UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

CubeSat Fase 3

Trabajo de graduación en modalidad de Megaproyecto presentado por:
Erick Andrés Rodas Montenegro,
Javier Alejandro Alay Pérez, y
Kevin Gabriel Barrios Trujillo
para optar al grado académico de Licenciados en Ingeniería Mecatrónica

Guatemala,
2016

CubeSat Fase 3

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

CubeSat Fase 3

Trabajo de graduación en modalidad de Megaproyecto presentado por:
Erick Andrés Rodas Montenegro,
Javier Alejandro Alay Pérez, y
Kevin Gabriel Barrios Trujillo
para optar al grado académico de Licenciados en Ingeniería Mecatrónica

Guatemala,
2016

Coordinador del Megaproyecto:

(f)_____
(Ing. Víctor Hugo Ayerdi)

Director de Ingeniería Mecánica:

(f)_____
(Ing. Víctor Hugo Ayerdi)

Director de Ingeniería Mecatrónica:

(f)_____
(Msc. Carlos Esquit)

Fecha de aprobación: Guatemala 25 de noviembre del 2016

CONTENIDO

LISTA DE CUADROS

LISTA DE FIGURAS

RESUMEN

Los CubeSat son nanosatélites de características estándar capaces de cumplir una misión espacial con objetivos específicos. Con estos satélites se analiza una problemática a la vez, por lo que los requerimientos de componentes, tamaño, tiempo y dinero son reducidos en comparación con los satélites regulares. A partir de este proyecto se pretende continuar con el desarrollo de un CubeSat 1U que cumpla con los estándares establecidos para este tipo de satélites, y que posteriormente, en el futuro, se pueda concluir en una fase posterior para ser lanzado al espacio. Se espera que el CubeSat se utilice para tomar imágenes, desde el espacio, que servirán para determinar la concentración de cianobactberia en el Lago de Atitlán, para que luego de ser analizadas, sean un elemento para tomar decisiones con el fin de intentar salvar este recurso natural.

Para cumplir con esta misión, el sistema de computadora a bordo, debe ser capaz de transferir la información correspondiente a tres imágenes tomadas por el módulo de la carga útil (Payload por su nombre en inglés) y enviarlas, por medio de comunicación serial, al sistema de comunicación (COMMS por sus siglas en inglés) para que este pudiera realizar la transmisión de la información a una estación en tierra. Se realizó la selección del procesador, dando como resultado la elección del microprocesador de la Raspberry Pi Zero. Al tener el procesador, se realizaron diferentes pruebas para determinar si era capaz de obtener las imágenes tomadas por el módulo payload, convertirlas a arreglos de bits, y enviarlas al módulo COMMS, para lo cual se verificó que este envío se realizara en menos de 4 minutos. Entre los resultados obtenidos se observó que, para poder cumplir con el envío de imágenes en menos de 4 minutos, era necesaria la utilización de imágenes de un tamaño alrededor de 800KB, esto fue debido a que el microcontrolador utilizado para COMMS tenía una tasa máxima de recepción de 115200 baudios. Se logró comprobar que el microprocesador seleccionado cumplió con los objetivos propuestos, se pudo comprobar que se lograba convertir y enviar las imágenes, tomando en cuenta que el principal factor de la tardanza en el envío de datos fue el microprocesador seleccionado para COMMS.

El módulo COMMS debe ser en cargado de poder recibir la información del sistema de computadora a bordo y poder transmitir la información por medio de radiofrecuencia a la estación base en tierra. Para poder lograr esto se debe seleccionar primero, con base a la ventana de comunicación y la cantidad de datos a transmitir, la banda de frecuencia que se va a utilizar. Luego se deben seleccionar cada uno de los componentes que conforman el módulo, tales como el controlador, el *transceiver* y el tipo de antena a utilizar. Con estos componentes es posible integrar el módulo e implementar mediante el controlador un software que permita configurar los registros del transceiver, luego la recepción de los datos del sistema de

computadora a bordo y que envíe estos datos al transceiver para que puedan ser enviados mediante radiofrecuencia.

Por último queda describir el módulo de potencia, este es el encargado de suministrar la energía requerida por el resto de módulos en un CubeSat 1U, la fuente energía del satélite será energía solar, la cual se transformará a energía eléctrica de corriente directa por medio de paneles solares. Además de esto, se debe contar con un sistema de almacenamiento de energía, pues a lo largo de la órbita en la que se encuentre el satélite, no permanecerá siempre en contacto con el Sol, por lo que debe ser capaz de almacenar la energía suficiente para poder funcionar sin llegar a apagarse y perder el control, pues esto podría hacer perder la estabilidad, siendo un riesgo para la misión.

I. INTRODUCCIÓN

A lo largo del tiempo, la evolución del hombre lo ha llevado a crear cosas inimaginables. Toda invención humana ha sido producto de querer imitar a nuestra madre tierra, a la naturaleza, y en ese proceso de imitación se ha dado cuenta que lo primordial es la observación. Tanto así ha sido la importancia de observar que, al momento de definir un método científico para llegar a un resultado objetivo, la observación es el primer paso para todo procedimiento. Como toda innovación, y como toda evolución en la tecnología, el hombre ha creado para observación grandes instrumentos, un ejemplo de ellos es el telescopio Hubble. Este telescopio actualmente se encuentra orbitando y gracias a él hemos podido obtener información que el hombre hace algunos años jamás hubiera imaginado tener. La desventaja de tener proyectos como este y como muchos otros para la observación terrestre es que requieren demasiada ingeniería y demasiados recursos que pocas instituciones a nivel mundial pueden llegar a tener. Sin embargo, como el hombre busca la innovación siempre y el avance en la tecnología, ha llegado a desarrollar soluciones como los nanosatélites, entre ellos el tipo CubeSat. Estos satélites son primordialmente utilizados en la observación terrestre, a través del remote sensing. Este tipo de satélites son de menor costo comparado con un satélite convencional y requieren de menor tiempo para lograr su desarrollo. Esto hace posible que incluso instituciones universitarias puedan implementar este tipo de proyectos por estudiantes de últimos años de algunas carreras como ingeniería y ciencias puras. Estos satélites están compuestos por diferentes módulos que hacen posible en conjunto, lograr una misión en específico. En Guatemala tenemos un recurso hidríco de gran importancia, el Lago Atitlán, sin embargo, este se ve actualmente afectado por la cianobacteria que se reproduce de manera más rápida, siendo algunos de sus efectos negativos la contaminación, afectando la fauna y flora del lago, así como posiblemente, la generación de enfermedades entre las personas que consumen el agua del lago. Las mediciones que se realizan actualmente de la cantidad de cianobacteria en el lago se hacen aproximando mediciones de cinco diferentes puntos del lago mensualmente, lo que no permite conocer exactamente la situación en todo el lago. Una solución que optimizaría la obtención de información es utilizar un satélite tipo CubeSat para poder tomar fotografías del lago y mediante procesamiento de imagen obtener el porcentaje de cianobacteria del lago con una frecuencia más elevada. Para poder desarrollar esta solución, a continuación, se presentan los diseños para el módulo de computadora de abordo, encargado de llevar el control del funcionamiento del satélite; el módulo de comunicaciones, encargado de descargar la información captada por el satélite a la base en tierra y el módulo de potencia, el cual es encargado de suministrar la cantidad necesaria de voltaje y corriente a cada componente.

II. OBJETIVOS

A. GENERAL DEL MEGAPROYECTO

Construir un CubeSat 1U para realizar pruebas de laboratorio de termovacío, vibraciones y de condiciones ambientales para una órbita LEO.

B. GENERAL DEL MÓDULO DE COMPUTADORA A BORDO

Elegir y diseñar una computadora a bordo para un CubeSat, capaz de integrar los módulos de Comunicación, Payload y Potencia.

C. ESPECÍFICOS DEL MÓDULO DE COMPUTADORA A BORDO

1. Investigar y elegir un microprocesador adecuado para cumplir con los requerimientos del módulo de computadora a bordo.
2. Convertir imágenes tomadas por el módulo Payload a arreglos de bits y enviarlas al módulo de comunicación
3. Desarrollar un algoritmo sobre el protocolo de comunicación UART que permita el envío de 3 imágenes en menos de 4 minutos.
4. Crear un programa que pueda orquestar a los módulos del CubeSat al ser encendido el microcontrolador.

D. GENERAL DEL MÓDULO DE COMUNICACIÓN

Diseñar e implementar un sistema de comunicación integrado al CubeSat.

E. ESPECÍFICOS DEL MÓDULO DE COMUNICACIÓN

1. Definir la frecuencia a utilizar para la comunicación del satélite.
2. Analizar y definir el tipo de antena a implementar en el CubeSat.
3. Definir los componentes a utilizar en el módulo de comunicaciones para que cumplan la función de Terminal Node Controller y transceiver.
4. Definir los circuitos de aplicación de los componentes a utilizar en el módulo de comunicaciones.
5. Integrar los componentes del módulo en sus circuitos de aplicación para formar el subsistema de comunicaciones del CubeSat.
6. Programar el subsistema de comunicaciones para poder enviar y recibir información.
7. Integrar el subsistema de comunicaciones con la computadora de abordo del CubeSat

F. GENERAL DEL MÓDULO DE POTENCIA

Diseñar y analizar un sistema que pueda generar y almacenar la energía necesaria para cada uno de los componentes electrónicos dentro del CubeSat 1U.

G. ESPECÍFICOS DEL MÓDULO DE POTENCIA

1. Diseñar e implementar un sistema de comunicación integrado al CubeSat.
2. Diseñar y analizar un sistema que pueda generar y almacenar la energía necesaria para cada uno de los componentes electrónicos dentro del CubeSat 1U de la Universidad del Valle de Guatemala.
3. Elegir y diseñar una computadora a bordo para un CubeSat, capaz de integrar los módulos de Comunicación, Payload y Potencia.

III. JUSTIFICACIÓN

Los CubeSat fueron la opción adecuada para que la industria aeroespacial redujera la utilización de recursos para desarrollar un proyecto de interés especial. Con estos satélites se analiza una problemática a la vez por lo que los requerimientos de componentes, tamaño, tiempo y dinero disminuyen notablemente. Esto es lo que se pretende aprovechar en este proyecto para un estudio medioambiental de relevancia para Guatemala.

En Guatemala es de especial interés el estudio de la cianobacteria en el Lago de Atitlán, sin embargo, este análisis requiere una gran cantidad de recursos porque abarca una amplia extensión de área. Esta investigación se puede realizar de forma más eficiente desde el espacio que desde el propio lago, ya que la medición remota de este fenómeno puede abarcar una mayor cantidad de puntos de control. Actualmente el proceso para la medición in situ de cianobacteria se hace en 5 puntos del lago con una frecuencia de 1 vez al mes. Emplear un CubeSat utilizando capturas con resolución espacial aproximada de 30 x 30 metros, puede proporcionar información de la concentración de cianobacteria en toda el área del lago, con una mayor frecuencia. Con estos datos sobre la problemática se podrá coordinar los esfuerzos y tomar decisiones acertadas para salvar este recurso natural, patrimonio nacional.

Este proyecto también está enfocado al aspecto académico, ya que es el primer acercamiento, tanto de la Universidad, como de Guatemala, de poner en órbita un satélite. Los conocimientos obtenidos por este proyecto, y el interés producido por la investigación y desarrollo del mismo, pueden ser oportunidades para nuevos proyectos, carreras y profesionales en el área.

IV. MARCO TEÓRICO

A. ¿QUÉ ES UN CUBESAT?

1. Satélite. Un satélite es una luna, planeta o máquina que gira alrededor de un planeta o una estrella. Por lo general, la palabra "satélite" se refiere a una máquina que se lanzó al espacio y se mueve alrededor de la Tierra o de otro cuerpo en el espacio. La Tierra y la Luna son ejemplos de satélites naturales. (Chanes, y Valencia. 2014)

2. Satélite artificial. Los satélites artificiales son máquinas hechas por el hombre y generalmente son los satélites que orbitan alrededor de la Tierra. Algunos toman fotografías del planeta, que ayuda a los meteorólogos a predecir el tiempo y rastrear los huracanes, otros toman fotografías de planetas, el Sol y galaxias lejanas. Estas imágenes ayudan a los científicos a entender mejor el sistema solar y el universo. Pero la mayoría de satélites se utilizan para las comunicaciones, como la radiante de señales de televisión y telefonía. (Chanes, y Valencia. 2014)

3. Clasificación de satélites artificiales. Los satélites artificiales se pueden clasificar de acuerdo a su altitud, tamaño y tipo de misión. La clasificación de satélites contiene las características más importantes. (Chanes, y Valencia. 2014)

Clasificación por su altitud (Chanes, y Valencia. 2014)

- ✓ Órbita baja terrestre (LEO): Una órbita geocéntrica con una altitud de 0 a 2000 km.
- ✓ Órbita media terrestre (MEO): Una órbita geocéntrica con una altitud entre 2000 km y hasta el límite de la órbita geosíncrona. También se la conoce como órbita circular intermedia.
- ✓ Órbita alta terrestre (HEO): Una órbita geocéntrica por encima de la órbita geosíncrona de 35,786 km; también conocida como órbita muy excéntrica u órbita muy elíptica.

Clasificación de los satélites por su tamaño (Chanes, y Valencia. 2014)

- ✓ Grandes satélites: cuyo peso sea mayor a 1000 kg.
- ✓ Satélites medianos: cuyo peso sea entre 500 y 1000 kg.
- ✓ Minisatélites: cuyo peso sea entre 100 y 500 kg.
- ✓ Microsatélites: cuyo peso sea entre 10 y 100 kg.
- ✓ Femtosatélites: cuyo peso sea menor a 100 g.

- ✓ Nanosatélites: cuyo peso sea entre 1 y 10 kg.
- ✓ Picosatélites: cuyo peso sea entre 0,1 y 1 kg.

Clasificación por el tipo de misión (Chanes, y Valencia. 2014)

- ✓ Satélites astronómicos: son satélites utilizados para la observación de planetas, galaxias y otros objetos astronómicos.
- ✓ Satélites de comunicaciones: son los empleados en las telecomunicaciones. Utilizan órbitas geosíncronas, órbitas de molniya u órbitas bajas terrestres.
- ✓ Satélites de observación terrestre: son utilizados para la observación del medio ambiente, meteorología, cartografía y para fines militares, son satélites utilizados principalmente para registrar el tiempo atmosférico y el clima de la Tierra, y Satélites de navegación.

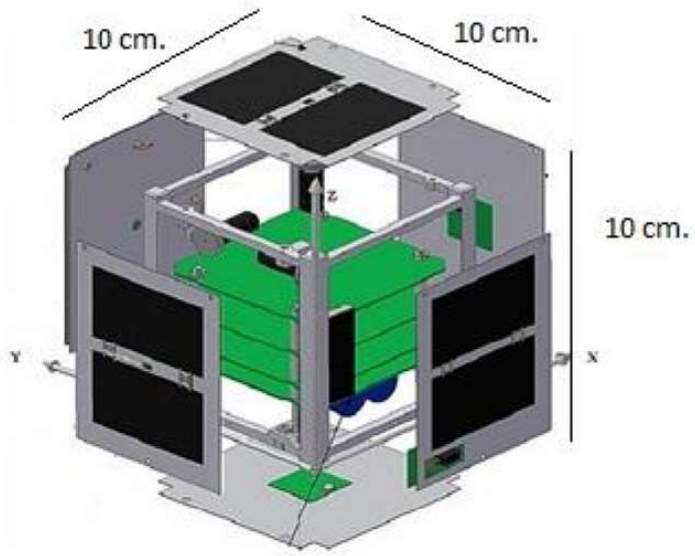
4. CubeSat. Los satélites CubeSat por su reducido tamaño están considerados dentro de los picosatélites, estos satélites pequeños tienen una tarea específica como lo es medir los niveles de radiación en la atmósfera, detectar cambios climáticos, entre otros. A estas tareas específicas que realizan los satélites se les conoce como carga útil. (Chanes, y Valencia. 2014)

5. Características principales de los satélites CubeSat. Las características que debe poseer un CubeSat es tener un tamaño y peso reducido, poseer una carga útil dependiendo del tipo de tarea que va a desempeñar, contar con dispositivos electrónicos resistentes a las extremas condiciones del espacio exterior, etc. Además, cada subsistema del satélite CubeSat debe contar con una función específica que asegure su buen funcionamiento. (Chanes, y Valencia. 2014)

Existen diferentes modelos y tamaños para un Cubesat, sin embargo, hay normas y protocolos que rigen a estos satélites. Un picosatélite puede contener varias cargas útiles, pero esto incrementa su tamaño y costo. Evidentemente el tiempo de construcción es más largo si el CubeSat realiza múltiples tareas. (Chanes, y Valencia. 2014)

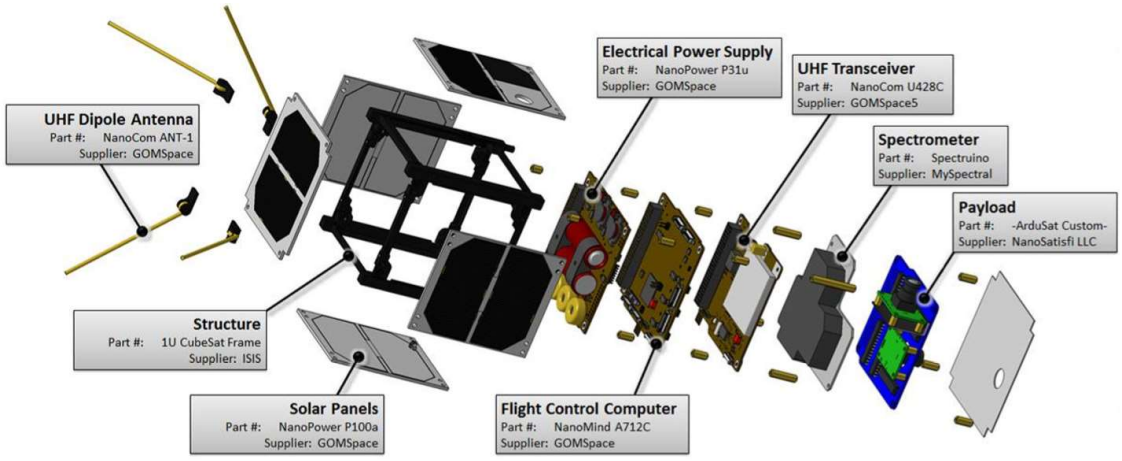
6. Arquitectura y dimensiones del satélite CubeSat. Las especificaciones CubeSat describen al satélite como un sistema en forma de cubo de 10 x 10 x 10 cm por lado, tienen un peso no mayor a 1 kg y su centro de masa se encuentra dentro de 2 cm de su centro geométrico. El acoplamiento de picosatélites permite diseñar sistemas de 2 o 3 Kg que conservan dos de sus dimensiones y una varía entre 227 mm y 340 mm, respectivamente. La siguiente figura muestra la forma geométrica de un picosatélite CubeSat. (Chanes, y Valencia. 2014)

Figura 1. Forma de un CubeSat.



Su arquitectura consiste en un conjunto de subsistemas de suministro de energía, determinación y control de posición, sistema de comunicaciones, sistema de sensores, sistema de manejo de comandos (computadora de vuelo) y un sistema de carga útil, principalmente. Tales subsistemas, en un CubeSat, se implantan por medio de tarjetas electrónicas de 9x9x1 cm interconectadas entre sí. (Chanes, y Valencia. 2014)

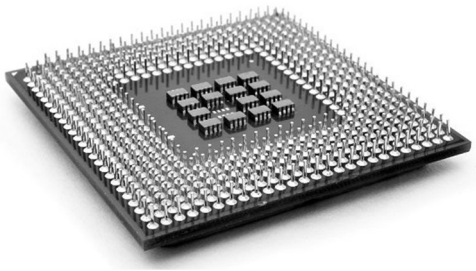
Figura 2. Estructura de un CubeSat.



B. MÓDULOS DE UN CUBESAT

1. Módulo de computadora de abordo. El módulo OBC tiene como función principal el orquestar a todos los subsistemas que integran al satélite, es decir que este funciona como un cerebro para el satélite. Este sistema es encargado de comunicarse con el resto de subsistemas y para la transmisión de información entre ellas. Este es el módulo que se estará expandiendo en el resto del documento. (Hamama, Zovaro, & Wu, 2006) (Swartwout, 2013)

Figura 3. Imagen de microprocesador ejemplificando al usado por el OBC.



a. Imágenes. Para poder cumplir con la misión propuesta para el CubeSat se debe hacer uso de una cámara para el módulo de payload, lo cual implica que se estarán manejando imágenes dentro del satélite que serán enviadas a una estación en tierra para poder ser analizadas. Las imágenes que se lleguen a tomar dentro del satélite pueden llegar a contener una vasta información útil, por lo cual es necesario conocer los diferentes tipos de formatos existentes en las cámaras y las ventajas y desventajas de la utilización de los mismos. Existen diferentes formatos de imágenes utilizados para diferentes aplicaciones, entre los diferentes formatos se pueden observar el RAW, JPEG, PNG, TIFF, etcétera.

1) RAW. Las imágenes RAW son un tipo de formato donde se obtiene la mayor cantidad de información por pixel que cualquier otro formato, esto es debido a que es un formato sin compresión, es decir que no ha pasado pon ningún tipo de procesamiento de imagen. Este formato es usado en aplicaciones en donde es necesario el acceso directo a la información de cada pixel, realizándolo sin la necesidad del uso de cálculos complejos para determinar la ubicación de los pixeles dentro de una corriente de datos comprimidos. El uso de este formato simplifica la lectura de la imagen para aplicaciones donde se realiza un procesamiento de imagen orientado a pixeles. Este también tiene la cualidad de ser usado en aplicaciones en el que se necesita el reescribir datos a imágenes. Este tipo de imágenes consisten en 4 bytes como identificador, seguido de 30 bytes de encabezado de la imagen y luego por un espacio arbitrario de 0 o más bytes seguido por los datos de los pixeles.

Cuadro 1. Estructura de los bytes 0:33 del encabezado de las imágenes RAW

Nombre	Byte(s)	Descripción
Identificador de imagen	0:3	Contiene 4 bytes de valores ASCII para indicar que está en formato RPIX, sirven para identificar la imagen como una imagen de pixeles RAW.
Largo de encabezado	4:7	Este espacio es utilizado para saber dónde comienzan los datos de la imagen de pixeles tipo RAW. Usualmente los datos empiezan en el byte 34.
Versión mayor	8	Contiene el número de la versión mayor para la codificación de la imagen. La versión actual es la 1.0, por lo tanto, el campo sería 1.
Versión menor	9	Contiene el número de la versión menor para la codificación de la imagen. La versión actual es la 1.0, por lo tanto, el campo sería 0.
Ancho de imagen	10:13	Contiene el ancho de pixeles de la imagen.
Altura de imagen	14:17	Contiene la altura de pixeles de la imagen.
Tipo de compresión	18	Contiene el tipo de compresión, se especifica en el Cuadro 2.
Orden del pixel	19	Este campo describe el orden del pixel dentro de la imagen. Típicamente, en un escaneo por línea, los pixeles están ordenados de izquierda a derecha por lo que se coloca 1 en este byte, sin embargo, ciertas aplicaciones requieren un orden de derecha a izquierda por lo que se coloca 2 para indicar este orden.
Orden del escaneo por línea	20	Este campo describe el orden de escaneo por línea. Usualmente son ordenados de arriba hacia abajo por lo que se coloca 1 en este byte, sin embargo, existen aplicaciones donde se requiere el uso de un orden de abajo hacia arriba por lo que se coloca 2 para indicar este orden.
Intervalo	21	Describe el intervalo de varios tipos de bandas dentro de la imagen RAW, referirse al Cuadro 3 para mayor información.
Número de bandas	22	Contiene el número de bandas o planos en la imagen, este debe ser un valor entre 1 y 255.
Número del canal rojo	23	Este campo especifica el número de banda que se utiliza como el canal rojo durante las operaciones de conversión de la imagen.
Número del canal verde	24	Este campo especifica el número de banda que se utiliza como el canal verde durante las operaciones de conversión de la imagen.
Número del canal azul	25	Este campo especifica el número de banda que se utiliza como el canal azul durante las operaciones de conversión de la imagen.
Área reservada	26:33	Esta área debe contener 0.

Cuadro 2. Tipos de compresión de imágenes RAW.

Valor	Nombre	Compresión
1	Ninguno	No hay compresión.
2	FAX3	Compresión del grupo 3 CCITT.
3	FAX4	Compresión del grupo 4 CCITT.

Cuadro 3. Tipos de intervalo para varios tipos de bandas dentro de las imágenes RAW.

Valor	Nombre	Intervalo
1	BIP	Intervalo de banda por pixel, o también llamada "chunky". Coloca las diferentes bandas o canales de cada pixel secuencialmente al siguiente pixel. Si las bandas son RGB, se observaría de la siguiente forma: Línea de escaneo 1: RGBRGBRGBRGBRGBRGB Línea de escaneo 2: RGBRGBRGBRGBRGBRGB Línea de escaneo 3: RGBRGBRGBRGBRGBRGB
2	BIL	Intervalo de banda por línea. Coloca las diferentes bandas del pixel esparcidas en las líneas de escaneo, colocadas secuencialmente. Si las bandas son RGB, se observaría de la siguiente forma: Línea de escaneo 1: RRRRRRRRRRRRRRRRRR GGGGGGGGGGGGGGGGGG BBBBBBBBBBBBBBBBBB
3	BSQ	Intervalo de banda secuencial o también llamada "planar". Coloca las diferentes bandas del pixel esparcida en diferentes planos de la imagen. Si las bandas son RGB, se observaría de la siguiente forma: Plano 1: RRRRRRRRRRRRRRRRRR RRRRRRRRRRRRRRRRRR Plano 2: GGGGGGGGGGGGGGGGGG GGGGGGGGGGGGGGGGGG Plano 3: BBBBBBBBBBBBBBBBBB BBBBBBBBBBBBBBBBBB

A partir del byte 33, puede existir un espacio entre el encabezado de la imagen y los datos de los pixeles de la imagen, en caso de no haber espacio, los datos de la imagen comenzarían inmediatamente. (Oracle, 2016)

2) JPEG. Existen diferentes formatos para poder observar una imagen, estos pueden ser imágenes no comprimidas, como las imágenes RAW que contiene toda la información obtenida del sensor de la cámara, y también existen las imágenes comprimidas, como las imágenes JPEG.

El método Joint Photographic Experts Group (JPEG por sus siglas en inglés), fue creado en el año de 1992. Este formato es un estándar de compresión de imágenes para reducir el tamaño del archivo, pero no tiene efecto en el número de pixeles en la imagen. Debido a esta reducción del tamaño del archivo, este tipo de compresión es llamado "Lossy" o compresión con pérdidas, esto quiere decir que cierta información de la imagen se pierde al realizar la compresión, y al tratar de hacer la descompresión, debido a las perdidas mencionadas, no se obtendrá como resultado la imagen original. Esto nos indica que, al realizar compresión JPEG, se pierde cierta calidad en detalle de la imagen.

Este es un formato usado en muchas cámaras, cada una de las cuales tienen diferentes formas de compresión. Dependiendo la calidad que se desee obtener en una imagen es el tamaño del archivo que se obtiene como resultado, esto impacta directamente en la calidad de la imagen.

Se debe tomar en cuenta que las imágenes comprimidas como JPEG son acumulativas en su pérdida de datos, es decir que, si se tiene una imagen JPEG y se le realiza algún tipo de edición, al guardarla como JPEG nuevamente se vuelve a comprimir la imagen, lo cual afecta directamente a la calidad de la imagen debido a que se perderán más datos. (Nikon, 2013)

Para poder comprimir una imagen a JPEG, se deben realizar cuatro fases, dividir la imagen, convertirla al dominio de frecuencias, cuantización y codificación de entropía. La primera parte inicia con la separación de la imagen en bloques de 8x8 pixeles, esto permite que el algoritmo de compresión puede tomar ventaja del parecido en el color de los pixeles en espacios pequeños.

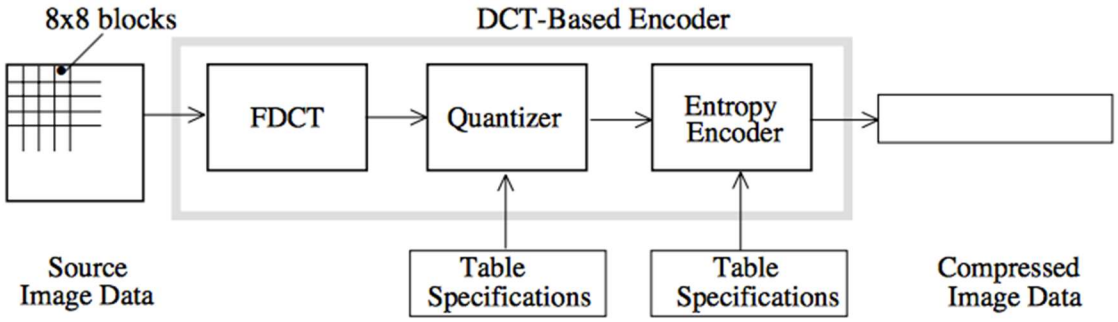
La segunda fase consiste en la conversión al dominio de frecuencias. Esta conversión hará que la parte de cuantización sea más fácil, debido a que se convierte la intensidad de un pixel en la amplitud de una función de coseno única, esto se hace con el fin de poder remover la información que no sea útil para la imagen. Se desecha las frecuencias altas debido a que el ojo humano no puede percibir los colores que se presenten a altas frecuencias, por lo que, al removerlas no se percibe una gran diferencia y para un humano tiene un gran parecido a la imagen original. Existen diferentes formas para pasar al dominio de frecuencias, entre estos se encuentran, la transformada rápida de Fourier (FFT por sus siglas en inglés) y la

transformada discreta de cosenos (DCT por sus siglas en inglés), siendo este último el más utilizado debido a que es de las formas más rápidas para pasar al dominio de frecuencias.

Para la tercera parte de la compresión se realiza un cuantización, esto consiste en la división de los valores de frecuencias entre una constante predeterminada, y luego haciendo un redondeo al entero más cercano del resultado entre esta división. Esta parte es en la que el usuario determina la calidad del cómo desea la imagen, esto es debido a que el valor por el que se dividirán las frecuencias dependerá de la calidad propuesta por el usuario, entre más grande sea el valor a dividir, menor será la calidad de la imagen por la pérdida de datos.

Por último, se realiza la codificación de entropía haciendo uso de las matrices que fueron divididas en pixeles de 8x8, y que luego fueron transformadas y cuantizadas. Esta fase consiste en codificar un grupo de símbolos en la mínima cantidad de bits requeridos para representar a estos. Entre los codificadores de entropía más populares se encuentran la codificación de Huffman y la codificación de Aritmética. (Kashyap, 2004) (Wallace, 1991)

Figura 4. Compresión de imagen a archivo JPEG.



b. Interfaz UART. El Transmisor/Receptor Asíncrono Universal (UART por sus siglas en inglés) se basa en el estándar industrial de elementos de comunicación asíncrona TL16C550. Este es un tipo de comunicación serial que trabaja con el concepto de primero en entrar, primero en salir (FIFO por sus siglas en inglés). (Texas Instruments, 2010)

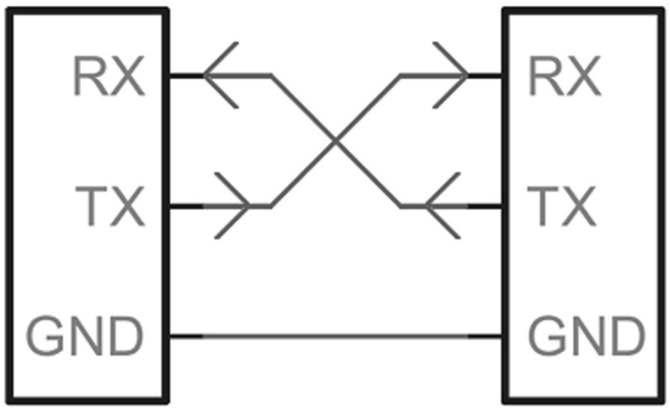
Esta interfaz realiza la transmisión de datos separando cada byte en una secuencia de bits para ser enviados, como se muestra en la Figura 4, se hace uso de un bit para indicar que se inicia el envío de un byte, los 8 bits del byte que se desea transmitir y un bit indicando que acaba de terminar la transmisión del byte, luego el receptor obtendrá primero el byte indicando el inicio de la transmisión, para luego poder juntar los siguientes 8 bits y por último, convertir los bits recibidos en el byte enviado, esta acción se realiza para cada uno del bytes a transmitir.

Figura 5. Modelo de transmisión de datos con la interfaz UART.

El método de transmisión serial asíncrona, permite que la información sea transmitida sin la necesidad que el transmisor envíe la señal de reloj al receptor, este tipo de transmisión depende de la selección previa de una velocidad de transmisión, y se le agregan bits para determinar el inicio y fin de transmisión de datos, así como se muestra en la Figura 4. Existe un estándar para las velocidades de transmisión, cada uno de los cuales está dado en baudios por segundo, estas velocidades pueden ser 1200, 2400, 4800, 9600 así sucesivamente, cada uno de los cuales indica que se están enviando esa cantidad de bits por segundo, es decir que cómo se están enviando 10 bits para poder transmitir un byte, con una velocidad de 9600 se estarían enviando hasta 960 bytes por segundo.

Para poder conectar la interfaz UART, se debe realizar la conexión como se muestra en la Figura 5, la cual indica que se debe colocar el pin Rx del transmisor conectado al pin Tx del receptor y viceversa, y la tierra del receptor conectada a la tierra del transmisor. (Sparkfun, 2014) (Durda, 2014)

Figura 6. Modo de conexión para la comunicación UART.



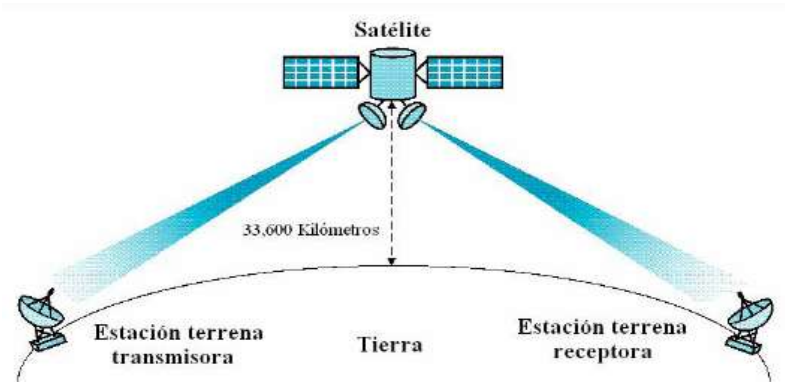
2. Módulo de comunicaciones

a. Radiofrecuencia.

En un satélite, una de las características más importantes es poderse comunicar con una estación en tierra que pueda obtener datos de él, tanto para obtener los datos de la carga útil como el estado de salud del satélite. Tomando en cuenta que, si no existe una vía de comunicación hacia la tierra, el satélite sería únicamente una estructura con componentes girando alrededor de la tierra, prácticamente basura espacial. La comunicación se hace utilizando el espectro electromagnético en el rango de las frecuencias de radio, mediante este tipo de onda utilizando una modulación y un protocolo de comunicación se puede transmitir y recibir información.

Existen diferentes rangos de frecuencia para las ondas de radio, que van desde los 30 Hz hasta los 300 GHz, utilizar un rango de frecuencias o una frecuencia específica va a depender de la aplicación que requiere el uso de radiofrecuencia para transmitir y recibir información. Para CubeSat, las frecuencias que se utilizan están en el rango de bandas de VHF (30-300 MHz) y UHF (300-3000 MHz). El uso de bajas o altas frecuencias se diferencian en ciertas características tales como; la eficiencia en la propagación de las ondas, inmunidad a cierto ruido y el tamaño de la antena transmisora-receptora. El uso de altas frecuencias se debe a que es más eficiente la propagación, presenta más inmunidad al ruido y el tamaño de la antena disminuye. Esto último se fundamenta en el hecho que las antenas usualmente son ¼ de la longitud de onda de la frecuencia que estemos utilizando. Si aumentamos en frecuencia disminuimos en longitud de onda y por tanto necesitaríamos antenas más pequeñas para poder transmitir y recibir ondas. (National Instruments, 2015)

Figura 7: Transmisión de datos de un satélite a estaciones en tierra mediante radiofrecuencia.



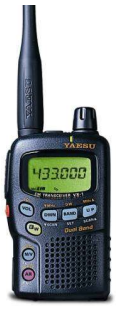
Las radiofrecuencias para aplicaciones de transmisión de datos en el CubeSat se concentran en la banda de radio amateur en su mayoría. Para algunos integrantes de la comunidad de radio amateur, los CubeSat únicamente consumen las frecuencias disponibles en esa banda, sin embargo, el hecho de tener CubeSats con sistemas de comunicaciones operando en el rango de frecuencias amateur trae algunos beneficios a la comunidad, tales como operadores de radio amateur, nuevos esquemas de modulaciones, colaboración internacional y educación de una nueva generación de operadores radio amateur entre otros. (Klofas et. Al., 2008) El uso de frecuencias amateur permite tener un ancho de banda necesario para poder llevar a cabo las misiones con un equipo de radio de un costo razonable con el único requisito de tener un operador licenciado en frecuencias de radio amateur al momento de utilizar la estación en tierra y una licencia respectiva para poder operar en esas frecuencias. El rango de frecuencias amateur permite obtener estas licencias de una manera más simplificada que otro tipo de rango de frecuencias. (Mehrparvar, 2014)

Utilizar radiofrecuencias para la comunicación de un satélite también representa tener en consideración ciertas restricciones, una de ellas es la ventana de comunicación que se pueda tener. Durante la órbita del satélite existe un momento en el cual la estación en tierra puede establecer contacto con el satélite y puede transmitir información al satélite y recibir información de él. La duración que tiene este momento es llamado ventana de comunicación y está definido por los parámetros de la órbita que tendrá el satélite. Esta ventana de comunicación está en el rango de minutos, no sobrepasando los 10 minutos de duración, aunque esta ventana se puede dar varias veces al día. Es por esto que una comunicación de tipo *Store-and-foward* puede ser utilizada para la comunicación del satélite. Este tipo de arquitectura de comunicación funciona de tal manera que envía algunos paquetes en una primera ventana de comunicación, luego al establecer una segunda ventana de comunicación continúa enviando los paquetes desde el último paquete que envió. Esto es muy útil puesto que las ventanas de comunicación son muy cortas, pero se tiene la ventaja que existen varias ventanas de comunicación en un rango de 24 horas. (Mehrparvar, 2014)

b. Tipos de módulo de comunicaciones. Los sistemas de comunicación utilizados en CubeSat pueden ser clasificados en tres grandes ramas, basado en el tipo de *hardware* que utilizan: (Noe, 2004) (Klofas *et. Al.*, 2008)

1) Dispositivos portátiles modificados. Los dispositivos portátiles de radio amateur modificados son placas de circuitos modificados de tal manera que contienen una interfaz para poder interactuar con el módulo de potencia y el procesador del CubeSat. Este tipo de sistemas simplifica la integración del módulo con el satélite puesto que todos los componentes están integrados en una sola placa. Sin embargo, tiene sus desventajas en cuanto a tamaño puesto que al tener todo un radio amateur integrado en una sola placa no se puede tener un tamaño adecuado para un CubeSat puesto que, dentro de él, el tamaño ya es reducido. Es por esto que muchas veces estos dispositivos son sometidos a remoción de las carcasas y componentes que no se utilizarán para poder reducir el volumen y la masa del sistema, puesto que es algo limitado en un CubeSat. También son maquinados de adaptación tales como barrenado para poder montarlo y adaptarlo, entre otros. Otra problemática significante en el uso de dispositivos portátiles de radio amateur modificados, es que se limita en cuanto a configuraciones y problemáticas con el consumo de potencia, se debe re programar en la mayoría de los casos el *transceiver* y se debe de incrementar la potencia de transmisión. Este tipo de modificaciones requieren ayuda externa del proveedor. Entre otras limitantes podemos encontrar también que muchos de los dispositivos portátiles son diseñados para condiciones de uso en tierra. Por tanto el uso de estos dispositivos en el espacio representa modificaciones que se deben hacer para que pueda funcionar desde el espacio, o adaptaciones que se deban hacer al satélite para que puedan operar estos aparatos haciendo que sean una opción no muy viable.

Figura 8: Dispositivo de radio portátil Yaesu VX-1R utilizado por algunos CubeSat



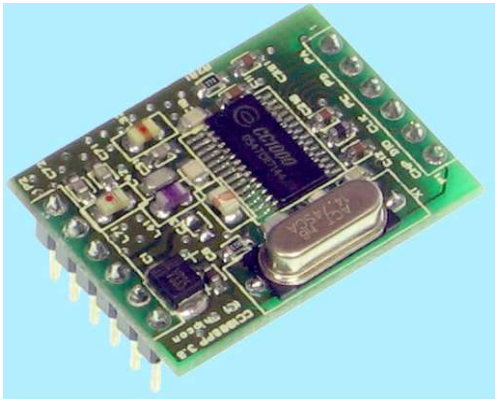
2) Dispositivos transceiver comerciales. Estos *transceiver* de uso aeroespacial simplifican los diseños del sistema de comunicaciones. Usualmente ya vienen configurados con un protocolo de comunicación, paquetización y control de error. Los protocolos y modulación utilizados son muchas veces propios de los fabricantes forzando a que se utilice una radio idéntica en la estación en tierra para la obtención de los datos. Una gran desventaja de este tipo de dispositivos es que son de costo más alto comparado con otros tipos de sistemas de comunicaciones para CubeSat y algunos son de tamaños que no son óptimos para CubeSat puesto que resultan ser muy grandes y ocupan mucha masa y volumen.

Figura 9: Transceiver comercial para CubeSat Syrlinks EWC31

3) Hardware modificado. Este tipo de sistema se refiere a diseñar e implementar un transceiver de componentes individuales. Esto requiere un proceso de diseño más profundo en cuanto a circuitos de radiofrecuencia y por tanto un conocimiento más sólido en la transmisión de información por medio de ondas de radio. Este tipo de diseños han sido menos exitosos debido a la complejidad que tiene el diseño de circuitos y placas de radiofrecuencia.

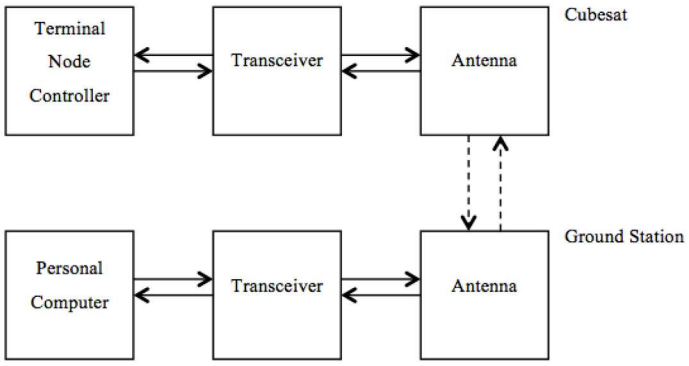
Este tipo de sistemas se componen de un Terminal Node Controller el cual es, usualmente, un microcontrolador para hacer la interfaz con un transceiver al cual se le configuran los registros que usualmente son adquiridos de desarrolladores como *Texas Instruments*, RF Microdevices o Analog Devices.

Figura 10: Transceiver CC1000 de *Texas Instruments* en su circuito de aplicación



Un sistema de comunicaciones utiliza un transceiver, un controlador usualmente llamado terminal node controller y dependiendo del presupuesto de enlace de la aplicación, se puede optar por un amplificador. El diagrama básico del sistema de comunicaciones de un CubeSat es com el que se muestra en la Figura 11.

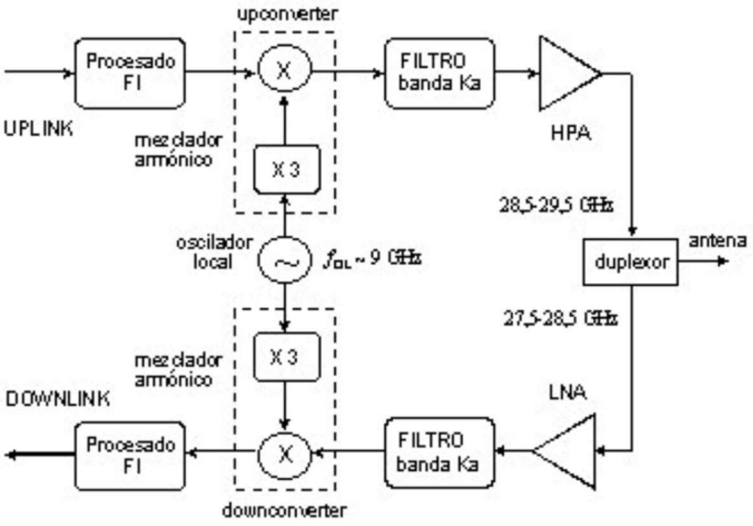
Figura 11: Diagrama de bloques de un sistema de comunicaciones



En la Figura 6 se muestran las interconexiones que hay entre los componentes que conforman el sistema de comunicaciones. La línea punteada denota la comunicación inalámbrica por medio de radiofrecuencia que tiene el sistema de comunicaciones del CubeSat con la estación en tierra. El diseño de una estación en tierra, utiliza los mismos componentes que el sistema de comunicaciones del CubeSat, a diferencia que el *transceiver* utilizado puede conectarse directamente con un ordenador. (Traussnig, 2007) Los componentes básicos que conforman el módulo de comunicaciones son:

a) Transceiver. Un transceiver es el encargado de poder transmitir la información de bytes a radiofrecuencia. Es un dispositivo de radio que transmite y recibe, utilizando ambos un circuito común. De manera general, un *transceiver* funciona de la siguiente manera:

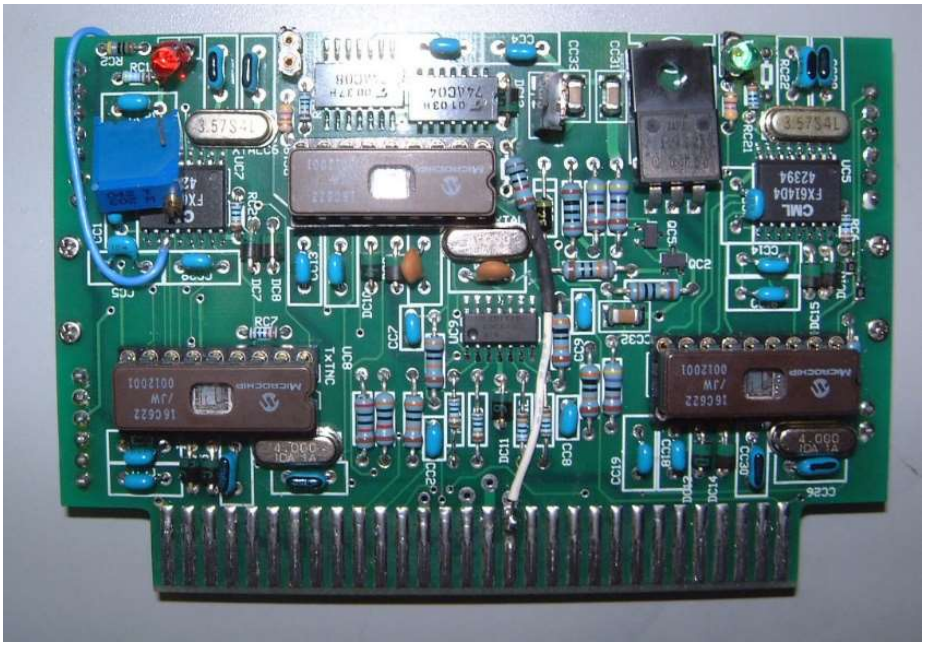
Figura 12: Diagrama de bloques del funcionamiento de un transceiver.



De la Figura 7 observamos de manera general el funcionamiento de un *transceiver*, el cual recibe datos por medio de la antena, es amplificada y pasa por un filtrado de señal, es multiplicada con la frecuencia base y luego es procesada. De igual manera si se desea enviar información entonces se hace el proceso inverso, se multiplica la señal procesada por la frecuencia base, es filtrada, amplificada y enviada a la antena. Con un duplexor se puede seleccionar el modo de operación del *transceiver*, es decir si éste está enviando o recibiendo. (Honcharenko, 1997)

b) Terminal Node Controller. Este componente es el encargado de hacer una interfaz entre la computadora de abordo del satélite y el *transceiver* del módulo de comunicaciones. Es este componente quien hace la aplicación del protocolo de comunicación a utilizar y permite obtener y enviar los datos serialmente entre la computadora de abordo y el módulo de comunicaciones. (Traussnig, 2007) Es también el que configura el *transceiver* para poder establecer los parámetros que requiere para su funcionamiento y el que envía esos valores al *transceiver*. Los *Terminal Node Controller* suelen ser microcontroladores puesto que contienen protocolos de comunicación varios para poder hacer la interfaz entre el *transceiver* y la computadora de abordo del satélite. (Noe, 2004)

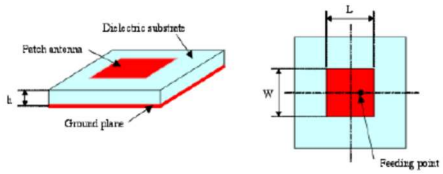
Figura 13: Terminal Node Controller del CubeSat japonés XI



c) Antena. Las antenas son componentes esenciales en la transmisión de datos mediante ondas electromagnéticas de radio. Son ellas las que hacen la transición de pasar una onda de un bus de datos al ambiente, y también realizan el trabajo en vía contraria, hacen que una onda en el ambiente pase a un bus de datos en un circuito. Existen diferentes tipos de antenas, entre ellas las antenas de parche, las antenas helicoidales y las antenas incrementales. (Traussnig, 2007)

1) Antenas de parche. Son antenas que utilizan pistas pequeñas de cobre, como las mostradas en la Figura 9. Las ventajas de este tipo de antenas es que pueden tomar cualquier forma geométrica y no ocupan mucha masa y volumen, características que son muy ventajosas en un CubeSat cuya gran restricción es la masa y el volumen de los componentes. Consiste en un plato rectangular de metal impreso en un material dieléctrico. Sin embargo, este tipo de antena requiere de un satélite estable y que la antena este directamente apuntando a la Tierra en todo momento para establecer comunicación con la base en Tierra.

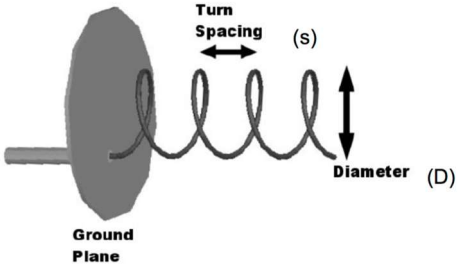
Figura 14: Geometría de una antena de parche de micro pistas



(Traussnig, 2007)

2) Antenas helicoidales. Este tipo de antena ha sido popular en aplicaciones donde se necesita polarización circular puesto que las antenas helicoidales proveen radiación circular polarizada. El elemento radiador es una hélice o cable en forma de hélice. Un ejemplo de este tipo de antena se muestra en la Figura 10.

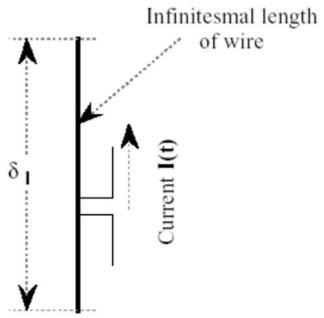
Figura 15: Antena helicoidal



(Traussnig, 2007)

3) Antena incremental: Este tipo de antena es de las más simples y comunes, es un conductor eléctrico conectado directamente al centro de una línea de alimentación de radiofrecuencia. Existen las antenas dipolo, las cuales son simétricas y son omnidireccionales puesto que transmiten la onda electromagnética en varias direcciones. Este tipo de antena consiste en un cable por el cual pasa una corriente alterna con una frecuencia, en la Figura 11 se muestra un ejemplo de este tipo de antena:

Figura 16: Antena dipolo



(Traussnig, 2007)

3. Módulo de potencia. Los satélites requieren de un suministro de potencia eléctrica sin interrupción y sin variaciones significativas en los niveles de voltaje y corriente. La cantidad de potencia requerida por cada uno en particular depende del modelo y de sus características de operación. (J.L. Tamasi, Mariana. 2003)

La fuente primaria de energía está constituida por un arreglo de celdas solares, las cuales funcionan bajo el principio del efecto fotovoltaico; cuanto mayor sea la densidad de flujo de radiación solar sobre ellas, mayor electricidad genera, aun a pesar de su pobre factor de eficiencia que ronda el 25%. Cada celda solar tiene un área de unos 5 a 32 cm^2, y uniendo muchas de ellas en serie y en paralelo se forma un módulo fotovoltaico. (J.L. Tamasi, Mariana. 2003)

La fuente secundaria está conformada por las baterías electroquímicas. Esta fuente tiene una función muy importante para los satélites, en especial aquellos destinados a las comunicaciones. Las baterías comienzan a funcionar cuando la fuente primaria, las celdas solares, deja de funcionar, esto es, cuando no tiene línea de vista con el Sol, lo que impide que los rayos solares sean captados por las celdas. Un ejemplo de esto son los eclipses, los cuales tienen una larga duración siendo hasta de 90 días por año. Para el caso de los satélites geoestacionarios, que generalmente son utilizados para comunicaciones, el no tener energía representaría no poder realizar sus funciones, lo cual impediría la comunicación por largos periodos de tiempo, y esto obviamente representaría un grave problema si no se contara con las baterías. (J.L. Tamasi, Mariana. 2003)

Estos dispositivos de almacenamiento de energía están conectados a un cargador que recibe energía cuando las celdas solares están trabajando, esta energía es suministrada a las baterías las cuales se cargan mientras las celdas solares estén funcionando. En el momento en que las celdas solares dejan de funcionar, el satélite tiene la función de conmutar la fuente de energía, pasando de la primaria a la secundaria, las pilas comienzan a trabajar y, durante el periodo que dure el eclipse, se descargan permitiendo que el satélite siga funcionando con normalidad. Una vez que el eclipse pasa, se vuelve a realizar un cambio de fuente de alimentación, lo que permite que las baterías se vuelvan a cargar gracias a la energía captada en las celdas solares. (J.L. Tamasi, Mariana. 2003)

Por último, tenemos al acondicionador de potencia, el cual está integrado por dispositivos como reguladores, convertidores y circuitos de protección, que permiten regular y distribuir la electricidad con los niveles adecuados a cada una de las partes del satélite. (J.L. Tamasi, Mariana. 2003)

a. Bases físicas de la conversión fotovoltaica. Las células solares están hechas de materiales semiconductores, que poseen electrones débilmente ligados ocupando una banda de energía denominada “banda de valencia”. Cuando se aplica un cuanto de energía por encima de un cierto valor a un electrón de valencia, el enlace se rompe y el electrón pasa a una nueva banda de energía llamada “banda de conducción”. Mediante un contacto selectivo, estos electrones pueden ser llevados a un circuito externo y realizar un trabajo útil, perdiendo así la energía captada y regresando por otro contacto a la banda de valencia con la energía inicial, anterior al proceso de absorción de un fotón luminoso. (J.L. Tamasi, Mariana. 2003)

El flujo de electrones en el circuito exterior se llama corriente de la célula y su producto por el voltaje con el que se liberan los electrones por los contactos selectivos determina la potencia generada. Todo esto ocurre a temperatura ambiente y sin partes móviles, pues las células solares, que convierten en electricidad sólo una parte de la energía de los fotones absorbidos se calientan sólo unos 25-30ºC por encima de la temperatura ambiente. (J.L. Tamasi, Mariana. 2003)

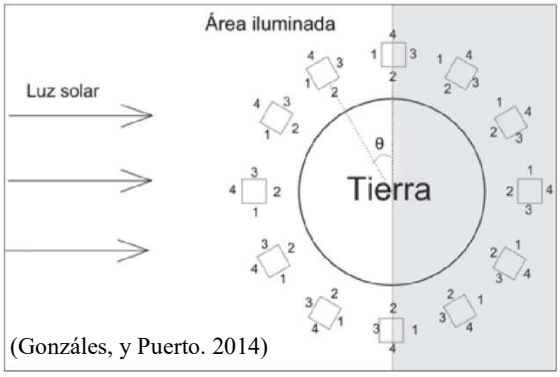
La estructura típica de una célula solar es una unión pn similar a los diodos semiconductores, pero no necesariamente la única posible. En las aplicaciones fotovoltaicas, las células solares se interconectan y encapsulan en elementos llamados módulos fotovoltaicos, que es el producto final vendido al usuario. (J.L. Tamasi, Mariana. 2003)

b. Producción de energía por medio de celdas fotovoltaicas. La generación de energía eléctrica del panel solar no permanecerá constante, sino que variará por distintos motivos. La necesidad de conocer *a priori* como será esta variación hace que se deban tener en cuenta todos los factores que influyen en este cambio para luego, mediante un modelo, poder predecir la generación del panel solar del satélite en órbita. A continuación, se expondrán los factores más relevantes que afectan a la generación eléctrica y se detallará cómo estos deben ser tomados en cuenta.

1) Eclipse. Durante la órbita en la cual se encuentre el satélite puede ser afectado por distintos objetos o cuerpos que puedan ser un obstáculo para la luz del Sol, teniendo un efecto conocido como eclipse en el cual los paneles solares no podrán producir energía eléctrica, de ser objetos como basura espacial no podrá ser predicho, pero sería un caso muy particular, dado que por las condiciones de distancias y velocidades entre los objetos que rodean al planeta Tierra, como la basura espacial, no podría generar una sombra la cual sea de tal magnitud que afecte al satélite, y que lo pueda llegar a hacer de forma periódica, llegando a ser un conflicto. Por lo que se ha de considerar únicamente a objetos de tal magnitud que sea imposible descartar la sombra que produzcan, es decir llegar al rango de cuerpos celestes, y por ser orbitas alrededor de la Tierra solo se puede considerar al planeta Tierra y a su satélite natural, la Luna.

2) Inclinación angular. La inclinación angular del panel solar respecto de los rayos incidentes del Sol cambiará considerablemente a lo largo del año y durante las maniobras. Estos cambios causarán un fuerte impacto en la energía eléctrica generada. Mientras mayor sea el ángulo de incidencia, menor será la producción eléctrica del panel. La variación del ángulo de incidencia se puede discriminar de acuerdo con dos procesos. El primero se debe a la alteración del ángulo de incidencia nominal del satélite a lo largo del año, debido a la naturaleza de la órbita. El segundo proceso que modifica la inclinación se debe a las maniobras que realiza el satélite para obtener imágenes de regiones terrestres. (Martínez. 2004)

Figura 17. Escenario de un CubeSat 1U con cara No. 2 hacia la Tierra.



(Gonzáles, y Puerto. 2014)

El voltaje de salida a circuito abierto y la corriente de corto circuito, para un panel solar montado en la cara No. 4, en la Figura 1, se pueden expresar de forma simplificada según la siguiente ecuación:

$$V_s = V_t \sin(\theta + \emptyset) + \Delta V \Delta T \ \ \{V\} \quad (1)$$

$$I_s = I_t \sin(\theta + \emptyset) + \Delta I \Delta T \ \ \{A\} \quad (2)$$

Este modelo simplificado se puede aplicar para el periodo de orbita en luz, dado que es el punto de referencia desde el cual se mide el ángulo de incidencia de la luz, en donde V_s e I_s, son el voltaje a circuito abierto y la corriente de corto circuito respectivamente, V_t e I_t, son los valores nominales teóricos para los mismos valores, ΔV y ΔI es la variación de voltaje a circuito abierto y de corriente de corto circuito en base al cambio de temperatura, θ es el ángulo de incidencia de la luz en el panel solar, $\emptyset$ es el ángulo causado por una maniobra del satélite, medido desde el plano paralelo a la cara No.2 con el plano perpendicular al eje de θ, y ΔT es la diferencia de temperatura con respecto al valor de temperatura establecida para los valores nominales teóricos. Los valores V_s, I_s, V_t, I_t, ΔV y ΔI generalmente serán dados por el fabricante del panel solar, o deben ser obtenidos de forma experimental. (Martínez. 2004)

c. Posibles paneles solares

1) Modelo propuesto en la Fase II.

Estos modelos fueron seleccionados en la Fase II del proyecto CubeSat de la Universidad del Valle de Guatemala. (Gonzalez. 2016)

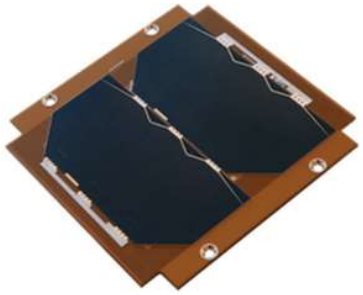
Cuadro 4. Modelos de paneles solares ensamblados, evaluados en la Fase II del proyecto.

Panel Solar	Voltaje (V)	Amperaje (mA)	Largo (mm)	Ancho (mm)	Grosor (mm)	Eficiencia (%)	Masa (Kg)	Tmax (°C)	Tmin (°C)
GaInP2/GaAs7Ge/ multiJunction	4.2	417	100	83	3	28.3	0.042	80	-40
NanoPower Solar P100-A/B	4.6	499	98	82.5	2.15	30	0.05	125	-40
ISIS CubeSat Solar Panels	3	760	98	82.5	2	28	0.05	125	-40

Cuadro 5. Trade Study de paneles solares ensamblados, evaluados en la Fase II del proyecto.

Peso de las variables		Valores normalizados		Trade Study					
				GalnP2		ISIS		NanoPower	
Variable	Peso (0-4)	1	10	Normalizado	Total	Normalizado	Total	Normalizado	Total
Eficiencia (%)	1	15	30	9	9	10	10	9	9
Voltaje (V)	4	2	8	6	24	6	24	4	16
Corriente (mA)	4	200	1000	5	20	5	20	8	32
Volumen (mm^3)	2	3.00E+04	1.50E+04	6	12	9	18	9	18
Masa (Kg)	1	0.06	0.025	6	6	5	5	5	5
Temperatura Max (°C)	4	100	200	4	16	7	28	7	28
Temperatura Min (°C)	2	-5	-60	7	14	7	14	7	14
Total					101		119		122

Figura 18. Panel Solar NanoPower.

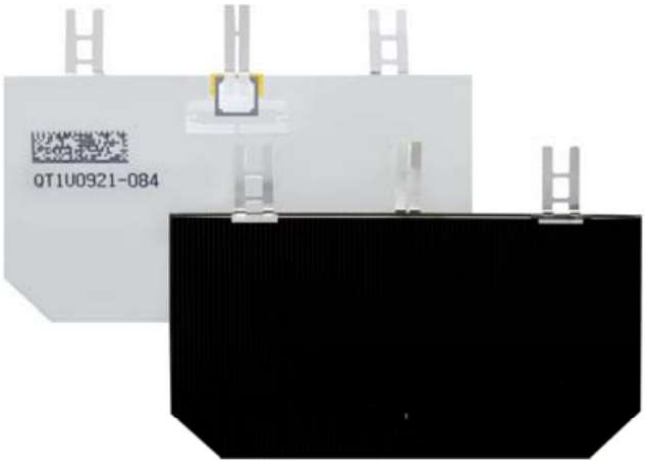


2) SpectroLab

Cuadro 6. Especificaciones técnicas de celdas solares fabricadas por SpectroLab.

Panel Solar	Voltaje (V)	Amperaje (mA)	Area (cm^2)	Eficiencia (%)	Masa (Kg)	Tmax (°C)	Tmin (°C)
NeXt Triple Junction (XTJ) Solar Cells	2.633	472.77	26.62	29.5	0.022	120	-40
Ultra Triple Junction (UTJ) Solar Cells	2.660	453.87	26.62	28.3	0.022	120	-40

Figura 19. Celdas de SpectroLab.



Es importante aclarar desde este punto que los modelos fabricados por SpectroLab no serán tomados en cuenta, ya que se contactó al proveedor y entre los requisitos de compra indica que el mínimo de compra es de $5,000.00 en producto, más $500.00 de manejo y envío, dando un total de gasto mínimo de $5,500.00, lo cual duplica e incluso supera el capital aproximado que se espera invertir en paneles solares, pues se espera una inversión cercana a $2,000.00. Además, el proveedor indicaba que el servicio de envío no se encuentra disponible para todos los países y que puede llegar a ser necesario cierto papeleo para poder ser enviados fuera de Estados Unidos. El proveedor no indicó el precio específico de cada modelo, por lo que se decidió dejar de considerar este proveedor como una opción para este proyecto.

3) Azur Space

Cuadro 7. Especificaciones técnicas de celdas solares fabricadas por Azur Space.

Panel solar	Voltaje (V)	Amperaje (mA)	Area (cm²)	Grosor (mm)	Eficiencia (%)	Masa (g)	Vidrio Protector	Tmax (°C)	Tmin (°C)	Precio (€)
TJ Solar Cell Assembly 3G30A	2.690	519.6	30.18	0.28	29.6	3.5	si	120	-40	252.00
TJ Solar Cell Assembly 3G28A	2.662	505.4	30.18	0.28	28.3	3.5	si	120	-40	247.00
TJ Solar Cell 3G30C-Advanced	2.700	1041.0	60.36	0.15	29.7	5.2	no	120	-40	475.00

Figura 20. Aspecto físico modelo 3G30A Y 3G28A.

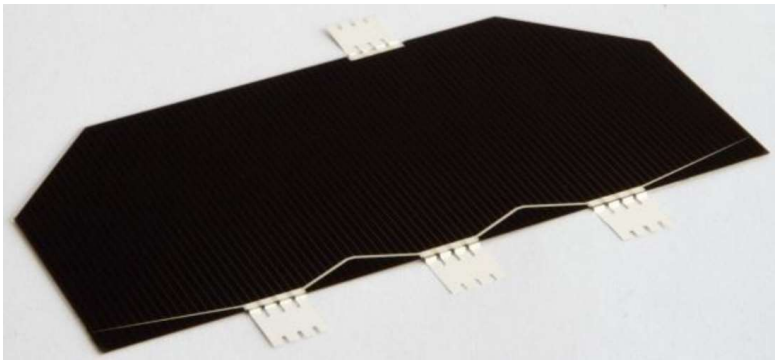


Figura 21. Aspecto físico modelo 3G30C.

En el caso de las celdas producidas por Azur Space, se descartará el modelo 3G30C, ya que no cuenta con vidrio protector, además que este modelo ocupa el doble de área a comparación de los otros dos, quitando la posibilidad de poder ser ensamblado según las necesidades del proyecto.

d. Baterías. La batería almacena energía para luego proveerla durante un eclipse y siempre que la potencia requerida por el equipamiento sobrepase a la tasa de generación del panel solar. La carga de la batería se realizará acumulando el excedente de energía generada por el panel cuando esté iluminado. Para que el sistema batería-panel solar sea adecuado y estable, deberán cumplirse necesariamente las siguientes condiciones: (Berbeglia, y Fernández, D. 2003)

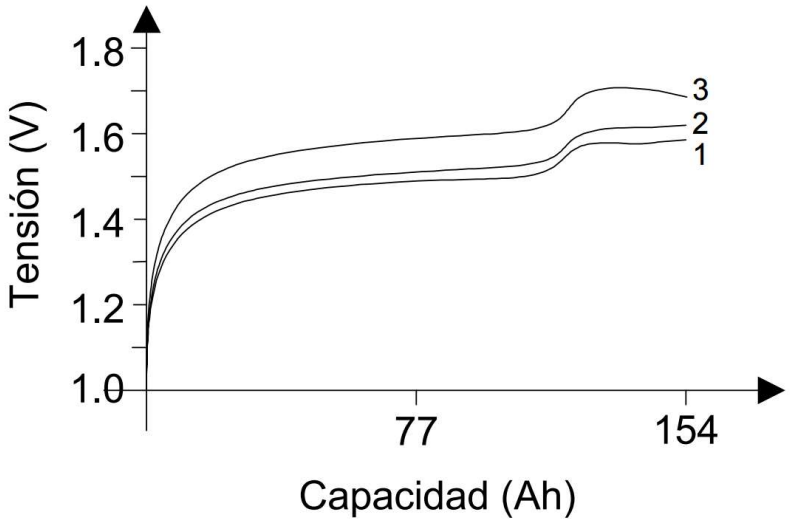
- ✓ La batería debe tener la capacidad de proveer la potencia máxima del satélite en la condición de máximo consumo.
- ✓ El panel debe entregar una potencia promedio superior a la potencia media de consumo del satélite. Además, debe proveer la tensión apropiada para que la batería se cargue.

Una batería está compuesta por una serie de celdas. Estas pueden ser de diferentes tipos, por ejemplo, níquel-hidrógeno, níquel-cadmio, etc.

La batería fija la tensión de trabajo del sistema. Esta tensión depende, entre otros factores, del estado de carga de la batería y de la corriente que recibe o entrega. En particular, la tensión varía considerablemente para estados de carga bajos y de plena carga, y cuando la batería pasa de carga a descarga o viceversa. (Berbeglia, y Fernández, D. 2003)

Para medir la capacidad de la batería se utiliza la unidad Ampere hora (Ah). Esta es una unidad que nos brinda información acerca de cuanta corriente es capaz de entregar la batería en un determinado tiempo. Bajo una corriente y una temperatura determinada, una batería puede caracterizarse con una curva que expresa cuál es la tensión dependiendo del estado de carga. Como se verá más adelante, esta curva resulta de gran utilidad para la simulación del sistema. Además, esta curva dependerá de la condición de carga o descarga de la batería. (Berbeglia, y Fernández, D. 2003)

Figura 22. Ejemplo de curvas de una batería de 154 Ah en estado de carga.

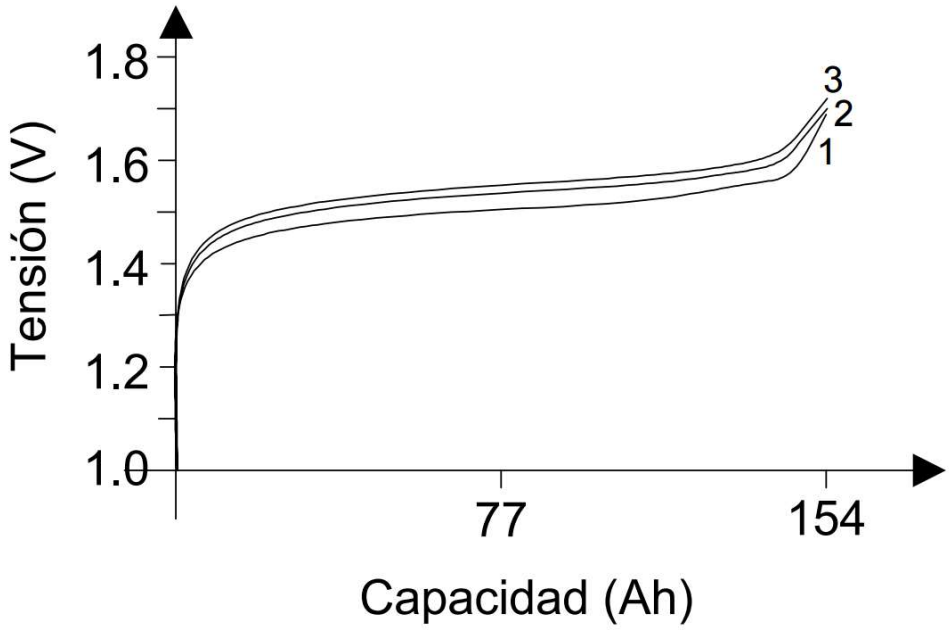


Las curvas 1, 2 y 3 corresponden a distintas corrientes de carga. (Berbeglia, y Fernández, D. 2003)

En la Figura 2 se muestran curvas tensión-carga cuando la batería se encuentra en carga. Las curvas 1, 2 y 3 corresponden a corrientes distintas en orden creciente. Se puede deducir que mientras menor sea la corriente, mayor es el rendimiento de la batería. Por otro lado, en la Figura 3 se muestran curvas tensión-

carga cuando la batería se encuentra en descarga. Las curvas 1, 2 y 3 corresponden a temperaturas distintas. Estas figuras muestran que, al igual que la curva I-V de un panel se modifica de acuerdo con el ángulo de incidencia, la curva de tensión-carga de la batería es afectada por el cambio en la temperatura y la corriente. Por esta razón, para conocer cómo es el comportamiento de la batería en cualquier momento debemos conocer no sólo la curva tensión-carga propia de la batería y su estado de carga, sino que además tenemos que considerar la corriente que aporta o recibe y cuál es su temperatura en ese momento. Esta curva tensión-carga será provista por el fabricante al igual que sus variaciones debido a cambios en la temperatura y corriente. (Berbeglia, y Fernández, D. 2003)

Figura 23. Ejemplo de curvas de batería en descarga.



Las curvas 1, 2 y 3 corresponden a distintas temperaturas. (Berbeglia, y Fernández, D. 2003)

El sentido de flujo de corriente en una batería determina si se está cargando o descargando. En general se asume que el sentido de la corriente es positivo cuando la batería brinda corriente al circuito, es decir, cuando se encuentra en estado de descarga. Si la corriente se produce en sentido contrario (corriente negativa), la batería se está cargando. (Berbeglia, y Fernández, D. 2003)

La vida útil de una batería puede medirse en la cantidad de ciclos que resiste, esto es, la cantidad de veces que puede pasar de carga a descarga o viceversa. Esto es fundamental para diseñar el subsistema dado que se debe garantizar el funcionamiento de la batería durante toda la vida útil del satélite. (Berbeglia, y Fernández, D. 2003)

En el Cuadro 8 se realiza una comparación entre las diferentes características que pueden poseer las baterías, para lo cual se obtuvieron datos de baterías Ni-MH, Li-Ion, Li-Po y $LiFePo_4$.

Cuadro 8. Comparación entre las características de diferentes tipos de baterías.

Tipo de batería	Ni-MH	Li-Ion	Li-Po	$LiFePo_4$
Densidad Energía-Peso (Wh/kg)	30-80	100-250	130-200	90-110
Densidad Energía-Volumen (Wh/dm^3)	140-300	250-360	300	220
Voltaje nominal (V)	1.2	3.6	3.7	3.3
Corriente de carga	Baja	Alta	Alta	Alta
Corriente de descarga	Baja	Alta	Alta	Alta
Rango de temperatura de carga (°C)	-10…45	0…45	0…45	-30…60
Rango de temperatura de descarga (°C)	-20…60	-20…60	-20…60	-30…60
Robustez	Media	Baja	Baja	Alta
Ciclos de carga	500 - 1000	400 - 1200	400 - 1200	2000 - 7000
Auto-descarga por mes (% del total)	20	25	10	2

e. Especificaciones técnicas batería tipo Li-Po (4000 mAh)

Información obtenida de (Homotix, S.a.S.)

Características

- ✓ Modelo: LP7035138
- ✓ Diseño: Batería de polímero de litio
- ✓ Largo: 138 ± 0.4 mm
- ✓ Ancho: 35 ± 0.4 mm
- ✓ Espesor: 7.0 ± 0.2 mm
- ✓ Cable: 150 ± 3.0 mm (26AWG UL 10 07)
- ✓ Peso: aprox. 75.0 g
- ✓ Conector: JST 2,54

Especificaciones eléctricas

- ✓ Rango de capacidad: 3980 mAh min, 4000 mAh tip.
- ✓ Voltaje Nominal: 3.7 V
- ✓ Rango de voltaje máximo de operación: 3.0 V a 4.2 V
- ✓ Voltaje de carga: 4.2 ± 50mV
- ✓ Corriente de carga inicial: carga estándar: 2000 mA, carga rápida: 4000 mA
- ✓ Desconexión de carga (A o B)
  A. Por tiempo: carga estándar: 7 horas, carga rápida: 4 horas
  B. Por corriente mínima: 80.0 mA
- ✓ Máxima corriente continua de descarga: 4000 mA
- ✓ Expectativa de ciclos de vida: >500 ciclos >70% de la capacidad inicial.
- ✓ Impedancia Interna (1kHz): aprox. 78 mOhm
- ✓ Protección de la celda
  Detección de sobrecarga: 4.325 ± 25 mV (0.5 a 2.0 seg. delay)
  Detección de carga baja: 3.00V ± 25mV (6.25 a 250 mseg. delay)

Condiciones ambientales

- ✓ Rango de temperatura
  Carga: 0 to +45 °C
  Descarga: -20 to +60 °C
- ✓ Capacidad de retención de almacenamiento:
  1 año entre -20 a 20 °C >70%, 3 meses entre -20 a 45 °C > 70%, 1 mes entre -20 a 60 °C > 70%

f. Especificaciones técnicas batería tipo Li-Po (2300 mAh)

Información obtenida de (Homotix, S.a.S.)

Características

- ✓ Modelo: LP654365
- ✓ Diseño: Batería de polímero de litio
- ✓ Largo: 65 ± 0.4 mm
- ✓ Ancho: 43 ± 0.4 mm
- ✓ Espesor: 6.5 ± 0.2 mm
- ✓ Cable: 150 ± 3.0 mm (26AWG UL 10 07)
- ✓ Peso: aprox. 40.0 g
- ✓ Conector: JST 2,54

Especificaciones eléctric

- ✓ Rango de capacidad: 2300 mAh min, 2300 mAh tip.
- ✓ Voltaje Nominal: 3.7 V
- ✓ Rango de voltaje máximo de operación: 3.0 V a 4.2 V
- ✓ Voltaje de carga: 4.2 ± 50mV
- ✓ Corriente de carga inicial: carga estándar: 1350 mA, carga rápida: 2300 mA
- ✓ Desconexión de carga (A o B)
  A. Por tiempo: carga estándar: 7 horas, carga rápida: 4 horas
  B. Por corriente mínima: 9.0 mA
- ✓ Máxima corriente continua de descarga: 2300 mA
- ✓ Expectativa de ciclos de vida: >500 ciclos >70% de la capacidad inicial.
- ✓ Impedancia Interna (1kHz): aprox. 78 mOhm
- ✓ Protección de la celda
  Detección de sobrecarga: 4.325 ± 25 mV (0.5 a 2.0 seg. delay)
  Detección de carga baja: 3.00V ± 25mV (6.25 a 250 mseg. delay)

Condiciones ambientales

- ✓ Rango de temperatura
  Carga: 0 to +45 °C
  Descarga: -20 to +60 °C
- ✓ Capacidad de retención de almacenamiento:
  1 año entre -20 a 20 °C >70%, 3 meses entre -20 a 45 °C > 70%, 1 mes entre -20 a 60 °C > 70%

g. Especificaciones técnicas batería tipo Li-Po (1500 mAh)

Información obtenida de (Homotix, S.a.S.)

Características

- ✓ Modelo: LP584070
- ✓ Diseño: Batería de polímero de litio
- ✓ Largo: 70 ± 0.4 mm
- ✓ Ancho: 40 ± 0.4 mm
- ✓ Espesor: 5.8 ± 0.2 mm
- ✓ Cable: 150 ± 3.0 mm (26AWG UL 10 07)
- ✓ Peso: aprox. 30.0 g
- ✓ Conector: JST 2,54

Especificaciones eléctricas

- ✓ Rango de capacidad: 1480 mAh min, 1500 mAh tip.
- ✓ Voltaje Nominal: 3.7 V
- ✓ Rango de voltaje máximo de operación: 3.0 V a 4.2 V
- ✓ Voltaje de carga: 4.2 ± 50mV
- ✓ Corriente de carga inicial: carga estándar: 750 mA, carga rápida: 1500 mA
- ✓ Desconexión de carga (A o B)
  A. Por tiempo: carga estándar: 7 horas, carga rápida: 4 horas
  B. Por corriente mínima: 9.0 mA
- ✓ Máxima corriente continua de descarga: 1500 mA
- ✓ Expectativa de ciclos de vida: >500 ciclos >70% de la capacidad inicial.
- ✓ Impedancia Interna (1kHz): aprox. 78 mOhm
- ✓ Protección de la celda
  Detección de sobrecarga: 4.325 ± 25 mV (0.5 a 2.0 seg. delay)
  Detección de carga baja: 3.00V ± 25mV (6.25 a 250 mseg. delay)

Condiciones ambientales

- ✓ Rango de temperatura
  Carga: 0 to +45 °C
  Descarga: -20 to +60 °C
- ✓ Capacidad de retención de almacenamiento:
  1 año entre -20 a 20 °C >70%, 3 meses entre -20 a 45 °C > 70%, 1 mes entre -20 a 60 °C > 70%

h. Especificaciones técnicas batería tipo Li-Po (250 mAh)

Información obtenida de (Homotix, S.a.S.)

Características

- ✓ Modelo: LP502030
- ✓ Diseño: Batería de polímero de litio
- ✓ Largo: 30 ± 0.4 mm
- ✓ Ancho: 35 ± 0.4 mm
- ✓ Espesor: 5.0 ± 0.2mm
- ✓ Cable: 150 ± 3.0 mm (26AWG UL 10 07)
- ✓ Peso: aprox. 3.5 g
- ✓ Conector: JST 2,54

Especificaciones eléctricas

- ✓ Rango de capacidad: 3980mAh min, 4000mAh tip.
- ✓ Voltaje Nominal: 3.7 V
- ✓ Rango de voltaje máximo de operación: 3.0 V a 4.2 V
- ✓ Voltaje de carga: 4.2 ± 50 mV
- ✓ Corriente de carga inicial: carga estándar: 125 mA, carga rápida: 250 mA
- ✓ Desconexión de carga (A o B)
  A. Por tiempo: carga estándar: 7 horas, carga rápida: 4 horas
  B. Por corriente mínima: 5.0 mA
- ✓ Máxima corriente continua de descarga: 250 mA
- ✓ Expectativa de ciclos de vida: >500 ciclos >70% de la capacidad inicial.
- ✓ Impedancia Interna (1kHz): aprox. 89 mOhm
- ✓ Protección de la celda

  Detección de sobrecarga: 4.325 ± 25 mV (0.5 to 2.0 sec. delay)

  Detección de carga baja: 3.00V ± 25mV (6.25 to 250 msec. delay)

Condiciones ambientales

- ✓ Rango de temperatura

  Carga: 0 to +45 °C

  Descarga: -20 to +60 °C
- ✓ Capacidad de retención de almacenamiento:

  1 año entre -20 a 20 °C >70%, 3 meses entre -20 a 45 °C > 70%, 1 mes entre -20 a 60 °C > 70%

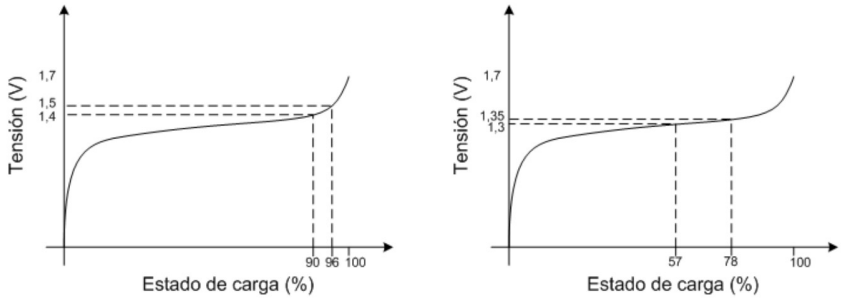
i. Regulador. El regulador es un componente electrónico cuyo objetivo es evitar la sobrecarga o descarga profunda de la batería. Para ello, determina a partir de la medición de la tensión el estado de carga de la batería y define la conexión o desconexión de módulos del panel solar. (Berbeglia, y Fernández, D. 2003)

Para medir el estado de carga, el regulador mide la tensión que la batería entrega. Esta tensión correspondiente a cada estado de carga depende de la temperatura y la corriente actual de la batería. Por esta razón no existe un único valor de desconexión, sino que habrá un valor asociado a cada curva de carga y descarga de la batería. Si la batería se está cargando y se atraviesa un determinado umbral, muy cercano al 100% de la carga, el regulador desconecta un módulo del panel y como consecuencia disminuirá la tasa de carga de la batería. Si luego de haber desconectado un módulo, la batería se comienza a descargar y cae debajo de un límite, el regulador conectará el módulo desconectado anteriormente. Para evitar oscilaciones, este último límite es menor que el valor de desconexión. (Berbeglia, y Fernández, D. 2003)

Por otro lado, si la tensión de la batería cae por debajo del mínimo aceptable, el satélite pasará a un estado de emergencia. En este estado, los requerimientos de potencia serán mínimos, por lo que el consumo eléctrico disminuirá considerablemente. El objetivo del estado de emergencia es evitar la descarga profunda, brindándole tiempo para recuperar el estado de carga adecuado. Para salir del estado de emergencia, la batería debe superar una tensión que garantice un estado de carga de la batería aceptable. Nuevamente, para evitar una oscilación, la tensión requerida para salir del estado de emergencia es mayor que la tensión para su entrada a dicho estado. (Berbeglia, y Fernández, D. 2003)

En la Figura 4 se muestra un ejemplo de una curva de batería con los valores de tensión del regulador. En este caso se observa que el regulador desconecta uno de los módulos del panel solar cuando la batería llega a un estado de carga del 96% (1,5V). Este módulo es reconectado cuando la carga de la batería cae por debajo de 90% (1,4V). (Berbeglia, y Fernández, D. 2003)

Figura 24. Puntos de desconexión y reconexión de módulos.



A la izquierda se muestran los puntos de desconexión y reconexión de módulos. A la derecha, los puntos de conexión de estado de emergencia y el punto de regreso al estado normal. (Berbeglia, y Fernández, D. 2003)

Por otro lado, en la segunda figura, se observa que el estado de emergencia se activa cuando la batería disminuye más allá del 57% (1,3V). El estado normal de funcionamiento se recupera cuando la batería vuelve a tener el 78% de la carga (1,35V). (Berbeglia, y Fernández, D. 2003)

j. Consumo de energía en misiones anteriores

1) Misión SwissCube-1 (Misión exitosa)

Cuadro 9. Consumo esperado de cada módulo en la misión SwissCube-1.

Modulo	Consumo (mW)
EPS	50
Payload	450
CDMS	150
Antena	300
Control ADCS	10
Sensor ADCS	50
Magnetorque ADCS	160
Contro y receptor RF principal	80
Transmisor RF principal	3000

2) Misión CanX-1 (Misión fallida)

Cuadro 10. Consumo esperado de cada módulo en la misión CanX-1.

Modulo	Consumo (mW)
OBC	380
Receptor	150
Transmisor	1110
Magnetorque	1000
Magnetometro	230
Sensor de Imagen (Camara)	300

C. ÓRBITA

1. Tipos de orbitas

a. LEO (Low Earth Orbit).

También llamada órbita baja terrestre, es el tipo de órbita de altura más baja, entre 200 y 2000 kilómetros de altura. En esta órbita es donde se encuentran la mayoría de los satélites, siendo la excepción los satélites de comunicaciones que requieren órbitas geoestacionarias. Un ejemplo de satélite que usa este tipo de órbita es el ENVISAT. (García-Hidalgo. 2013)

b. MEO (Medium Earth Orbit).

Son todas aquellas órbitas que se encuentran entre los 2000 kilómetros y los 35726 kilómetros (entre las órbitas LEO y Geoestacionaria respectivamente). Los satélites en esta franja se usan comúnmente como satélites de navegación y comunicaciones, o como satélites de experimentación. Como ejemplos de satélites que se encuentran en órbitas MEO están el sistema de satélites GPS (Global Positioning System), Glonass (Sistema de navegación por satélite diseñado por Rusia) o Galileo (Sistema de navegación por satélite diseñado por UE). (García-Hidalgo. 2013)

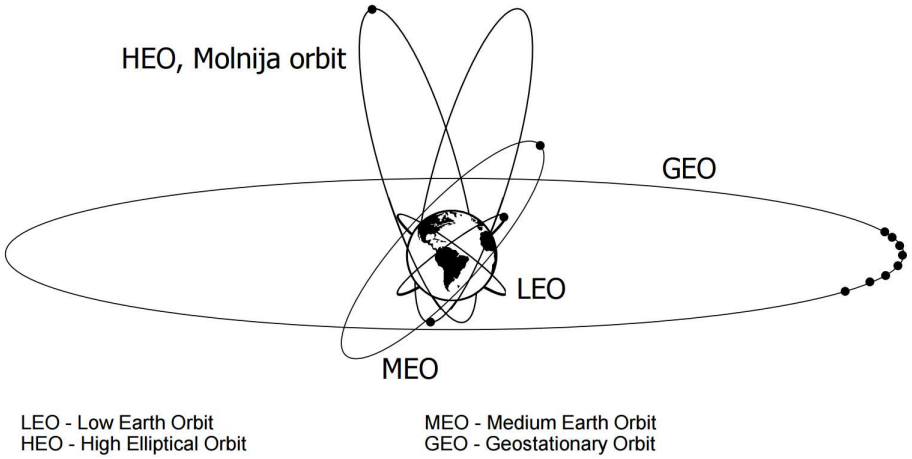
c. GEO (Geoestacionary Orbit).

La órbita geoestacionaria es una órbita cuya característica principal es que tiene el mismo periodo orbital que la Tierra, además tiene una altura de 35726 kilómetros sobre el ecuador. Es la órbita en la que se encuentran la mayoría de los satélites de comunicaciones, debido a que no requieren que la estaciones tracen su posición, que siempre será la misma con respecto a un observador en tierra. Un ejemplo de satélite que se encuentra en esta órbita es el satélite europeo Meteosat. (García-Hidalgo. 2013)

d. HEO (Molnija Orbit).

Es un tipo de órbita especial caracterizada por ser muy elíptica, con un perigeo de 1000 kilómetros y apogeo de 39400 kilómetros, tiene una inclinación de 63.4º y su periodo orbital es de aproximadamente 12 horas. La principal característica es que el satélite en esta órbita pasa gran cantidad de tiempo en el apogeo (fenómeno llamado “pozo de apogeo”), debido a esto es interesante su uso en los polos ya que permite una gran cobertura de los mismos e incluso una cobertura completa con tres satélites. Los principales inconvenientes son que la distancia satélite-tierra cambia continuamente y se necesitan dos estaciones de radio en tierra. (García-Hidalgo. 2013)

Figura 25. Tipos de órbitas.



2. Parámetros de una órbita LEO. Dado que para la finalidad de la misión del CubeSat de la Universidad del Valle de Guatemala se determinó utilizar una órbita tipo LEO, se profundizará en los parámetros que sirven para definir a las órbitas orientadas hacia los detalles específicos de órbitas tipo LEO. Los parámetros para definir una órbita son los siguientes:

a. Semieje mayor (a). Podemos expresar el semieje mayor en términos de la distancia desde el centro de la Tierra hasta el apogeo (Rapogeo) y el perigeo (Rperigeo). El semieje mayor (a) puede obtenerse aplicando: (García-Hidalgo. 2013)

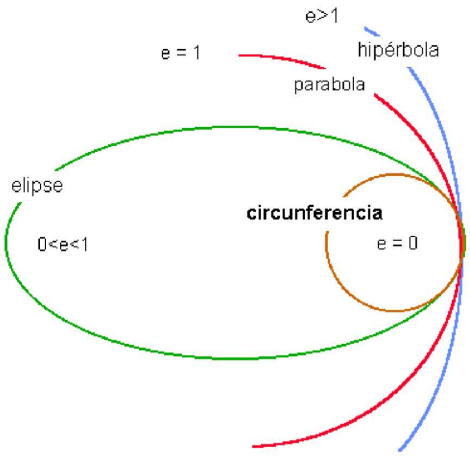
Figura 26. Semieje mayor de una órbita.

$$a = \frac{R_{apogeo} + R_{perigeo}}{2} \quad (3)$$

b. Excentricidad (e). Se define la excentricidad de la órbita como la diferencia entre la circunferencia y la cónica de la órbita. En la práctica no es posible conseguir una órbita perfectamente circular o parabólica. (García-Hidalgo. 2013)

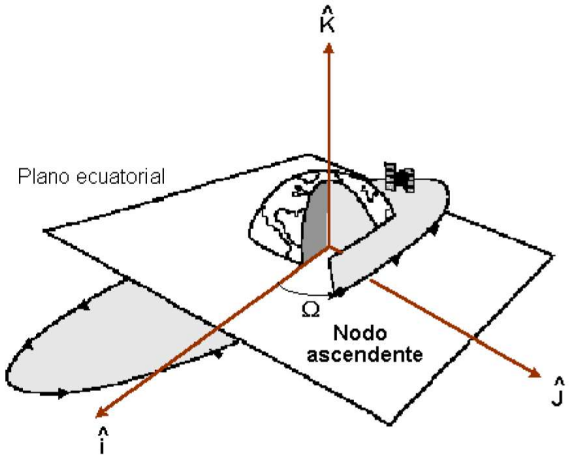
- ✓ Una circunferencia perfecta tiene una excentricidad de 0.
- ✓ Una órbita elíptica tiene una excentricidad inferior a la unidad.
- ✓ Una órbita parabólica tiene una excentricidad igual a 1.
- ✓ Una órbita hiperbólica tiene una excentricidad superior a la unidad.

Figura 27. Excentricidad de la órbita.



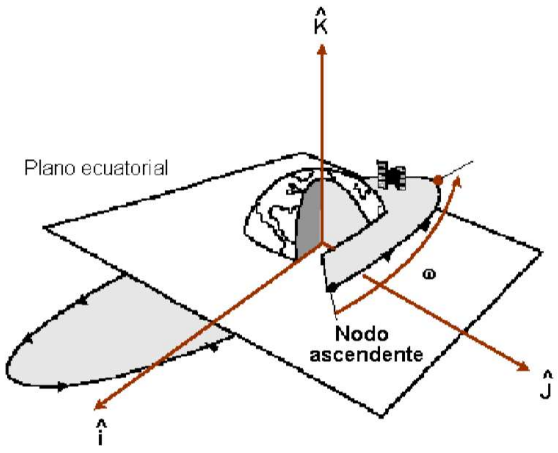
c. Nodo ascendente (Ω). Para medir lo inclinado que está una órbita, se define el nodo ascendente como el punto en el que el satélite cruza el plano ecuatorial en dirección sur-norte. Este punto está referenciado a la dirección I, que apunta al equinocio vernal (Figura 7). El ángulo entre la dirección I y el nodo ascendente se conoce como la ascensión recta del nodo ascendente, RAAN. (García-Hidalgo. 2013)

Figura 28. Nodo ascendente.



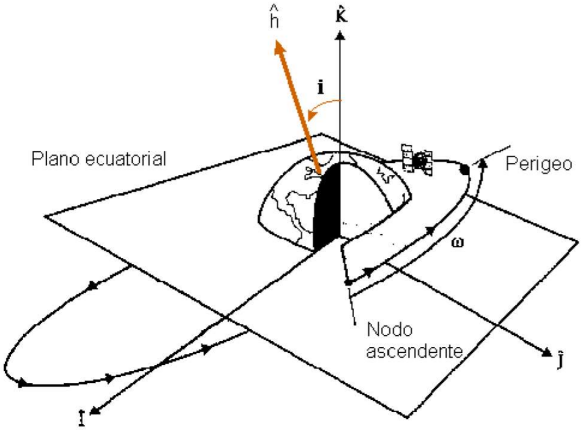
d. Argumento del perigeo (ϖ). La orientación de la órbita queda descrita localizando el perigeo con respecto al nodo ascendente. Este ángulo, se conoce como el argumento del perigeo (Figura 8) y se mide positivamente en el sentido de movimiento del satélite. (García-Hidalgo. 2013)

Figura 29. Argumento del perigeo.

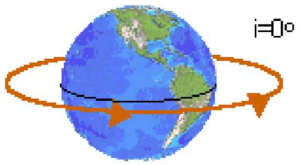
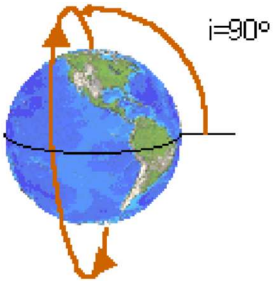
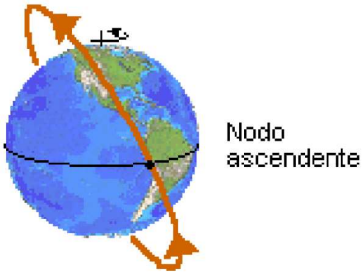


e. La inclinación (i). Indica cuánto está inclinada una órbita. Una órbita que está justo en el plano del Ecuador tiene una inclinación de 0 grados y se conoce como órbita ecuatorial. Una órbita que pasa justo por los polos Norte y Sur debe tener una inclinación de 90 grados y se llama órbita polar. En la Figura 9, se muestra el ángulo entre el vector unidad, $\hat{k}$ (que coincide con el eje de rotación de la Tierra), y otro vector unidad, $\hat{h}$, que es perpendicular al plano de la órbita. (García-Hidalgo. 2013)

Figura 30. Inclinación de la órbita.

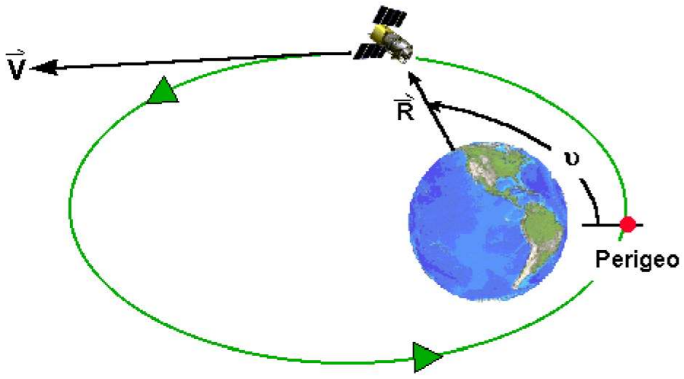


Cuadro 11. Tipos de órbitas según su inclinación.

Inclinación	Tipo de órbita	Diagrama
i = 0°, i = 180°	Ecuatorial	
i = 90°	Polar	
0° < i < 90°	Directa	
90° < i < 180°	Indirecta o retrógrada	

f. Anomalía verdadera (υ). Describe la posición instantánea del satélite con respecto al perigeo. Es un ángulo que se mide positivamente en la dirección del movimiento, entre el perigeo y la posición del satélite. De los seis elementos orbitales, la anomalía verdadera es el único que cambia continuamente (ignorando perturbaciones). (García-Hidalgo. 2013)

Figura 31. Anomalía verdadera.



g. Velocidad de satelización (v_c). Es la velocidad requerida por el satélite para mantenerse en la órbita circular descrita. (García-Hidalgo. 2013)

$$v_c = \sqrt{\frac{\mu}{a}} \tag{4}$$

Siendo μ el parámetro de gravitación, cuyo valor para la Tierra es de $3.986 \cdot 10^{14}$ m³ /s².

h. Periodo (T). El periodo orbital es el tiempo que tarda el satélite en realizar una vuelta completa alrededor de la órbita. Su cálculo se obtiene a partir de la velocidad lineal. (García-Hidalgo. 2013)

$$T = \frac{2 \cdot \pi \cdot a}{v_c} \tag{5}$$

3. Parámetros de las de orbitas analizadas

Cuadro 12. Parámetros kepplerianos de las órbitas evaluadas.

Parámetro	ISS	Spaceflight						Gauss Dnepr
		S500-27D	S575-98D	S500-60D	S515-98D	S450-20D	S500-98D	G600-97D
Semieje mayor (km)	6783.5	6871	6946	6871	6886	6821	6871	6974
Excentricidad	0.0006722	0	0	0	0	0	0	0.008
Inclinación	51.6421°	27.5°	98°	60°	98°	20°	98°	97.62°
Argumento del Perigeo	60.1442°	0°	35°	30°	35°	0°	35°	0°
RAAN	29.0926°	185°	88°	112°	88°	185°	88°	120°
Anomalía Media	225°	85°	110°	110°	110°	85°	110°	302°

4. STK (Satellite Tool Kit).

Satellite Tool Kit, conocido por sus iniciales STK, es el software de Analytical Graphics, Inc. (AGI) que permite a ingenieros y científicos diseñar y desarrollar simulaciones complejas y dinámicas de problemas reales en satélites artificiales. Al principio se utilizaba en la comunidad aeroespacial para solucionar problemas sobre satélites que orbitaban alrededor de la Tierra y ahora se usa tanto en comunidades de defensa como aeroespaciales. (Benzal, y Velasco. 2010)

STK se creó en 1989 por los fundadores de Analytical Graphics, Inc. como herramienta alternativa para software de proyectos aeroespaciales específicos. Fue adoptado primero por la comunidad aeroespacial para el análisis de órbitas y cálculos de acceso (es el momento en que el satélite puede ver la estación terrestre). Con el tiempo se fue expandiendo con más módulos que incluían la posibilidad de simular cálculos de sistemas de comunicaciones, radares, misiones interplanetarias y evasión de colisiones de órbitas. Más adelante, se añadió al programa la posibilidad de trabajar en 3D, que permitía a los usuarios la visualización en tiempo real de contingentes militares de aire, de tierra y marítimos, así como el componente espacial. (Benzal, y Velasco. 2010)

V. METODOLOGÍA

A. MÓDULO DE COMPUTADORA A BORDO

Se investigaron satélites de características similares al que se está realizando. Esta investigación se realizó como punto de partida, con la cual se pudo empezar a trabajar para poder cumplir los objetivos propuestos.

Se investigaron y seleccionó un microprocesador adecuando para ser usado como OBC del CubeSat. Se realizó un diseño de integración para los módulos de Payload, comunicación y potencia que se utilizan actualmente en el CubeSat, con el fin de poder tener una guía de cómo empezar a trabajar en el módulo y tener una idea de cómo realizar la integración de este subsistema dentro del satélite.

Al haber terminado el diseño de integración se creó un programa capaz de probar la funcionalidad y las limitaciones del subsistema OBC. Este programa tenía como fin el poder convertir tres imágenes en arreglos de bits para ser enviados al módulo de comunicaciones, por medio del protocolo de comunicación UART en un tiempo menor a 4 minutos. Se trató de alcanzar estos resultados realizando simulaciones de conexión con los demás subsistemas. Entre estas simulaciones se encuentra la conexión del OBC con una PC. Estos se conectaron para simular el envío de información al sistema de comunicaciones, se hizo uso de imágenes guardadas en una memoria convertidas a arreglos de bits, las cuales simularon imágenes tomadas por el módulo payload. Realizando estas simulaciones se midió el tiempo de respuesta que se obtuvo al enviar las imágenes del OBC a la computadora, para así tener el conocimiento si era necesario realizar algún cambio o si es este microprocesador era apto para cumplir la misión a realizar por el CubeSat. Posterior a esta prueba se creó una rutina dentro del procesador para poder empezar el programa de conversión de imágenes y de envío de datos desde que se inicia el procesador.

Para poder realizar estas prueba fueron necesarios un microprocesador y una computadora para poder realizar las pruebas de conversión y envío de datos, un convertidor UART a USB para poder conectar a la computadora con el OBC, un monitor para poder programar dentro del sistema operativo que incluía el microprocesador, imágenes de extensión ".raw" que simularon las imágenes tomadas por el módulo de Payload para ser convertidos a arreglos de bits. Cabe recalcar que las imágenes utilizadas, para realizar las pruebas, fueron similares a las que han sido tomadas por el módulo Payload.

Para poder realizar la selección del microprocesador fue necesaria la utilización de un sistema de elección llamado *Trade Study*. Este es un sistema utilizado por la NASA para la elección de los componentes a utilizar. Los *Trade Studies* consisten en la ponderación de que tan crítica es cada característica de cierto componente. Esto se realiza evaluando que tan necesaria es cada característica para el funcionamiento del satélite, se le coloca una ponderación más alta si la característica es muy esencial en el funcionamiento, y se le pondera con un valor bajo si no es tan esencial. Al tener las ponderaciones se

verifican cuáles son los valores más altos y bajos de cada una de las variables a evaluar. Esto se hace con el fin de hacer una normalización de diez valores entre el valor más bajo y el más alto, siendo 1 el valor que se califique como menos funcional para el satélite y diez como el mejor valor que podría llegar a tener esta característica para el componente que irá dentro del satélite. Luego se evalúa cuál de los 10 valores se acerca más a el valor que tiene cada una de las características de cada componente. Por último, para cada característica de un componente, se multiplica la ponderación dada por el número entre 1 y 10 elegido al evaluar cada una de estas, luego se suman todos estos valores de cada característica de un componente evaluado para luego comparar todos los componentes y elegir el que tenga el valor más alto de esta sumatoria.

B. MÓDULO DE COMUNICACIÓN

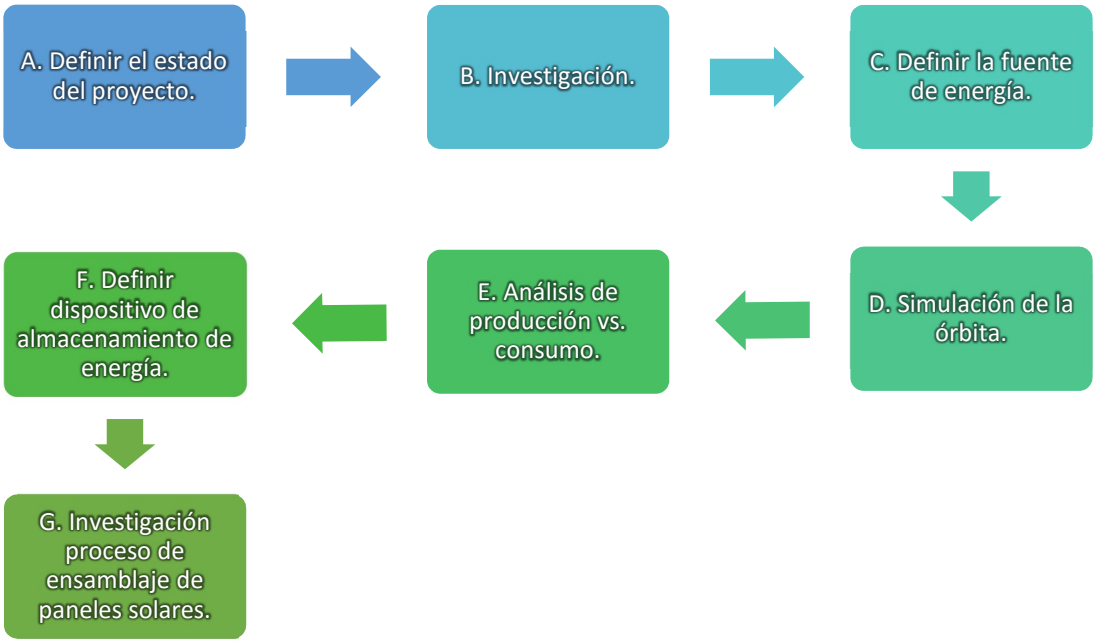
La comunicación de un satélite mediante radio frecuencia requiere de ciertos componentes para su funcionamiento, explicados anteriormente en el marco teórico. En el proceso de la comunicación, la antena juega un papel importante ya que transmite y/o recibe los datos de una onda de radio frecuencia. Luego de enviar o recibir la información mediante la antena, esta debe de ser transformada a bytes utilizando un *transceiver* y luego descodificada puesto que viene en un protocolo de comunicación en forma de paquetes. Esta información luego va a un ordenador o al *On Board Computer* del satélite, siendo estas últimas dos tareas ejecutadas por el *Terminal Node Controller*.

Para la realización del módulo de comunicaciones, se debe empezar por definir la frecuencia a utilizar en la comunicación. Es importante iniciar por la frecuencia puesto que depende de la aplicación qué frecuencia utilizar, y es de la cual dependen los componentes electrónicos que se deben implementar en el módulo. Para la frecuencia seleccionada se debe de utilizar los componentes correspondientes a esa frecuencia, siendo estos componentes la antena y el transceiver. Para la señal a enviar y la señal a recibir se debe de implementar un protocolo de comunicación apto para la aplicación. Luego definir qué tipo de antena utilizar y que tipo de componentes, se procede al desarrollo del software como tal. Sabiendo que el *Terminal Node Controller* será la interfaz entre el *transceiver* y la computadora de abordo, se debe desarrollar un software implementado en un componente el cual obtenga los bytes de la computadora de abordo, aplique un protocolo de comunicación y luego los envíe al *transceiver* después de que éste fuese configurado correctamente mediante el *Terminal Node Controller.* De manera inversa se debe obtener la onda de radio de la antena mediante el *transceiver* y obtener los datos mediante el *Terminal Node Controller.* Luego de que se desempaquete la información del protocolo de comunicación entonces se envían los datos a la computadora de abordo para que ella decida, con base a la información obtenida, como debe ejecutar las tareas necesarias.

Se seleccionará e implementará tanto el software y hardware necesario para poder enviar y recibir información tanto desde el satélite como de la base en Tierra. Se realizarán pruebas de funcionamiento,

integración y comunicación para comprobar y validar la correcta ejecución del módulo de comunicaciones del satélite.

C. MÓDULO DE POTENCIA



1. Definir el estado del proyecto. Se debe aclarar que esta es la tercera fase de desarrollo del módulo de potencia del proyecto CubeSat de la Universidad del Valle de Guatemala, por lo que existen resultados previos. Pero el desarrollo de los resultados obtenidos anteriormente no ha llegado al punto de poder seleccionar elementos que se puedan instalar dentro del satélite, integrarse con los otros módulos de este proyecto y poder operar en condiciones espaciales.

El resultado más cercano a este alcance, ha sido un análisis de paneles solares ya ensamblados para elegir un modelo fiable que pudiera ser instalado en el satélite, pero este punto debe ser analizado de nuevo, ya que el costo de los paneles solares ensamblados es elevado, por lo que se repetirá este análisis agregando más opciones considerando ensamblar los paneles en los laboratorios de la Universidad del Valle de Guatemala.

Por lo tanto, teniendo este punto de partida, se espera llegar a definir un sistema que sea capaz de funcionar en condiciones espaciales, con los requerimientos que lo hagan capaz de funcionar dentro del

CubeSat que se está desarrollando, por lo que se deben considerar las limitaciones que se tienen como el consumo de cada uno de los nódulos, los tiempos de luz y sombra en la órbita elegida, llegando a utilizar la menor cantidad de paneles solares para así reducir el costo al mínimo posible.

2. Investigación. Esta etapa fue necesaria para poder reforzar las decisiones que se dieron para guiar al desarrollo de este proyecto, aunque fue más que una etapa, ya que se debió realizar en repetidas ocasiones dado que era necesario reforzar la teoría y conocimientos en los siguientes puntos:

- ✓ Proveedores de celdas fotovoltaicas
- ✓ Caracterización de paneles solares
- ✓ Órbitas espaciales
- ✓ Simulación de órbitas espaciales
- ✓ Caracterización de baterías de tipo Ion de Litio y de Polímero de Litio

3. Definir la fuente de energía. En esta etapa, se debió contactar a proveedores de celdas fotovoltaicas para conocer las especificaciones técnicas y precio de sus productos, en específico se contactó a SpectroLab y a Azur Space, esto debido a que son proveedores confiables de este tipo de productos y fueron seleccionados por el asesor de este proyecto, Ph.D. Luis Zea, como proveedores fiables para el proyecto.

Luego de conocer la información proporcionada por los proveedores, se realizó una comparación entre las opciones por medio del método "Trade Study", dado que este es el método establecido para este proyecto para la toma de decisiones entre distintas posibilidades que puedan ser comparadas entre sí.

En esta etapa se debió considerar la selección realizada en la Fase II de este proyecto, pero dado que en la Fase II se realizó en base a paneles solares ensamblados, y en esta fase se consideró comprar las celdas fotovoltaicas para ensamblarlas en un panel solar dentro de los laboratorios de la Universidad del Valle de Guatemala, se debió hacer una comparación en base a las características teóricas que tendría los paneles solares ya ensamblados con las celdas fotovoltaicas individuales.

4. Simulación de la órbita. El primer paso en esta etapa fue realizar una investigación sobre los parámetros necesarios para definir una órbita, y que representa cada uno de ellos. De igual forma se debió investigar como simular una órbita con software de forma que se pudiera obtener la mayor cantidad de información sobre la órbita y las condiciones a las que estaría sometido el satélite, entre las opciones encontradas estaban las siguientes:

- ✓ SPENVIS
- ✓ Matlab
- ✓ STK (Satellite Tool Kit)

De estas la opción elegida fue STK, dado que este programa es el más sencillo de configurar entre los anteriores, es más intuitivo dado que muestra la órbita simulada en 3D y en 2D, y puede generar una gran variedad de reportes de información relevante como, por ejemplo:

- ✓ Ángulos de sobrevuelo
- ✓ Efecto de eclipse sobre el satélite (por parte de la Tierra y la Luna)
- ✓ Temperatura experimentada por el satélite a lo largo de la órbita (En la versión Pro)
- ✓ Altura del satélite en cada instante
- ✓ Umbra y penumbra del satélite y el área de estudio (Tiempos de luz del satélite)
- ✓ Vectores de velocidad y posición del satélite, a lo largo de la órbita y sobre volando el área de estudio

Para configurar una simulación de la órbita, el primer paso era conocer los parámetros keplerianos de la órbita, los parámetros necesarios para esto son los siguientes:

- ✓ Semieje mayor
- ✓ Excentricidad
- ✓ Inclinación
- ✓ Argumento del Perigeo
- ✓ Nodo ascendente
- ✓ Anomalía media

Con estos datos se crea un “escenario” añadiendo un satélite eligiendo la opción “Define Properties” en el simulador de STK, dando como resultado lo siguiente:

Figura 32. Órbita simulada en STK, grafico 2D.

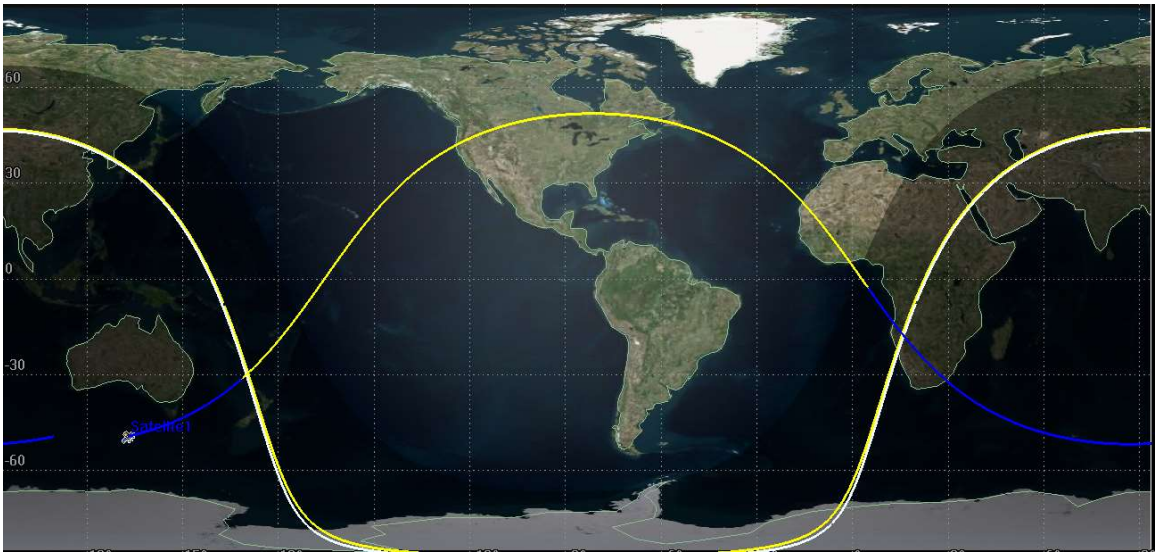
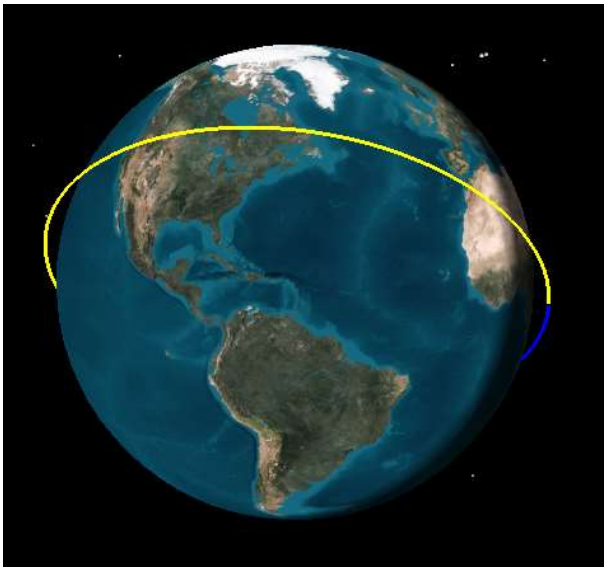


Figura 33. Órbita simulada en STK, grafico 3D.

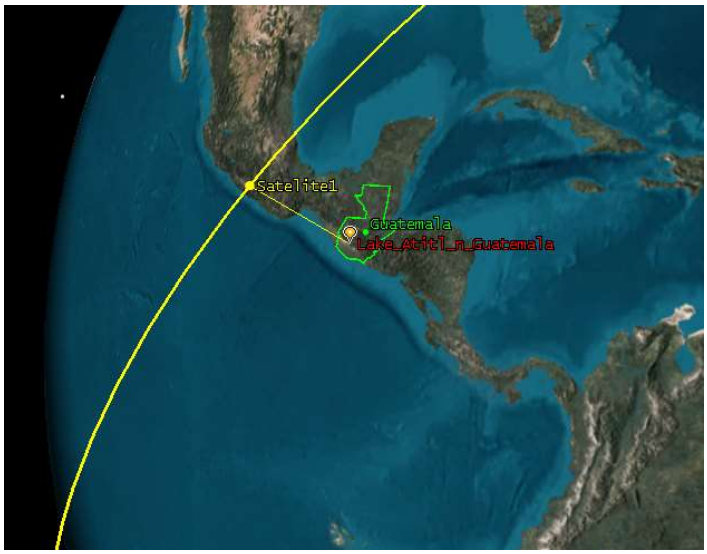


Con esto definido, lo siguiente es determinar el área de estudio, para poder hacer análisis de tiempo, posición y velocidad, para ello se agregó como área de estudio, a Guatemala, y como lugar, se estableció el Lago Atitlán.

Figura 34. Área de estudio definida para la simulación.

Con una órbita y área de estudio establecida, con ayuda del simulador se puede hacer una relación, con la opción “Access…”, para poder conocer cada vez que la órbita pase por una zona en la cual puede ser captado desde el área de estudio. Como se puede ver en la siguiente imagen, con esto el simulador calcula cuando es posible tener contacto desde el área de estudio con el satélite, también se puede hacer un análisis de en qué hora y fecha puede ocurrir este sobrevuelo del área de estudio.

Figura 35. Satélite sobrevolando el área de estudio, gráfico en 3D.



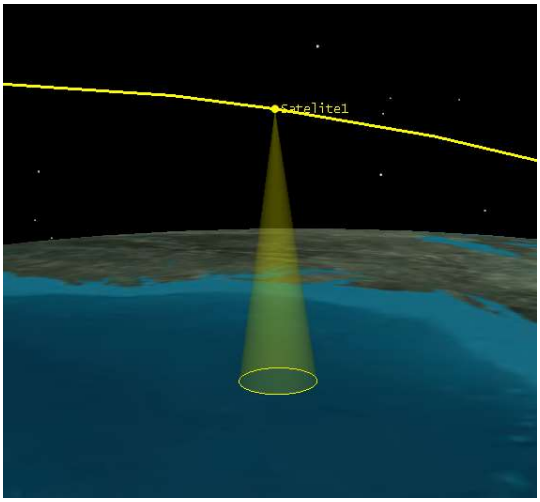
Para poder corroborar que esta órbita es la opción correcta para cumplir con los objetivos de la misión, se tomó en cuenta el objetivo principal de la misión, la cual consiste en tomar imágenes del Lago de Atitlán, por lo que el siguiente paso consistía en agregar los parámetros del sensor que tomara la imagen a la simulación, el único parámetro necesario para representar dicho sensor es el siguiente:

- ✓ Semiángulo de cono de visión: 6.31°

Este fue obtenido por el módulo de Payload. (Muñoz. 2016)

Como resultado, en la simulación es visible un cono desde el punto que representa al satélite, orientado perpendicularmente a la superficie del planeta, este representa el rango de visión del sensor desde la órbita establecida, como se puede apreciar en la siguiente imagen.

Figura 36. Cono de visión del sensor, gráfico en 3D.

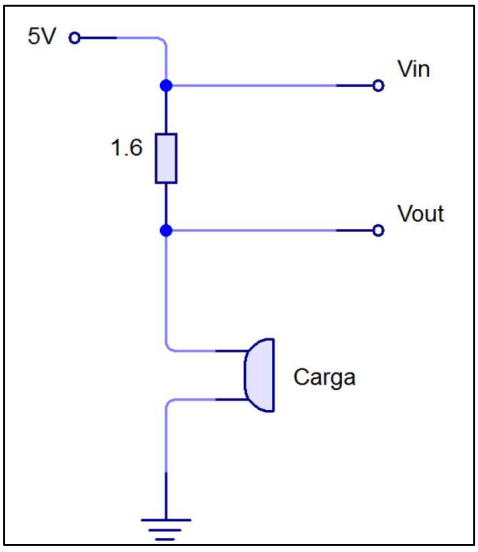


Con ayuda de las herramientas del programa utilizado, STK, es posible conocer los sobrevuelos desde los cuales el sensor puede tomar una imagen del lago.

5. Análisis de producción vs. consumo. Ya que en este punto ya se conocía la potencia generada por la fuente de energía, y los tiempos de luz y sombra de la órbita, solo faltaba conocer el consumo de cada uno de los módulos que se encontrara instalado dentro del satélite y funcionara con base en la energía generada por este módulo. Lamentablemente para el punto de desarrollo de esta fase del proyecto, solo se encuentran definidos los componentes que se utilizaran en los módulos de Payload y OBC, por lo que para establecer valores de consumo para los módulos de ADCS y Comunicaciones, se buscaron valores teóricos representativos, en el caso de Comunicaciones, se buscó información de misiones anteriores para conocer el consumo esperado que se planteó para este módulo, pero en el caso de ADCS, se tomó la información de las misiones anteriores únicamente como referencia, dado que en el desarrollo de la Fase IV de este proyecto, ya se ha delimitado a una posible opción factible para ser instalada en este satélite, por lo que se utilizó el valor de consumo de este como referencia, pues su consumo es aún mayor al establecido en misiones anteriores.

Para conocer el consumo real de los módulos Payload y OBC se utilizó el circuito que se muestra en la siguiente imagen.

Figura 37. Circuito utilizado para medición de consumo.



En este circuito, la resistencia de 1.6 Ω es una resistencia de alta potencia, y la forma de determinar el consumo de la carga en el circuito, es midiendo la caída de voltaje en dicha resistencia, pues si se conoce su valor de impedancia, se puede calcular la corriente que pasa en ella, siendo esta la misma que está utilizando la carga, y dado que el voltaje en la salida de la resistencia es el utilizado por la carga, es posible obtener la potencia consumida por la carga.

Después de ya haber establecido los valores de consumo de cada uno de los módulos, se debía establecer los tiempos y etapas de consumo, por lo que se definieron tres escenarios en los cuales se pudiera simplificar los tipos de sobrevuelos que se puedan dar en el transcurso de la misión, estos son los siguientes:

a) Sobrevuelos sin una tarea específica, estos son los sobrevuelos los cuales no crucen sobre el área de estudio, y no sea posible tomar una foto o establecer comunicación entre el satélite y la estación en tierra.
b) Sobrevuelos dentro del rango de comunicación, estos son los sobrevuelos que crucen dentro del área en la cual es posible establecer comunicación entre el satélite y la estación en tierra.
c) Sobrevuelos sobre el Lago de Atitlán, estos son los sobrevuelos que cruzan sobre el área de estudio y es posible tomar una fotografía.

Estableciendo estos escenarios, se determinó los periodos de funcionamiento que tendría cada uno de los módulos, y con esto se calculó la potencia de producción y consumo a lo largo de la órbita en cada uno de los tipos de escenarios, con lo que se pudo determinar la producción vs el consumo de energía, lo que llevo a obtener la energía que se debía almacenar para poder hacer funcionar el satélite en los periodos de sombra, en los cuales los paneles solares no pueden transformar energía.

Este proceso se repitió analizando la producción de energía que se tendría utilizando un sistema con cinco paneles solares, hasta la producción que se tendría utilizando dos paneles solares, sin cambiar el consumo de ninguno de los módulos, esto con el fin de establecer el menor número de paneles solares con el cual el satélite puede funcionar de forma correcta.

6. Definir el dispositivo de almacenamiento de energía. La capacidad de la batería que es la mejor opción para este proyecto, se definió en base a cuanta energía se consume a lo largo de la etapa de sombra de la órbita, y mientras la potencia de los paneles solares no es capaz de igualar y superar al valor de la potencia de consumo. Esto se calculó haciendo la sumatoria del consumo de los módulos que se encuentran en funcionamiento durante estos periodos, en concreto son los módulos OBC y ADCS.

7. Ensamblaje de paneles solares. Este proceso es totalmente ilustrativo, dado que no se pudo llegar a esta etapa de desarrollo debido a que cuestiones económicas no fue posible contar con las celdas fotovoltaicas a tiempo para poder ensamblar los paneles solares en esta fase del proyecto.

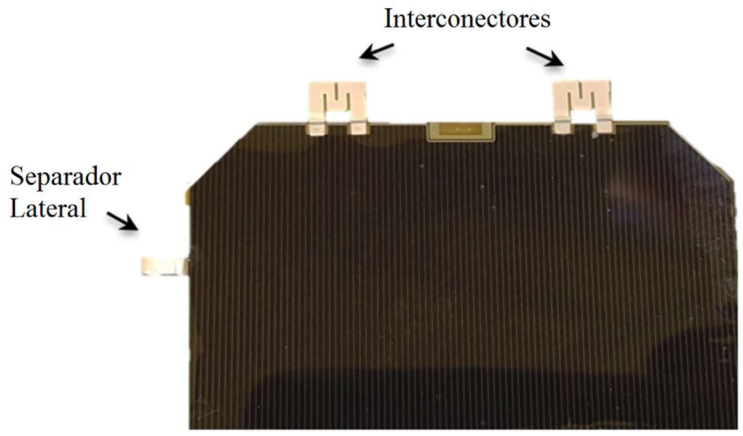
Es necesario contar con los siguientes elementos:

- ✓ Base calefactora
- ✓ Solador tipo lápiz para componentes de superficie

- ✓ Pistola de calor
- ✓ Cinta adhesiva de doble cara (ej. NuSil CV4-1161-5 o LINQTAPE PIT2SD)
- ✓ Cinta de poliamida
- ✓ Pasta de soldadura (punto de fundición > 180 °C)
- ✓ Epoxy Technology - EPO-TEK® H20E
- ✓ PCB para altas temperaturas (ej. ISOLA IS410 o Shengyi S1000-2)

Para ser unidas eléctricamente, las celdas solares deben ser separadas por los interconectores que sirven para conectar las celdas al circuito de potencia eléctrica. Actualmente, las celdas solares de estándar espacial pueden ser compradas con este tipo de conexión ya instalada, estas son conocidas como celdas "CID'd", además existe la opción de ser compradas con los este tipo de conexión y el vidrio de protección instalado, estas son conocidas como celdas "CIC's".

Figura 38. Ejemplo de una celda CID'd.



a. Soldadura de conexiones eléctricas. El proceso de soldadura es igual al de soldar componentes electrónicos de superficie, con la diferencia de que se deben utilizar materiales que resistan altas temperaturas, preferiblemente de estándar para uso espacial. Por la finalidad de las conexiones se debe procurar nunca tocar ninguna de las superficies directamente con la mano, dado que esto puede generar deterioro de los componentes y la soldadura después de un tiempo, también se debe asegurar que todas las superficies se encuentren totalmente limpias. El proceso consiste en lo siguiente:

a. Se debe pre-calentar la superficie de la PCB en la sección donde se hará la soldadura, esto se hace hasta llevarlo a una temperatura cercana a 150 °C. Se debe elevar la temperatura de forma lenta por medio de la base de calefacción, y la pistola de calor de ser necesario.

b. Al tener una temperatura relativamente estable alrededor de 150 °C, se deben unir las superficies que serán unidas.
c. Se debe aplicar la pasta para soldadura sobre el punto de unión deseado, procurando no utilizar más de lo necesario.
d. Luego con ayuda del soldador, se aplica aire caliente sobre la pasta en la zona que se desea soldar, hasta ver una unión consistente. Si se desea se pueden colocar masas pequeñas sobre la celda, con el cuidado de que estas no sean lo suficientemente pesadas para dañar la celda.
e. Se deja enfriar la zona de forma lenta.
f. Se limpia la zona con ayuda de un hisopo y limpia contactos.

Figura 39. Pre-calentamiento de terminales de una celda.

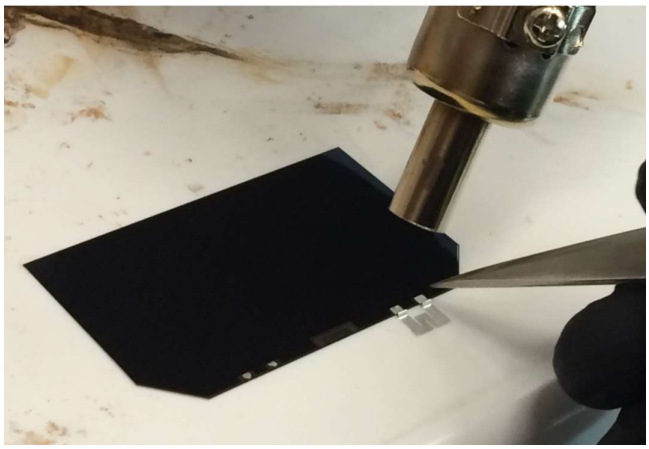


Figura 40. Unión entre celdas.

b. Unión de las celdas solares a la PCB. Se deben utilizar adhesivos los cuales sean resistentes a altas temperaturas, preferiblemente de estándar para uso espacial, procurando evitar dejar burbujas de aire en las uniones, dado que esto es un riesgo para la misión, pues al ser expuestas al vació las burbujas tenderán a escapar, lo que podría llegar a ocasionar daños en las celdas. El proceso consta de lo siguiente:

1) Preparar la superficie donde se pretende adherir la celda solar, pasando una lija suave para crear una superficie porosa y luego limpiándola.
2) Si es posible y necesario, remover los componentes en la parte trasera de la celda.

Figura 41. Superficie de una celda ya preparada.

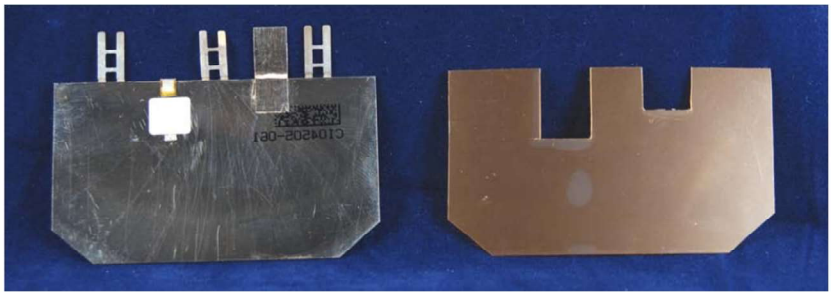
En la izquierda se muestra una celda nueva, a la derecha una celda con el diodo removido.

3) Pegar una pequeña lámina de metal a la parte trasera, con EPO-TEK® H20E, esta servirá como contacto.

Figura 42. Celda con el contacto de lámina adherido.

4) Dejar reposar la unión durante 3 horas a 80 °C.
5) Cortar y aplicar la cinta de poliamida en la superficie donde se sujetará la celda, cubriendo toda el área que ocupará la celda.
6) Cortar la lámina de cinta adhesiva de doble cara (NuSil CV4-1161-5) al tamaño de la celda o un poco menos.

Figura 43. NuSil CV4-1161-5 cortado.



A la derecha se muestra la cinta adhesiva de doble cara, cortada según el contorno de la celda.

7) Remover el primer protector de la cinta adhesiva y colocar en la parte trasera de la celda.
8) Remover el segundo protector de la cinta adhesiva, y pegar la celda a la superficie deseada.

Figura 44. Celda solar adherida a un CubeSat.

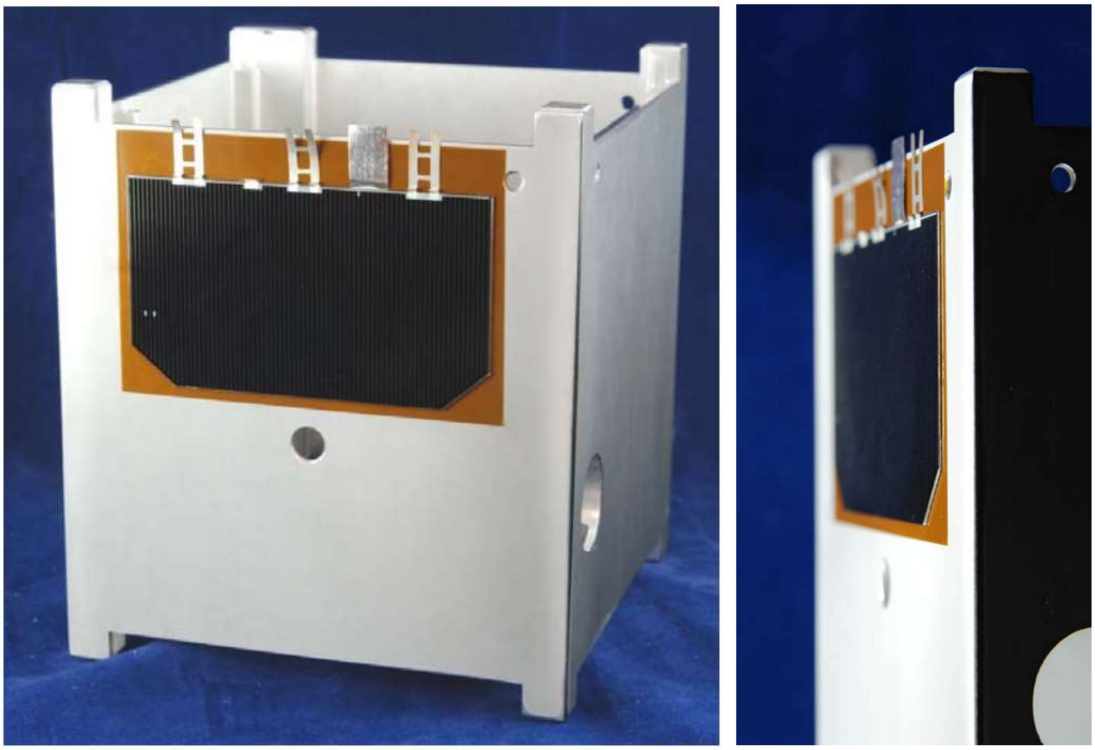


Figura 45. Aplicación de adhesivos a la celda.

8. Referencias. Este proceso fue resumido con base en los siguientes documentos:

[a] CapLinq. 2014. *PIT2SD Series, 2-mil Double-Sided Polyimide Tape with Single Side Release Liner*. http://www.caplinq.com/2-mil-polyimide-kapton-tape-silicone-adhesive-double-sided-pit2sd-series.html

[b] Dahir, Andrew. 2015. *Edited Solar Cell Mounting Procedure*. University of Colorado-Boulder.

[c] EPOXY TECHNOLOGY, Inc. 2014. *EPO-TEK® H20E Technical Data Shet*. https://www.tedpella.com/technote_html/16014_H20E_TN.pdf

[d] Isola. 2015. *IS410 Epoxy Laminate and Prepreg Data Sheet*. http://www.isola-group.com/wp-content/uploads/2015/12/IS410-Lead%C2%ADfree-Epoxy-Laminate-and-Prepreg-Data-Sheet.pdf

[e] Karuza, Petras; y D. Hinkley. 2015. *Solar Cell Installation Using Double Sided Polysiloxane Pressure Sensitive Adhesive (PSA) Polyimide Film*. Cal Poly. The Aerospace Corporation Mechanics Research Department.

[f] Mason, James; y B. Mero. 2015. *MinXSS Solar Cell Mounting FAI*. Laboratory for Atmospheric and Space Physics.

[g] NuSil. 2014. *CV4-1161-5 Controlled Volatility Pressure Sensitive Adhesive Tape*. https://nusil.com/services/downloadfile.ashx?productcode=CV4-1161-5&originalname=CV4-1161-5P.pdf

[h] Sandberg, Ariel. 2016. *Streamlining Cubesat Solar Panel Fabrication Processes*. Michigan Exploration Laboratory. University of Michigan. USA. 12 págs.

[i] Shengui. 2015. *S1000-2 FR-4 Data Sheet*. https://www.multi-circuit-boards.eu/fileadmin/pdf/leiterplatten_material/shengyi_s1000-2__www.multi-circuit-boards.eu.pdf

VI. RESULTADOS Y DISCUSIÓN

A. COMPUTADORA A BORDO

Antes de realizar la selección del microprocesador, se investigaron varios procesadores que podían ser utilizados como OBC. Para cada uno de los microcontroladores investigados se buscaron datos técnicos con los cuales podían ser comparados con otros microprocesadores y así poder realizar una mejor selección para ser usado como OBC.

Cuadro 13. Ponderación de las variables esenciales de un procesador para el CubeSat.

Variable	Ponderación	1	2	3	4	5	6	7	8	9	10
Masa (g)	3	77.00	70.20	63.40	56.60	49.80	43.00	36.20	29.40	22.60	9.00
Volumen (mm^3)	3	117325.00	106567.50	95810.00	85052.50	74295.00	63537.50	52780.00	42022.50	31265.00	9750.00
Potencia (W)	4	9.00	8.14	7.27	6.41	5.54	4.68	3.82	2.95	2.09	0.36
Velocidad de reloj (GHz)	2	0.02	0.12	0.22	0.32	0.41	0.51	0.61	0.71	0.80	1.00
RAM (MB)	2	0.015	102.41	204.81	307.21	409.61	512.01	614.41	716.80	819.20	1024.00
Temperatura mínima (ºC)	3	0.00	-4.00	-8.00	-12.00	-16.00	-20.00	-24.00	-28.00	-32.00	-40.00
Temperatura máxima (ºC)	3	0.00	8.50	17.00	25.50	34.00	42.50	51.00	59.50	68.00	85.00
Diseñando para el espacio	1	NO									YES

Los cuadros 13 y 14 son el resultado del uso del método *Trade Study* para la selección del microprocesador a utilizar como OBC del CubeSat. Para este método se realizó una ponderación de todas las variables, como se observar en el Cuadro 13, junto con la normalización de diez valores de cada característica esencial para el CubeSat, colocando los valores menos funcionales en el uno y los más funcionales para el satélite en el diez. Luego se evaluó cada componente que se estaba comparando para ser utilizado como OBC.

Cuadro 14. Selección del microprocesador, haciendo uso del método *Trade Study*, para ser usado como OBC del CubeSat

Variable	Raspberry Pi A+		Raspberry Pi Zero		TI MSP430F6779		TI MSP430F6745		AeroFlex UT699 LEON3FT		Pumpkin Motherboard	
	Normalizado	Total	Normalizado	Total	Normalizado	Total	Normalizado	Total	Normalizado	Total	Normalizado	Total
Masa (g)	5	15	10	30	7	21	7	21	7	21	1	3
Volumen en placa (mm^3)	4	12	10	30	1	3	1	3	1	3	2	6
Potencia en placa (W)	9	36	10	40	3	12	3	12	1	4	3	12
Velocidad de reloj (GHz)	8	16	10	20	3	6	3	6	2	4	1	2
RAM (MB)	3	6	6	12	1	2	1	2	10	20	1	2
Temperatura mínima (ºC)	10	30	10	30	10	30	10	30	10	30	10	30
Temperatura máxima (ºC)	10	30	10	30	10	30	10	30	10	30	10	30
Diseñando para el espacio	1	1	1	1	1	1	1	1	10	10	10	10
Total		146		193		105		105		122		95

En el Cuadro 14 se observan varios datos en columnas con el nombre “Normalizado”, estos se obtuvieron al comparar cada característica de todos los componentes con los diez valores normalizados en el Cuadro 13, ej. si la Raspberry Pi Zero tiene una masa de 9 gramos, se observa que valor normalizado del Cuadro 13 que se acerca a esos 9 gramos está en la columna 10, por lo que en la columna “Normalizado” de la Raspberry Pi Zero, a la característica de la masa, se le coloca diez, y así se realiza con todas las características. Se evalúa cada característica de todos los componentes, y luego se multiplica por la ponderación dada en el Cuadro 13, dando como resultado los valores que se observan en las columnas llamadas “Total” del Cuadro 14. Por último, se hace la sumatoria de todos los valores de las columnas “Total” de cada componente, colocando el resultado en la fila llamada “Total”.

Después de haber realizado una investigación acerca de microprocesadores para ser usados como OBC, y al hacer uso del método Trade Study, se logró seleccionar el microprocesador que se encuentra en la Raspberry Pi Zero, el cual será el que se evaluará para ser usado como computadora a bordo dentro del satélite. Esta selección de procesador facilita la programación y la conexión con otros sistemas, esto es debido a que este utiliza un sistema operativo bastante conocido y fácil de manejar.

Para poder afianzar nuestra selección del procesador, se debía determinar las funcionalidades y limitaciones del mismo, y así poder comprobar si sería capaz de poder ser parte del satélite como OBC. Las pruebas a realizar constaron de la conversión a arreglos de bits de imágenes guardadas en la memoria de la Raspberry Pi que simulaban imágenes tomadas por el módulo payload. Luego de haber sido convertidas

todas las imágenes, se realizaron pruebas de envío y recepción. Esto se hizo usando comunicación serial por medio de la interfaz UART. Para comprobar el envío y recepción de datos se conectó la Raspberry Pi Zero con una PC, se midieron los tiempos que tardaba la Raspberry Pi en enviar los datos, y los tiempos que tardaba la computadora en recibir los mismos. Los siguientes cuadros muestran los resultados obtenidos al realizar estas pruebas

Cuadro 15. Tiempo promedio de conversión, envío y recepción para 1000 repeticiones de una imagen de 500KB, para 600 repeticiones de una imagen 1MB, y para 200 repeticiones de una imagen de 3MB.

Imagen [MB]	Tiempo de conversión [s]	Tiempo de envío [min]	Tiempo para recepción [min]
0.5	2.27	0.73	0.74
1	5.61	1.51	1.51
3	15.10	4.53	4.54

En el Cuadro 15 se puede observar los promedios de 1000, 600 y 200 repeticiones, obteniendo los tiempos de conversión, envío, y recepción de datos para una imagen de 500KB, 1MB y una imagen de 3MB respectivamente. Esto resultados fueron obtenidos utilizando los programas que se encuentran en la parte de anexos, a los cuales se les fue cambiando la imagen a ser utilizada para el envío, y la cantidad de repeticiones. Para poder obtener el promedio de tiempo para 1000 repeticiones de las imágenes de 500KB y 1MB, y el promedio de tiempo para 400 repeticiones de la imagen de 3MB, se estuvieron realizando los envíos por 1.5 días, con lo cual se pudieron obtener los resultados que se muestran en el Cuadro 15.

Cómo se puede observar en los resultados del Cuadro 15 los promedios de tiempo al hacer los envíos y la recepción de datos son elevados, esto es debido a la tasa utilizada en la comunicación serial. Aunque la tasa máxima de envío de datos de la Raspberry Pi Zero pueda llegar a ser hasta 1600000 baudios, para realizar la comunicación entre los sistemas se utilizó una tasa de envío de 115200 baudios, esto se debió a que esta era la tasa máxima de recepción de datos para el microcontrolador elegido en COMMS. El utilizar esta tasa de envío y recepción pudo causar que la comunicación de OBC hacía COMMS tarde más de los esperado, lo cual tendría como consecuencia que se pierda mucho tiempo en enviar las imágenes tomadas por el módulo payload. Este problema podría afectar al en el envío de datos desde COMMS a la estación en tierra.

Tomando en cuenta lo anterior, en el Cuadro 16 se observan los resultados de pruebas de envío de datos de tres imágenes, con esto se trató de evidenciar qué tamaño de imágenes se podrían utilizar para ser enviados en menos de 4 minutos.

Cuadro 16. Tiempo de conversión, envío y recepción de datos para tres imágenes de 500KB, 3 imágenes de 1MB y 3 imágenes de 3MB, y la potencia promedio consumida en cada una de estas.

Imagen [MB]	Tiempo de conversión [s]	Tiempo de envío [min]	Tiempo para recepción [min]	Potencia [W]
0.5	7.93	2.21	2.22	0.604
1	13.77	4.54	4.54	0.597
3	47.91	13.61	13.61	0.607

Como se observa en el Cuadro 16, se realizaron pruebas en las que se obtuvo el tiempo promedio de 20 repeticiones para la conversión a arreglos de bits, el tiempo de envío y recepción de datos para tres imágenes de 500KB, tres imágenes de 1MB y tres imágenes de 3MB, y también se realizó medición de potencia para cada una de las repeticiones.

En el Cuadro 17, se presentan los resultados que se obtuvieron en las pruebas de envío y recepción de datos para las nueve imágenes, se puede observar que para el envío de tres imágenes en menos de 4 minutos es ideal usar imágenes que tengan un tamaño mayor 500KB y que sean un poco menor a 1MB. Al observar los promedios de tiempo para el envío de tres imágenes de 3MB, se logra observar que el envío de datos llega a tardar hasta 13.6 minutos aproximadamente, lo cual supera por mucho el tiempo propuesto para el envío de datos.

Aunque el tiempo de envío para tres imágenes de 500MB es menor a 4 minuto, indicando que existe la posibilidad de enviar tres o más imágenes de 500KB, es preferible tener menos imágenes con mejor resolución, que tener más imágenes que no se pueden procesar.

Al observar las potencias consumidas por cada una de los promedios, se puede notar que este consume menos de 0.7W de potencia, esto indica que el sistema de potencia debe suministrar al menos esta cantidad para que OBC pueda operar correctamente.

Cuadro 17. Tiempos de conversión, envío y recepción para imágenes de diferentes tamaños.

Tamaño [MB]	Tiempo de conversión [s]	Tiempo de envío [min]	Tiempo de recepción [min]
0.033	0.14	0.04	0.05
0.067	0.27	0.09	0.10
0.10	0.40	0.14	0.15
0.20	0.92	0.29	0.30
0.50	2.14	0.73	0.74
1	4.43	1.51	1.51
3	12.46	4.53	4.54
5	20.22	7.57	7.58
7	34.89	10.60	10.61
10	62.29	14.55	14.56

Figura 46. Gráfica de tiempos de conversión para imágenes en un rango entre 0.033 y 10MB.

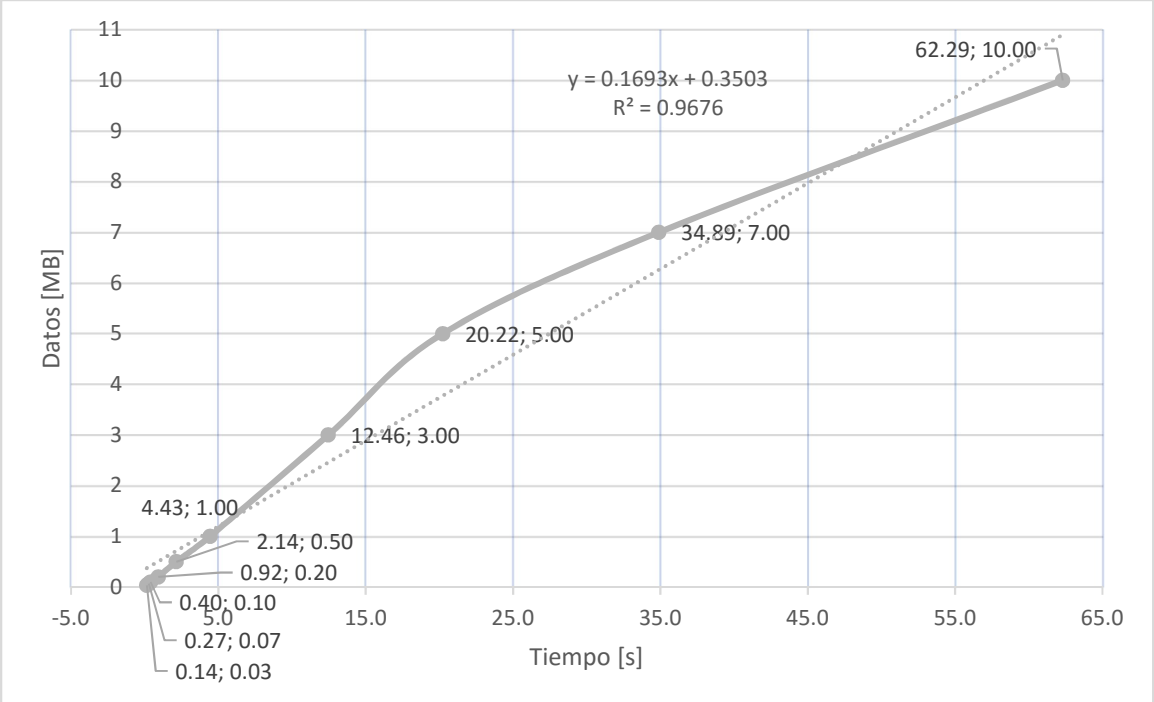


Figura 47. Gráfica de tiempos de envío para imágenes en un rango entre 0.033 y 10MB.

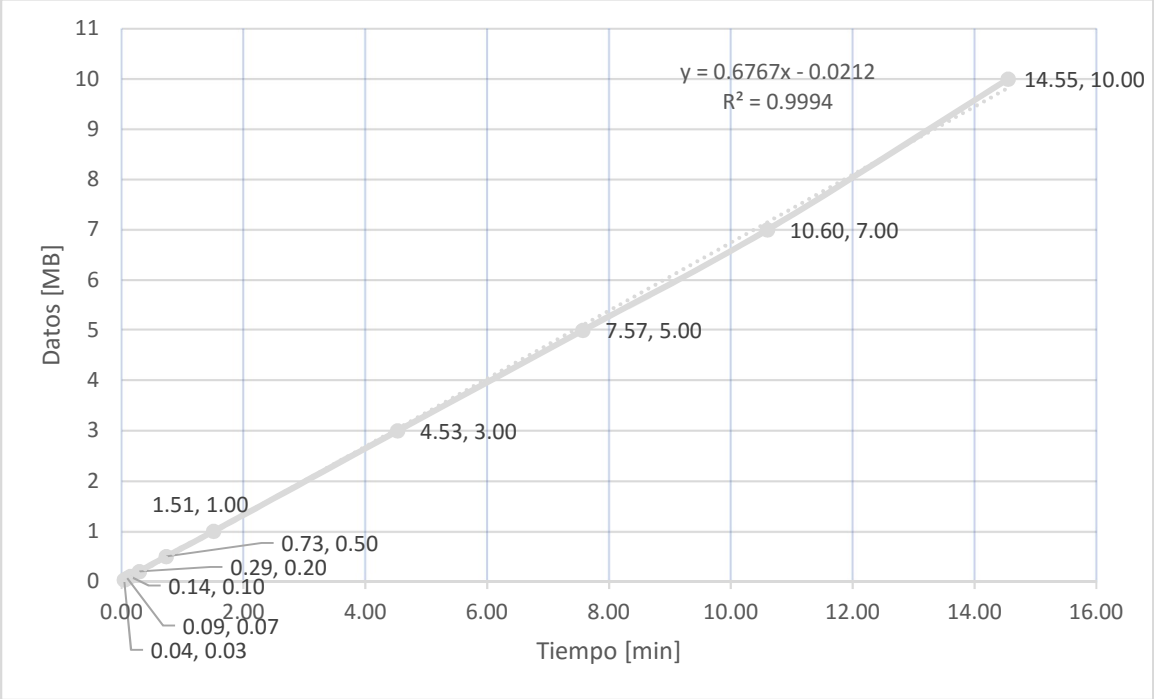
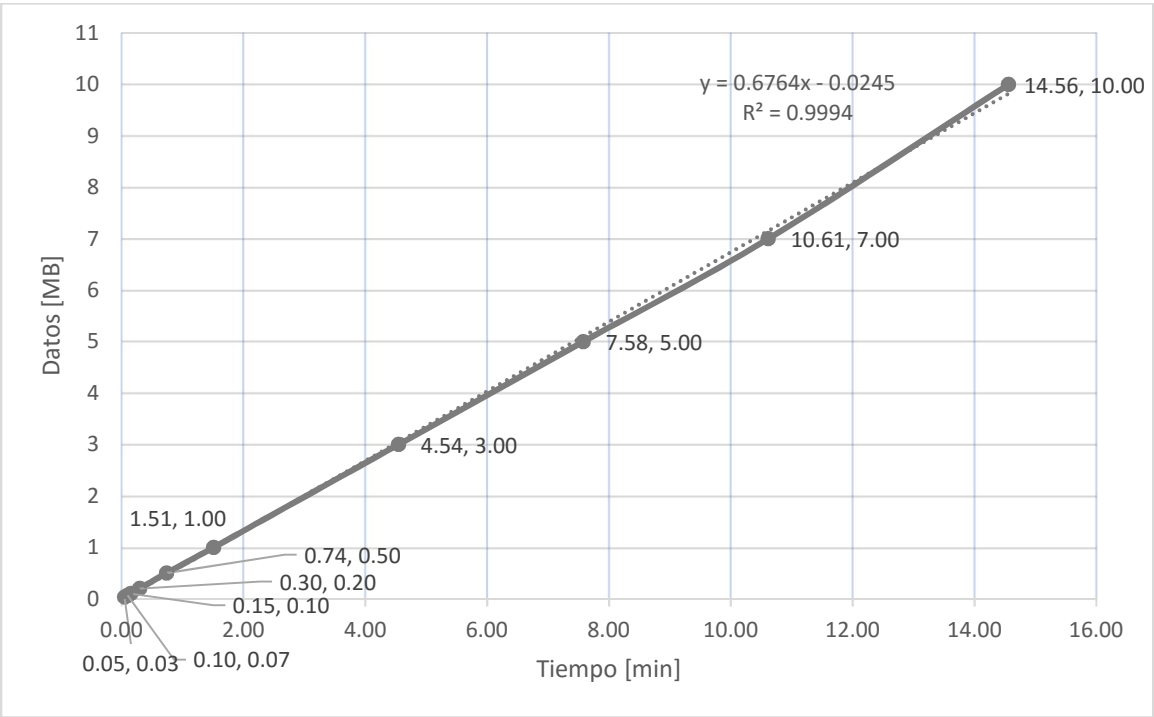


Figura 48. Gráfica de tiempos de recepción para imágenes en un rango entre 0.033 y 10MB.



En el Cuadro 8 se presentan los tiempos obtenidos de conversión, envío y recepción de datos para diferentes tamaños de imágenes. Se utilizaron diferentes imágenes para realizar las pruebas, utilizando como máximo una imagen de 10MB, tomando en cuenta que este es el tamaño de las imágenes que serán tomadas por Payload. En las figuras 46, 47, 48 se puede observar de una forma gráfica la tendencia que siguen los tiempos de conversión, envío y recepción de datos respectivamente. Con los resultados obtenidos para imágenes en el rango entre 0.033 y 10MB podemos notar que el tiempo de conversión, y los tiempos de envío y recepción de datos tienen una tendencia lineal. Estos datos nos indican el cómo se están comportando los tiempos con respecto a la cantidad de datos que se están convirtiendo, enviando o recibiendo, con lo cual se podría tomar una mejor decisión en cuanto a que valores de tamaño de imágenes se pueden utilizar, para envío de imágenes en menos de 4 minutos con la comunicación UART. Tomando en cuenta esto, se obtiene que una imagen ideal que puede ser convertida y enviar en menos de 4 minutos es aproximadamente de 2MB, o en caso que se desee enviar tres imágenes en menos de 4 minutos, idealmente se debería usar imágenes de un tamaño de 800KB.

Observando los datos que se recopilaron de las pruebas realizadas, se logró determinar que el microcontrolador es capaz de cumplir con los requerimientos del módulo OBC, y aunque los tiempos de envío y recepción de datos son significativamente altos en comparación con los tamaños de imágenes que se están enviando, esto fue causado por la tasa máxima de recepción de datos del microcontrolador del módulo COMMS puede soportar. También se puede observar que el microcontrolador es capaz de hacer la conversión de una imagen a arreglos de bits en poco tiempo, con un máximo de 62.3 segundos para una imagen de 10MB, por lo que se puede comprobar que es capaz de convertir una imagen a arreglos de bits y enviarlas al módulo de comunicaciones sin ningún problema.

Cuadro 18. Tiempo promedio de envío y recepción de imagen de 10MB con diferentes tasas de envío y recepción para diez repeticiones.

Tasa de transmisión [baud/s]	Tiempo de envío [min]	Tiempo para recepción [min]	Datos recibidos [B]
57600	29.10	29.11	10,077,696
115200	14.55	14.56	10,077,696
230400	11.75	11.76	10,077,696
460800	11.69	11.70	10,077,666
921600	11.11	11.17	9,735,081

Como se observar en el Cuadro 18 se realizaron pruebas para medir el tiempo de envío y recepción de datos para diferentes tasas de transmisión. Se observó que al aumentar la tasa de transmisión se redujo el

tiempo de envío y recepción de datos, y también se notó que se comenzaron a perder datos, lo cual se comprobó repitiendo la prueba diez veces para cada una de las diferentes tasas de transmisión. Esta pérdida de datos implica que la imagen a recibir no estará completa, lo cual impedirá que se vuelvan a ensamblar los datos como una imagen para poder ser analizada en tierra.

Cuadro 19. Tiempos de conversión, envío y recepción para una imagen de 500MB RAW con compresiones JPEG, PNG y TIFF.

Tipo	Tiempo de conversión [s]	Tiempo de envío [min]	Tiempo para recibir [min]	Tamaño [B]
JPEG	0.689	0.249	0.254	176558
PNG	2.814	0.759	0.791	528623
TIFF	2.245	0.766	0.770	533720

Cuadro 20. Tiempos de conversión, envío y recepción para una imagen de 1MB con compresiones JPEG, PNG y TIFF.

Tipo	Tiempo de conversión [s]	Tiempo de envío [min]	Tiempo para recibir [min]	Tamaño [B]
JPEG	1.642	0.492	0.496	344252
PNG	3.882	1.190	1.192	826748
TIFF	4.937	1.543	1.546	1071072

Cuadro 21. Tiempos de conversión, envío y recepción para una imagen de 3MB con compresiones JPEG, PNG y TIFF.

Tipo	Tiempo de conversión [s]	Tiempo de envío [min]	Tiempo para recibir [min]	Tamaño [B]
JPEG	4.957	1.706	1.724	1183621
PNG	10.339	3.885	3.882	2690003
TIFF	16.438	4.571	4.583	3163808

Para el módulo payload se tenía como referencia la utilización de imágenes RAW de 10MB, pero por lo que se logra observar en los resultados obtenidos, al momento de enviar las imágenes al módulo COMMS, este tarda bastante en ser enviados, por lo que se decidió realizar pruebas para verificar qué tanta diferencia existiría si se desean utilizar imágenes con algún tipo de compresión. Entre los resultados que se observan en los cuadros 19, 20 y 21, se logró reducir el tiempo de conversión, envío y recepción al utilizar imágenes con la compresión JPEG. El uso de imágenes comprimidas JPEG para ser enviadas hacía el sistema de

comunicaciones implica que se podría enviar una imagen casi tres veces más rápido que al hacer el envío de imágenes RAW.

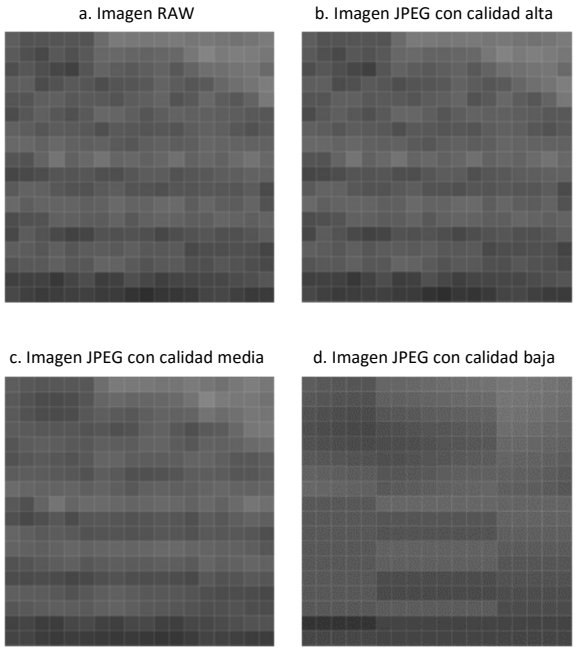
El problema que se puede presentar al hacer uso de imágenes JPEG es la pérdida de datos, esto es debido a que cada pixel de una imagen RAW contiene toda la información recopilada por el sensor de la cámara, mientras que una imagen con compresión JPEG ha perdido gran parte de esta información, esto es debido al algoritmo utilizado en la compresión, el cual descarta datos que para el ojo humano no son perceptibles, pero pueden ser de gran utilidad para aplicaciones en donde se desea hacer procesamiento de imágenes para el reconocimiento de contornos o aplicaciones similares. Debido a esto se realizaron pruebas para comparar qué tan similar podrían ser una imagen JPEG con respecto a la imagen RAW, y qué tanto se reduce el tamaño de cada imagen.

En las pruebas que se realizaron para comprobar la similitud entre las imágenes JPEG y las imágenes RAW, se usaron imágenes de 67KB, 1MB y de 10MB. Se usaron estas imágenes para poder observar si existía alguna diferencia al comprimir una imagen de poca resolución con respecto a una imagen de bastante resolución.

Cuadro 22. Porcentajes de similitud y de compresión de cantidad de datos de imágenes JPEG con diferentes calidades, con respecto a una imagen RAW de 68,926 Bytes.

Calidad	Similitud [%]	Tamaño [%]
Bajo	6.79	25.53
Medio	11.95	39.33
Alto	70.81	84.43

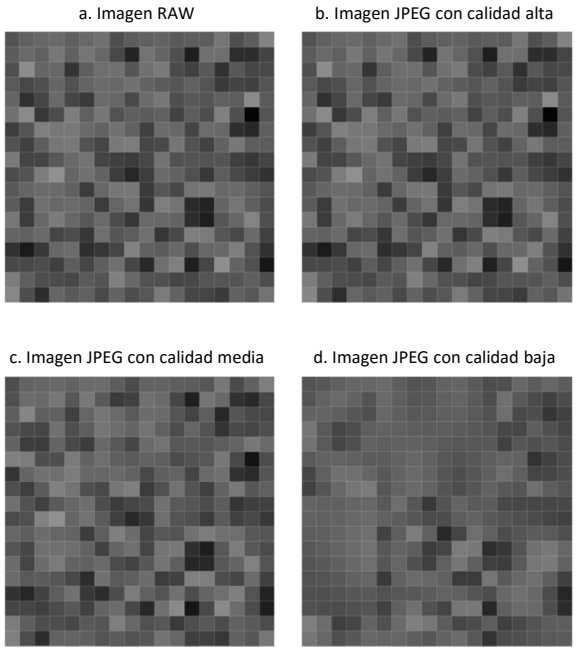
Figura 49. Zoom en imágenes RAW y JPEG para observar la degeneración de pixeles al comprimir una imagen RAW de 68,926 Bytes a diferentes calidades de JPEG.



Cuadro 23. Porcentajes de similitud y de compresión de cantidad de datos de imágenes JPEG con diferentes calidades, con respecto a una imagen RAW de 1,049,215 Bytes.

Calidad	Similitud [%]	Tamaño [%]
Bajo	3.02	6.86
Medio	6.97	32.81
Alto	70.94	83.89

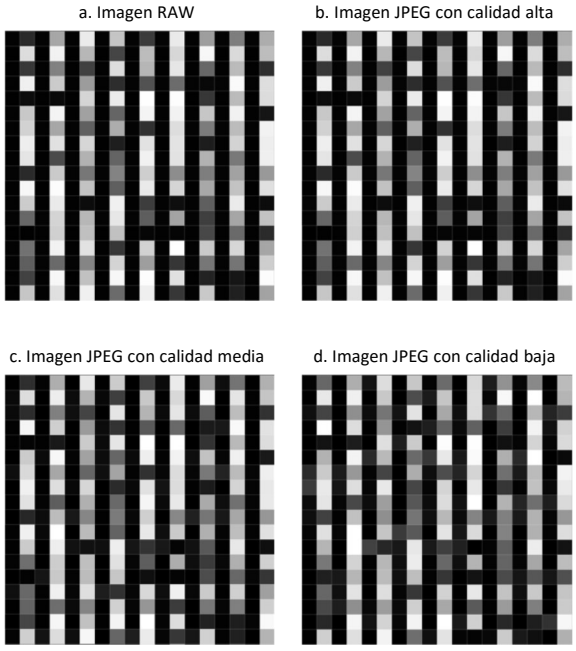
Figura 50. Zoom en imágenes RAW y JPEG para observar la degeneración de pixeles al comprimir una imagen RAW de 1,049,215 Bytes a diferentes calidades de JPEG.



Cuadro 24. Porcentajes de similitud y de compresión de cantidad de datos de imágenes JPEG con diferentes calidades, con respecto a una imagen RAW de 10,077,696 Bytes.

Calidad	Similitud [%]	Tamaño [%]
Bajo	5.44	49.80
Medio	9.34	59.96
Alto	66.02	113.19

Figura 51. Zoom en imágenes RAW y JPEG para observar la degeneración de pixeles al comprimir una imagen RAW de 10,077,696 Bytes a diferentes calidades de JPEG.

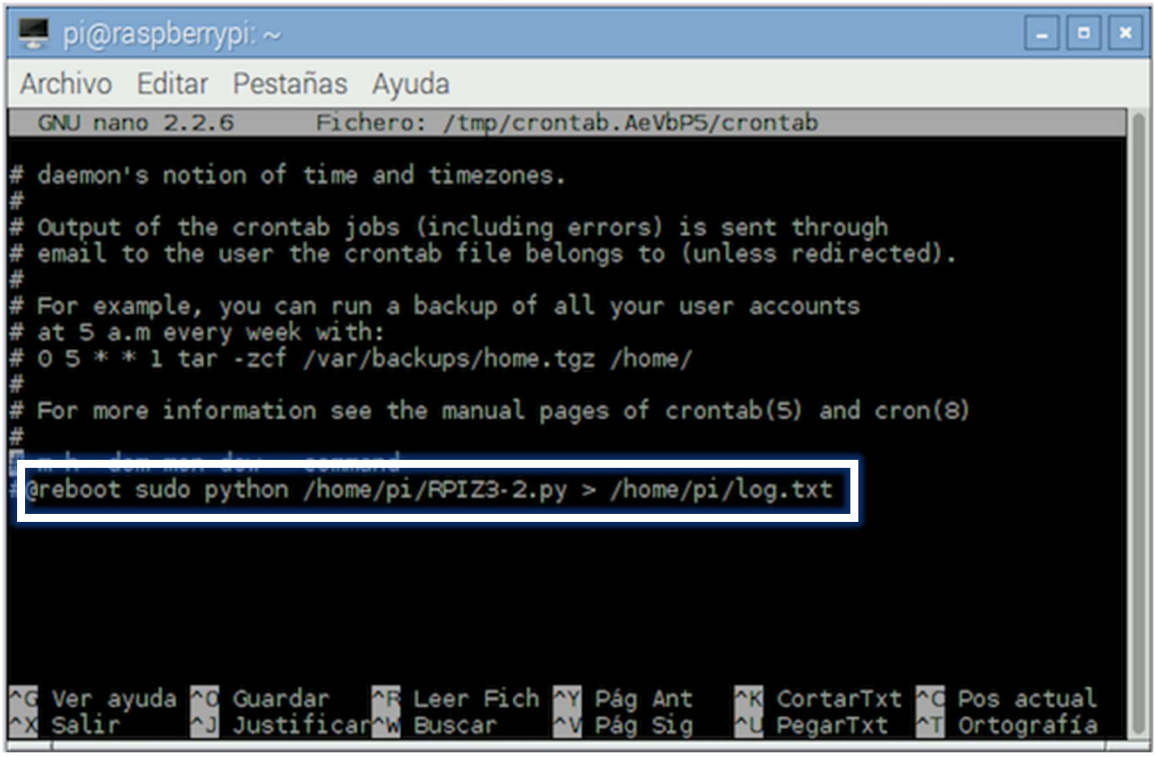


En los cuadros 22, 23 y 24 se obtuvo la similitud entre las matrices de pixeles de diferentes imágenes JPEG, a diferentes calidades, con respecto a una imagen RAW, y el tamaño en bytes de cada una de estas imágenes. Los datos de similitud se obtuvieron a partir de la comparación de la matriz de pixeles. Para realizar esta comparación se obtuvo la matriz de pixeles de la imagen RAW, y la matriz de pixeles de cada una de las imágenes JPEG. Luego se realizó la operación de diferencia entre matrices, con la cual a la matriz RAW se le sustrajo cada una de las matrices JPEG, guardando la matriz resultante en otra variable. Por último, se obtuvo el porcentaje de pixeles iguales, los cuales serían ceros entre la diferencia de las matrices de pixeles, que existen entre la imagen original con formato RAW, con respecto a cada una de las imágenes comprimidas JPEG, para lo cual se contabilizó la cantidad de ceros y a este número se le dividió la cantidad de pixeles que tiene la imagen. El resultado obtenido es el porcentaje de pixeles que siguen siendo iguales después de haber realizado la compresión, esto se hizo con el fin de poder observar qué tan parecidas son las imágenes a la imagen original después de haber realizado la compresión JPEG. Las figuras 49, 50 y 51 muestran un acercamiento de los pixeles para cada imagen RAW y para cada imagen comprimida JPEG, estos se utilizan para demostrar de una forma visual las diferencias entre cada pixel de las imágenes.

Lo que se puede observar en los resultados mostrados en los cuadros 22, 23 y 24 es que la similitud de pixeles entre la imagen original y la imagen con una calidad baja es muy reducida. Al comparar de forma

visual las figuras 49a, 49d, 50a, 50d, 51a, y 51d, las imágenes RAW y las imágenes JPEG de calidad baja, se puede observar el cambio en varios pixeles lo que comprueba lo encontrado en el porcentaje de similitud. Las imágenes JPEG con la mayor similitud a la imagen RAW son las que tienen una calidad alta, esto lo podemos comprobar al observar las figuras 49a, 49b, 50a, 50b, 51a, y 51b, en las cuales se nota la poca diferencia entre los pixeles. Aunque los valores en similitud son bastante altos para las imágenes de calidad alta, 70.81% en el caso de la imagen de 67KB, 70.94% en el caso de la imagen de 1MB y 66.02% en el caso de la imagen de 10MB, no se puede concluir que estos sean viables para ser enviados en vez de las imágenes RAW. Esto es debido a que la compresión del tamaño de la imagen no es suficiente para justificar la perdida de similitud con la imagen original, en la cual se redujo a un 84.43% el tamaño de la imagen de 67KB, 83.89% en el caso de la imagen de 1MB y se aumentó al 113.19% en el caso de la imagen de 10MB.

Figura 52. Configuración de arranque para iniciar el programa en Python desde el encendido del microprocesador.



Como se observa en la Figura 52, la última línea marcada es la configuración realizada al archivo de arranque del sistema operativo, con lo cual el programa creado pudo empezar a funcionar desde el encendido del microprocesador. Esto se realizó con el fin de poder orquestar a los diferentes módulos conectados al OBC desde el arranque del microprocesador, sin intervención humana.

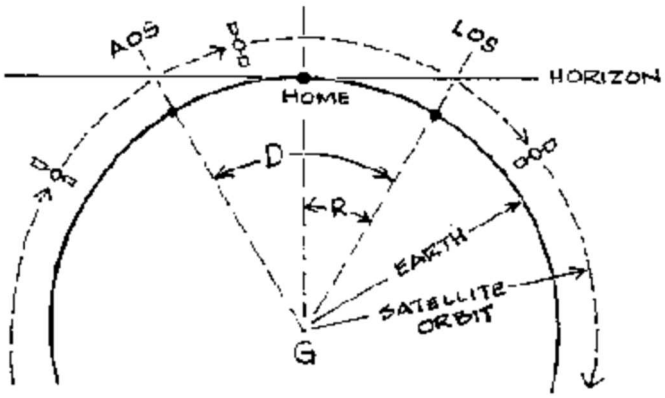
B. COMUNICACIÓN

1. Análisis de la ventana de comunicación.

Referirnos a la ventana de comunicación es referirnos a la cantidad de tiempo que el satélite estará comunicándose con la base en tierra. Durante este tiempo el satélite será capaz de transmitir y recibir información con la base en tierra. Este tiempo de comunicación depende directamente de los parámetros de la órbita en la cual el satélite estará orbitando alrededor de la tierra.

Se hicieron simulaciones de órbita en el simulador STK para el CubeSat de la Universidad del Valle de Guatemala y se logró determinar que la ventana de comunicación para la órbita seleccionada fue de 10 minutos, según el promedio obtenido de la simulación del comportamiento del satélite en la órbita (Lassen, 2014). Debido a la curvatura de la tierra y la curvatura de la órbita, el paso del satélite sobre la estación en tierra logra una comunicación exitosa definida entre un punto de adquisición de señal y un punto de pérdida de señal como se muestra gráficamente en la Figura 12 (Noe, 2004), definiendo entonces nuestra ventana de comunicación entre los puntos de adquisición de señal y pérdida de señal.

Figura 53: Ventana de comunicación



Otro dato muy importante para poder determinar la ventana de comunicación es la cantidad de pasadas que el satélite haría sobre la base en tierra. En las últimas simulaciones de la órbita elegida se obtuvo que el número de veces que pasa el satélite sobre Guatemala es de dos veces al día. Por tanto, tenemos en conclusión una ventana de comunicación de 10 minutos en promedio, cada ventana se da dos veces en un período de 24 horas. Para entonces continuar con la selección del ancho de banda que se requiere entonces se debe definir la cantidad de datos que el satélite va a transmitir. Puesto que el objetivo principal del satélite es tomar tres fotografías del lago de Atitlán, siendo el peso de cada una de las imágenes de 10MB, tenemos que debemos descargar 30MB en los 10 minutos que se tienen de ventana de comunicación. Por tanto, teniendo estos parámetros podemos llegar a determinar cuánto tiene que ser la velocidad de

transmisión de datos mínima, considerando 1MB = 8Mb y 1024kbps = 1Mbps, como se muestra en el Cuadro 1.

Cuadro 25. Parámetros de comunicación y determinación de la velocidad de transmisión mínima.

Parámetro	Valor
Ventana de comunicación	10 min
Tamaño de datos a transmitir	30 MB
Velocidad de transmisión mínima	409.60 kbps

2. Análisis de selección de frecuencia. La selección de la frecuencia a utilizar depende mucho de la aplicación que se requiera. Para la aplicación del CubeSat de la Universidad del Valle de Guatemala se requiere que el módulo de comunicaciones fuese capaz de poder descargar en 10 minutos un paquete de datos de 30 MB, necesitando, del Cuadro 25, una velocidad de 409.60 kbps. Las frecuencias más utilizadas por los satélites CubeSat están entre las bandas *Ultra High Frequency* (UHF), *Very High Frequency* (VHF) y Banda S en los rangos acorde a la IEEE. Estas bandas corresponden a los siguientes rangos de frecuencia:

Cuadro 26. Rango de frecuencias de las bandas utilizadas en CubeSat.

Banda	Rango de frecuencias
VHF	30 MHz a 300 MHz
UHF	300 MHz a 1 GHz
Banda S	2 GHz a 4 GHz

En este rango de frecuencias existen proveedores que brindan módulos de comunicaciones para ser utilizados en satélites CubeSat, algunas de estas características se pueden observar en el Cuadro 27. De las cuales podríamos calcular cuantas fotos podríamos descargar del satélite si se usa cada una de las bandas. La misión del CubeSat de la Universidad del Valle de Guatemala tendrá una misión de seis meses de duración, el equivalente a 180 días aproximadamente. Puesto que para la toma de fotografías se requiere la misma condición de luz para que el análisis de procesamiento de imagen no tenga mucha variación entonces se hace una toma por día. En total tendremos tres fotos en cada toma, 180 tomas, por tanto 540 fotos para toda la misión. Tomando en cuenta lo anterior, en el Cuadro 28 se presentan algunos datos para cada una de las bandas.

Cuadro 27. Características de módulos comerciales

Característica	VHF	UHF	Banda S
Consumo de voltaje [V]	6	9	6.5-30
Consumo de potencia [W]	4	0.5	3.5-6
Masa [g]	75	90	62-70
Velocidad de transmisión [kbps]	1.2	9.6	2000
Volumen [cm^3]	130	130	130
Banda utilizada [MHz]	140-150	420-450	2400 – 2483

(Lassen, 2014)(Clyde Space, 2015)(Clyde Space, 2015)(CubeSat Shop, 2015)(CubeSat Shop, 2015)

Cuadro 28. Cantidad de colecciones de imágenes que se pudieran descargar utilizando las distintas bandas comunes.

Característica	VHF	UHF	Banda S
Tiempo de descarga de colección de 3 imágenes [min]	3,413	427	2
Cantidad máxima de colecciones al día	0.003	0.023	5
Cantidad máxima de colecciones al mes	0.09	0.69	150
Cantidad máxima de colecciones obtenidas durante la misión	0.54	4.14	900

Con estos datos se realiza un estudio de caso para poder determinar de manera más objetiva qué banda de frecuencia es la más óptima para el CubeSat de la Universidad del Valle de Guatemala. Definimos por tanto las variables que vamos a considerar características importantes de cada una de las bandas, y se le asigna un peso de importancia según sea el caso. Normalizamos los valores de las variables en un rango de 1 a 10 com se muestra en el Cuadro 29.

Cuadro 29. Factor de peso y normalización de las variables.

Factor de peso		Valores normalizados de las variables									
Variable	Peso (1-4)	1	2	3	4	5	6	7	8	9	10
Consumo de potencia del módulo (W)	3	6	5.5	5	4.5	4	3.5	3	2.5	2	1
Consumo de potencia de la antena (W)	3	5	4.6	4.2	3.8	3.4	3	2.6	2.2	1.8	1
Tamaño de la antena del CubeSat (mm)	2	100	91	82	73	64	55	46	37	28	10
Velocidad de descarga de datos (kbps)	4	1.2	201.08	400.96	600.84	800.72	1000.6	1200.48	1400.36	1600.24	2000
Masa (g)	2	90	88	86	84	82	80	78	76	74	70
Volumen de la antena del CubeSat afuera del satélite(mm^3)	2	98028	89025.2	80022.4	71019.6	62016.8	53014	44011.2	35008.4	26005.6	8000
Consumo de voltaje (V)	3	30	27.6	25.2	22.8	20.4	18	15.6	13.2	10.8	6

Teniendo las variables normalizadas y con un peso según sea la prioridad, se hace la matriz de decisión para poder elegir de que banda se utilizará, esto se puede observar en el Cuadro 30.

Cuadro 30. Factor de peso y matriz de decisión.

Factor de peso		Matriz de decisión					
		VHF		UHF		Banda S	
Variable	Peso (1-4)	Normalizado	Total	Normalizado	Total	Normalizado	Total
Consumo de potencia del módulo (W)	3	10	30	7	21	1	3
Consumo de potencia de la antena (W)	3	8	24	8	24	1	3
Tamaño de la antena del CubeSat (mm)	2	1	2	1	2	10	20
Velocidad de descarga de datos (kbps)	4	1	4	1	4	10	40
Masa (g)	2	1	2	7	14	10	20
Volumen de la antena del CubeSat afuera del satélite (mm^3)	2	1	2	1	2	10	20
Consumo de voltaje (V)	3	10	30	10	30	1	3
Total			94		97		109

Se consideraron siete características importantes para poder seleccionar la banda que se utilizaría para el módulo de comunicaciones del CubeSat. El consumo de potencia y voltaje del módulo y consumo de potencia de la antena puesto que sabemos que la potencia en un CubeSat, más en uno de tamaño 1U, es limitada tanto la que pueden entregar los paneles solares como las baterías. El tamaño de la antena puesto que las antenas de tipo incremental necesitan un sistema de despliegue y es importante considerarlo ya que esto agregaría consumo, masa y volumen al CubeSat. La masa del módulo como tal y el volumen de la antena fuera del CubeSat ya que afecta el centro de masa del satélite. Y la última característica considerada, siendo una de las más importantes, es la velocidad de descarga de datos. Es de suma importancia puesto que la finalidad del CubeSat de la Universidad del Valle de Guatemala es poder obtener fotografías del

Lago de Atitlán, del Cuadro 28 podemos ver que la cantidad de colecciones de fotografías para análisis que podemos obtener es menor utilizando el rango de frecuencias bajas, y es mayor utilizando el rango de frecuencias altas, debido a la velocidad de descarga y la ventana de comunicación limitada que se tiene. Por tanto, si el objetivo es obtener la mayor cantidad de información posible, debido al peso de las imágenes y la limitada ventana de comunicación entonces es necesaria una velocidad de descarga de la cual se pueda obtener la mayor cantidad de información posible. Como resultado del estudio de caso se obtuvo la banda S como la banda de frecuencia que debe ser utilizada para el CubeSat de la Universidad del Valle de Guatemala.

3. Análisis de selección de componentes del módulo. Definir la banda de frecuencia que se va a utilizar es uno de los primeros pasos puesto que nos permite poder seleccionar los componentes necesarios para poder integrar el módulo de comunicaciones del satélite. Esto es puesto que los componentes funcionan para un rango de frecuencias específicas. Para poder seleccionar los componentes que conforman el módulo de comunicaciones del satélite, se analizó primero los componentes que han conformado el módulo de comunicaciones de CubeSat ya lanzados y puestos en órbita. Estos componentes se muestran en la siguiente tabla podemos observar que la mayoría utilizan microcontroladores como controlador del módulo, es común el uso del protocolo AX.25 para la transmisión y recepción de paquetes de datos. El *transceiver* utilizado varía entre los satélites y la banda de frecuencia más usada es la banda VHF. Por tanto, podemos partir en estudiar componentes similares para el CubeSat de la Universidad del Valle de Guatemala, cuyo funcionamiento sea posible en la banda seleccionada anteriormente.

Cuadro 31. Resumen de los componentes utilizados en misiones CubeSat

Satellite	Object	Size	Radio	Frequency	License	Power	TNC	Protocol	Baud Rate/Modulation	Downloaded	Lifetime	Antenna
AAU1 CubeSat	27846	1U	Wood & Douglas SX450	437.475 MHz	amateur	500 mW	MX909	AX.25, Mobitex	9600 baud GMSK	1 kB	3 months	dipole
DTUsat-1	27842	1U	RFMD RF2905	437.475 MHz	amateur	400 mW		AX.25	2400 baud FSK	0[1]	0 days	canted turnstile
CanX-1	27847	1U	Melexis	437.880 MHz	amateur	500 mW		Custom	1200 baud MSK	0[1]	0 days	crossed dipoles
Cute-1	27844	1U	Maki Denki (Beacon)	436.8375 MHz	amateur	100 mW	PIC16LC73A	CW	50 WPM	N/A	65+ months	monopole
(CO-55)			Alinco DJ-C4 (Data)	437.470 MHz	amateur	350 mW	MX614	AX.25	1200 baud AFSK	>10 MB		monopole
QuakeSat-1	27845	3U	Tekk KS-960	436.675 MHz	amateur	2 W	BayPac BP-96A	AX.25[2]	9600 baud FSK	423 MB	7 months	turnstile
XI-IV	27848	1U	Nishi RF Lab (Beacon)	436.8475 MHz	amateur	80 mW	PIC16C716	CW	50 WPM	N/A	65+ months	dipole
(CO-57)			Nishi RF Lab (Data)	437.490 MHz	amateur	1 W	PIC16C622	AX.25	1200 baud AFSK	>11 MB		dipole
XI-V	28895	1U	Nishi RF Lab (Beacon)	437.465 MHz	amateur	80 mW	PIC16C716	CW	50 WPM	N/A	36+ months	dipole
(CO-58)			Nishi RF Lab (Data)	437.345 MHz	amateur	1 W	PIC16C622	AX.25	1200 baud AFSK			dipole
NCube-2	28897[3]	1U		437.505 MHz	amateur			AX.25	1200 baud AFSK	0[1]	0 days	monopole
UWE-1	28892	1U	PR430	437.505 MHz	amateur	1 W	H8S/2674R[4]	AX.25	1200/9600 baud AFSK		0.75 months	end-fed dipole
Cute-1.7+APD	28941	2U	Telemetry Beacon	437.385 MHz	amateur	100 mW	H8S/2328[4]	CW	50 WPM	N/A	2.5 months	dipole
(CO-56)			Alinco DJ-C5	437.505 MHz	amateur	300 mW	CMX589A	AX.25/SRLL	1200 AFSK/9600 GMSK	<1 MB		dipole
GeneSat-1	29655	3U+	Atmel ATA8402 (Beacon)	437.067 MHz	amateur	500 mW	PIC12C617	AX.25	1200 baud AFSK	N/A	3 months	monopole
			Microhard MHX-2400	2.4 GHz	ISM	1 W	Integrated[5]	Proprietary		500 kB		patch
CSTB1	31122	1U	Commercial[6]	400.0375 MHz	Experimental	<1 W	PIC	Proprietary	1200 baud AFSK	6.77 MB[7]	19+ months	dipole
AeroCube-2	31133	1U	Commercial[6]	902-928 MHz	ISM	2 W	Integrated[5]	Proprietary	38.4 kbaud	500 kB	0.25 months	patch
CP4	31132	1U	TI CC1000	437.325 MHz	amateur	1 W	PIC18LF6720	AX.25	1200 baud FSK	487 kB	2 months	dipole
Libertad-1	31128	1U	Stensat	437.405 MHz	amateur	400 mW		AX.25	1200 baud AFSK	0[8]	1 month	monopole
CAPE1	31130	1U	TI CC1020	435.245 MHz	amateur	1 W	PIC16LF452	AX.25	9600 baud FSK	0[9]	4 months	dipole
CP3	31129	1U	TI CC1000	436.845 MHz	Experimental	1 W	PIC18LF6720	AX.25	1200 baud FSK	2.0 MB[7]	19+ months	dipole
MAST[10]	31126	3U	Microhard MHX-2400	2.4 GHz	ISM	1 W	Intgrated[5]	Proprietary	15 kbps	>2 MB	0.75 months	monopole
Delfi-C3	32789	3U	Custom Beacon	145.870 MHz	amateur	400 mW	PIC18LF4680	AX.25	1200 baud BPSK	60 MB[11]	7+ months	turnstile
(DO-64)			Custom Transponder	145.9-435.55 MHz	amateur	200 mW	N/A	Linear	40 kHz wide	N/A		turnstile
Seeds-2	32791	1U	Musashino Electric (Beacon)	437.485 MHz	amateur	90 mW		CW		N/A	7+ months	monopole
(CO-66)			Musashino Electric (Data)	437.485 MHz	amateur	450 mW		AX.25	1200 baud AFSK	500 kB		monopole
CanX-2	32790	3U	Custom S-Band	2.2 GHz	Space Research[12]	500 mW	Integrated	NSP	16kbps-256kbps BPSK	250 MB	7+ months	patch
AAUSAT-II	32788	1U	Holger Eckhardt (DF2FQ)	437.425 MHz	amateur	610 mW	PIC18LF6680	AX.25	1200 baud MSK	8 MB[13]	7+ months	dipole
Cute 1.7+APD II	32785	3U+[14]	Invax (Beacon)	437.275 MHz	amateur	100 mW	H8S/2328	CW	50 WPM	N/A	7+ months	monopole
(CO-65)			Alinco DJ-C5 (Data)	437.475 MHz	amateur	300 mW	H8S/2328, CMX589A	AX.25/SRLL	1200 AFSK/9600 GMSK	21 MB[15]		monopole
Compass-1	32787	1U	BC549 (Beacon)	437.275 MHz	amateur	200 mW	PIC12F629	CW	15 WPM	N/A	7+ months	dipole
			Holger Eckhardt (Data)	437.405 MHz	amateur	300 mW	C8051F123, FX614	AX.25	1200 baud AFSK/MSK	<1 MB		dipole

(Klofas *et. Al.*, 2008)

4. Terminal Node Controller. Se analizaron distintas propuestas de microcontroladores para poder controlar el módulo de comunicaciones, entre estas opciones están:

Cuadro 32: Características de los microcontroladores analizados

Variable	PIC16F887	PIC18F45K22	PIC18FL6720	ATMEGA328P	Teensy 3.2
Consumo de potencia (mW)	110	60.5	125	1200	69.19
Número de pines	40	40	64	28	34
Memoria de programa (KB)	14	32	128	32	256
RAM (KB)	368	1536	3840	2048	65536
EEPROM (KB)	256	256	1024	1024	2048

(Microchip)(Microchip)(Atmel)(Microchip)(Freescale)

Se hizo un estudio de caso considerando un peso para cada variable para seleccionar de manera objetiva el controlador más adecuado para la aplicación. Se obtuvo, por tanto:

Cuadro 33: Factor de peso y normalización de las características a comparar de cada microcontrolador.

Tabla de factor de peso		Valores de variables normalizadas									
Variable	Peso (1-4)	1	2	3	4	5	6	7	8	9	10
Consumo de potencia (mW)	4	1200	1086.05	972.1	858.15	744.2	630.25	516.3	402.35	288.4	60.5
Número de pines	3	28	31.6	35.2	38.8	42.4	46	49.6	53.2	56.8	64
Memoria de programa (KB)	4	14	38.2	62.4	86.6	110.8	135	159.2	183.4	207.6	256
RAM (KB)	2	368	6884.8	13401.6	19918.4	26435.2	32952	39468.8	45985.6	52502.4	65536
EEPROM (KB)	4	256	435.2	614.4	793.6	972.8	1152	1331.2	1510.4	1689.6	2048

Cuadro 34: Factor de peso de las características y matriz de decisión de los microcontroladores analizados

Tabla de factor de	Matriz de decisión

peso		PIC16F887		PIC18F45K22		PIC18FL6720		ATMEGA328P		Teensy 3.2	
Variable	Peso (1-4)	Normalizado	Total	Normalizado	Total	Normalizado	Total	Normalizado	Total	Normalizado	Total
Consumo de potencia (mW)	4	10	40	10	40	10	40	1	4	10	40
Número de pines	3	4	12	4	12	10	30	1	3	2	6
Memoria de programa (KB)	4	1	4	1	4	5	20	1	4	10	40
RAM (KB)	2	1	2	1	2	1	2	1	2	5	10
EEPROM (KB)	4	1	4	1	4	5	20	5	20	10	40
Total			62		62		112		33		136

Se tomó en cuenta para la decisión el consumo de potencia de los microcontroladores puesto que la potencia del CubeSat es limitada, el número de pines puesto, la capacidad de memoria siendo la de mayor peso la memoria EEPROM debido a que es la que permanece guardando todo incluso si el microcontrolador no recibe energía ya que puede llegar a quedarse sin energía llegando incluso a cero en potencia. De la matriz de decisión se concluye que la opción más adecuada para implementar como controlador del módulo es el Teensy 3.2.

a. Transceiver. Se analizaron las siguientes propuestas de *transceiver.*

Cuadro 35: Características de los transceiver analizados

Variable	MHX - 2400	CC2500	MC13202	AMB2520
Consumo de potencia en Rx (mW)	200	17	42	21
Consumo de potencia Tx (mW)	450	21.5	35	25
Salida RF (mW)	1000	1	2	2
Máxima tasa de transmisión de datos (kbps)	115	500	250	500

(Microhard System)(Freescale)(Amber Wireless)(*Texas Instruments*)

Cuadro 36: Factores de peso y variables normalizadas

Tabla de factor de peso		Valores de variables normalizadas									
Variable	Peso (1-4)	1	2	3	4	5	6	7	8	9	10
Consumo de potencia en Rx (mW)	2	200	181.7	163.4	145.1	126.8	108.5	90.2	71.9	53.6	17
Consumo de potencia Tx (mW)	3	450	407.15	364.3	321.45	278.6	235.75	192.9	150.05	107.2	21.5
Salida RF (mW)	3	1000	900.1	800.2	700.3	600.4	500.5	400.6	300.7	200.8	1
Máxima tasa de transmisión de datos (kbps)	4	115	153.5	192	230.5	269	307.5	346	384.5	423	500

Las opciones MHX-2400 y CC2500 son versiones de *transeivers* que ya han sido utilizados en módulos de comunicaciones, para las misiones de CubeSat. Las opciones MC13202 y AMB2520 son módulos similares a las opciones ya implementadas en CubeSats de los fabricantes Freescale y Amber Wireless respctivamente. Analizando las características se realizó un estudio de caso para seleccionar de manera objetiva el componente a utilizar, teniendo como resultado:

Cuadro 37: Matriz de decisión del transceiver con el valor total obtenido

Matriz de decisión							
MHX - 2400		CC2500		MC13202		AMB2520	
Normalizado	Total	Normalizado	Total	Normalizado	Total	Normalizado	Total
1	2	10	20	10	20	10	20
1	3	10	30	10	30	10	30
1	3	10	30	10	30	10	30
1	4	10	40	4	16	10	40
	12		120		96		120

Se analizaron las potencias de consumo para transmisión, recepción, salida y la tasa máxima de transmisión de datos. Las potencias son de suma importancia debido a que la producción de energía del CubeSat es limitado, y el módulo de comunicaciones necesita potencia para poder tranmsitir la onda. La máxima transmisión de datos es importante puesto que debemos transmitir paquetes grandes de información por lo que necesitamos que la velocidad de descarga del módulo sea lo más rápida posible, siempre manteniendo el consumo de potencia dentro de los límites permitidos y cumpliendo con el mínimo de velocidad que se obtuvo del análisis de ventana de comunicación de 409.6 kbps. Del caso de estudio obtenemos dos opciones principales, el CC2500 y el AMB2520. Analizando estas dos opciones se tiene que la transmisión de datos cumple con el requerimiento mínimo, y ya que ambas tienen la misma velocidad de transmisión de datos por lo que nos queda únicamente como punto de comparación para poder elegir una opción a la potencia que consumen los módulos. El CC2500 consume menos potencia que el AMB2520 por lo que el resultado de selección del *transceiver* es el CC2500 de Texas Instrument. Este componente será configurado y operado por el control seleccionado.

b. Antena. Para poder seleccionar el tipo de antena a utilizar en el CubeSat de la Universidad del Valle de Guatemala se hizo un estudio de caso entre tres tipos de antena: De parche, de dipolo y helicoidal. Las características consideradas para evaluar las antenas fueron:

Cuadro 38: Características de los tipos de antena analizados

Variable	Antena de parche	Antena de Dipolo	Antena helicoidal
Ganancia máxima (dB)	7	2.5	4.78
Número de bandas de frecuencia en las que puede operar	9	6	9
Tamaño (mm^2)	1166	4720	3184
Peso (g)	70	20	50

(Blevins, 1999)(Jamali, 2012)(Brown, 2002)

Estas tres opciones fueron descritas en el marco teórico, se calcularon los tamaños de cada antena para la banda seleccionada en el análisis de frecuencia de la siguiente manera:

Para la antena de dipolo, la longitud total de la antena es normalmente la mitad de la longitud de onda que se esté utilizando (National Instruments, 2015). Siendo la banda S el rango de frecuencia a utilizar se calculó el tamaño de la antena aproximado para la mitad de la longitud de onda de la frecuencia.

Para la antena de tipo parche se calcula con base a las propiedades del PCB en el cual se vaya a manufacturar. Se utilizaron las siguientes fórmulas (Afridi, 2015):

Ecuación 5: Cálculo del ancho del parche de la antena

$$W = \frac{C_0}{2f_r}\sqrt{\frac{2}{\varepsilon_r + 1}}$$

Where,
W = Width of the patch
C_0= Speed of light
ε_r= value of the dielectric substrate

Para poder calcular el largo del parche la antena es necesario calcular el valor efectivo de la constante dieléctrica del material y el incremento que tiene debido a los márgenes:

Ecuación 6: Cálculo del valor efectivo de la constante dieléctrica del material

$$\varepsilon_{reff} = \frac{\varepsilon_r + 1}{2} + \frac{\varepsilon_r - 1}{2}\left[1 + 12\frac{h}{W}\right]^{-1/2}, {}^{W}/_{h} > 1$$

Ecuación 7: Cálculo del incremento del largo de la antena debido a los márgenes

$$\frac{\Delta L}{h} = 0.412\frac{\left(\varepsilon_{reff} + 0.3\right)\left(\frac{W}{h} + 0.264\right)}{\left(\varepsilon_{reff} - 0.258\right)\left(\frac{W}{h} + 0.8\right)}$$

Where 'h'= height of the substrate

Obteniendo los valores anteriores se puede calcular el largo total del parche de la antena:

Ecuación 8: Largo del parche de la antena

$$L = \frac{C_0}{2f_r\sqrt{\varepsilon_{reff}}} - 2\Delta L$$

Obteniendo el valor de largo y ancho del parche se puede obtener las dimensiones finales de la antena en el PCB con las siguientes fórmulas:

Ecuación 9: Largo y ancho del PCB de la antena

$$L_g = 6h + L$$
$$W_g = 6h + W$$

Se utilizaron las siguientes características de PCB para el cálculo de la antena (Prototype and Circuit Board):

Cuadro 39: Características del PCB utilizado para el cálculo de la antena

Característica	Valor
Material	FR4 Glass Epoxi Resin
Constante del dieléctrico	4.1 – 4.4
Alto del dieléctrico	0.0014 pulgadas

Para la antena helicoidal se calculó el tamaño utilizando las siguientes fórmulas:

Sea:

$$N = 1$$

$$S = 0.25$$

$$f = 2.4\ GHz$$

Donde:

N = Número de vueltas de la hélice

S = Espacio entre hélices

f = Frecuencia a utilizar

La ganancia esta dada por:

$$G = 10.8 + 10log(\frac{C^2 NS}{\lambda}) \qquad (9)$$

Donde:

C = 1, la circunferencia, que se selecciona cercana a una longitud de onda

Se analizaron las características de cada tipo de antena en un estudio de caso obteniendo los siguientes resultados:

Cuadro 40: Pesos y normalización de las características de las antenas a evaluar

Tabla de factor de peso		Valores de variables normalizadas									
Variable	Peso (1-4)	1	2	3	4	5	6	7	8	9	10
Ganancia máxima (dB)	4	2.5	2.95	3.4	3.85	4.3	4.75	5.2	5.65	6.1	7
Número de bandas de frecuencia en las que puede operar	2	6	6.3	6.6	6.9	7.2	7.5	7.8	8.1	8.4	9
Tamaño (mm^2)	4	4720	4364.6	4009.2	3653.8	3298.4	2943	2587.6	2232.2	1876.8	1166
Peso (g)	3	70	65	60	55	50	45	40	35	30	20

Cuadro 41: Características de las antenas analizadas y matriz de decisión

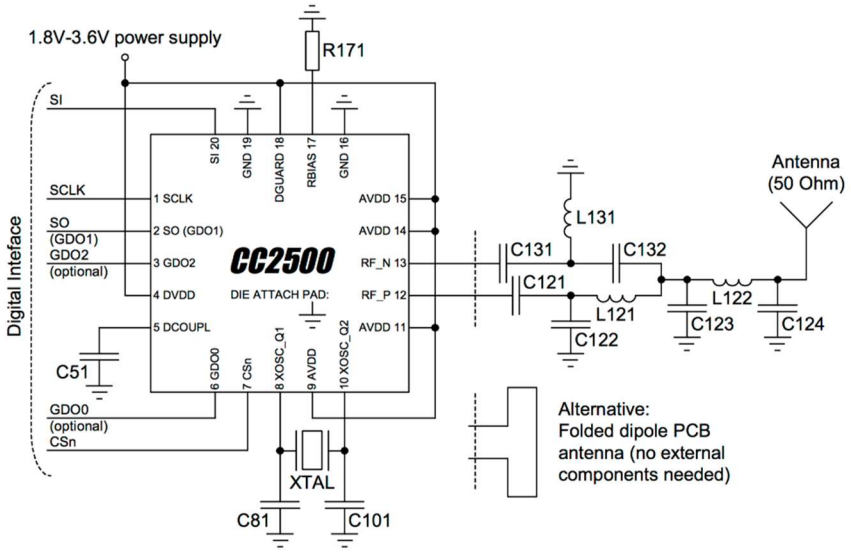
Matriz de decisión					
Antena de parche		Antena de dipolo		Antena de agujero	
Normalizado	Total	Normalizado	Total	Normalizado	Total
10	40	1	4	6	24
10	20	1	2	10	20
10	40	1	4	6	24
1	3	10	30	5	15
	103		40		83

Se analizó la ganancia máxima obtenida por la antena, el número de bandas en la cual puede operar ese tipo de antena, el tamaño aproximado para la banda seleccionada y el peso de la antena. La ganancia máxima y el tamaño son de suma importancia debido a la distancia que se quiere cubrir y el tamaño limitado que se tiene del CubeSat. El peso es importante debido a la restricción que tiene el CubeSat en su diseño. Las bandas en las que puede operar es un factor a considerar puesto que si se escala el proyecto y se desea utilizar el diseño de antena no es necesario cambiar el tipo de antena si se mantiene bajo los rangos que se puede operar ese tipo de antena por lo que tener más rangos de banda de frecuencia disponible es una ventaja considerable. Con base a la matriz de decisión del Cuadro 41 se concluye que el tipo de antena seleccionada para el proyecto es de tipo parche.

5. Diseño e implementación del sistema de comunicaciones.

Ya seleccionados los componentes se procede a diseñar el sistema. El *transceiver* utilizado tiene un circuito de aplicación que brinda el fabricante, el cual se muestra en la Figura 54

Figura 54: Circuito de aplicación para el transceiver CC2500 por Texas Instruments



(*Texas Instruments*)

Los componentes utilizados tienen las siguientes funciones:

Cuadro 42: Descripción de los componentes externos del transceiver

Componente	Descripción
C51	Capacitor de desacople para el regulador del chip de la parte digital
C81/C101	Capacitores de carga para el cristal/oscilador
C121/C131	Capacitores de bloqueo DC del convertidor de señal balanceada a señal desbalanceada
C122/C132	Capacitores de acople del convertidor de señal balanceada a señal desbalanceada
C123/C124	Capactor para filtro LC
L121/L131	Inductores para convertidor de señal balanceada a señal desbalanceada
L122	Inductor para filtro LC
R171	Resistencia para la referencia de corriente para el circuito bias interno
XTAL	26-27 MHz cristal/oscilador

(*Texas Instruments*)

Estos componentes tienen los siguientes valores:

Cuadro 43: Valores de los componentes del circuito de aplicación

Componente	Valor
C51	100 nF
C81	27 pF
C101	27 pF
C121	100 pF
C122	1.0 pF
C123	1.8 pF
C124	1.5 pF
C131	100 pF
C132	1.0 pF
L121	1.2 nH
L122	1.2 nH
L131	1.2 nH
R171	56 kΩ
XTAL	26 Hz

(*Texas Instruments*)

Se utilizó el *transceiver* CC2500 en su circuito de aplicación proporcionado por el fabricante Mikroelektronika, el cual implementa una antena de tipo parche para su transmisión y recepción.

Figura 55: Parte frontal de la implementación de Mikroelektronika

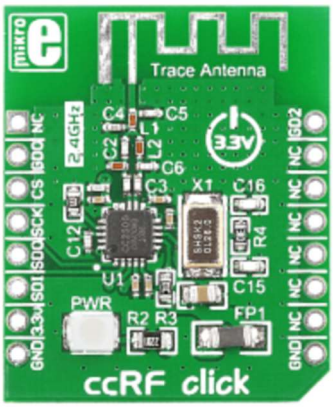
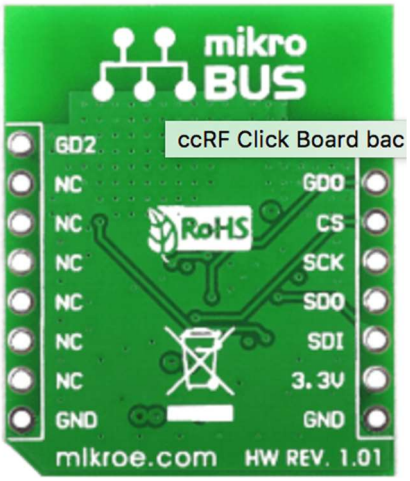


Figura 56: Parte trasera de la implementación de Mikroelektronika



Esta implementación se controlará por medio del transceiver seleccionado Teensy 3.2. La interfaz de conexión entre el transceiver y el Teensy será mediante SPI ya que es mediante este protocolo de comunicación que se configuran los registros necesarios para poder enviar y recibir información mediante el *transceiver.* Las conexiones se hacen de la siguiente manera:

Figura 57: Circuito de la implementación de Mikroelecktronika con los valores utilizados

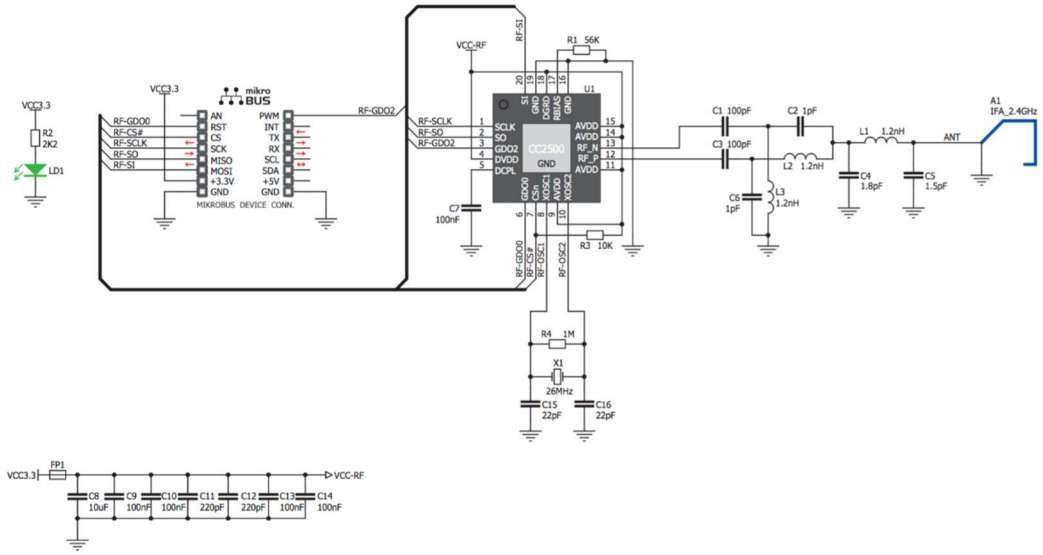
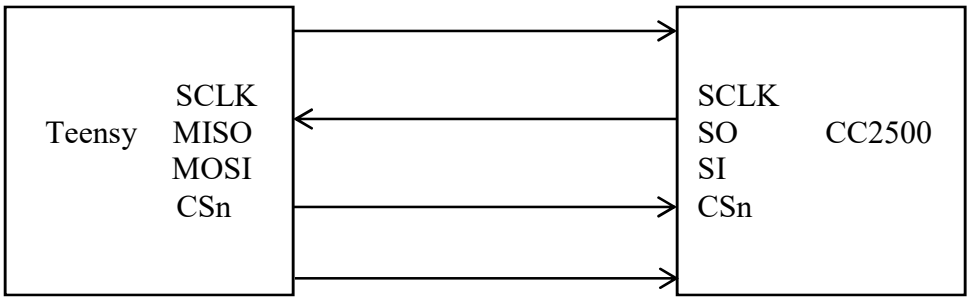


Figura 58: Diagrama de bloque de conexiones entre el Terminal Node Controller y el Transceiver



6. Pruebas e integración. Para las pruebas de la implementación del módulo se realizó la programación del módulo en alto nivel utilizando como IDE de programación el proporcionado por Arduino: *Arduino Software 1.6.11.* El *Terminal Node Controller* implementado puede ser programado mediante este IDE, utilizando un complemento para poder cargar el código de programación llamado *Teensyduino 1.30.* El IDE se encarga de compilar el programa en lenguaje de alto nivel y de traducirlo a un código en hexadecimal, para poder ser cargado en el *Terminal Node Controller.* El código utilizado para poder enviar y recibir información se encuentra en anexos. Se utilizaron los siguientes módulos del TNC:

1. SPI: Este módulo se encarga de poder enviar y recibir instrucciones mediante el protocolo de comunicaciones SPI, se utilizó para poder configurar los registros del *transceiver* y poder enviar los bytes a transmitir por radiofrecuencia.
2. Serial: Este módulo se encarga de poder enviar y transmitir bytes de manera asíncrona mediante el protocolo de comunicación RS232. Este módulo se encargó de comunicarse con la computadora de a bordo del CubeSat, la cual enviaba los bytes de la foto a enviar.

Se transmitieron dos paquetes de datos, uno indicando el estado de salud del CubeSat, el cual se está enviando en todo momento, y otro paquete simulando la foto que se recibe de la computadora de abordo.

Figura 59: Código utilizado para envío de fotos mediante radiofrecuencia obtenidas de la computadora de abordo

```
void sendPacketP() {
  WriteReg(REG_IOCFG1,0x06);
  // Make sure that the radio is in IDLE state before flushing the FIFO
  SendStrobe(CC2500_IDLE);
  // Flush TX FIFO
  SendStrobe(CC2500_FTX);
  // prepare Packet
  int length = 13;
  unsigned char packet[length];
  // First Byte = Length Of Packet
  packet[0] = length;
  packet[1] = 'P';
  packet[2] = 'H';
  packet[3] = 'O';
  packet[4] = 'T';
  packet[5] = 'O';
  packet[6] = ' ';
  packet[7] = 'H';
  packet[8] = 'E';
  packet[9] = 'R';
  packet[10] = 'E';
  packet[11] = '<';
  packet[12] = '-';
  // SIDLE: exit RX/TX
  SendStrobe(CC2500_IDLE);
  Serial.println("Transmitting ");
  for(int i = 0; i < length; i++)
  {
      WriteReg(CC2500_TXFIFO,packet[i]);
  }
```

Figura 60: Código utilizado para envío de estado de salud del satélite mediante radiofrecuencia

```
void sendPacketB() {
  WriteReg(REG_IOCFG1,0x06);
  // Make sure that the radio is in IDLE state before flushing the FIFO
  SendStrobe(CC2500_IDLE);
  // Flush TX FIFO
  SendStrobe(CC2500_FTX);
  // prepare Packet
  int length = 10;
  unsigned char packet[length];
  // First Byte = Length Of Packet
  packet[0] = length;
  packet[1] = 'C';
  packet[2] = 'u';
  packet[3] = 'b';
  packet[4] = 'e';
  packet[5] = 'S';
  packet[6] = 'a';
  packet[7] = 't';
  packet[8] = 'O';
  packet[9] = 'K';

  // SIDLE: exit RX/TX
  SendStrobe(CC2500_IDLE);

  Serial.println("Transmitting ");
  for(int i = 0; i < length; i++)
  {
      WriteReg(CC2500_TXFIFO,packet[i]);
  }
```

Figura 61: Recepción de paquetes enviados por radiofrecuencia

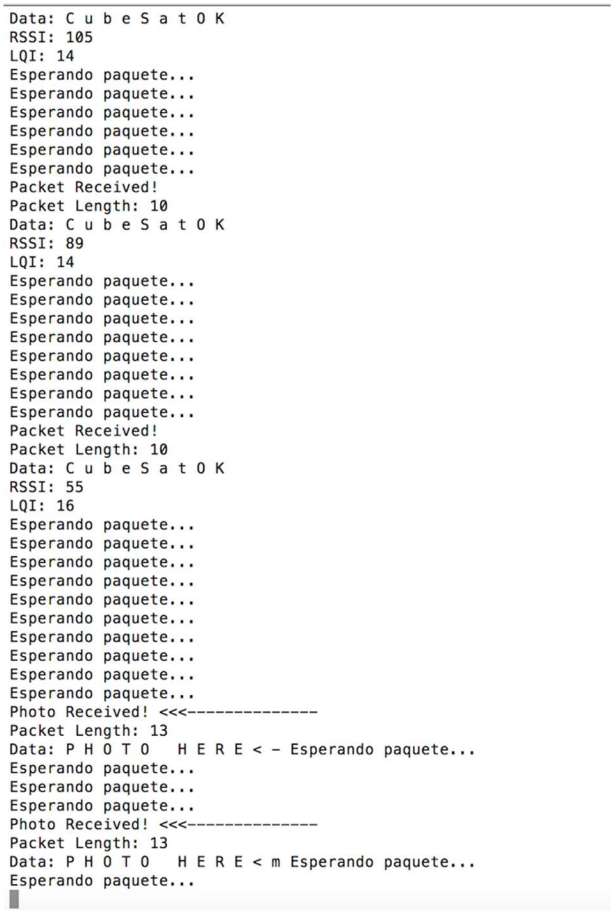


Figura 62: Código utilizado, recepción de paquetes y envío de paquetes

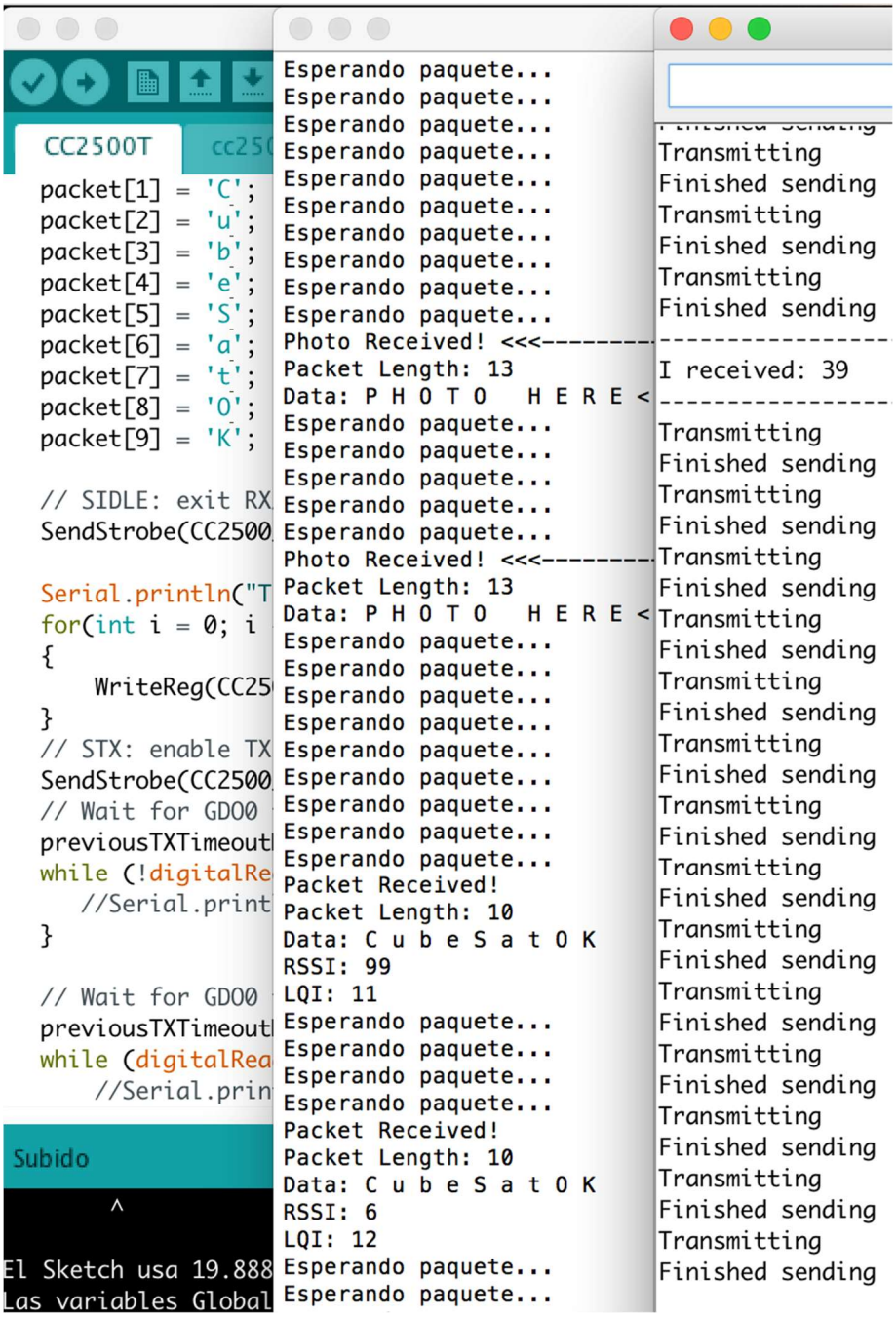


Figura 63: Integración con computadora de abordo

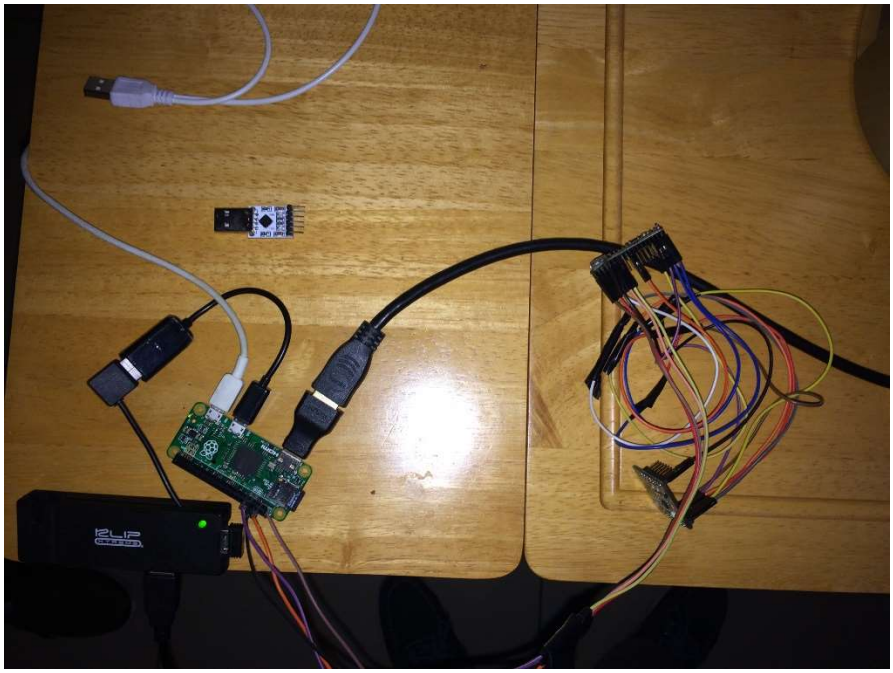


Figura 64: Pruebas de integración con computadora de abordo

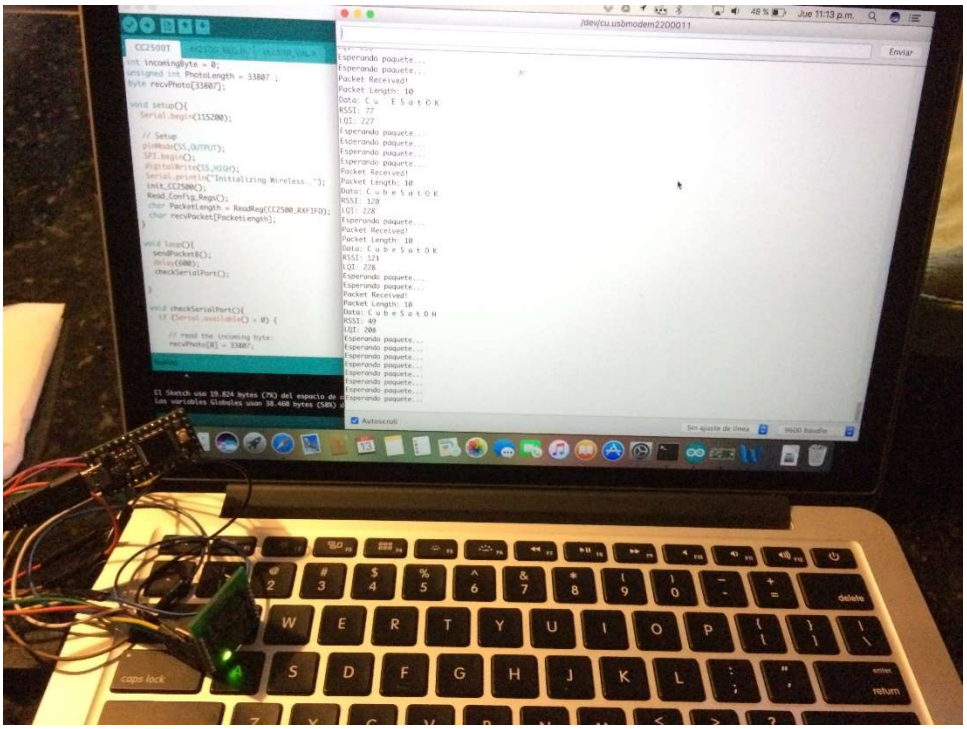
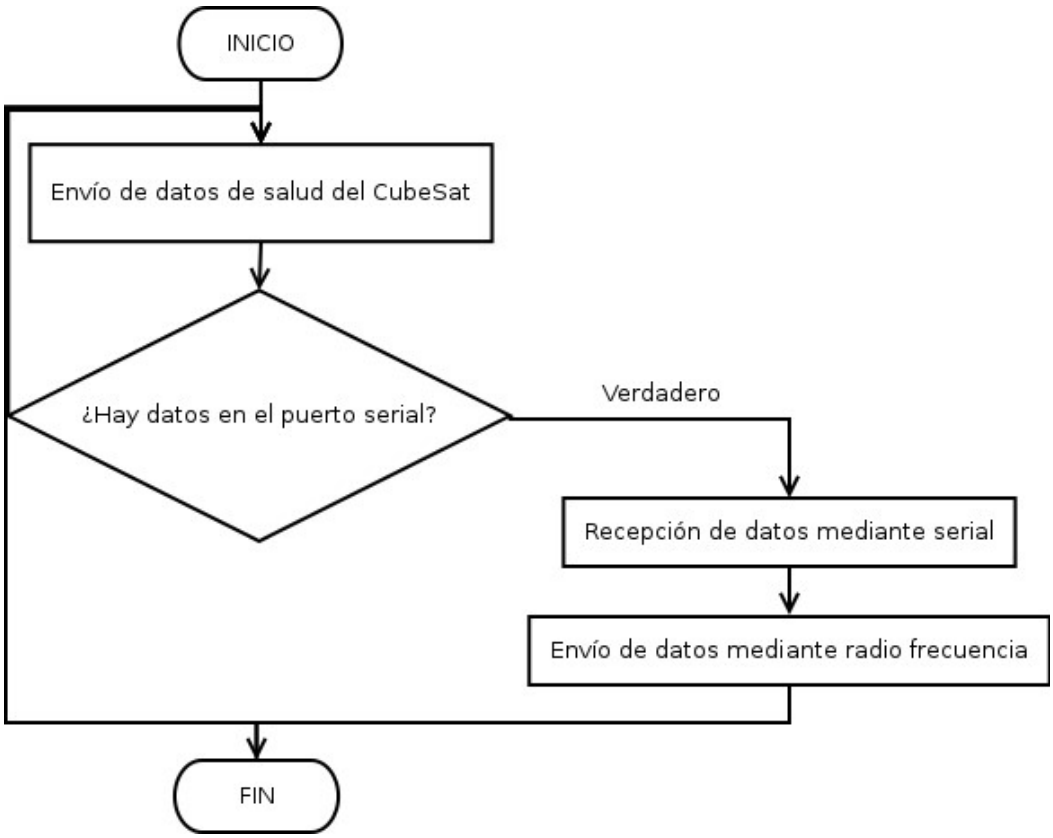


Figura 65: Diagrama de flujo del funcionamiento de la transmisión de datos



C. POTENCIA

1. Selección de paneles solares

Cuadro 44. Trade Study de posibles paneles solares.

Factores de peso		Valores Normalizados		Matriz de decisión					
				NanoPower (Fase II)		3G28A		3G30A	
Variable	Peso (0-4)	1	10	Normalizado	Total	Normalizado	Total	Normalizado	Total
Eficiencia (%)	1	15	30	10	10	9	9	10	10
Voltaje (V)	4	2	8	6	24	7	28	8	32
Amperaje (mA)	4	200	1000	5	20	6	24	7	28
Volumen (mm^3)	2	3.00E+04	1.50E+04	9	18	9	18	9	18
Masa (Kg)	1	0.06	0.025	5	5	5	5	5	5
Temperatura Max (°C)	4	100	200	7	28	7	28	7	28
Temperatura Min (°C)	2	-5	-60	7	14	7	14	7	14
Precio (€)	3	2000	240	1	3	8	24	7	21
					122		150		156

Es importante aclarar que dado que los modelos 3G28A y 3G30A, son únicamente celdas fotovoltaicas y no un panel solar en conjunto, para poder ser comparado con el modelo NanoPower seleccionado en la Fase II del proyecto, se utilizaron los valores de corriente, voltaje y precio de dos celdas por panel solar, además con respecto a la masa y volumen, se asumió que en conjunto las celdas fotovoltaicas ya unidas a una placa tendrían las mismas características que el modelo de NanoPower para que dicho valor no fuera relevante, dado que es únicamente una suposición, pero en la vida real lo más probable es que estas características no varíen de forma drástica al construir un panel solar con dichas celdas.

Como resultado final se obtuvo un mejor resultado para las celdas del modelo 3G30A, dado que ambos modelos fabricados por Azur Space tienen mejores valores de corriente, voltaje y precio, con respecto al modelo de NanoPower, la decisión tal y como lo muestra la comparación, genera resultados a favor de utilizar celdas fotovoltaicas, y dado que el modelo 3G30A tiene una mejor eficiencia con respecto al modelo 3G28A, de igual forma genera mayor corriente y voltaje de salida, aunque su precio sea de igual forma mayor que el del modelo 3G28A, sigue siendo un factor más relevante las características como

fuente de poder para el proyecto. Por lo tanto, tal y como se mencionó anteriormente, la mejor opción es utilizar celdas solares Azur Space modelo 3G30A.

2. Selección de la órbita

Cuadro 45. Resultados de la simulación en STK de cada órbita.

Dato	ISS	Spaceflight						Gauss Dnepr
		S500-27D	S575-98D	S500-60D	S515-98D	S450-20D	S500-98D	G600-97D
A	63	140	70	65	63	212	63	54
B	25	65	35	27	29	95	26	26
C	29.2	11.23	20.86	27.03	25.17	7.68	28.08	28.08
D	427	808	344	391	340	1486	344	254
E	1.71	0.90	2.12	1.87	2.15	0.49	2.12	2.87
F	202	365	175	173	159	683	157	120
G	3.61	2.00	4.17	4.17	4.59	1.07	4.65	6.08

A. Total de sobrevuelos sobre el Lago de Atitlán.
B. Sobrevuelos sobre el Lago de Atitlán en periodos de luz.
C. Promedio de días entre cada sobrevuelo sobre el Lago de Atitlán en periodos de luz.
D. Total de sobrevuelos sobre Guatemala.
E. Promedio de días entre cada sobrevuelo sobre Guatemala.
F. Sobrevuelos sobre Guatemala en periodos de luz.
G. Promedio de días entre cada sobrevuelo sobre Guatemala en periodos de luz.

Cuadro 46. Periodos de luz y sombra de la órbita elegida.

Dato	Tiempo (s)	Tiempo (min)
Periodo de luz	3553	59.22
Periodo de sombra	2030	33.83
Duración de la órbita	5583	93.05

Figura 66. Simulación 3D en STK de la órbita S450-D20.

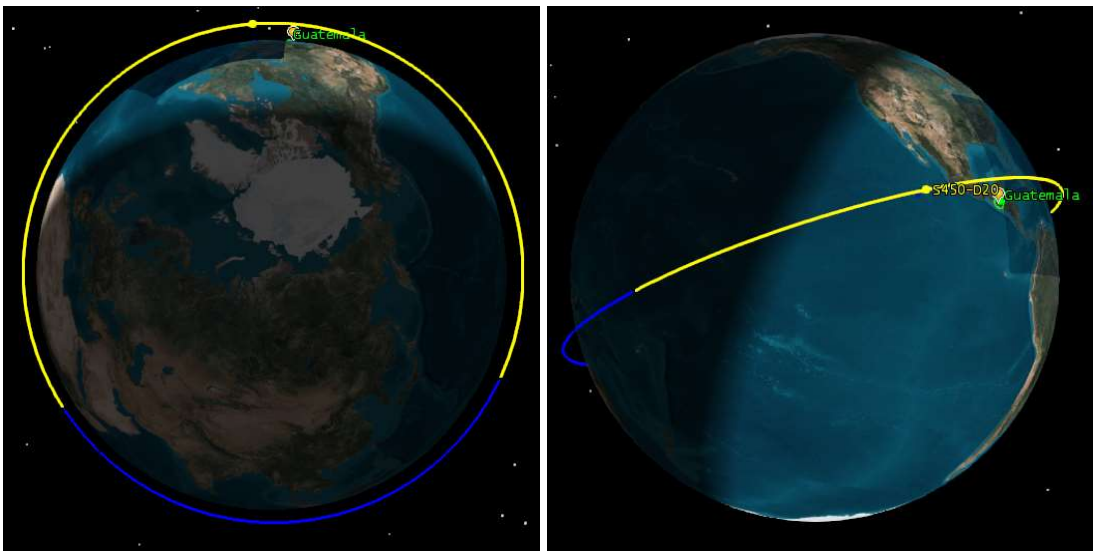


Figura 67. Simulación 2D en STK de la órbita S450-D20.

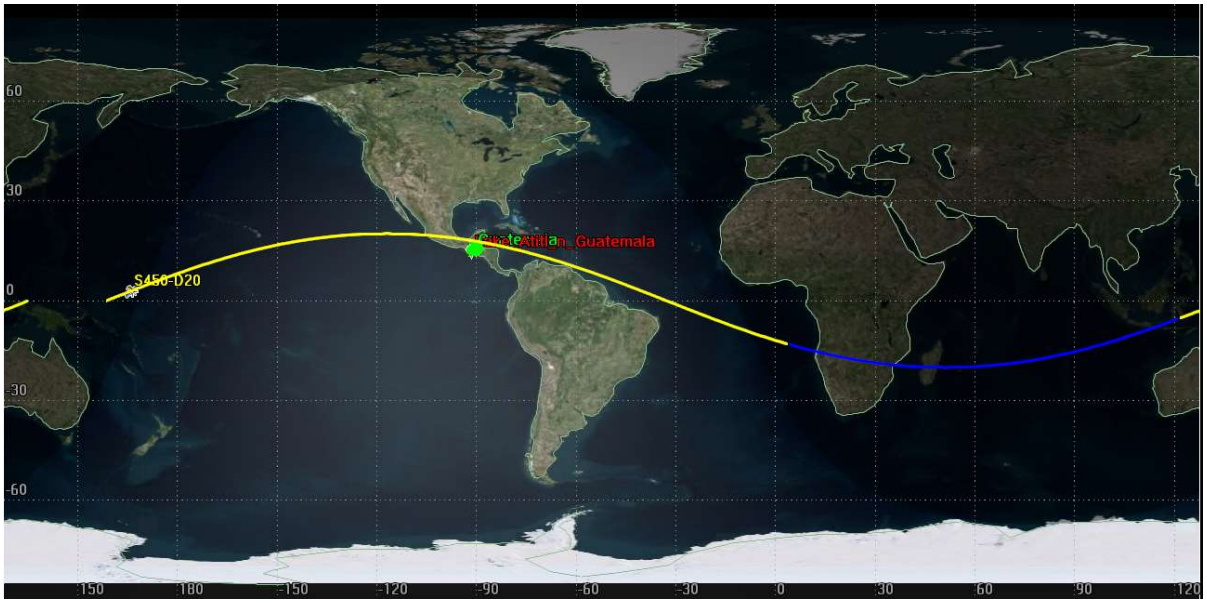


Figura 68. Periodo de luz útil de la órbita para el sistema de paneles solares.

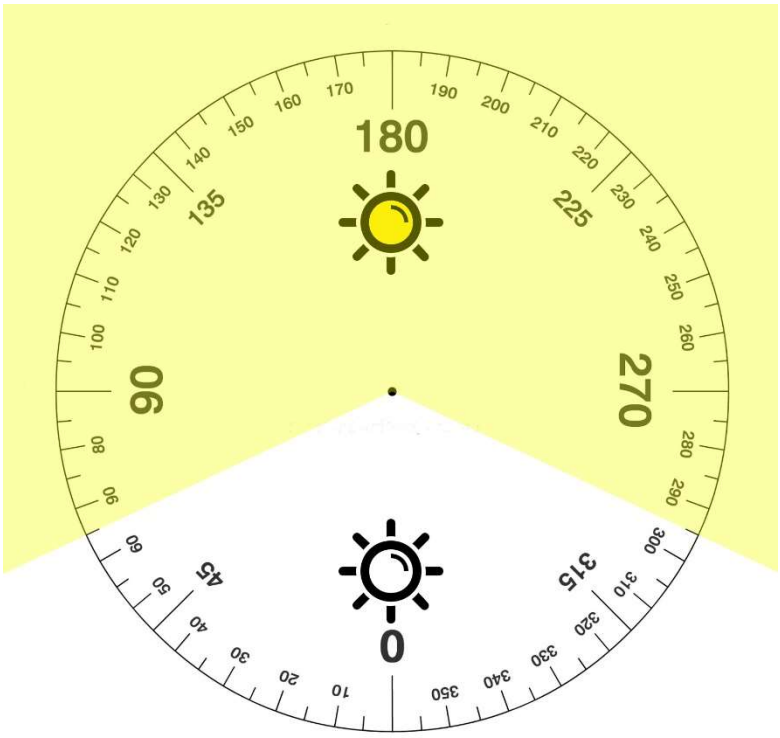
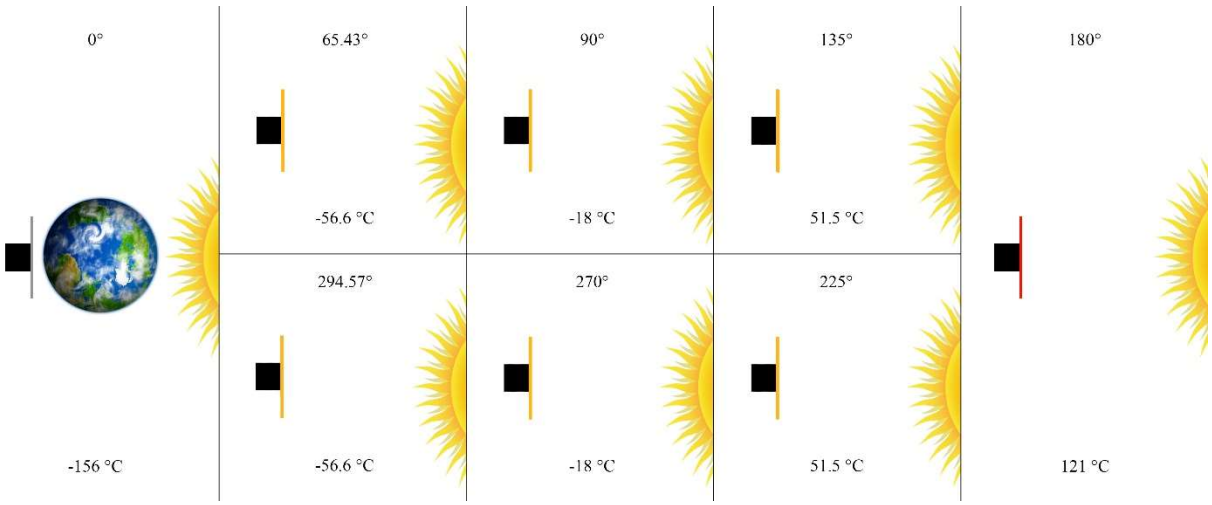


Figura 69. Posición del satélite con respecto al Sol a lo largo de la órbita.



Cuadro 47. Tiempo posible de contacto entre el satélite y la estación en Tierra.

Dato	Tiempo (s)	Tiempo (min)
Duración mínima	3.96	0.066
Duración máxima	757.12	12.63
Duración media	655.92	10.93

3. Consumo de los módulos dentro del cubesat

Cuadro 48. Consumo definido para cada módulo.

Módulo	Consumo (mW)	Tiempo de Operación (s)
OBC	600	5583
Payload	1,500	758
ADCS	1,400	5586
Communicacion	3,380	758

Cuadro 49. Energía consumida por cada módulo en una órbita.

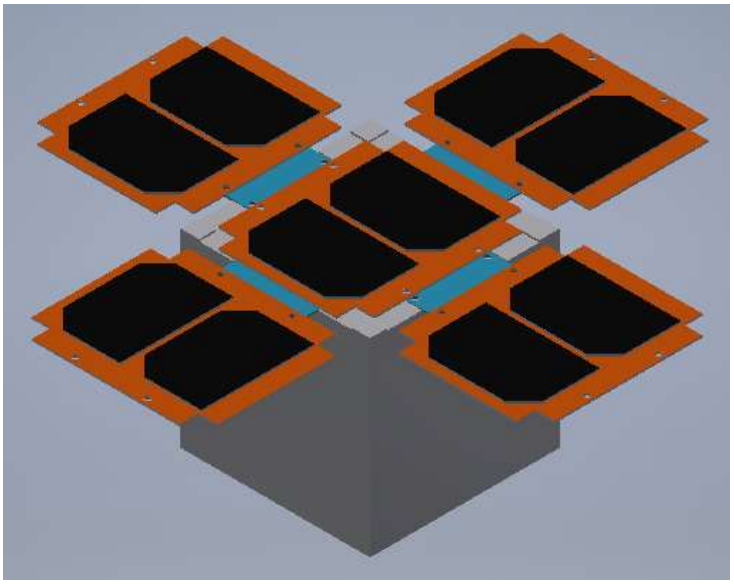
Módulo	Energía (Joules)	Energía (mAh)*
OBC	3,349.80	251.49
Payload	1,137.00	85.36
ADCS	7,820.40	587.12
Communicacion	2,562.04	192.35

*Calculado para una batería con 3.7 V como voltaje nominal.

4. Modelo con cinco paneles solares

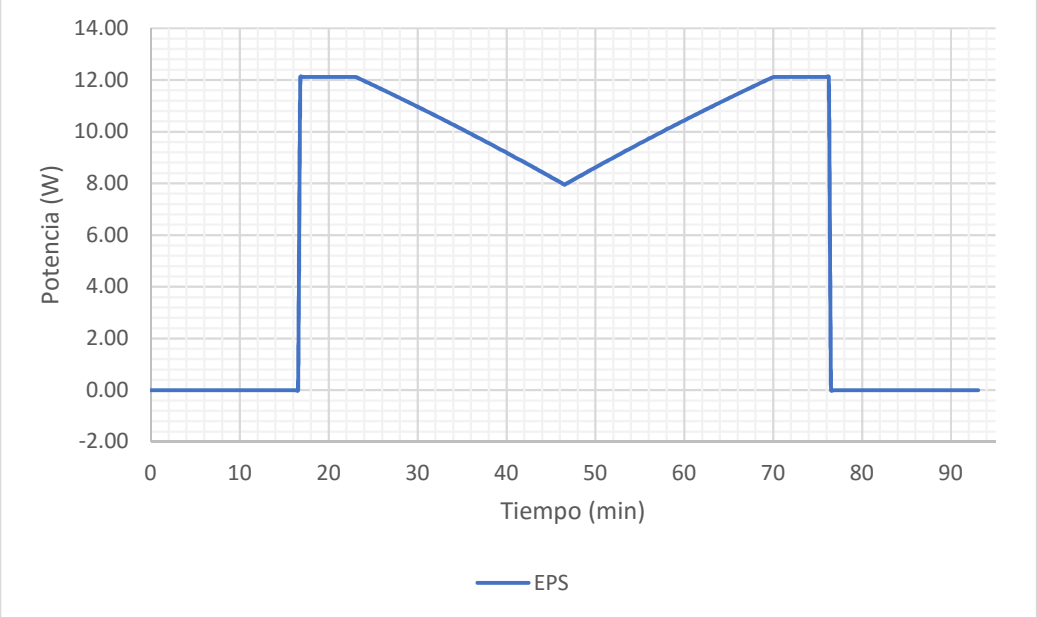
a. Aspecto físico

Figura 70. Modelo 5PS, con cinco paneles solares.



b. Potencia generada

Figura 71. Potencia a lo largo de la órbita, Modelo 5PS.



c. Potencia generada vs. potencia consumida, Escenario A

Figura 72. Potencias a lo largo de la órbita escenario A, Modelo 5PS.

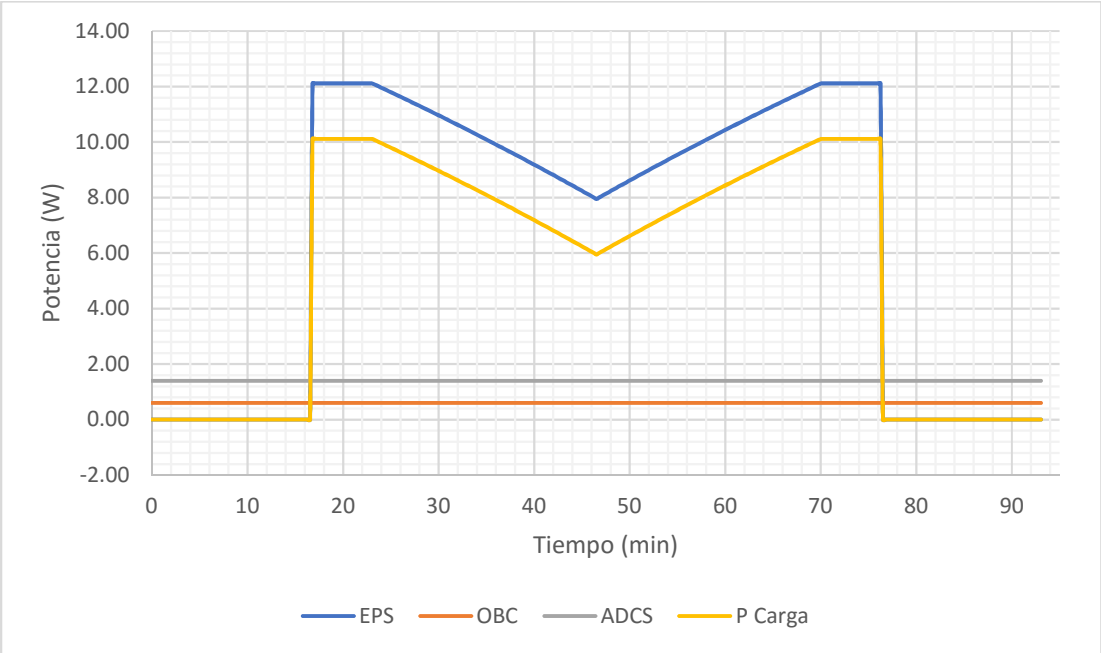
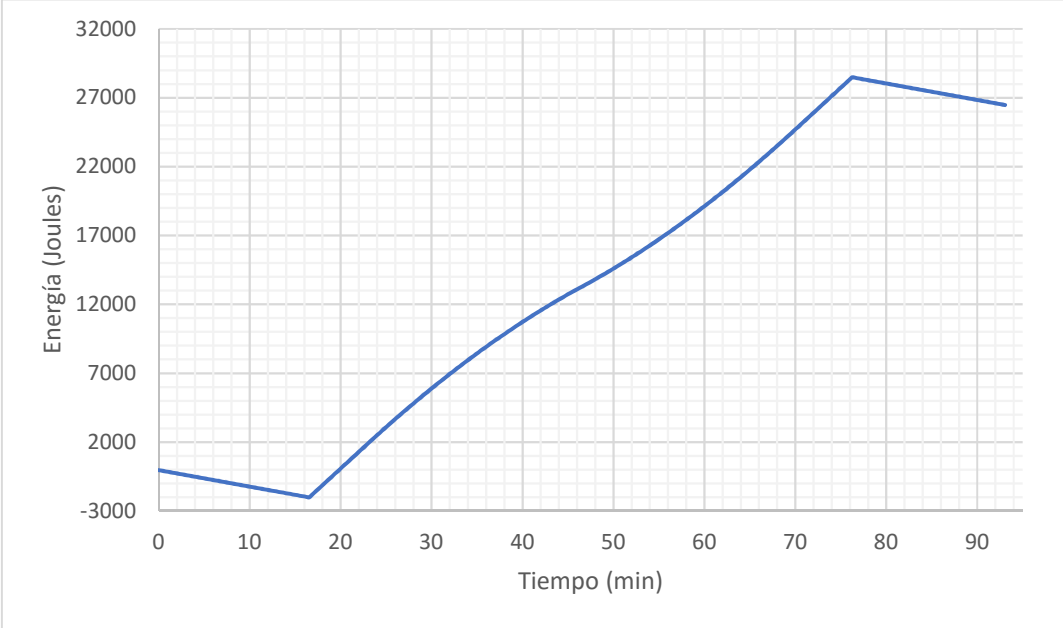


Figura 73. Almacenamiento de energía a lo largo de la órbita, escenario A, modelo 5PS.



d. Potencia generada vs. potencia consumida, Escenario B

Figura 74. Potencias a lo largo de la órbita escenario B, modelo 5PS.

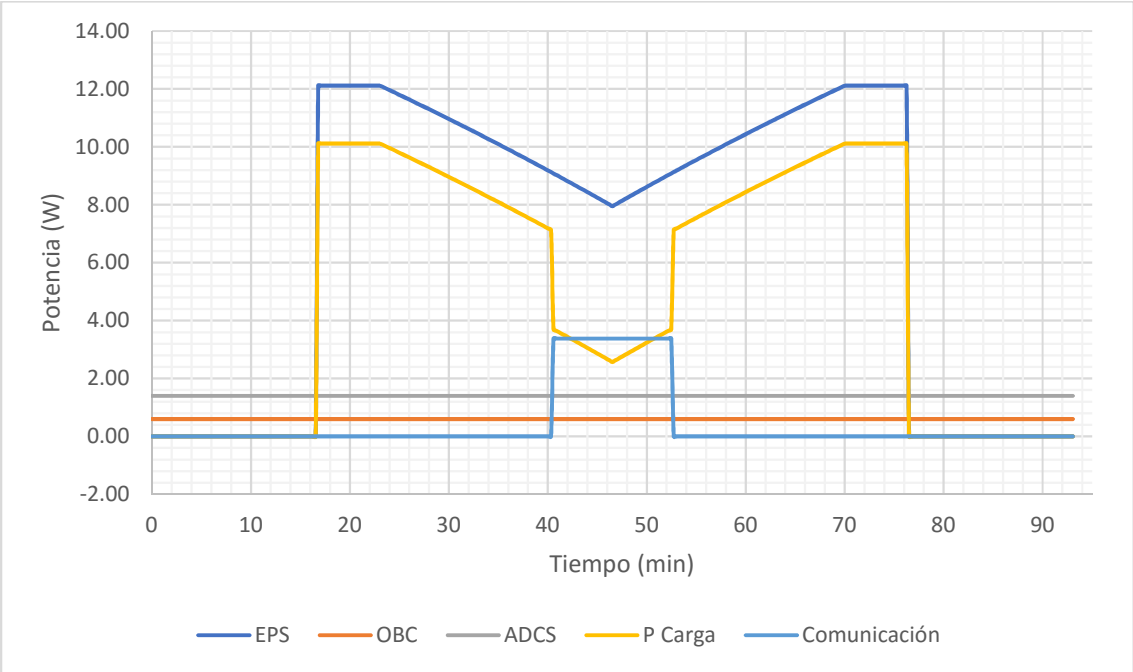
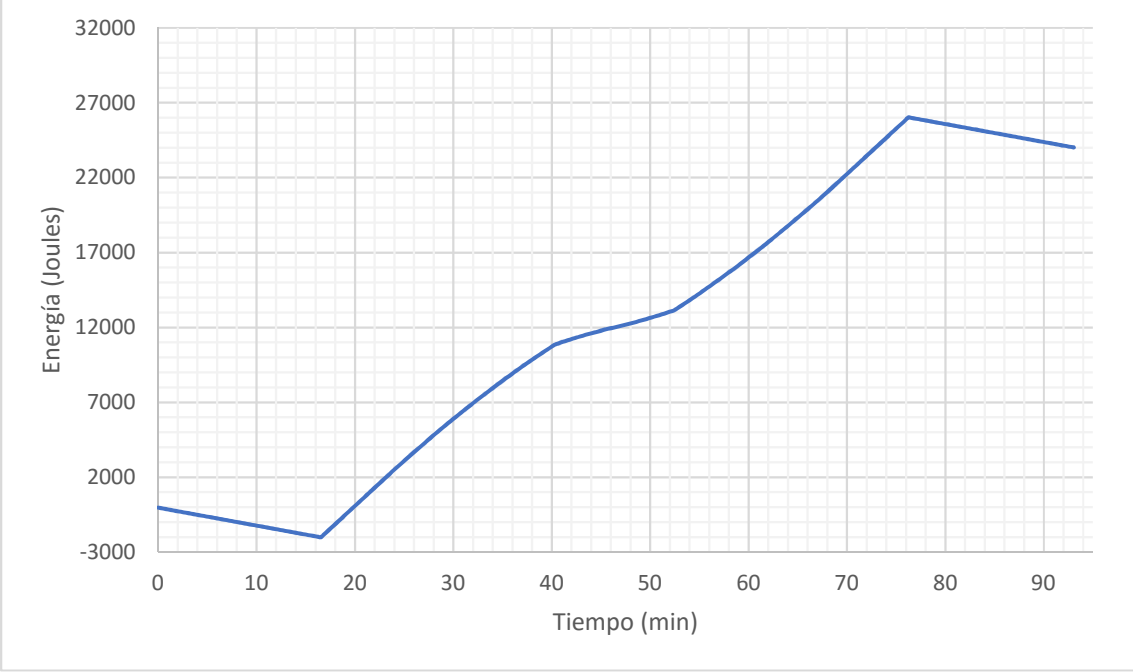


Figura 75. Almacenamiento de energía a lo largo de la órbita, escenario B, modelo 5PS.



e. Potencia generada vs. potencia consumida, Escenario C

Figura 76. Potencias a lo largo de la órbita escenario C, modelo 5PS.

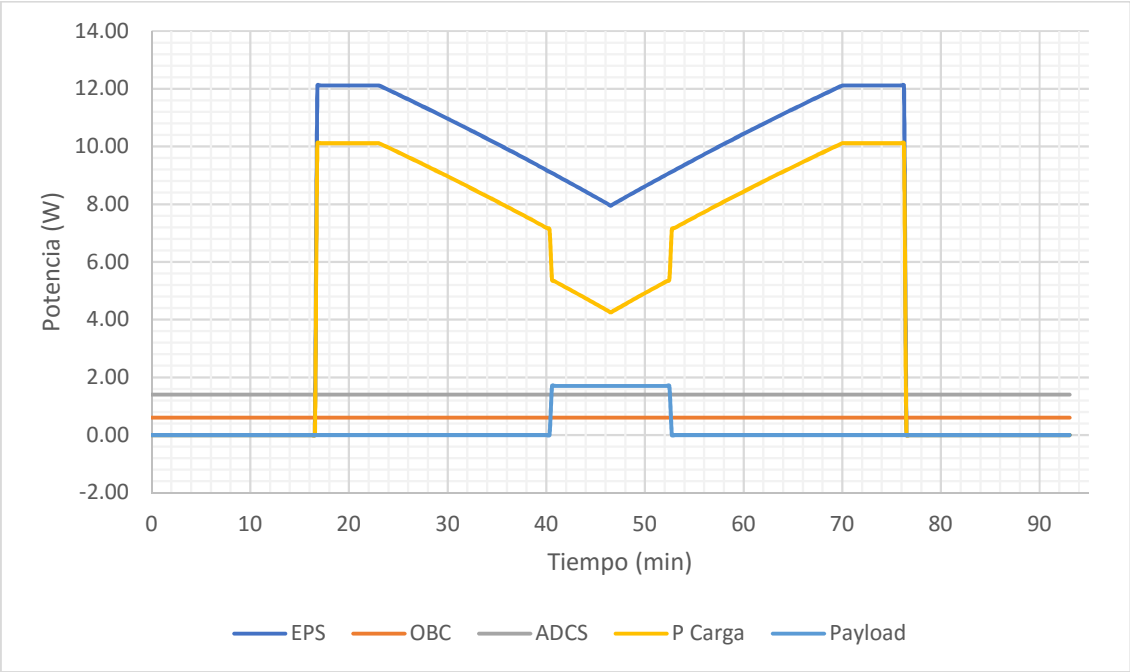
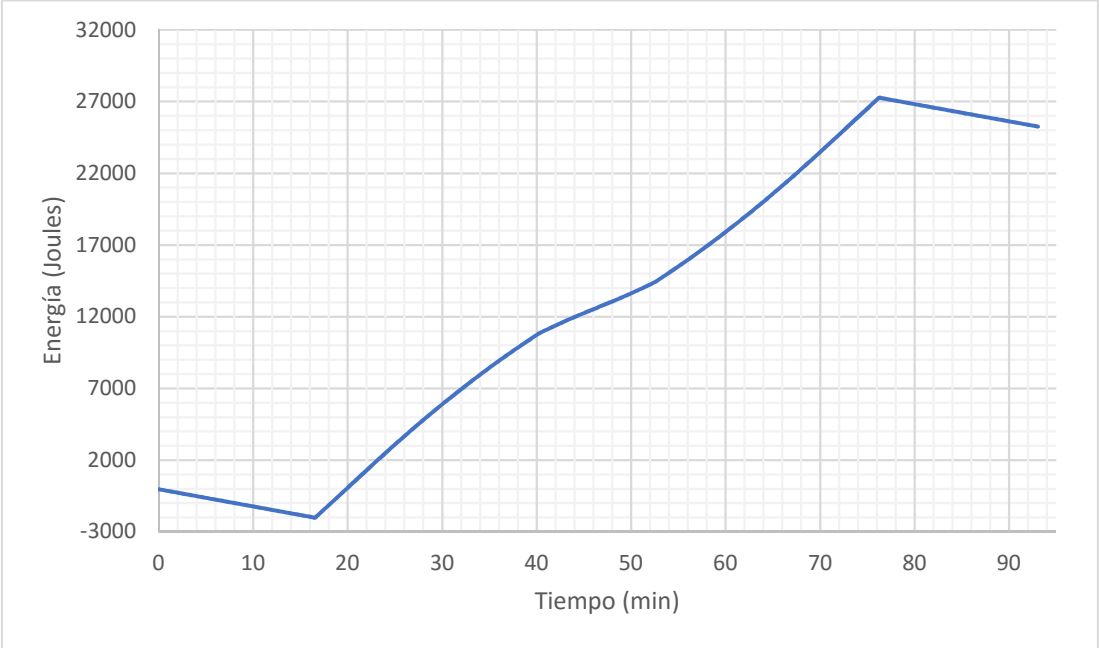


Figura 77. Almacenamiento de energía a lo largo de la órbita, escenario C, modelo 5PS.



f. Energía almacenada al final de cada escenario

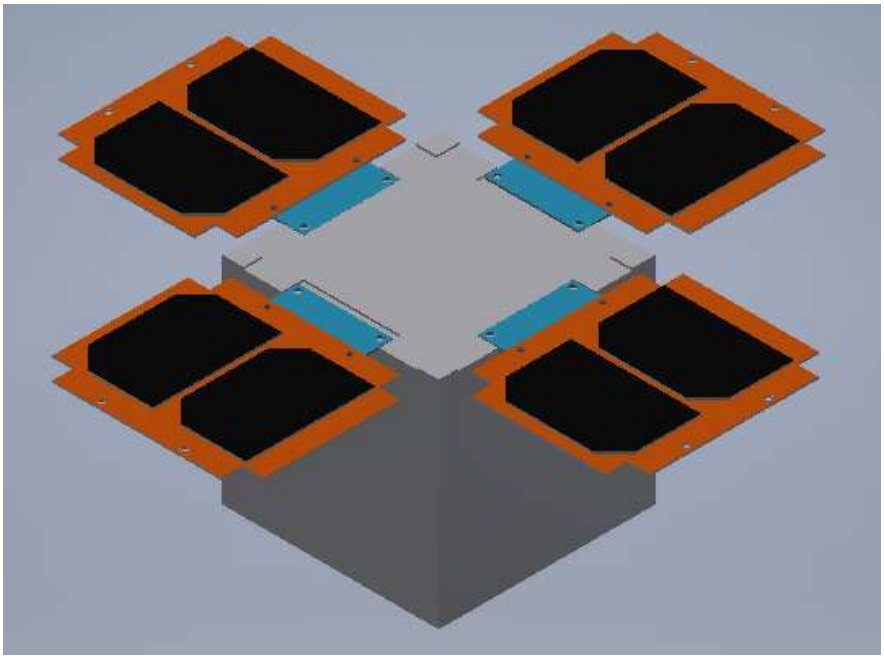
Cuadro 50. Diferencia de energía almacenada al final de cada escenario, modelo de 5PS.

Escenario	Energía (Joules)	Energía (mAh)
A	+ 26,474	+ 1,751
B	+ 24,015	+ 1,588
C	+ 25,258	+ 1,671

5. Modelo con cuatro paneles solares

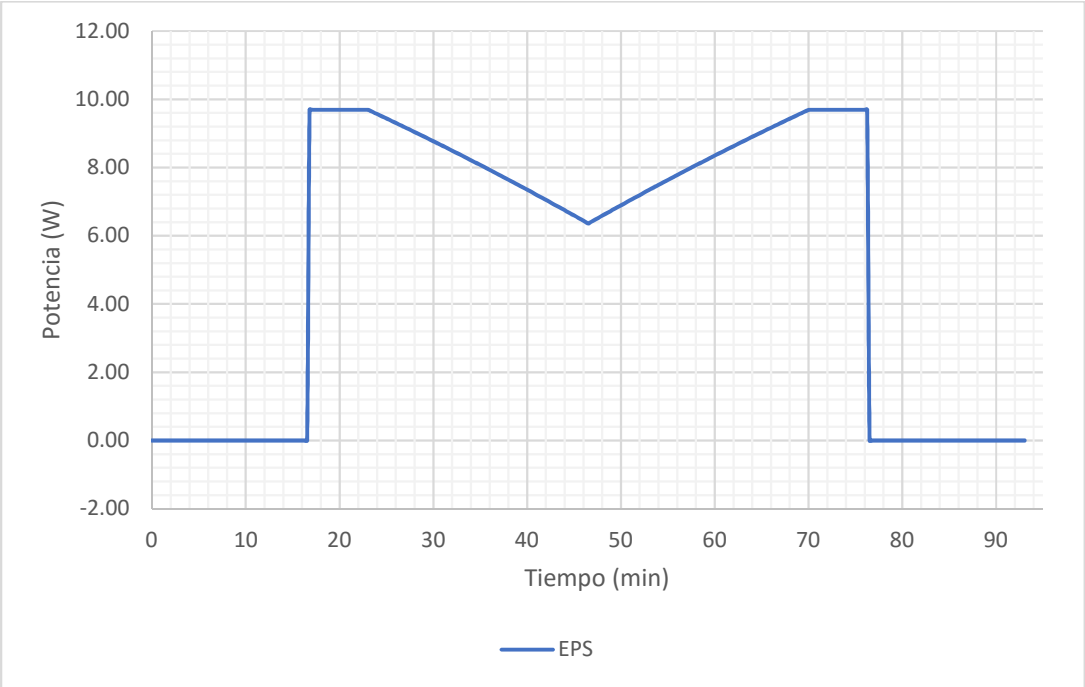
a. Aspecto físico

Figura 78. Modelo 4PS, con cuatro paneles solares.



b. Potencia generada

Figura 79. Potencia a lo largo de la órbita, Modelo 4PS.



c. Potencia generada vs. potencia consumida, Escenario A

Figura 80. Potencias a lo largo de la órbita escenario A, Modelo 4PS.

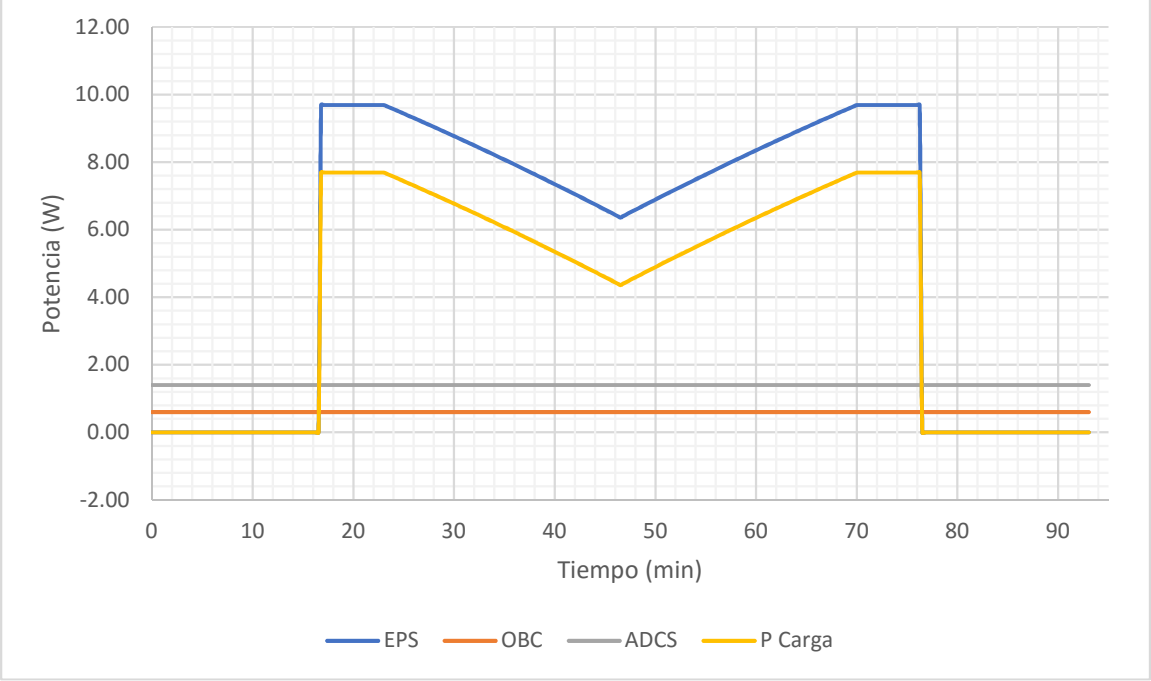
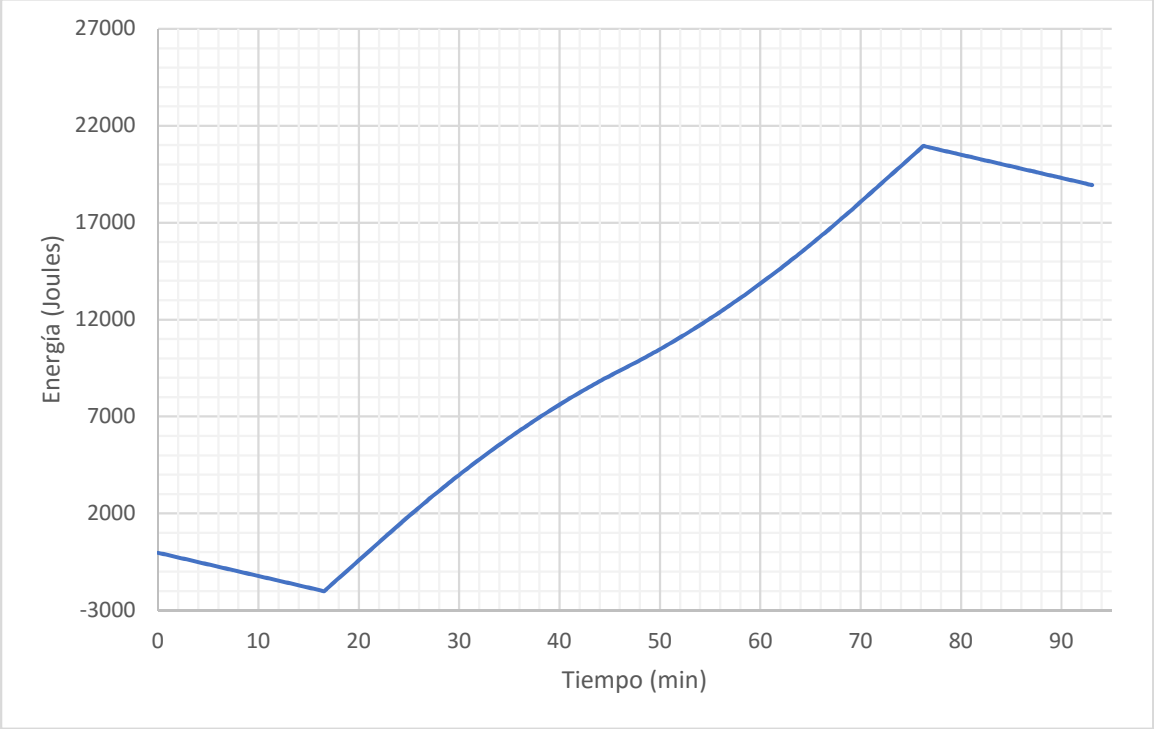


Figura 81. Almacenamiento de energía a lo largo de la órbita, escenario A, modelo 4PS.



d. Potencia generada vs. potencia consumida, Escenario B

Figura 82. Potencias a lo largo de la órbita escenario B, Modelo 4PS.

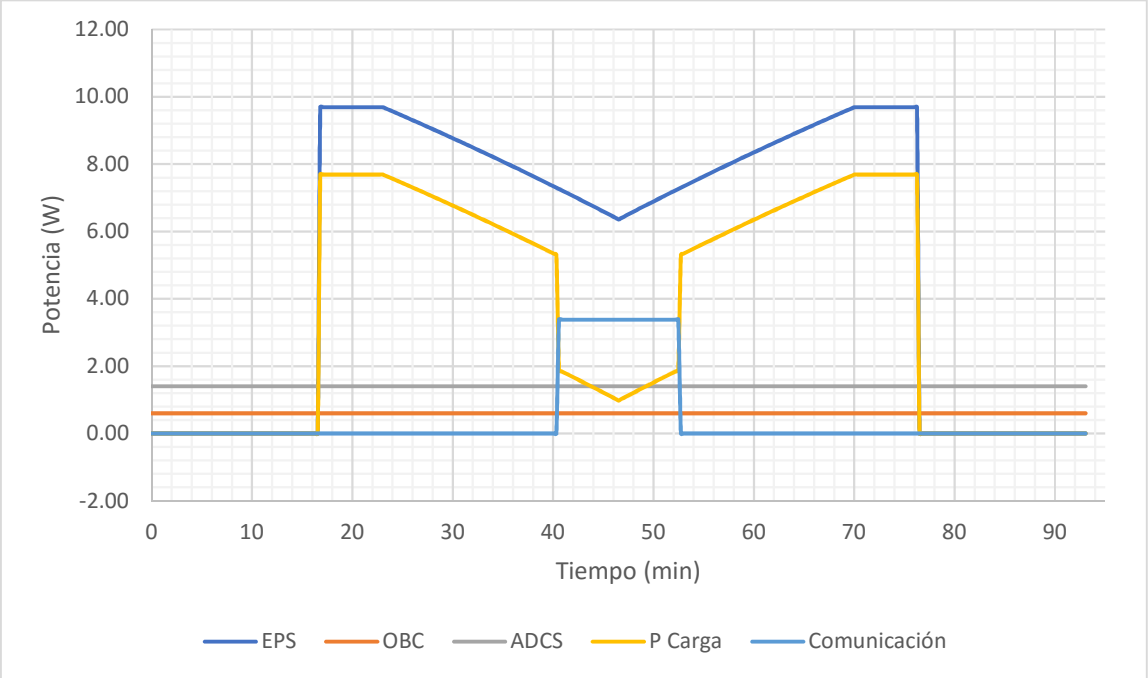
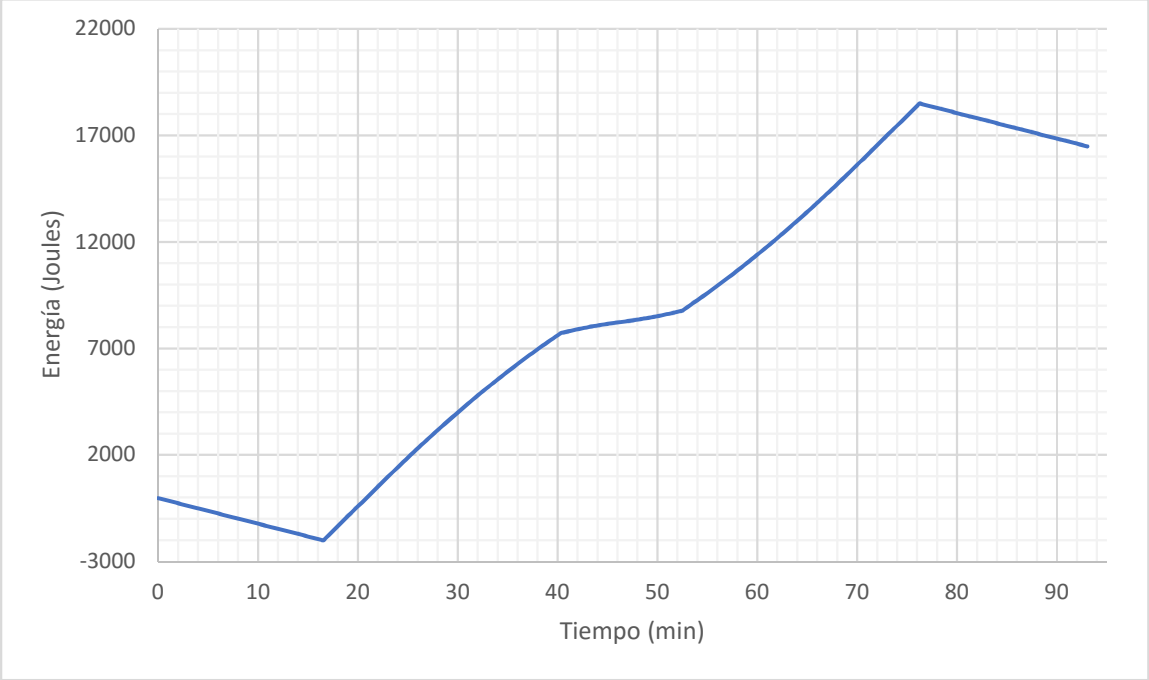


Figura 83. Almacenamiento de energía a lo largo de la órbita, escenario B, modelo 4PS.



e. Potencia generada vs. potencia consumida, Escenario C

Figura 84. Potencias a lo largo de la órbita escenario C, Modelo 4PS.

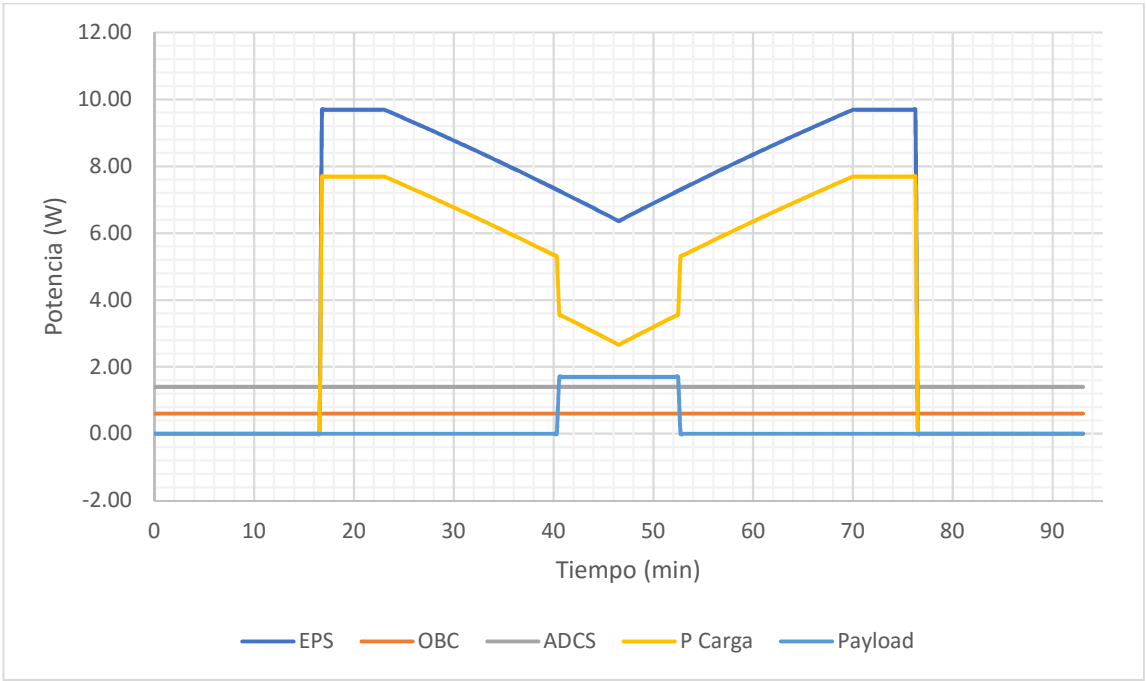
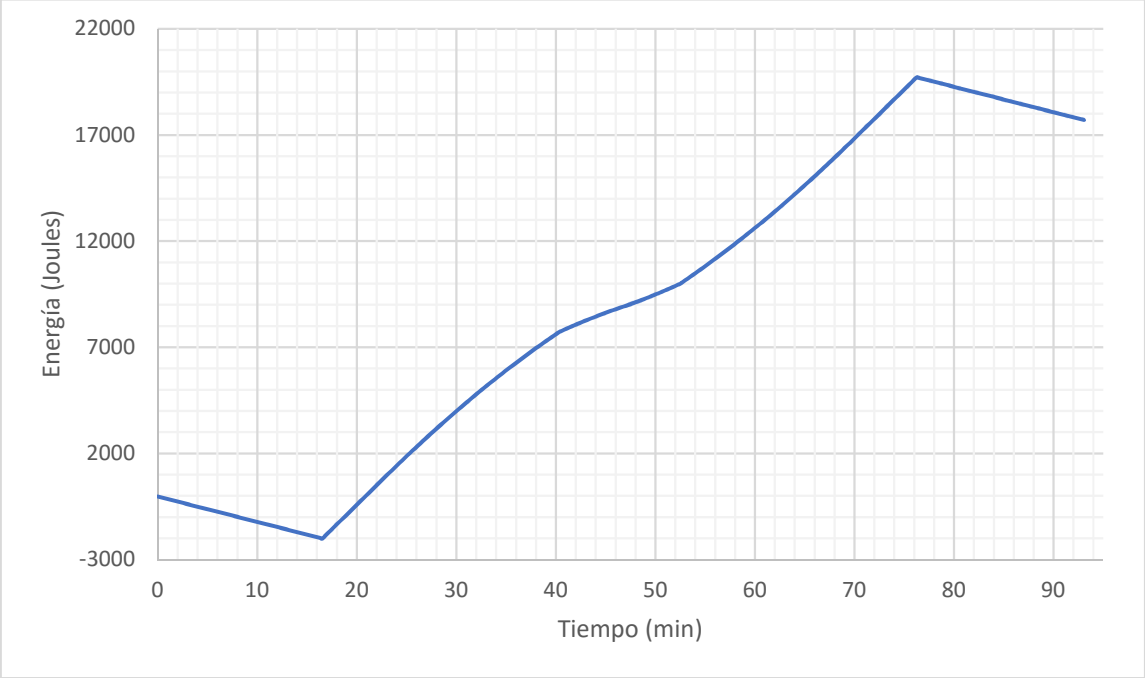


Figura 85. Almacenamiento de energía a lo largo de la órbita, escenario C, modelo 4PS.



f. Energía almacenada al final de cada escenario

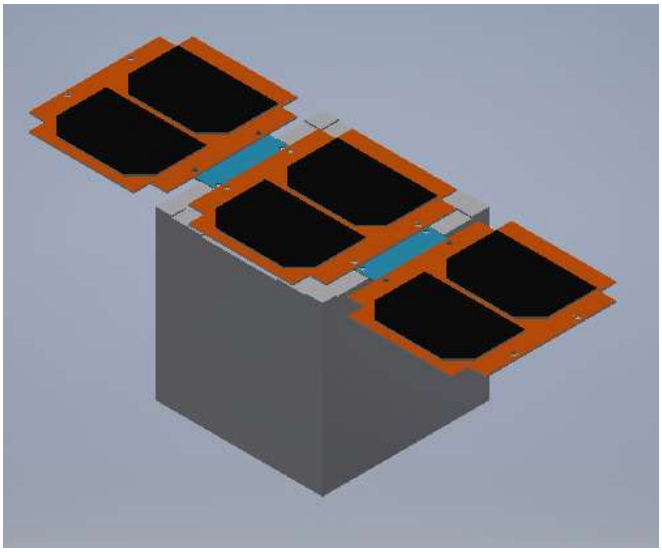
Cuadro 51. Diferencia de energía almacenada al final de cada escenario, modelo con 4PS.

Escenario	Energía (Joules)	Energía (mAh)
A	+ 18,944	+ 1,253
B	+ 16,485	+ 1,090
C	+ 17,707	+ 1,171

6. Modelo con tres paneles solares

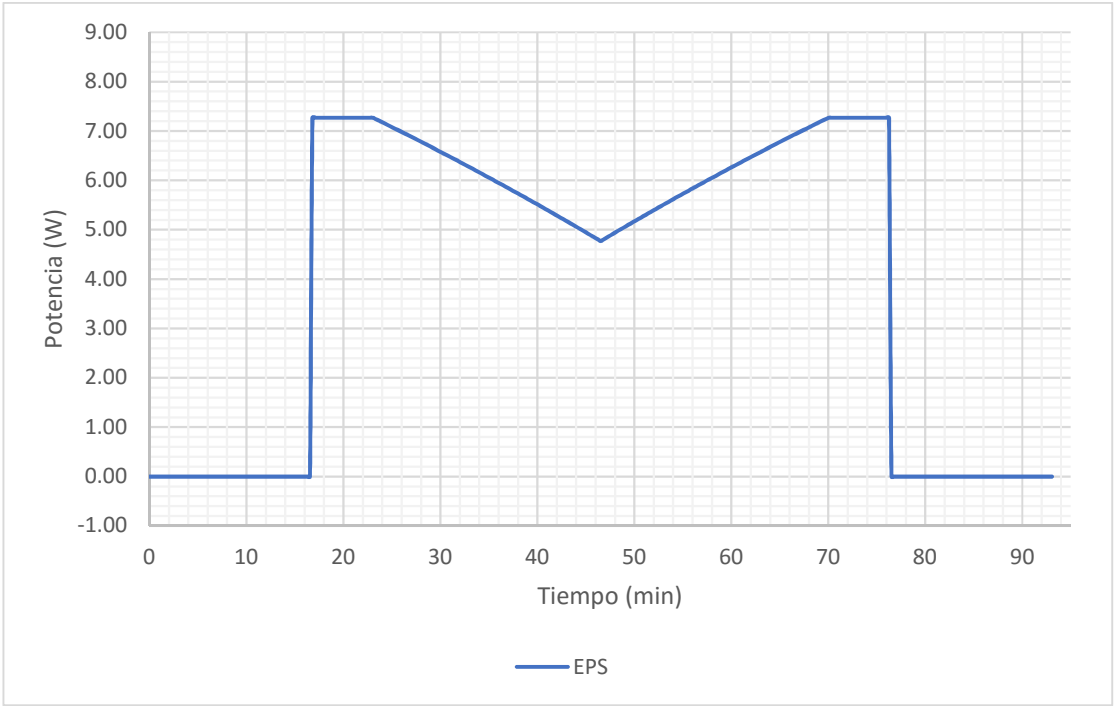
a. Aspecto físico

Figura 86. Modelo 3PS, con dos paneles solares.



b. Potencia generada

Figura 87. Potencia a lo largo de la órbita, Modelo 3PS.



c. Potencia generada vs. potencia consumida, Escenario A

Figura 88. Potencias a lo largo de la órbita escenario A, Modelo 3PS.

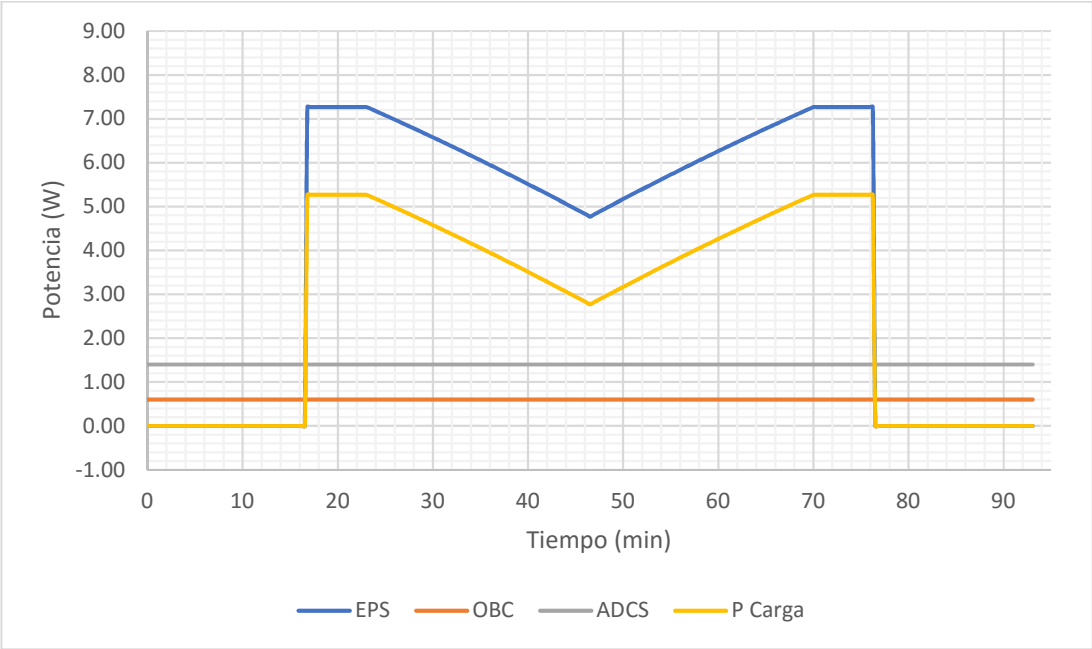
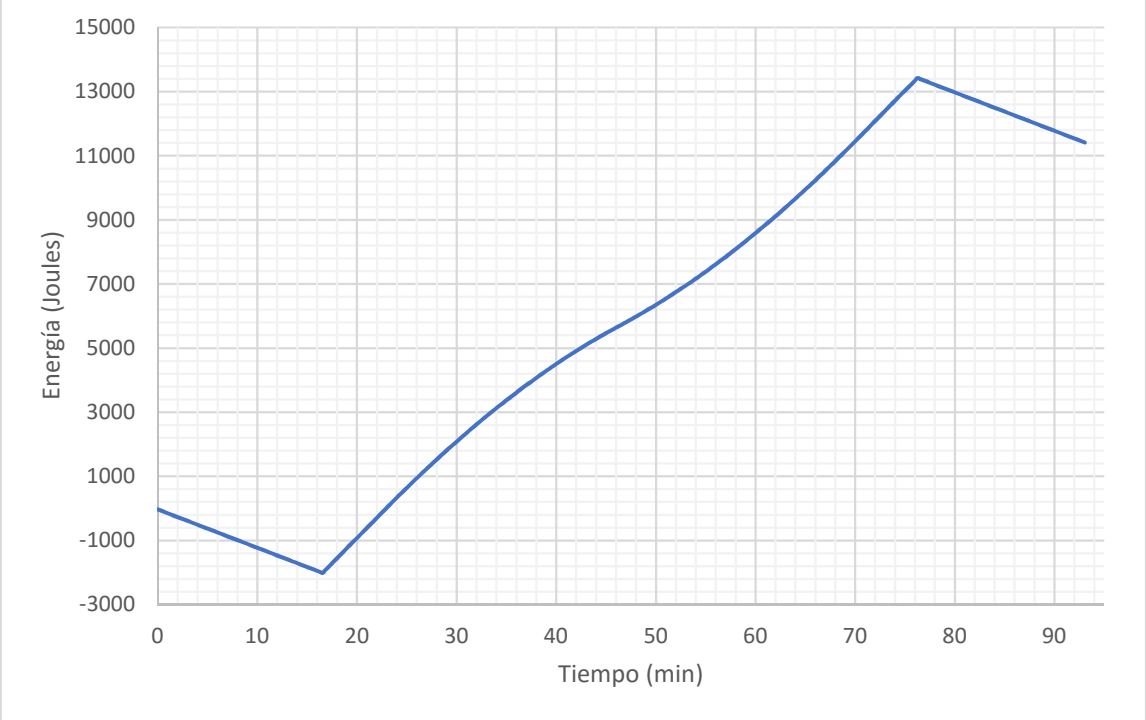


Figura 89. Almacenamiento de energía a lo largo de la órbita, escenario A, modelo 3PS.



d. Potencia generada vs. potencia consumida, Escenario B

Figura 90. Potencias a lo largo de la órbita escenario B, Modelo 3PS.

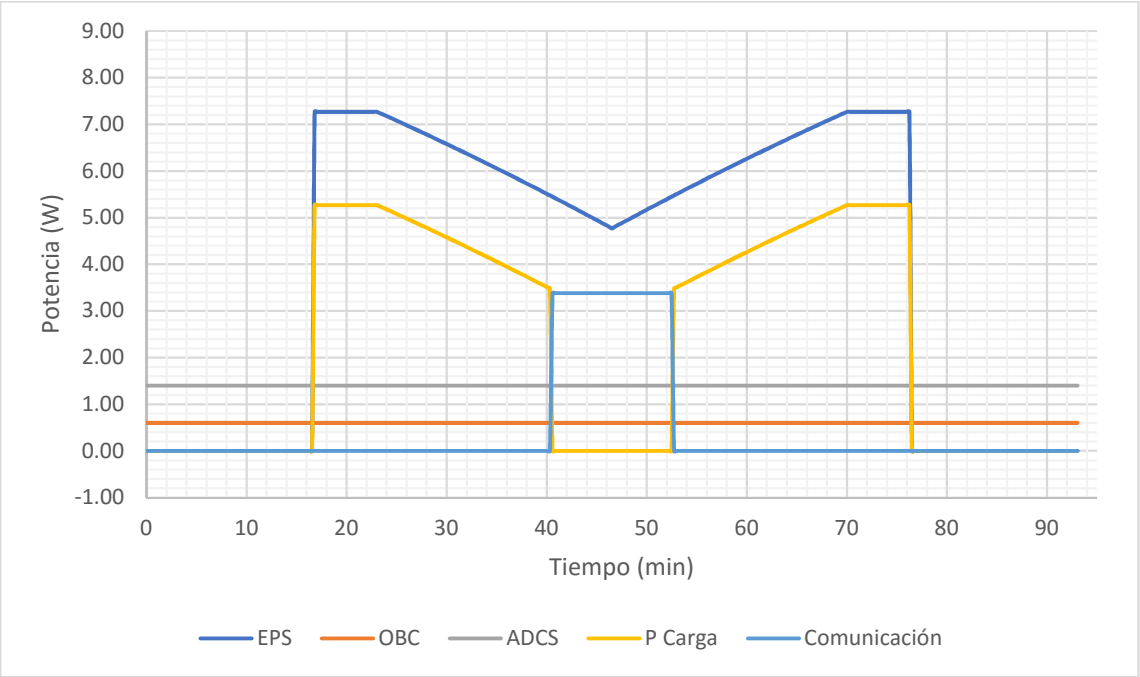
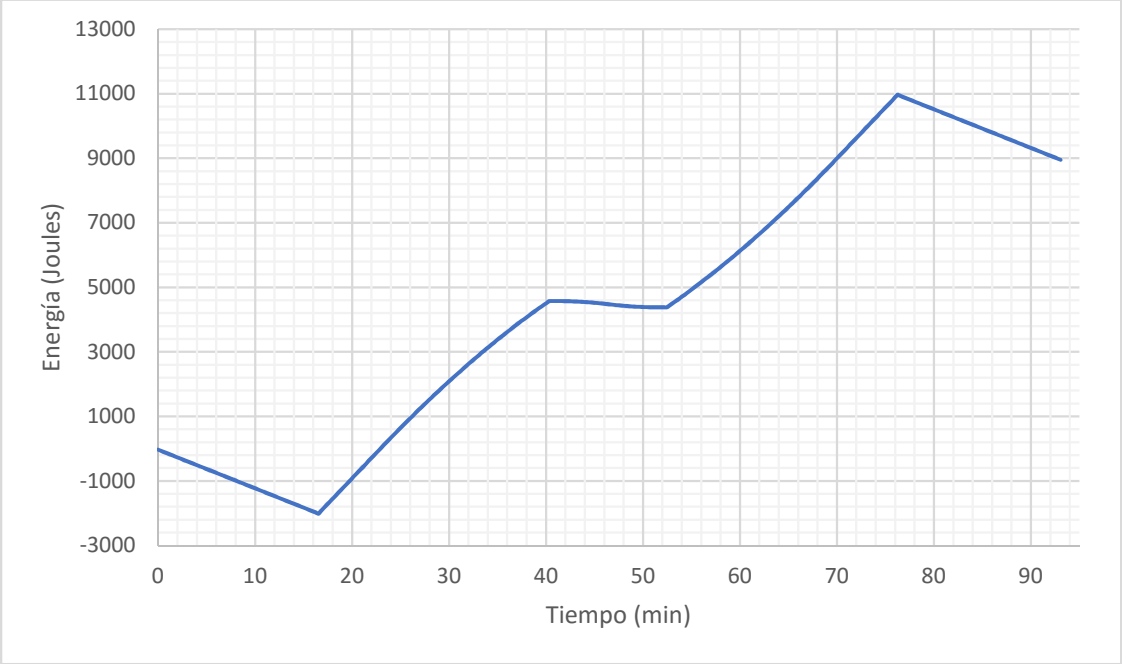


Figura 91. Almacenamiento de energía a lo largo de la órbita, escenario B, modelo 3PS.



e. Potencia generada vs. potencia consumida, Escenario C

Figura 92. Potencias a lo largo de la órbita escenario C, Modelo 3PS.

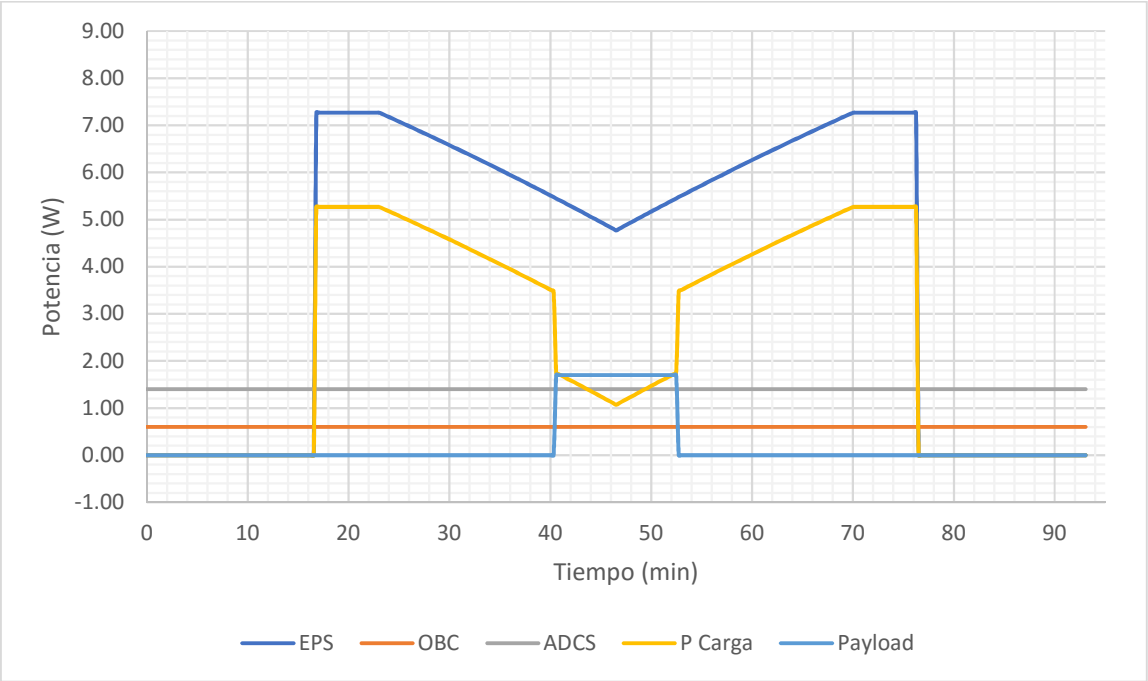
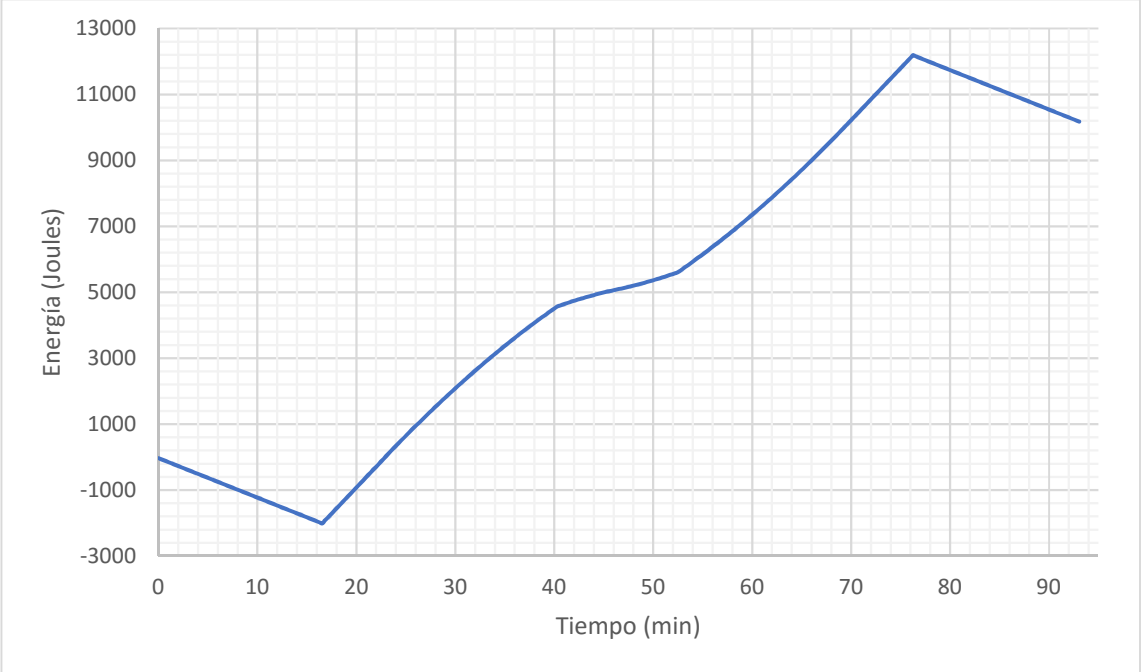


Figura 93. Almacenamiento de energía a lo largo de la órbita, escenario C, modelo 3PS.



f. Energía almacenada al final de cada escenario

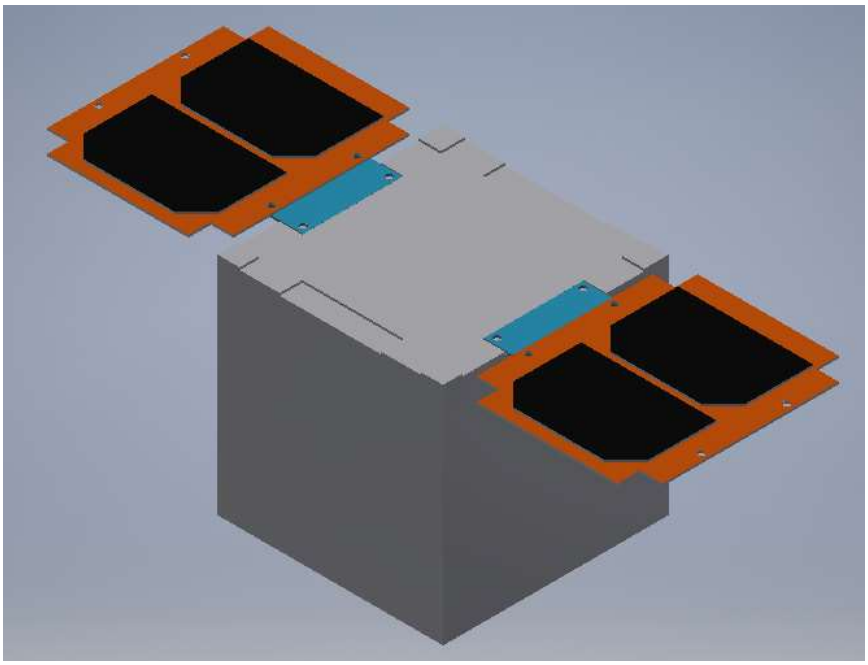
Cuadro 52. Diferencia de energía almacenada al final de cada escenario, modelo con 3PS.

Escenario	Energía (Joules)	Energía (mAh)
A	+ 11,414	+ 755
B	+ 8,955	+ 592
C	+ 10,177	+ 673

7. Modelo con dos paneles solares

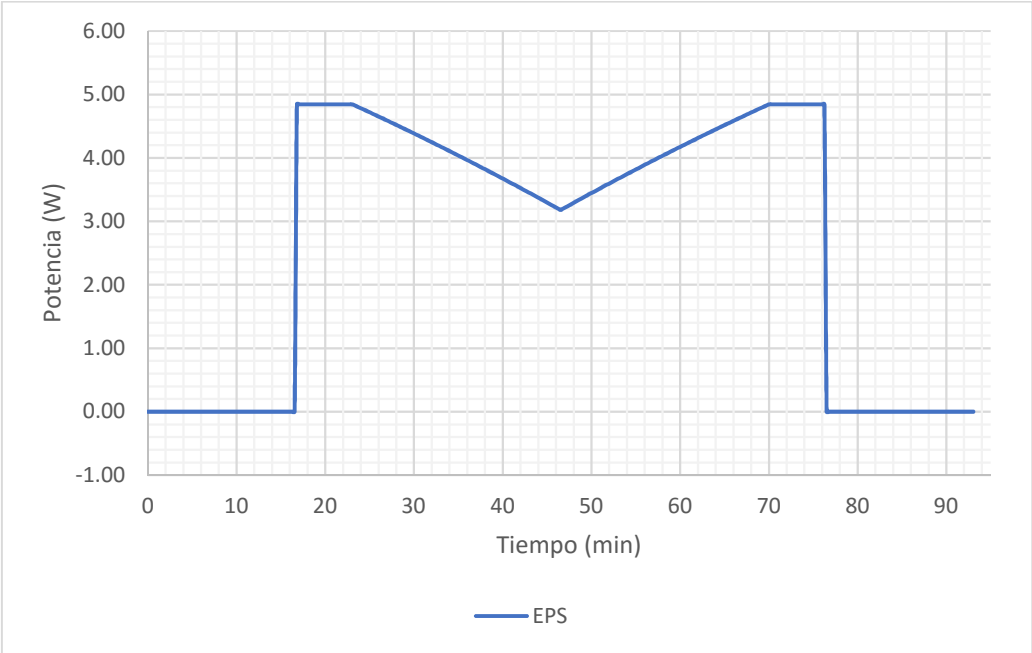
a. Aspecto físico

Figura 94. Modelo 2PS, con dos paneles solares.



b. Potencia generada

Figura 95. Potencia a lo largo de la órbita, Modelo 2PS.



c. Potencia generada vs. potencia consumida, Escenario A

Figura 96. Potencias a lo largo de la órbita escenario A, Modelo 2PS.

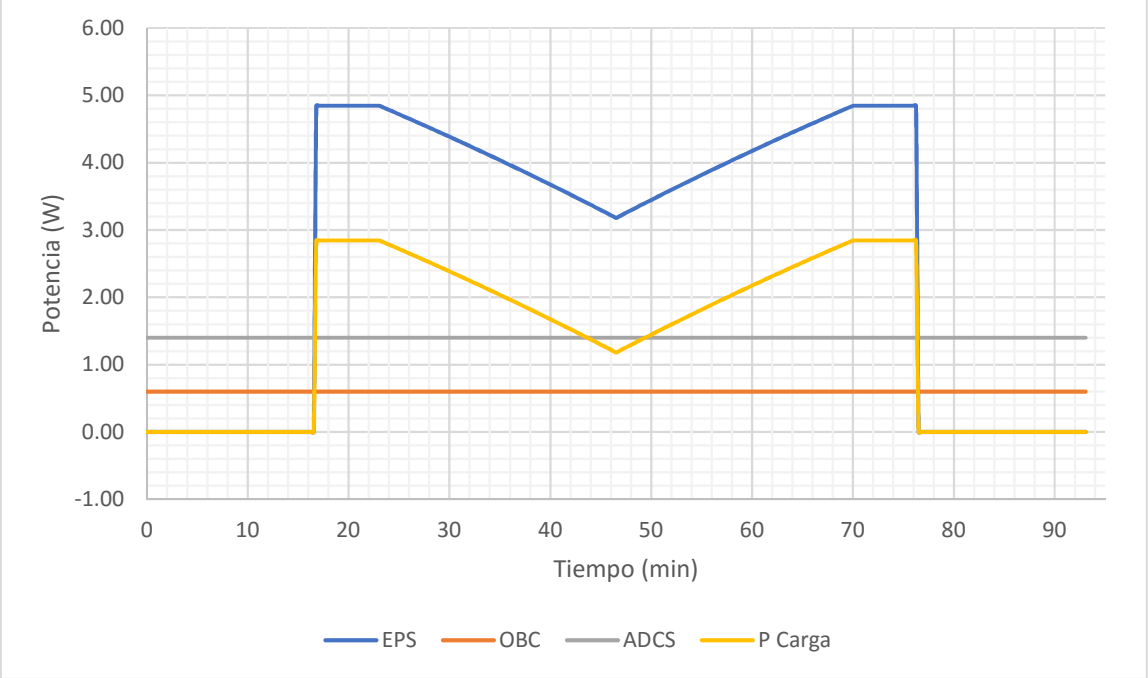
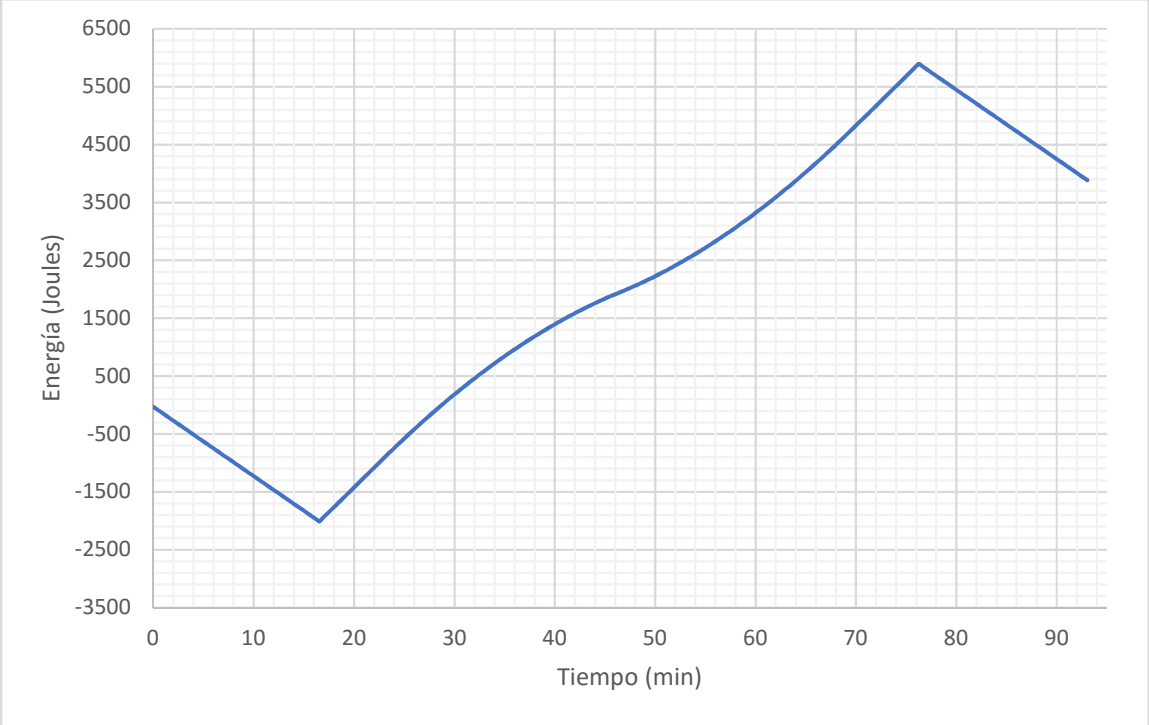


Figura 97. Almacenamiento de energía a lo largo de la órbita, escenario A, modelo 2PS.



d. Potencia generada vs. potencia consumida, Escenario B

Figura 98. Potencias a lo largo de la órbita escenario B, Modelo 2PS.

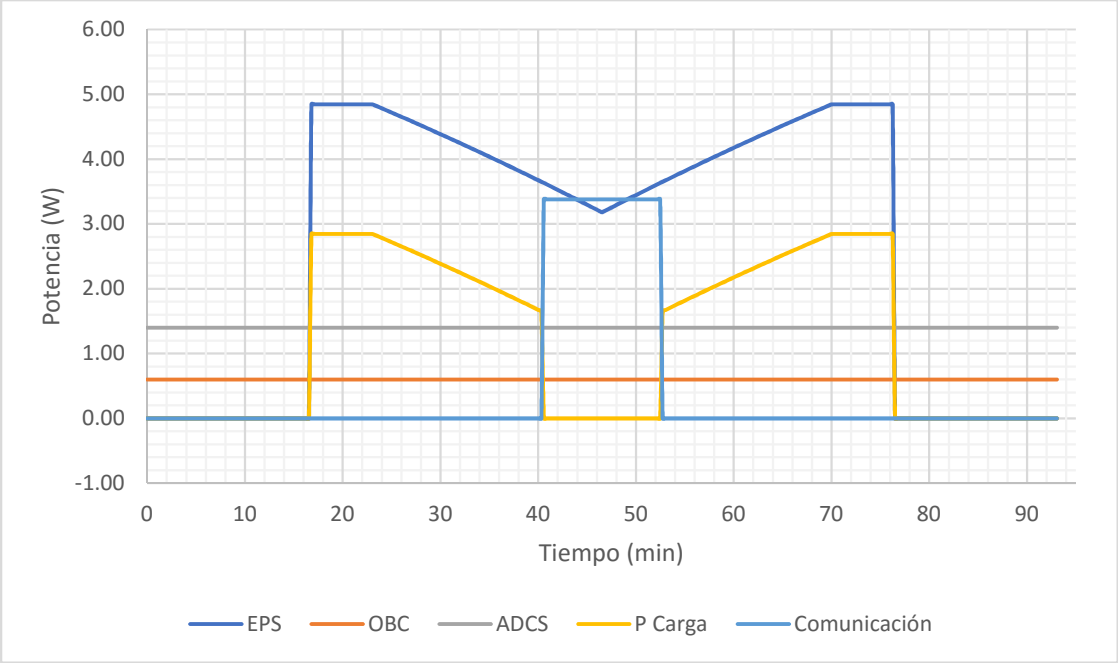
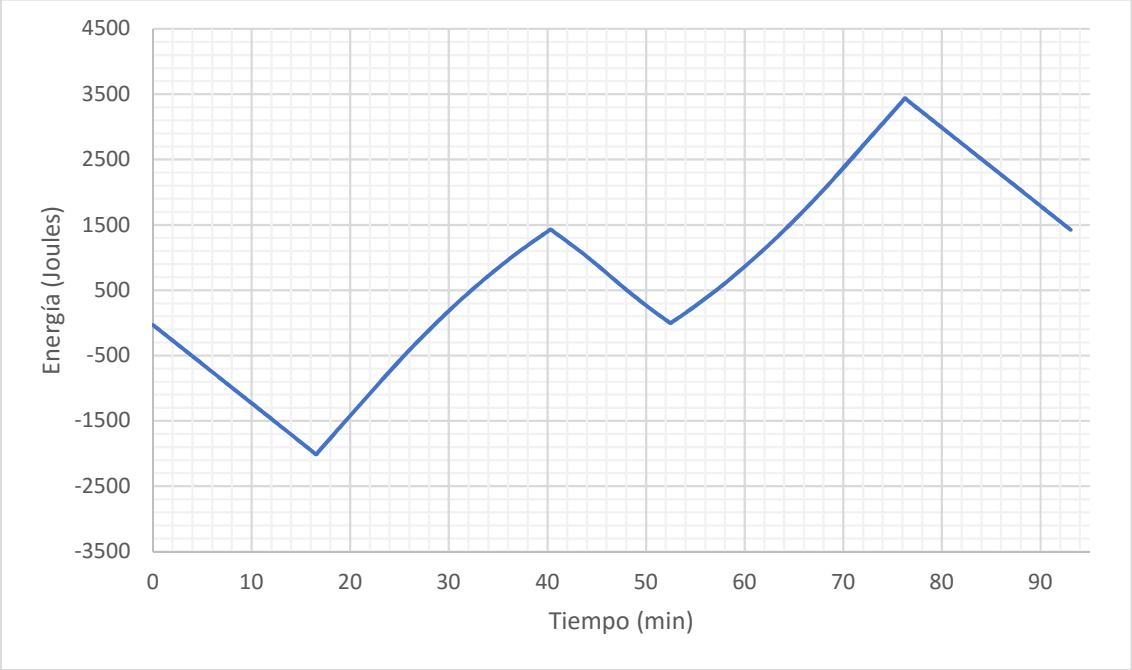


Figura 99. Almacenamiento de energía a lo largo de la órbita, escenario B, modelo 2PS.



e. Potencia generada vs. potencia consumida, Escenario C

Figura 100. Potencias a lo largo de la órbita escenario C, Modelo 2PS.

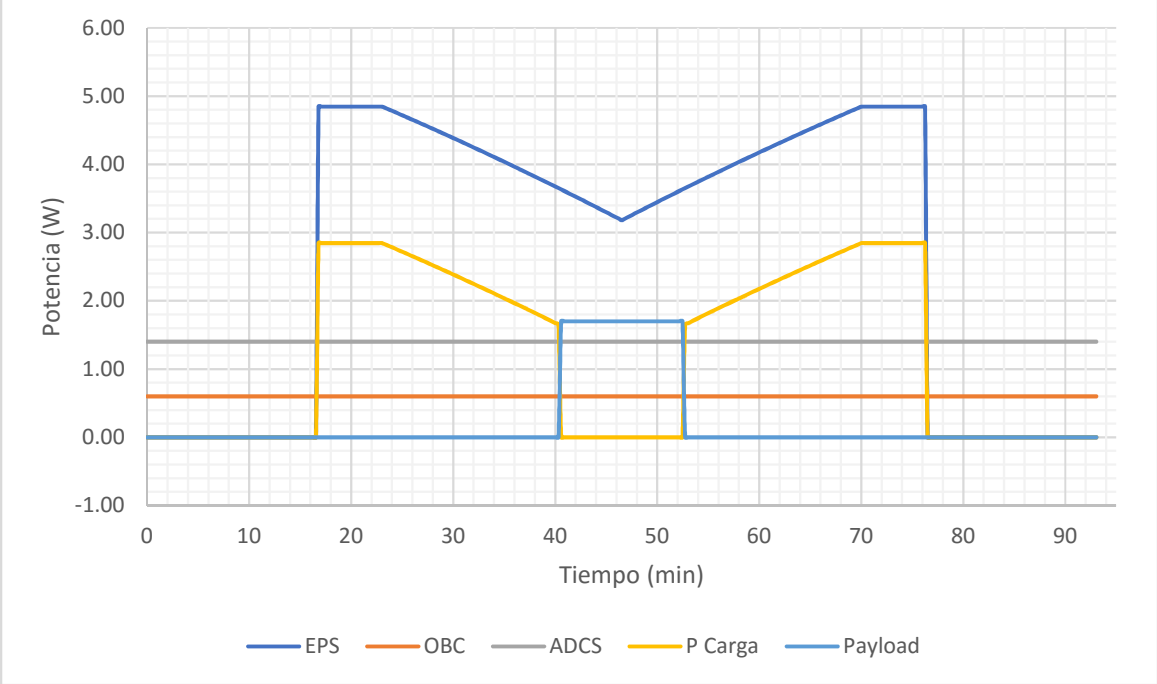
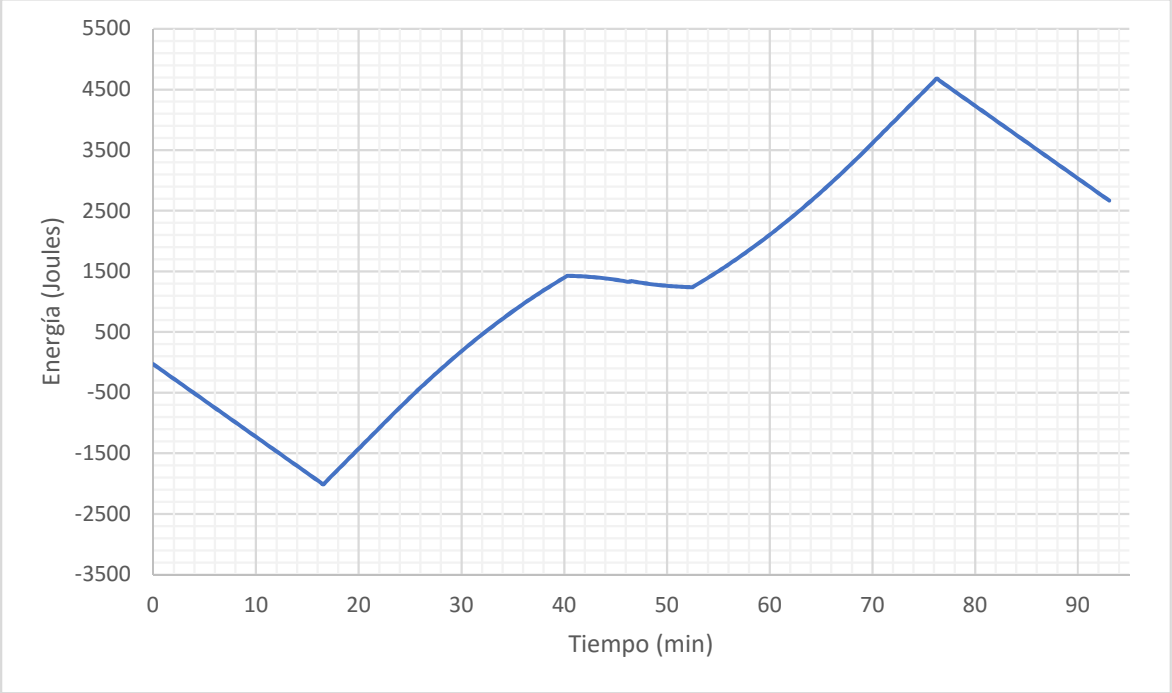


Figura 101. Almacenamiento de energía a lo largo de la órbita, escenario C, modelo 2PS.



f. Energía almacenada al final de cada escenario

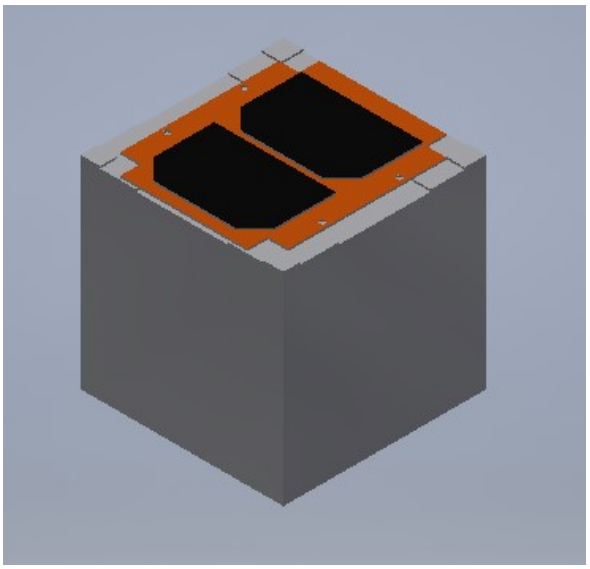
Cuadro 53. Diferencia de energía almacenada al final de cada escenario, modelo con 2PS.

Escenario	Energía (Joules)	Energía (mAh)
A	+ 3,884	+ 257
B	+ 1,425	+ 94
C	+ 2,668	+ 176

8. Modelo con un panel solar

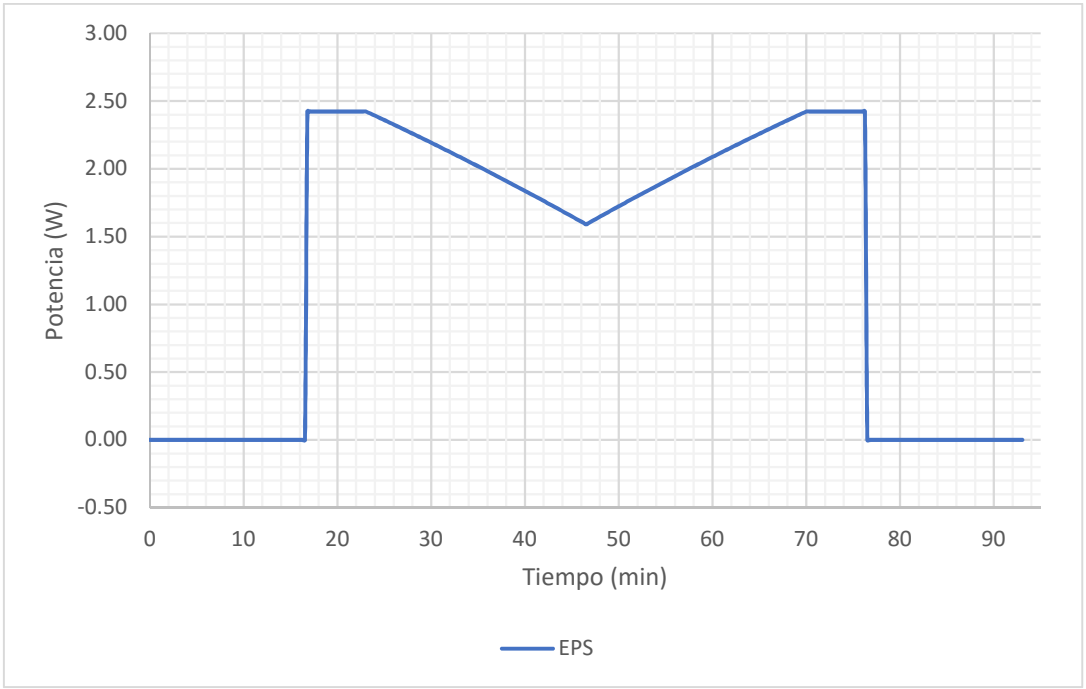
a. Aspecto físico

Figura 102. Modelo 1PS, con dos paneles solares.



b. Potencia generada

Figura 103. Potencia a lo largo de la órbita, Modelo 1PS.



c. Potencia generada vs. potencia consumida, Escenario A

Figura 104. Potencias a lo largo de la órbita escenario A, Modelo 1PS.

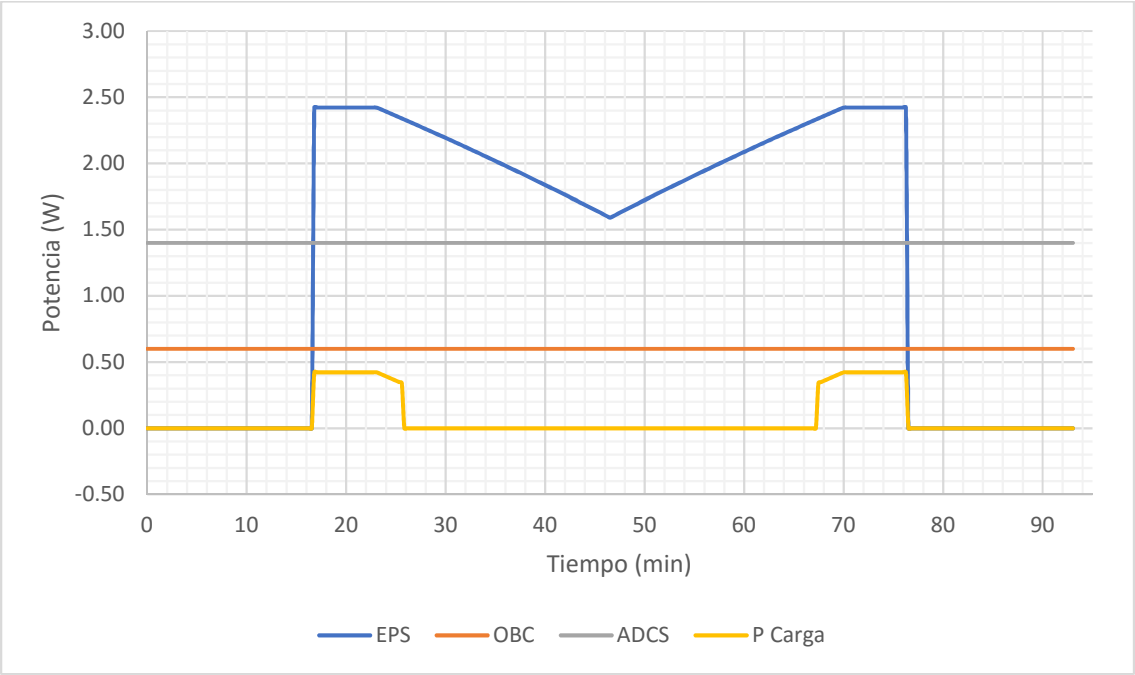
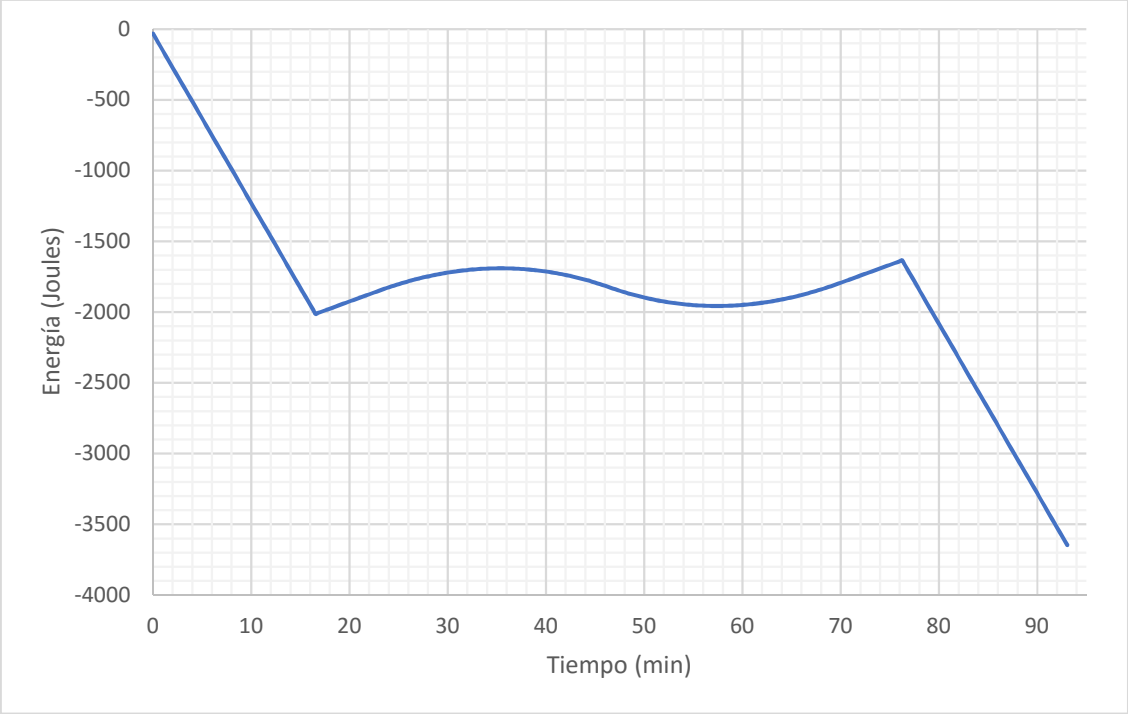


Figura 105. Almacenamiento de energía a lo largo de la órbita, escenario A, modelo 1PS.



d. Potencia generada vs. potencia consumida, Escenario B

Figura 106. Potencias a lo largo de la órbita escenario B, Modelo 1PS.

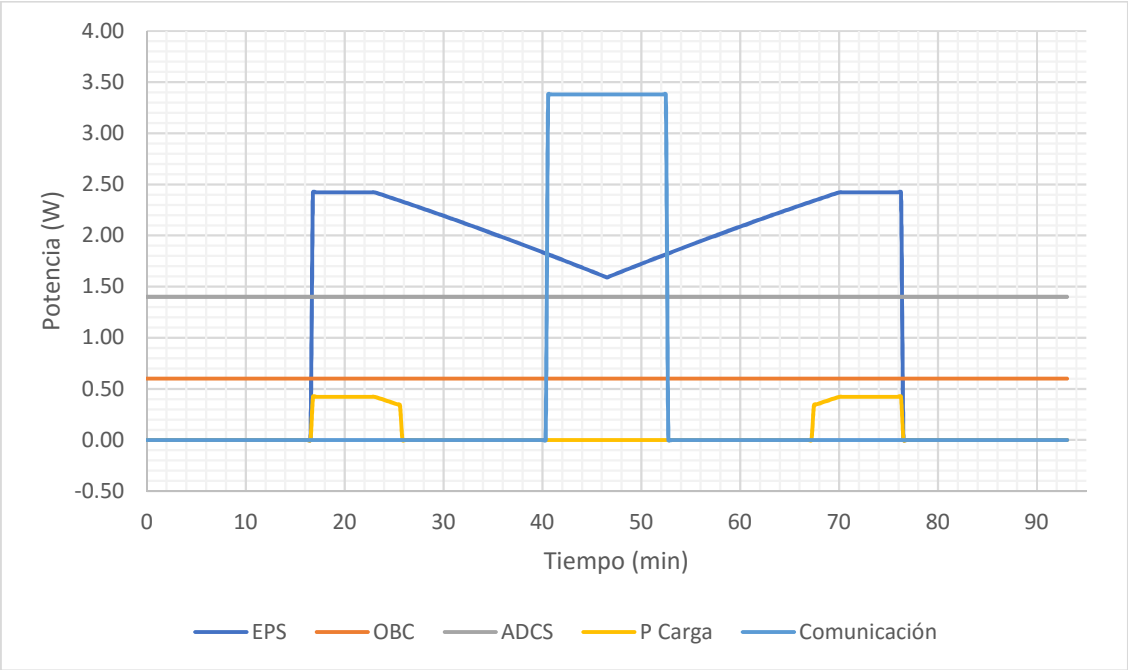
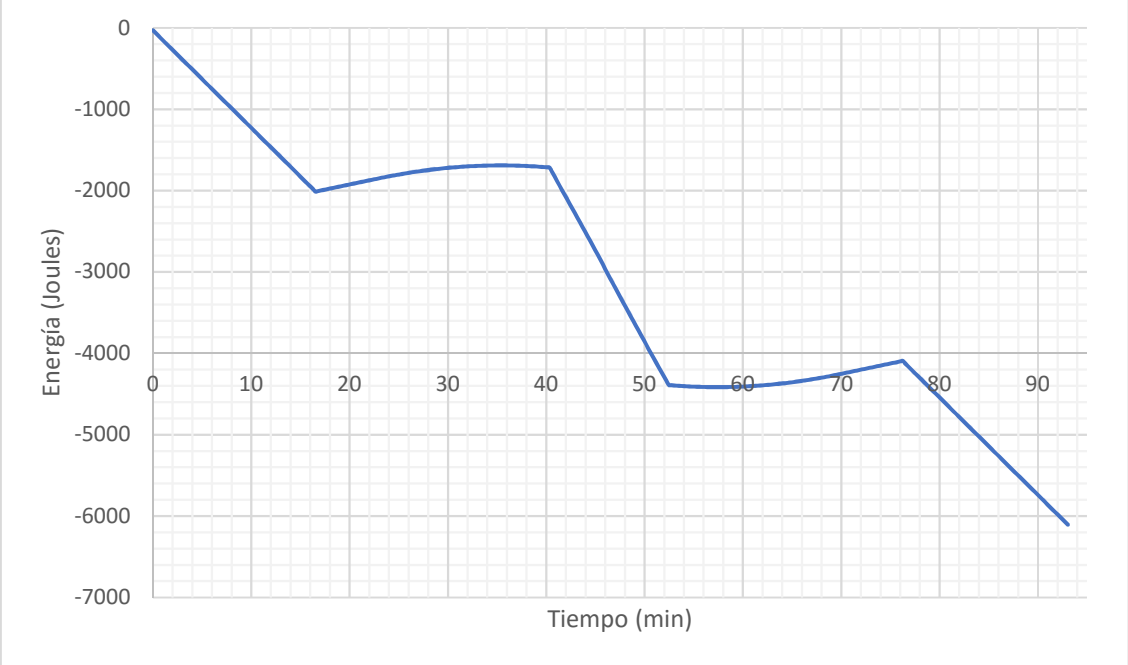


Figura 107. Almacenamiento de energía a lo largo de la órbita, escenario B, modelo 1PS.



e. Potencia generada vs. potencia consumida, Escenario C

Figura 108. Potencias a lo largo de la órbita escenario C, Modelo 1PS.

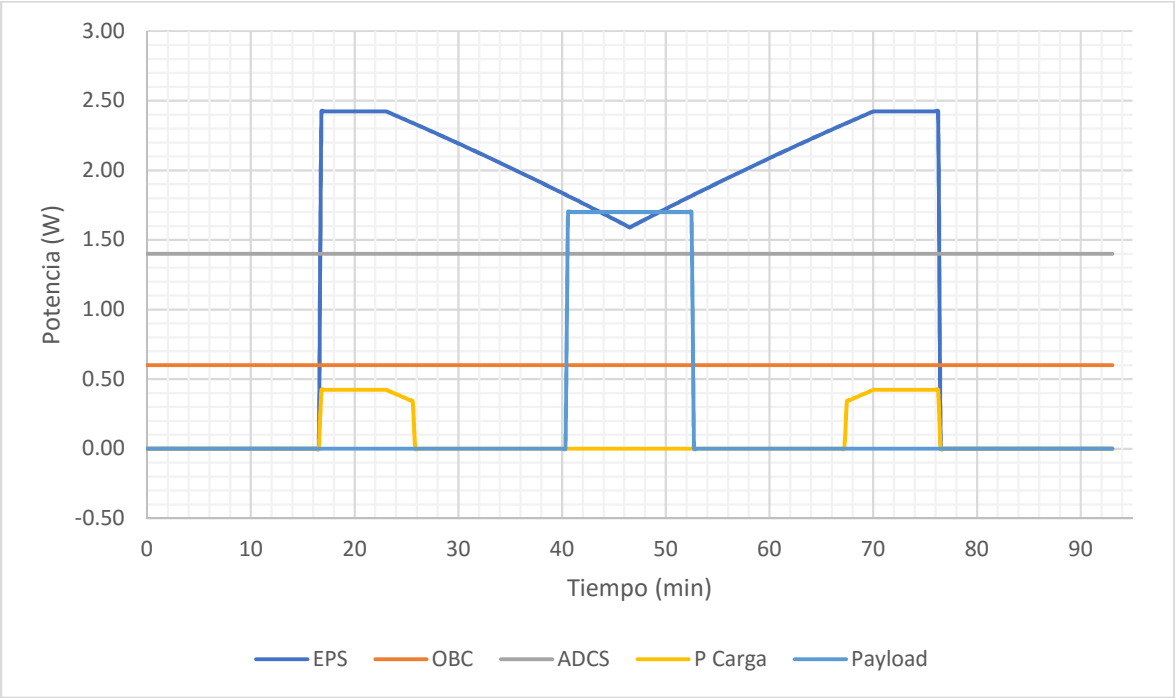
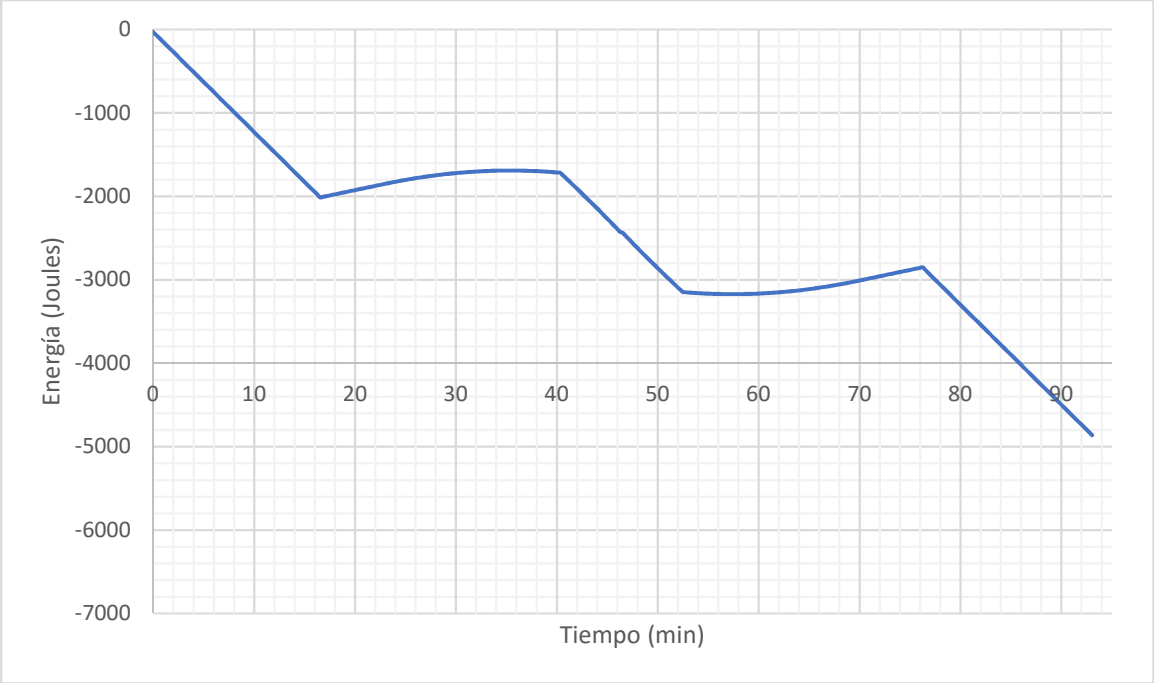


Figura 109. Almacenamiento de energía a lo largo de la órbita, escenario C, modelo 1PS.



f. Energía almacenada al final de cada escenario

Cuadro 54. Diferencia de energía almacenada al final de cada escenario, modelo con 1PS.

Escenario	Energía (Joules)	Energía (mAh)
A	- 3,646	- 241
B	- 6,106	- 404
C	- 4862	- 322

9. Capacidad de almacenamiento de la batería

Cuadro 55. Energía consumida en la etapa de sombra para cada modelo.

Modelo	Periodo De Consumo > Producción (min)		Energía (Joules)	Total de Energía (mAh)*
	Inicio	Fin		
5PS	0	16.54	2,012.4	302.2
	76.25	93.05	2,012.4	
4PS	0	16.54	2,012.4	302.2
	76.25	93.05	2,012.4	
3PS	0	16.54	2,012.4	302.2
	76.25	93.05	2,012.4	
2PS	0	16.54	2,012.4	302.2
	76.25	93.05	2,012.4	
1PS	0	16.54	2,012.4	302.2
	76.25	93.05	2,012.4	

*Se calculó para una batería con 3.7 V de voltaje nominal.

Cuadro 56. Corriente máxima de carga para una batería con 4.2 V como voltaje de carga.

Modelo	Potencia Máxima de Carga (W)	Corriente Máxima de Carga (A)
5PS	12.11	2.88
4PS	9.69	2.31
3PS	7.27	1.73
2PS	4.85	1.15
1PS	2.42	0.58

10. Resumen de resultados

Cuadro 57. Resumen comparativo de los resultados de todos los modelos.

Modelo	5PS	4PS	3PS	2PS	1PS
Potencia Máxima Teórica Producida (W)	12.11	9.69	7.27	4.85	2.42
Potencia Máxima de Carga para baterías (W)	12.11	9.69	7.27	4.85	2.42
*Corriente Máxima de Carga para baterías (A)	2.88	2.31	1.73	1.15	0.58
Energía consumida en periodo de sombra (J)	4,425	4,025	4,025	4,025	4,025
Diferencial de energía al final de Escenario A (J)	+ 26,474	+ 18,944	+ 11,414	+ 3,884	- 3,646
Diferencial de energía al final de Escenario B (J)	+ 24,015	+ 16,485	+ 8,955	+ 1,425	- 6,106
Diferencial de energía al final de Escenario C (J)	+ 25,258	+ 17,707	+ 10,177	+ 2,668	- 4,862
*Energía consumida en periodo de sombra (mAh)	302.2	302.2	302.2	302.2	302.2

*Se calculó para una batería con 4.2 V como voltaje de carga, y 3.7 V como voltaje nominal.

11. Análisis de resultados.

El primer paso en el desarrollo de este proyecto, consistía en seleccionar una fuente de energía viable, tanto para los recursos del proyecto, como para el funcionamiento del mismo. Por lo que se tomó en cuenta la opción de selección de celdas fotovoltaicas por separados para luego ensamblar los paneles solares dentro de los laboratorios de la Universidad del Valle de Guatemala. Por este motivo se contactó Azur Space y SpectroLab, los cuales son proveedores, para obtener las especificaciones técnicas de su producto. Estos detalles se pueden ver en el inciso c "Posibles Paneles Solares" en la sección Marco Teórico, Módulos de un CubeSat, Modulo de potencia. Es importante resaltar el punto de que se decidió no considera a Spectro Lab dentro de las opciones factibles para este proyecto, dado que el proveedor no indico el precio especifico de sus productos, y el valor de compra mínima excede el presupuesto del proyecto, establecido para estos elementos.

En la Fase II del proyecto CubeSat, ya se había seleccionado un tipo de paneles solares ya ensamblados, pero este se tomó como referencia para comparar las nuevas opciones, dado que el precio de los paneles solares ya ensamblados se incrementa de manera considerable. Esta comparación se hizo en base al método "Trade Study", y los resultados de ella se pueden ver en el Cuadro 44, en el inciso 1 "Selección De Paneles Solares", de la sección de Resultados, Potencia. En donde se determinó que la mejor opción es utilizar las celdas solares 3G30A producidas por Azur Space, esto debido a que son las de mejor eficiencia, 29.6% un poco mayor a las del modelo 3G28A, también producidas por Azur Space, pero dado que se desea contar con la fuente más eficiente posible y la diferencia entre los precios es de €5.00, lo que no es algo relativamente alto, se estableció que esta sería la mejor fuente de energía que se pueda disponer para este proyecto.

Luego de establecer la fuente de energía que se utilizaría para el satélite, para poder llegar a hacer un análisis de producción vs consumo, era indispensable conocer el entorno en el cual se encontraría funcionando el satélite, por lo que el siguiente paso fue estudiar las posibles órbitas en las cuales se pudiera colocar el satélite, y seleccionar la mejor opción en base a la finalidad de la misión, tomar fotos del Lago de Atitlán para realizar un estudio de la cianobacteria.

Dado esto, se realizó una selección de orbitas de diferentes proveedores, todas siendo órbitas tipo LEO, esto debido a que no se desea una variación de la altura del satélite a lo largo de la órbita, y se espera mantener al satélite en un entorno relativamente estable a lo largo de la misión. Los parámetros keplerianos de las órbitas seleccionadas para su análisis se pueden ver en el Cuadro 12, en el inciso 3 "Parámetros de las órbitas analizadas", en la sección de Marco Teórico, Órbita.

Para realizar la simulación de la órbita, se optó por utilizar el software STK "Satellite Tool Kit", esto debido a que es un software amigable, bastante intuitivo del cual se puede obtener bastante información sobre órbitas espaciales con gran facilidad, el único inconveniente es que únicamente se contó con la licencia para estudiante que provee el creador de STK, AGI, y dado que el precio de las licencias es

realmente elevado, se contaba con esta limitante. El problema de contar únicamente con este tipo de licencia es que se encontraba limitado a no proporcionar información relevante de la órbita, como lo es la temperatura y la radiación que experimenta el satélite a lo largo de la órbita. Pero aun así es posible contar con información como velocidad, posición y altura del satélite a lo largo de la órbita, si se establece un área de estudio, se puede conocer en que instante es posible hacer contacto con el satélite desde el área delimitada, la duración del contacto, si se encontraba de día o de noche, entre otros datos. También permite añadir a la simulación un sensor que represente a la cámara que estará montada en el satélite, y con esto se puede establecer la frecuencia en la cual la cámara pasara sobre el área de estudio y podrá tomar fotografías de la misma.

Con base en la información anterior se determinó la órbita que representaba la mejor opción para la misión, por lo que se eligió a la órbita en la cual el satélite tuviera la mayor frecuencia de sobrevuelos sobre Guatemala y sobre el Lago de Atitlán. Como se mencionó anteriormente, la simulación de cada una de las órbitas se realizó con ayuda del software STK, el cual se configuro para hacer la simulación en dos años de funcionamiento, desde el 1 de enero de 2018, hasta el 31 de diciembre de 2019, ya que, aunque se espera una vida útil para la misión, de cerca de seis meses, este es un análisis basado en estadística, por lo que es importante abarcar un periodo mayor que pueda indicar que tanto pueden varias las condiciones del satélite estando en funcionamiento.

Los resultados de todas las órbitas analizadas se pueden ver en el Cuadro 45, en el inciso 2 “Selección de la órbita”, de la sección de Resultados, Potencia. En este se puede apreciar que la mejor opción es la órbita S450-20D, la cual tiene un total de sobrevuelos sobre de Guatemala de 1486, sobrevuelos sobre Guatemala en periodos de luz de 683, y sobre vuelos sobre el lago en periodos de luz de 212. Los cuales son poco menos del doble de sobre vuelos comparado a la segunda órbita con mejores resultados, S500-27D, por lo tanto, se estableció esta órbita sobre la cual se obtendrían los datos relevantes tanto para este módulo como para el resto de módulos involucrados en el proyecto CubeSat.

Para la orientación de este proyecto es necesario conocer toda la información descrita anteriormente, pero para hacer el análisis de producción vs consumo de energía es indispensable conocer los tiempos de luz y sombra de la órbita, dado que de esto depende la fuente de energía del satélite, por lo que al igual que en los procesos anteriores, con ayuda de STK se determinaron los periodos de sombra y luz, en el mismo periodo de simulación que el dato anterior. Los resultados de ello se pueden ver en el Cuadro 46, en el inciso 2 “Selección de la órbita”, de la sección de Resultados, Potencia. En donde se obtuvo un periodo de duración de la órbita de 93.05 minutos, con un periodo de luz de 59.22 minutos y un periodo de sombra de 33.83 minutos. En la simulación los resultados de forma gráfica se vieron como la Figura 32 y Figura 33, dentro del mismo inciso, en donde se puede apreciar marcado en amarrillo el periodo de luz y en azul el periodo de sombra, siendo interesante el hecho existe un lapso en el cual a pesar de encontrarse por detrás

de planeta aun recibe luz solar, esto debido a la magnitud de los cuerpos celeste, y el comportamiento de la luz.

Por lo tanto, el satélite únicamente podría funcionar con la energía producida por los paneles solares durante 59.22 minutos, y luego deberá poder almacenar la energía suficiente para poder funcionar durante 33.83 minutos. En base a esta información se realizó el análisis de producción vs consumo, con la finalidad de determinar la cantidad mínima de paneles solares con la cual el satélite pueda funcionar de forma correcta, y la capacidad de almacenamiento que se necesita para que el satélite pueda funcionar en los periodos de sombra.

Para realizar el análisis, se realizó una simulación de la producción y consumo de energía que tendría el satélite en el transcurso de una órbita completa, por lo que se representó a la órbita como un circulo completo en donde a 0° el satélite se encontraría a la mitad del periodo de sombra, en el punto de temperatura más baja, y a 180° el satélite se encontraría a la mitad del periodo de luz, siendo este el punto más cercano al Sol, y por lo tanto con la mayor temperatura. Este planteamiento se puede apreciar en la Figura 68, en el inciso 2 “Selección de la órbita”, de la sección de Resultados, Potencia. Dado que era necesario conocer la variación de temperatura para poder establecer la variación en la eficiencia de los paneles solares, era necesario conocer esta información, pero como se mencionó anteriormente, este dado no era posible obtenerlo con STK, por lo que se asumieron los datos de temperatura de la Estación Espacial Internacional, pues esta se encuentra en una órbita muy similar a la analizada, estos datos se obtuvieron de (Price. 2001), en donde indican que los límites de la temperatura de la estación espacial alcanzan -157 °C en su punto más bajo y 121 °C en su punto más alto. Y dado que no se conocen los materiales de satélite no es posible determinar la variación de temperatura que sufriría siendo sometida a esta variación de temperatura en el periodo de la órbita, se definió una variación lineal de temperatura, siendo el punto más a menor temperatura, el instante donde es satélite se encuentra a la mitad del periodo de sombra, y el punto a mayor temperatura, en la mitad del periodo de luz, esto se puede entender mejor con la Figura 69, en el mismo inciso mencionado. En esta última figura se puede apreciar que se establece que el satélite siempre se encontrará con los paneles solares orientados en dirección al Sol, esto debido a que se pretende que el módulo ADCS sea “activo” por lo que el satélite podrá tener este comportamiento.

Después de haber establecido el comportamiento de la órbita para definir la producción de energía, se definieron tres escenarios de consumo:

a. Sobrevuelos sin una tarea específica, estos son los sobrevuelos los cuales no crucen sobre el área de estudio, y no sea posible tomar una foto o establecer comunicación entre el satélite y la estación en Tierra.
b. Sobrevuelos dentro del rango de comunicación, estos son los sobrevuelos que crucen dentro del área en la cual es posible establecer comunicación entre el satélite y la estación en Tierra.

c. Sobrevuelos sobre el Lago de Atitlán, estos son los sobrevuelos que cruzan sobre el área de estudio y es posible tomar una fotografía.

Además se establecieron los tiempos de funcionamiento de cada uno de los módulos, definiendo que el módulo OBC y ADCS, por su finalidad se encontrarían operando a lo largo de toda la órbita, mientras que COM podría operar como máximo, el tiempo en el cual se pueda realizar contacto con el satélite desde la estación en tierra, mientras que para el módulo Payload, no se ha terminado de definir su tiempo de funcionamiento, pero dado su funcionalidad, el tiempo máximo en el cual puede funcionar es el tiempo máximo desde el cual se puede hacer contacto con el satélite desde la estación en tierra, este se definió para tener los datos enfocados al peor escenario posible. Por lo tanto, era necesario conocer el periodo máximo de tiempo en el cual se puede hacer contacto con el satélite desde la estación en tierra, esta información si fue posible obtenerla con ayuda de STK, los resultados se muestran en el Cuadro 47, en el inciso 2 "Selección de la órbita", de la sección de Resultados, Potencia, en donde se puede ver que el periodo máximo obtenido fue de 12.63 minutos, asumiendo este valor como el tiempo máximo de los módulos COM y Payload, pues en el peor de los casos es este tiempo en el que necesitarían consumir energía.

Con todo lo anterior ya definido, para determinar el menor número de paneles con los cuales es funcional el satélite, se realizó una simulación de producción vs consumo a lo largo de cada uno de los escenarios, con modelos desde cinco paneles solares (5PS), hasta un panel solar (1PS), es importante aclarar que dado que se eligió a las celdas 3G30A como fuente de energía, se estableció que cada panel solar será ensamblado con dos de estas celdas conectadas en serie, por lo que cada panel solar producirá el doble de voltaje a la misma corriente. Los resultados de cada uno de estos modelos se pueden apreciar desde el inciso 4 al inciso 8, en la sección de Resultados, Potencia. Un resumen comparativo de los resultados de cada uno de los modelos se puede ver en el Cuadro 57, en el inciso 10 "Resumen de resultados", de la sección de Resultados, Potencia. En este se puede apreciar que, para la mayoría de modelos, al final de todos los escenarios se obtiene más energía de la que se consume, con excepción del modelo con un panel solar "1PS", el cual dio como resultado más energía consumida que la producida en todos los escenarios, por lo tanto, se puede establecer que el menor número de paneles solares con el cual el satélite es funcional para todos los escenarios es 2, pero dado los riesgos de la misión el modelo que debe ser seleccionado es de tres paneles solares, pues no es posible determinar si habrá un problema al desplegar a uno de los paneles o si alguno sufrirá daño causado por desechos espaciales en el transcurso de la misión, por lo que de perder un panel o el equivalente a la producción de uno de ellos, aun se contaría con suficiente producción de energía para mantener al satélite funcionando de forma correcta.

Para determinar la capacidad de la batería para que el satélite pueda funcionar a lo largo del periodo de sombra de la órbita, se utilizó información técnica de modelos de batería estándar, utilizando baterías tipo Li-Po, dado que estas pueden funcionar en condiciones espaciales dentro de un satélite aislado, esta información se puede ver en el inciso e al h, en la sección de Marco Teórico, Módulos de un CubeSat,

Modulo de potencia. En esto se puede notar que, a pesar de ser baterías con una gran diferencia en su capacidad de almacenamiento, el voltaje de carga es 4.2 V y el voltaje nominal es 3.7 V para ambas. Por lo que es posible encontrar baterías entre estos valores de capacidades, 250 mAh y 4000 mAh, en donde el voltaje nominal y de carga siga siendo el mismo.

La magnitud de la batería seria determinada en base a la energía consumida en la etapa de sombra, dado que sin importar si se utiliza el modelo de tres paneles solares o de dos paneles solares, en cualquiera de los escenarios la energía producida era mayor a la energía consumida, no era relevante considerar si dentro de los periodos de luz se necesitaba consumir energía de la batería, pues esta al final se encontraría con la suficiente energía almacenada para funcionar en el periodo de sombra. La energía consumida en los periodos de sombra para cada uno de los modelos se puede ver en el Cuadro 55, en el inciso 9 “Capacidad de almacenamiento de la batería” de la sección de Resultados, Potencia, en donde se puede apreciar que sin importar el modelo la energía consumida en el periodo de sombra es el mismo, esto se debe a que los paneles se encontraran enfocados a la luz desde que entren hasta que salgan del periodo de luz, el periodo de producción de energía será el mismo sin importar el modelo, y siendo el consumo el mismo sin importar el modelo la energía necesaria para funcionar en los periodos de sombra no depende del modelo que se elija. En cuanto a la energía consumida en el periodo de sombra se obtuvo asumiendo un voltaje nominal de la batería de 3.7 V, con lo cual se obtiene que en el periodo de sombra se consumen 302.2 mAh.

El valor de la capacidad de la energía de la batería elegida no puede ser cercano a este valor, ya que, de ser así, cada orbita completa representaría un ciclo de la batería, y dado que una órbita dura 93.05 minutos, y una batería como las mencionadas tiene cerca de 500 ciclos, el tiempo de vida de la batería seria cercana a un mes de duración, lo cual no es ni la mitad de lo esperado. Por lo que siguiendo lo establecido en (Hoffart. 2008), para extender la vida de una batería de 5 a 10 veces su uso, el consumó debe ser únicamente entre el 20% al 30%, por lo tanto, si se establece que 302.2 mAh representan un 20% de la capacidad de la batería, la capacidad de la batería que se debe usar debe ser 1511 mAh, lo cual como modelo estándar seria el mostrado en el inciso H, de la sección Marco Teórico, con una capacidad de 2300 mAh, siendo este un valor comercial. Tomando en cuenta que la corriente de carga es distinta para todas las baterías, es importante recalcar que entre menor sea la capacidad de almacenamiento de la batería, menor será su corriente máxima de carga, pero como se puede ver en el Cuadro 56, en el inciso 9 “Capacidad de almacenamiento de la batería” de la sección de Resultados, Potencia, la corriente de carga máxima para el modelo de tres paneles solares (3PS) es de 1.73 A, mientras que para la batería de 2300 mAh la corriente máxima de carga es de 2.3 A, por lo que puede ser cargada con esta corriente, a diferencia del siguiente valor comercial de 1500 mAh, pues como se puede ver en el inciso I, de la sección Marco Teórico, para esta batería la corriente máxima es de 1.5 A.

VII. CONCLUSIONES

Con base a la investigación y haciendo uso de un estudio de caso, se eligió el microprocesador Raspberry Pi Zero, del cual se observó que es capaz de cumplir con las necesidades de conexión internas del satélite, requerimientos de masa y volumen para ser considerado como OBC. Luego se documentó la potencia que el Raspberry Pi Zero necesita para funcionar como OBC, esta es de al menos 0.7 W.

Con el resultado de las pruebas obtenidos por medio del programa de envío y recepción de datos, se obtuvo que es posible enviar información de tres imágenes, de 800 kB, al módulo COMMS en menos de 4 minutos a una tasa de transferencia de 115200 baudios. Al analizar lo anterior, se concluyó que, con la limitante de la tasa de transferencia del microcontrolador del módulo COMMS, no es posible transferir una imagen de 10 MB en menos de 4 minutos. Al aumentar la tasa de transferencia no hubo una mejora considerable en el tiempo de transmisión. Existe un fenómeno de perdida de datos a partir una tasa de transferencia de 460800 baud/s hasta la tasa de transferencia de 921600 baud/s.

Al cambiar las configuraciones de inicio de la Rasbperry Pi Zero, se logró crear un programa en el lenguaje Python, capaz de orquestar a los módulos conectados a OBC, desde el encendido del microcontrolador.

Se definieron los componentes del módulo de comunicaciones para satélites CubeSat, siendo seleccionado el microcontrolador en la placa de aplicación Teensy 3.2 como *Terminal Node Controller,* el chip CC2500 de Texas Instruments en su circuito de aplicación como *Transceiver*, y la antena de tipo parche para la frecuencia seleccionada.

El módulo de comunicaciones realizado es aplicable no solo a un satélite de tipo CubeSat sino a cualquier otro proyecto que necesite usar transferencia de datos inalámbrica a una tasa de 2 Mbps, usando únicamente comunicación serial con el módulo para poder enviar y recibir información. Estas aplicaciones pueden ser desde controles remotos con botones de funciones sencillas hasta una HMI para acciones más complejas.

El componente *Terminal Node Controller* en una aplicación de CubeSat puede ser implementado mediante software en la computadora abordo del sistema, puesto que este tipo de computadoras contienen los protocolos de comunicación que utilizan los *transceiver* comerciales que son usados en aplicaciones aeroespaciales, logrando así una reducción de la masa, volumen y consumo de potencia del módulo de comunicaciones.

Entre las órbitas analizadas para la misión, la que cuenta con la mayor frecuencia de vuelos sobre Guatemala, es la S450-20D, del proveedor Space Flight, la cual es una órbita baja, con una altura de 449 km, y una inclinación de 20°, en esta órbita se puede obtener un vuelo sobre Guatemala cerca de dos veces

por día, siendo uno en periodo de luz. Se obtiene también un sobrevuelo en periodo de luz sobre el Lago de Atitlán, aproximadamente una vez por semana.

La fuente de energía seleccionada para el satélite, tras el análisis realizado, es la celda fotovoltaica 3G30A fabricadas por Azur Space, esto debido a que entre las opciones analizadas son las que presentan mayor eficiencia de producción de energía, a un costo razonable en base al presupuesto del proyecto.

En el análisis de producción vs. consumo, se conocían únicamente el definido para el módulo OBC y Payload, por lo que para los consumos de los módulos ADCS y COM se utilizaron valores teóricos en base a una misión exitosa y a la selección de la siguiente fase del proyecto respectivamente, y para el tiempo de funcionamiento de cada uno se utilizaron los tiempos máximos posibles de funcionamiento para hacerlo en base al "peor escenario".

Con base en los resultados, se concluyó que el menor número de paneles con los que puede funcionar el satélite son dos paneles instalados, pero debido a las condiciones de funcionamiento y el riesgo que implica si fallara alguno al desplegarse o ser dañado por basura espacial, la mejor elección para el proyecto es utilizar el modelo de tres paneles solares, ya que de perder uno de los paneles aún podría funcionar, pero con 1 solo panel útil no es capaz de funcionar durante una órbita completa, obteniendo una diferencia negativa de energía al final de cualquier escenario.

Conociendo la producción de energía obtenida, y la energía necesaria para poder funcionar durante el periodo de sombra, teniendo un tiempo de funcionamiento aceptable para la misión, se obtuvo que el valor de la capacidad mínima para la batería, es de 1511 mAh, pero dado que esto no es un valor comercial, el más cercano es de 2300 mAh. No se consideró el modelo comercial de 1500 mAh ya que la corriente máxima de carga de este modelo es menor a la corriente máxima de carga que se produce con el modelo de tres paneles solares. Utilizando una batería de 2300 mAh, es posible almacenar la energía necesaria para que el satélite pueda funcionar de forma correcta sin ningún riesgo, además puede ser capaz de entregar más que la corriente máxima de consumo y soportar más que la corriente máxima de carga, teniendo una expectativa de vida de entre 10 a 20 meses.

VIII. RECOMENDACIONES

Basado en los resultados obtenido en las pruebas de tiempo, tomando en cuenta que se desea descargar imágenes de 10 MB de tamaño, y conociendo el tiempo máximo que pasa el satélite sobre Guatemala, se recomienda la toma de imágenes de menor tamaño, o menor resolución, para poder tener el tiempo suficiente para lograr enviar las imágenes a una estación en tierra.

Se recomienda la realización de pruebas con los módulos ADCS y COMMS a utilizar en el satélite para poder comprobar de una mejor manera que la Raspberry Pi Zero es una opción capaz de ser usada como OBC del satélite.

Debido al fenómeno que se presentó durante las pruebas realizadas, se recomienda realizar pruebas adicionales de transferencia de imágenes a velocidades superiores a 115200 baud/s.

Al observar los resultados obtenidos al comparar las imágenes, se recomienda realizar un análisis de frecuencias para respaldar los datos obtenidos en la verificación de similitudes entre las imágenes RAW y las imágenes comprimidas JPEG.

La utilización de un FPGA en la implementación del módulo de comunicaciones permitiría que se puedan hacer correcciones de circuitería en el módulo incluso estando el satélite en órbita, pudiendo así reducir riesgos en la misión y asegurar el correcto funcionamiento de todo.

Según investigación de integrantes del megaproyecto, las regulaciones de radio en Guatemala no tienen apartado para utilización de frecuencias amateur en aplicaciones satelitales para el uso de instituciones, por lo que se recomienda revisar las frecuencias amateur disponibles para poder recibir y enviar información entre estaciones en Guatemala y el satélite.

Se recomienda buscar otras estaciones en tierra alrededor del mundo que puedan obtener la información del satélite y enviarla a Guatemala de tal manera que la ventana de comunicación sea más amplia.

El uso de protocolos de comunicación entre el módulo y la computadora abordo debe tener como objetivo que el envío de información entre ellos sea rápido y confiable para que el módulo pueda enviar y recibir la información en la ventana de comunicación disponible.

Si la transmisión de datos entre módulo de COMMS y computadora de abordo es lenta se recomienda implementar una memoria flash en el módulo para poder almacenar la información a enviar o recibir, o implementar el *Terminal Node Controller* en la computadora de abordo.

Se recomienda realizar un análisis de la computadora de abordo del satélite incluyendo el *Terminal Node Controller* de este trabajo como una opción ya que presenta características muy completas para realizar el trabajo de computadora de abordo.

Es importante resaltar que los datos obtenidos serán válidos únicamente para la órbita establecida, pues de cambiar la órbita los tiempos de sombra y de luz serán diferentes, lo que podría afectar el resultado de energía almacenada al final de cada órbita, y los tiempos establecidos para comunicación y captura de imagen, siendo necesario cambiar la capacidad de la batería, la cantidad de paneles o ambos. Por lo que cada vez que se altere algún parámetro de la órbita lo mejor es repetir este análisis, por lo menos hasta determinar la duración de los periodos de luz y sombra, pues también es posible que exista más de una órbita con periodos de luz y sombra muy similares.

Además de esto, los resultados de producción están con base en la producción mínima según el flujo de electrones por los paneles, pero el modelado a lo largo de la órbita es un modelo simplificado en el cual solo se toma en cuenta cómo afecta la variación de temperatura, siendo una aproximación aceptable y valida, pero en funcionamiento, los paneles también varían su producción en base a la radiación a la que se encuentran expuestos, este modelo se puede encontrar en (Gonzáles. 2014). Por lo que, si se desea tener una simulación mucho más precisa, se deberá de contar con los valores del entorno del satélite que no se pudieron obtener con STK, en su versión para estudiantes, junto con los valores exactos de la temperatura a lo largo de la órbita.

IX. BIBLIOGRAFÍA

[1]. Amber Wireless. *AMB2520.* Disponible en: https://www.amber-wireless.com/fileadmin//user_upload/DOWNLOADS/amb2520_ds.pdf

[2]. Analytical Graphics, Inc. STK Systems Tool Kit and Systems Tool Kit. 2015. https://www.agi.com/downloads/corporate/partners/edu/ates_practica1.pdf [25/01/2016] https://www.agi.com/resources/educational-alliance-program/curriculum_exercises_labs/Barcelona/Barcelona_p1_STK.pdf [25/01/2016] https://www.agi.com/resources/help/online/stk/10.1/STK/pdf/stk.pdf [28/01/2015]

[3]. Atmel. ATmega48A/PA/88A/PA/168A/PA/328/P. Dispoinible en: http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf

[4]. Azur Space. 2010. Triple Junction GaAs Solar Cell Assembly. http://www.azurspace.com/images/products/HNR_0003805-01-00.pdf

[5]. B. Blevins. 1999. *Small Spacecraft Antenna Selection Tutorial.* AIAA Conference on Small Satelites, Utah

[6]. Benzal, Luis A. y Velasco, V. 2010. Viabilidad de un satélite en órbita LEO. Universidad Politécnica de Cataluña. Barcelona, España. 102 págs.

[7]. Berbeglia, Gerardo; y Fernández, D. 2003. Simulación de sistemas de potencia solares para satélites. Facultad de Ciencias Exactas y Naturales. Universidad De Buenos Aires. Argentina. 66 págs.

[8]. C. Brown. 2002. *Elements of Spacecraft Design*. Virginia: AIAA.

[9]. Chanes, Sebastián; y P. Valencia. 2014. Diseño de una aplicación de monitoreo para un satélite CubeSat. Escuela Superior De Ingeniería Mecánica Y Eléctrica. Instituto Politécnico Nacional. México D.F. 85 págs.

[10]. Chris Peat, Heavens-Above GmbH. 2016. http://heavens-above.com/orbit.aspx?satid=25544 [25/01/2016]

[11]. Clyde Space. 2015. *CubeSat S band transmitter*. Disponible en: http://www.clyde-space.com/cubesat_shop/communication_systems/ 301_cubesat-s-band-transmitter

[12]. Clyde Space. 2015. CubeSat *UHF/VHF transciever*. Disponible en: http:// www.clyde-space.com/cubesat_shop/communication_systems/170_cmc- cubesat-vutrx Cote, K., Gabriel, J., Patel, B., Ridley, N., Taillefer, Z., & Tetreault, S. (2011). Mechanical, Power, and Propulsion

Subsystem Design for a CubeSat. Massachusetts: Worcester Polytechnic Institute.

[13]. CubeSat Shop. 2015. *ISIS TXS S band transmitter*. Disponible en: http:// www.cubesatshop.com/index.php?page=shop.product_details&flypage=flypage.tpl&product_id=9 &category_i d=5&option=com_virtuemart&Itemid=67

[14]. CubeSat Shop. 2015. *ISIS UHF VHF duplex transciever*. Disponible en: http://www.cubesatshop.com/index.php?page=shop.product_details&flypage=flypage.tpl&product _id=11&category_id=5&option=com_virtuemart&Itemid=67&vmcchk=1&Itemid=67

[15]. Durda, F. (29 de 4 de 2014). Serial and UART Tutorial. Obtenido de FreeBSD: https://www.freebsd.org/doc/en_US.ISO8859-1/articles/serial-uart/

[16]. E. Ogheohwo. 2015. *Design,construction and performance analysis of helical antenna operating at 5.8GHz.* Universidad de Jos, Plateau State, Nigeria. Disponible en: https://www.arcjournals.org/pdfs/ijarps/v2-i11/5.pdf

[17]. Elbrecht, Alexander; S. Dech y A. Gottscheber. 2011.1U CubeSat Design for increased Power Generation. SRH University Heidelberg. Alemania. 7 págs.

[18]. Freescale. *K20P64M50SF0.* Disponible en: http://ww1.microchip.com/downloads/en/DeviceDoc/39609b.pdf

[19]. Freescale. *MC13202 Datasheet with Addendum.* Disponible en: http://cache.nxp.com/files/rf_if/doc/data_sheet/MC13202.pdf?pspll=1

[20]. G.A.U.S.S. 2012. Launch Services. http://www.gaussteam.com/services/launch-services/

[21]. García-Hidalgo, Adrián. 2013. Diseño Del Sistema De Potencia Para Un Satélite De Órbita Baja (LEO). Escuela Universitaria de Ingeniería Técnica Aeronáutica. Universidad Politécnica de Madrid. Madrid, España. 110 págs.

[22]. Gonzáles, Jesús; y G. Puerto. 2014. Estimación de la cantidad de potencia suministrada por las celdas fotovoltaicas de un cubesat. Universidad Sergio Arboleda. Bogotá, Colombia. 11 págs.

[23]. González, Kevin. 2016. Módulo De Potencia Para Un CubeSat 1U. Tesis Universidad del Valle de Guatemala. Guatemala.

[24]. Grupo de Nuevas Actividades Profesionales. 2002. Energía Solar Fotovoltaica. Colegio Oficial de Ingenieros de Telecomunicación. Madrid, España.

[25]. Hamama, S., Zovaro, A., & Wu, X. (2006). GammaSat: A Low-Cost Student-Led CubeSat Design. Sydney: University of Sydney.

[26]. Heliophysics Science Division, Code 670. NASA Goddard Space Flight Center Greenbelt, USA. 2013. http://eclipse.gsfc.nasa.gov/SEdecade/SEdecade2011.html [30/01/2016]

[27]. Hoffart, Fran. 2008. Proper Care Extends Li-Ion Battery Life. Power Electronics Technology. 5 págs.

[28]. Homotix, S.a.S. Datasheet Batterie LiPo. http://hyperiontechnologies.nl/products/iadcs100/

http://www.homotix.eu/pimages/FCKeditorFiles/Image/Datasheet_Batterie_LiPo-.pdf
http://www.azurspace.com/images/products/0003421-01-01_DB_3G30C_80x80.pdf
http://www.azurspace.com/images/products/HNR_0002490-00-03.pdf

[29]. Hyperion Technologies B.V. 2016. iADCS100 Product description. Retrieved from Attitude Determination and Control Systems:

[30]. J.L. Tamasi, Mariana. 2003. Celdas Solares para Uso Espacial: Optimización de Procesos y Caracterización. Instituto De Tecnología. Universidad Nacional De General San Martín. República Argentina. 188 págs.

[31]. Kashyap, N. (02 de Junio de 2004). JPEG Image Code Format. Obtenido de Universidad de Massey: http://www.massey.ac.nz/~mjjohnso/notes/59731/presentations/jpeg.pdf

[32]. Klofas, Bryan; J. Anderson. 2008. *A survey of CubeSat communication system.* San Luis Obispo, California: Universidad Politécnica de California. Disponible en: http://www.klofas.com/papers/CommSurvey-Bryan_Klofas.pdf

[33]. Krogh, K., Schreder, E., & Bak, T. (2002). Attitude Determination for AAU CubeSat. Aalborg: Aalborg University.

[34]. M. Afridi. 2015. *Microstrip patch antenna – Designing at 2.4 GHz Frequency*. Univsersidad de Ingenieria y Tecnología. Peshawar, Pakistan. Disponible en: http://www.ss-pub.org/wp-content/uploads/2015/03/3-BCR-E20140906-02.pdf

[35]. M. Jamali. 2012. *A study on Conformal Antenna Solutions for Cube Satellites.* Utah State University. Disponible en: http://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=2339&context=etd

[36]. Martin, Daniel. 2011. Evaluating The Effectiveness Of Peak Power Tracking Technologies For Solar Arrays On Small Spacecraft. University of Kentucky. USA. 74 págs.

[37]. Martínez, Mónica G. 2004. Diseño, elaboración, caracterización y ensayos de dispositivos fotovoltaicos para usos espaciales. Instituto de Tecnología. Universidad Nacional de General San Martín. República Argentina. 198 págs.

[38]. Mehrparvar, Arash. 2014. *CubeSat Design Specification Rev 13.* San Luis Obispo, California: Universidad Politécnica de California. Disponible en: https://static1.squarespace.com/static/5418c831e4b0fa4ecac1bacd/t/56e9b62337013b6c063a655a/1458157095454/cds_rev13_final2.pdf

[39]. Microchip. *PIC16F882/883/884/886/887.* Disponible en: http://ww1.microchip.com/downloads/en/DeviceDoc/40001291H.pdf

[40]. Microchip. *PIC18(L)F2X/4XK22.* Disponible en: http://ww1.microchip.com/downloads/en/DeviceDoc/40001412G.pdf

[41]. Microchip. *PIC18F6520/8520/6620/8620/6720/8720.* Disponible en: http://ww1.microchip.com/downloads/en/DeviceDoc/39609b.pdf

[42]. Microhard System. *MHX-2400*. Disponible en: http://spacegrant.colorado.edu/COSGC_Projects/co3sat/downloads/MHX2400ManualRev1.56.pdf

[43]. Muñoz, Antonio. 2016. Desarrollo del Prototipo de la Carga Útil de un Satélite para la Determinación de Niveles de Cianobacteria en el Lago Atitlán. Tesis Universidad del Valle de Guatemala. Guatemala.

[44]. National Instruments. 2015. *Introduction to RF & Wireless communication.* Disponible en: http://www.ni.com/tutorial/3541/en/

[45]. Nikon. (25 de 9 de 2013). Differences between RAW, TIFF, and JPG file formats. Obtenido de Nikon: http://nikonasia-en.custhelp.com/app/answers/detail/a_id/885/~/differences-between-raw,-tiff,-and-jpg-file-formats%3F

[46]. Noe, Chris. 2004. *Design and Implementation of the Communications Subsystem for de Cal Poly CP2 CubeSat Project.* San Luis Obispo, California: Universidad Politécnica de California. Disponible en: http://www.crn2.inpe.br/conasat1/projetos_cubesat/subsistemas/COM/CP2%20-%20COM%20-%20Communication%20Subsystem.pdf

[47]. Oracle. (2016). Image RAW Pixel Format. Obtenido de Oracle: https://docs.oracle.com/cd/B28359_01/appdev.111/b28414/ap_imgraw.htm

[48]. Pereda, Isidro E. 2005. Celdas Fotovoltaicas En Generación Distribuida. Escuela de Ingeniería. Pontificia Universidad Católica de Chile. Santiago, Chile. 173 págs.

[49]. Price, Steve; Dr. T. Phillips y G. Knier. 2001. Manteniéndose Fresco en la Estación Espacial. https://ciencia.nasa.gov/science-at-nasa/2001/ast21mar_1/

[50]. Prototype and Circuit Board. PCB2810. Disponible en: http://www.onlinecomponents.com/datasheet/pcb2810.aspx?p=12176292

[51]. Spaceflight Industries, Inc. 2016. Schedule & Pricing. http://www.spaceflight.com/schedule-pricing/

[52]. Sparkfun. (2014). Serial Communication. Obtenido de Sparkfun: https://learn.sparkfun.com/tutorials/serial-communication

[53]. SpectroLab. 2016. Triple Junction GaAs Solar Cell Assembly. http://www.spectrolab.com/DataSheets/cells/XTJ_Prime_Data_Sheet_7-28-2016.pdf

[54]. Sterling, Scott; R. Nuzzaci y A. Gordon-Ross. 2012. Energy Budgeting for CubeSats with an Integrated FPGA. IEEE. Department of Electrical and Computer Engineering, University of Florida. USA. 14 págs.

[55]. Swartwout, M. (2013). e First One Hundred CubeSats: A Statistical Look. Journal of Small Satellites, 213-233.

[56]. SwissCube. 2008. SwissCube Mission and System Overview. UNINE / HES-SO / EPFL. Lausanne, Suiza. 141 págs.

[57]. T. Lassen. 2014. *Long-range RF communication. Texas Instruments*. Disponible en

http://www.ti.com/lit/wp/swry006/swry006.pdf

[58]. Texas Instruments. (Noviembre de 2010). KeyStone Architecture Universal Asynchronous Receiver/Transmitter (UART). Obtenido de Texas instruments: http://www.ti.com/lit/ug/sprugp1/sprugp1.pdf

[59]. Texas Instruments. *Low-Cost Low-Power 2.4 GHz RF Transceiver.* Disponible en: http://www.ti.com/lit/ds/swrs040c/swrs040c.pdf

[60]. Traussnig, Wolfgang. 2007. *Design of a Communication and Navigation Subsystem for a CubeSat Mission.* Graz, Austria: Universidad de Graz. Disponible en: http://physik.uni-graz.at/spacesciences/archive/files/ULG_II_Master_Thesis_Traussnig.pdf

[61]. W. Honcharenko, J. P. Kruys, D. Y. Lee, N. J. Shah. *Broadband wireless Access*. IEEE Communications Magazine. pp. 20-26, 1997

[62]. Wallace, G. K. (1991). The JPEG Still Picture Compression Standard. Massachusetts: IEEE Transactions on Consumer Electronics.

[63]. Wells, James; L. Stras y T. Jeans. 2003. Canada's Smallest Satellite: The Canadian Advanced Nanospace eXperiment (CanX-1). Space Flight Laboratory. University of Toronto Institute For Aerospace Studies. Toronto, Canadá. 12 págs.

X. ANEXOS

A. REQUERIMIENTOS

1. Especificaciones del CubeSat

a. Requerimientos generales

1) Todas las partes deben permanecer sujetadas al CubeSat durante el lanzamiento, eyección y operación. No se crearán desperdicios adicionales en el espacio.

2) Los materiales del CubeSat deben satisfacer los siguientes criterios de baja emisión de gases para prevenir la contaminación a hacia otros aparatos espaciales durante la integración, pruebas y lanzamiento.

a) Los materiales deben tener una pérdida total de masa menor a 1.0%.
b) Los materiales deben tener materiales volátiles condensados recopilados menor a 0.1%.

b. Requerimientos mecánicos

1) La masa máxima dentro de un CubeSat 1U debe ser de 1.33kg.
2) Las dimensiones máximas para un CubeSat 1U son de 10cm x 10cm x 10cm

c. Requerimientos de operación

1) Ningún CubeSat debe generar o transmitir alguna señal desde la integración dentro del P-POD hasta 45 minutos después de haber sido puesto en órbita desde el P-POD. Sin embargo, el CubeSat sí podrá estar encendido después de haber sido puesto en órbita por el P-POD.

2) Ninguna parte desplegable será extendida por al menos 30 minutos después de haber sido puesto en órbita por el P-POD.

2. Requerimientos de prueba

a. Pruebas de vibración. El CubeSat deberá soportar pruebas de vibraciones aleatorias, estas serán definidas por el proveedor del lanzamiento.

b. Pruebas térmicas. El CubeSat debe soportar pruebas de horneado de vacío térmico para asegurar la correcta emisión de gas de los componentes. Estas pruebas serán definidas por el proveedor del lanzamiento.

c. Pruebas de choque. El CubeSat deberá soportar pruebas de choque, estas serán definidas por el proveedor del lanzamiento.

d. Inspección visual. Se realizará una inspección visual del CubeSat y serán medidas áreas críticas para comprobar el listado de aceptación del CubeSat.

B. PROGRAMAS UTILIZADOS

1. COMMS: Programa de envío de datos

```
/**
 * CC2500 transceiver code
 * Adapted from https://github.com/yasiralijaved/Arduino-CC2500-Library
 * Modify by Javier Alay
 *
 * Hardware SPI:
 * MISO -> 12
 * MOSI -> 11
 * SCLK/SCK -> 13
 * CSN/SS - > 10
 */
#include "cc2500_REG.h"
#include "cc2500_VAL.h"

#include <SPI.h>

#define CC2500_IDLE    0x36      // Exit RX / TX, turn
#define CC2500_TX      0x35      // Enable TX. If in RX state, only enable TX if CCA passes
#define CC2500_RX      0x34      // Enable RX. Perform calibration if enabled
#define CC2500_FTX     0x3B      // Flush the TX FIFO buffer. Only issue SFTX in IDLE or
TXFIFO_UNDERFLOW states
#define CC2500_FRX     0x3A      // Flush the RX FIFO buffer. Only issue SFRX in IDLE or
RXFIFO_OVERFLOW states
#define CC2500_SWOR    0x38
#define CC2500_TXFIFO  0x3F
#define CC2500_RXFIFO  0x3F

#define TX_TIMEOUT 10 // Timeouts are added
```

```
long previousTXTimeoutMillis = 0;
long previousMillis = 0;
long sendInterval = 400; // in milliseconds

void setup(){
  Serial.begin(9600);

  // Setup
  pinMode(SS,OUTPUT);
  SPI.begin();
  digitalWrite(SS,HIGH);
  Serial.println("Initializing Wireless..");
  init_CC2500();
  Read_Config_Regs();
}

void loop(){
  listenForPacket();
  delay(300);
}

void listenForPacket() {
  Serial.println("Esperando paquete...");
  SendStrobe(CC2500_RX);
  // Switch MISO to output if a packet has been received or not
  WriteReg(REG_IOCFG1,0x01);
  delay(20);
  unsigned long currentMillis = millis();
  if (digitalRead(MISO)) {
    char PacketLength = ReadReg(CC2500_RXFIFO);
    char recvPacket[PacketLength];
    if(PacketLength == 10) {
      Serial.println("Packet Received!");
      Serial.print("Packet Length: ");
      Serial.println(PacketLength, DEC);
      Serial.print("Data: ");
      for(int i = 1; i < PacketLength; i++){
```

```
    recvPacket[i] = ReadReg(CC2500_RXFIFO);
    Serial.print(recvPacket[i]);
    Serial.print(" ");
   }
   Serial.println(" ");
   // Print quality information
   byte rssi = ReadReg(CC2500_RXFIFO);
   byte lqi = ReadReg(CC2500_RXFIFO);
   Serial.print("RSSI: ");
   Serial.println(rssi);
   Serial.print("LQI: ");
   Serial.println(lqi);
 }
 if(PacketLength == 13){
   Serial.println("Photo Received! <<<----------");
   Serial.print("Packet Length: ");
   Serial.println(PacketLength, DEC);
   Serial.print("Data: ");
   for(int i = 1; i < PacketLength; i++){
   recvPacket[i] = ReadReg(CC2500_RXFIFO);
    Serial.print(recvPacket[i]);
    Serial.print(" ");
   }
 if(PacketLength == 33807){
   Serial.println("---------------------------------");
   Serial.println("-----GETTING PHOTO-----");
   for(int i = 1; i < PacketLength; i++){
    recvPacket[i] = ReadReg(CC2500_RXFIFO);
    Serial.println(recvPacket[i]);









   }
 }
```

```
    }

    // Make sure that the radio is in IDLE state before flushing the FIFO
    // (Unless RXOFF_MODE has been changed, the radio should be in IDLE state at this point)
    SendStrobe(CC2500_IDLE);
    delay(50);
    // Flush RX FIFO
    SendStrobe(CC2500_FRX);
  } else {

  }
}

void sendPacket() {
  WriteReg(REG_IOCFG1,0x06);
  // Make sure that the radio is in IDLE state before flushing the FIFO
  SendStrobe(CC2500_IDLE);
  // Flush TX FIFO
  SendStrobe(CC2500_FTX);
  // prepare Packet
  int length = 6;
  unsigned char packet[length];
  // First Byte = Length Of Packet
  packet[0] = length;
  packet[1] = 'H';
  packet[2] = 'e';
  packet[3] = 'l';
  packet[4] = 'l';
  packet[5] = 'o';

  // SIDLE: exit RX/TX
  SendStrobe(CC2500_IDLE);

  Serial.println("Transmitting ");
  for(int i = 0; i < length; i++)
  {
     WriteReg(CC2500_TXFIFO,packet[i]);
```

```
  }
  // STX: enable TX
  SendStrobe(CC2500_TX);
  // Wait for GDO0 to be set -> sync transmitted
  previousTXTimeoutMillis = millis();
  while (!digitalRead(MISO) && (millis() - previousTXTimeoutMillis) <= TX_TIMEOUT) {
    //Serial.println("GDO0 = 0");
  }

  // Wait for GDO0 to be cleared -> end of packet
  previousTXTimeoutMillis = millis();
  while (digitalRead(MISO) && (millis() - previousTXTimeoutMillis) <= TX_TIMEOUT) {
     //Serial.println("GDO0 = 1");
  }
  Serial.println("Finished sending");
  SendStrobe(CC2500_IDLE);
}

void WriteReg(char addr, char value){
  digitalWrite(SS,LOW);

  while (digitalRead(MISO) == HIGH) {
    };

  SPI.transfer(addr);
  delay(10);
  SPI.transfer(value);
  digitalWrite(SS,HIGH);
}

char ReadReg(char addr){
  addr = addr + 0x80;
  digitalWrite(SS,LOW);
  while (digitalRead(MISO) == HIGH) {
    };
  char x = SPI.transfer(addr);
  delay(10);
```

```
  char y = SPI.transfer(0);
  digitalWrite(SS,HIGH);
  return y;
}

char SendStrobe(char strobe){
  digitalWrite(SS,LOW);

  while (digitalRead(MISO) == HIGH) {
  };

  char result =  SPI.transfer(strobe);
  digitalWrite(SS,HIGH);
  delay(10);
  return result;
}

void init_CC2500(){
  WriteReg(REG_IOCFG2,VAL_IOCFG2);
  WriteReg(REG_IOCFG0,VAL_IOCFG0);
  WriteReg(REG_IOCFG1,VAL_IOCFG1);

  WriteReg(REG_FIFOTHR, VAL_FIFOTHR);
  WriteReg(REG_SYNC1,VAL_SYNC1);
  WriteReg(REG_SYNC0,VAL_SYNC0);
  WriteReg(REG_PKTLEN,VAL_PKTLEN);
  WriteReg(REG_PKTCTRL1,VAL_PKTCTRL1);
  WriteReg(REG_PKTCTRL0, VAL_PKTCTRL0);

  WriteReg(REG_ADDR,VAL_ADDR);
  WriteReg(REG_CHANNR,VAL_CHANNR);
  WriteReg(REG_FSCTRL1,VAL_FSCTRL1);
  WriteReg(REG_FSCTRL0,VAL_FSCTRL0);
  WriteReg(REG_FREQ2,VAL_FREQ2);
  WriteReg(REG_FREQ1,VAL_FREQ1);
  WriteReg(REG_FREQ0,VAL_FREQ0);
  WriteReg(REG_MDMCFG4,VAL_MDMCFG4);
```

```
  WriteReg(REG_MDMCFG3,VAL_MDMCFG3);
  WriteReg(REG_MDMCFG2,VAL_MDMCFG2);
  WriteReg(REG_MDMCFG1,VAL_MDMCFG1);
  WriteReg(REG_MDMCFG0,VAL_MDMCFG0);
  WriteReg(REG_DEVIATN,VAL_DEVIATN);
  WriteReg(REG_MCSM2,VAL_MCSM2);
  WriteReg(REG_MCSM1,VAL_MCSM1);
  WriteReg(REG_MCSM0,VAL_MCSM0);
  WriteReg(REG_FOCCFG,VAL_FOCCFG);

  WriteReg(REG_BSCFG,VAL_BSCFG);
  WriteReg(REG_AGCCTRL2,VAL_AGCCTRL2);
  WriteReg(REG_AGCCTRL1,VAL_AGCCTRL1);
  WriteReg(REG_AGCCTRL0,VAL_AGCCTRL0);
  WriteReg(REG_WOREVT1,VAL_WOREVT1);
  WriteReg(REG_WOREVT0,VAL_WOREVT0);
  WriteReg(REG_WORCTRL,VAL_WORCTRL);
  WriteReg(REG_FREND1,VAL_FREND1);
  WriteReg(REG_FREND0,VAL_FREND0);
  WriteReg(REG_FSCAL3,VAL_FSCAL3);
  WriteReg(REG_FSCAL2,VAL_FSCAL2);
  WriteReg(REG_FSCAL1,VAL_FSCAL1);
  WriteReg(REG_FSCAL0,VAL_FSCAL0);
  WriteReg(REG_RCCTRL1,VAL_RCCTRL1);
  WriteReg(REG_RCCTRL0,VAL_RCCTRL0);
  WriteReg(REG_FSTEST,VAL_FSTEST);
  WriteReg(REG_PTEST,VAL_PTEST);
  WriteReg(REG_AGCTEST,VAL_AGCTEST);
  WriteReg(REG_TEST2,VAL_TEST2);
  WriteReg(REG_TEST1,VAL_TEST1);
  WriteReg(REG_TEST0,VAL_TEST0);
}

void Read_Config_Regs(void){
  Serial.print("G2: ");
  Serial.println(ReadReg(REG_IOCFG2),HEX);
   delay(10);
```

```
 Serial.print("G1: ");
Serial.println(ReadReg(REG_IOCFG1),HEX);
 delay(10);
 Serial.print("G0: ");
Serial.println(ReadReg(REG_IOCFG0),HEX);
 delay(10);
 Serial.print("FIFO: ");
Serial.println(ReadReg(REG_FIFOTHR),HEX);
 delay(10);
/*  Serial.println(ReadReg(REG_SYNC1),HEX);
 delay(10);
Serial.println(ReadReg(REG_SYNC0),HEX);
 delay(10);
Serial.println(ReadReg(REG_PKTLEN),HEX);
 delay(10);
Serial.println(ReadReg(REG_PKTCTRL1),HEX);
 delay(10);
Serial.println(ReadReg(REG_PKTCTRL0),HEX);
 delay(10);
Serial.println(ReadReg(REG_ADDR),HEX);
 delay(10);
Serial.println(ReadReg(REG_CHANNR),HEX);
 delay(10);
Serial.println(ReadReg(REG_FSCTRL1),HEX);
 delay(10);
Serial.println(ReadReg(REG_FSCTRL0),HEX);
 delay(10);
Serial.println(ReadReg(REG_FREQ2),HEX);
 delay(10);
Serial.println(ReadReg(REG_FREQ1),HEX);
 delay(10);
Serial.println(ReadReg(REG_FREQ0),HEX);
 delay(10);
Serial.println(ReadReg(REG_MDMCFG4),HEX);
 delay(10);
Serial.println(ReadReg(REG_MDMCFG3),HEX);
 delay(10);
```

```
Serial.println(ReadReg(REG_MDMCFG2),HEX);
 delay(10);
Serial.println(ReadReg(REG_MDMCFG1),HEX);
 delay(10);
Serial.println(ReadReg(REG_MDMCFG0),HEX);
 delay(10);
Serial.println(ReadReg(REG_DEVIATN),HEX);
 delay(10);
Serial.println(ReadReg(REG_MCSM2),HEX);
 delay(10);
Serial.println(ReadReg(REG_MCSM1),HEX);
 delay(10);
Serial.println(ReadReg(REG_MCSM0),HEX);
 delay(10);
Serial.println(ReadReg(REG_FOCCFG),HEX);
 delay(10);

Serial.println(ReadReg(REG_BSCFG),HEX);
 delay(10);
Serial.println(ReadReg(REG_AGCCTRL2),HEX);
 delay(10);
Serial.println(ReadReg(REG_AGCCTRL1),HEX);
 delay(10);
Serial.println(ReadReg(REG_AGCCTRL0),HEX);
 delay(10);
Serial.println(ReadReg(REG_WOREVT1),HEX);
 delay(10);
Serial.println(ReadReg(REG_WOREVT0),HEX);
 delay(10);
Serial.println(ReadReg(REG_WORCTRL),HEX);
 delay(10);
Serial.println(ReadReg(REG_FREND1),HEX);
 delay(10);
Serial.println(ReadReg(REG_FREND0),HEX);
 delay(10);
Serial.println(ReadReg(REG_FSCAL3),HEX);
 delay(10);
```

```
Serial.println(ReadReg(REG_FSCAL2),HEX);
 delay(10);
Serial.println(ReadReg(REG_FSCAL1),HEX);
 delay(10);
Serial.println(ReadReg(REG_FSCAL0),HEX);
 delay(10);
Serial.println(ReadReg(REG_RCCTRL1),HEX);
 delay(10);
Serial.println(ReadReg(REG_RCCTRL0),HEX);
 delay(10);
Serial.println(ReadReg(REG_FSTEST),HEX);
 delay(10);
Serial.println(ReadReg(REG_PTEST),HEX);
 delay(10);
Serial.println(ReadReg(REG_AGCTEST),HEX);
 delay(10);
Serial.println(ReadReg(REG_TEST2),HEX);
 delay(10);
Serial.println(ReadReg(REG_TEST1),HEX);
 delay(10);
Serial.println(ReadReg(REG_TEST0),HEX);
 delay(10);
*/
Serial.println(ReadReg(REG_PARTNUM),HEX);
 delay(1000);
Serial.println(ReadReg(REG_VERSION),HEX);
 delay(1000);
Serial.println(ReadReg(REG_FREQEST),HEX);
 delay(1000);
Serial.println(ReadReg(REG_LQI),HEX);
 delay(1000);
Serial.println(ReadReg(REG_RSSI),HEX);
 delay(1000);
Serial.println(ReadReg(REG_MARCSTATE),HEX);
 delay(1000);
Serial.println(ReadReg(REG_WORTIME1),HEX);
 delay(1000);
```

```
 Serial.println(ReadReg(REG_WORTIME0),HEX);
  delay(1000);
 Serial.println(ReadReg(REG_PKTSTATUS),HEX);
  delay(1000);
 Serial.println(ReadReg(REG_VCO_VC_DAC),HEX);
  delay(1000);
 Serial.println(ReadReg(REG_TXBYTES),HEX);
  delay(1000);
 Serial.println(ReadReg(REG_RXBYTES),HEX);
  delay(1000);
 Serial.println(ReadReg(REG_RCCTRL1_STATUS),HEX);
  delay(1000);
 Serial.println(ReadReg(REG_RCCTRL0_STATUS),HEX);
  delay(1000);

}
```

2. COMMS: Valores de direcciones de registros.

```
/* Sync word qualifier mode = 30/32 sync word bits detected */
/* CRC autoflush = false */
/* Channel spacing = 199.951172 */
/* Data format = Normal mode */
/* Data rate = 2.39897 */
/* RX filter BW = 203.125000 */
/* Preamble count = 4 */
/* Whitening = false */
/* Address config = No address check */
/* Carrier frequency = 2432.999908 */
/* Device address = 0 */
/* TX power = 0 */
/* Manchester enable = false */
/* CRC enable = true */
/* Deviation = 38.085938 */
/* Packet length mode = Variable packet length mode. Packet length configured by the first byte after sync word */
/* Packet length = 255 */
/* Modulation format = 2-FSK */
/* Base frequency = 2432.999908 */
```

```
/* Modulated = true */
/* Channel number = 0 */
/* PA table */
#define PA_TABLE {0xFE,0x00,0x00,0x00,0x00,0x00,0x00,0x00,}
/*****************************************************************
 *  SmartRF Studio(tm) Export
 *
 *  Radio register settings specifed with C-code
 *  compatible #define statements.
 *
 *  RF device: CC2500
 *
 *****************************************************************/

#define REG_IOCFG2          0x0000
#define REG_IOCFG1          0x0001
#define REG_IOCFG0          0x0002
#define REG_FIFOTHR         0x0003
#define REG_SYNC1           0x0004
#define REG_SYNC0           0x0005
#define REG_PKTLEN          0x0006
#define REG_PKTCTRL1        0x0007
#define REG_PKTCTRL0        0x0008
#define REG_ADDR            0x0009
#define REG_CHANNR          0x000A
#define REG_FSCTRL1         0x000B
#define REG_FSCTRL0         0x000C
#define REG_FREQ2           0x000D
#define REG_FREQ1           0x000E
#define REG_FREQ0           0x000F
#define REG_MDMCFG4         0x0010
#define REG_MDMCFG3         0x0011
#define REG_MDMCFG2         0x0012
#define REG_MDMCFG1         0x0013
#define REG_MDMCFG0         0x0014
#define REG_DEVIATN         0x0015
#define REG_MCSM2           0x0016
```

```
#define REG_MCSM1           0x0017
#define REG_MCSM0           0x0018
#define REG_FOCCFG          0x0019
#define REG_BSCFG           0x001A
#define REG_AGCCTRL2        0x001B
#define REG_AGCCTRL1        0x001C
#define REG_AGCCTRL0        0x001D
#define REG_WOREVT1         0x001E
#define REG_WOREVT0         0x001F
#define REG_WORCTRL         0x0020
#define REG_FREND1          0x0021
#define REG_FREND0          0x0022
#define REG_FSCAL3          0x0023
#define REG_FSCAL2          0x0024
#define REG_FSCAL1          0x0025
#define REG_FSCAL0          0x0026
#define REG_RCCTRL1         0x0027
#define REG_RCCTRL0         0x0028
#define REG_FSTEST          0x0029
#define REG_PTEST           0x002A
#define REG_AGCTEST         0x002B
#define REG_TEST2           0x002C
#define REG_TEST1           0x002D
#define REG_TEST0           0x002E
#define REG_PARTNUM         0x0030
#define REG_VERSION         0x0031
#define REG_FREQEST         0x0032
#define REG_LQI             0x0033
#define REG_RSSI            0x0034
#define REG_MARCSTATE       0x0035
#define REG_WORTIME1        0x0036
#define REG_WORTIME0        0x0037
#define REG_PKTSTATUS       0x0038
#define REG_VCO_VC_DAC      0x0039
#define REG_TXBYTES         0x003A
#define REG_RXBYTES         0x003B
#define REG_RCCTRL1_STATUS  0x003C
```

```
#define REG_RCCTRL0_STATUS   0x003D
```

3. COMMS: Valores a configurar de los registros.

```
/* Sync word qualifier mode = 30/32 sync word bits detected */
/* CRC autoflush = false */
/* Channel spacing = 199.951172 */
/* Data format = Normal mode */
/* Data rate = 2.39897 */
/* RX filter BW = 203.125000 */
/* Preamble count = 4 */
/* Whitening = false */
/* Address config = No address check */
/* Carrier frequency = 2432.999908 */
/* Device address = 0 */
/* TX power = 0 */
/* Manchester enable = false */
/* CRC enable = true */
/* Deviation = 38.085938 */
/* Packet length mode = Variable packet length mode. Packet length configured by the first byte after sync word */
/* Packet length = 255 */
/* Modulation format = 2-FSK */
/* Base frequency = 2432.999908 */
/* Modulated = true */
/* Channel number = 0 */
/* PA table */
#define PA_TABLE {0xfe,0x00,0x00,0x00,0x00,0x00,0x00,0x00,}
/*****************************************************************
 *  SmartRF Studio(tm) Export
 *
 *  Radio register settings specifed with C-code
 *  compatible #define statements.
 *
 *  RF device: CC2500
 *
 *****************************************************************/
```

```
#define VAL_IOCFG2          0x29
#define VAL_IOCFG1          0x2E
#define VAL_IOCFG0          0x06
#define VAL_FIFOTHR         0x07
#define VAL_SYNC1           0xD3
#define VAL_SYNC0           0x91
#define VAL_PKTLEN          0xFF
#define VAL_PKTCTRL1        0x04
#define VAL_PKTCTRL0        0x05
#define VAL_ADDR            0x00
#define VAL_CHANNR          0x00
#define VAL_FSCTRL1         0x0A
#define VAL_FSCTRL0         0x00
#define VAL_FREQ2           0x5D
#define VAL_FREQ1           0x44
#define VAL_FREQ0           0xEC
#define VAL_MDMCFG4         0x2D
#define VAL_MDMCFG3         0x3B
#define VAL_MDMCFG2         0x03
#define VAL_MDMCFG1         0x73
#define VAL_MDMCFG0         0x3B
#define VAL_DEVIATN         0x01
#define VAL_MCSM2           0x07
#define VAL_MCSM1           0x30
#define VAL_MCSM0           0x18
#define VAL_FOCCFG          0x1D
#define VAL_BSCFG           0x1C
#define VAL_AGCCTRL2        0xC7
#define VAL_AGCCTRL1        0x00
#define VAL_AGCCTRL0        0xB0
#define VAL_WOREVT1         0x87
#define VAL_WOREVT0         0x6B
#define VAL_WORCTRL         0xF8
#define VAL_FREND1          0xB6
#define VAL_FREND0          0x10
#define VAL_FSCAL3          0xEA
#define VAL_FSCAL2          0x0A
```

```
#define VAL_FSCAL1          0x00
#define VAL_FSCAL0          0x11
#define VAL_RCCTRL1         0x41
#define VAL_RCCTRL0         0x00
#define VAL_FSTEST          0x59
#define VAL_PTEST           0x7F
#define VAL_AGCTEST         0x3F
#define VAL_TEST2           0x88
#define VAL_TEST1           0x31
#define VAL_TEST0           0x0B
#define VAL_PARTNUM         0x80
#define VAL_VERSION         0x03
#define VAL_FREQEST         0x00
#define VAL_LQI             0x00
#define VAL_RSSI            0x00
#define VAL_MARCSTATE       0x00
#define VAL_WORTIME1        0x00
#define VAL_WORTIME0        0x00
#define VAL_PKTSTATUS       0x00
#define VAL_VCO_VC_DAC      0x00
#define VAL_TXBYTES         0x00
#define VAL_RXBYTES         0x00
#define VAL_RCCTRL1_STATUS   0x00
#define VAL_RCCTRL0_STATUS   0x00
```

4. OBC: Programa de envío de imágenes

```
#FUCNTION TO READ THE IMAGE AND RETURN A BYTE ARRAY
def bytes_from_file(filename):
    with open(filename, "rb") as f:
        while True:
            chunk = f.read()
            if chunk:
                for b in chunk:
                    yield b
            else:
                break
```

```
#PROGRAM
#LIBRARIES USED
import serial
import time

#VARIABLE
#STATE:
#           1 PAYLOAD
#           2 COMMS
state = 1

#THE FIRST STATE MAKES 3 ARRAYS OF IMAGES, IF THE IMAGES ARE WHERE THEY ARE SUPPOSED
#TO BE THE CONVERTION TO BYTES WILL START, IF THERES NO IMAGE IT WILL SKIP THE CONVERTION
#AND THE ARRAY WILL HAVE LESS THAN 9 DATA APPENDED. IF THE ARRAYS HAVE LESS THAN 9 DATA
#APPENDED, THE COMMUNICATION WITH COMMS WONT START.
count = 1
#TIME OF CONVERTION
artime = [] #TIME TO CONVERT THE FIRST ARRAY
ar2time = [] #TIME TO CONVERT THE SECOND ARRAY
ar3time = [] #TIME TO CONVERT THE THIRD ARRAY
ar4time = [] #TIME TO CONVERT THE FORTH ARRAY
ar5time = [] #TIME TO CONVERT THE FIFTH ARRAY
ar6time = [] #TIME TO CONVERT THE SIXTH ARRAY
ar7time = [] #TIME TO CONVERT THE SEVENTH ARRAY
ar8time = [] #TIME TO CONVERT THE EIGTH ARRAY
ar9time = [] #TIME TO CONVERT THE NINETH ARRAY
ar10time = [] #TIME TO CONVERT THE NINETH ARRAY
#TIME TO SEND
arrtime = [] #TIME TO SEND THE FIRST ARRAY
arr2time = [] #TIME TO SEND THE SECOND ARRAY
arr3time = [] #TIME TO SEND THE THIRD ARRAY
arr4time = [] #TIME TO SEND THE FORTH ARRAY
arr5time = [] #TIME TO SEND THE FIFTH ARRAY
arr6time = [] #TIME TO SEND THE SIXTH ARRAY
```

```
arr7time = [] #TIME TO SEND THE SEVENTH ARRAY
arr8time = [] #TIME TO SEND THE EIGTH ARRAY
arr9time = [] #TIME TO SEND THE NINETH ARRAY
arr10time = [] #TIME TO SEND THE NINETH ARRAY
#BEGINNING OF SERIAL COMMUNICATION
port = "/dev/ttyAMA0"
baud = 115200
ser = serial.Serial(port, baud, parity=serial.PARITY_NONE, stopbits=serial.STOPBITS_ONE, bytesize=serial.EIGHTBITS, timeout=1)

while count<2:
        #ARRAY
        arr = [] #FIRST IMAGE
        arr2 = [] #SECOND IMAGE
        arr3 = [] #THIRD IMAGE
        arr4 = [] #FORTH IMAGE
        arr5 = [] #FIFTH IMAGE
        arr6 = [] #SIXTH IMAGE
        arr7 = [] #SEVENTH IMAGE
        arr8 = [] #EIGTH IMAGE
        arr9 = [] #NINETH IMAGE
        arr10 = [] #NINETH IMAGE
        #PAYLOAD
        if state == 1:
           #FIRST ARRAY
           #MEASURING TIME
           start = time.time()
           #BEGINNING OF FIRST ARRAY
           #PK33.raw IT'S A RAW IMAGE OF 33KB
           for b in bytes_from_file('/home/pi/PK33.raw'):
              #APPENDED DATA TO ARRAY
              arr.append(b)
           # END OF FIRST ARRAY

           #IF DATA APPENDED TO ARRAY IS MORE THAN 8 IT PRINTS THE TIME LAPSED TO CONVERT
           #THE IMAGE TO A BYTE ARRAY
```

```
 if len(arr)>8:
     end = time.time()
     artime.append((end-start))
#---------------------------------------------
#SECOND ARRAY
#MEASURING TIME
start = time.time()
#BEGINNING OF SECOND ARRAY
#LOOP TO GET DATA
for b in bytes_from_file('/home/pi/PK67.raw'):
   #APPENDED DATA TO ARRAY
   arr2.append(b)
# END OF SECOND ARRAY

#IF DATA APPENDED TO ARRAY IS MORE THAN 8 IT PRINTS THE TIME LAPSED TO
CONVERT
#THE IMAGE TO A BYTE ARRAY
if len(arr2)>8:
   end = time.time()
   ar2time.append((end-start))
#---------------------------------------------
#THIRD ARRAY
#MEASURING TIME
start = time.time()
#BEGINNING OF THIRD ARRAY
#LOOP TO GET DATA
for b in bytes_from_file('/home/pi/PK100.raw'):
   #APPENDED DATA TO ARRAY
   arr3.append(b)
# END OF THIRD ARRAY

#IF DATA APPENDED TO ARRAY IS MORE THAN 8 IT PRINTS THE TIME LAPSED TO
CONVERT
#THE IMAGE TO A BYTE ARRAY
if len(arr2)>8:
   end = time.time()
   ar3time.append((end-start))
```

```
        #---------------------------------------------
        #FOURT ARRAY
        #MEASURING TIME
        start = time.time()
        #BEGINNING OF FIRST ARRAY
        #LOOP TO GET DATA
        for b in bytes_from_file('/home/pi/PK200.raw'):
            #APPENDED DATA TO ARRAY
            arr4.append(b)
        # END OF FIRST ARRAY

        #IF DATA APPENDED TO ARRAY IS MORE THAN 8 IT PRINTS THE TIME LAPSED TO
CONVERT
        #THE IMAGE TO A BYTE ARRAY
        if len(arr4)>8:
            end = time.time()
            ar4time.append((end-start))
        #---------------------------------------------
        #FIFTH ARRAY
        #MEASURING TIME
        start = time.time()
        #BEGINNING OF SECOND ARRAY
        #LOOP TO GET DATA
        for b in bytes_from_file('/home/pi/PK500.raw'):
            #APPENDED DATA TO ARRAY
            arr5.append(b)
        # END OF SECOND ARRAY

        #IF DATA APPENDED TO ARRAY IS MORE THAN 8 IT PRINTS THE TIME LAPSED TO
CONVERT
        #THE IMAGE TO A BYTE ARRAY
        if len(arr5)>8:
            end = time.time()
            ar5time.append((end-start))
        #---------------------------------------------
        #SIXTH ARRAY
        #MEASURING TIME
```

```
        start = time.time()
        #BEGINNING OF THIRD ARRAY
        #LOOP TO GET DATA
        for b in bytes_from_file('/home/pi/PKM1.raw'):
            #APPENDED DATA TO ARRAY
            arr6.append(b)
        # END OF THIRD ARRAY

        #IF DATA APPENDED TO ARRAY IS MORE THAN 8 IT PRINTS THE TIME LAPSED TO
CONVERT
        #THE IMAGE TO A BYTE ARRAY
        if len(arr6)>8:
            end = time.time()
            ar6time.append((end-start))
        #---------------------------------------------
        #SEVENTH ARRAY
        #MEASURING TIME
        start = time.time()
        #BEGINNING OF FIRST ARRAY
        #LOOP TO GET DATA
        for b in bytes_from_file('/home/pi/PKM3.raw'):
            #APPENDED DATA TO ARRAY
            arr7.append(b)
        # END OF FIRST ARRAY

        #IF DATA APPENDED TO ARRAY IS MORE THAN 8 IT PRINTS THE TIME LAPSED TO
CONVERT
        #THE IMAGE TO A BYTE ARRAY
        if len(arr7)>8:
            end = time.time()
            ar7time.append((end-start))
        #---------------------------------------------
        #EIGTH ARRAY
        #MEASURING TIME
        start = time.time()
        #BEGINNING OF SECOND ARRAY
        #LOOP TO GET DATA
```

```
        for b in bytes_from_file('/home/pi/PKM5.raw'):
            #APPENDED DATA TO ARRAY
            arr8.append(b)
        # END OF SECOND ARRAY

        #IF DATA APPENDED TO ARRAY IS MORE THAN 8 IT PRINTS THE TIME LAPSED TO
CONVERT
        #THE IMAGE TO A BYTE ARRAY
        if len(arr8)>8:
            end = time.time()
            ar8time.append((end-start))
        #---------------------------------------------
        #NINETH ARRAY
        #MEASURING TIME
        start = time.time()
        #BEGINNING OF SECOND ARRAY
        #LOOP TO GET DATA
        for b in bytes_from_file('/home/pi/PKM7.raw'):
            #APPENDED DATA TO ARRAY
            arr9.append(b)
        # END OF SECOND ARRAY

        #IF DATA APPENDED TO ARRAY IS MORE THAN 8 IT PRINTS THE TIME LAPSED TO
CONVERT
        #THE IMAGE TO A BYTE ARRAY
        if len(arr9)>8:
            end = time.time()
            ar9time.append((end-start))
        #---------------------------------------------
        #NINETH ARRAY
        #MEASURING TIME
        start = time.time()
        #BEGINNING OF SECOND ARRAY
        #LOOP TO GET DATA
        for b in bytes_from_file('/home/pi/PKM7.raw'):
            #APPENDED DATA TO ARRAY
            arr10.append(b)
```

```
        # END OF SECOND ARRAY

        #IF DATA APPENDED TO ARRAY IS MORE THAN 8 IT PRINTS THE TIME LAPSED TO
CONVERT
        #THE IMAGE TO A BYTE ARRAY
        if len(arr10)>8:
           end = time.time()
           ar10time.append((end-start))
        #-----------------------------------------------
        #IF DATA IS MORE THAN 8 BYTES IT WILL END THE LOOP
        if len(arr)>9:
                if len(arr2)>9:
                if len(arr3)>9:
                        if len(arr4)>9:
                        if len(arr5)>9:
                        if len(arr6)>9:
                        if len(arr7)>9:
                                if len(arr8)>9:
                                if len(arr9)>9:
                                        if len(arr10)>9:
                                                                                state = 2;
           #IN CASE IT'S NOT DONE WITH THE CONVERTION
           else:
              print ("THE CONVERTION DID NOT FINISHED.")
       #-------------------------------------------------------------------------
       #COMMS
       if state == 2:

          #FIRST ARRAY
          #MEASURE TIME
          start2 = time.time()
          for bser in arr:
             #CONVERTS THE DATA IN BYTES
             env = bytes(bser)
             #SEND THE BYTE
             ser.write(env)
          #END OF DATA
```

```
ser.write('1')
ser.write('a')
ser.write('2')
ser.write('b')
ser.write('3')
ser.write('c')
#TIME OF ENDING OF SERIAL COMMUNICATION
end2 = time.time()
tim = (end2-start2)/60
arrtime.append(tim)
        #----------------------------------------
        #SECOND ARRAY
        #MEASURE TIME
        start2 = time.time()
        for bser in arr2:
          #CONVERTS THE DATA IN BYTES
          env = bytes(bser)
          #SEND THE BYTE
          ser.write(env)
        #END OF DATA
        ser.write('1')
        ser.write('a')
        ser.write('2')
        ser.write('b')
        ser.write('3')
        ser.write('c')
        #TIME OF ENDING OF SERIAL COMMUNICATION
        end2 = time.time()
        tim = (end2-start2)/60
        arr2time.append(tim)
        #----------------------------------------
        #THIRD ARRAY
        #MEASURE TIME
        start2 = time.time()
        for bser in arr3:
          #CONVERTS THE DATA IN BYTES
          env = bytes(bser)
```

```
  #SEND THE BYTE
  ser.write(env)
#END OF DATA
ser.write('1')
ser.write('a')
ser.write('2')
ser.write('b')
ser.write('3')
ser.write('c')
#TIME OF ENDING OF SERIAL COMMUNICATION
end2 = time.time()
tim = (end2-start2)/60
arr3time.append(tim)
#----------------------------------------
#FORTH ARRAY
#MEASURE TIME
start2 = time.time()
for bser in arr4:
  #CONVERTS THE DATA IN BYTES
  env = bytes(bser)
  #SEND THE BYTE
  ser.write(env)
#END OF DATA
ser.write('1')
ser.write('a')
ser.write('2')
ser.write('b')
ser.write('3')
ser.write('c')
#TIME OF ENDING OF SERIAL COMMUNICATION
end2 = time.time()
tim = (end2-start2)/60
arr4time.append(tim)
#----------------------------------------
#FIFTH ARRAY
#MEASURE TIME
start2 = time.time()
```

```
for bser in arr5:
  #CONVERTS THE DATA IN BYTES
  env = bytes(bser)
  #SEND THE BYTE
  ser.write(env)
#END OF DATA
ser.write('1')
ser.write('a')
ser.write('2')
ser.write('b')
ser.write('3')
ser.write('c')
#TIME OF ENDING OF SERIAL COMMUNICATION
end2 = time.time()
tim = (end2-start2)/60
arr5time.append(tim)
#----------------------------------------
#SIXTH ARRAY
#MEASURE TIME
start2 = time.time()
for bser in arr6:
  #CONVERTS THE DATA IN BYTES
  env = bytes(bser)
  #SEND THE BYTE
  ser.write(env)
#END OF DATA
ser.write('1')
ser.write('a')
ser.write('2')
ser.write('b')
ser.write('3')
ser.write('c')
ser.write('d')
#TIME OF ENDING OF SERIAL COMMUNICATION
end2 = time.time()
tim = (end2-start2)/60
arr6time.append(tim)
```

```
#---------------------------------------
#SEVENTH ARRAY
#MEASURE TIME
start2 = time.time()
for bser in arr7:
  #CONVERTS THE DATA IN BYTES
  env = bytes(bser)
  #SEND THE BYTE
  ser.write(env)
#END OF DATA
ser.write('1')
ser.write('a')
ser.write('2')
ser.write('b')
ser.write('3')
ser.write('c')
#TIME OF ENDING OF SERIAL COMMUNICATION
end2 = time.time()
tim = (end2-start2)/60
arr7time.append(tim)
#---------------------------------------
#EIGTH ARRAY
#MEASURE TIME
start2 = time.time()
for bser in arr8:
  #CONVERTS THE DATA IN BYTES
  env = bytes(bser)
  #SEND THE BYTE
  ser.write(env)
#END OF DATA
ser.write('1')
ser.write('a')
ser.write('2')
ser.write('b')
ser.write('3')
ser.write('c')
#TIME OF ENDING OF SERIAL COMMUNICATION
```

```
end2 = time.time()
tim = (end2-start2)/60
arr8time.append(tim)
#---------------------------------------
#NINETH ARRAY
#MEASURE TIME
start2 = time.time()
for bser in arr9:
  #CONVERTS THE DATA IN BYTES
  env = bytes(bser)
  #SEND THE BYTE
  ser.write(env)
#END OF DATA
ser.write('1')
ser.write('a')
ser.write('2')
ser.write('b')
ser.write('3')
ser.write('c')
#TIME OF ENDING OF SERIAL COMMUNICATION
end2 = time.time()
tim = (end2-start2)/60
arr9time.append(tim)
#---------------------------------------
#NINETH ARRAY
#MEASURE TIME
start2 = time.time()
for bser in arr10:
  #CONVERTS THE DATA IN BYTES
  env = bytes(bser)
  #SEND THE BYTE
  ser.write(env)
#END OF DATA
ser.write('1')
ser.write('a')
ser.write('2')
ser.write('b')
```

```
                    ser.write('3')
                    ser.write('c')
                    #TIME OF ENDING OF SERIAL COMMUNICATION
                    end2 = time.time()
                    tim = (end2-start2)/60
                    arr10time.append(tim)
                #----------------------------------------
                #PAYLOAD
                state = 1
                #END OF PROGRAM
                start2 = 0
    count = count+1
ser.close()
#AVERAGE 1
d=0
for tmp in artime:
    d = d+tmp
print "Promedio total conversion 1: ",(d/len(artime))
d=0
for tmp in arrtime:
    d = d+tmp
print "Promedio total envio 1:",(d/len(arrtime))
#AVERAGE 2
d=0
for tmp in ar2time:
   d = d+tmp
print "Promedio total conversion 2: ",(d/len(ar2time))
d=0
for tmp in arr2time:
   d = d+tmp
print "Promedio total envio 2:",(d/len(arr2time))
#AVERAGE 3
d=0
for tmp in ar3time:
   d = d+tmp
print "Promedio total conversion 3:",(d/len(ar3time))
d=0
```

```
for tmp in arr3time:
  d = d+tmp
print "Promedio total envio 3:",(d/len(arr3time))
#AVERAGE 4
d=0
for tmp in ar4time:
  d = d+tmp
print "Promedio total conversion 4:",(d/len(ar4time))
d=0
for tmp in arr4time:
  d = d+tmp
print "Promedio total envio 4:",(d/len(arr4time))
#AVERAGE 5
d=0
for tmp in ar5time:
  d = d+tmp
print "Promedio total conversion 5:",(d/len(ar5time))
d=0
for tmp in arr5time:
  d = d+tmp
print "Promedio total envio 5:",(d/len(arr5time))
#AVERAGE 6
d=0
for tmp in ar6time:
  d = d+tmp
print "Promedio total conversion 6: ",(d/len(ar6time))
d=0
for tmp in arr6time:
  d = d+tmp
print "Promedio total envio 6:",(d/len(arr6time))
#AVERAGE 7
d=0
for tmp in ar7time:
  d = d+tmp
print "Promedio total conversion 7:",(d/len(ar7time))
d=0
for tmp in arr7time:
```

```
  d = d+tmp
print "Promedio total envio 7:",(d/len(arr7time))
#AVERAGE 8
d=0
for tmp in ar8time:
  d = d+tmp
print "Promedio total conversion 8:",(d/len(ar8time))
d=0
for tmp in arr8time:
  d = d+tmp
print "Promedio total envio 8:",(d/len(arr8time))
#AVERAGE 9
d=0
for tmp in ar9time:
  d = d+tmp
print "Promedio total conversion 9:",(d/len(ar9time))
d=0
for tmp in arr9time:
  d = d+tmp
#AVERAGE 10
d=0
for tmp in ar10time:
  d = d+tmp
print "Promedio total conversion 9:",(d/len(ar10time))
d=0
for tmp in arr10time:
  d = d+tmp
print "Promedio total envio 9:",(d/len(arr10time))
```

5. OBC: Programa de recepción de imágenes

```
#LIBRARIES TO BE USED
import serial
import time
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np
```

```
#SERIAL COMMUNICATION
port = '\COM3'
baud = 115200
ser = serial.Serial(port, baud, parity=serial.PARITY_NONE, stopbits=serial.STOPBITS_ONE, bytesize=serial.EIGHTBITS, timeout=1)

#VARIABLES
data=[]
data.append('\\PK33.raw')
data.append('\\PK67.raw')
data.append('\\PK100.raw')
data.append('\\PK200.raw')
data.append('\\PK500.raw')
data.append('\\PM1.raw')
data.append('\\PM3.raw')
data.append('\\PM5.raw')
data.append('\\PM7.raw')
data.append('\\PM10.raw')

#VARIABLE TO BE USED TO KNOW IN WHICH ARRAY HAS TO BE THE RECEIVED DATA
var=0
count = 1
#ARRAYS
arrtime = [] #TIME TO RECEIVE FIRST IMAGE
arr2time = [] #TIME TO RECEIVE SECOND IMAGE
arr3time = [] #TIME TO RECEIVE THIRD IMAGE
arr4time = [] #TIME TO RECEIVE FORTH IMAGE
arr5time = [] #TIME TO RECEIVE FIFTH IMAGE
arr6time = [] #TIME TO RECEIVE SIXTH IMAGE
arr7time = [] #TIME TO RECEIVE SEVENTH IMAGE
arr8time = [] #TIME TO RECEIVE EIGTH IMAGE
arr9time = [] #TIME TO RECEIVE NINETH IMAGE
arr10time = [] #TIME TO RECEIVE TENTH IMAGE
while count<2:
   #ARRAYS
   arr = [] #FIRST IMAGE
```

```
arr2 = [] #SECOND IMAGE
arr3 = [] #THIRD IMAGE
arr4 = [] #FORTH IMAGE
arr5 = [] #FIFTH IMAGE
arr6 = [] #SISTH IMAGE
arr7 = [] #SEVENTH IMAGE
arr8 = [] #EIGTH IMAGE
arr9 = [] #NINETH IMAGE
arr10 = [] #TENTH IMAGE
t=0
inic=1
eDt = 0;
while t<1:
   #FIRST IMAGE
   if t==0:
      #ARRAYS OF DATA
      #DATA RECEIVED FROM SERIAL COMMUNICATION
      if ser.inWaiting():
         if inic==1:
            #MEASURING TIME
            start = time.time()
            inic=2
         serial = ser.read()
         #ARRAYS TO BE USED FOR EACH FILE
         #APPEND DATA TO ARRAY
         arr.append(ord(serial))
         var=0
         if (len(arr)>7):
            if (arr[len(arr)-6]==ord(b'1')):
               if (arr[len(arr)-5]==ord(b'a')):
                  if (arr[len(arr)-4]==ord(b'2')):
                     if (arr[len(arr)-3]==ord(b'b')):
                        if (arr[len(arr)-2]==ord(b'3')):
                           if (arr[len(arr)-1]==ord(b'c')):
                              #IF THE ARRAY IT'S COMPLETED THE LOOP WILL BE OVER AND T
WILL BE INCREASED BY ONE
                              t = t+1
```

```
                        end = time.time()
                        tim = (end-start)/60
                        arrtime.append(tim)
                        del arr[len(arr)-6:len(arr)]


    #SECOND IMAGE
    if t==1:
      #ARRAYS OF DATA
      #DATA RECEIVED FROM SERIAL COMMUNICATION
      if ser.inWaiting():
        if inic==2:
          #MEASURING TIME
          start = time.time()
          inic=3
        serial = ser.read()
        #ARRAYS TO BE USED FOR EACH FILE
        #APPEND DATA TO ARRAY
        arr2.append(ord(serial))
        var=0
        if (len(arr2)>7):
          if (arr2[len(arr2)-6]==ord(b'1')):
            if (arr2[len(arr2)-5]==ord(b'a')):
              if (arr2[len(arr2)-4]==ord(b'2')):
                if (arr2[len(arr2)-3]==ord(b'b')):
                  if (arr2[len(arr2)-2]==ord(b'3')):
                    if (arr2[len(arr2)-1]==ord(b'c')):
                      #IF THE ARRAY IT'S COMPLETED THE LOOP WILL BE OVER AND T
WILL BE INCREASED BY ONE
                      t = t+1
                      end = time.time()
                      tim = (end-start)/60
                      arr2time.append(tim)
                      del arr2[len(arr2)-6:len(arr2)]
    #THIRD IMAGE
    if t==2:
      #ARRAYS OF DATA
```

```
        #DATA RECEIVED FROM SERIAL COMMUNICATION
        if ser.inWaiting():
            if inic==3:
                #MEASURING TIME
                start = time.time()
                inic=4
            serial = ser.read()
            #ARRAYS TO BE USED FOR EACH FILE
            #APPEND DATA TO ARRAY
            arr3.append(ord(serial))
            var=0
            if (len(arr3)>7):
                if (arr3[len(arr3)-6]==ord(b'1')):
                    if (arr3[len(arr3)-5]==ord(b'a')):
                        if (arr3[len(arr3)-4]==ord(b'2')):
                            if (arr3[len(arr3)-3]==ord(b'b')):
                                if (arr3[len(arr3)-2]==ord(b'3')):
                                    if (arr3[len(arr3)-1]==ord(b'c')):
                                        #IF THE ARRAY IT'S COMPLETED THE LOOP WILL BE OVER AND T
WILL BE INCREASED BY ONE
                                        t = t+1
                                        end = time.time()
                                        tim = (end-start)/60
                                        arr3time.append(tim)
                                        del arr3[len(arr3)-6:len(arr3)]
    #FORTH IMAGE
    if t==3:
        #ARRAYS OF DATA
        #DATA RECEIVED FROM SERIAL COMMUNICATION
        if ser.inWaiting():
            if inic==4:
                print ("4")
                #MEASURING TIME
                start = time.time()
                inic=5
            serial = ser.read()
            #ARRAYS TO BE USED FOR EACH FILE
```

```
            #APPEND DATA TO ARRAY
            arr4.append(ord(serial))
            var=0
            if (len(arr4)>7):
               if (arr4[len(arr4)-6]==ord(b'1')):
                  if (arr4[len(arr4)-5]==ord(b'a')):
                     if (arr4[len(arr4)-4]==ord(b'2')):
                        if (arr4[len(arr4)-3]==ord(b'b')):
                           if (arr4[len(arr4)-2]==ord(b'3')):
                              if (arr4[len(arr4)-1]==ord(b'c')):
                                 #IF THE ARRAY IT'S COMPLETED THE LOOP WILL BE OVER AND T
WILL BE INCREASED BY ONE
                                 t = t+1
                                 end = time.time()
                                 tim = (end-start)/60
                                 arr4time.append(tim)
                                 del arr4[len(arr4)-6:len(arr4)]

      #FIFTH IMAGE
      if t==4:
         #ARRAYS OF DATA
         #DATA RECEIVED FROM SERIAL COMMUNICATION
         if ser.inWaiting():
            if inic==5:
               print ("5")
               #MEASURING TIME
               start = time.time()
               inic=6
            serial = ser.read()
            #ARRAYS TO BE USED FOR EACH FILE
            #APPEND DATA TO ARRAY
            arr4.append(ord(serial))
            var=0
            if (len(arr5)>7):
               if (arr5[len(arr5)-6]==ord(b'1')):
                  if (arr5[len(arr5)-5]==ord(b'a')):
                     if (arr5[len(arr5)-4]==ord(b'2')):
```

```
                    if (arr5[len(arr5)-3]==ord(b'b')):
                        if (arr5[len(arr5)-2]==ord(b'3')):
                            if (arr5[len(arr5)-1]==ord(b'c')):
                                #IF THE ARRAY IT'S COMPLETED THE LOOP WILL BE OVER AND T
WILL BE INCREASED BY ONE
                                t = t+1
                                end = time.time()
                                tim = (end-start)/60
                                arr5time.append(tim)
                                del arr5[len(arr5)-6:len(arr5)]
    #SIXTH IMAGE
    if t==5:
        #ARRAYS OF DATA
        #DATA RECEIVED FROM SERIAL COMMUNICATION
        if ser.inWaiting():
            if inic==6:
                print ("6")
                #MEASURING TIME
                start = time.time()
                inic=7
            serial = ser.read()
            #ARRAYS TO BE USED FOR EACH FILE
            #APPEND DATA TO ARRAY
            arr6.append(ord(serial))
            var=0
            if (len(arr6)>7):
                if (arr6[len(arr6)-6]==ord(b'1')):
                    if (arr6[len(arr6)-5]==ord(b'a')):
                        if (arr6[len(arr6)-4]==ord(b'2')):
                            if (arr6[len(arr6)-3]==ord(b'b')):
                                if (arr6[len(arr6)-2]==ord(b'3')):
                                    if (arr6[len(arr6)-1]==ord(b'c')):
                                        #IF THE ARRAY IT'S COMPLETED THE LOOP WILL BE OVER AND T
WILL BE INCREASED BY ONE
                                        t = t+1
                                        end = time.time()
                                        tim = (end-start)/60
```

```
                    arr6time.append(tim)
                    del arr6[len(arr6)-6:len(arr6)]
    #SEVENTH IMAGE
    if t==6:
      #ARRAYS OF DATA
      #DATA RECEIVED FROM SERIAL COMMUNICATION
      if ser.inWaiting():
        if inic==7:
          print ("7")
          #MEASURING TIME
          start = time.time()
          inic=8
        serial = ser.read()
        #ARRAYS TO BE USED FOR EACH FILE
        #APPEND DATA TO ARRAY
        arr7.append(ord(serial))
        var=0
        if (len(arr7)>7):
          if (arr7[len(arr7)-6]==ord(b'1')):
            if (arr7[len(arr7)-5]==ord(b'a')):
              if (arr7[len(arr7)-4]==ord(b'2')):
                if (arr7[len(arr7)-3]==ord(b'b')):
                  if (arr7[len(arr7)-2]==ord(b'3')):
                    if (arr7[len(arr7)-1]==ord(b'c')):
                      #IF THE ARRAY IT'S COMPLETED THE LOOP WILL BE OVER AND T
WILL BE INCREASED BY ONE
                      t = t+1
                      end = time.time()
                      tim = (end-start)/60
                      arr7time.append(tim)
                      del arr7[len(arr7)-6:len(arr7)]
    #EIGTH IMAGE
    if t==7:
      #ARRAYS OF DATA
      #DATA RECEIVED FROM SERIAL COMMUNICATION
      if ser.inWaiting():
        if inic==8:
```

```
            #MEASURING TIME
            start = time.time()
            inic=9
        serial = ser.read()
        #ARRAYS TO BE USED FOR EACH FILE
        #APPEND DATA TO ARRAY
        arr8.append(ord(serial))
        var=0
        if (len(arr8)>7):
            if (arr8[len(arr8)-6]==ord(b'1')):
                if (arr8[len(arr8)-5]==ord(b'a')):
                    if (arr8[len(arr8)-4]==ord(b'2')):
                        if (arr8[len(arr8)-3]==ord(b'b')):
                            if (arr8[len(arr8)-2]==ord(b'3')):
                                if (arr8[len(arr8)-1]==ord(b'c')):
                                    #IF THE ARRAY IT'S COMPLETED THE LOOP WILL BE OVER AND T
WILL BE INCREASED BY ONE
                                    t = t+1
                                    end = time.time()
                                    tim = (end-start)/60
                                    arr8time.append(tim)
                                    del arr8[len(arr8)-6:len(arr8)]
    #NINETH IMAGE
    if t==8:
        #ARRAYS OF DATA
        #DATA RECEIVED FROM SERIAL COMMUNICATION
        if ser.inWaiting():
            if inic==9:
                #MEASURING TIME
                start = time.time()
                inic=0
            serial = ser.read()
            #ARRAYS TO BE USED FOR EACH FILE
            #APPEND DATA TO ARRAY
            arr9.append(ord(serial))
            var=0
            if (len(arr9)>7):
```

```
            if (arr9[len(arr9)-6]==ord(b'1')):
                if (arr9[len(arr9)-5]==ord(b'a')):
                    if (arr9[len(arr9)-4]==ord(b'2')):
                        if (arr9[len(arr9)-3]==ord(b'b')):
                            if (arr9[len(arr9)-2]==ord(b'3')):
                                if (arr9[len(arr9)-1]==ord(b'c')):
                                    #IF THE ARRAY IT'S COMPLETED THE LOOP WILL BE OVER AND T
WILL BE INCREASED BY ONE
                                    t = t+1
                                    end = time.time()
                                    tim = (end-start)/60
                                    arr9time.append(tim)
                                    del arr9[len(arr9)-6:len(arr9)]
    #TENTH IMAGE
    if t==8:
        #ARRAYS OF DATA
        #DATA RECEIVED FROM SERIAL COMMUNICATION
        if ser.inWaiting():
            if inic==9:
                #MEASURING TIME
                start = time.time()
                inic=0
            serial = ser.read()
            #ARRAYS TO BE USED FOR EACH FILE
            #APPEND DATA TO ARRAY
            arr10.append(ord(serial))
            var=0
            if (len(arr10)>7):
                if (arr10[len(arr10)-6]==ord(b'1')):
                    if (arr10[len(arr10)-5]==ord(b'a')):
                        if (arr10[len(arr10)-4]==ord(b'2')):
                            if (arr10[len(arr10)-3]==ord(b'b')):
                                if (arr10[len(arr10)-2]==ord(b'3')):
                                    if (arr10[len(arr10)-1]==ord(b'c')):
                                        #IF THE ARRAY IT'S COMPLETED THE LOOP WILL BE OVER AND T
WILL BE INCREASED BY ONE
                                        t = t+1
```

```
                                end = time.time()
                                tim = (end-start)/60
                                arr10time.append(tim)
                                del arr10[len(arr10)-6:len(arr10)]
    count = count+1
#Closing serial communication
ser.close()

#TIMES
#FIRST IMAGE
d=0
for tmp in arrtime:
    d = d+tmp
print ("Promedio 1: ",(d/len(arrtime)))
#SECOND IMAGE
d=0
for tmp in arr2time:
   d = d+tmp
print ("Promedio 2: ",(d/len(arr2time)))
#THIRD IMAGE
d=0
for tmp in arr3time:
   d = d+tmp
print ("Promedio 3: ",(d/len(arr3time)))
#FORTH IMAGE
d=0
for tmp in arr4time:
   d = d+tmp
print ("Promedio 4: ",(d/len(arr4time)))
#FIFTH IMAGE
d=0
for tmp in arr5time:
   d = d+tmp
print ("Promedio 5: ",(d/len(arr5time)))
#SIXTH IMAGE
d=0
for tmp in arr6time:
```

```
  d = d+tmp
print ("Promedio 6: ",(d/len(arr6time)))
#SEVENTH IMAGE
d=0
for tmp in arr7time:
  d = d+tmp
print ("Promedio 7: ",(d/len(arr7time)))
#EIGTH IMAGE
d=0
for tmp in arr8time:
  d = d+tmp
print ("Promedio 8: ",(d/len(arr8time)))
#NINETH IMAGE
d=0
for tmp in arr9time:
  d = d+tmp
print ("Promedio 9: ",(d/len(arr9time)))
#TENTH IMAGE
d=0
for tmp in arr10time:
  d = d+tmp
print ("Promedio 9: ",(d/len(arr10time)))

#CONVERTION OF DATA TO FILES
#FIRST IMAGE
with open(data[0], 'wb') as file:
   file.write(bin(arr))
#SECOND IMAGE
with open(data[1], 'wb') as file:
   file.write(bytes(arr2))
#THIRD IMAGE
with open(data[2], 'wb') as file:
   file.write(bytes(arr3))
#FORTH IMAGE
with open(data[3], 'wb') as file:
   file.write(str(arr4))
#FIFTH IMAGE
```

```
with open(data[4], 'wb') as file:
   file.write(bytes(arr5))
#SIXTH IMAGE
with open(data[5], 'wb') as file:
   file.write(bytes(arr6))
#SEVENTH IMAGE
with open(data[6], 'wb') as file:
   file.write(str(arr7))
#EIGTH IMAGE
with open(data[7], 'wb') as file:
   file.write(bytes(arr8))
#NINETH IMAGE
with open(data[8], 'wb') as file:
   file.write(bytes(arr9))
#TENTH IMAGE
with open(data[9], 'wb') as file:
   file.write(bytes(arr10))
```

6. OBC: Programa de similitud de imágenes

```
% MATRICES DE IMAGENES de 67KB
B = double(imread('/Users/erickrodas/Google Drive/CubeSat/Phase 3 (2015 - 2016)/OBC/Programs/Images/67k/PK67-Q1.jpg'));
C = double(imread('/Users/erickrodas/Google Drive/CubeSat/Phase 3 (2015 - 2016)/OBC/Programs/Images/67k/PK67-Q6.jpg'));
D = double(imread('/Users/erickrodas/Google Drive/CubeSat/Phase 3 (2015 - 2016)/OBC/Programs/Images/67k/PK67-Q12.jpg'));
fid=fopen('/Users/erickrodas/Google Drive/CubeSat/Phase 3 (2015 - 2016)/OBC/Programs/Images/PK67.raw');
img=uint16(zeros(286,241));
img(:)=fread(fid);
A=double(img');

% CONVERSION A VECTORES DE CADA MATRIZ DE IMAGENES
a = reshape(A, [numel(A), 1]);
b = reshape(B, [numel(B), 1]);
c = reshape(C, [numel(C), 1]);
d = reshape(D, [numel(D), 1]);
```

```
%% ====================================
% MATRICES DE IMAGENES de 1MB
N = double(imread('/Users/erickrodas/Google Drive/CubeSat/Phase 3 (2015 - 2016)/OBC/Programs/Images/1M/PM1-Q1.jpg'));
O = double(imread('/Users/erickrodas/Google Drive/CubeSat/Phase 3 (2015 - 2016)/OBC/Programs/Images/1M/PM1-Q6.jpg'));
P = double(imread('/Users/erickrodas/Google Drive/CubeSat/Phase 3 (2015 - 2016)/OBC/Programs/Images/1M/PM1-Q12.jpg'));
fid=fopen('/Users/erickrodas/Google Drive/CubeSat/Phase 3 (2015 - 2016)/OBC/Programs/Images/PM1.raw');
img=uint16(zeros(1115,941));
img(:)=fread(fid);
M=double(img');

% CONVERSION A VECTORES DE CADA MATRIZ DE IMAGENES
m = reshape(M, [numel(M), 1]);
n = reshape(N, [numel(N), 1]);
o = reshape(O, [numel(O), 1]);
p = reshape(P, [numel(P), 1]);

%% ====================================
% MATRICES DE IMAGENES de 10MB
X = double(imread('/Users/erickrodas/Google Drive/CubeSat/Phase 3 (2015 - 2016)/OBC/Programs/Images/PM10-1.jpg'));
Y = double(imread('/Users/erickrodas/Google Drive/CubeSat/Phase 3 (2015 - 2016)/OBC/Programs/Images/PM10-6.jpg'));
Z = double(imread('/Users/erickrodas/Google Drive/CubeSat/Phase 3 (2015 - 2016)/OBC/Programs/Images/PM10-12.jpg'));
fid=fopen('/Users/erickrodas/Google Drive/CubeSat/Phase 3 (2015 - 2016)/OBC/Programs/Images/PM10.raw');
img=uint16(zeros(3456,2916));
img(:)=fread(fid);
V=double(img');

% CONVERSION A VECTORES DE CADA MATRIZ DE IMAGENES
v = reshape(V, [numel(V), 1]);
```

```
x = reshape(X, [numel(X), 1]);
y = reshape(Y, [numel(Y), 1]);
z = reshape(Z, [numel(Z), 1]);

%% ====================================
% DIFERENCIA ENTRE MATRICES

% CEROS EN MATRIZ DE IMAGEN DE 67KB
RB = (A-B);
RC = (A-C);
RD = (A-D);
ZB = (sum(RB(:)==0)/length(a))* 100
ZC = (sum(RC(:)==0)/length(a))* 100
ZD = (sum(RD(:)==0)/length(a))* 100

% CEROS EN MATRIZ DE IMAGEN DE 1MB
RN = (M-N);
RO = (M-O);
RP = (M-P);
ZN = (sum(RN(:)==0)/length(m))* 100
ZO = (sum(RO(:)==0)/length(m))* 100
ZP = (sum(RP(:)==0)/length(m))* 100

% CEROS EN MATRIZ DE IMAGEN DE 10MB
RX = (V-X);
RY = (V-Y);
RZ = (V-Z);
ZX = (sum(RX(:)==0)/length(v))* 100
ZY = (sum(RY(:)==0)/length(v))* 100
ZZ = (sum(RZ(:)==0)/length(v))* 100
```

XI. GLOSARIO

#PS	–	# Paneles Solares
ADCS	–	Sistema de Control y Determinación de Actitud.
AGI	–	Analytical Graphics, Inc.
Cal Poly	–	Universidad Estatal Politécnica de California.
COM	–	Sistema de Comunicaciones.
COMMS	–	Sistema de Comunicaciones.
DCT	–	Transformada Discreta de Fourier.
EOL	–	End Of Life
EPS	–	Electrical Power System
FFT	–	Transformada Rápida de Fourier.
FIFO	–	Primero en Entrar, Primero en Salir.
GaAs	–	Gallium Arsenide
JPEG	–	Joint Photographic Experts Group.
Li-Ion	–	Lithium-Ion
Li-Po	–	Lithium.Polymer
OBC	–	On-Board Computer.
P Carga	–	Potencia a la cual se está cargando la batería
P/L	–	Payload
PCB	–	Printed Circuit Board
STK	–	Satellite Tool Kit
UART	–	Transmisión y Recepción Asíncrono Universal.