Seguidor de satélites LEO a lazo abierto para comunicaciones ópticas

Autor:
Alan MELLONI

Tutor:
Dr. Federico IZRAELEVITCH

Informe de Proyecto Final Integrador para optar por el título de Ingeniería electrónica

desarrollado en

Laboratorio de Integración NanoElectrónica
Escuela de Ciencia y Tecnología

17 de noviembre de 2022

«El presente es de ellos; el futuro, para el que tanto trabajé, es mío.»

Nikola Tesla

UNIVERSIDAD NACIONAL DE SAN MARTÍN

Resumen

Escuela de Ciencia y Tecnología

Ingeniería electrónica

Seguidor de satélites LEO a lazo abierto para comunicaciones ópticas

por Alan MELLONI

El proyecto LabOSat cuenta con 9 misiones en órbita, donde se probaron distintos dispositivos electrónicos. Una de las ramas de investigación del proyecto busca caracterizar fotomultiplicadores de Silicio, sensor capaz de detectar fotones individuales. Para lograr esto, uno de los experimentos busca apuntar con un haz de luz a un fotomultiplicador a bordo de un satélite LEO. Con este fin, se desarrolló una estación terrena, encargada de seguir la órbita del satélite, la cual operará a lazo abierto, es decir, sin obtener realimentación de la posición del satélite. Para predecir la órbita, se implementó el algoritmo SGP4. Sobre la estación irá montada una fuente de luz que no forma parte de este proyecto.

Se desarrolló un prototipo encargado de realizar el seguimiento. El mismo Fue requisito de diseño que el sistema opere de manera independiente, sin estar conectado a otro dispositivo. Esto implicó que el algoritmo de predicción sea implementado con anterioridad a la pasada. Para realizar el seguimiento, se utilizaron dos motores paso a paso.

Por otra parte, se indagó sobre la incertidumbre causada por el algoritmo, el paso de los motores y por el intervalo de tiempo entre cada punto de la órbita calculado, como también la relación entre ellos.

Se comprobó visualmente el correcto funcionamiento de la estación terrena. No obstante, no fue posible verificar la exactitud del seguimiento, ya que el fotomultiplicador no se encontraba en órbita para cuando se finalizó el diseño.

Índice general

Índice de figuras

Índice de tablas

A la Scaloneta

Capítulo 1

Introducción

La industria espacial es una actividad que requiere aportes de diferentes áreas del conocimiento para llevar a cabo proyectos de alta complejidad. Entre las múltiples disciplinas que participan en la concreción de estos proyectos podemos mencionar a la electrónica, física, informática, mecánica o telecomunicaciones, entre otras. A su vez, el ensayo y caracterización de dispositivos y sistemas electrónicos es de especial importancia debido a la necesidad de desarrollar misiones espaciales con componentes probados, de forma de reducir el riesgo de fallas. Desde hace casi una década, el Grupo LabOSat aborda este problema realizando ensayos y experimentos en Tierra y en órbita, gracias a la utilización de una plataforma de caracterización dedicada diseñada *ad hoc* (LabOSat-01, LS-01) para estudiar el comportamiento de dispositivos electrónicos fabricados a baja escala en el laboratorio o dispositivos comerciales (COTS, *Commercial off-the-shelf*) [1]. En 9 misiones en órbita baja terrestre (LEO, *Low Earth Orbit*), el Grupo ha estudiado dispositivos de conmutación resistiva (RRAM), transistores de efecto de campo (FET, *Field Effect Transistor*) de película delgada y transistores de tecnología SOI (*Silicon on insulator*) así como una multiplicidad de dispositivos COTS, entre los que se pueden mencionar los fotomultiplicadores de silicio (SiPMs) [2].

Actualmente, estos últimos conforman una importante rama de investigación dentro del Grupo. Los fotomultiplicadores de silicio fueron desarrollados a comienzos de los años dos mil, como resultado de la búsqueda de sensores de fotones de mayor sensibilidad y rapidez, bajo el impulso de la comunidad de Física de Partículas Elementales. En la última década, a pesar de que la tecnología no se encuentra completamente madura, las aplicaciones de SiPMs han proliferado. Entre otras, podemos mencionar los tomógrafos por emisión de positrones [3], sistemas de conducción asistida por LIDAR por tiempo de vuelo directo [4], detectores de quimioluminiscencia [5] y captura de imágenes de fluorescencia de dos fotones tejidos [6]. En este marco, el Grupo LabOSat busca caracterizar estos dispositivos en el frente de las comunicaciones espaciales.

1.1. Motivación

Con el fin de caracterizar los SiPM en el ámbito de las comunicaciones espaciales, se desea establecer un enlace Tierra-Aire, con el fin de detectar fotones individuales en órbita. Para este fin, una fuente de luz debe ser montada sobre una estación terrena encargada de apuntar a un satélite durante toda su órbita visible. Este proyecto consiste en el desarrollo de dicha estación.

1.2. Objetivos y consideraciones

Se debe desarrollar un sistema que sea capaz de apuntar a un satélite durante la parte visible de su órbita. El seguimiento será realizado a lazo abierto, es decir, sin recibir datos de posición por parte del satélite, por lo que deberá predecirse su desplazamiento con anterioridad. Además, debe ser portable e independiente, es decir, debe funcionar sin conexión alguna con otro dispositivo. El sistema debe llevar montada la fuente de luz utilizada para la comunicación óptica.

1.3. Relación Costo-Precisión

Hay dos parámetros presentes a la hora de decidir la precisión del sistema: la resolución mecánica y el ancho del haz de luz emitido a la altura del satélite (Figura 1.1). Mientras mayor sea el ancho del haz, menor será la resolución mecánica requerida. Sin embargo, agrandar el ancho implica un aumento en la potencia necesaria para que el número de fotones por unidad de área se mantenga constante, lo que se traduce en costos más elevados. Asimismo, el aumento de la resolución mecánica también conlleva mayores costos. Por lo tanto, debe lograrse un balance entre ambos parámetros, considerando los costos que suponen sus variaciones.

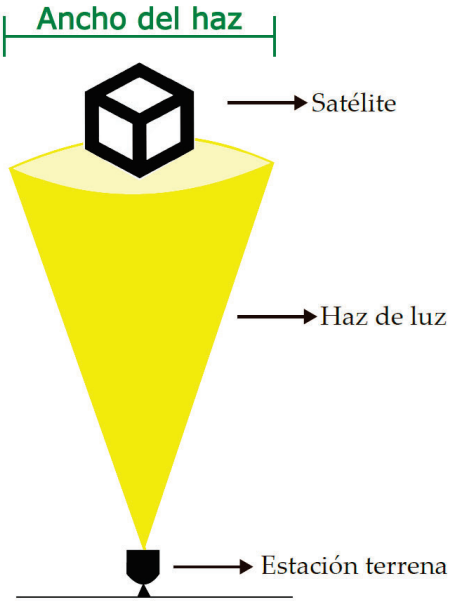


FIGURA 1.1: Representación del haz de luz emitido. El ancho del haz de luz es un parámetro de diseño.

Es importante notar que la predicción de la órbita del satélite cuenta con un error intrínseco (Sección 2.3.3), que establecerá un ancho de luz mínimo.

Para simplificar el proceso de diseño, la estación terrena fue desarrollada considerando la precisión mecánica como parámetro de entrada, de forma que la misma pueda variarse según se necesite, sin requerir una modificación del sistema.

Capítulo 2

Conceptos fundamentales

2.1. Sistema de referencia

Para poder realizar el seguimiento del satélite es necesario conocer su posición con respecto al observador en Tierra. Para esto, se utiliza el sistema de coordenadas horizontal, el cual toma el horizonte como plano fundamental y define dos coordenadas angulares:

- Azimutal: ángulo con respecto al norte geográfico. Toma un valor entre 0° y 360°, comenzando en el Norte y aumentando hacia el Este.
- Elevación: ángulo con respecto al horizonte. Comienza en 0° y aumenta hasta 90°, donde se encuentra el cénit[1] del observador.

La Figura 2.1 ilustra el sistema de coordenadas mencionado:

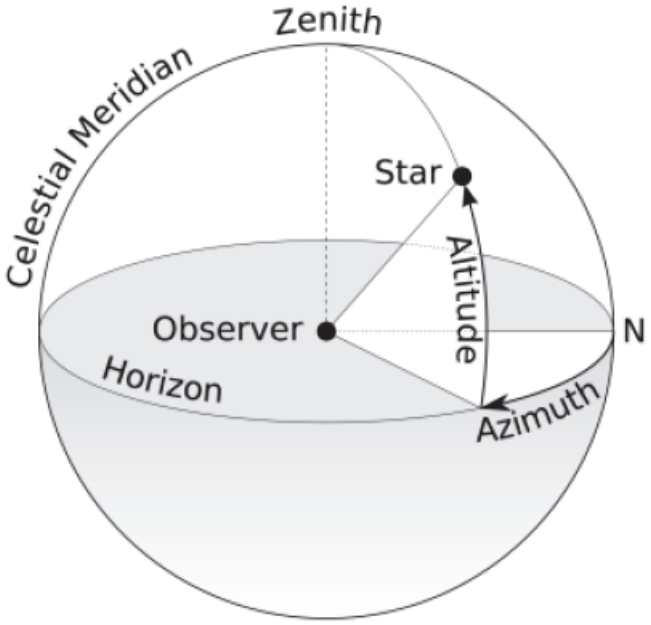


FIGURA 2.1: Sistema de coordenadas horizontal. Se muestran los ángulos azimutal y elevación (también denominado "altura") [7].

Para realizar el seguimiento del satélite, se requiere conocer sus coordenadas horizontales durante toda su órbita visible, junto con las coordenadas geográficas del sistema.

[1] Punto del hemisferio celeste situado sobre la vertical del observador. Ver Figura 2.1.

2.2. Tipos de órbitas satelitales

Al clasificar las órbitas satelitales por altitud, se pueden identificar cuatro tipos de órbitas:

- LEO (*Low Earth Orbit*): órbitas entre 200 y 2000 kilómetros.
- MEO (*Medium Earth Orbit*): órbitas entre 2000 y 35786 kilómetros.
- GEO (*Geostationary Orbit*): órbitas de 35786 kilómetros.
- HEO (*Highly Elliptical Orbit*): órbitas con apogeo[2] de más de 35786 kilómetros.

Debido al algoritmo utilizado, el sistema será capaz de realizar el seguimiento de satélites de órbitas LEO, o MEO de baja altitud.

2.3. Cálculo de la órbita LEO

El primer paso para realizar el seguimiento consiste en conocer la posición del objetivo durante toda la fracción visible de su órbita. Esto se logra mediante el algoritmo SGP4 (*Simplified General Perturbations*) [8].[3]

2.3.1. Algoritmo SGP4

El algoritmo es utilizado para obtener los vectores de estado orbital (posición y velocidad) de satélites y fragmentos de basura espacial. Este modelo tiene en cuenta perturbaciones como el arrastre causado por la atmósfera, la geometría de la Tierra, radiación, la atracción gravitatoria del Sol y la Luna, entre otros.

El modelo es válido para órbitas de períodos menores a 225 minutos, lo que equivale a altitudes de menos de 5800 kilómetros [8]. El algoritmo SDP4 (*Simplified Deep Space Perturbations*) se encarga de estimar el estado de objetos más lejanos.

Para implementarlo, es necesario indicar un instante de tiempo, las coordenadas geográficas del observador y un *Two-line Element* (TLE), el cual será explayado en la Sección 2.3.2. Así, el algoritmo devuelve los ángulos azimutal y elevación para el instante dado. Esto se representa en la Figura 2.2.

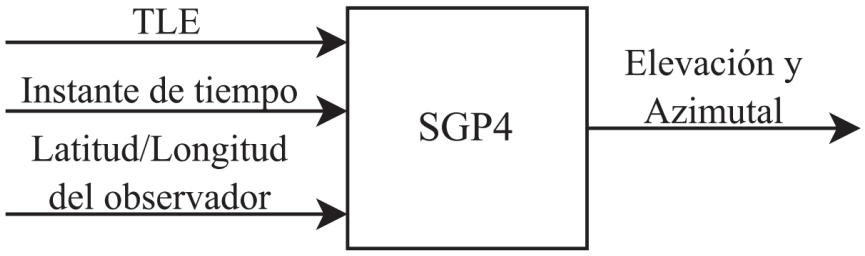


FIGURA 2.2: Esquema del algoritmo SGP4. Brindándole un instante de tiempo, las coordenadas geográficas del observador y un TLE, el algoritmo computa las coordenadas horizontales del objeto.

[2] Altitud máxima alcanzada en la órbita.

[3] El número 4 en su sigla lo diferencia de una versión anterior del mismo algoritmo: SGP.

2.3.2. *Two-line Element*

Cada objeto que orbita la Tierra está descripto por un TLE, dos líneas de texto que describen su órbita para un instante de tiempo en particular denominado "época" (*epoch*, en inglés). La Fuerza Espacial Estadounidense (anteriormente conocida como Comando Espacial de la Fuerza Aérea) realiza observaciones sobre la mayoría de los objetos que orbitan la Tierra, genera los TLE correspondientes y los publica en la web [9].

Este elemento contiene información como el número de catálogo del satélite, fecha de lanzamiento, época, excentricidad de la órbita, inclinación, entre otros. En la Figura 2.3 se muestra un ejemplo de un TLE, mientras que en el Anexo A se describen todos sus campos.

1 40941U 15054B 22204.47618715 -.00000223 00000+0 00000+0 0 9998
2 40941 0.0517 278.8333 0001518 198.7676 273.9990 1.00271733 25039

FIGURA 2.3: Ejemplo de un TLE del ARSAT 2.

Si bien el TLE informa las características de la órbita para un instante de tiempo determinado (su época), el algoritmo SGP4 extrapola para otros instantes, tanto futuros como pasados, estimando así el estado del objeto. Sin embargo, realizar estimaciones para momentos alejados de la época implica errores considerables. Esto será ampliado en la Sección 2.3.3.

Por simplicidad, la diferencia entre la época del TLE y el instante de tiempo aplicado por el algoritmo se representará como $\Delta t_{época}$.

$$\Delta t_{\text{época}} = \text{ Instante utilizado por SGP4} - \text{ Época del TLE} \tag{2.1}$$

Un $\Delta t_{época}$ mayor que cero indica que el algoritmo está siendo implementado para un punto posterior a la época, mientras que uno menor que cero indica que está siendo implementado para un punto anterior.

2.3.3. Incertidumbre intrínseca

Existen dos factores que contribuyen significativamente al error del algoritmo (δP). Por un lado, el arrastre causado por la atmósfera aumenta al disminuir la altitud del satélite, causando que las predicciones resulten menos exactas. Por el otro, a medida que el instante de tiempo utilizado se aleja de la época del TLE, el error aumenta.

Varios estudios enuncian una incertidumbre de entre 1 y 3 kilómetros en la época ([10], [11]), mientras que en [12] se presenta la Tabla 2.1, que enuncia el error del algoritmo para órbitas de diferentes altitudes.[4]

[4]Los valores de error mostrados en la Tabla 2.1 corresponden a órbitas de excentricidades cercanas a cero, es decir, casi circulares. En [12] se encuentra la tabla completa con distintas excentricidades.

Altitud de apogeo [km]	Error de algoritmo (δP) [km]
200–300	26,97
300–400	3,50
400–500	2,34
500–600	1,04
600–700	0,37
700–800	0,31
800–900	0,34
900–1000	0,20

TABLA 2.1: Relación entre altitud de apogeo y error intrínseco [12].

Queda en evidencia que, desde el punto de vista de la estación terrena, es ideal pensar en una misión cuya altitud sea la mayor posible, de forma de tener mayor exactitud en el conocimiento de su trayectoria y reducir los requerimientos de apuntamiento. Sin embargo, existe una relación de compromiso, ya que el aumento de altitud implica una mayor potencia del haz de luz necesaria.

Por otro lado, las estimaciones realizadas a partir de un TLE empeoran a medida que se alejan de la época. La Figura 2.4, extraída de [13], muestra la relación entre el error RMS en kilómetros y la diferencia de días con la época para tres satélites diferentes. Se observa que incluso para una estimación un día luego de la época, el error llega a duplicarse. A causa de esto, los TLE son actualizados varias veces por día, por lo que un sistema que estime el estado de un satélite de manera prolongada deberá consultar una base de datos constantemente.

Existen varios repositorios que contienen los TLE de todos los satélites activos, por ejemplo [14] y [15].

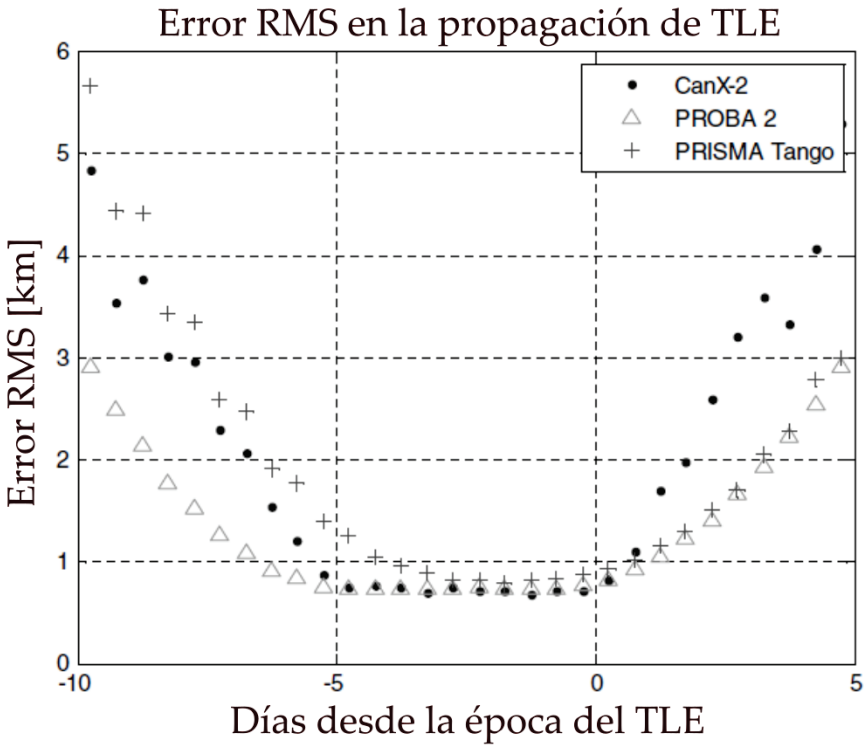


FIGURA 2.4: Error RMS de la estimación con respecto a la diferencia de días con la época para tres satélites diferentes. Extraído de [13].

Con la intención de explorar experimentalmente esta incertidumbre, se utilizaron datos de posicionamiento de una de las placas LabOSat-01, a bordo del satélite ÑUSAT-7, y se compararon con la predicción del algoritmo SPG4 para ese satélite, implementado con un programa de *Python* desarrollado *ad hoc*. Este satélite orbita la Tierra a aproximadamente 500 kilómetros de altura.

Como únicamente se contaba con tres instantes de tiempo cercanos entre sí, se optó por considerar un instante en particular e implementar el algoritmo con distintos TLE. Al tener $\Delta t_{época}$ diferentes, es esperable que aquellos cuyas épocas estén alejadas del punto a calcular cuenten con errores más considerables. En particular se utilizaron tres TLE, uno con época 2 horas después del punto, otro 6 horas antes, y otro 40 horas antes, este último con el fin de evaluar el error al emplear un TLE desactualizado.

Los resultados se muestran en la Tabla 2.2.

$\Delta t_{época}$ [hs]	Error de algoritmo (δP) [km]
-2	2,25
6	3,03
40	8,1

TABLA 2.2: Error del algoritmo SGP4 para distintos $\Delta t_{época}$ obtenido experimentalmente. Se puede ver que el error aumenta significativamente luego de 40 horas.

Se obtuvo un δP de 3 km para puntos 6 horas luego de la época. Además, al alejarse más de un día, el error aumenta significativamente. Se observó que los TLE de este satélite se actualizan cada 12 horas aproximadamente, por lo que nunca se presentará el último caso.

2.3.4. Incertidumbre angular intrínseca

Conociendo el error δP, se puede obtener la incertidumbre angular intrínseca del sistema, es decir, la incertidumbre de las coordenadas horizontales devueltas por el algoritmo.

Es necesario notar que el error angular aumenta a medida que la distancia entre el satélite y el observador disminuye, ya que el error del algoritmo es independiente de la posición de este último. Por ende, el momento de mayor error ocurrirá cuando esta distancia sea mínima, que corresponde a cuando el satélite alcanza su elevación máxima. Esto es válido para la incertidumbre de ambos ángulos, azimutal y elevación.

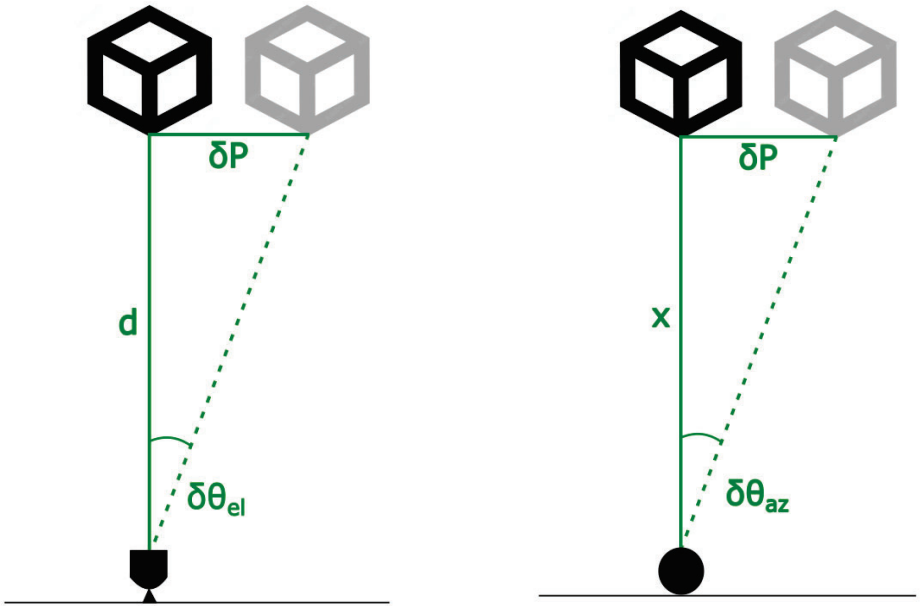
En la Figura 2.5a se muestra una vista lateral, a partir de la cual puede observarse que el error del ángulo de elevación se obtiene según:

$$tan(\delta\theta_{el}) = \frac{\delta P}{d} \tag{2.2}$$

$$\delta\theta_{el} = tan^{-1}\left(\frac{\delta P}{d}\right) \tag{2.3}$$

Por otro lado, la Figura 2.5b muestra una vista superior, con la cual se puede inferir que el error del ángulo azimutal es:

$$\delta\theta_{az} = tan^{-1}\left(\frac{\delta P}{x}\right) \tag{2.4}$$



(A) Vista lateral a partir de la cual se obtiene la expresión del error de elevación.

(B) Vista superior a partir de la cual se obtiene la expresión del error azimutal.

FIGURA 2.5: Diagramas a partir de los cuales se obtienen los errores angulares. En negro se representa la posición real del satélite y en gris, su posición estimada por el algoritmo.

Donde x es la proyección de la órbita sobre la superficie de la tierra. A partir de la altura de la órbita (h) y la distancia entre satélite y observador (d), se obtiene su valor:

$$x = \sqrt{d^2 - h^2} \tag{2.5}$$

Conociendo h y tomando un valor de δP, el cual podría seleccionarse a partir de la Tabla 2.2, se pueden obtener ambos errores angulares en función de la distancia. Como la distancia se corresponde con un ángulo de elevación determinado, se puede graficar en función de este parámetro.[5]

A modo de ejemplo, se toma el *ÑUSAT 7*, cuya órbita tiene una altitud $h \approx 500\ km$, y un error $\delta P = 3\ km$ (según lo obtenido en la Tabla 2.2), obteniendo la Figura 2.6.[6] El error azimutal se representa logarítmicamente debido a su abrupto aumento en altas elevaciones.

Se observa que la componente azimutal del error es dominante. Este notable aumento podría ser inhibido utilizando una montura ecuatorial, la cual presenta el pico de error en los hemisferios (elevaciones cercanas a cero), donde no se busca establecer el enlace óptico.

[5]Realizar esto matemáticamente es complejo, ya que debe tenerse en cuenta la curvatura de la tierra. Para evitar esto, se utiliza una órbita calculada con el algoritmo para obtener el gráfico.

[6]La Figura sobreestima el error ligeramente, con el fin de simplificar el análisis requerido para elaborar los gráficos, ya que toma el vector tridimensional de error. Sin embargo, el error angular se obtiene en base a la proyección bidimensional de dicho vector sobre el plano perpendicular a la visión del observador. Por ejemplo, si el satélite se encuentra en el cénit del observador y existe únicamente un error de 2 km en la componente altitud, el error angular será nulo.

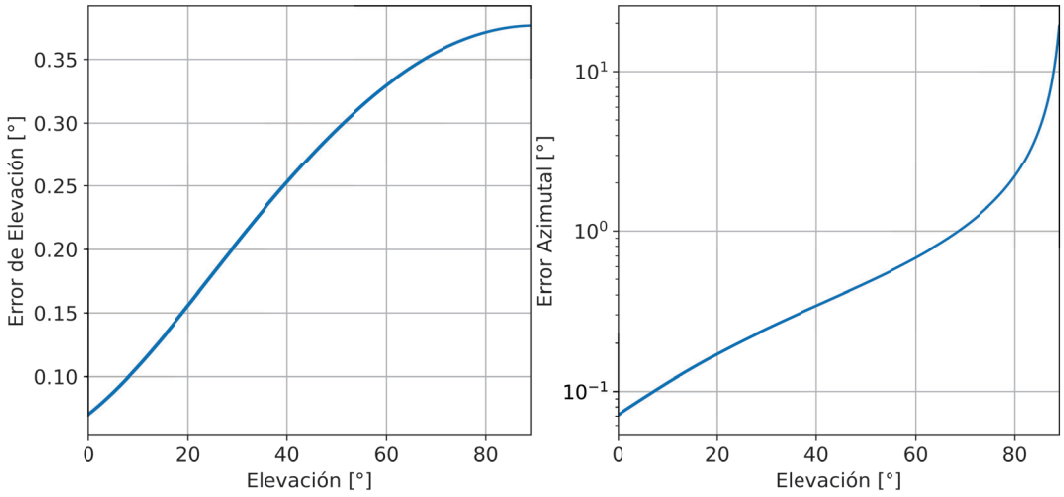


FIGURA 2.6: Errores angulares intrínsecos en función de la elevación para una órbita de 500 km de altura (ÑUSAT-7), considerando un error de algoritmo de 3 km. El error azimutal se representa logarítmicamente.

A partir de estos resultados, debe seleccionarse un valor de resolución mecánica (θ_{mec}) que sea despreciable en comparación a la incertidumbre del algoritmo:

$$\theta_{mec} \ll \delta\theta \tag{2.6}$$

2.4. Intervalos de tiempo del programa

Para poder realizar el seguimiento del satélite durante su órbita visible, es necesario conocer su posición en todo momento. Para lograr esto, debe implementarse el algoritmo SGP4 para la mayor cantidad de instantes posibles. El intervalo de tiempo entre los puntos calculados (Δt) debe minimizarse, ya que el mismo define la frecuencia de actualización del sistema. Entre un punto y otro, es decir, en un tiempo Δt, el satélite recorre un determinado arco, que corresponde a un ángulo de apertura $\Delta\theta$. Esto se muestra en la Figura 2.7.

Con respecto a la resolución mecánica, mientras menor sea θ_{mec}, más movimientos serán necesarios para recorrer el mismo ángulo. Sabiendo esto, se observan dos valores que influyen sobre la precisión angular del sistema: $\Delta\theta$ y θ_{mec}. Si bien ambos deben ser reducidos lo más posible, es más sencillo decrementar $\Delta\theta$: simplemente se requieren más iteraciones.

Por otro lado, el ángulo recorrido entre dos puntos calculados no es constante durante la pasada. En el caso del ángulo de elevación, $\Delta\theta$ es máximo cuando el satélite sale por sobre el horizonte y mínimo en el cénit. Con el ángulo azimutal ocurre lo inverso, $\Delta\theta$ es máximo en el cénit y mínimo en el horizonte. Conociendo el ángulo máximo ($\Delta\theta_{max}$), se puede comparar con θ_{mec} para obtener el Δt adecuado. Se busca entonces que el limitante sea θ_{mec}, para lo cual:

$$\Delta\theta_{max} < \theta_{mec} \tag{2.7}$$

La Figura 2.8a muestra el caso donde $\Delta\theta > \theta_{mec}$, donde la precisión angular del sistema está definida por $\Delta\theta$. La misma podría mejorarse aumentando la cantidad de

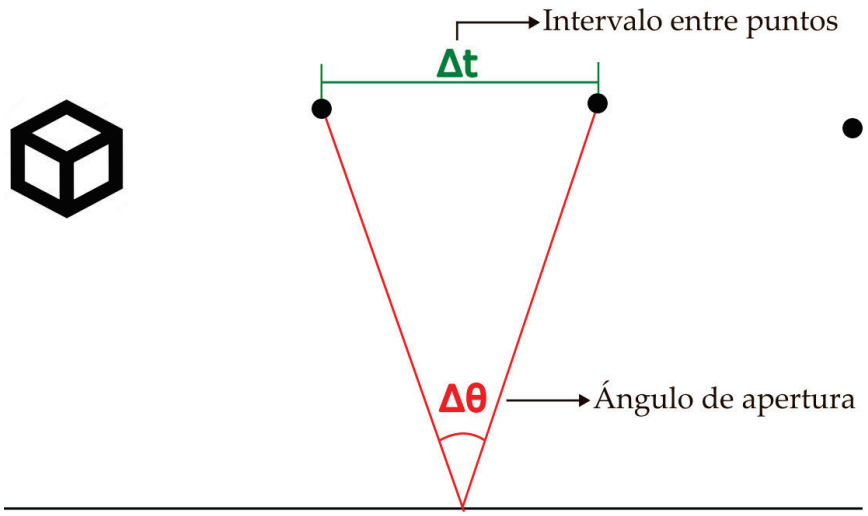


FIGURA 2.7: Ilustración de los parámetros Δt y $\Delta\theta$. Cada punto calculado mediante el algoritmo se ilustra como un círculo negro. El satélite recorre un ángulo de apretura $\Delta\theta$ en un tiempo Δt.

puntos calculados, como se muestra en la Figura 2.8b, donde la precisión es definida por θ_{mec}.

El tiempo de procesamiento del algoritmo aumenta considerablemente al disminuir Δt. La Tabla 2.3 muestra el tiempo requerido para calcular la órbita para distintos Δt, utilizando un programa desarrollado en *Python* (Sección 3.1).[7]

Δt	Tiempo de cómputo
1 s	3 segundos
100 ms	30 segundos
10 ms	15 minutos
1 ms	60 minutos

TABLA 2.3: Tiempo requerido para ejecutar el algoritmo para distintos Δt.

Experimentalmente se observó (mediante el programa de *Python* desarrollado) que un satélite con una órbita de 500 kilómetros de altitud tarda 700 ms en que su elevación pase de 0° a 0,05°. Por otro lado, a una elevación de 80° tarda 200 ms en recorrer 0,05° azimutales.[8]

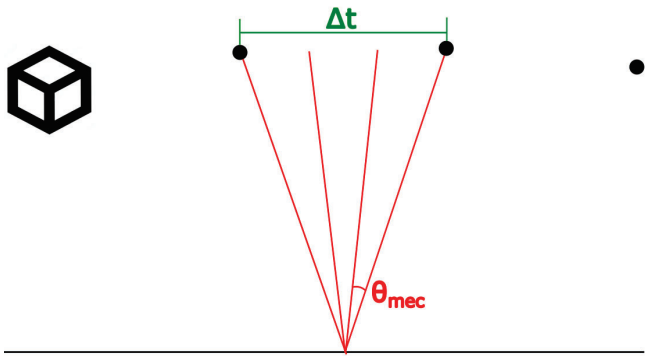
Considerando que el movimiento azimutal es el que más contribuye a la incertidumbre angular, se plantea la siguiente expresión:

$$\Delta t = \frac{200\ ms}{0,05^\circ} \cdot \theta_{mec} \tag{2.8}$$

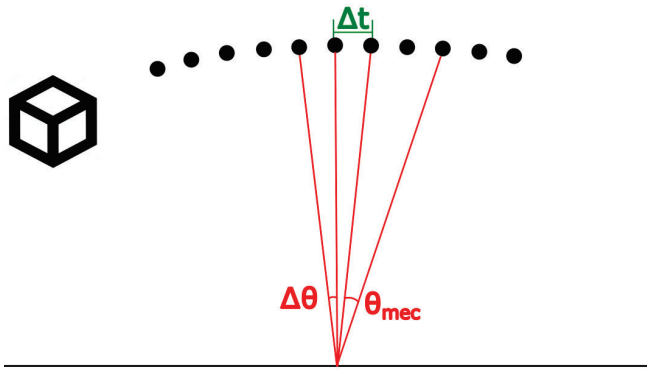
Este análisis está basado en la suposición de que el SiPM estará orbitando a 500 kilómetros, altura a la cual se espera que se encuentren las futuras misiones

[7]El experimento fue llevado a cabo en el IDE *Spyder*, con un procesador Intel i5-7300HQ y 8 GB de RAM.

[8]A elevaciones mayores, cercanas a 90°, el desplazamiento azimutal es mucho más rápido, por lo que se requeriría un $\Delta\theta$ extremadamente pequeño para cumplir con la condición 2.7. Para evitar utilizar Δt muy chicos de forma de alcanzar esta velocidad, se toma 80° como referencia, lo que provocará un seguimiento discontinuo para elevaciones mayores. Esto no es un inconveniente ya que la incertidumbre intrínseca es mucho mayor que el error inducido por la discontinuidad del seguimiento.



(A) El movimiento angular mínimo del sistema, θ_{mec}, es menor que $\Delta\theta$, por lo que cada movimiento constará de varios θ_{mec}. Se está desaprovechando la precisión mecánica del sistema.



(B) θ_{mec} es mayor que $\Delta\theta$, cada movimiento consistirá de un θ_{mec}.

FIGURA 2.8: Casos donde $\theta_{mec} < \Delta\theta$ (A) y $\theta_{mec} > \Delta\theta$ (B).

de LabOSat. El mismo deberá volver a realizarse si la altitud difiere. Cabe destacar que, para órbitas más altas, el error intrínseco disminuye significativamente, por lo que Δt deberá ser menor.

Así, a partir de las expresiones 2.6, 2.7 y 2.8, se pueden definir todos los parámetros necesarios para implementar el algoritmo para un satélite deseado.

Capítulo 3

Obtención de la órbita

Para predecir la pasada del satélite, se desarrolló un programa en *Python*, el cual hace uso de la librería *Skyfield*. La misma posee métodos que permiten implementar el algoritmo SGP4 en un instante determinado a partir de un TLE. El código desarrollado está disponible en [16].

3.1. Implementación del programa

Para calcular la órbita, es necesario especificar los siguientes parámetros de entrada:

- Latitud y longitud del observador.
- Nombre identificador del satélite.
- Intervalo de tiempo entre puntos (Δt).
- Cantidad de grados por movimiento del sistema (θ_{mec}).
- Ángulo de elevación desde el cual se comienza a calcular la órbita. Por ejemplo, si se define como 30, la órbita será calculada únicamente para el intervalo donde el satélite tenga una elevación mayor a 30°.

La Figura 3.1 muestra un diagrama de alto nivel del programa implementado. Se observa que, una vez obtenida la órbita, la misma se convierte a pasos de motor, para luego comprimir cada dato en 4 bytes y enviar la información al microcontrolador. Esto será ampliado a continuación.

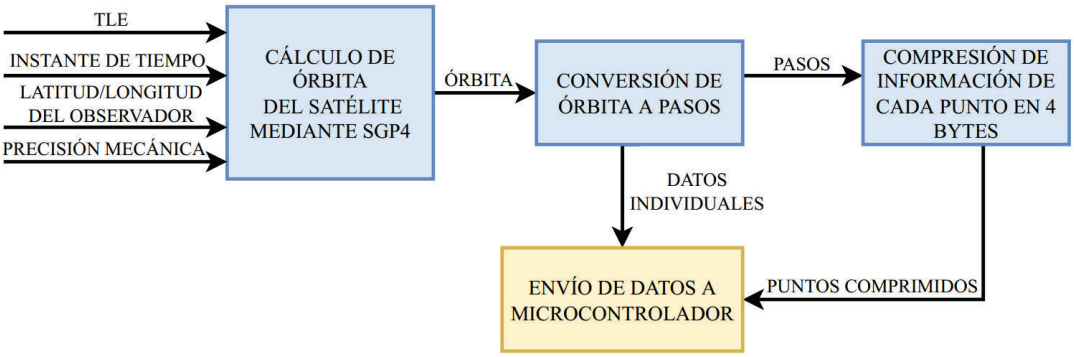


FIGURA 3.1: Diagrama de alto nivel del programa de *Python* desarrollado para predecir la órbita del satélite.

A partir de los parámetros de entrada, el programa calcula los puntos necesarios aplicando el algoritmo SGP4. Inicialmente, se genera una tabla con todos los puntos

calculados desde el inicio hasta el final de la órbita. Cada uno contiene el instante de tiempo y los ángulos azimutal y elevación. El tiempo se especifica en base al sistema UNIX, debido a que es el utilizado por *Skyfield*, el cual representa un instante como la cantidad de segundos desde la medianoche del 1 de enero de 1970, mientras que los ángulos se representan en grados. A modo de ejemplo, se muestran en la Tabla 3.1 algunos de los puntos devueltos por el programa, donde se utilizó $\Delta t = 100$ ms.

Punto	Tiempo [s]	Elevación [°]	Azimutal [°]
...	...	...	...
2537	1659995768.483	38.707	254.586
2538	1659995768.583	38.753	254.570
2539	1659995768.683	38.801	254.554
2540	1659995768.783	38.847	254.538
...	...	...	...

TABLA 3.1: Puntos de la órbita calculados, que consisten en los ángulos de elevación y azimutal para un instante de tiempo.

Con el fin de reconocer en qué momento se debe realizar un movimiento, se toma la derivada de ambos ángulos, es decir, se obtiene la diferencia entre cada valor y el anterior, y se colocan en nuevas columnas.

Luego, se divide el valor de la derivada obtenido por la resolución mecánica (θ_{mec}). El cociente (pasos) que deberán realizarse en ese instante, mientras que el resto de la división se acumula a la derivada del siguiente punto. De esta forma, se obtienen las columnas de pasos de ambos ángulos.

Se muestra un ejemplo en la Tabla 3.2.

Punto	Tiempo [s]	Δ El. [°]	Δ Az. [°]	Pasos El.	Pasos Az.
...	...	...	...	...	...
2537	1659995768.483	0.0463	-0.0158	1	0
2538	1659995768.582	0.0463	-0.0159	1	0
2539	1659995768.683	0.0464	-0.0159	0	0
2540	1659995768.783	0.0465	-0.0160	1	1
...	...	...	...	...	...

TABLA 3.2: Puntos de la órbita junto con la derivada de los ángulos y la cantidad de pasos por punto.

De esta forma, toda la información de la órbita necesaria para realizar el seguimiento está contenida en las columnas de tiempo y de pasos. Por ende, se elimina el resto de las columnas. Por otro lado, los puntos que no contienen pasos no aportan información, por lo que también son eliminados. En consecuencia, se disminuye la cantidad de datos que debe ser almacenada.

Sin embargo, mediante este procedimiento se pierde información crítica para el seguimiento: los ángulos azimutal y elevación iniciales, el instante donde la elevación del satélite comienza a disminuir y el sentido de giro azimutal. Estos datos deben ser almacenados por separado.

Los ángulos iniciales fueron obtenidos anteriormente (consisten en el primer punto obtenido al aplicar el algoritmo). El instante donde la elevación empieza a disminuir corresponde al punto donde ocurre un cambio de signo en su derivada. El sentido de giro azimutal se obtiene a partir del signo de su derivada: si el signo es positivo, el movimiento es horario, si es negativo, antihorario.

Por otro lado, el sistema UNIX está pensado para que el valor en segundos ocupe 32 bits. Como en esta aplicación es necesaria mayor exactitud, es decir, se deben contemplar los milisegundos de cada instante, el tamaño del valor de tiempo supera la cantidad de bits mencionada.

Dado que la órbita deberá ser almacenada, se busca minimizar el espacio requerido, para lo cual se opta por tomar la parte entera del valor de tiempo del primer punto de la tabla y restarlo al resto de los puntos, quedando este valor como tiempo de referencia, que será almacenado por separado. Además, para simplificar el envío de los datos hacia el microcontrolador, se expresan los tiempos directamente en milisegundos. De esta forma, la salida del programa queda según se muestra en la Tabla 3.3.

Punto	Tiempo [ms]	Pasos El.	Pasos Az.
...	...	...	...
2537	254683	1	0
2538	254782	1	0
2540	254983	1	1
...	...	...	...

Parámetro	Valor
Tiempo de referencia [s]	1659995514
Elevación de inicio	0
Azimutal de inicio	1410
Cambio de elevación [ms]	270907
Sentido azimutal	-1

TABLA 3.3: Formato final de la órbita. Únicamente se almacenan los tiempos con respecto al tiempo de referencia y los pasos de cada punto. Además, se guardan por separado otros parámetros necesarios para realizar el seguimiento.

La información de cada punto debe comprimirse lo más posible para disminuir la necesidad de memoria. Se opta por describir cada punto en 32 bits, como se muestra en la Tabla 3.4. Se dedican 4 bits para cada valor de pasos, permitiendo almacenar una cantidad máxima de $2^4 - 1 = 15$ pasos por punto, y 24 para el valor de tiempo en milisegundos, alcanzando un valor máximo de $2^{24} - 1 = 16777216$, o 279 minutos. Siendo que una pasada de un satélite LEO dura aproximadamente 15 minutos, hay espacio suficiente para almacenar los valores.

bits [31:8]	bits [7:4]	bits [3:0]
Instante de tiempo [ms]	Pasos Az.	Pasos El.

TABLA 3.4: Distribución de cada punto en 32 bits. El valor de tiempo ocupa las posiciones 31 a 8, mientras que los campos de pasos ocupan 4 bits cada uno (7 a 4 y 3 a 0).

Cabe destacar que si el Δt es suficientemente grande como para que haya más de 15 pasos por punto, se desfasará el seguimiento. Esto no es un inconveniente, ya que para estos casos la precisión es tan baja que el experimento carece de sentido. De esta forma, se requieren tantas palabras de 4 bytes como puntos calculados.

3.2. Envío de datos al microcontrolador

Esta información debe ser enviada al microcontrolador mediante el puerto serie. Como los puntos de la pasada ocupan 4 bytes, la transmisión se realiza en su totalidad en bloques de ese tamaño.

En primer lugar, es necesario sincronizar la cuenta del Reloj de Tiempo Real (*RTC*, por sus siglas en inglés) del microcontrolador, por lo cual el primer valor enviado es el horario en ese momento en el formato UNIX. Este valor, generado por *Python*, tiene un error de aproximadamente 1 *ms*, despreciable en esta aplicación.

Como la cuenta en segundos de un instante en el formato UNIX ocupa 4 Bytes, no es posible enviar información sobre los milisegundos en un mismo bloque. Para evitar tener que enviar dos datos, el programa espera al momento donde los milisegundos son nulos para realizar la transmisión. La comunicación se realiza a 115200 bits por segundo, por lo que el tiempo de transmisión es despreciable.

Luego, se transmiten los datos individuales (ver Tabla 3.3) junto con la cantidad total de puntos calculados, que será utilizada por el microcontrolador para reconocer cuándo cortar la comunicación. Por último, se transmiten todos los puntos de la órbita.

Como la memoria RAM del microcontrolador es limitada y más chica que la cantidad total de información a recibir, es necesario pausar la trasmisión de datos para que el microcontrolador los guarde en una memoria ROM externa, permitiendo vaciar la RAM. Se optó por recibir 1000 puntos, es decir 4 kB, antes de almacenarlos en la memoria externa. Se eligió este valor con el fin de situarse significativamente por debajo del tamaño de la RAM del microcontrolador, de 20 kB.

La Figura 3.2 muestra el diagrama de la comunicación entre el programa de *Python* y el microcontrolador. La comunicación es interrumpida varias veces para esperar la confirmación del microcontrolador. Esta necesidad será explicada en la Sección 4.3.1.

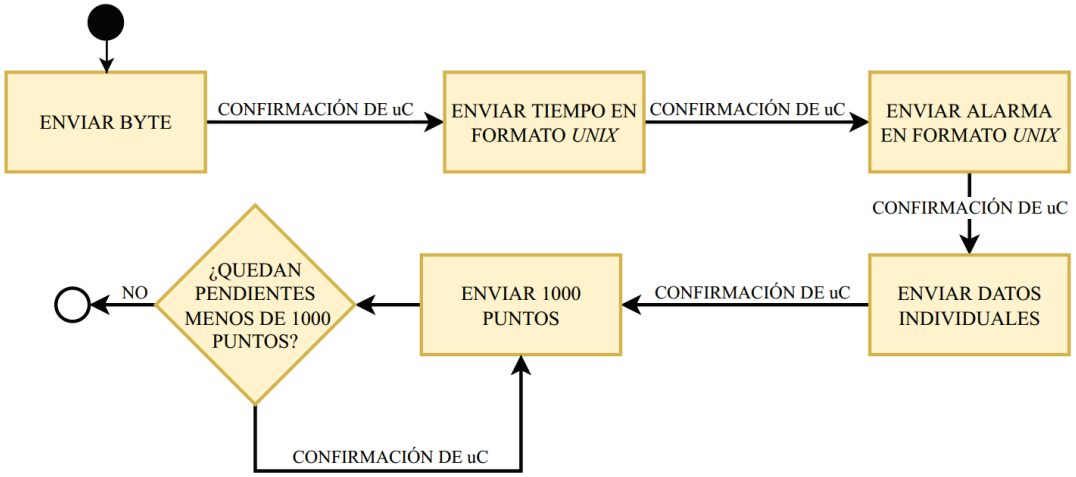


FIGURA 3.2: Diagrama que muestra la comunicación entre *Python* y el microcontrolador.

3.3. Interfaz de usuario

Por otro lado, se desarrolló una interfaz de usuario mediante consola. La misma permite configurar el sistema (latitud, longitud, Δt y resoluciones mecánicas de

ambas coordenadas) y calcular la órbita del satélite seleccionado. Además, permite obtener la órbita de algún satélite que presente una órbita visible en la brevedad, con el fin de comprobar el correcto funcionamiento del aparato. Para utilizarla, es necesario que la computadora tenga una consola de *Python* instalada. La Figura 3.3 muestra la interfaz desarrollada.

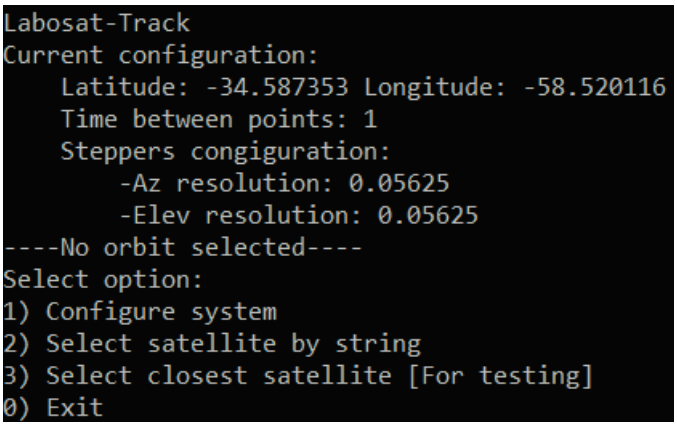


FIGURA 3.3: Captura de la interfaz de usuario en consola desarrollada para operar el sistema.

Capítulo 4

Implementación del sistema

Como prueba de concepto inicial, se había desarrollado un programa en *Python* que calculaba en tiempo real el instante en el que debía realizarse un movimiento mecánico y actuaba dos motores mediante un Arduino. Esta prueba fue exitosa, pero no cumplía con el requerimiento de que el sistema funcione sin estar conectado a otro dispositivo. Para esto, fue necesario desarrollar el sistema de modo que estime la órbita con anterioridad y la almacene en una memoria para su posterior lectura.

Con respecto a la orientación del ángulo azimutal, es necesario que el mismo apunte al norte antes de comenzar la órbita. Inicialmente se implementó un magnetómetro para lograrlo. Sin embargo, su incertidumbre era demasiado elevada y los dispositivos más certeros, demasiado costosos para la aplicación. Por ende, el usuario deberá orientar el sistema manualmente.

El diagrama en bloques del sistema desarrollado se muestra en la Figura 4.1. El mismo será explicado en detalle en la Sección 4.1.

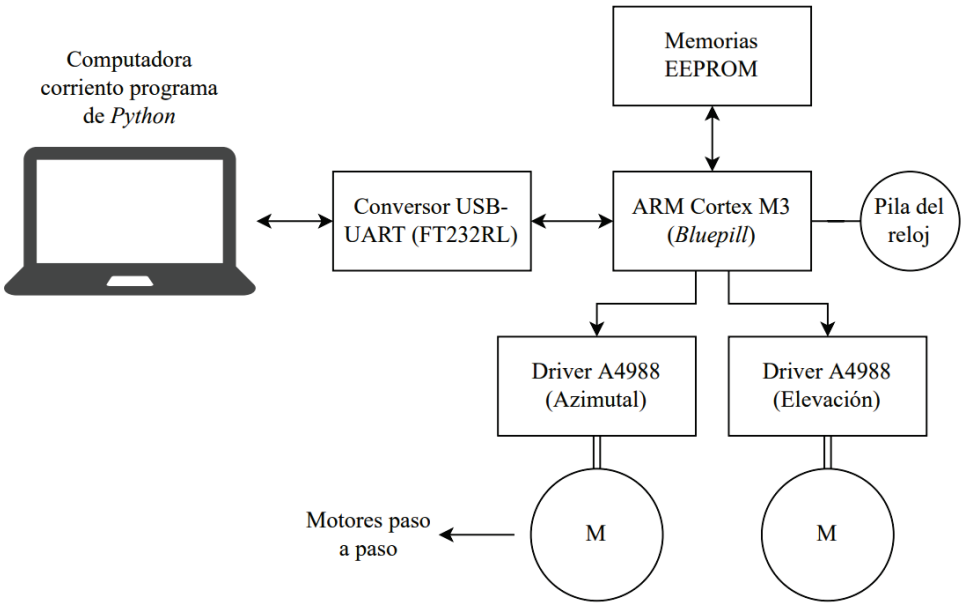


FIGURA 4.1: Diagrama en bloques del sistema desarrollado.

4.1. Hardware

Se optó por utilizar dos motores paso a paso de 0,9° por paso, uno para cada coordenada horizontal. Otra opción viable son servomotores, sin embargo, los motores paso a paso permiten realizar micropasos, lo que brinda una precisión superior.

Además, la elección de esta configuración fue basada en proyectos similares ([17], [18]).

Para controlar los motores, se utilizaron dos *drivers* A4988, que permiten realizar hasta 16 micropasos.[1] Un micropaso consiste en disminuir el ángulo de giro por paso, a costa de una disminución de torque. Así, aplicando 16 micropasos, los motores logran realizar pasos de 0,056°. La Tabla 4.1 muestra la relación entre la reducción por micropasos y la pérdida de torque estático [19].

Micropasos	**% Torque estático**
1	100 %
2	70,7 %
4	38,3 %
8	19,5 %
16	9,8 %
32	4,9 %
64	2,5 %
128	1,2 %
256	0,6 %

TABLA 4.1: Relación entre micropasos y torque [19].

Con respecto al microcontrolador, se optó por utilizar la placa de desarrollo *Blue pill*, debido a su bajo costo y a que cuenta con un RTC interno, con función de *Back-up*, la cual permite el funcionamiento ininterrumpido del módulo, incluso con el sistema desenergizado. Para este fin, es necesario conectar una pila de reloj a uno de los pines. Esta placa contiene un microcontrolador ARM *Cortex*-M3 (STM32F103C8T6), que cuenta con 20 kB de RAM y 80 kB de memoria *Flash*.

Una desventaja del microcontrolador elegido es que su memoria *Flash* no es suficiente para almacenar todos los datos de una órbita, por lo que resulta necesario utilizar memorias externas. Se implementaron las EEPROM AT24C256, de 32 kB, las cuales permiten colocar hasta cuatro en paralelo para obtener un total de 1 Mbit. La necesidad y elección de las memorias será ampliada más adelante.

La placa *Blue pill* no cuenta con conversor serie a USB, por lo que se requiere uno externo. En particular, se utilizó el FT232RL.

Con respecto a la energización del sistema, la misma se logra a través de una batería. Para utilizar los motores paso a paso, se pueden aplicar entre 8 y 35 V. La tensión lógica del microcontrolador es de 3,3 V, la cual se genera mediante un regulador LM1117, el cual admite tensiones de entrada de hasta 20 V. De esta forma, se puede utilizar una batería de entre 8 y 20 V. En particular, se optó por una de 12 V.

El esquemático del circuito se muestra en la Figura 4.2. A la izquierda se observan las cuatro memorias EEPROM, junto con las resistencias de *Pull-up* requeridas para la comunicación I^2C. En el centro se encuentra el microcontrolador y en la parte inferior, los *drivers* de los motores. Cada uno de ellos cuenta con tres *jumpers*, cuya función es configurar los micropasos. En la parte superior se muestra la bornera que recibirá 12 V del exterior, junto con el regulador de tensión a 3,3 V. A la derecha aparece el conversor USB a puerto serie, la pila de reloj y un *switch* que será utilizado para obtener la referencia del ángulo de elevación.

Se desarrolló un circuito impreso para albergar todos los componentes mediante el programa KiCAD, el cual se muestra en la Figura 4.3. El mismo fue pensado para una placa de cobre de 10x10 centímetros.

[1] Otros *drivers* más costosos, permiten configurar hasta 256 micropasos.

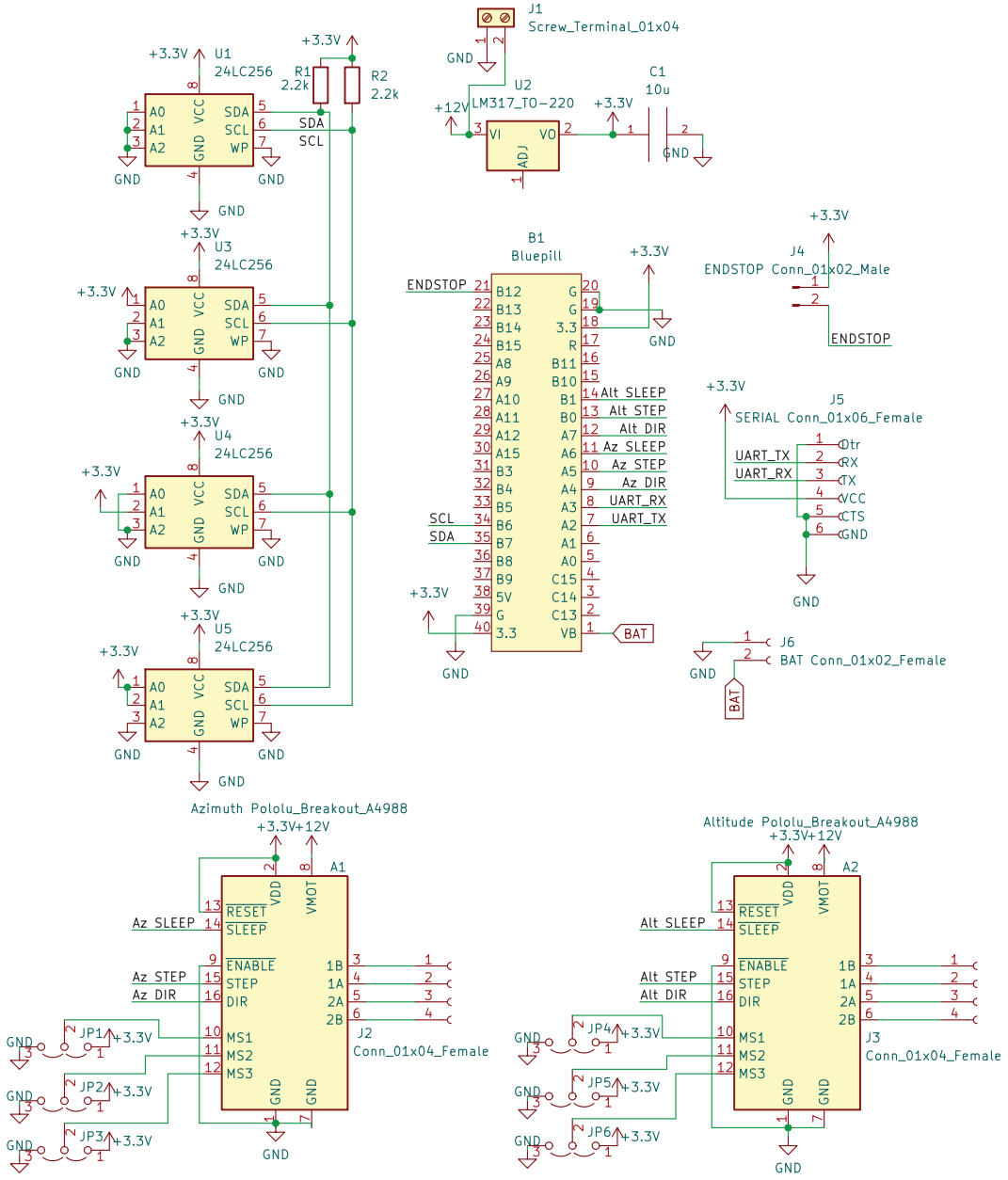


FIGURA 4.2: Esquemático del sistema.

En la figura 4.4 se muestra el prototipo desarrollado. Se observa el circuito impreso con sus componentes junto con los dos motores paso a paso.

4.2. Consumo del sistema

Dado que el sistema funciona a batería, fue necesario caracterizar su consumo. Los motores paso a paso pueden consumir hasta 1,6 A, dependiendo de la limitación impuesta por el *driver* que lo controla. Siendo que una pasada puede llegar a durar hasta 15 minutos y que se cuenta con 2 motores, se obtiene un consumo de:

$$1,6\ \text{A} \cdot 2 \cdot \frac{1\ \textit{hora}}{15\ \textit{minutos}} = 12800\ \text{mAh} \tag{4.1}$$

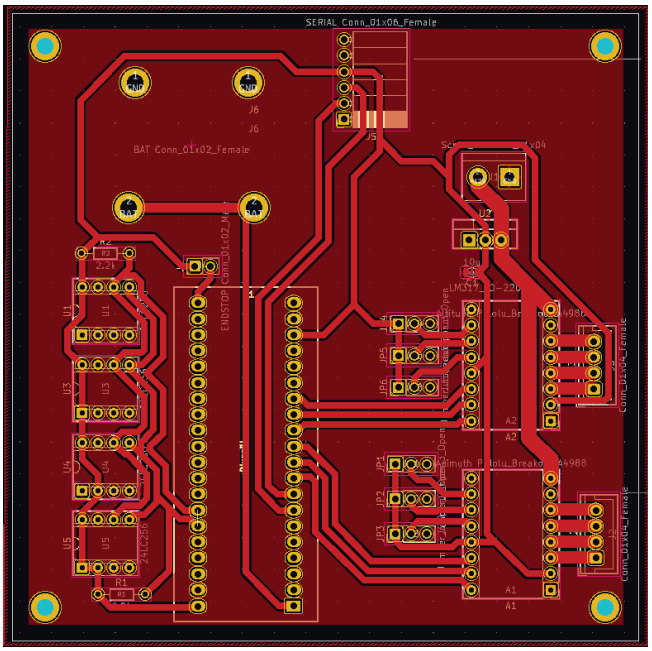


FIGURA 4.3: Circuito impreso desarrollado.

Esto debe tenerse en cuenta a la hora de seleccionar una batería, requiriendo de al menos 3200 *mAh* para una única pasada.

4.3. Firmware

El *firmware* fue implementado en su totalidad utilizando la capa de abstracción de *hardware* (*HAL*, por sus siglas en inglés) provista por el fabricante, *ST Microelectronics*. El diagrama de alto nivel se representa en la Figura 4.5.

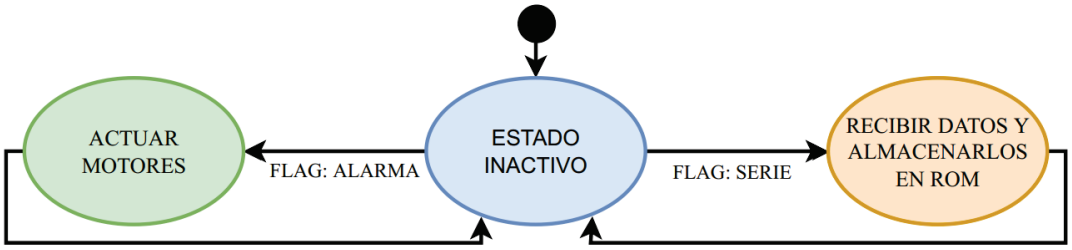


FIGURA 4.5: Diagrama de estado del microcontrolador.

Se observan dos rutinas principales: una de recepción y almacenamiento de datos, y otra encargada de controlar los motores. Ambas serán descriptas a continuación.

4.3.1. Recepción serie y almacenamiento en memorias EEPROM

Al recibir un *byte* por el puerto serie, se dispara una interrupción que comienza la rutina de recepción. La misma se presenta en la Figura 4.6. La comunicación es realizada de manera bloqueante, es decir, el procesador queda bloqueado hasta recibir la totalidad de los datos. Si bien esto no es recomendable en casi ninguna

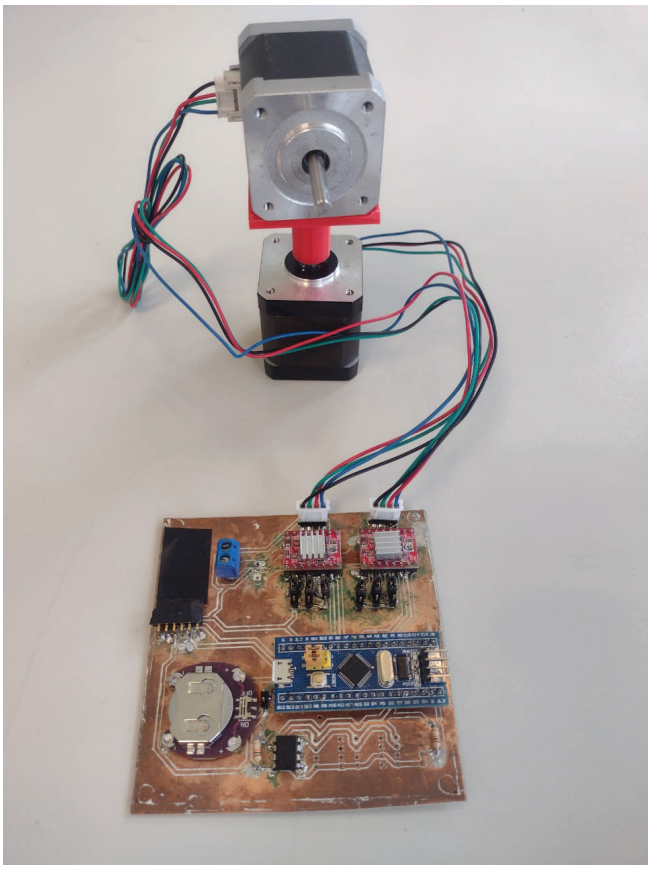


FIGURA 4.4: Prototipo del sistema desarrollado. Se observan el circuito impreso con sus componentes y los motores paso a paso.

aplicación, es este caso no trae inconveniente alguno, ya que la recepción es la única tarea presente. De esta forma se simplifica el diseño.

Como primer dato recibido, se obtiene el valor del tiempo en ese momento, con el cual se actualiza la cuenta del RTC. Además, se escribe sobre los registros de *Back-up* del RTC, de forma que el mismo siga contando incluso con el microcontrolador desenergizado, siempre y cuando el pin correspondiente esté conectado a una pila externa.

Luego se recibe el tiempo donde comienza la órbita, con el cual se configura la alarma del RTC.

Como se requiere orientar los motores antes de comenzar a seguir la órbita, la interrupción de la alarma debe dispararse con anterioridad, por lo que se implementa un *offset* de 60 s, es decir, la alarma se disparará 60 s antes de que comience la órbita.

A continuación, se reciben los datos individuales, para luego recibir los puntos de la órbita. Una vez recibidos 1000 puntos, la comunicación se detiene con el fin de que los mismos sean almacenados en las memorias EEPROM. Una vez guardados, el microcontrolador informa que está disponible y continúa la comunicación. Esto se repite hasta que se reciban todos los datos.

Mientras se desarrollaba esta rutina, se observó que actualizar el RTC o configurar la alarma llevaba un tiempo comparable con el de la comunicación, causando pérdida de datos. Para evitar esto, se estableció la comunicación de forma que el programa de *Python* esperase confirmaciones del microcontrolador para seguir transmitiendo. Las mismas se situaron luego del envío de ambos valores de tiempo, de los datos individuales y de recibir 1000 datos.

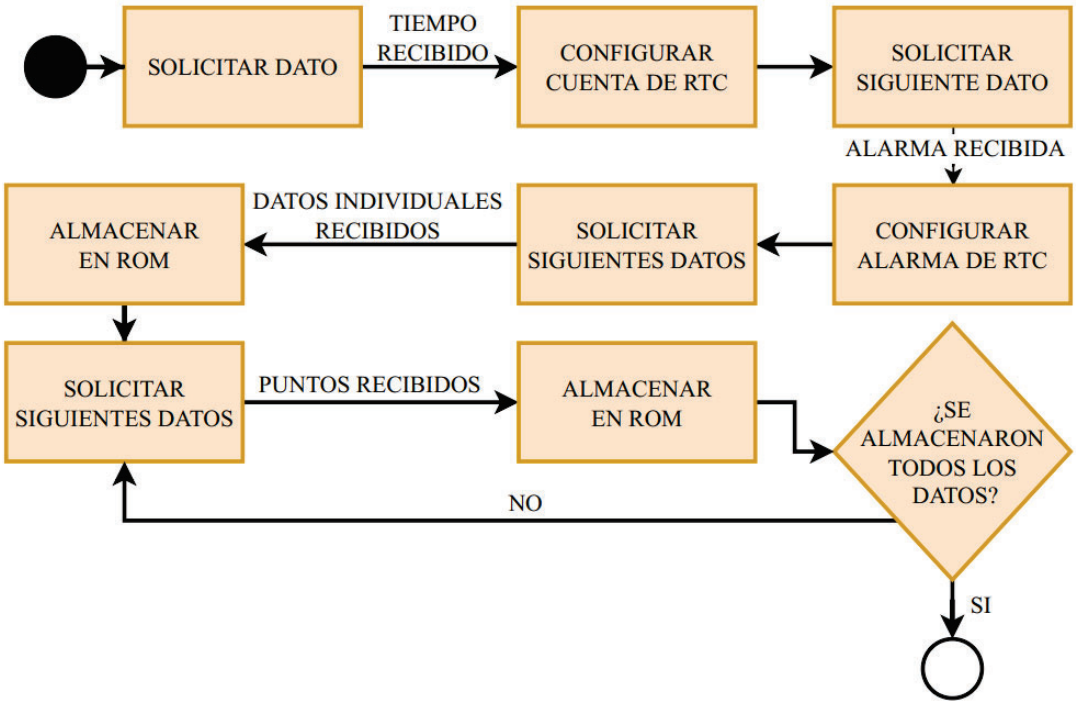


FIGURA 4.6: Diagrama de flujo de la rutina de recepción y almacenamiento de datos.

4.3.2. Actuación de los motores paso a paso

Habiendo almacenado todos los datos, el sistema queda inactivo hasta que la cuenta del RTC alcance el valor de alarma configurado. Una vez ocurra esto, se activa un *timer* que comienza a contar hasta 60 s y se entra en la rutina de actuación de los motores. La misma se muestra en la Figura 4.7.

La rutina comienza desactivando el modo *Sleep* de los drivers, para luego rotar el motor de elevación hasta interactuar con el interruptor colocado de manera que el ángulo sea 0°. El motor azimutal debe ser orientado hacia el norte geográfico por el usuario.

Luego, se leen los datos de la ROM, obteniendo los pasos correspondientes a los ángulos iniciales, a partir de los cuales se orientan ambos motores.

A continuación, se espera hasta que el timer alcance los 60 s, a partir de lo cual se activa otro timer, cuya cuenta está configurada en milisegundos. Este timer debe ser capaz de llevar la cuenta del tiempo durante más de 15 minutos, por lo que se requiere de un registro de 32 bits. Si bien el microcontrolador no cuenta con *timers* de 32 bits, permite utilizar dos de 16 bits en cascada, lo cual fue implementado.

Se compara la cuenta del *timer* con el valor de tiempo de cada punto y, una vez que lo supere, se realizan los pasos correspondientes. Esto se repite hasta que termine la órbita, para luego volver al estado inactivo.

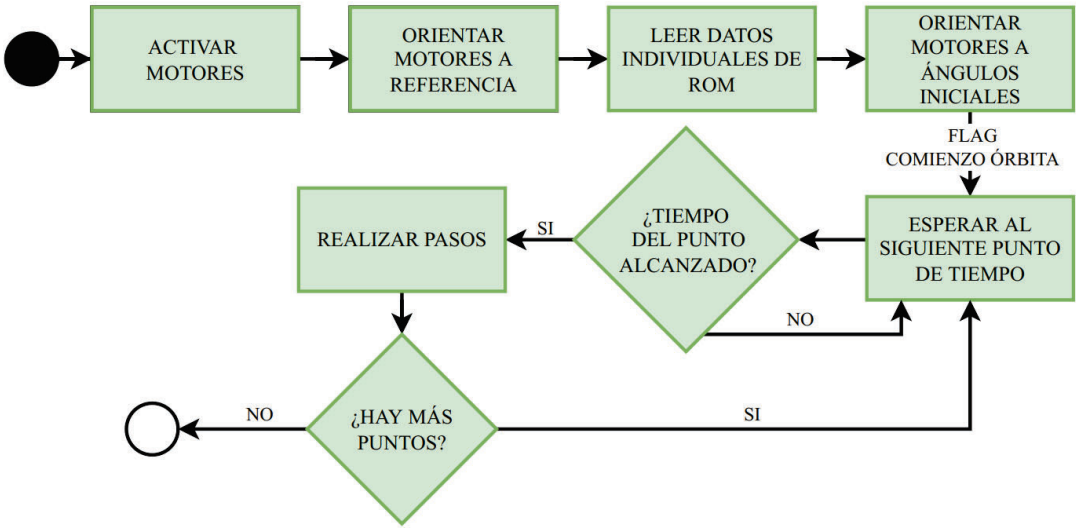


FIGURA 4.7: Diagrama de flujo de la rutina encargada de actuar los motores para realizar el seguimiento del satélite.

El *Firmware* está disponible en [20].

Capítulo 5

Conclusiones

Se investigó y comprobó experimentalmente la incertidumbre intrínseca del algoritmo SGP4 y su implicancia sobre el diseño, concluyendo que es conveniente adoptar una montura ecuatorial, debido al gran error obtenido en altas elevaciones con la montura azimutal. Por otro lado, se observó que el error intrínseco disminuye considerablemente con el aumento de la altitud de la órbita, lo que impone otro parámetro de consideración a la hora de decidir la altura a la que estará orbitando el SiPM.

El proyecto fue diseñado e implementado con éxito, pudiendo verificar visualmente su correcto funcionamiento. No se pudo caracterizar su precisión, ya que el fotomultiplicador no se encontraba en órbita para cuando se finalizó el diseño.

5.1. Posibles mejoras

Durante el desarrollo del proyecto se descubrieron varias mejoras a implementar:

- Implementar el algoritmo directamente en el microcontrolador. De esta forma, no sería necesario el programa de *Python* ni las memorias externas. Así, el único parámetro de entrada necesario sería un TLE, ya que la posición geográfica del observador podría ser obtenida con un módulo GPS. El autor ha intentado esta implementación sin éxito.

  Otra opción viable es realizar tanto la obtención de la pasada como el control de los motores mediante *Python*. Para esto sería necesario contar con un microcontrolador que pueda ejecutar dicho lenguaje de programación, como una *Raspberry Pi*. Esta opción fue descartada por el aumento de costos, como también por la disminución de conocimientos pertinentes adquiridos por parte del autor que esto conlleva.

- Implementar una montura Ecuatorial. La montura azimutal pierde precisión en altas elevaciones, mientras que la ecuatorial hace lo propio en el horizonte, donde no se pretende establecer un enlace óptico.

- Orientar la componente azimutal del sistema automáticamente. Para esto se podría utilizar un magnetómetro, de forma de obtener el norte magnético, a partir del cual se podría conseguir el geográfico, conociendo la declinación magnética. El autor intentó implementar esto, descubriendo que la incertidumbre del sensor es demasiado grande y que los magnetómetros de alta precisión tienen costos injustificables.

- Variar el Δt según la variación de $\Delta\theta$. Si bien tanto en el comienzo y en el punto más alto de la pasada el $\Delta\theta$ sufre su máxima variación, en los puntos medios

se podría aumentar el intervalo de tiempo entre puntos sin afectar la precisión, disminuyendo el tiempo de cómputo.

- Hacer uso de una referencia externa del RTC. El módulo RTC de la *Bluepill* permite agregar un oscilador externo que controle la cuenta del reloj. Esto mejoraría la exactitud del mismo. Si bien la precisión actual es suficiente para la aplicación, podría no serlo si la pasada se realiza varios días después de la época del TLE, llegando a tener un desfase de algunos milisegundos. Si bien este escenario carece de sentido a órbitas bajas por su considerable error, podría presentarse para órbitas más altas.
- Reducir el tamaño del PCB. El PCB fue diseñado para que quepa en un área de $10x10\,cm^2$. Sin embargo, al haber espacio de sobra, el mismo podría rediseñarse en un formato más pequeño.
- Realizar la transmisión de datos vía *Wi-fi*. De este modo, se evita la conexión física del dispositivo con una computadora.
- Considerar la obsolescencia del sistema UNIX. El 19 de enero del 2038, el formato que representa el tiempo en 32 bits sufrirá *Overflow*, por lo que el cálculo de la órbita no podrá realizarse. Para evitar esto, el procedimiento de obtención de la órbita debería ser rediseñado, ya que la librería *Skyfield* utiliza este sistema.

Apéndice A

Descripción de los campos del TLE

En este anexo se describen todos los campos presentes en un TLE ([9]). La figura A.1 muestra un TLE de la estación espacial internacional, a partir del cual se describirán sus campos.

1 25544U 98067A 22234.80516302 .00007508 00000+0 13799-3 0 9999
2 25544 51.6443 6.5497 0005169 147.0432 333.3572 15.50290805355518

FIGURA A.1: Ejemplo de un TLE de la estación espacial internacional.

La primera línea del TLE posee los siguientes parámetros:

- Número de la línea (1).
- Número de catálogo del satélite y clasificación (25544U): la clasificación puede ser U (no clasificado), C (clasificado) o S (secreto).
- Designador internacional (98067A): está formado por los últimos dos dígitos del año de lanzamiento (98), el número de lanzamiento del año (67) y la parte del lanzamiento (A).
- Época (22234.80516302): instante de tiempo para el cual son válidos los otros parámetros del TLE. Los primeros dos dígitos corresponden a los últimos dos dígitos del año (22), mientras que el resto corresponde al día del año y a la fracción del mismo (234.80516302).
- Primera derivada del movimiento promedio o coeficiente balístico (.00007508): consiste en la variación de las revoluciones completadas por día dividido por 2.
- Segunda derivada del movimiento promedio (00000+0): consiste en la derivada del coeficiente balístico, dividido por 6. Los últimos dos dígitos corresponden a una potencia de 10.
- Coeficiente de arrastre (13799-3): parámetro que define el arrastre atmosférico. Los últimos dos dígitos corresponden a una potencia de 10.
- Tipo de efemérides (0): siempre es nulo.
- Número de TLE y control de suma (9999): los primeros tres dígitos corresponden al número de TLE, que se incrementa cada vez que se genera uno nuevo, hasta reiniciarse al superar 999. El último dígito actúa como control de suma de toda la línea.

Y la segunda línea, cuenta con los siguientes parámetros:

- Número de línea (2).
- Número del satélite (25544).
- Inclinación (51.6443): ángulo en grados entre el ecuador y el plano de la órbita.
- Ascensión recta del nodo ascendente (6.5497): ángulo en grados entre el equinoccio de primavera y el punto donde la órbita cruza el ecuador.
- Excentricidad (0005169): constante que describe la forma de la órbita, siendo 0 una órbita circular.
- Argumento del perigeo (147.0432): ángulo en grados entre el nodo ascendente y el perigeo (punto de menor altitud de la órbita)
- Anomalía promedio (333.3572): fracción del período de la órbita desde que el satélite pasó por el periastro[1].
- Movimiento promedio (15.50290805): promedio de órbitas completadas por día.
- Número de revolución y control de suma (355518): número de la órbita en la época (35551) y control de suma de la línea 2 (8).

[1]Punto en una órbita elíptica donde la distancia entre los dos cuerpos es mínima.

Bibliografía

[1] M. Barella et al. «LabOSat: Low cost measurement platform designed for hazardous environments». En: *2016 Seventh Argentine Conference on Embedded Systems (CASE)*. 2016, págs. 1-6. DOI: 10.1109/SASE-CASE.2016.7968107.

[2] GA Sanca et al. *LabOSat: nine missions in a decade*.

[3] Stefan Seifert et al. «First characterization of a digital SiPM based time-of-flight PET detector with 1 mm spatial resolution». En: *Physics in Medicine & Biology* 58.9 (2013), pág. 3061.

[4] Salvatore Gnecchi y Carl Jackson. «A 1× 16 SiPM array for automotive 3D imaging LiDAR systems». En: *Proceedings of the 2017 International Image Sensor Workshop (IISW), Hiroshima, Japan*. 2017, págs. 133-136.

[5] M Baszczyk et al. «Chemiluminescence detection method using sipm with dedicated readout circuit». En: *2017 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*. IEEE. 2017, págs. 1-3.

[6] Mehrab N Modi et al. «Two-photon imaging with silicon photomultipliers». En: *Optics express* 27.24 (2019), págs. 35830-35841.

[7] Samet Aydın et al. «Tracking of low earth orbit satellites by optical systems». En: *2015 7th International Conference on Recent Advances in Space Technologies (RAST)*. IEEE. 2015, págs. 175-180.

[8] David Vallado y Paul Crawford. «SGP4 orbit determination». En: *AIAA - AAS Astrodynamics Specialist Conference and Exhibit* (2008), pág. 6770.

[9] David A. Vallado. *Fundamentals of astrodynamics and applications*. Vol. 12. Springer Science & Business Media, 2001.

[10] David Vallado et al. «Revisiting spacetrack report# 3». En: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit* (2006), pág. 6753.

[11] Wei Dong y Zhao Chang-yin. «An accuracy analysis of the SGP4/SDP4 model». En: *Chinese Astronomy and Astrophysics* 34.1 (2010), págs. 69-76.

[12] R Wang, J Liu y QM Zhang. «Propagation errors analysis of TLE data». En: *Advances in Space Research* 43.7 (2009), págs. 1065-1069.

[13] Erin Kahr, Oliver Montenbruck y Kyle PG O'Keefe. «Estimation and analysis of two-line elements for small satellites». En: *Journal of Spacecraft and Rockets* 50.2 (2013), págs. 433-439.

[14] Thomas Sean Kelso. *Celestrack*. [En línea, accedido 8 de julio de 2023]. URL: \mbox{https://celestrak.org/NORAD/elements}.

[15] *Space-track*. [En línea, accedido 8 de julio de 2023]. URL: \mbox{https://www.space-track.org}.

[16] *Código de python para calcular la órbita*. [En línea, accedido 8 de julio de 2023]. URL: https://github.com/alanmelloni97/Labosat-Track_Python.

[17] Cameron Aume et al. «TrackInk: An IoT-Enabled Real-Time Object Tracking System in Space». En: *Sensors* 22.2 (2022), pág. 608.

[18] Agfianto Eko Putra, Bakhtiar Alldino Ardi Sumbada y Anas Nurbaqin. «Satellite tracking control system for UGM ground station based on TLE calculation». En: *2016 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)* (2016), págs. 91-96.

[19] *How to Improve Motion Smoothness and Accuracy of Stepper Motors*. Texas Instruments. 2020. URL: https://www.ti.com/lit/an/sloa293a/sloa293a.pdf.

[20] *Firmware del sistema*. [En línea, accedido 8 de julio de 2023]. URL: https://github.com/alanmelloni97/LabosatTrack-Firmware.