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

DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO PARA LA ESTIMACIÓN DE LA ORIENTACIÓN PARA NANOSATÉLITES EN UN SISTEMA EMBEBIDO CON RESTRICCIONES DE TIEMPO REAL

MEMORIA PARA OPTAR AL TÍTULO DE

INGENIERO CIVIL ELÉCTRICO

FELIPE RAÚL DÍAZ HUENUPÁN

PROFESOR GUÍA:
MARCOS DÍAZ QUEZADA

MIEMBROS DE LA COMISIÓN:
ELIAS OBREQUE SEPULVEDA
FRANCISCO RIVERA SERRANO

SANTIAGO DE CHILE
2023

RESUMEN DE LA MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO
POR: FELIPE RAÚL DÍAZ HUENUPÁN
FECHA: 2023
PROF. GUÍA: MARCOS DÍAZ QUEZADA

DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO PARA LA ESTIMACIÓN DE LA ORIENTACIÓN PARA NANOSATÉLITES EN UN SISTEMA EMBEBIDO CON RESTRICCIONES DE TIEMPO REAL

El control sobre la orientación en satélites tiene una gran influencia en importantes funcionalidades como el apuntamiento de antenas para mejorar comunicaciones, el apuntamiento de paneles solares para cargar baterías, o el apuntamiento de cámaras fotográficas a objetos de interés. Como ejemplo particular, en próximas misiones de lanzamientos de satélites desarrollados por SPEL, se tiene por objetivo apuntar a la Luna para sacar fotografías. Por lo que el sistema de control de orientación se vuelve totalmente indispensable. Los satélites que actualmente se encuentran en operación por SPEL, como el SUCHAI-2, el SUCHAI-3 y el PlanSat, poseen actuadores que permiten girar los satélites, sin embargo, carecen de un sistema de cálculo de orientación, lo que impide realizar un control sobre el mismo.

Los actuales satélites en operación llevan consigo cámaras fotográficas que nos han permitido ver distintos fenómenos del cosmos desde el espacio. Con las fotografías capturadas, en donde se ha logra divisar la Tierra o el Sol, se ha intentado estimar la orientación de los satélites con el objetivo de investigar el movimiento que siguen. Esto es totalmente el paso inverso a lo que en realidad se necesita, lo cual es poder apuntar la cámara a donde se quiera para conseguir una foto deseada. Dado esto, es que se vuelve muy necesario el desarrollo de un sistema que permita el cálculo de la orientación con el objetivo de complementar los sistemas de actuación como las ruedas de reacción o *magnetorquer* y apuntar al desarrollo de un sistema de control de orientación preciso.

El desarrollo de esta memoria abarca la implementación de un algoritmo de estimación de orientación en un sistema embebido que se basa en un filtro de Kalman y utiliza 3 tipos de sensores: un giroscopio, un magnetómetro y sensores de sol fino. La memoria abarca fuertemente el desarrollo de la componente en hardware del sistema e implementa en tiempo real un algoritmo de estimación ya implementado en simuladores de SPEL. El sistema desarrollado logra una buena integración del sensor de giroscopio, pero queda por corregir la integración matemática de los otros 2 sensores, para mejorar los resultados que presentan errores acumulativos.

A mi familia, amigos, amigas,
compañeros y compañeras

Agradecimientos

El presente trabajo representa el apogeo de un largo camino de mucho esfuerzo, lleno de fallos, algunos éxitos y mucho aprendizaje. Un camino que tuve la suerte de recorrer junto a maravillosas personas, familiares y amigos, a quienes dedico este pequeño relato que sin duda no alcanza a describir lo agradecido que estoy con ellos.

En primer lugar, agradezco a mi mamá, Mónica, a quien espero honrar su memoria, estoy seguro de que estaría muy feliz de verme terminar este proceso. Le agradezco a mi papá, Raúl, que ha dedicado su vida a mi hermano y a mi para ayudarnos siempre en todo lo que a podido. Todo su trabajo y esfuerzo fueron fundamentales para llegar a este momento. A mi hermano Gustavo, a quien quiero y admiro muchísimo, le agradezco todo su apoyo y consejos en todo momento de mi vida. A Mari que siempre me ayuda y es muy preocupada.

A mi querida tía Marisol, a mis primos: Elias, Judith, Pame y Camilo, a mis pequeñas sobrinas: Loreto y Leticia. Todos y todas a quienes quiero mucho. Les agradezco por estar en todo momento que necesité, tanto para mi o para mi hermano. Por todos lo consejos y lecciones de vida que me dejan, por todos los recuerdos bonitos que hemos construido. Agradezco a toda mi familia, a los que no alcanzo a nombrar, por todos lo valores que me han dado.

Agradezco enormemente a todos mis amigos y amigas que conocí en la Universidad, quienes me han ayudado a enfrentar muchos desafíos en estos 6 (o 7) años. Los momentos de estudio, las noches sin dormir y los momentos de disfrute y alegría siempre estarán en mis recuerdos. En especial a kike, un gran amigo y compañero de toda la carrera, con quien vivimos muy buenos y muy malos momentos, pero siempre apoyándonos y superándonos. A Geraldine, a quien me ayudó en muchos momentos tanto en la universidad como en la vida, gracias por tanto. A una persona muy especial que conocí casi al final de este proceso, Catalina, por los momentos felices que tenemos.

Agradezco al FabLab, especialmente a Joakin Ugalde y a Daniça Peric quienes me facilitaron una de sus creaciones, lo que hizo posible realizar un parte importante de este trabajo. Los momentos en el FabLab, particularmente durante las competencias del *Beauchef Robotics Challenge*, fueron siempre muy entretenidas y llenas de aprendizaje, por lo que también agradezco a todos sus organizadores en sus hasta ahora tres versiones.

Agradezco al profesor Marcos Díaz por todos sus consejos y por darme la oportunidad de realizar esta memoria. A Elias Obreque y al profesor Francisco Rivera por sus buenas observaciones a este trabajo. A los integrantes del equipo SPEL, con quienes siempre tuvimos buenos momentos, con muchas risas, aprendizajes e interesantes conversaciones.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

1. Introducción

1.1. Motivación

El laboratorio de exploración espacial y planetaria (SPEL, por sus siglas en inglés) de la Universidad de Chile, recientemente construyó y lanzó el primer enjambre de satélites desarrollados en el país. Este enjambre está constituido por los nano-satélites: SUCHAI-2, SUCHAI-3 y PlanSat. Cada uno de estos cuenta con distintos *payloads*, los cuales son principalmente 4: *computador a bordo*, *sistema de comunicación*, *sistema de energía* y *sistema de control de orientación*. Este último se divide en dos. Por una parte, un sistema de actuación que permite rotar el satélite sobre sus propios ejes en cualquier dirección; por otra parte, algoritmos de estimación de orientación que permiten determinar la orientación del satélite respecto a un sistema de referencia. Actualmente, los satélites lanzados poseen sistemas de actuación como las ruedas de reacción y *magnetorquers* que permiten girar el satélite sobre cualquiera de sus 3 ejes. Sin embargo, se carece de una estimación efectiva de la orientación en tiempo real, lo que impide hasta el momento, desarrollar un sistema de control de orientación preciso.

La importancia de un control sobre la orientación tiene influencia en varias de las funcionalidades de los satélites como lo son: la orientación de antenas para mejoramiento de la comunicación tanto con la estación terrena como con otros satélites, el apuntamiento de cámaras para tomar fotografías a determinados objetivos, el apuntamiento de paneles solares para recargar baterías, entro otros. Por lo tanto, es necesario el desarrollo de un sistema de control de orientación.

Dada estas condiciones, el trabajo de esta memoria consiste en desarrollar e implementar un algoritmo de estimación de la orientación para nuevos satélites construidos por SPEL que podrían ser lanzados en nuevas misiones. Para, de esta forma, apuntar al desarrollo de un sistema de control de orientación completo, vinculando el sistema de ruedas de reacción y *magnetorquer* con el algoritmo a implementar.

1.2. Objetivos

1.2.1. Objetivo General

Diseñar e implementar un algoritmo para la estimación de la orientación para nano-satélites en un sistema embebido con restricciones de tiempo real. Esto, con el objetivo de complementar el sistema de control de orientación de los satélites de SPEL. Este desarrollo en conjunto con mecanismos de actuación, como las *ruedas de reacción* y los *magnetorquer*, permitirían orientar en el espacio satélites tipo *cubesat*.

1.2.2. Objetivos Específicos

- Evaluar el hardware a utilizar según las restricciones del problema. En particular, evaluar el microcontrolador o *single board computer* (SBC) a utilizar para ejecutar el algoritmo a implementar, considerando que la complejidad de este implica un alto uso de memoria.
- Probar, conceptualmente, el algoritmo de estimación de orientación que se encuentra implementado en los simuladores de SPEL (*spacecraft simulator*)
- Implementación del algoritmo en tiempo real en un sistema embebido que utilice sensores reales.
- Desarrollo de un satélite de prueba tipo *cubesat* que concentre el hardware utilizado y permita la realización de pruebas.
- Utilización de plataforma de prueba y diseño de una metodología que permita evaluar cuantitativamente el rendimiento del algoritmo de estimación de orientación en Tierra.

2. Marco Teórico

2.1. Sistema de referencia

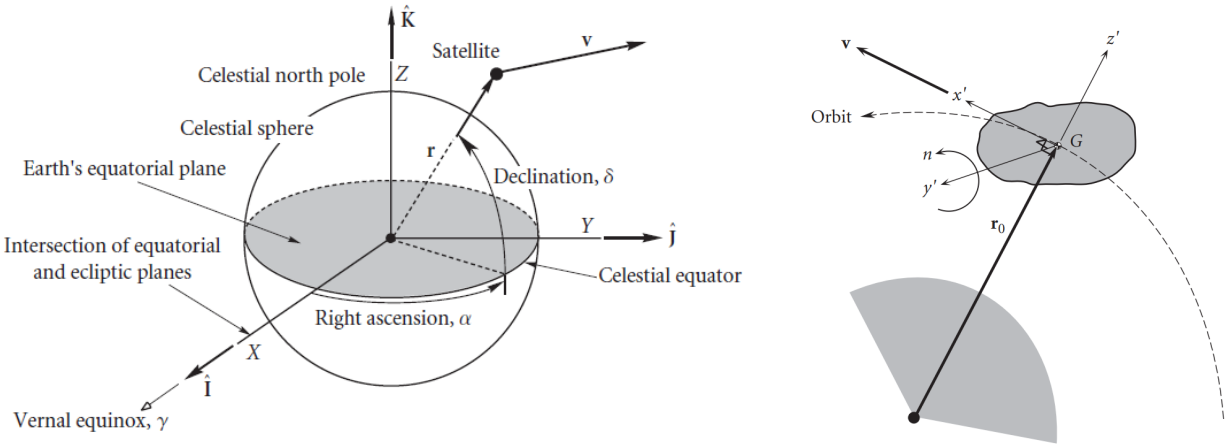
Para poder estimar la orientación del satélite es necesario definir un sistema de referencia. Se debe notar que es necesario definir una referencia para la posición y orientación absoluta del sistema y la ubicación de su origen. En particular, se utiliza un *sistema de referencia inercial* y un *sistema de referencia móvil* (obtenido de [4]), lo cuales se definen a continuación.

2.1.1. Sistema de referencia inercial

Tiene como origen el centro de la tierra y está fijo respecto a las estrellas. Se usa como notación los vectores unitarios $(\hat{i}, \hat{j}, \hat{k})$. Además, para la notación de posición se utiliza el subíndice I, como sigue $(x_I,\ y_I,\ z_I)$. Los vectores $\hat{i}$ y $\hat{j}$ están sobre el plano ecuatorial, perpendicular a este se tiene el vector $\hat{k}$ que coincide con el polo norte geográfico. (ver Figura 2.1 a))

2.1.1.1. Sistema de referencia móvil

Determina la posición y orientación del satélite (cuerpo rígido) con respecto al sistema de referencia inercial. En este caso particular, se definen los ejes de este sistema de referencia acoplados al cuerpo, es decir, el sistema rota en conjunto con el cuerpo. El origen del sistema se elige como el centro de masa del cuerpo, en este caso, el satélite. En la Figura 2.1 b) se observa un sistema de referencia móvil definido respecto a un sistema inercial (con origen en el centro de la Tierra). Además, se utiliza la notación con un subíndice B para la posición con respecto a este sistema de referencia, como sigue $(x_B,\ y_B,\ z_B)$.



(a) Sistema de referencia inercial

(b) Sistema de referencia móvil

Figura 2.1: Sistemas de referencia a utilizar [5]

2.2. Representación de orientación

2.2.1. Cuaterniones

Existen varias formas de representar orientación o rotaciones en el espacio, entre ellas: ángulos de Euler, matrices de rotación y cuaterniones. La ventaja de la representación en cuaterniones es que no tiene indefiniciones en todo su dominio. Además, el álgebra de cuaterniones es más simple y por lo tanto, su resolución tiene un menor costo computacional, lo que favorece su implementación en sistemas embebidos, como es el caso de esta memoria.

Los cuaterniones se pueden definir como una extensión de los números complejos, pero que tienen 3 unidades imaginarias [7] [6], definidas en el espacio de cuaterniones $\mathbb{H}$. Un cuaternión se puede escribir de la forma:

$$Q = q_w + q_x i + q_y j + q_z k \in \mathbb{H} \tag{2.1}$$

En donde, $\{q_w, q_x, q_y, q_z\}$ son números reales y $\{i, j, k\}$ son las unidades imaginarias. Equivalentemente, el mismo cuaternión se puede escribir como un vector de 4 dimensiones:

$$Q = \begin{bmatrix} q_w \\ q_v \end{bmatrix} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} \tag{2.2}$$

En donde q_w es la parte escalar y $q_v = [q_x, q_y, q_z]$ es la parte imaginaria. Es importante determinar el orden en que se escriben sus partes real e imaginarias, pues con una u otra, cambirán el orden de las rotaciónes. En particular, en esta memoria se utiliza la convención de Hamilton [7], que es el orden definido en la Ecuación 2.2, es decir, primero la parte real y luego la imaginaria. De esta forma, las rotaciones se pueden obtener utilizando la *regla de la mano derecha*, de donde deriva el siguiente álgebra de cuaterniones:

$$ij = -ji = k, \quad jk = -kj = i, \quad ki = -ik = j \tag{2.3}$$

Además, los cuaterniones cumple con varias propiedades [7] que permiten su operación como: suma, multiplicación, identidad, conjugado, norma, inverso, entre otras. A continuación, se explicitan el conjugado y la multiplicación.

- **Conjugado:** Dado $q = [q_w, q_v]$, su conjugado se expresa como:

$$q^* = \begin{bmatrix} q_w \\ -q_v \end{bmatrix} \tag{2.4}$$

- **Multiplicación:** Dados $p = [p_x, p_y, p_z, p_w]$ y $q = [q_x, q_y, q_z, q_w]$, se tiene lo siguiente:

$$p \otimes q = \begin{pmatrix} p_w q_W - p_x q_x - p_y p_y - p_z p_z \\ p_w p_x + p_x q_w + p_y q_z - p_z p_y \\ p_w p_y - p_x q_z + p_y q_w + p_z p_x \\ p_w p_z + p_x q_y - p_y q_x + p_z p_w \end{pmatrix} \tag{2.5}$$

2.2.1.1. Magnitud de error

Uno de las etapas importante en el desarrollo de la memoria, será determinar un error de estimación en el algoritmo implementado. Esto es, realizar una comparación entre dos cuaterniones y determinar cuantitativamente una forma de error entre ellos. Para esto, se calcula el ángulo de rotación mínimo para llegar de un cuaternión al otro.

Más en detalle, dados los cuaterniones $p = [p_x, p_y, p_z, p_w]$ y $q = [q_x, q_y, q_z, q_w]$, se calcula un tercer cuaternión z como sigue [7]:

$$z = p^* \otimes q \tag{2.6}$$

Luego, se calcula el ángulo de error (θ_{error}) según la Ecuación 2.7.

$$\theta_{error} = 2 \cdot \arccos(z_w) \tag{2.7}$$

2.2.1.2. Conversión a ángulos de Euler

Una desventaja de la implementación con cuaterniones, es que son poco intuitivos gráficamente. Al ver el valor de un cuaternión, es decir, solo un vector con 4 componentes, es complejo imaginar o intuir hacia a donde apunta. Dado esto, es importante también considerar una conversión del cuaternión a ángulos Euler, que ayuda a la interpretación gráfica de los datos. Esta conversión se muestra en la Ecuación 2.8.

$$\begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix} = \begin{pmatrix} \arctan \frac{2(q_w q_x + q_y q_z)}{1-2(q_x^2+q_y^2)} \\ \arcsin\left(2(q_w q_y - q_z q_x)\right) \\ \arctan \frac{2(q_w q_z + q_x q_y)}{1-2(q_y^2+q_z^2)} \end{pmatrix} \tag{2.8}$$

En donde, se considera lo siguiente para rotaciones respecto al sistema de referencia móvil:

- Ángulo ϕ para rotaciones alrededor de X_B
- Ángulo θ para rotaciones alrededor de Y_B
- Ángulo ψ para rotaciones alrededor de Z_B

2.3. Simuladores de SPEL

2.3.1. Spacecraft Simulator

El simulador *Spacecraft* fue desarrollado por ingenieros e integrantes de SPEL. Tiene por objetivo simular la dinámica y cinemática del movimiento de satélites en el espacio, incluyendo también un ambiente espacial simulado con cuerpos celestes como la Tierra, Marte, la Luna y el Sol. El simulador permite probar distintos algoritmos que están en fase de desarrollo para que puedan ser evaluados antes de implementarlos en un determinado hardware.

Este simulador genera un conjunto de datos (archivo .CSV) con variables de interés que ayudan a comprender la órbita y movimientos del satélite simulado. Parte de las variables generadas, son las siguientes:

- Velocidad angular
- Velocidad lineal
- Torque
- Cuaterniones
- Campo Magnético
- Posición
- Ground track[1]

Así mismo, dentro de los componentes posibles a simular se encuentran los siguientes:

- **Sensores**: Giroscopio, magnetómetro, sensores de sol, *Star Tracker*[2].
- **Actuadores**: Ruedas de reacción, *magnetorquer*.
- **Componentes**: Paneles solares, baterías.
- **Ambiente**: Radiación solar, campo magnético, atmósfera y componentes de disturbio.

2.3.1.1. Modos de iniciación de órbita

Actualmente, hay implementados dos modos de iniciación para la simulación. Estos modos definen las condiciones iniciales para la propagación de la órbita del satélite alrededor de la Tierra. Se definen a continuación:

- **TLE:** por sus siglas en Inglés (*Two-Line Element Sets*). Es un conjunto de datos de 2 lineas que contiene información sobre la órbita de un determinado satélite en una época específica. La Figura 2.2 muestra el TLE del cubesat SUCHAI I disponible en el simulador para estudiar su órbita. Este TLE en particular es del 27 de enero de 2020. Dado que el formato TLE es un estándar utilizado internacionalmente, el simulador fue desarrollado para que funcione con un TLE de cualquier satélite.

[1] Trayectoria de la órbita proyectada sobre la Tierra. Puede ser representada por Latitud y Longitud.
[2] Sistema de estimación de actitud basada en la toma de fotos y comparaciones con un catálogo de estrellas

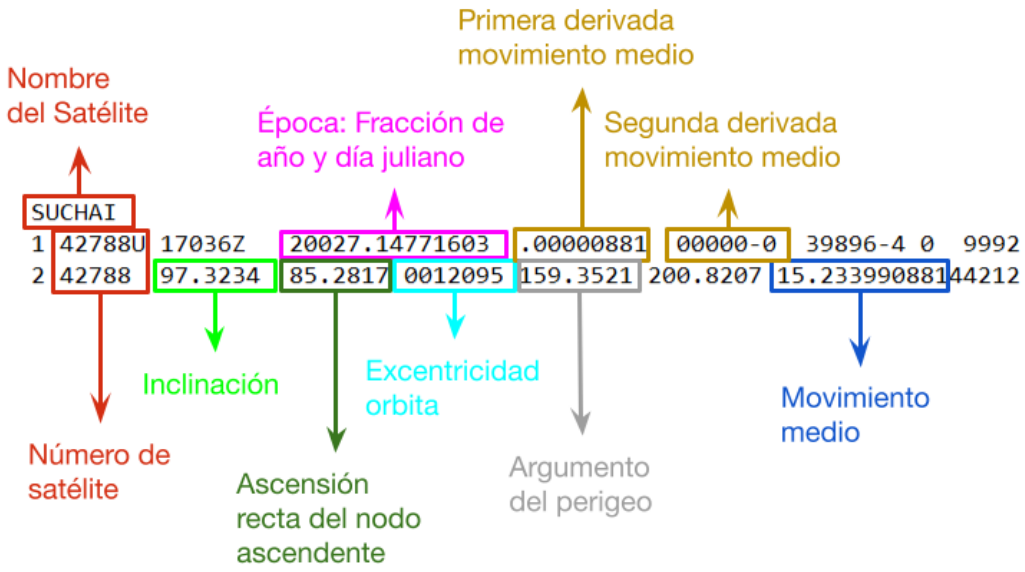


Figura 2.2: TLE de satélite SUCHAI I. Se detallas los parámetros principales mostrados en un TLE de datos órbitales.

- **Condiciones iniciales:** si no se utiliza un TLE, es posible propagar una órbita particular a partir de dos variables que debe definir el usuario: posición inicial y velocidad inicial del satélite. En función de estos parámetros se puede elegir el tipo de órbita (polar o ecuatorial), la excentricidad, la altura, el periodo, entre otras variables.

2.3.2. Herramientas de Visualización

Dado los datos generados por *Spacecraft Simulator*, también existen simuladores que permiten la visualización de la órbita simulada. La Figura 2.3 muestra un ejemplo de visualización. A la izquierda se muestra parte de la trayectoria de la órbita sobre la tierra, a la derecha se muestra la orientación del satélite.

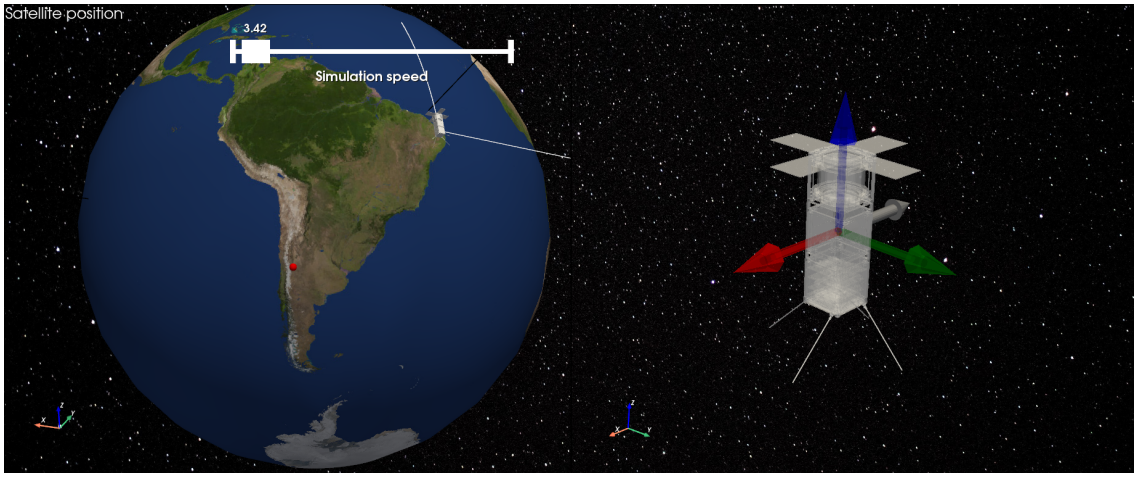


Figura 2.3: Ejemplo de visualización de órbita utilizando las herramientas de visualización de SPEL.

2.4. Error State Kalman Filter

La Figura 2.4 muestra el algoritmo *Error State Kalman Filter* (ESKF, por sus siglas en Inglés) implementado en *Spacecraft Simulator* basado en [7], [8], [9]. Este utiliza cuatro sensores: giroscopio, magnetómetro, sensor de sol grueso y sensor de sol fino. Es importante hacer una distinción entre dos tipos de sensores que se utilizan. Por un lado, los sensores intrínsecos, que únicamente miden estados internos del sistema, como el es el caso del giroscopio, que mide la velocidad angular del satélite u objeto. Por otro lado, los sensores extrínsecos, que miden estados externos del sistema, como el caso del magnetómetro y los sensores del sol. El magnetómetro mide el campo magnético de la Tierra, que es algo totalmente externo al satélite que solo depende de la posición y orientación relativa entre la Tierra y el satélite. Al igual que los sensores de sol, que miden la forma con la que llega el sol al satélite y que, por lo tanto, también es externo al satélite.

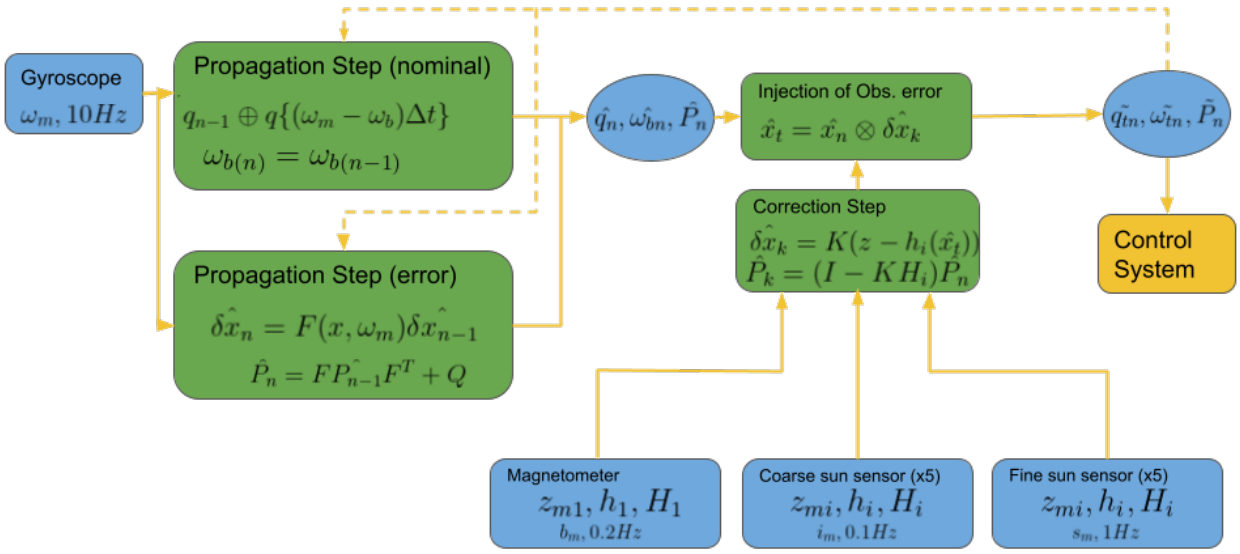


Figura 2.4: Diagrama de bloques de algoritmo de estimación implementado en el simulador basado en *Error State Kalman Filter (ESKF)*

El paso *Propagation Step* toma un cuaternión inicial arbitrario, en este caso un cuaterión unitario $q_0 = [0, 0, 0, 1]$ y lo propaga en el tiempo utilizando las velocidades angulares obtenidas por el giroscopio, al mismo tiempo estima el *bias* $(\omega_b(n))$ del giroscopio. En este paso se calculan los cuaterniones q_{n-1} que depende del cuaternión anterior (q_n), las velocidades angulares y del tiempo entre una medición y otra del sensor. Hasta este punto se tiene una estimación simple sobre el cuaternión, ya que esta solo considera la velocidad angular, que además contiene un *bias* significativo que induce un error acumulativo. Por otro lado, tampoco se consideran posibles perturbaciones externas que puedan haber en el sistema. El cuaternión q_{n-1} y el *bias* $\omega_b(n)$ definen la variable de estado $\hat{x}_n = [q_{n-1}, \omega_b(n)] = [q_x, q_y, q_z, q_w, \omega_{b1}, \omega_{b2}, \omega_{b3}]$ en un vector de dimensiones 7x1.

El paso *Propagetion step* realiza la actualización de la propagación del estado del error ($\delta\hat{x}_n$) que depende de la matriz de transición F [7] y del estado del error anterior (($\delta\hat{x}_{n-1}$)). Calculando también la matriz de covarianzas ($\hat{P}_n$). Luego, se combina el estado nominal $\hat{x}_n$ y el estado del error $\delta\hat{x}_k$ en el paso *Injection of observed error*.

Se continua con el paso *Correction Step*, que es el paso que mejora realmente la estimación utilizando los sensores: magnetómetro y sensores de sol. Esta calcula una corrección para la estimación actual, que esta dada por la ganancia de Kalman K multiplicada por $(z - h_i(\hat{x}_t))$. En donde z corresponde a la medición actual de uno de los sensores (magnetómetro o sensores de sol) y $h_i()$ corresponde al "valor esperado" o modelo del sensor en particular. En especifico, el modelo del campo magnético de la Tierra y el modelo de posición solar, que dependen de la posición y orientación satélite en un momento determinado. Es decir, dependiendo de la posición y orientación del satélite, se tendrá un valor esperado de campo magnético y de dirección de luz solar el cual se compara con la medición del sensor real correspondiente. Así mismo, se actualiza el valor de la matriz de covarianzas $\hat{P}_k$, en función del jacobiano de la función $h_i()$ y de K.

Para cada medición de cada sensor, se ejecuta el paso de correción, con lo que se actualiza la estimación de la variable de estado $\hat{x}_t$, que es la que contiene el cuaternión y el *bias* estimado. Finalmente, la variable de salida del ESKF, que es la orientación (cuaternión), se espera que se conecte con el sistema de control de orientación del satélite.

2.5. Revisión de Hardware

2.5.1. Procesador

Un *Single Board Computer* o SBC es un computador diseñado en una sola placa de circuito impreso, que contiene un microprocesador, memoria y periféricos. En particular, para el desarrollo de esta memoria de desea utilizar una *Raspberry* (o RPI), lo cual es un hardware que es parte los satélites ya implementados por SPEL. A este tipo de SBC es posible instalarle un sistema operativo basado en Linux el cual es: Raspbian. De esta forma, la *Raspberry* puede ser programada en lenguaje como Python, C/C++, java, entre otros. En especifico, se desea utilizar el modelo Raspberry Pi 3B+, la cual tiene las siguientes características:

- CPU 1.4GHz
- Memoria Ram 1GB
- Memoria flash >8GB

Se espera que con las características de esta SBC sea posible resolver el problema sin estar limitado por la capacidad de procesamiento o capacidad de memoria.

2.5.2. Unidad de Medición Inercial

Unidad de Medición Inercial o IMU (por sus siglas en ingles *Inertial Measurement Unit*) es un dispositivo electrónico que permite medir: velocidad angular, aceleración lineal y fuerzas gravitacionales. Esto lo hace a través de los sensores: giroscopio, acelerómetro y magnetómetro. Usualmente, son usados en robótica y aplicaciones de navegación de vehículos autónomos o semi autónomos. En particular, durante el desarrollo de esta memoria se usará el modelo MPU9250 que se muestra en la Figura 2.5. Es deseable utilizar este modelo en específico dado que ya es parte del hardware actual de los satélites desarrollados por SPEL.

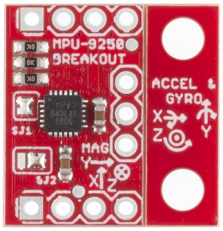


Figura 2.5: Unidad de medición inercial (IMU) [10]

2.5.3. Sensor de Sol Fino

Sensor de sol fino o FSS, por sus siglas en Inglés (*Fine Sun Sensor*), es un sensor ampliamente utilizado por naves de tipo aéreas o espaciales, que tiene por objetivo calcular la dirección con la que llega el sol. Esta información es utilizada en sistemas de navegación o de determinación de orientación. Otro tipo de aplicaciones terrestre son en sistemas meteorológicos y seguidores de sol para el posicionamiento de paneles solares. En la Figura 2.6 se muestra un FSS desarrollado por la empresa *GomSpace* que fue adquirido por SPEL y utilizado en el cubesat SUCHAI II.

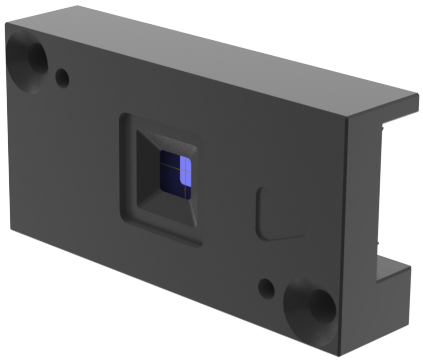
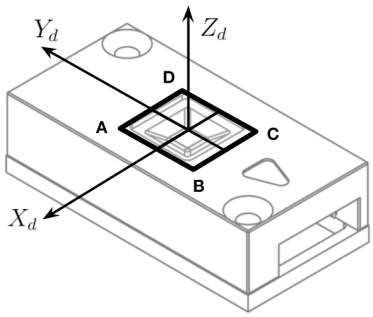
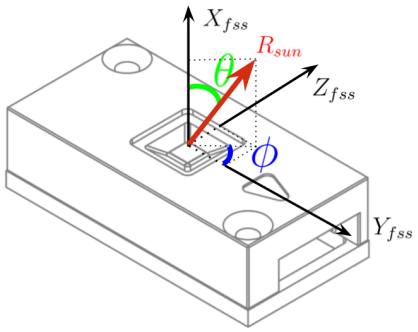


Figura 2.6: Sensor de sol fino desarrollado por *GomSpace*. Utilizado en el cubesat SUCHAI II. En el centro del sensor se posiciona un fotodiodo de 4 cuadrantes que se excita con la luz del sol

El funcionamiento de un FSS se basa en la utilización de 4 fotodiodos o de un fotodiodo de 4 cuadrantes. Cada fotodiodo es excitado por la intensidad de luz que percibe, lo que genera una salida de corriente del orden los micro Ampere (μA). De esta forma, mientras más intensidad lumínica llega a un fotodiodo más corriente se genera en su salida. Además, como los fotodiodos se encuentran equidistantes entre sí, se puede calcular el vector de incidencia de la luz en función de la corriente en la salida de cada fotodiodo.



(a) Cuadrantes del fotodiodo: A, B, C y D. Sistema de referencia del diodo: X_d, Y_d y Z_d.



(b) Sistema de referencia del sensor propio. Los ángulos ϕ y θ definen el ángulo de dirección del sol R_{run}.

Figura 2.7: Diseño *CAD* del FSS de *GomSpace* [11]. Sobre este, se muestran los sistema de referencia correspondientes para calcular el vector de dirección del sol R_{sun}.

La Figura 2.7 muestra un diagrama *CAD* del FSS de *GomSpace* recuperado de su documentación oficial en [11]. En la Figura 2.7 (a) se observan los 4 cuadrantes del fotodoiodo: A, B, C y D. Por otro lado, se muestran el sistema de referencia del diodo [12] representado por X_d, Y_d y Z_d. La Figura 2.7 (b) muestra el sistema de referencia del FSS en el cual se mide el vector de dirección del sol R_{sun}, el cual está definido por los ángulos ϕ y θ. Es relevante notar que ϕ está en un rango de $[-180^o, 180^o]$ y θ está en un rango de $[0^o, 90^o]$. Además, el rango de θ representa el campo de visión del sensor. Si este logra estar en el rango señalado, significa que el sensor es capaz de leer todas las direcciones posibles en el plano. Esto es particularmente difícil de lograr en sensores reales. Este FSS logra un campo de visión máximo de 60^o según la especificaciones de la Tabla 2.1.

Tabla 2.1: Campo de visión y precisión de FSS de *GomSpace*.

Campo de visión	Precisión
$[0^o, 45^o]$	$\pm 0.5^o$
$[45^o, 60^o]$	$\pm 2^o$

Entonces, dados las mediciones de corriente en los cuadrantes A, B, C y D del fotodiodo, se realiza el procedimiento que sigue a continuación para calcular el vector de incidencia del sol (basado en [12]). En donde, adicionalmente, se consideran los valores: h, x_0, y_0, δ, como parámetros de calibración calculados empíricamente con cada sensor real.

Se define la matriz de rotación:

$$T = \begin{pmatrix} cos(\delta) & sen(\delta) \\ -sen(\delta) & cos(\delta) \end{pmatrix} \tag{2.9}$$

Luego, se definen las coordenadas de la proyección del sol sobre los fotodiodos:

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = T \cdot \begin{pmatrix} \frac{A+B-C-D}{A+B+C+D} + x_0 \\ \frac{A+D-B-C}{A+B+C+D} + y_0 \end{pmatrix} \tag{2.10}$$

Continuando, se calcula la dirección del vector de incidencia del sol en dos ángulos, ϕ y θ:

$$\phi = atan(\frac{x_d}{y_d}) \tag{2.11}$$

$$\theta = atan(\frac{\sqrt{x_d^2}}{y_d^2}) \tag{2.12}$$

Finalmente, la dirección del vector de sol se calcula como sigue:

$$R_{sun} = \begin{pmatrix} cos(\theta) \\ sen(\theta) \cdot cos(\phi)) \\ sen(\theta) \cdot sen(\phi)) \end{pmatrix} = \begin{pmatrix} x_{fss} \\ y_{fss} \\ z_{fss} \end{pmatrix} \tag{2.13}$$

3. Estado del Arte

3.1. Estimación de Orientación

La utilización de algoritmos de estimación de orientación es altamente usada en robótica, en especifico, en la programación de vehículos no tripulados. La determinación de la orientación es fundamental para ejecutar un buena navegación en distintos tipos de aplicaciones. Para el caso particular de satélites, el control de orientación influye en muchas funciones. Por ejemplo, para mejorar las comunicaciones es necesario apuntar antenas en determinadas direcciones. También se utiliza para el control de carga de baterías apuntando paneles solares hacia el sol cuando sea necesario y también para apuntar cámaras fotográficas para estudiar determinados fenómenos del cosmos.

Uno de los algoritmos más utilizados en la literatura para estimar la orientación, es el Filtro de Kalman y todas sus derivadas, como lo son el Filtro de Kalman extendido y el filtro de Kalman indirecto con fusión de sensores. Un filtro de Kalman es un método que tiene como finalidad identificar el estado no medible de un sistema dinámico lineal o no lineal. También sirve en los casos en que el sistemas está sometido a ruido blanco auditivo, como en el caso de las mediciones de una Unidad de medición inercial (IMU, por sus siglas en Inglés). Es un algoritmo recursivo que permite estimar en tiempo real, utilizando únicamente: mediciones actuales del sistema, el estado previo y una matriz de incertidumbre.

En [1] se utiliza un filtro de Kalman para estimar la orientación de un dispositivo médico que ayuda a la detección temprana de ciertas enfermedades en ancianos. En [2] se aplica un filtro de Kalman extendido para estimar la orientación de vehículos autónomos submarinos, con el objetivo de mejorar la navegación. En [3] se aplica un filtro de Kalman extendido utilizando de sensores de giroscopio y acelerómetro para la estimación de orientación de cuadricópteros con el objetivo de mejorar la estabilidad en vuelo. Todos estos ejemplos tienen en común que utilizan un filtro de Kalman para estimar la orientación usando giroscopios y acelerómetros como entrada.

La estimación de orientación estando dentro de la Tierra es un problema que ya ha sido resuelto. Una de sus soluciones es la implementación de un filtro de Kalman tal como se menciona en los ejemplos anteriores. Uno de los sensores que más información aporta es el acelerómetro. Sin el importar el lugar de la Tierra en que se esté, siempre existirá el vector gravedad apuntando al centro de la Tierra. Por lo tanto, al tener un acelerómetro de 3 ejes, esta información por si sola, ya da indicios de una orientación, al menos en un eje. Al propagar esta información en conjunto con los datos de velocidad angular (Giroscopio), es posible obtener una orientación con buena precisión.

Fuera de la Tierra, específicamente al considerar un satélite en órbita en torno ella, se tiene un ambiente de microgravedad. Esto debido a que los objetos en órbita están en caída libre y por lo tanto, están en un ambiente en que se percibe una gravedad mínima. Dado esta condición, para un *cubesat* en órbita, un acelerómetro no aporta información relevante. Esto provoca que la estimación de orientación para un objeto en órbita sea mucho más compleja que para un objeto dentro de la Tierra. El no poder utilizar acelerómetros, obliga a utilizar otros sensores y otras variables para la estimación de orientación; algunas de ellas son: la inclinación con la que llega el Sol y el campo magnético de la tierra percibido en distintas posiciones.

4. Desarrollo del Sistema

4.1. Pruebas de ESKF en *Spacecraft Simulator*

4.1.1. Condiciones de simulación

Para probar el algoritmo *ESKF* de estimación de orientación implementado en *Spacecraft simulator*, se generan condiciones de simulación específicas. En particular, se genera un ambiente de simulación sin actuadores de ningún tipo, esto es, sin ejercer ningún torque sobre el satélite. Sistemas como las *ruedas de reacción* y *magnetorquer*, se desactivan. Esto genera el ambiente de simulación más básico para lograr una estimación de orientación.

Dado esto, la única fuerza que actúa sobre el satélite es la gravedad, ignorando fuerzas de arrastre secundarias y fuerzas gravitaciones distintas a las ejercidas por la Tierra. Además, considerando que el tamaño del satélite es muy pequeño con respecto a la Tierra, la fuerza gravitacional se ejerce principalmente en el centro de masa G [5]. Por lo tanto, el satélite seguirá un movimiento libre de torque, esto quiere decir, que el torque neto sobre el satélite es cero. En particular, la derivada del momento angular H_G es cero.

$$H_G = 0 \tag{4.1}$$

Luego, cambiado al sistema de referencia móvil, se tiene lo siguiente:

$$\dot{H_{Grel}} + \omega \times H_G = 0 \tag{4.2}$$

Tomando en cuenta la matriz de inercia:

$$\begin{pmatrix} A & 0 & 0 \\ 0 & B & 0 \\ 0 & 0 & C \end{pmatrix} \tag{4.3}$$

Considerando el momento de inercia neto $M_g = 0$, se tienen la llamadas *Ecuaciones de Euler*:

$$A\dot{\omega_x} + (C - B)\omega_z\omega_y = 0 \tag{4.4}$$
$$B\dot{\omega_y} + (A - C)\omega_x\omega_z = 0 \tag{4.5}$$
$$C\dot{\omega_z} + (B - A)\omega_y\omega_x = 0 \tag{4.6}$$

Si, además, se considera un cuerpo como el de la Figura 4.1, en el cual el eje z es un eje de simetría rotacional y que ,además, es un cuerpo con masa uniformemente distribuida. Entonces, se tiene una matriz de inercia con dos de sus componentes iguales. En este caso, las componente de I_{xx} e I_{yy}, es decir, de la Ecuación 4.3 se tiene $A = B$. En particular, esto puede suceder en los cubesat de 1U y 3U, como los desarrollados por SPEL.

Por lo tanto, las ecuaciones de Euler se simplifican a la siguiente expresión:

$$A\dot{\omega}_x + (C - B)\omega_z\omega_y = 0 \tag{4.7}$$
$$B\dot{\omega}_y + (A - C)\omega_x\omega_z = 0 \tag{4.8}$$
$$C\dot{\omega}_z = 0 \tag{4.9}$$

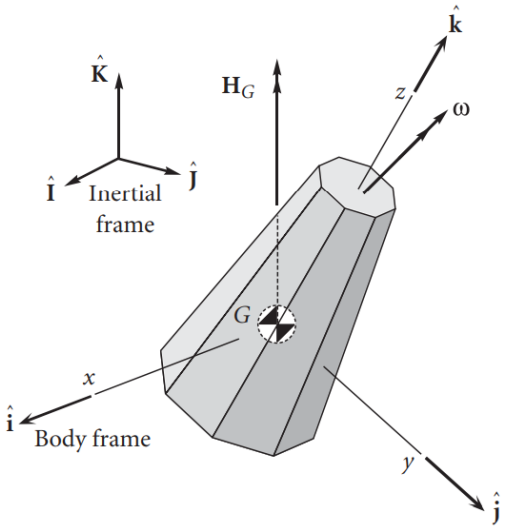


Figura 4.1: Cuerpo simétrico en movimiento libre de torque. Se considera el sistema de referencia móvil acoplado con los ejes x, y, z definidos respecto al sistema inercial con ejes $\hat{I}$, $\hat{J}$ y $\hat{K}$. Con un centro de masa G y un momento angular H_G. Extraído de [5].

De donde se desprende que la velocidad en el eje z es constante.

$$\omega_z = \omega_0 \tag{4.10}$$

Continuando, se define una variable auxiliar para resolver las otras dos ecuaciones diferenciales.

$$\lambda = \frac{A - C}{A}\omega_0 \tag{4.11}$$

Con lo que las soluciones para las velocidades angulares en los ejes x e y son las siguientes:

$$\omega_x = \Omega sen(\lambda t) \tag{4.12}$$
$$\omega_y = \Omega cos(\lambda t) \tag{4.13}$$

En donde, la amplitud Ω se determina en función de las condiciones iniciales del sistema. Este resultado, implica que las velocidades angulares en los ejes x e y tiene una forma sinusoidal y están desfasadas en 90^o (dada las propiedades trigonométricas del seno y el coseno). Y la velocidad angular en el eje z es constante.

4.1.2. Pruebas de simulación

Con las condiciones descritas se ejecuta una simulación. Para esto se utilizan los datos de TLE mostrados en el ejemplo de la Figura 2.2 del cubesat SUCHAI I. Este sigue una órbita terrestre baja (LEO, por sus siglas en inglés, *Low Earth Orbit*) manteniéndose a una altura de 500 km, aproximadamente. Más en particular, sigue una órbita polar, lo que implica que el satélite pasa por los polos del planeta. En la Figura 4.2 se muestra la trayectoria de la órbita simulada.

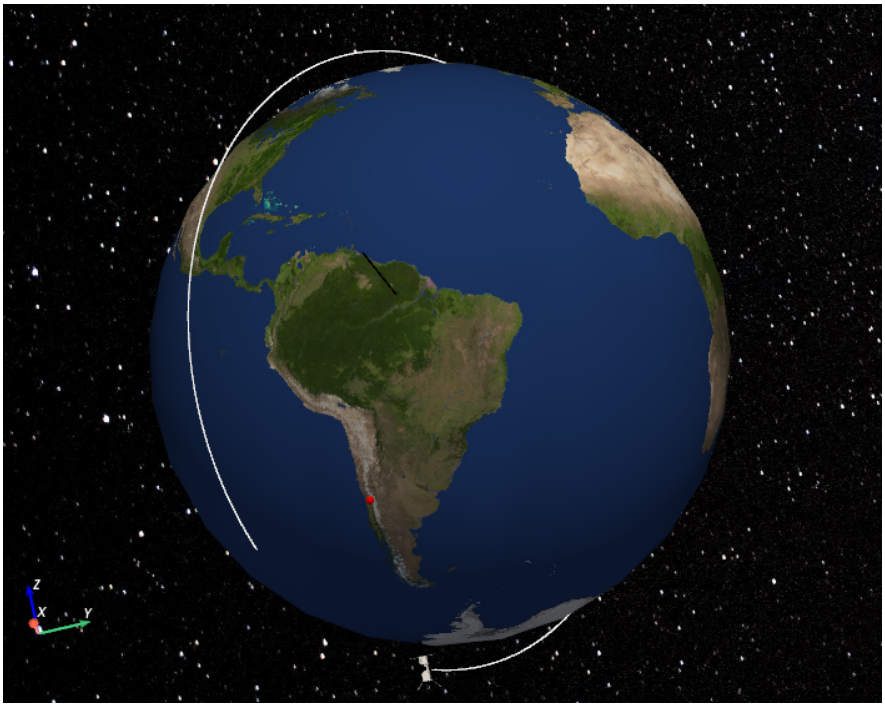
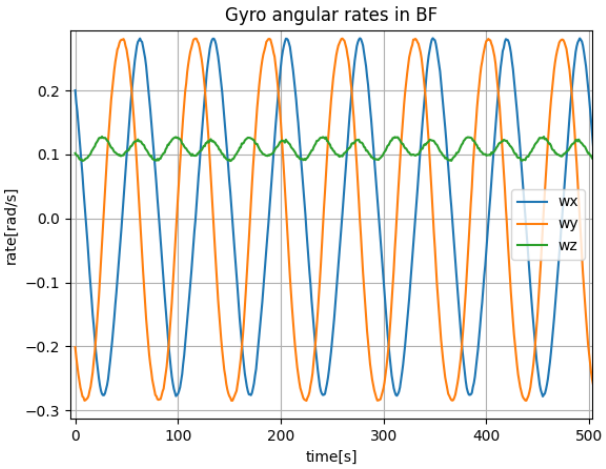
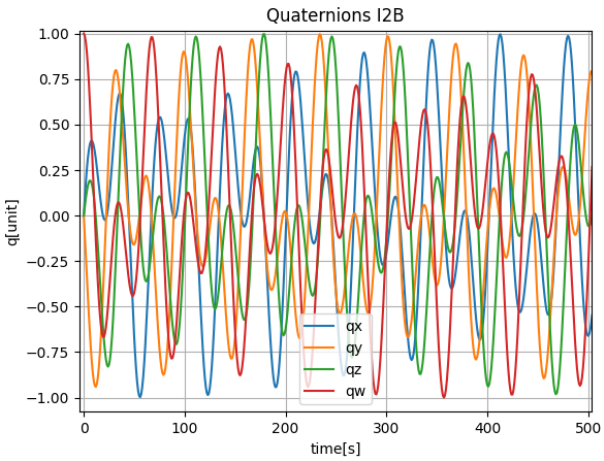


Figura 4.2: Órbita polar simulada usando TLE del cubesat SUCHAI I

Luego, se extraen los datos generados por *Spacecraft Simulator* y se visualizan variables de interés. En la Figura 4.3 (a) se muestra la velocidad angular (simulada) del satélite. Se verifica que sus componentes en los ejes x e y son sinusoidales desfasadas, aproximadamente en 90^o, lo que corresponde con lo esperado de las condiciones de simulación dadas y, en particular, que corresponde con las Ecuaciones 4.12. Por otro lado, su componente en el eje z es aproximadamente constante en torno al valor 0.1 $\frac{rad}{s}$. Si bien este último valor no es absolutamente constante, puesto que tiene asociada una pequeña amplitud, esta se explica dada los errores que se inyectan deliberadamente en el simulador con el fin de crear entornos más realistas. La Figura 4.3 (b) muestra el cuaternión del satélite en función del tiempo, el cual determina su orientación.



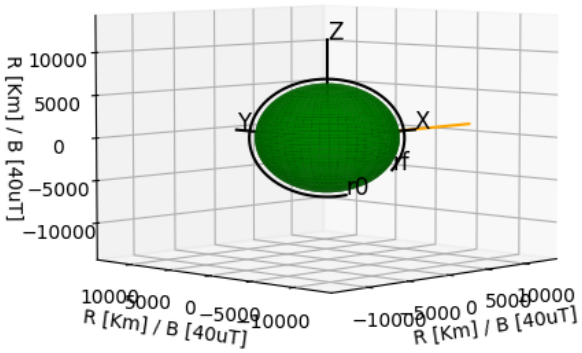
(a) Velocidad angular simulada del satélite en los 3 ejes en función del tiempo



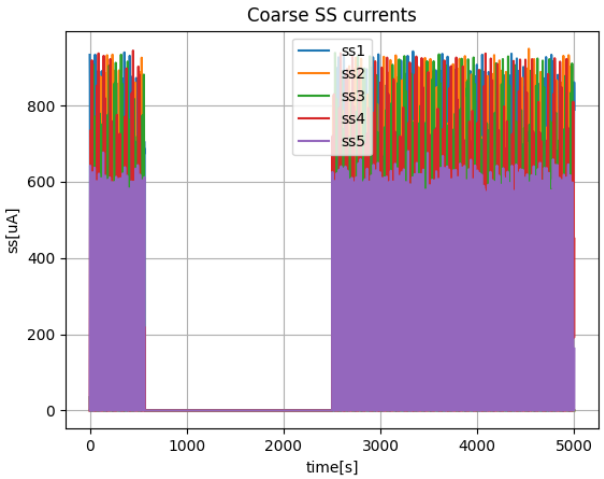
(b) Orientación del satélite representado por cuaternión en función del tiempo

Figura 4.3: Resultados de simulación. Variables de interés.

La Figura 4.4 (a) muestra la trayectoria de la órbita del satélite y la dirección con la que incide el sol (recta amarilla). Se observa que existe una parte de la órbita en la cual el satélite se encuentra detrás de la Tierra, punto donde la sombra de la Tierra se proyecta sobre el satélite y, por lo tanto, este se encuentra completamente oscuro. La Figura 4.4 (b) muestra las corrientes generadas por los fotodiodos del sensor de sol grueso (CSS). En ella, se observa una ventana de tiempo en donde las corrientes del sensor se encuentran en cero. Esto coincide con el periodo de tiempo en que el satélite se encuentra detrás de la Tierra. Esto implica que los sensores de sol, tanto el CSS como el FSS, durante este periodo de tiempo, son inutilizables y ,por lo tanto, no es posible ejecutar el paso de corrección del ESKF con estos sensores.



(a) Trayectoria de la órbita del satélite simulada. En verde se muestra la tierra. La curva negra es la trayectoria del satélite. La recta amarilla representa la dirección con la incide la luz del sol, que coincide con el eje X.

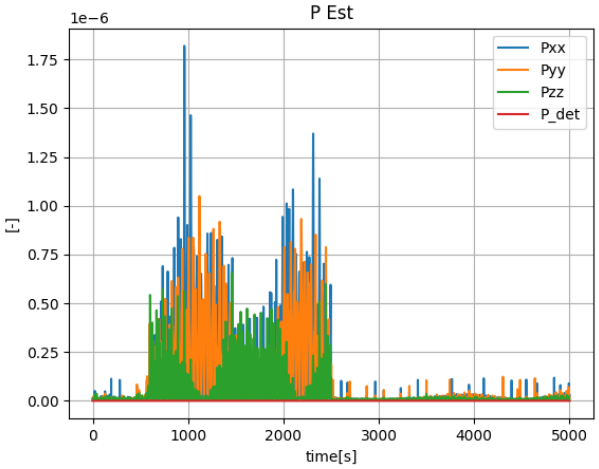


(b) Corriente de los 5 sensores de sol grueso (CSS) en función del tiempo. La ventana de tiempo de (650, 2500) s muestra el periodo de tiempo en que los CSS mantuvieron sus corrientes en cero.

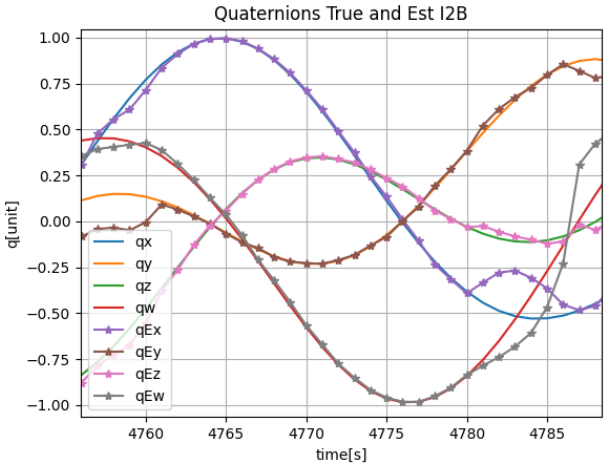
Figura 4.4: Resultados de simulación. Visualización de la incidencia del sol

La Figura 4.5 (a) muestra las componentes de la diagonal de la matriz de covarianzas del ESKF y su determinante. Esta es una forma de cuantificar el error de estimación. Se observa que durante la misma ventana de tiempo, en la cual el satélite no estaba expuesto al sol y en donde no se puede ejecutar el paso de corrección del ESKF con los sensores de sol, los valores de la componente de la matriz aumentan considerablemente. Esto implica, que durante esa ventana de tiempo el error de estimación aumenta.

La Figura 4.5 (b) muestra una comparación entre el cuaternión real (simulado) y el cuaternión estimado por el ESKF durante una ventana de tiempo que el satélite sí se encuentra expuesto al sol. Las 4 curvas suaves (qx, qy, qz, qw) son las variables de referencia. Las 4 curvas con asteriscos (qEx, qEy, qEz, qEw) representan el cuaternión estimado. Se observa que las curvas correspondientes son similares y, por lo tanto, la simulación está funcionando.



(a) Componentes de la diagonal de la matriz de covarianzas del ESKF (P_{xx}, P_{yy}, P_{zz}) y su determinante (P_{det}) en función del tiempo.



(b) Comparación entre el cuaternión real simulado (qx, qy, qz, qw) y el cuarternión estimado (qEx, qEy, qEz, qEw).

Figura 4.5: Resultado de simulación. Visualización del los errores de estimación

4.2. Solución propuesta

4.2.1. Diseño del sistema

Se requiere utilizar y modificar el ESKF que se encuentra en el *Spacecraft Simulator* para poder ejecutarlo en tiempo real en un *Single Board Computer*, en particular en la Raspberry. Para esto, es necesario crear un *script* de python que corra el ESKF, pero utilizando lectura de sensores reales y no simulados.

La Figura 4.6 se muestra el diseño del sistema que se implementa. En ella se observa un bloque de "determinación de Orientación" que será el sistema de principal desarrollo en esta memoria. Este se compone de un procesador central que puede ser un SBC o Microcontrolador. También posee 2 tipos de sensores, una IMU (usando Giroscopio y Magnetómetro) y tres sensores de sol fino (FSS). Los sensores se conectan al procesador (SBC), este último procesa sus datos y ejecuta el algoritmo de estimación, el cual va calculando el cuaternión en tiempo real. Luego, los datos de la salida del algoritmo de estimación se envían al bloque del sistema de control para ser usados en conjunto con los actuadores.

El bloque de determinación de orientación funciona como un *payloads* que deberá ser agregado al resto de los sistemas del satélite, por lo tanto, es deseable que exista una interfaz de comunicación entre el OBC y la estimación de orientación, esta se observa particularmente entre el SBC y el OBC. A su vez, ya existe una comunicación entre el sistema de control y el OBC. Por lo tanto, el bloque de determinación de orientación y el sistema de control, también podrían comunicarse indirectamente entre sí a través del OBC.

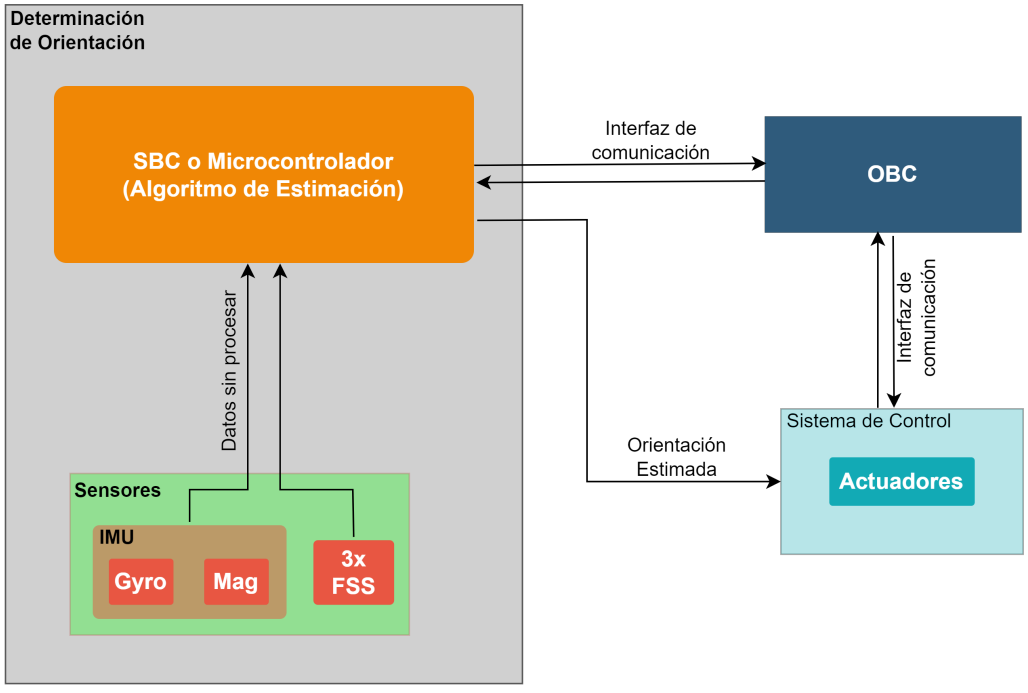


Figura 4.6: Diagrama de bloques del sistema de determinación de orientación en tiempo real. Se compone de 3 bloques principales: OBC, sistema de control y determinación de orientación

Es relevante mencionar que la forma más precisa de ejecutar el ESKF es utilizando 6 FSS, uno en cada cara del *cubesat*. De esta forma, sin importar la orientación en la que se encuentre el satélite, siempre habrá uno de los FSS que esté apuntando hacia el sol que podrá ejecutar el cálculo de la dirección del sol, exceptuando cuando sea de noche y la sombra de la tierra se proyecte sobre el satélite. En el caso del satélite SUCHAI II, actualmente en operación, se instalaron FSS en 5 de las 6 caras. En tal configuración de igual forma se tiene una buena aproximación para el cálculo del vector dirección de sol, dado que en el peor de los casos, la cara sin FSS queda apuntando totalmente hacia el sol, pero como esa cara es conocida de antemano, se puede aproximar un cuarternión para esa orientación. Además, este caso se puede diferenciar del caso en que es de noche, evaluando la posición del satélite dada por el TLE. Para el caso de la solución propuesta se utilizan 3 FSS, que son la cantidad de sensores actualmente disponibles en el Laboratorio SPEL.

En paralelo al bloque de "Determinación de Orientación" se ejecutará un Procesador de movimiento digital (DMP por sus siglas en ingles *Digital Motion Processor*) que puede estimar la orientación de un objeto con buena precisión para aplicaciones dentro de la Tierra. Esto se utilizará para comparar con la estimación lograda con el ESKF y poder calcular cuantitativamente un error de estimación. El procesador de movimiento digital es totalmente independiente del ESKF y por lo tanto no pertenece al diseño final de la solución propuesta, este se implementa con el único objetivo de evaluar el error de estimación. El DMP también utiliza una IMU MPU9250 para calcular la orientación. En secciones siguientes se explica más en detalle su funcionamiento y cómo se utiliza para evaluar el algoritmo de estimación implementado.

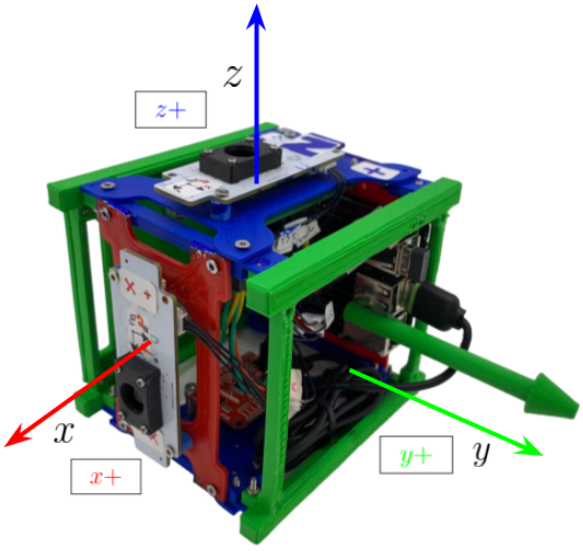
4.2.2. Integración mecánica

Se construye un satélite de prueba tipo *cubesat* de 1U con el objetivo de realizar pruebas experimentales con el algoritmo implementado. En esta se integran tanto partes mecánicas como partes eléctricas. La estructura principal está impresa en 3D en material PLA en colores rojo, verde y azul. Cada color es especialmente utilizado para representar el sistema de referencia móvil del satélite, siguiendo la siguiente convención:

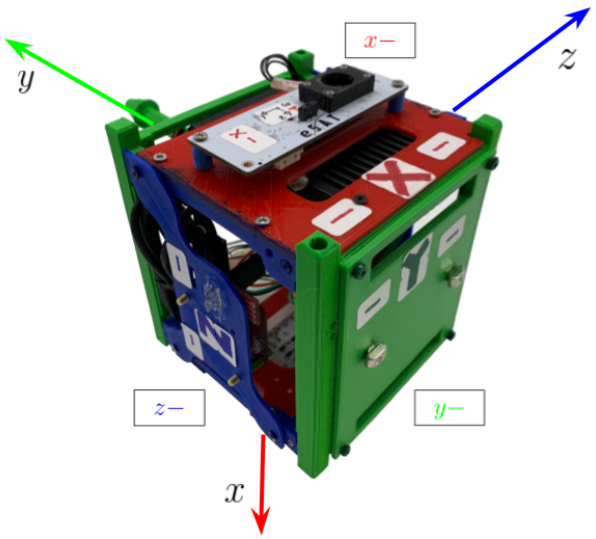
- Eje X: color rojo
- Eje Y: color verde
- Eje Z: color Azul

Además, se representa el sentido de los ejes con una respectiva etiqueta indicando un signo positivo (+) o signo negativo (-) sobre las caras. En particular, el signo del eje Y+, se presenta con una flecha impresa en 3D de color verde. Esta flecha es especialmente útil para cualificar de forma visual el error de estimación al comparar la orientación del satélite real con la orientación estimada.

Pasando por varias iteraciones en el modelo mecánico, se diseñan piezas especiales para integrar todo el hardware en el satélite, tanto dentro de él como en sus caras exteriores e interiores. Dentro de la estructura se encuentran: la *Raspberry*, el módulo de batería, la IMU, el procesador de movimiento digital (DMP) y la placa de conexiones (pcb). Por otro lado, en las caras exteriores se posicionan 3 sensores de sol fino. En la Figura 4.7 se observa el satélite de prueba desarrollado. En particular, se muestran las 6 caras del satélite identificando las caras que representan el sentido positivo y negativo de cada eje; estas son: $x+$, $x-$, $y+$, $y-$, $z+$ y $z-$.



(a) Se muestran las caras con los ejes positivos del sistema de referencias móvil. Caras $(x+)$, $(y+)$ y $(z+)$



(b) Se muestran las caras con los ejes negativos del sistema de referencias móvil. Caras $(x-)$, $(y-)$ y $(z-)$

Figura 4.7: Integración mecánica del cubesat de prueba. Se montan todos los sensores utilizados, las PCB respectivas y baterías. Los colores de las caras del cubesat representan los ejes del sistema de referencia móvil. Además, sobre la estructura se dibujan los ejes del sistema de referencia móvil.

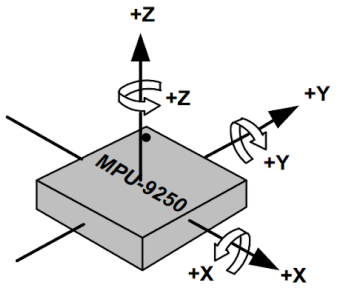
4.2.3. Integración de sensores

4.2.3.1. Unidad de medición inercial

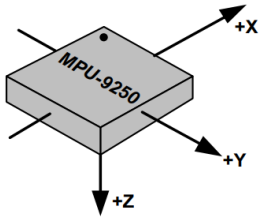
Se integra la lectura de la IMU en la ejecución del ESKF en tiempo real. Para esto se utiliza el bus de comunicación I2C a través de la librería de python *mpu9250-jmdev*[3] para procesar la lectura del sensor con la *Raspberry*. Se configura la comunicación de tal forma que la *Respberry* actúe como maestro y la IMU como esclavo. De esta forma, ejecuta la lectura del giroscopio y el magnetómetro, y se inyectan en el algoritmo ESKF. La lectura del acelerómetro no se utiliza, por razones antes mencionadas. Además, se genera un cambio de unidades correspondiente para calzar con las unidades utilizadas en el algoritmo de estimación. En particular se ejecuta la siguiente conversión:

- **Velocidad angular:** $\frac{\textbf{grados}}{s} \rightarrow \frac{rad}{s}$
- **Campo magnético:** $\mu T \rightarrow nT$

El chip MPU9250 define sus propios sistemas de referencia a partir de los cuales toma la medición de sus 3 sensores (acelerómetro, giroscopio y magnetómetro). Estos se muestran en la Figura 4.8 que se obtienen desde la hoja de datos del componente en [10]. Es muy relevante notar las diferencias entre los sistema de referencia del acelerómetro y giroscopio con el sistema de referencia del magnetómetro. En donde el eje z cambia de signo, además, los ejes x e y están intercambiados. Dado esto, además de la conversión de unidades también se debe realizar una conversión de los ejes de referencia.



(a) Sistema de referencia $(\hat{x}_{acc}, \hat{y}_{acc}, \hat{z}_{acc})$ para el acelerómetro y para rotaciones en el giroscopio



(b) Sistema de referencia $(\hat{y}_{mag}, \hat{x}_{mag}, -\hat{z}_{mag})$ para magnetómetro

Figura 4.8: Orientación de ejes y polaridad de la IMU [10]

Se decide tomar el sistema de referencia del acelerómetro y giroscopio como sistema de referencia principal y bajo el cual se realizan todos los siguientes cálculos. Por lo tanto, se debe transformar el sistema de referencia del magnetómetro como se muestra en la Ecuación 4.14 para tener una medición de campo magnético consistente con el resto de sensores.

$$(\hat{x}_{acc}, \hat{y}_{acc}, \hat{z}_{acc}) \equiv (\hat{y}_{mag}, \hat{x}_{mag}, -\hat{z}_{mag}) \tag{4.14}$$

[3] Documentación oficial de la librería en https://pypi.org/project/mpu9250-jmdev/

Luego, el sistema de referencia móvil del satélite es descrito a partir del sistema de referencia del acelerómetro (ver Figura 4.8 (a)). Por lo tanto la posición y la orientación en la que se instale el MPU9250 dentro del satélite, definirá los ejes del sistema de referencia móvil. Dado que, tanto la IMU como el DMP utilizan el chip MPU9250, se debe tener especial cuidado de que ambos queden en la misma orientación respecto al cubo para que la orientación y la medición de los sensores de ambos pueda ser comparada consistentemente. La Figura 4.9 muestra la posición en la que se instala la IMU y el DMP dentro del satélite de prueba, los cuales se posicionan en la cara interior ($z-$). Además, se muestra el sistema de referencia móvil (x_B, y_B, z_B) que corresponde con el sistema de referencia del MPU9250.

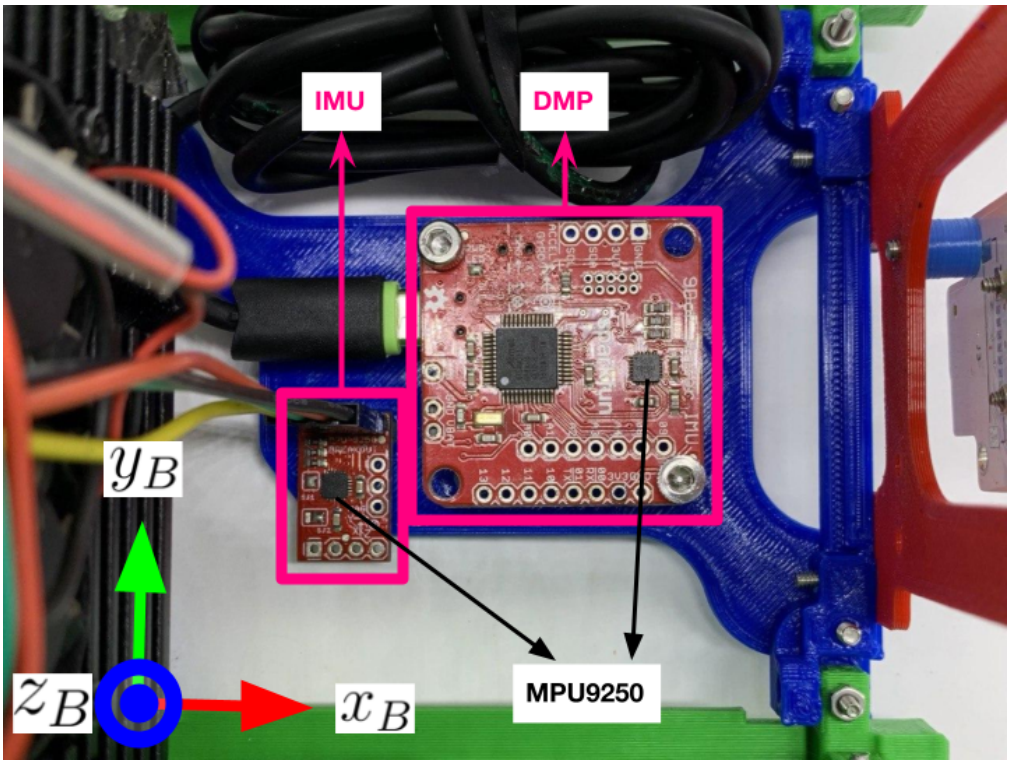


Figura 4.9: Posición de IMU y DMP dentro del satélite de prueba. Ambos chip MPU9250 fueron colocados en la misma orientación sobre la cara interior $z-$ del satélite.

La comunicación I2C permite utilizar direcciones de hasta 7 *bits*, es decir, 128 direcciones en total. Estas suelen representarse en números hexadecimales desde el $0x00$ hasta el $0x7F$. En principio, la lectura de un sensor solo necesita usar una de estas direcciones para poder ejecutar su lectura. Sin embargo, algunos sensores requieren de una configuración previa que puede utilizar muchas más direcciones. Parte de esta configuración, puede ser, la definición de la tasa de comunicación, la frecuencia de muestreo, parámetros de calibración, modos de operación, entre otros. En el Caso de la MPU9250, utiliza 100 direcciones para su configuración, la cuales se observan en la Tabla 4.1. Esta tabla muestra las 128 direcciones I2C existentes. Cada dirección se representa por la posición relativa que tiene en la tabla respecto a la primera fila y columna. El número de la la primera columna representa el dígito más significativo y el número de la primera fila representa el dígito menos significado del número hexadecimal que define la dirección.

Tabla 4.1: Direcciones I2C usadas en la configuración de la IMU MPU9250. La "x" indica una dirección usada, y "-" indica una dirección no usada. Elaborada a partir del *mapa de registro y descripciones* de la hoja de datos de la MPU9250 [13]

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
00	x	x	x	-	-	-	-	-	-	-	-	-	-	x	x	x
10	-	-	-	x	x	x	x	x	x	x	x	x	x	x	x	x
20	-	-	-	x	x	x	x	x	x	x	x	x	x	x	x	x
30	x	x	x	x	x	x	x	x	x	-	x	x	x	x	x	x
40	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
50	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
60	x	-	-	x	x	x	x	x	x	x	x	x	x	-	-	-
70	-	-	x	x	x	x	-	x	x	x	x	-	-	x	x	-

La configuración de la MPU9250 debe ejecutarse solo una vez antes de iniciar las lecturas de los sensores y solo durante esta configuración, la direcciones mostradas en la Tabla 4.1 están ocupadas. Durante la lectura de los sensores (Giroscopio y Magnetómetro) en modo esclavo solo se utilizan la dirección $0x68$ para el giroscopio y la dirección $0x0C$ para el magnetómetro. Es importante no utilizar ninguna dirección que ya esté en uso, pues podría provocar que la comunicación con este y el resto de los sensores fracase.

4.2.3.1.1. Procesamiento de señal digital en tiempo real

Los datos de las lecturas reales del giroscopio, acelerómetro y magnetómetro, presentan pequeñas variaciones o desviaciones respecto a su valor medio, incluso si estos se encuentran complemente quietos. A este fenómeno se le llama *bias* o sesgo del sensor. Se produce debido a las propiedades físicas ligadas de la propia construcción y funcionamiento de estos sensores. También es común que el sesgo cambie según el lugar y tiempo en que se use el sensor. Para compensar el *bias*, se pueden realizar distintos tipos o métodos de calibración que lo estiman para luego considerarlo en la lectura del sensor. Estos métodos, algunos más eficaces que otros en función de los usos que se dan a los sensores, logran disminuir el *bias* pero no lo eliminan por completo. En particular, se utiliza un método de calibración documentado en la librería *mpu9250-jmdev*.

Otra método que apunta a disminuir el *bias*, es utilizar un procesamiento de señales digitales, en particular, utilizar un filtro pasa bajos digital. Existe bastante documentación acerca de filtros de señales utilizando *python*, por ejemplo, la librería *scipy*[4]. Esta librería contiene varías herramientas y funciones que permiten realizar distintos tipos de filtro a señales digitales. Sin embargo, estos están diseñados para realizarlos en un post-procesamiento, luego de medir una variable en una determinada ventana de tiempo. Dado que se requiere ejecutar el ESKF en tiempo real, es necesario que también el filtro digital se calcule en tiempo real. Para aplicaciones de filtros digitales en tiempo real no existe mucha documentación, ni tampoco librerías que lo realicen directamente, por lo tanto, se hizo necesaria la implementación de funciones que lo hagan.

[4] Documentación oficial de la librería en: https://docs.scipy.org/doc/scipy/reference/signal.html

Para esta implementación se utiliza la definición a través de la forma canónica de filtros IIR utilizando la ecuación de diferencias finitas basada en [14] y [15]. Esta se muestra en la Ecuación 4.15. En donde $x[n]$ e $y[n]$ son las entradas y salidas del filtro, respectivamente. Por otro lado, a_i y b_i representan los polos y ceros de la función de transferencia del filtro diseñado, respectivamente, teniendo M polos y N ceros.

$$y[n] = \sum_{i=0}^{M} b_i \cdot x[n-i] - \sum_{j=1}^{N} a_i \cdot y[n-i] \tag{4.15}$$

La misma ecuación se puede escribir de forma vectorial con el objetivo de facilitar su resolución computacional. Se definen los vectores $\vec{a}$ y $\vec{b}$ que contienen los coeficientes del filtro. Estos vectores son de dimensión (1, M) y (1, N), respectivamente. Así mismo, se definen los vectores $\vec{x}$ e $\vec{y}$ que contiene una ventana de datos de las entradas y salidas previas del filtro. Ambos también tiene una dimensión (1, M) y (1, N), respectivamente. Además, se tiene el valor y_n que representa un valor filtrado para el dato n. Entonces, la ecuación de diferencias finitas del filtro es reescrita vectorialmente como una diferencia de dos productos puntos como se muestra en la Ecuación 4.16.

$$y_n = \vec{b} \cdot \vec{x} - \vec{a} \cdot \vec{y} \tag{4.16}$$

Los coeficientes $\vec{a}$ y $\vec{b}$ se calculan usando funciones de *scipy*. En particular, para esto se utiliza el filtro *butter*[5]. Luego, para una determinada lectura de un sensor, se aplica la Ecuación 4.16, con lo que se obtiene el dato filtrado. Cada ventana de datos $\vec{x}$ e $\vec{y}$ se va llenando conforme pasan las iteraciones y los datos recibidos de las lecturas de los sensores. De tal forma que para calcular el primer elemento y_1, el vector $\vec{x}$ es de la forma $(x_1, 0_2, ..., 0_m)$, contiene en su primer valor la primera lectura de un sensor seguido de ceros según el tamaño del vector $\vec{b}$. Para las siguientes iteraciones se van llenando los vectores $\vec{x}$ e $\vec{y}$ por la izquierda. Cada vez que se agrega un valor a la izquierda, debe eliminar el último valor de la derecha, de esta forma siempre se mantiene el mismo tamaño para tales vectores. El tamaño de la ventana de datos depende únicamente del orden del filtro, es decir, de la cantidad de polos y ceros de la función de transferencia del filtro.

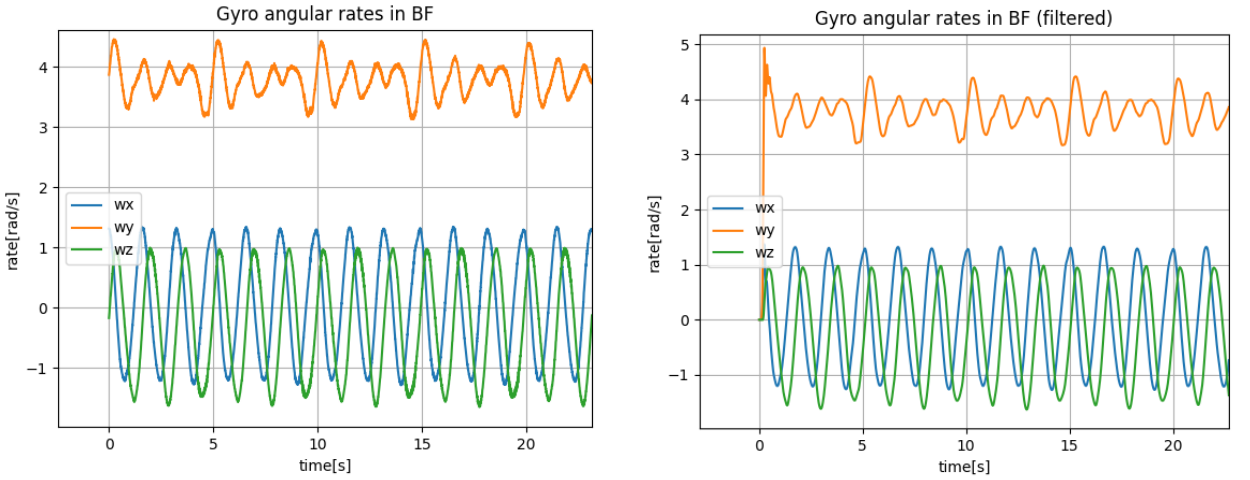
Implementada la función, se diseñan e implementan dos tipos de filtros *butter* pasa bajos en tiempo real para suavizar las señales medidas del giroscopio y del magnetómetro. A continuación, se detallan los parámetros de estos filtros.

- **Para giroscopio:**
  - Orden: 10
  - Frecuencia critica (normalizada): 0.1
- **Para magnetómetro:**
  - Orden: 10
  - Frecuencia critica (normalizada): 0.03

[5] Documentado en: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.butter.html

Es relevante mencionar que dado que cada sensor tiene una lectura independiente por cada uno de los 3 ejes del sistema de referencia, es decir, se tienen 3 señales digitales para cada sensor, entonces, se deben diseñar 3 filtros digitales para cada sensor. Por lo tanto, en total se están filtrando 6 señales en tiempo real, 3 provenientes del giroscopio y 3 provenientes del magnetómetro.

En la Figura 4.10 (a) se muestra una medición directa del giroscopio en una ventana de 20s. Luego, en la Figura 4.10 (b) se muestra la misma señal luego de ser filtrada en tiempo real. Se observa que la señal wy (Velocidad angular en el eje y) comienza cercano a $0\frac{rad}{s}$ a diferencia de la señal no filtrada. Esto se debe justamente a la magnitud del orden del filtro diseñado. Como se tiene un filtro de orden 10, las ventanas de datos $\vec{x}$ e $\vec{y}$ tienen una dimensión de (1, 10) y, por lo tanto, se deben tomar al menos 10 datos, para que la Ecuación 4.16 tenga una buena aproximación al filtro. Antes de alcanzar a tomar esos 10 datos, el cálculo del filtro no es exacto y, por lo tanto, la señal filtrada es más lejana a la señal no filtrada. A pesar de esto, luego de pasar tal ventana de tiempo, se observa que el filtro logra suavizar correctamente las señales obtenidas directamente del giroscopio, por lo que se logra compensar en parte el *bias* del sensor.



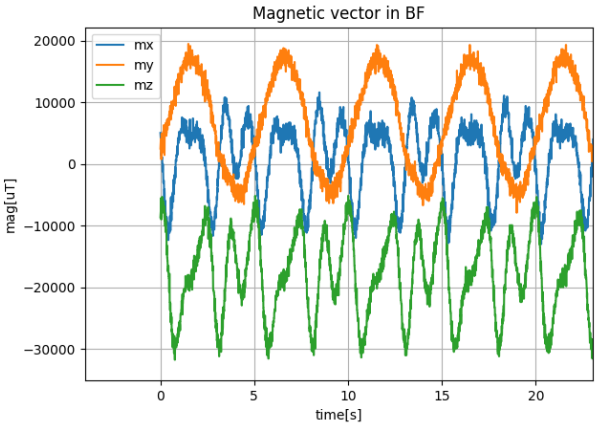
(a) Señales obtenidas directamente del giroscopio

(b) Señales filtradas luego de aplicar el filtro pasa bajos en tiempo real

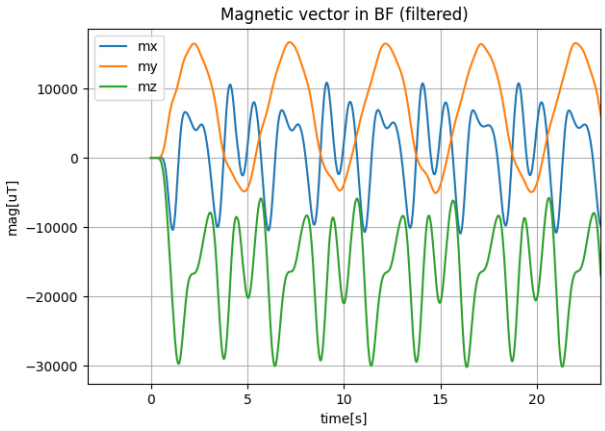
Figura 4.10: Velocidades angulares en 3 ejes medidas con el giroscopio. Se compara la señal medida directamente con el sensor y las mismas señales luego de aplicar el filtro pasa bajos en tiempo real.

La Figura 4.11 muestra la comparación entre las señales obtenidas directamente del magnetómetro (Figura 4.11 (a)) y su correspondiente señal filtra en tiempo real (Figura 4.11 (b)). Se observa que el filtro logra mantener la forma de la señal original, además, la suaviza logrando compensar el *bias* del sensor.

Los filtros mencionados para el giroscopio y el magnetómetro son los que se utilizan para ejecutar el ESKF en tiempo real. Cada interación del ESKF, toma una lectura de los sensores, luego se filtran pasa bajos y, finalmente, se inyectan como entrada en el filtro de Kalman



(a) Señales obtenidas directamente del magnetómetro

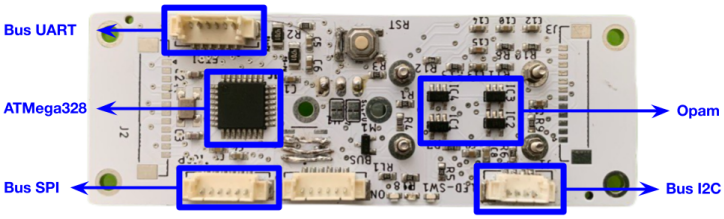


(b) Señales filtradas luego de aplicar el filtro pasa bajos en tiempo real

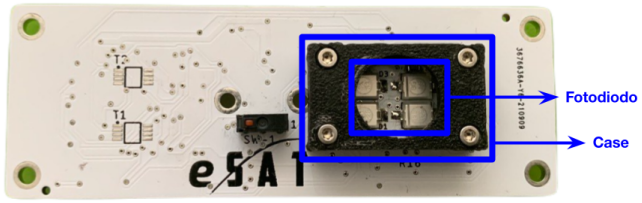
Figura 4.11: Campo magnético en 3 ejes medidos con el magnetómetro. Se compara la señal medida directamente con el sensor y las mismas señales luego de aplicar el filtro pasa bajos en tiempo real.

4.2.4. Sensor de Sol Fino

Se utiliza un diseño de sensor de sol fino desarrollado por el Ingeniero de SPEL Elias Obreque, el cual se muestra en la Figura 4.12, en donde se pueden observar sus principales componentes. El funcionamiento de este sensor se basa en la utilización de 4 fotodiodos equiespaciados entre sí (mostrados en la Figura 4.12 (b)) y una carcasa que provoca una sombra sobre los fotodiodos para mejorar la exactitud de la medición. Cada fotodidio emite una corriente en el intervalo $[0, 90]$ µA y su corriente máxima la emite cuando es excitado por luces en longitudes de onda en los $[600, 900]$ nm, es decir, luces dentro del espectro infrarrojo (datos extraídos de la hoja de datos del componente en [16]). Es necesario amplificar estas corrientes y convertirlas en una señal de voltaje para que puedan ser interpretadas por el conversor analógico digital del microcontrolador ATMega328p utilizado. Para esto, se usa un amplificador operacional por cada fotodiodo.



(a) PCB por delante. Buses de comunicación: UART, SPI, I2C. Microcontrolador ATMega328p. Amplificadores operacionales (Opam)



(b) PCB por detrás. se muestran 4 fotodiodos equiespaciados y una carcasa (*case*) impresa en 3D en material PLA para proyectar una sombra sobre los fotodiodos

Figura 4.12: Sensor de sol fino (FSS). Diseño tipo A. Detalle de componentes. El diseño tipo B se puede observar en el Anexo A

En la Figura 4.12 (a) se muestran los buses de comunicación que tiene disponible el sensor. Estos se especifican a continuación detallando sus usos.

- **Bus SPI:** se utiliza para cargar el *bootloader* del micontrolador ATMega328p. Esto se hace clonando el *bootleader* de la tarjeta de desarrollo *Ardunino Nano* que utiliza el mismo microcontrolador.
- **Bus UART:** se utiliza para programar el microcontrolador, el cual se hace a través de una interfaz FTDI/USB, que permite la comunicación serial entre el microcontrolador y un computador. El sensor es programado en el *IDLE* de Arduino (lenguaje basado en C++).
- **Bus I2C:** se utiliza para la transmisión bidireccional de datos entre el sensor y la *Raspberry*.

El diseño mostrado en la Figura 4.12 es una de las primera iteraciones del desarrollo del sensor. Dado esto, antes de poder utilizarlo directamente, fue necesario entender la lógica de su funcionamiento y también hacer algunas modificaciones tanto en hardware como en software. Existen 2 diseños distintos de FSS disponibles, los cuales son llamados tipo A y tipo B. Hay existencias de 1 FSS tipo A y 2 FSS tipo B. A las 3 placas se les terminó de soldar algunos componentes y se les corrigió un error en la conexión del botón de reinicio del microcontrolador. Además, a los 3 sensores se les cargó el bootloader correspondiente y luego se les programó en lenguaje C++.

La Figura 4.13 muestra el circuito utilizado para la lectura de las corrientes del fotodiodo. Este circuito se basa en un conversor corriente tensión. De esta forma, se convierte la salida de corriente del fotodiodo en una salida del voltaje que pueda ser interpretado por el conversor analógico-digital del microcontrolador.

Se hace una modificación en el circuito corriente tensión. Se agrega un pequeño voltaje v_a en la entrada no inversora del OPAM, lo cual se hace agregando un divisor de voltaje creado a partir de la fuente V_{cc}. El objetivo de agregar el va es evitar la saturación de la OPAM cuando el fotodiodo no emita corriente, lo cual impide que la salida del circuito V_{out} sea cero. Esto es útil para la detección de errores y problemas cuando el sensor esté en uso. En particular, esto permitirá diferenciar el estado de cuando el fotodiodo no está recibiendo luz (sin emisión de corriente), de cuando existe otro tipo de problema, por ejemplo, de comunicación.

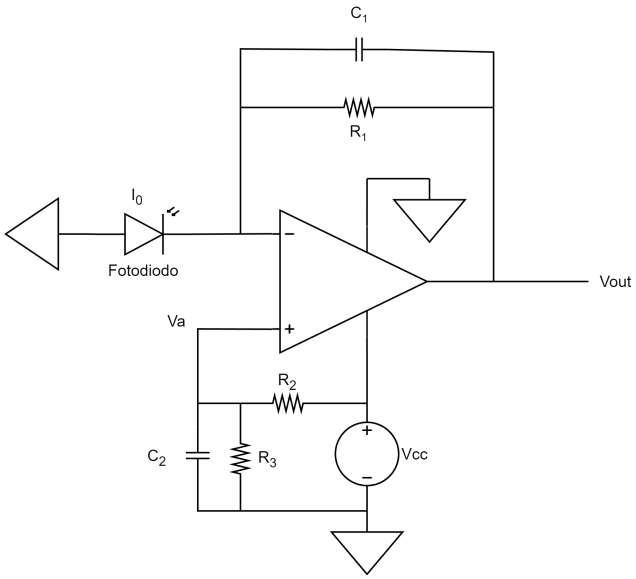


Figura 4.13: Circuito conversor corriente tensión para fotodiodos del FSS.
La corriente I_0 es emitida por el fotodiodo. Se utiliza un $V_{cc} = 5V$

La Ecuación 4.17 muestra la salida de voltaje del circuito en función de la corriente I_0 del fotodiodo, la resistencia R_1 que es responsable de la amplificación y el voltaje V_a. Por otro lado, la Ecuación 4.18 muestra el cálculo del voltaje v_a, el cual es el resultado del divisor de voltaje formado por las resistencias R_2, R_3 y la fuente de voltaje $V_{cc} = 5[V]$.

$$V_{out} = I_0 \cdot R_1 + V_a \tag{4.17}$$

$$V_a = V_{cc} \cdot \frac{R_3}{R_3 + R_2} \tag{4.18}$$

De esta forma, el circuito tiene las siguientes características.

- **Entrada:** $[0, 90]$ µA
- **Salida:** $[V_a, 5]$ V

En cuanto a software, se hicieron modificaciones en la interfaz de comunicación I2C del sensor. Esta interfaz es programada por el lado del sensor (Lenguaje C++ en el microcontrolador ATMega328) y por el lado de la Raspberry (lenguaje Python). Ambos deben ser capaces de recibir y enviar datos. El objetivo de esto es que la Raspberry envíe un comando a un FSS es particular y este responda con una medición nueva de un vector de sol.

Uno de los primeros pasos es definir las direcciones I2C que se usarán en la comunicación. En particular, cada FSS necesita solo una dirección para poder enviar y recibir datos. Para elegirlas se tiene especial cuidado de no utilizar una dirección ya usada por la IMU. Para esto se construye la Tabla 4.2 que muestra las direcciones I2C usadas en todo el sistema, tanto las usadas por la IMU (giroscopio y magnetómetro) y sus configuraciones, como las usadas por todos los FSS. Se definen direcciones para 6 FSS, considerando que en futuras iteraciones del sistema se puedan agregar más fácilmente los sensores restantes. Las direcciones realmente usadas, son las simbolizadas por: fss1, fss2 y fss3 en la tabla.

Tabla 4.2: Direcciones i2c utilizadas en todo el sistema. Se muestra las direcciones usadas por lo sensores de sol (FSS), para la lectura del giroscopio (gyro) y para la lectura del magnetómetro (mag). Las "x" muestran las direcciones usadas en la configuración de la MPU9250. Los "-" muestran las direcciones libres.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	x	x	x	fss1	fss2	fss3	fss4	fss5	fss6	-	-	-	mag	x	x	x
10	-	-	-	x	x	x	x	x	x	x	x	x	x	x	x	x
20	-	-	-	x	x	x	x	x	x	x	x	x	x	x	x	x
30	x	x	x	x	x	x	x	x	x	-	x	x	x	x	x	x
40	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
50	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
60	x	-	-	x	x	x	x	x	gyro	x	x	x	x	-	-	-
70	-	-	x	x	x	x	-	x	x	x	x	-	-	x	x	-

Siguiendo con la interfaz de comunicación, por el lado del microcontrolador, se tienen funciones que ejecutan secuencias cuando se reciben y cuando se solicitan datos. Al encender el sensor, el microcontrolador queda esperando que se reciba algún comando. Mientras este no llegue, el sensor solo se queda esperando. Cuando se recibe un comando, el sensor comienza a ejecutar la tarea respectiva que significa ese comando. En particular, se tiene un solo comando programado, que es el número "39", sin embargo, dada la lógica ya implementada, se pueden fácilmente agregar nuevos comandos en futuras iteraciones. Este comando, hace que el microcontrolador comience la lectura de los fotodiodos y calcule el vector de dirección del sol. Además, se crea una estructura que empaqueta los datos del vector de sol, la cual se se muestra en la Tabla 4.3. Cada paquete tiene un tamaño de 20 [bytes]. Mientras no se reciba otro comando, el sensor calcula el vector de sol indefinidamente dentro de un *While True*. Cuando se solicitan datos, se activa una interrupción por hardware en el microcontrolador, lo que interrumpe el *While True* y el sensor envía el último vector de sol calculado enviando el paquete de datos correspondiente. Luego de esto, se vuelve dentro del *While True* y se continua calculando el vector de sol, indefinidamente.

Tabla 4.3: Paquete de datos enviado por el FSS. Contiene el vector de dirección de sol normalizado y los ángulos ϕ y θ que definen tal dirección. También se muestra el rango que puede tomar cada variable. El tamaño total del paquete es de 20 bytes.

Paquete		**Rango**	**Tipo**	**Tamaño en bytes**
Vector de sol normalizado	Rx	[-1, 1]	float	4
	Ry	[-1, 1]	float	4
	Rz	[-1, 1]	float	4
Vector de sol Ángulos	ϕ	$[-\pi, \pi]$	float	4
	θ	$[0, \frac{\pi}{2}]$	float	4
Total				**20**

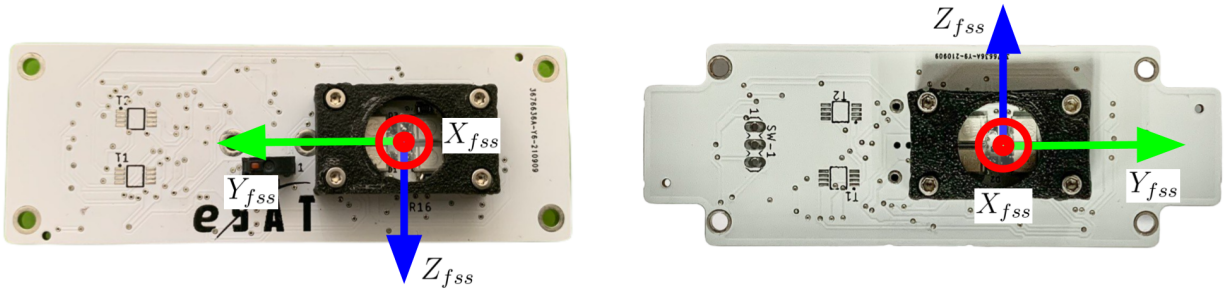
Por el lado de la Raspberry. Se programan dos principales funciones:

- Función *init*: comienza la comunicación con uno de los FSS enviando un comando a una determinada dirección I2C (dependiendo del FSS que se quiera leer). Al enviar este comando, comienza la comunicación entre FSS y Raspberry. En particular, el comando "39", hace que el sensor comience a calcular el vector de sol.
- Función *read*: solicita al FSS una cantidad determinada en bytes para recibir. En particular, esta función solicita la lectura de 20 bytes, que es el tamaño justo del paquete de datos enviado por el FSS. Al recibir este paquete, se decodifica según el orden definido en la Tabla 4.3 y retorna el vector de sol.

Dada la interfaz de comunicación desarrollada, se logra una frecuencia de muestreo de $380\frac{paquetes}{s}$. Es decir, se logra una medición nueva de un vector de sol cada $2.63ms$.

4.2.4.1. Sistema de referencia del sensor

Cada FSS tiene su propio sistema de referencia respecto al cual se mide el sensor de sol. Este depende del orden en que se posicionan y se lean cada uno de los fotodiodos sobre la PCB del sensor. Dadas algunas diferencias entre los diseños de los FSS tipo A y B, ambos sensores tienen un sistema de referencia distinto. Estos se especifican en la Figura 4.14. En particular, el eje X_{fss} está saliendo perpendicularmente del plano de la PCB del sensor. Luego, los ejes y_{fss} y z_{fss} se encuentra sobre el plano de la PCB según lo especificado en la Figura 4.14.



(a) Sistema de referencia del diseño tipo A

(b) Sistema de referencia del diseño tipo B

Figura 4.14: Definición de sistema de referencia en cada diseño de FSS disponible

4.2.4.2. Pruebas de medición

Dado que esta es una de las primeras iteraciones en el desarrollo de un FSS, es necesario comprobar la precisión que logra el sensor en el cálculo del vector de dirección del sol. Sin embargo, ejecutar pruebas que permitan calcular la precisión exacta del sensor puede requerir de mucho tiempo de desarrollo dado que se requiere de un método externo que permita calcular precisamente la dirección real del sol. Ya que este no es uno de los objetivo principales de la memoria, solo se realizan pruebas simples que permitan dar a conocer cualitativamente la precisión del sensor.

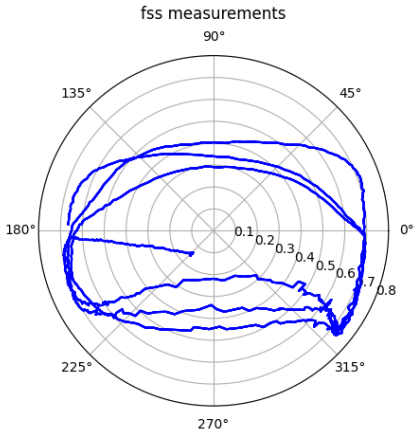
Se diseñan dos pruebas. La primera, utilizando una lámpara halógena que logra emitir luz en el rango medible de los fotodiodos. La segunda, apuntando el sensor directamente a la luz del sol. El montaje de medición se puede observar en la Figura 4.15. El experimento consiste en apuntar un FSS directamente a la fuente de luz y luego realizar un movimiento en forma circular en torno a ella tal como se indica en la Figura 4.15. Si las mediciones son correctas, las componentes en los ejes y y z del vector de sol deberían formar un circulo graficadas en un sistemas de coordenadas polares, o análogamente, al graficar ϕ en función del tiempo, se debería observar una curva que vaya desde -180^o a 180^o o viceversa.

(a) Montaje de medición utilizando una lámpara halógena.

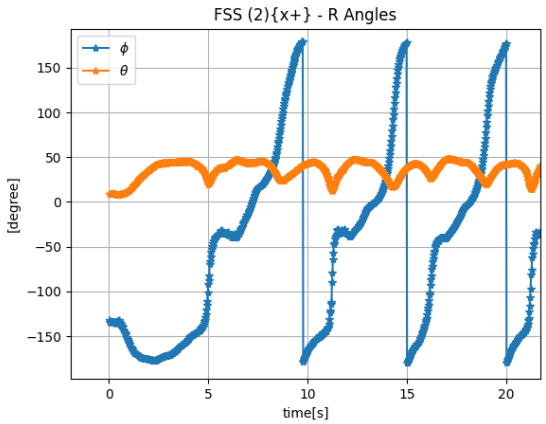
(b) Montaje de medición midiendo directamente la luz del sol.

Figura 4.15: Montaje de medición para pruebas del FSS midiendo la luz generada por una lámpara halógena y el sol. En ambos experimentos se apunta un FSS directamente hacia el sol y se simula manualmente un movimiento circular.

La Figura 4.16 muestra los resultados de la prueba utilizando la lámpara halógena. La Figura 4.16 (a) logra mostrar un curva ovalada, similar a un círculo, lo que es concordante con lo esperado. La forma achatada del círculo puede deberse a una mala ejecución del movimiento circular durante el experimento, o a la forma del espectro de luz generado por la lámpara, debido a que este es más grande de forma vertical que de forma horizontal. Por otro lado, en la Figura 4.16 (b) se observa que el ángulo ϕ en el intervalo [2, 10] s, forma una curva creciente que va desde -180^o a 180^o, lo cual también es concordante con lo esperado, pues sin importar la inexactitud del movimiento circular ejecutado, demuestra que dio un giro completo. De la misma figura, también fue posible extraer el campo visión logrado por el sensor, el cual se estima como el valor máximo medido en el ángulo θ que es de aproximadamente 48^o. Esto es 12^o menos que el FSS de *gomspace*.



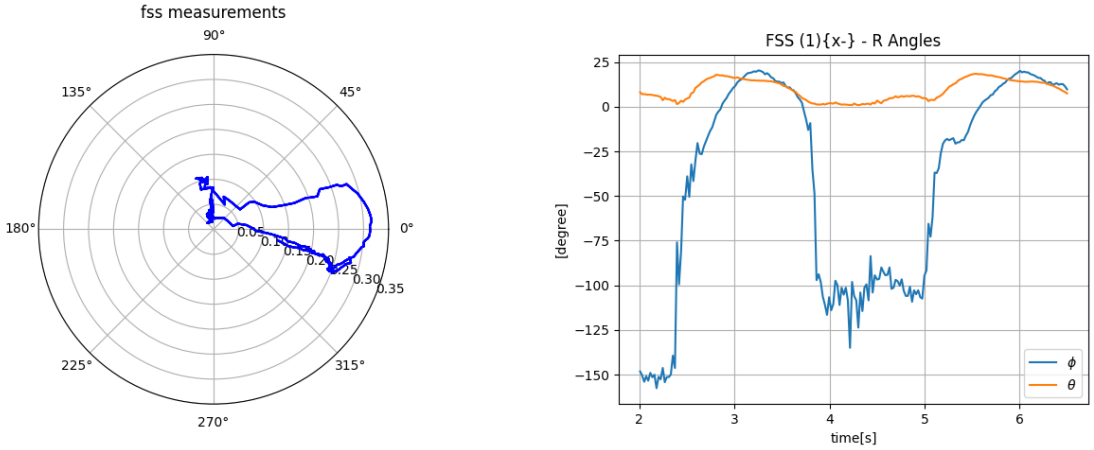
(a) Componentes y y z del vector de sol graficadas en coordenadas polares.



(b) Ángulos θ y ϕ del vector de sol en función del tiempo.

Figura 4.16: Resultados de prueba realizada con lampara halógena

La Figura 4.17 muestra los resultados de la prueba utilizando como fuente de luz el sol. En la Figura 4.17(a) se observa una curva que se encuentra principalmente entre los 315^o y 45^o. La forma de esta curva se explica debido al tamaño del sol, en comparación con la lámpara, el área de iluminación es mucho mayor, por lo tanto, el movimiento circular realizado con el brazo no fue suficiente para abarcar los 360^o en ϕ. Al observar la Figura 4.17 (b), se confirma este resultado, pues se muestra que el ángulo ϕ solo se mueve de -150^o a 25^o.



(a) Componentes y y z del vector de sol graficadas en coordenadas polares.

(b) Ángulos θ y ϕ del vector de sol en función del tiempo.

Figura 4.17: Resultados de prueba realizada con la luz del sol

Estos resultados muestran que el FSS logra calcular vectores de dirección del sol con un campo de visión de hasta 48^o en θ. Vectores dentro de este rango son medibles, sin embargo, no se calcula la precisión del sensor.

4.2.4.3. Transformación de ejes de referencia

Debido a que cada FSS tiene su propio sistema de referencia y a que son posicionados en distintas caras del *cubesat*, cada sistema de referencia de los FSS debe ser transformado para que concuerde con el sistema de referencia móvil del satélite. De esta forma, cada medida de vector de sol estará calculado respecto al sistema de referencia del satélite y ,por lo tanto, será consistente con el resto de sensores.

Se posicionan FSS en las caras $(x-)$, $(x+)$ y $(z+)$ del *cubesat* de prueba. La Figura 4.18 muestra la posición de los 3 FSS indicando cada uno de sus sistemas de referencias, los cuales se dibujan referidos al sistema de referencia móvil. Respecto a la Figura 4.18, es posible calcular las rotaciones necesarias que se deben aplicar para pasar de un sistema de referencia a otro. Cada rotación es representada por un cuartenión, los cuales son indicados en Figura 4.18 en función de la cara correspondiente como sigue: q_{B2x-}, q_{B2x+} y q_{B2z+}

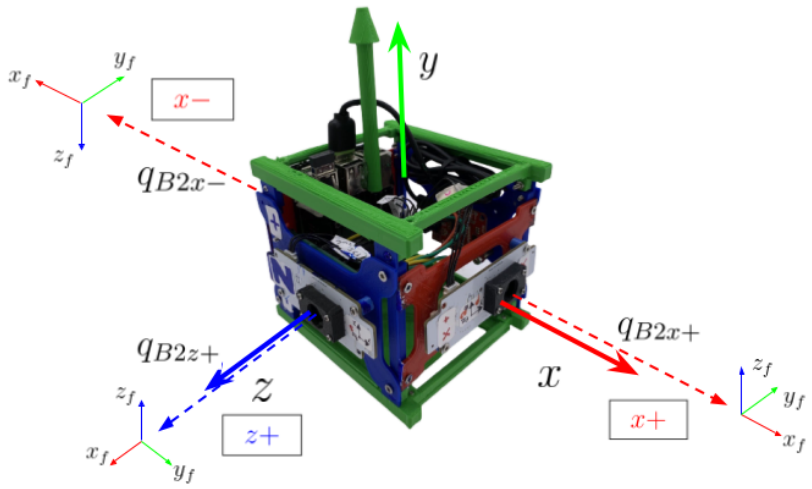


Figura 4.18: Ejes de referencias de cada FSS referenciado al sistema de referencia móvil del satélite

A continuación se detallan y se calculan los cuaterniones necesarios para rotar los sistemas de referencias de cada uno de los 3 FSS y transformarlos al sistema de referencia móvil:

- **FSS cara (x+):** se debe aplicar una rotación en $\frac{\pi}{2}$ respecto al X_{fss}. La Ecuación 4.19 muestra el cuaternión.

$$R\left\{\frac{\pi}{2}X_{fss}\right\} \rightarrow \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}\hat{i} \tag{4.19}$$

- **FSS cara (x-):** se debe aplicar una rotación en $\frac{\pi}{2}$ respecto al X_{fss} y luego, una rotación en π respecto al Z_{fss}. La Ecuación 4.22 muestra el cuaternión final, resultado de la multiplicación de los cuaterniones de ambas rotaciones dichas.

$$R\left\{\frac{\pi}{2}X_{fss}\right\} \rightarrow \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}\hat{i} \tag{4.20}$$

$$R\left\{\pi Z_{fss}\right\} \rightarrow \hat{k} \tag{4.21}$$

$$\left(\frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}\hat{i}\right) \otimes \hat{k} = \frac{\sqrt{2}}{2}\hat{j} + \frac{\sqrt{2}}{2}\hat{k} \tag{4.22}$$

- **FSS cara (z+):** se debe aplicar una rotación en $\frac{\pi}{2}$ respecto al Z_{fss} y luego, una rotación en $\frac{\pi}{2}$ respecto al X_{fss}. La Ecuación 4.25 muestra el cuaternión final, que es resultado de la multiplicación de los dos cuaterniones de ambas rotaciones mencionadas.

$$R\left\{\frac{\pi}{2}Z_{fss}\right\} \rightarrow \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}\hat{k} \tag{4.23}$$

$$R\left\{\frac{\pi}{2}X_{fss}\right\} \rightarrow \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}\hat{i} \tag{4.24}$$

$$\left(\frac{\sqrt{2}}{2}+\frac{\sqrt{2}}{2}\hat{k}\right)\otimes\left(\frac{\sqrt{2}}{2}+\frac{\sqrt{2}}{2}\hat{i}\right)=\frac{1}{2}+\frac{1}{2}\hat{i}+\frac{1}{2}\hat{j}+\frac{1}{2}\hat{k} \tag{4.25}$$

Finalmente, la Tabla 4.4 muestra un resumen de los cuaterniones necesarios para llegar de los sistemas de referencia de cada FSS al sistema de referencia móvil.

Tabla 4.4: Cuaterniones para las rotaciones para pasar del sistema de referencia del móvil al sistema de referencia de cada FSS

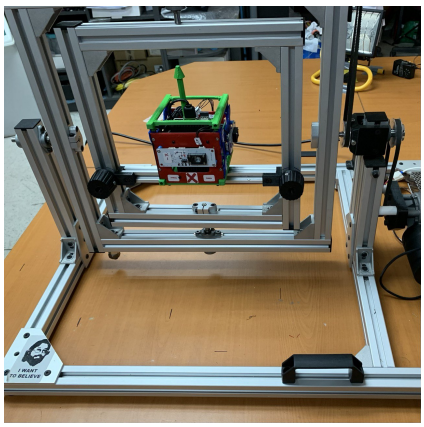
Cuaternión	q_{B2x+}	q_{B2x-}	q_{B2z+}
Rotación	$R\left\{\frac{\pi}{2}X_{fss}\right\}$	$R\left\{\frac{\pi}{2}X_{fss}\right\}\rightarrow R\{\pi Z_{fss}\}$	$R\left\{\frac{\pi}{2}Z_{fss}\right\}\rightarrow R\left\{\frac{\pi}{2}X_{fss}\right\}$
[qw, qx, qy, qz]	$[\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2},0,0]$	$[0,0,\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2}]$	$[\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2}]$

4.3. Plataforma de Prueba

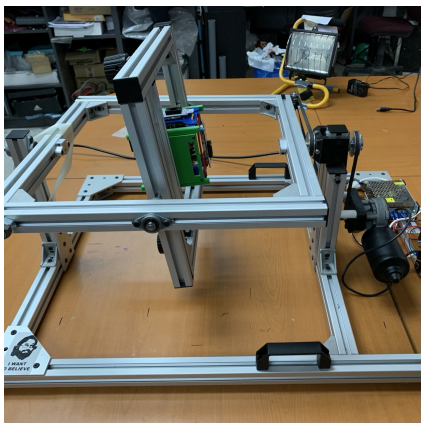
Se requiere de una plataforma de prueba que permita rotar el *cubesat*, generando movimientos continuos que simulen el movimiento del satélite en el espacio. Esto permitirá ejecutar el algoritmo de estimación mientras la máquina este en funcionamiento y de esta forma evaluar su rendimiento. Para esto, se gestiona una colaboración con el Laboratorio de Fabricación Digital[6] de la Universidad (FabLab Uchile), quienes facilitan una de sus máquinas para ser usada en el desarrollo de esta memoria. Esta se muestra en la Figura 4.19.

La máquina fue desarrollada con el objetivo de rotar continuamente los objetos que se encuentran en su interior. Su funcionamiento se basa en la utilización de un motor DC para mover dos ejes de rotación que se encuentran acoplados entre sí a través de correas y engranajes. En el Anexo B se observa un detalle de los componentes de la máquina. El satélite de prueba es acoplado a la estructura de la máquina a través de tuercas tipo T como se especifica en el Anexo B.1.

La finalidad de utilizar la máquina, es poder rotar constantemente el satélite mientras se ejecutan en paralelo el ESKF y el DMP en tiempo real para estimar la orientación. Luego de esto, se pondrán comparar los resultados de ambos y se podrá calcular el error de estimación del ESKF.



(a) Vista frontal. Ejes posicionados paralelamente.



(b) Vista frontal. Ejes rotados 90 grados desde la posición inicial.

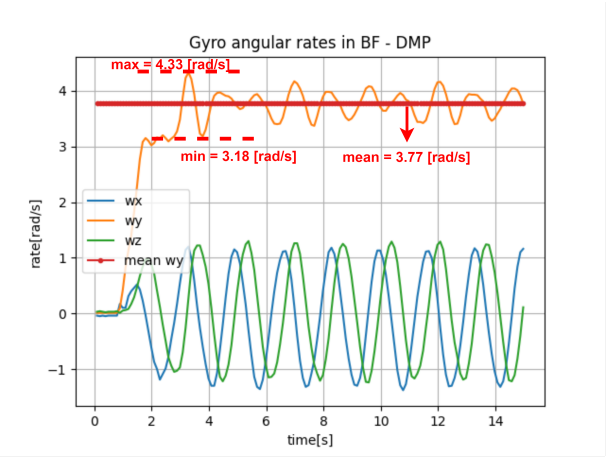
Figura 4.19: Maquina “rotomoldeadora” desarrollada por ingenieros del FabLab. El cubesat de prueba se acopla al centro de la máquina.

Es relevante notar que dado el movimiento de la máquina, no es posible comunicar el hardware del satélite de prueba con el computador de forma alámbrica. Por lo tanto, es totalmente necesario el desarrollo de una comunicación inalámbrica.

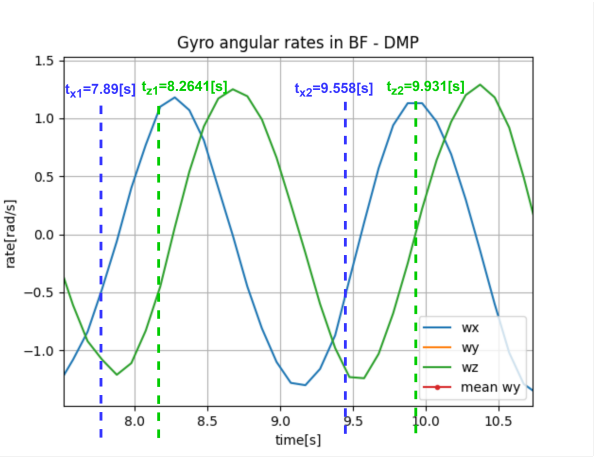
Se realizan pruebas preliminares para estudiar el movimiento generado por la máquina. Con el satélite dentro se toman datos de la velocidad angular que se generan. Los resultados se muestran en la Figura 4.20 que muestra la velocidad angular en $[\frac{rad}{s}]$ en los 3 ejes. Se observa que dos de sus ejes, x y z, presentan una curva de forma seudo sinusoidal que están acopladas. Estas se mueven en un rango aproximado de entre $[-1, 1]\frac{rad}{s}$. Por otro lado, el eje y muestra

[6] Sitio oficial del Laboratorio: http://www.fablab.uchile.cl/

una velocidad angular levemente superior en magnitud, pero que tiene una amplitud notablemente menor. La figura 4.20 (a) muestra el promedio, valor máximo y mínimo de velocidad de angular en el eje y pasados los 2 primeros segundos de prueba. Estos datos muestran que la velocidad angular en este eje es en promedio $3.77\frac{rad}{s}$ y que su valor varía entre en $[3.18, 4.33]\frac{rad}{s}$.



(a) Velocidad angular en los 3 ejes. Se muestra el valor máximo, mínimo y promedio de la componente en el eje Y.



(b) Velocidad angular en los ejes X y Z. Se muestra el tiempo de inicio y final de un periodo para cada eje.

Figura 4.20: Velocidad angular del movimiento generado por la máquina

La Figura 4.20 (b) muestra un zoom a los ejes x y z. De esta forma, se estudia el periodo de ambas curvas, mostrando los tiempos en que esta curvas pasan por el valor $0\frac{rad}{s}$ en un ciclo completo. Las Ecuaciones 4.26 y 4.27 muestran en valor del periodo de las curvas en el eje x y z, respectivamente. Luego, tomando el tiempo del inicio de un periodo entre ambas curvas, se estima el ángulo de desfase entre ambas curvas usando la Ecuación 4.28, el cual es de 80.794^o.

En síntesis, el movimiento provocado por la máquina es similar al movimiento que sigue un satélite en órbita en condiciones de un ambiente libre de torque.

$$T_{x1} = t_{x2} - t_{x1} = 9.558 - 7.89 = 1.668[s] \tag{4.26}$$

$$T_{z1} = t_{z2} - t_{z1} = 9.931 - 8.2641 = 1.6669[s] \tag{4.27}$$

$$\Delta\theta = \left(\frac{t_{x1} - t_{z1}}{T_{z1}} \cdot 360^o\right) = \frac{8.2641 - 7.89}{1.6669} \cdot 360 = 80.794^o \tag{4.28}$$

4.4. Orientación de referencia

Se requiere un método para medir de forma precisa la orientación del satélite en Tierra durante la ejecución de las pruebas, con el objetivo de poder usarla como referencia verdadera para evaluar el ESKF. Para esto, se utiliza un Procesador de movimiento digital (DMP, por sus siglas en inglés, *Digital Motion Processor*). En particular, se utiliza una tarjeta de desarrollo de *sparkfun* que se muestra en la Figura 4.21. Esta se compone principalmente de un microcontrolador SAMD21 y de una IMU MPU9250. El DMP implementa un filtro de Kalman propagando el giroscopio y realizando un paso correctivo con el acelerómetro. Para aplicaciones dentro de la Tierra, este es un método con buenos resultados, ya que la aceleración de gravedad aporta una buena referencia para el cálculo de la orientación. De esta forma, el DMP consigue calcular un cuaternión con buena precisión.

Existen librerías para la utilización del DMP, sin embargo, solo están disponibles para *C++*. Debido a que el resto del sistema se ejecuta en la Raspberry en lenguaje *python*, es necesario crear una interfaz de comunicación que envíe los datos del cuaternión, calculado en el microcontrolador, hacia la raspberry, para que de esta forma puedan ser leídos en *python*.

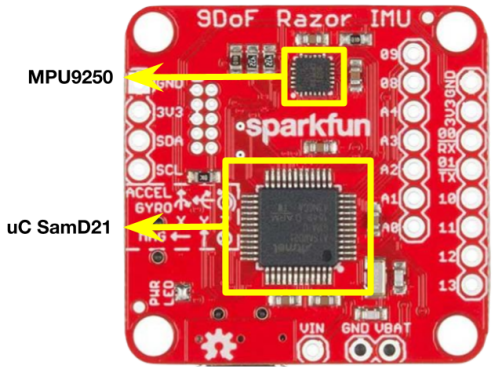


Figura 4.21: Tarjeta de desarrollo de *sparkfun 9Dof Razor IMU*. Se compone de una IMU MPU9250 y un microcontrolador Samd21

Una de las opciones para la comunicación entre DMP y Raspberry es I2C, sin embargo, ya que el DMP es un componente externo al sistema de estimación propuesto, es conveniente que la electrónica que use sea la menos influyente posible en el sistema. Por esta razón, se decide usar la interfaz UART. De esta forma, el DMP se conecta a través del puerto serial con un cable USB con la Raspberry. Dado que la comunicación UART es asíncrona, es necesario crear un protocolo de comunicación por el lado de la Raspberry que permita recibir correctamente los paquetes de datos.

Se programa el DMP en *C++*, definiendo una frecuencia de actualización del filtro de Kalman de 10hz. Es decir, habrá una actualización del cuaternión cada $100ms$, aproximadamente. Además del cuaternión, también se obtienen los datos del giroscopio, a pesar de que este dato ya se obtiene a partir de la IMU usada en el ESKF, también se obtiene desde el DMP con el objetivo de hacer comparaciones. De esta forma, se define un paquete de datos para enviar a la Raspberry que se muestra en la Tabla 4.5.

Tabla 4.5: Paquete de datos dmp

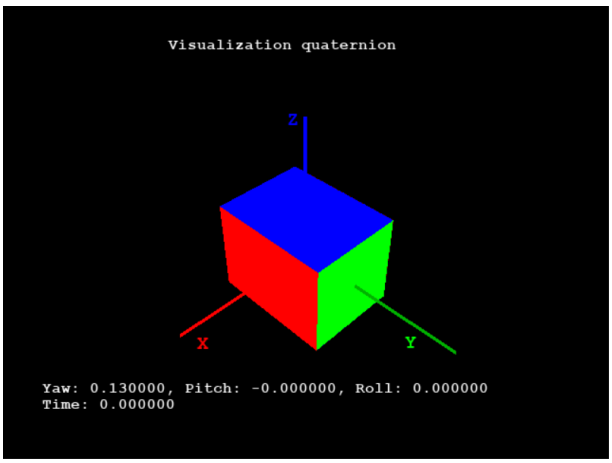
Paquete		**tipo**	**tamaño en bytes**
Cuaternión	qw	float	4
	qx	float	4
	qy	float	4
	qz	float	4
Velocidad angular Medido en rad	gyro_x	float	4
	gyro_y	float	4
	gyro_z		4
Variable de verificación	Checksum	uint8	1
Total			**29**

El cuaternión y los datos del giroscopio son enviados en un tipo *float*. Además, se agrega una "variable de verificación" (llamada checksum), que se utiliza en el protocolo de comunicación para comprobar que el paquete fue recibido correctamente. Esta variable es calculada como la suma en bytes de cada componente del paquete de datos. La suma se hace tanto en el microcontrolador como en la *Raspberry*. De esta forma, si la variable *checksum* en la raspberry es distinta a que se recibió en el paquete de datos, quiere decir que el paquete de datos fue recibido incorrectamente. En este caso, el paquete es descartado ya que no hay forma de saber cual de los datos se recibió con errores.

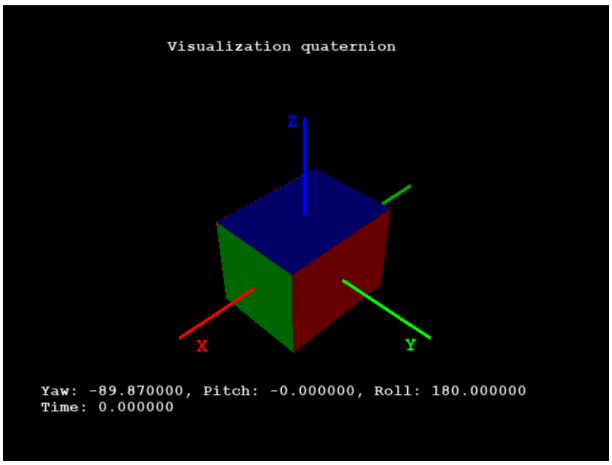
4.5. Herramienta de visualización de orientación

Se desarrolla una aplicación que permite visualizar la orientación estimada en tiempo real. El objetivo de esta, es poder comparar en tiempo real la orientación del satélite de prueba, mientras se encuentra dentro de la plataforma de prueba, con la estimación realizada por el ESKF y con el DMP al mismo tiempo. La aplicación recibe un cuaternión y lo visualiza en forma de cubo, asimilando la forma del satélite de prueba. También, se muestra los ángulos *yaw-pitch-roll* y el tiempo en segundo desde el inicio de la estimación. Ejemplos de la ventana de esta aplicación se muestran en la Figura 4.22.

Dentro de la aplicación de la visualización también se muestran los dos sistema de referencia, inercial y móvil. El primero, se representa con las 3 rectas fijas perpendiculares de colores: rojo para el eje x, verde para el eje y y azul para eje z. El segundo, es representado con los colores del las caras, análogamente a como fue definido en el satélite de prueba real. Las caras $x+$, $y+$ y $z+$ son representadas por los colores más claros e intensos (ver Figura 4.22 (a)). Las caras $x-$, $y-$ y $z-$ son representadas por colores más oscuros (ver Figura 4.22 (b)). Además, se agrega una asimetría al cubo a lo largo del eje y, para que visualmente sea más fácil de reconocer la orientación. Así mismo, también análogo al satélite de prueba, se agrega una flecha de color verde claro en la cara $y+$, con el objetivo de que sea más fácil y más comparable con el *cubesat* real.



(a) Ejes positivos del sistema de referencia móvil



(b) Ejes negativos del sistema de referencia móvil

Figura 4.22: Visualización de las 6 caras del cubo visualizado

La aplicación se desarrolla utilizando las librerías *PyOpenGL* y *PyGame* de python. Es capaz de visualizar hasta 135FPS, es decir, es capaz de recibir 135 cuaterniones por segundo, procesarlos y visualizarlos. Por lo tanto es suficientemente rápida para ser ejecutada en tiempo real junto con el algoritmo de estimación.

5. Diseño de pruebas

5.1. Metodología de ejecución de pruebas

La metodología para ejecutar las pruebas de estimación con todo el sistema, tiene 4 principales elementos que se realizan en paralelo y se detallan como sigue. Además, en el Anexo C se puede observar el montaje experimental.

1. **Plataforma de prueba:** con el satélite de prueba acoplado a la máquina, se comienza a hacer girar. Este movimiento se mantiene durante toda las prueba, dejando el motor DC de la máquina a una velocidad constante.

2. **Orientación de referencia:** utilizando el DMP se calcula la orientación (cuaternión) que se usa como referencia verdadera para evaluar el ESKF. Para esto, se genera un *script* en *python* que lee los paquetes de datos desde el DMP, decodifica el dato del cuaternión y los envía, inalámbricamente, a un computador para que puedan ser visualizados. Este *script* mide y guarda las variables que se observan en el Anexo E. En las siguientes secciones se le llama "cuaternión real" al cuaternión calculado por el DMP.

3. **Orientación estimada:** se ejecuta el ESKF en tiempo real. El cuaternión estimado se envía, inalámbricamente, a un computador para ser visualizados. Este *script* mide y guarda las variables que observan en el Anexo D

4. **Visualización:** el computador recibe la orientación estimada por el ESKF y el DMP y visualiza ambos paralelamente en tiempo real.

La Figura 5.1 muestra el diagrama de funcionamiento del sistema general. En ella se observa un bloque de "cubesat de prueba" que ejecuta el algoritmo de estimación ESKF y también el cálculo de la orientación de referencia con el DMP. Para esto, la *Raspberry* está ejecutando 4 *threads* en paralelo. El funcionamiento de cada *threads* se detalla a continuación.

- ***Thread* 1:** el microcontrolador Samd21 del DMP, estará calculando el cuaternión de referencia indefinidamente dentro de un ciclo sin fin y, a su vez, enviará estos datos por el bus UART. Este *thread* se comunica con el DMP por UART y lee los datos de la orientación de referencia. A su vez, envía estos datos al *Thread* 3.

- ***Thread* 2:** este *thread* estará ejecutando el algoritmo de estimación. Esto es, procesando la lectura de la IMU y de los sensores de sol, además, ejecutando el ESKF. Así mismo, enviará los datos de orientación estimada al *Thread* 4.

- ***Thread* 3:** Está corriendo un *Socket* con protocolo TCP. Recibe los datos del *Thread* 1, los codifica y los envía, inalámbricamente, hacia un computador que actúa como cliente del socket TCP.

- **Thread 4:** similar al *Thread* 3, este recibe los datos de orientación estimada, y los envía, inalámbricamente, hacia un computador que actúa como cliente del *socket*.

Por el lado del computador, se tienen corriendo 2 *threads* más en paralelo. El funcionamiento de estos se detalla a continuación.

- ***Thread* 1:** recibe los datos de orientación de referencia desde el socket del thread 3 de la RPI y los decodifica. Así mismo, visualiza en tiempo real el cuaternión de referencia recibido utilizando la herramienta de visualización.
- ***Thread* 2:** recibe los datos de orientación estimada, recibida desde el *socket* del *thread* 4 de la RPI y los decodifica. De esta forma, visualiza en tiempo real el cuarternión estimado utilizando la herramienta de visualización.
- **Evaluación:** posterior al término de cada experimento, en la RPI se guarda la base de datos de todos los parámetros de estudio del algoritmo en archivos de tipo .csv. Finalizados los experimentos, se transfieren a través de un tunel SSH hacia el computador. De esta forma, se pueden estudiar de forma posterior todos los datos generados.

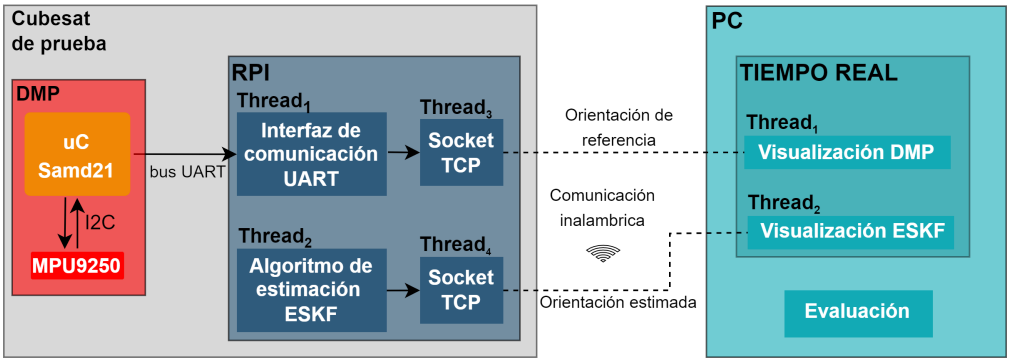


Figura 5.1: Diagrama de bloques del sistema completo. Bloque de cubesat de prueba comunicándose inalámbricamente con un computador que se encarga de la evaluación del algoritmo.

La Figura 5.2 muestra las conexiones entre todos los componentes del sistema. En particular, la IMU se conectará por I2C con la *Raspberry*. Los FSS se conectarán por I2C a la *Raspberry*, pero estos pasarán por un conversor de niveles para disminuir su voltaje lógico de 3.3V a 5V para ser compatible con el voltaje lógico de la *Raspberry*. Por otro lado, el DMP se conectará a través de UART con la *Raspberry*, utilizando uno de los puertos USB de esta. Finalmente, La *Raspberry* se conectará inalámbricamente a través de *sockets* con protocolo TCP con un computador.

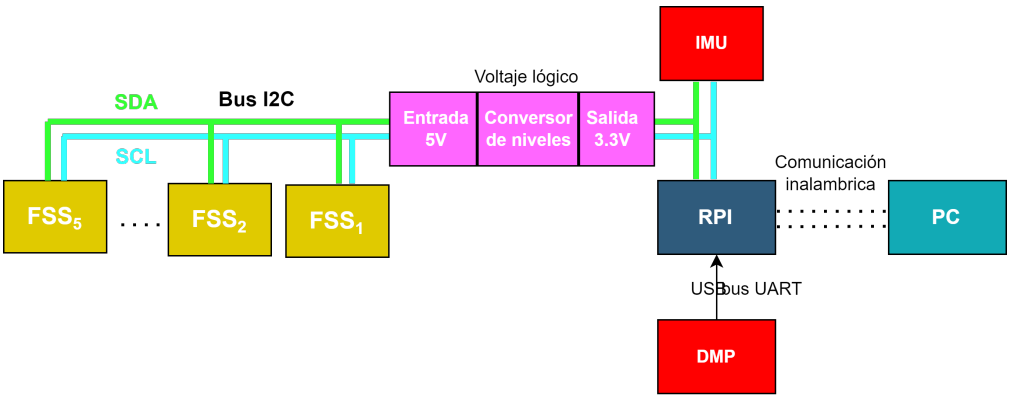


Figura 5.2: Conexiones generales del sistema. El Bus I2C comunica todos los sensores con la *Raspberry*. El Bus UART comunica el DMP con la *Raspberry*. La *Raspberry* se comunica a través de un *socket* con un computador

5.2. Definición de pruebas

A continuación se definen las pruebas que se realizaron con el objetivo final de evaluar el rendimiento del ESKF a través de resultados que permitieron calcular el error de estimación. Para esto, se especifican distintos experimentos.

5.2.1. Pruebas de estabilidad con DMP

El cálculo del cuaternión que realiza el DMP, se toma como orientación de referencia verdadera, por lo tanto, es importante conocer qué tan bueno es. El DMP implementa un filtro de Kalman utilizando giroscopio y acelerómetro, por lo que, es interesante ponerse en distintos escenarios en donde la aceleración de gravedad pueda afectar de distintas formas. En particular, se realizan dos pruebas detalladas a continuación.

1. **Experimento 1:** se deja el satélite en reposo en una posición en donde la gravedad se proyecte sobre los 3 ejes del sistema de referencia móvil del satélite.
2. **Experimento 2:** se deja el satélite en reposo en donde uno de los ejes del sistema de referencia móvil está paralelo a la gravedad.

5.2.2. Pruebas de estimación con ESKF

Los siguientes experimentos se realizan utilizando la estimación de orientación del ESKF y el cálculo de orientación de referencia verdadera del DMP. Es importante conocer el comportamiento del algoritmo con cada uno de los sensores, por lo que las pruebas están separadas según el uso o no de algunos de ellos. Además, algunas de las pruebas son realizas con el satélite en reposo y otras con el satélite en movimiento dentro de la plataforma de prueba tal cual se especifica en la Sección 5.1.

5.2.2.1. Utilizando solo giroscopio:

Los siguientes experimentos se realizan únicamente calculando el cuaternión a través de la propagación de un cuaternión inicial (orientación inicial arbitraria) utilizando las velocidades angulares obtenidas del giroscopio. Es decir, no se ejecuta el paso de corrección ni con el magnetómetro ni con los sensores del sol. Dado esto, es importante que el momento en que se inicie la prueba, la orientación del satélite debe coincidir con el cuaternión inicial dado en el ESKF, de lo contrario, se inducirá un error inicial sobre el cuartenión estimado. Es particularmente complejo lograr esto, dado que una rotación milimétrica del satélite tiene un cambio relevante en el cuaternión. Considerando esto, para que las comparaciones entre los ángulos de error entre una prueba y otra sean consistentes, se debe considerar la amplitud del error.

3. **Experimento 3:** Satélite en reposo sin utilizar filtro pasa bajos sobre la medición del giroscopio.
4. **Experimento 4:** Satélite en reposo aplicando el filtro pasa bajos sobre la medición del giroscopio.
5. **Experimento 5:** Satélite en movimiento dentro de la máquina sin utilizar filtro pasa bajos sobre la medición del giroscopio.
6. **Experimento 6:** Satélite en movimiento dentro de la máquina, aplicando el filtro pasa bajos sobre la medición del giroscopio.

5.2.2.2. Utilizando giroscopio, magnetómetro y FSS

El siguiente experimento se realiza utilizando el ESKF completo, es decir, utilizando el giroscopio para propagar un cuaternión inicial y aplicando el paso de corrección con el magnetómetro y 3 FSS.

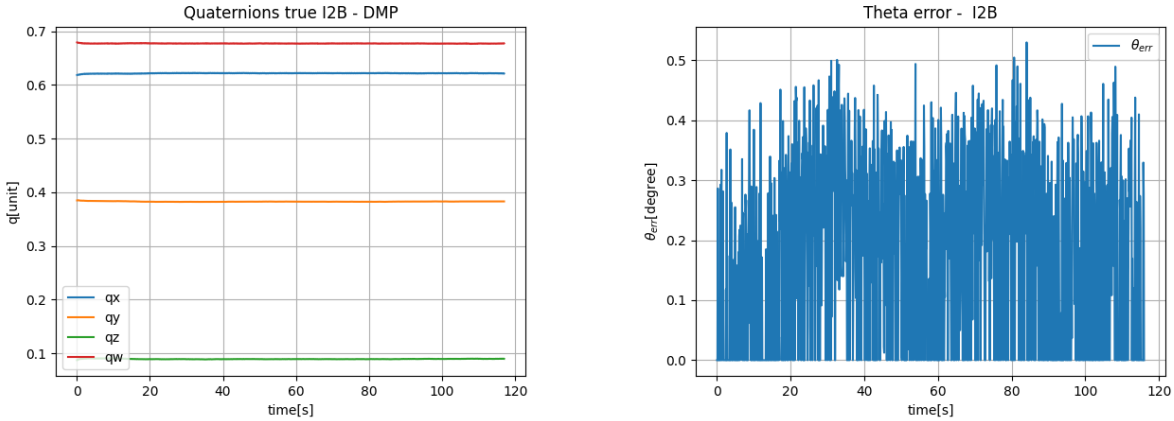
7. **Experimento 7:** Satélite en movimiento dentro de la máquina, aplicando el filtro pasa bajos sobre le medición del giroscopio y el magnetómetro.

6. Resultados y Análisis

6.1. Pruebas de estabilidad con DMP

6.1.1. Experimento 1

La Figura 6.1 muestra los resultados del experimento 1.En (a) se muestra el cuaternión calculado y en (b) se muestra el ángulo de error del cuaternión comparado con el primer cuarternión estimado. Esto muestra que el ángulo de error se mantiene dentro de un rango aceptable menor a 0.5^o durante los $120[s]$ en la que se ejecutó la prueba.



(a) Cuaternión calculado por DMP en función del tiempo.

(b) Ángulo de error θ_{error} entre cuaternión inicial y cada cuaternión calculado por el DMP en el tiempo

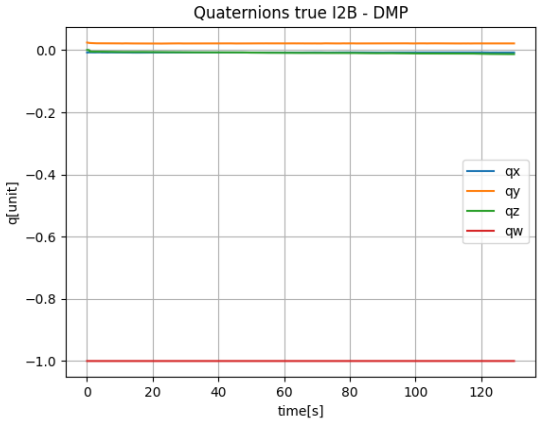
Figura 6.1: Resultados experimento 1. La prueba se ejecuta durante $120[s]$

6.1.2. Experimento 2

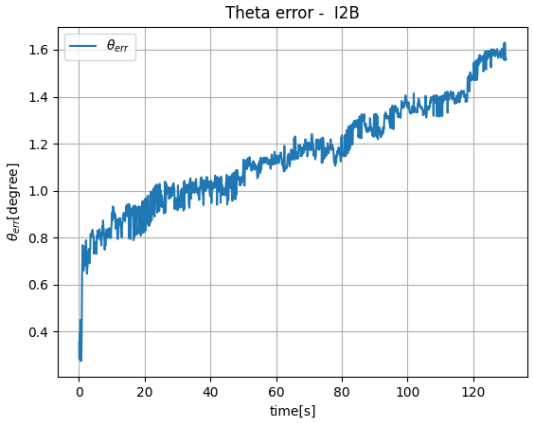
La Figura 6.2 muestra los resultados del experimento 2. En (a) se muestra el cuaternión calculado, en donde se observa que sus 4 componentes parecen ser bastante planas, lo que indica poco o casi nulo movimiento. Sin embargo, en (b) se muestra el ángulo de error del cuaternión, en donde se observa que el error aumenta con el tiempo, es decir, en esta posición el cálculo del cuaternión presenta un error acumulativo. Al final de los $120[s]$ de la prueba, se presenta un error de 1.6^o.

Esto se explica debido a la influencia de la aceleración de gravedad para el cálculo de la orientación. En la segunda prueba, la gravedad solo se proyectaba sobre un eje, por lo tanto,

la corrección del filtro de Kalman que utiliza la aceleración tiene mayoritariamente influencia sobre ese eje y no sobre los otros dos. Esto provoca en el cálculo de la orientación, que el eje paralelo a la gravedad se mantenga quieto, es decir, la cara perpendicular a la gravedad se mantiene quieta, pero sus otros dos ejes comienzan a girar en torno a este. Esto, dado que la corrección de la aceración influye solo mínimamente en ellos.



(a) Cuaternión calculado por DMP en función del tiempo.



(b) Ángulo de error θ_{error} entre cuaternión inicial y cada cuaternión calculado por el DMP en el tiempo

Figura 6.2: Resultados experimento 2. Prueba ejecutada durante 120[s].

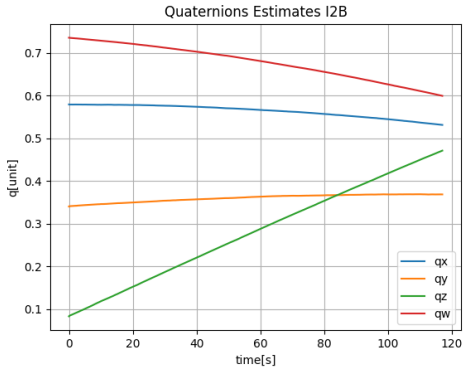
Los resultados del experimento 1 y 2, muestran que el cálculo del cuaternión del DMP es impreciso cuando se cumplen dos condiciones simultáneamente. La primera, es que el satélite se encuentre quieto. La segunda, es que uno de los ejes del sistema de referencia móvil sea paralelo a la gravedad. Al realizar pruebas utilizando la máquina, esta dos condiciones nunca se cumplen simultáneamente, por lo tanto, el DMP tiene una medida confiable de orientación para ejecutar los experimentos siguientes.

6.2. Pruebas de estimación

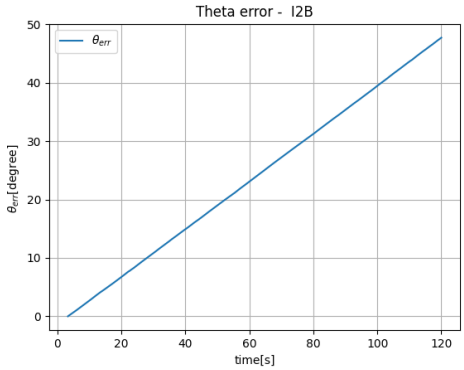
6.2.1. Utilizando solo giroscopio

6.2.1.1. Experimento 3

La Figura 6.3 muestra los resultado de la prueba realizada, manteniendo el cubo en reposo sin aplicar el filtro pasa bajos. En (a) se muestra el cuaternión estimado en función del tiempo. Dado que el cubo no estaba en movimiento, la forma del cuaternión, debió ser una recta horizontal en sus 4 componentes, sin embargo, se observa un claro error acumulativo en el tiempo, en donde las curvas presentan una clara variación. En (b) se muestra el ángulo de error del cuaternión. En ella se observa también claramente el mismo error acumulativo, en donde se distingue que el error del ángulo es una recta creciente. Al finalizar la prueba, luego de los 120[s], se acumula un error de 47.87^o.



(a) Cuaternión estimado por ESKF en función del tiempo.

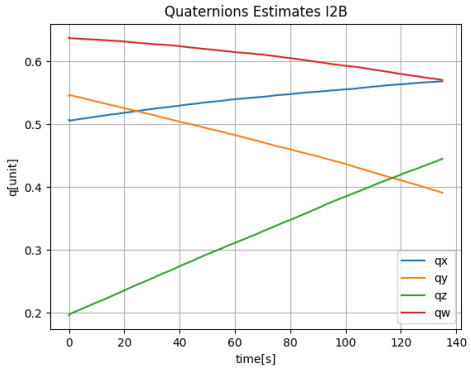


(b) Ángulo de error θ_{error} entre cuaternión calculado por el DMP y el cuaternión estimado por el ESKF en función del tiempo

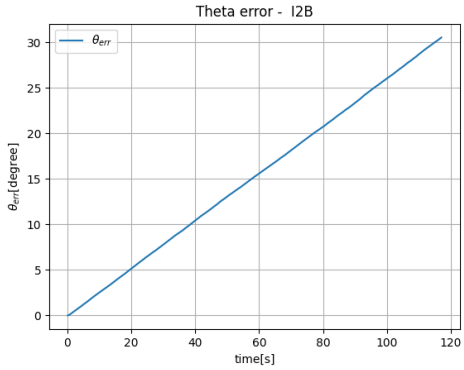
Figura 6.3: Resultados experimento 3. Prueba realizada durante 120$[s]$

6.2.1.2. Experimento 4

La Figura 6.4 muestra la prueba realizada solo dejando el cubo en reposo y aplicando el filtro pasa bajos sobre el giroscopio. En (a) se muestra el gráfico del cuaternión en función del tiempo, en donde nuevamente se observa un claro error acumulativo. En (b) se muestra el gráfico del ángulo de error del cuaternión, en este, se puede confirma la presencia del error acumulativo. Al finalizar la prueba luego de 120$[s]$ se observa un error de 30.51^o. Tal cual era lo esperado, el error acumulativo sigue apareciendo, sin embargo, aplicar el filtro pasa bajos, hizo que el error disminuyera considerablemente. Al comparar el experimento 2 se tiene una disminución del error de 17.3^o, luego de 120$[s]$ de la ejecución del ESKF.



(a) Cuaternión estimado por ESKF en función del tiempo, en una ventana de 0s a 15s.

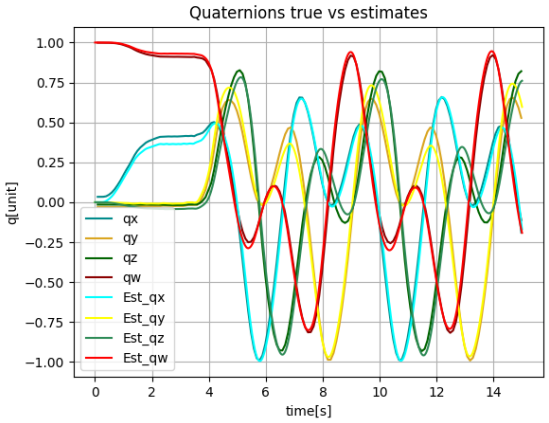


(b) Ángulo de error θ_{error} entre cuaternión calculado por el DMP y el cuaternión estimado por el ESKF en función del tiempo, en una ventana de 0s a 120s.

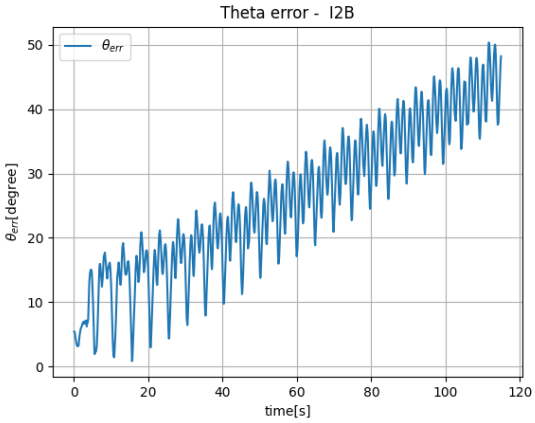
Figura 6.4: Resultados experimento 4

6.2.1.3. Experimento 5

La Figura 6.5 muestra los resultados del experimento 5 con el satélite dentro de la máquina y sin aplicar el filtro pasa bajos. En (a) se muestra la curva del cuaternión real en contraste con la curva del cuaternión estimado en una ventana de tiempo de $[0, 15]s$. Se observa que las 4 componentes del cuaternión estimado, se asemejan mucho a la forma del cuaternión real, observando solo unas pequeñas diferencias. En (b) se muestra el ángulo de error durante los $120[s]$ de la prueba. Nuevamente, se observa que está presente un error acumulativo, al igual que en las pruebas sin movimiento. La amplitud del error en esta prueba es de 49.54^o en la ventana de tiempo de los $120[s]$. A pesar de que en (a), las curvas de los cuaterniones real y estimado parecen muy cercanas, incluso al final de la ventana de tiempo en los $15s$, en (b) se muestra que el error en el mismo instante, está cercano a los 20^o.



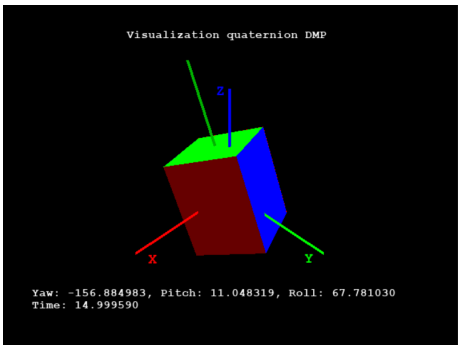
(a) Cuaternión calculado por DMP y estimado por ESKF en función del tiempo, en una ventana de 0s a 15s



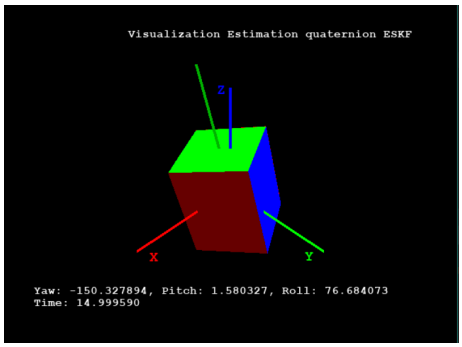
(b) Ángulo de error θ_{error} entre cuaternión calculado por el DMP y el cuaternión estimado por el ESKF en una ventana de 0s a 120s.

Figura 6.5: Resultados experimento 5

La Figura 6.6 muestra la visualización de los cuaterniones real y estimados del experimento 5 en el instante $t = 15s$ de iniciada la prueba. Se observa levemente una diferencia entre ambas visualizaciones, que se traduce en un ángulo de error de 20^o.



(a) Cuaternión real

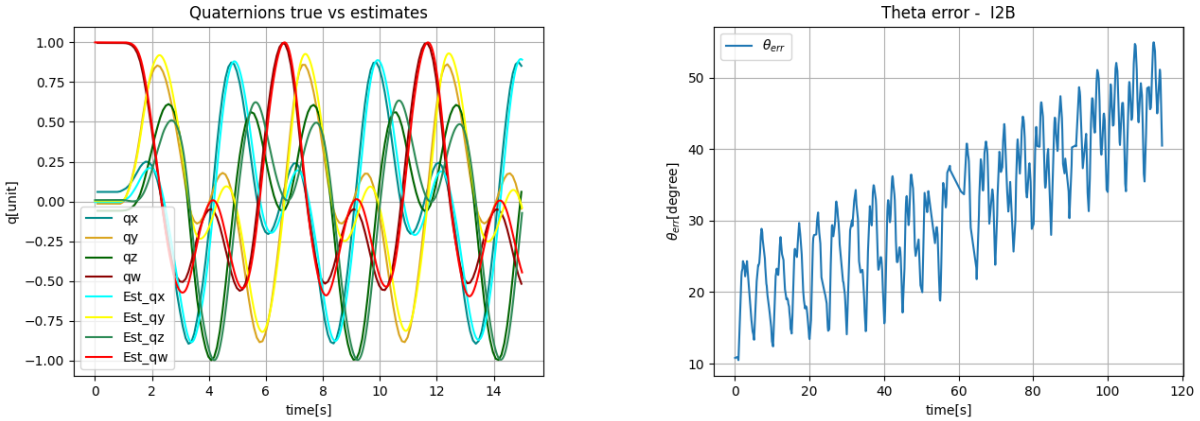


(b) Cuaternión estimado

Figura 6.6: Visualización del cuaternión real y estimado en el instante $t = 15s$ de la prueba.

6.2.1.4. Experimento 6

La Figura 6.7 muestra los resultadodel experimento 6 con el satélite dentro de la máquina y aplicando el filtro pasa bajos. En (a) se muestra las curvas del cuaternión real en contraste con el estimado. Se observa que la la forma de ambas curva es similar, teniendo un resultado similar al del experimento 5. En (b) se muestra el ángulo de error del cuaternión durante los $120s$ de prueba. Nuevamente se observa un error acumulativo que tiene un mínimo de 10.48^o y un máximo de 54.91^o, por lo tanto, la amplitud del error es de 44.43. Es decir, se logra una disminución del error respecto al experimento 5, que no aplicaba el filtro pasa bajos.

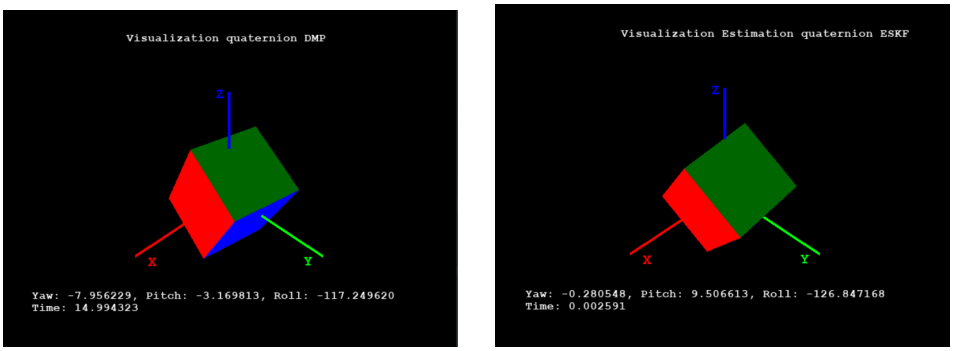


(a) Cuaternión calculado por DMP y estimado por ESKF en función del tiempo, en una ventana de 0s a 15s.

(b) Ángulo de error θ_{error} entre cuaternión calculado por el DMP y el cuaternión estimado por el ESKF en función del tiempo, en una ventana de 0s a 120s.

Figura 6.7: Resultados experimento 6.

La Figura 6.8 muestra la visualización de los cuaterniones real y estimado del experimento 6 en el instante $t = 15s$ de la prueba. Tienen un ángulo de error cercano a los 25^o.



(a) Cuaternión real

(b) Cuaternión estimado

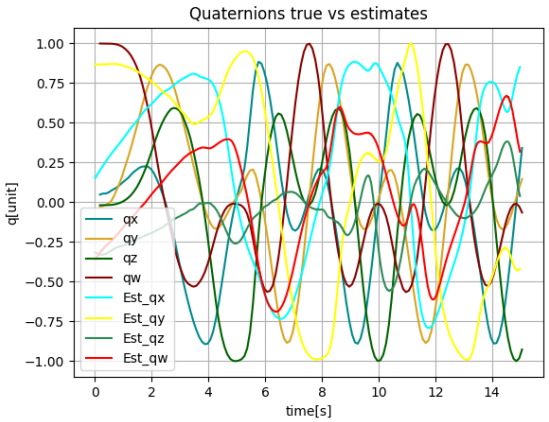
Figura 6.8: Visualización del cuaternión real y estimado en el instante $t = 15s$ de la prueba.

Todos los resultados utilizando solo giroscopio muestran un error acumulativo, el cual se logra disminuir efectivamente aplicando un filtro pasa bajos en tiempo real sobre las señales del sensor. Estos resultados están dentro de lo esperado, pues el error acumulativo se debe específicamente a la acumulación del *bias* sobre la propagación del cuaternión inicial. Se espera que utilizando el paso de corrección del ESKF usando magnetómetro y FSS, se logre corregir este error acumulativo. Se destaca que estas pruebas, lograron demostrar que la aplicación de un filtro pasa bajos en tiempo real sobre las señales del sensor, disminuye considerablemente el error, por lo tanto, en los siguientes experimentos se utiliza el filtro pasa bajos en el giroscopio y en el magnetómetro.

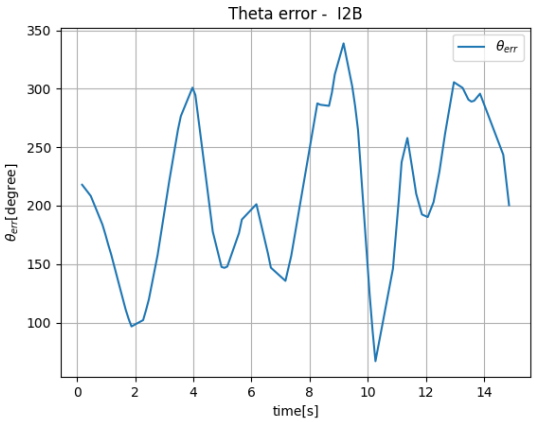
6.2.2. Utilizando giroscopio, magnetómetro y FSS

6.2.2.1. Experimento 7

La Figura 6.9 muestra los resultados del experimento 7 con el satélite dentro de la máquina y utilizando un filtro pasa bajos sobre las señales del magnetómetro y del giroscopio. En (a) se observa la curva del cuaternión real en contraste con el estimado. Ambas curvas son muy distintas, por lo tanto, la estimación en esta prueba es mala. Observando (b) se confirma que ángulo de error es muy alto y, por lo tanto, la estimación de orientación no es confiable.



(a) Cuaternión calculado por DMP y estimado por ESKF en función del tiempo, en una ventana de 0s a 15s



(b) Ángulo de error θ_{error} entre cuaternión calculado por el DMP y el cuaternión estimado por el ESKF en función del tiempo, en una ventana de 0s a 15s.

Figura 6.9: Resultados experimento 7.

7. Conclusiones

La presente memoria presenta el desarrollo de un sistema embebido que implementa un algoritmo de estimación de orientación para nano-satelites. Se cumple el objetivo de desarrollar un satélite de prueba tipo *cubesat* a escala real que el reúne el hardware necesario para el sistema. De esta forma, se logra ejecutar y probar el algoritmo de estimación con medidas de sensores reales. Se logra avanzar un paso más allá de las simulaciones, desarrollando el sistema en un hardware apto para funcionar en el espacio.

Se logra diseñar una metodología que permite la evaluación del algoritmo de estimación de forma cuantitativa y cualitativamente, utilizando la plataforma de prueba facilitada por el FabLab. Esto permitió calcular el error de estimación de forma numérica respecto a un cálculo de orientación más preciso y ,también, de forma visual al comparar la orientación del satélite real con la orientación estimada en la herramienta de visualización desarrollada.

Dadas la metodología de evaluación del algoritmo, se obtiene que la integración del sensor de giroscopio fue bien realizada, pues sus resultados coinciden con lo esperado teóricamente. Así mismo, también se reconocen los malos resultados obtenidos al utilizar el magnetómetro y FSS para la inyección del error observable. En un trabajo futuro debe seguir trabajando en corregir la integración matemática del paso de corrección de ambos sensores.

Una de las partes del desarrollo que más tomó tiempo fue la integración del sensor de sol fino. Dado que el sensor usado es una de las primeras iteraciones en su desarrollo, se debieron efectuar varias pruebas que permitieran determinar que sus mediciones eran correctas. En comparación con el FSS de *gomspace* que es un sensor de alto costo, especialmente diseñado para satélites tipo *cubesats*, el FSS utilizado en esta memoria tiene resultados bastante favorables. Aún así, también se reconocen sus desventajas, como lo son su menor campo de visión y la incógnita en su precisión que hasta ahora no ha podido ser medida. Estas razones pudieron influir en los malos resultados obtenidos, utilizando el ESKF con la corrección del FSS.

Para siguientes iteraciones en el desarrollo del sistema de estimación de orientación, se deben considerar FSS en las 6 caras del cubo. También puede ser un gran avance diseñar pruebas que utilicen como fuente del iluminación el sol y no lámparas.

Bibliografía

[1] G. Arencibia, F. E. Hernandez, J. Menéndez, J. R. Rodríguez and A. Pérez, "Estimación de orientación, basada en filtro de kalman, usando unidad de medición inercial sin magnetometro", *Revista Investigación Operacional*, VOL. 41 NO. 3, 369-378, January 2020.

[2] P. J. Cárdenas, E. A. Barros, "Filtro de Kalman extendido aplicado en la navegación de un AUV", *Ingenius*, no. 13, pp 12-19, June 2015.

[3] C. Paz, G. Infante, J. Báes, F. Diaz, C. Cavenio, "Implementación de un filtro extendido de kalman para la estimación de la orientación de un UAV utilizando el estándar CMSIS", In Proc. V Congreso de Microelectrónica Aplicada - UEA2014, Córdova, Argentina, 2014.

[4] M. J. Sidi, *Spacecraft Dynamics and Control: A Practical Engineering Approach*, New York: Cambridge University Press, 1997.

[5] H. Curtis, *Orbital Mechanics for Engineering Students*. 3rd ed. Amsterdam: Elsevier Butterworth Heinemann, 2005.

[6] J. Voight, *Quaternion algebras*, Graduate Texts in Mathematics, vol. 288, Hanover NH:Springer, 2021.

[7] J. Sola, "Quaternion kinematics for the error-state kalman filter," Tech. Rep. IRI-TR-16-02, Institute Robòtica i Informàtica Industrial, CSICUPC, November 2017.

[8] M. Teshnizi and A. Shirazi, "Attitude Estimation and Sensor Identification Utilizing Nonlinear Filters Based on a Low-Cost MEMS Magnetometer and Sun Sensor". IEEE Aerospace and Electronic Systems Magazine. vol. 30. pp. 20-33. December 2015.

[9] N. Trawny and S. I. Roumeliotis, "Indirect Kalman filter for 3D attitude estimation," University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep., Mar. 2005.

[10] "MPU-9250 Product Specification Revision 1.0", Invensense Inc., United States of America, Datasheet, RM-MPU-9250A-00 v1.0, 2014.

[11] MABO, "NanoSense, Fine Sun Sensor Datasheet High precision and ultra small vector sun sensor with digital interface", GomSpace A/S, Denmark, Datasheet, 1018157 v2.4, 2019.

[12] "NanoSense, Fine Sun Sensor Manual High precision and ultra small vector sun sensor with digital interface", GomSpace A/S, Denmark, Manual, v2.1, 2018.

[13] "MPU-9250 Register Map and Descriptions Revision 1.4", Invensense Inc., United States of America, Register Map, RM-MPU-9250A-00 v1.4, 2013.

[14] "Filtros digitales 2", class notes for cm2007, Center for Computer Research in Music and Acoustics, Stanford University, Summer 2007.

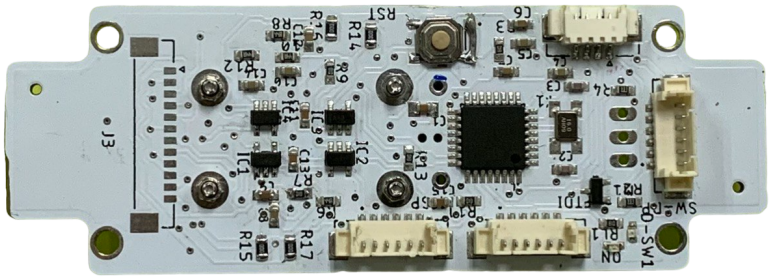
[15] R. G. Lyons211-283, *Understanding Digital Signal Processing*, 2nd ed., New Jersey: Pearson Education. 2004.

[16] "BPW 34S DIL SMT Silicon PIN Photodiode", OSRAM Opto Semiconductors GmbH, Chinese, Datasheet, V1.7, 2021.

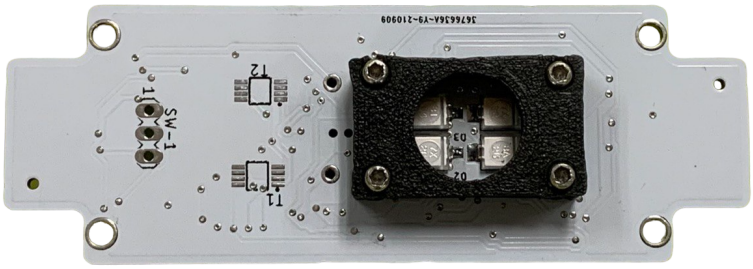
Anexos

Anexo A. Sensor de Sol Fino

La Figura A.1 muestra el diseño tipo B del FSS. Este se compone de una electrónica muy similar al diseño A. El mayor cambio visible es el diseño del borde de la PCB.



(a) PCB por delante

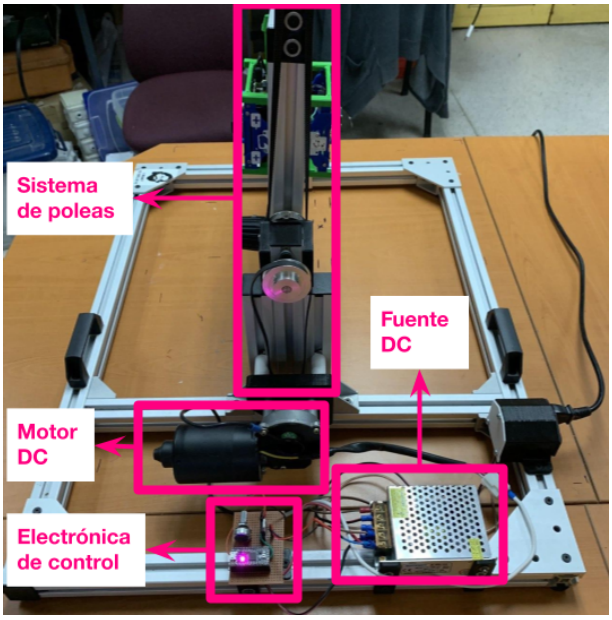


(b) PCB por atrás

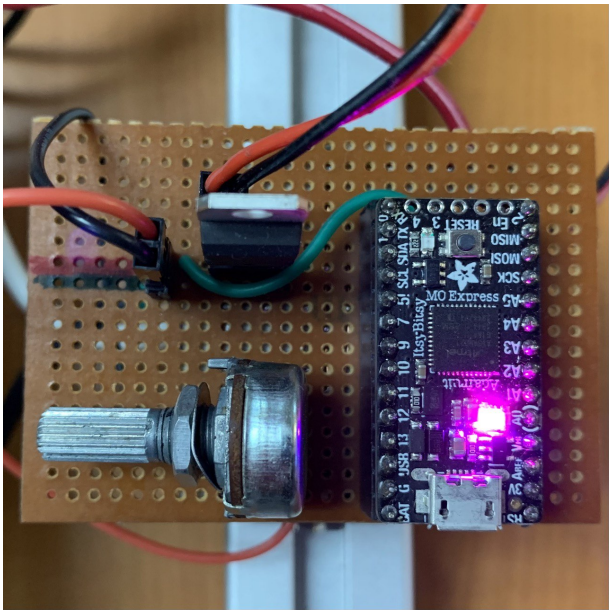
Figura A.1: Diseño de FSS tipo B

Anexo B. Plataforma de prueba

La Figura B.1(a) muestra las partes que componen la máquina. Se observa una fuente DC que alimenta el motor DC y la electrónica de control. El motor DC está acoplado a un sistema de poleas que mueve la estructura en dos ejes. Este se controla con una señal *pwm* que es amplificada con un transistor *mosfet*. El *duty cycle* del *pwm* es controlado a través de un potenciómetro, con el cual, finalmente, se controla la velocidad de giro del motor. En la Figura B.1(b) se detallan los componentes de la PCB de la electrónica de control que se desarrolló para la máquina. Se compone de una tarjeta de desarrollo llamada *Itsy Bitsy M0* que utiliza un microntrolador SamD21. Esta es alimentada desde un regulador de voltaje LM7805.



(a) Componentes de la máquina



(b) Componentes de la PCB de la electrónica de control de la máquina.

Figura B.1: Partes de la plataforma de pruebas

B.1. Acople de satélite de prueba a la máquina

En la cara $y-$ se ubican pernos "M3" con tuercas "tipo T" que se utilizan como sujeción para la integración del satélite en la plataforma de prueba. Estas se pueden ver en la Figura B.2. Las rotaciones que se hacen en la plataforma de prueba son principalmente respecto a esta cara, por lo tanto, es relevante tener particular cuidado con la distribución de masas dentro del *cubesat*, de tal forma que la mayor concentración de masa esté cerca a esta cara. Esto es útil para evitar que el propio peso del satélite ejerza un torque respecto a la sujeción e intervenga con el movimiento de la máquina.

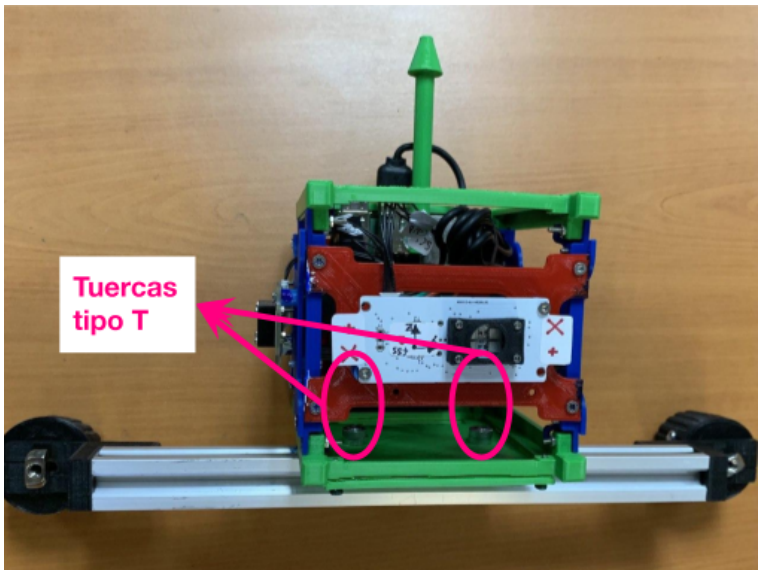


Figura B.2: Acople del satélite de prueba a la estructura de la máquina.

Anexo C. Montaje para toma de pruebas

La Figura C.1 muestra el montaje para realizar cada una las pruebas. La plataforma de prueba mantiene el satélite en movimiento y, en paralelo, en el computador se calcula y visualiza en tiempo real la orientación utilizando el DMP y el ESKF por separado.

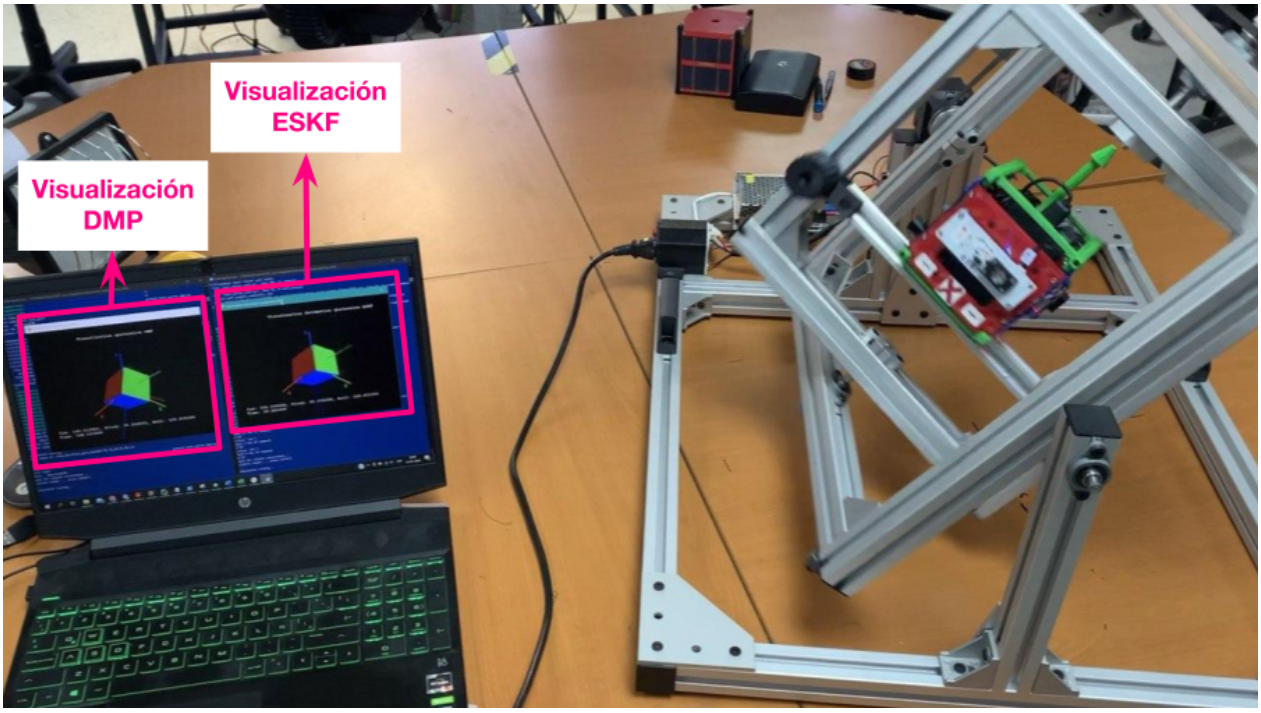


Figura C.1: Montaje para toma de pruebas. La visualización de la izquierda del computador corresponde a la orientación calculada por el DMP. La de la derecha corresponde a la orientación estimada por el ESKF.

Anexo D. Variables guardadas en la ejecución del ESKF

Tabla D.1: Variables medidas y guardadas al ejecutar el ESKF

Descripción variable	**Nombre de la columna**	**Índice**
Tiempo actual en que se toma un paquete de datos. Medido en segundos.	time[sec]	0
Diferencia de tiempo entre un paquete de datos y otro. Medido en segundos.	dtime[sec]	1
Cuaternión estimado del satélite.	ADCS_q_estEskfTemp_i2b(0)[-]	2
	ADCS_q_estEskfTemp_i2b(1)[-]	3
	ADCS_q_estEskfTemp_i2b(2)[-]	4
	ADCS_q_estEskfTemp_i2b(3)[-]	5
Ángulos de euler. Medido en radianes [rad].	ADCS_eskf_phi(X)[rad]	6
	ADCS_eskf_phi(Y)[rad]	7
	ADCS_eskf_phi(Z)[rad]	8
Velocidad angular en 3 ejes medido en el sistema de referencia móvil. Medido en $[rad/s]$	gyro_omega_ADCS_c(X)[rad/s]	9
	gyro_omega_ADCS_c(Y)[rad/s]	10
	gyro_omega_ADCS_c(Z)[rad/s]	11
Velocidad angular en 3 ejes filtrado pasa bajos en tiempo real. Medido en $[rad/s]$	gyro_omega_ADCS_c_filtered(X)[rad/s]	12
	gyro_omega_ADCS_c_filtered(Y)[rad/s]	13
	gyro_omega_ADCS_c_filtered(Z)[rad/s]	14
Vector de campo magnético (3 ejes) de la tierra medido en el sistema de referencia móvil. Medido en nano Tesla [nT].	magnetometer_vect_ADCS_c(X)[nT]	15
	magnetometer_vect_ADCS_c(Y)[nT]	16
	magnetometer_vect_ADCS_c(Z)[nT]	17
Vector de campo magnético de la tierra filtrado pasa bajos en tiempo real. Medido en [nT]	magnetometer_vect_ADCS_c_filtered(X)[nT]	18
	magnetometer_vect_ADCS_c_filtered(Y)[nT]	19
	magnetometer_vect_ADCS_c_filtered(Z)[nT]	20
Vector de campo magnético estimado por el ESKF. Vector Normalizado	magnetometer_vect_ADCS_est(X)[]	21
	magnetometer_vect_ADCS_est(Y)[]	22
	magnetometer_vect_ADCS_est(Z)[]	23
Componentes de la diagonal y determinante de la matriz de covarianzas del ESKF	ADCS_P_est(0,0)[-]	24
	ADCS_P_est(1,1)[-]	25
	ADCS_P_est(2,2)[-]	26
	ADCS_P_est_det[-]	27
Error estimado del ESKF. Medido en [rad].	ADCS_error_est_dtheta(X)[rad]	28
	ADCS_error_est_dtheta(Y)[rad]	29
	ADCS_error_est_dtheta(Z)[rad]	30
Residuo del ESKF. Se calcula como (z-h) en el ESKF.	ADCS_eskfRes(X)[-]	31
	ADCS_eskfRes(Y)[-]	32
	ADCS_eskfRes(Z)[-]	33
Vector de dirección de sol normalizado medido con el FSS n. Vector normalizado	ADCS_fss_(n)_rx_[-]	34
	ADCS_fss_(n)_ry_[-]	35
	ADCS_fss_(n)_rz_[-]	36
Angulo de vector de sol. Medido en [rad]	ADCS_fss_(n)_phi_[rad]	37
	ADCS_fss_(n)_theta_[rad]	38

Anexo E. Variables guardadas en la ejecución del DMP

Tabla E.1: Variables medidas y guardadas al ejecutar el DMP

Descripción variable	Nombre de la columna	Indice
Tiempo actual en que se toma un paquete de datos. Medido en segundos.	time[sec]	0
Cuaternión de referencia (real) calculado por DMP.	q_t_i2b(0)[-]	2
	q_t_i2b(1)[-]	3
	q_t_i2b(2)[-]	4
	q_t_i2b(3)[-]	5
Ángulos de euler. Medido en radianes [rad].	ADCS_eskf_true_phi(X)[rad]	6
	ADCS_eskf_true_phi(Y)[rad]	7
	ADCS_eskf_true_phi(Z)[rad]	8
Velocidad angular en 3 ejes medido en el sistema de referencia móvil. Medido en $[rad/s]$	omega_t_b(X)[rad/s]	9
	omega_t_b(Y)[rad/s]	10
	omega_t_b(Z)[rad/s]	11