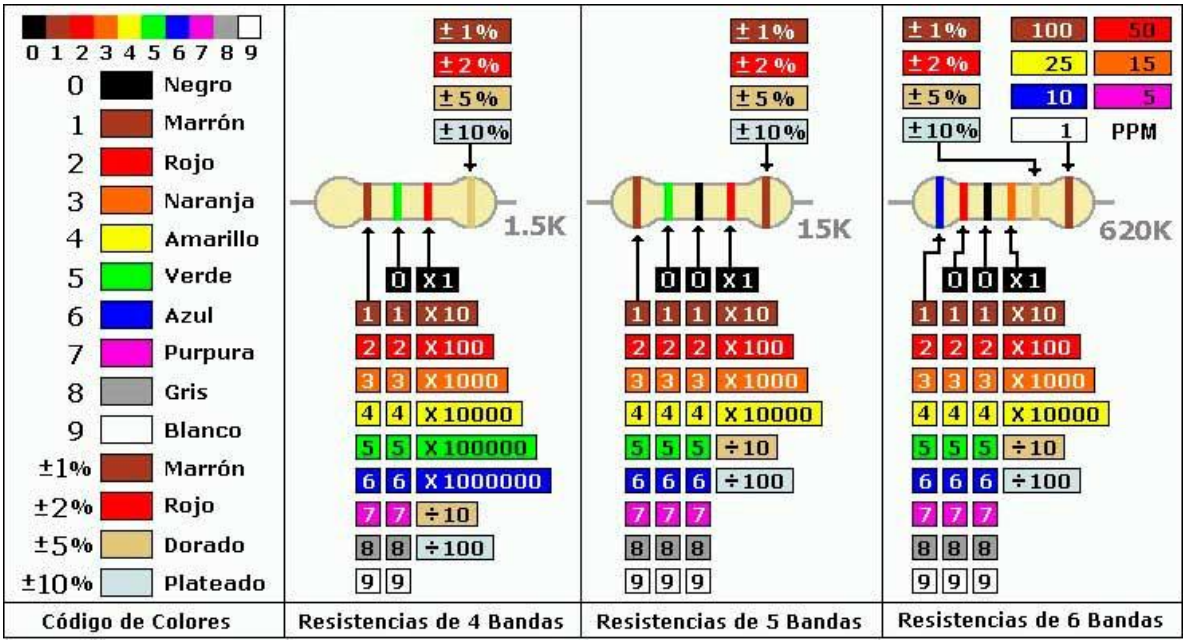


En el caso de querer medir Corriente, este debe ser en serie dentro del circuito, como muestra la figura:



Resistencias

Las resistencias tienen un código de color que indican su valor. Este se puede ver en la figura adjunta:



Formulario de Evaluación

Entregar este formulario completado luego de terminar la experiencia.

Fecha: Clase/Curso:

Profesor:

Grupo:

Integrantes:

- _____
- _____
- _____

Tabla de aceptación y verificación para el EPS

Rellenar esta tabla como desarrollo de la actividad de integración del EPS. Se sugiere realizar esto en paralelo a la actividad.

Ítem	Masa [g]	Ref. [g]	Tensión [V]	Ref. [V]	Aceptado?
PCB EPS		54.5	N/A	N/A	
Batería		46.5		3.7	
HW-373		19.5		3.7	
INA-3221		5.0	N/A	N/A	
CH-1	N/A	N/A		3.7	
CH-2	N/A	N/A		5.0	
CH-3	N/A	N/A		3.3	
Booster 5V [5v]		1.25		5.0	
Booster 5V [3v3]		1.25		5.0	
AMS-1117		0.75		3.3	
1N5817-1		0.075		3.3	
1N5817-2		0.075		5.0	
Barra 5V	**N/A**	**N/A**		**5.0**	
Barra 3v3	**N/A**	**N/A**		**3.3**	
EPS TOTAL		**128.9**	**N/A**	**N/A**	

Preguntas de desarrollo para la Actividad de Integración de EPS

1. Posterior a realizar el proceso de aceptación de los componentes del EPS:
   i. ¿Qué observaciones se han realizado?
   ii. ¿Se ha cumplido el presupuesto de masas del EPS?
   iii. ¿Se aprueba el inicio del proceso de integración? Justifique.

2. Durante el proceso de integración:
   i. ¿Qué hallazgos o complicaciones han enfrentado?

3. Al completar el EPS:
   i. ¿Se verifican las funciones del EPS?
   ii. ¿Cuál de las 4 funciones típicas de un EPS no cumple este ejemplo?

4. Posterior a la experiencia:
   i. Tiempo tomado para realizar la actividad (hh:mm): ____ : ____
   Indicar qué problemas encontraron durante el desarrollo de la actividad.

Anexo: Planos y material de apoyo

H1				H2			
1	UART0 TX (NV	2		1		2	
3	UART0 RX (NV	4		3		4	
5		6		5		6	
7		8		7		8	
9		10		9		10	
11		12		11		12	
13		14		13		14	
15		16		15		16	
17		18		17		18	
19		20		19		20	
21		22		21		22	
23		24		23		24	
25	5V-O	26		25		26	
27	3.3V-O	28		27		28	
29	GND-O	30		29		30	
31		32		31		32	
33		34		33		34	
35		36		35		36	
37		38		37		38	
39		40		39		40	
41	I2C_SDA-I/O	42		41		42	
43	I2C_SCL-I/O	44		43		44	
45		46		45		46	
47	LUPO	48	LUPO	47		48	
49	LUPO	50	LUPO	49		50	
51	LUPO	52	LUPO	51		52	

Figura 16: Guía de conexiones del bus PC/104

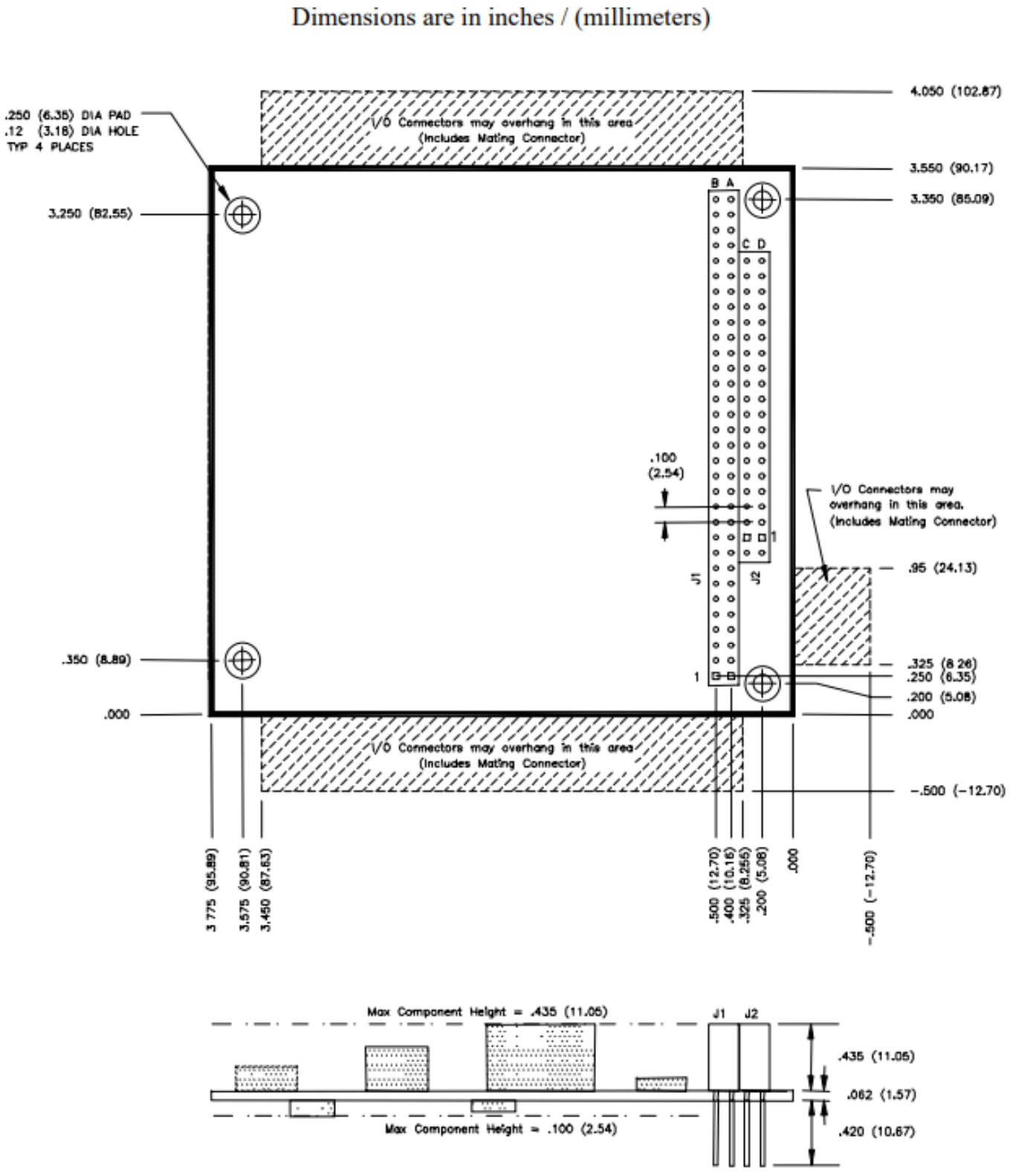


Figura 17: Dimensiones de PCB de acuerdo con estándar PC/104

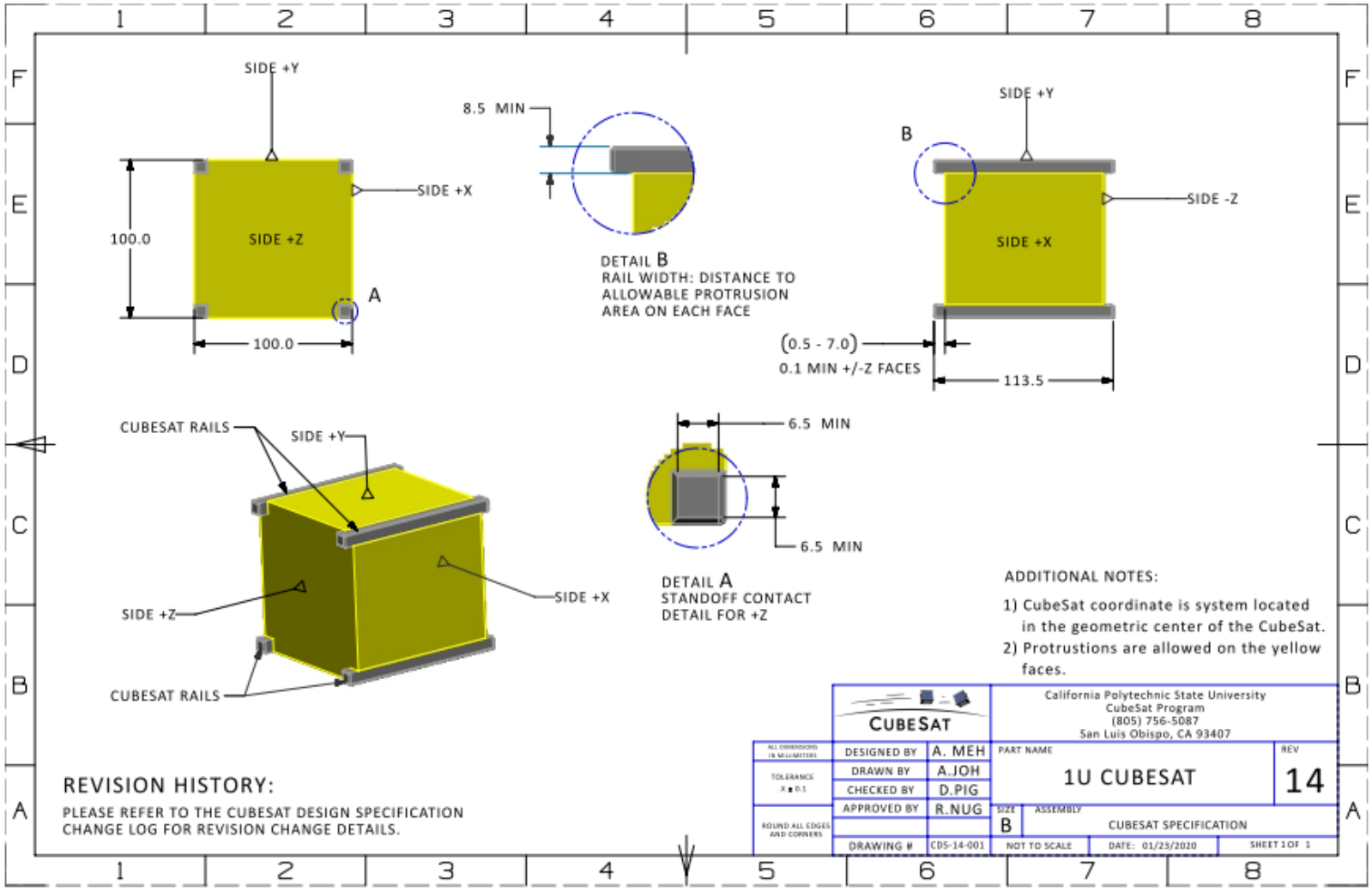
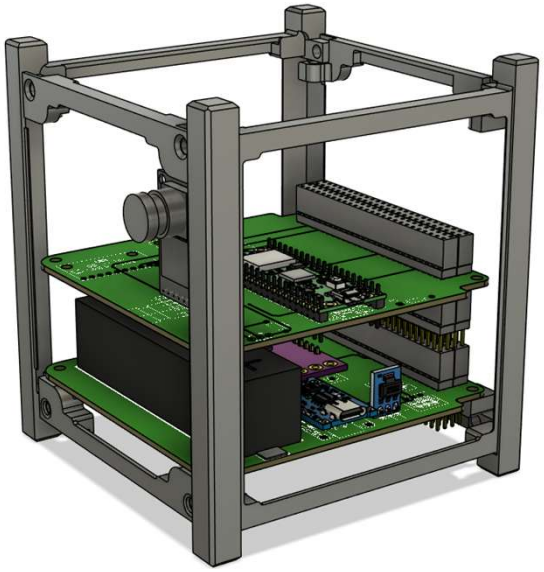


Figura 18: Plano con dimensiones para un CubeSat de 1U.

Anexo E: Presentación realizada como parte de la experiencia piloto

Experiencia piloto AIT/AIV HAISE-Sat II

MT - Diego González
02-12-2025



1

Contenidos

- Contexto relevante: Ciclo de vida, Procesos AIT/AIV, CubeSat
- Integración de EPS
- Integración de sensor y comunicaciones
- Cierre

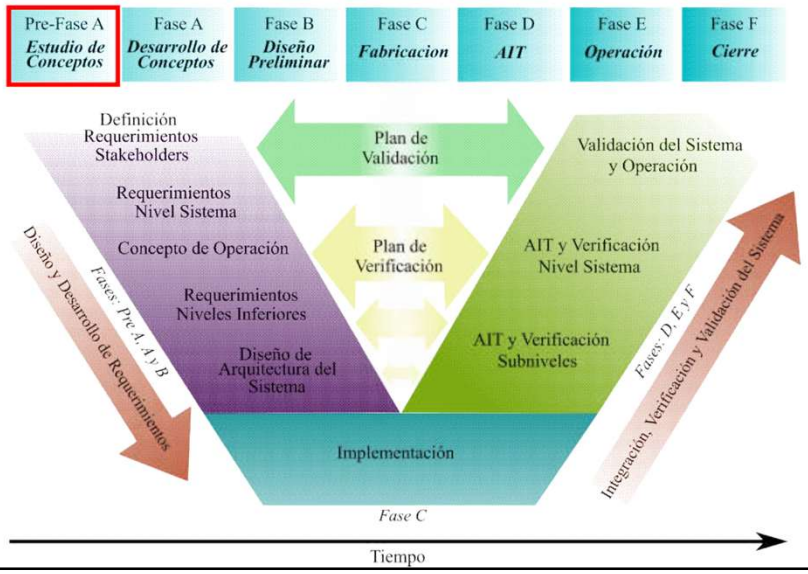
2

Aprendizajes esperados

- Concepto de Integración
- Procesos de AIT
- Conceptos fundamentales de una especificación (CubeSat)
- Procesos de testeo en componentes electrónicos
- Integración de componentes, subsistemas y elementos

3

Contexto: Ciclo de vida de un sistema



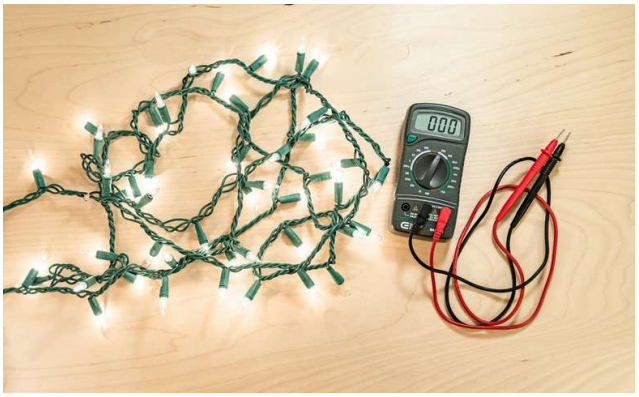
4

Contexto: AIT

- Siglas para Assembly, Integration and Testing (Ensamblaje, integración y testeo)

5

¿Por qué se hace este proceso?



En términos simples, por dos motivos:

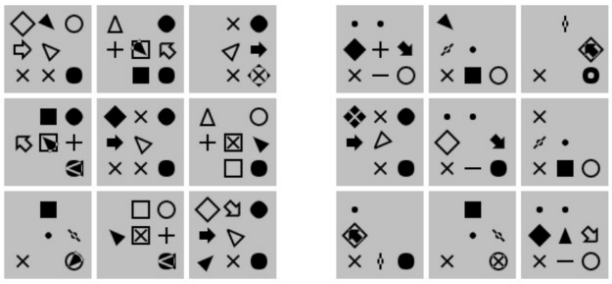
1. Por propagación de errores
2. Por conveniencia

6

Encontrar problemas dentro de un sistema integrado es más difícil que en componentes discretos

La heterogeneidad del sistema también dificulta el proceso

Sistema homogéneo



Sistema heterogéneo

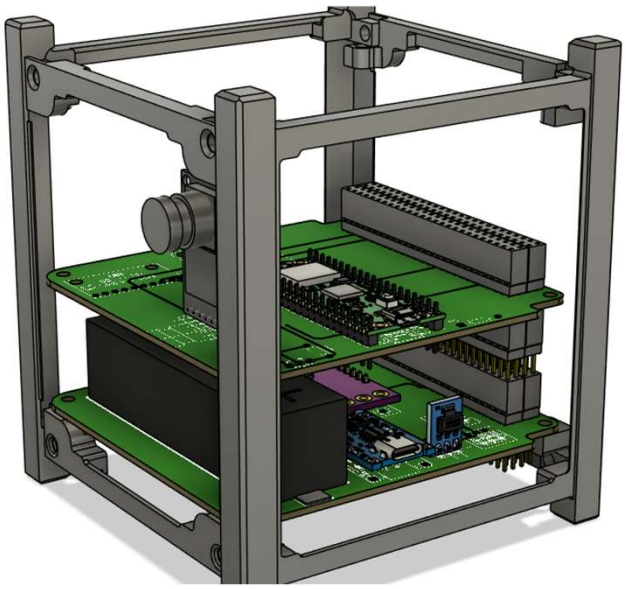
7

Contexto:

CubeSat

Satélites cuya principal caracteristica es el factor de forma: Son cubos con dimensiones y volumenes definidos.

Esto permite una integración estandarizada a otros sistemas, como vehiculos de lanzamiento o deployers (POD).



8

Sistema de CubeSat

- De una misión que podría ser “Observación terrestre”, se descomponen las funciones del sistema y se presenta una arquitectura.
- Existen dos segmentos de interés en este caso:
  - Segmento de vuelo: Satélite
  - Segmento terrestre: Estación de comunicaciones

El enfoque de esta experiencia es en el segmento de vuelo.

9

Hardware de la experiencia

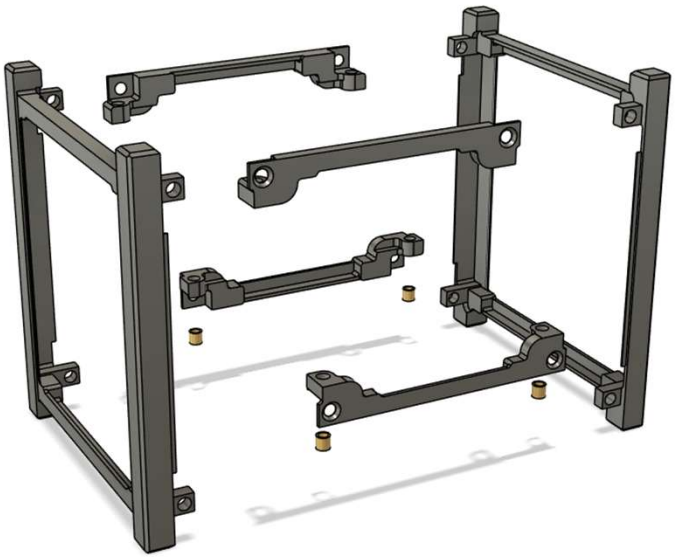
- El enfoque es en el elemento del CubeSat
- Este consiste en 3 ensambles de subsistemas:
  - Estructura
  - Potencia eléctrica [EPS]
  - OBC (Controlador [C&DH], comunicaciones [COM], sensores [ADCS] y payload óptico [PLEO])

10

Estructura

Consiste en:

- 2x Marcos laterales
- 2x Conectores L
- 2x Conectores M
- 4x Varillas guía
- 8x Tornillos M3x6mm

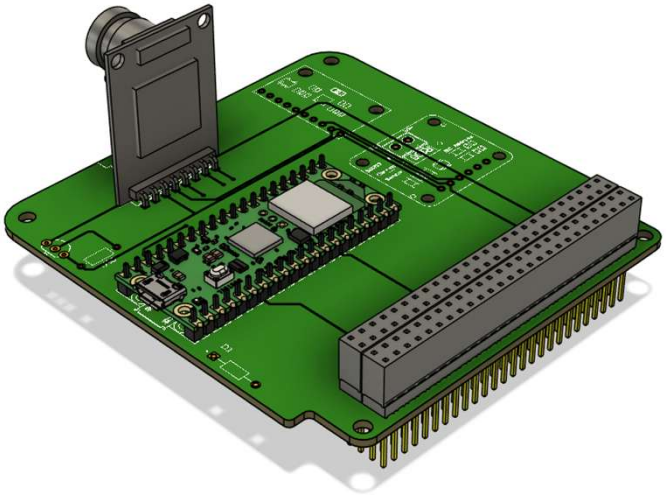


11

OBC, ADCS y PLEO

La segunda placa del sistema, posee conectores largos (macho hacia abajo y hembra hacia arriba) y es reconocible por la Raspberry Pi Pico W y la cámara Arducam.

Instrucciones de AIT junto a descripciones más detalladas de este subsistema se entregan en la guía complementaria de la experiencia.

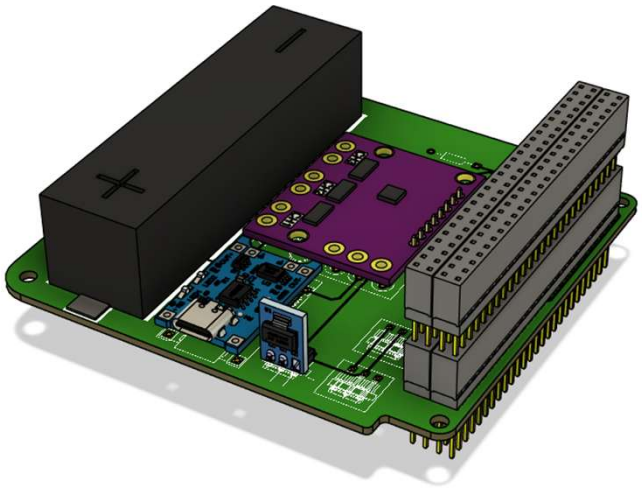


12

EPS

Fácil de reconocer por el portabaterías, la presencia del interruptor principal, y por tener conectores solo hembra hacia arriba.

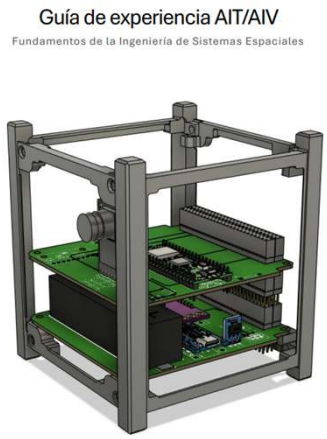
Este será el enfoque principal de la experiencia.



13

Actividad: Integración de EPS

- Seguir guía de experiencia



Guía de experiencia AIT/AIV

Fundamentos de la Ingeniería de Sistemas Espaciales

Diego González

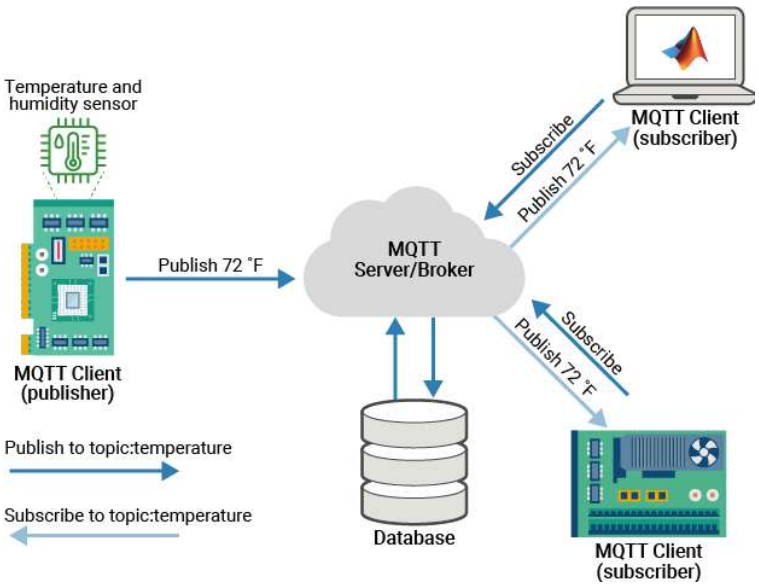
2 de Diciembre, 2025

1

14

Actividad: Integración de sensores y comunicaciones

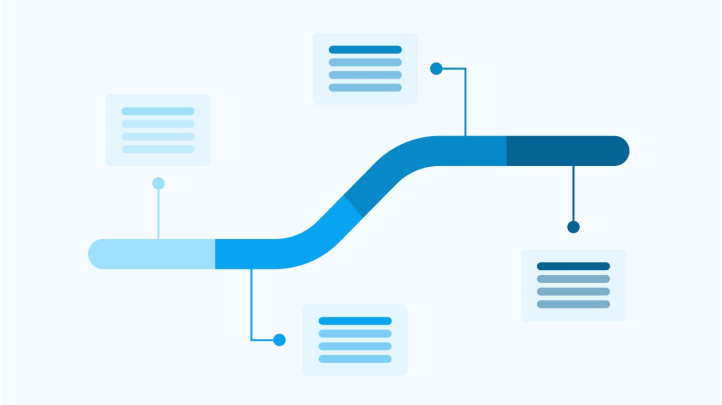
- Instrucciones del ayudante



15

Cierre

Iteración y paciencia; esta es una experiencia piloto, la idea es seguir refinando y mejorando



16

¿Preguntas?
¿Comentarios?

17

Anexo F: Formulario de evaluación realizado al final de la experiencia piloto

Formulario de evaluación Experiencia piloto AIT/AIV

Este formulario se presenta como cierre de la experiencia piloto AIT/AIV. La intención de este formulario es evaluar la efectividad de los métodos utilizados como herramientas de apoyo a la educación, pero también la percepción y claridad de estos por parte de los estudiantes.
Las preguntas de este formulario buscan encontrar puntos de mejora y retroalimentación respecto a varias técnicas utilizadas durante la experiencia. Por favor, responder a conciencia. Este formulario no es una evaluación sobre los estudiantes, ni tampoco representa una nota en el curso asociado.
Las respuestas entregadas serán utilizadas para la formulación y validación de métodos de enseñanza relacionados con la memoria de título. Las respuestas serán anonimizadas al momento de ser presentadas.

* Required

* This form will record your name, please fill your name.

Parte 1: Evaluación de conocimientos

Responda a las siguientes preguntas basandose en los conocimientos adquiridos durante la experiencia práctica.
Estas preguntas no se correlacionan con una evaluación, por lo tanto responder libremente considerando la respuesta que parezca más correcta. En las preguntas de texto, por favor responder detalladamente.

1. ¿Qué tan seguro(a) se siente al explicar la función de cada subsistema presente en la herramienta de apoyo CubeSat? *

	1. Nada	2. Poco	3. Moderado	4. Bastante	5. Mucho
OBC (Raspberry)	◯	◯	◯	◯	◯
ADCS (Sensores de actitud)	◯	◯	◯	◯	◯
EPS (Potencia eléctrica)	◯	◯	◯	◯	◯
Carga útil	◯	◯	◯	◯	◯
Estructura	◯	◯	◯	◯	◯

2. Explica en tus propias palabras las funciones de los subsistemas anteriormente indicados *

3. Explique qué riesgos y limitaciones posee el EPS *

4. Explique los conceptos de integración, AIT, y por qué son relevantes en los ciclos de vida de un proyecto *

5. Question *

Parte 2: Impresiones de la experiencia de integración y testeo

6. ¿Qué tan claras fueron las instrucciones entregadas en el documento de apoyo a la experiencia para cada etapa? *

	1. Muy poco claras	2	3	4	5. Completamente claras
Identificación de componentes	◯	◯	◯	◯	◯
Pruebas de aceptación	◯	◯	◯	◯	◯
Ensamblaje y testeo	◯	◯	◯	◯	◯
Precauciones e instrucciones especiales	◯	◯	◯	◯	◯
Información complementaria	◯	◯	◯	◯	◯

7. ¿Qué actividad le ayudó más a comprender la integración de sistemas? *

8. ¿Las pruebas de aceptación le permitieron comprender cómo cada componente individual afecta al sistema completo? *

◯ Sí

◯ No

9. ¿Hubo algún momento en que el comportamiento del sistema le resultara inesperado o contraintuitivo? Descríbalo brevemente. *

10. Opcional: Comentario libre respecto a la experiencia de integración

Parte 3: Trabajo en equipo

11. ¿Qué tan equilibrada fue la distribución de las tareas dentro de su equipo? *

1	2	3	4	5

12. ¿Qué tan efectiva fue la comunicación y colaboración de su equipo durante la actividad? *

1	2	3	4	5

13. ¿Qué roles o tareas asumió personalmente durante la sesión? *

14. ¿Sintió que alguna tarea fue demasiado fácil, difícil o repetitiva/sin utilidad? *

15. ¿Qué tan necesaria es la guía de un profesor o ayudante durante la experiencia? *

1	2	3	4	5

16. Opcional: Comentarios respecto al trabajo en equipo y el rol de profesores/ayudantes durante la experiencia

Parte 4: Materiales y recursos

17. ¿Qué tan útiles le fueron los siguientes recursos para apoyar su trabajo? (1 = Nada útil, 5 = Muy útil)

	1	2	3	4	5
Guía de experiencia impresa	◯	◯	◯	◯	◯
Guía de experiencia digital (PDF)	◯	◯	◯	◯	◯
Presentación en vivo	◯	◯	◯	◯	◯
Guía directa del instructor	◯	◯	◯	◯	◯
Kit físico de componentes	◯	◯	◯	◯	◯
Herramientas de apoyo (multimetro, balanza, etc)	◯	◯	◯	◯	◯

18. Respecto a la guía de apoyo entregada, evalúe (1 - nada, 5 - mucho) *

	1	2	3	4	5
La utilidad del documento impreso	◯	◯	◯	◯	◯
La claridad de las instrucciones entregadas	◯	◯	◯	◯	◯
La utilidad del documento en PDF	◯	◯	◯	◯	◯
La claridad de las imágenes (impresas)	◯	◯	◯	◯	◯
La claridad de la secuencia de integración	◯	◯	◯	◯	◯
Las indicaciones complementarias sobre herramientas y componentes	◯	◯	◯	◯	◯
El aporte a la capacidad de avanzar independiente mente (sin guía directa del ayudante)	◯	◯	◯	◯	◯

19. ¿Qué tan clara fue la presentación en vivo realizada durante la experiencia? *

1	2	3	4	5

20. ¿Hubo algún recurso que no utilizó o no estuvo disponible pero que considera que habría sido de ayuda? *

21. ¿Tiene sugerencias para mejorar la accesibilidad o claridad de los materiales? *

22. Opcional: Describa brevemente su impresión de los recursos presentes durante la experiencia

Parte 5: Reflexión general

23. ¿Cuál fue el aprendizaje más valioso que obtuvo de esta experiencia? *

24. ¿Cuál fue el aspecto más desafiante o frustrante de la experiencia? *

25. Si realizara esta actividad nuevamente, ¿qué haría diferente? *

26. Califique su satisfacción general con la experiencia *

☆ ☆ ☆ ☆ ☆

27. Opcional: Reflexiones y comentarios libres respecto a la experiencia