UNIVERSIDAD LATINA DE PANAMÁ

FACULTAD DE INGENIERÍA

PROPUESTA

"DESARROLLAR UNA MONTURA ECUATORIAL ROBÓTICA PARA EL SEGUIMIENTO DE UNA COMUNICACIÓN SATELITAL"

Proyecto final de graduación presentado como requisito para optar por el título de Ingeniero Mecatrónico en la Universidad Latina de Panamá

Juan Carlos Castillo James
C.I. 8-1055-463

Profesor asesor: Carlos A. Fernández V.

Panamá, República de Panamá
2026

Dedicatoria

A mi familia que siempre me ha apoyado.

Agradecimiento

En este trabajo agradezco a mi familia que me apoyo financiera y emocionalmente. Al profesor supervisor ingeniero Carlos Fernández por confiar en mí y estar pendiente en el proceso de elaboración del proyecto.

UNIVERSIDAD LATINA DE PANAMÁ

DECLARACIÓN JURADA

Yo _____Juan Carlos Castillo James_____ con cédula de identidad personal número, _____8-1055-463_____ estudiante graduando del programa/carrera de _____Licenciatura en Ingeniería Mecatrónica_____ declaro bajo la gravedad del juramento que el material que aparece en este trabajo de graduación, en la opción:_____Proyecto Final_____ (Tesis, proyecto final, pasantía, otro), es de mi producción intelectual, en razón de lo cual exonero a la Universidad Latina de Panamá de cualquier responsabilidad relacionada con este aspecto.

Como constancia firmo la presente declaración el día ___19_____ del mes de __08___del año__2025__.

Firma del estudiante: _____

Cédula: __8-1055-463__

Índice General

Índice de tablas e ilustraciones

Introducción

El proyecto de investigación está dedicado a la disciplina **mecatrónica**, enfocado al área de automatización industrial. Consiste en la ***elaboración de un prototipo de montura ecuatorial robótica para el seguimiento de una comunicación satelital.*** Este proyecto tiene el propósito de servir como una solución automatizada para aficionados a la radioastronomía y radioafición que participan en concursos empleando métodos manuales para el seguimiento de satélites de telecomunicaciones. El cual está destinado a mejorar la experiencia de los participantes aficionados al optimizar sus procesos.

Este proyecto proporciona documentación detallada para la fabricación de un prototipo de montura ecuatorial robótica automatizada destinada al seguimiento de satélites, beneficiando a los radioaficionados en Panamá y otras regiones. Además, puede servir como un instrumento de estudio para estudiantes de la facultad de Ingeniería. Esta iniciativa busca fomentar el avance de las actividades de radioafición y radioastronomía, promoviendo el desarrollo tecnológico y científico en el país.

1. Capítulo 1.0 El Problema

1.1. Antecedentes del problema de investigación

En Panamá y en el mundo se realizan competencias de fines de semana por radioaficionados donde intentan encontrar y seguir objetos celestes como planetas, estrellas, satélites naturales y/o artificiales. Para estas competencias se utilizan instrumentos manuales como telescopios motorizados controlados por mando.

1.2. Planteamiento del problema

¿Puede un sistema automatizado de seguimiento de satélites en Panamá contribuir a la eficiencia de los radioastrónomos y radioaficionados? La comunidad de aficionados a la radioastronomía y radioafición en Panamá enfrenta el desafío de utilizar métodos manuales para el seguimiento de satélites de comunicaciones. Este método no solo demanda una constante supervisión contra el sistema, sino que también limita la precisión y la ciencia en actividades como la recepción de datos y la participación en competencias. Algunos de los síntomas de este problema comprenden la pérdida de señales que provienen de la dificultad para el seguimiento continuo de satélites que afectan la calidad de la comunicación satelital.

Las causas de esta problemática se deben a la falta de sistemas automatizados que sean accesibles, debido a esto, dependen de métodos manuales.

Los beneficios pueden comprender: La mejora de la calidad de la señal recibida, la capacidad de operación del sistema de manera continua sin intervención humana y la reducción de errores de rastreo.

1.3. Justificación de la investigación

La viabilidad del proyecto es alta, debido a los componentes económicos y al acceso de tecnología para su implementación en un periodo corto de tiempo.

El valor teórico de este proyecto integra el *software Stellarium* que proporciona coordenadas en tiempo real y el control de motores paso a paso mediante el uso de microcontroladores. Al usar materiales de bajo costo y software de código abierto, se establece un modelo accesible para diversas instituciones, promoviendo la innovación y el perfeccionamiento continuo en el diseño de sistemas automatizados.

El contexto geográfico de este problema de investigación está enfocado en un entorno universitario, donde la práctica de radioastronomía y radioafición puede contribuir a la formación técnica de los estudiantes.

La relevancia de la investigación se centra en la manera de cambiar los métodos manuales tradicionales por un proceso automatizado más eficiente, avanzando la tecnología y ciencia en Panamá. Al crear la automatización del seguimiento de satélites se estimula la innovación, la formación de profesionales en tecnología y puede atraer futuras colaboraciones a nivel nacional e internacional en el campo aeroespacial.

1.4. Objetivos

1.4.1. Objetivos Generales

Desarrollar una montura ecuatorial robótica de dos ejes para automatizar y dar seguimiento a satélites que transmiten información.

1.4.2. Objetivos Específicos

I. Diseñar un sistema *PID* (proporcional-integral-derivativo) para integrar el software de seguimiento de satélites.

II. Desarrollar e integrar un microcontrolador con el diseño mecánico de una montura ecuatorial robótica de dos ejes.

III. Implementar una solución que permite la conexión de la antena con la recepción y visualización de datos satelitales en tiempo real.

1.5. Alcance y límites de la investigación

El alcance de investigación se centra en diseñar una montura ecuatorial robótica que logre automatizar el proceso de seguimiento de satélites y cuerpos celestes en general, apoyándose de la plataforma de *software Stellarium* y de un microcontrolador Arduino. La montura será puesta a prueba en condiciones reales para verificar el sistema de localización y seguimiento del sistema.

En la investigación no se cubrirá el diseño de antenas, solo el sistema de seguimiento. La precisión para localizar los satélites depende de los motores que se van a emplear y su calidad.

1.6. Línea de investigación

La línea de investigación de este proyecto corresponda a la automatización industrial. Mediante la integración de tecnologías de radioastronomía con el *hardware* que permite controlar y automatizar un sistema de seguimiento de satélites.

2. Capítulo 2.0 Marco Teórico

2.1. Antecedentes de investigaciones realizadas en el tema

Onstep es un proyecto de un controlador *Go-to* de telescopio computarizado, basado en el control de motores paso a paso NEMA-17 para Arduino. (Hjd1964, s.f.).

Control de Telescopio con el *software Stellarium* utilizando el protocolo LX200 (Epsilon Photo, s.f.).

2.2. Bases teóricas que sustentan la investigación

Protocolo de Control Serial de Telescopio Meade LX200: Permite controlar telescopios MEADE LX200 desde una computadora remota con programas de terceros (Meade Instruments, s.f.) (Company Seven, s.f.).

Comunicación Serial: Método de transferencia y recibimiento de datos entre computadores y otros dispositivos (International Business Machines Corporation [IBM], 2024).

Control de Motor Paso a Paso: Pulso PWM que se envían por medio del microcontrolador Arduino a los motores paso a paso.

2.3. Variables

2.3.1. Definición conceptual de las variables

- Sistema de control PI (Proporcional-Integral): método de control automático utilizado para ajustar una señal en función del error en el sistema, para mantener la variable controlada con precisión.

- Montura ecuatorial robótica de dos ejes: Sistema electromecánico con dos ejes para orientar y seguir objetos celestes, de Ascensión Recta (RA) y Declinación (DEC).
- Sistema de comunicación y visualización de datos: Elementos de hardware y software.

2.3.2. Definición operacional de las variables

- Sistema de control PI (Proporcional-Integral):

El sistema de control PI es empleado para corregir el error angular presente entre la posición real y la posición deseada en los ejes, utilizando los parámetros Kp, Ki y Kd. Su desempeño se evalúa mediante el error angular, el tiempo de respuesta, la estabilidad del sistema y la capacidad de mantener el seguimiento continuo del satélite.

- Montura ecuatorial robótica de dos ejes:

Encargado del movimiento en los ejes de *RA* y *DEC* en ángulos. Su funcionamiento se mide a través del rango de movimiento, la precisión angular, la estabilidad estructural, la correcta ejecución de comandos de posicionamiento y la capacidad de sostener y orientar la antena durante el seguimiento satelital.

- Sistema de comunicación y visualización de datos satelitales:

Se presenta mediante la manifestación de los datos de RA/DEC en la pantalla *LCD*. Su funcionamiento se basa en la visualización y actualización de estos datos en tiempo real.

2.3.3. Mapa de Variables

Título de la Investigación: ***"DESARROLLAR UNA MONTURA ECUATORIAL ROBÓTICA PARA EL SEGUIMIENTO DE UNA COMUNICACIÓN SATELITAL"***

Objetivo General:	Desarrollar una montura ecuatorial robótica de dos ejes para automatizar y dar seguimiento a satélites que transmiten información.		
Objetivos Específicos	Variable(s)	Dimensiones	Indicadores
I. Diseñar un sistema PID (proporcional-integral-derivativo) para integrar el software de seguimiento de satélites.	Sistema de control PID	Parámetros del controlador PID	• Ganancia proporcional (Kp) • Ganancia integral (Ki) • Ganancia derivativa (Kd)
		Precisión del seguimiento	• Error angular en RA (°) • Error angular en DEC (°)
		Estabilidad del sistema	• Oscilaciones durante el seguimiento • Sobreimpulso (%)
		Tiempo de respuesta	• Tiempo de establecimiento (s) • Tiempo de subida (s)
		Integración software–hardware	• Comunicación correcta entre Stellarium y microcontrolador
II. Desarrollar e integrar un microcontrolador con el diseño mecánico de una montura ecuatorial robótica de dos ejes.	Montura ecuatorial robótica de dos ejes	Diseño mecánico	• Rango de movimiento del eje RA (°) • Rango de movimiento del eje DEC (°)
		Sistema de accionamiento	• Tipo de motor (paso a paso) • Resolución angular (°/paso)

		Integración electrónica	• Funcionamiento del microcontrolador (Arduino)
III. Implementar una solución que permite la conexión de la antena con la recepción y visualización de datos satelitales en tiempo real.	Sistema de comunicación y visualización de datos RA/DEC	Visualización de datos	Datos mostrados en tiempo real
		Comunicación de datos	Estabilidad de conexión

2.4. Glosario de términos (solo para términos técnicos)

- ***Stellarium:*** Planetario simulado en software de código abierto disponible en la web, computadoras y dispositivos móviles (https://stellarium.org/es/).
- **Sistema PID (proporcional-integral-derivativo):** Es un sistema de control que reúne las características de control de la acción proporcional, acción integral y la acción derivativa. La acción proporcional sirve para estabilizar la oscilación natural de la variable controlada. La acción integral se implementa para mantener a variable controlada sobre el punto de consigna. La acción derivativa se encarga de anticipar el efecto proporcional mediante la estabilización de la variable controlada cuando sufra cambios durante el proceso (Acedo Sánchez, 2003).

- **Ascensión recta (RA):** Es el ángulo que se mide desde el punto de Aries hasta el ecuador celeste en dirección Este. Es la forma equivalente a la longitud a la forma de coordenadas celestes (“Ascensión recta”, 2025).
- **Declinación (DEC):** Es el ángulo que existe entre un astro y el ecuador celeste. Es equivalente a la latitud para la forma de conseguir coordenadas celestes (“Declinación (astronomía)”, 2025).
- **Arduino:** Es una plataforma de código abierto que incluye microcontroladores y software para la programación (https://www.arduino.cc/).
- **Montura ecuatorial:** Es una montura cuya estructura está diseñada para compensar la rotación de la Tierra en donde se utiliza un eje de rotación paralelo al eje de rotación, que permite apuntar a cualquier cuerpo celeste (“Equatorial mount”, 2025).
- **Motor paso a paso:** “Un motor paso a paso es un dispositivo electromecánico que convierte unos pulsos eléctricos en movimientos mecánicos discretos” (Danilo Muñoz Jaramillo, s.f., párr. 6) (Electrónica Caribe, s.f.).
- **Pantalla LCD:** Pantalla con capas de moléculas de cristal líquido que reciben luz polarizada que funcionan con la aplicación de un campo eléctrico (Orient Display, s.f.) (Electrónica PTY, s.f.).

3. Capítulo 3.0 Marco Metodológico

3.1. Tipo y diseño de la investigación

El proyecto consiste en crear y probar un prototipo funcional de una montura ecuatorial robótica en donde se debe medir múltiples variables. Esta investigación es de tipo

experimental, cuantitativo y aplicado. El proyecto es experimental ya que se van a manipular varias variables independientes como el satélite que se selecciona, la cantidad de pasos del motor para así observar los cambios de las variables dependientes, como el valor de los ángulos de ascensión recta y declinación, y el cambio de la orientación de la antena para el seguimiento del satélite. También es correlacional, ya que relaciona esas variables técnicas entre sí. El objetivo del diseño es diseñar la montura para luego probar el sistema con distintos objetos satelitales para comprobar el seguimiento de satélites con los datos que proporciona el *software Stellarium*.

El proyecto tiene un enfoque cuantitativo porque se utilizan datos cuantitativos como los pasos del motor y los ángulos que corresponde a las coordenadas astronómicas de los satélites.

El proyecto es un tipo de investigación aplicada porque pretende resolver una pregunta a través de la implementación de un diseño y construcción de un prototipo funcional para su función para aplicaciones de radioastronomía y radioafición.

3.2. Población y muestra

Población: Estudiantes y docentes de la Facultad de Ingeniería.

Muestra: Compuesta por 1 estudiante de la carrera de Ing. Mecatrónica.

3.2.1. Cálculo del muestro

La validación del prototipo se basó en pruebas funcionales específicas:

- Exactitud del movimiento en eje *RA* y *DEC*.

- Actualización en tiempo real de coordenadas en la pantalla *LCD*.
- Capacidad de seguimiento de objetos seleccionados en Stellarium.
- Comportamiento de los motores bajo carga mecánica simulada.

La prueba de estas variables se realizó a través de un número limitado de ensayos técnicos controlados, para comprobar la viabilidad del diseño.

3.3. Hipótesis

“Si se implementa una montura ecuatorial robótica con control automático de bajo costo, entonces se mejorará la precisión del seguimiento de objetos satelitales para contribuir a la eficiencia de los radioastrónomos y radioaficionados".

3.4. Descripción del instrumento

1. **Arduino:** Es un microcontrolador que se utiliza para recibir los valores de ascensión recta y declinación que se muestra en la pantalla LCD. También se encarga de controlar los movimientos de los motores paso a paso.
2. **Pantalla LCD 16x2:** Su función sirve para visualizar en tiempo real los cambios de las coordenadas de ascensión recta y declinación del satélite seleccionado.
3. ***Stellarium y Script Python:*** Son los programas de software que sirven para proveer las coordenadas astronómicas en tiempo real hacia el Arduino.
4. **PC:** Para el procesamiento de los datos, coordenadas astronómicas, por ejemplo.
5. **Motores 28BYJ-48 con controlador ULN2003:** Realiza el movimiento de la montura.

Se empleó un sistema de control electrónico compuesto por una placa Arduino UNO, una pantalla LCD 16x2, dos motores paso a paso 28BYJ-48 y un conjunto de poleas impresas en 3D, poleas de aluminio y correas para emular el movimiento de una montura ecuatorial robótica.

3.5. Procedimiento de la Investigación

1. Recolección de información técnica: *Software Stellarium, Onstep*.
2. Diseño CAD en Fusion 360: Realicé la estructura de la montura ecuatorial robótica que consiste de un mecanismo reductor empleando un sistema de correas y poleas, tanto para *RA* como para *DEC*.
3. Selección y simulación de componentes mecánicos:

Se unen poleas de sincronización dentadas *GT2* a los dos motores 28BYJ-48. Las poleas conductoras tienen un espaciado de dientes de 2mm, un acople a eje de 5mm, hecho de aluminio y con el máximo ancho de correa que admite siendo de 6mm. Se utilizaron dos correas dentadas cerradas *GT2* de 6mm de ancho, de 200mm y 300mm de longitud. La correa de 300mm será utilizada en el eje de declinación y la correa de 200mm en el eje de ascensión recta. Las poleas conducidas fueron hechas con impresora 3D. La polea conducida para *DEC* tiene un diámetro de paso de 40mm; con 63 dientes. La segunda polea *(RA)* tiene un diámetro de paso de 44mm; con 69 dientes. Ambas poleas conducidas tienen un acople a eje de 16.5mm, un ancho de 6mm y altura de guía de la polea de 1.5mm. Para diseñar las poleas conducidas en Fusion360 instalé una extensión para crear poleas *GT2*. Para determinar la distancia

entre ejes para los mecanismos de polea y correa en *DEC* y *RA* usé la siguiente fórmula.

a) Cálculo de distancia entre eje

$$L = \frac{\pi}{2}(D_1 + D_2) + 2C + \frac{(D_2 - D_1)^2}{4C}$$

Para eje DEC: D_1 ,siendo el diámetro de paso de la polea *GT2* de 10.2mm. D_2, siendo el diámetro de paso de la polea impresa en 3D de 40mm. L, representa la longitud de la correa de 300mm. C es la distancia entre centros.

Para eje RA: D_1 ,siendo el diámetro de paso de la polea *GT2* de 10.2mm. D_2, siendo el diámetro de paso de la polea impresa en 3D de 44mm. L, representa la longitud de la correa de 200mm. C es la distancia entre centros.

Despejar C:

$$300 = \frac{\pi}{2}(10.2 + 40) + 2C + \frac{(40 - 10.2)^2}{4C}$$

$$300 = 78.86 + 2C + \frac{222.01}{C}$$

$$2C^2 - 221.14C + 222.01 = 0$$

$$C = \frac{-(-221.14) \pm \sqrt{221.14^2 - 4(2)(222.01)}}{2(2)}$$

$$C_1 = \frac{221.14 - 217.09}{4} \approx 1.01mm$$

$$C_2 = \frac{221.14 + 217.09}{4} \approx 109.6mm$$

La distancia entre ejes para DEC es $C_2 = 109.6mm$.

$$200 = \frac{\pi}{2}(10.2 + 43.94) + 2C + \frac{(43.94 - 10.2)^2}{4C}$$

$$200 = 85.0429 + 2C + \frac{284.5969}{C}$$

$$2C^2 - 114.9571C + 284.5969 = 0$$

$$C = \frac{114.9571 \pm \sqrt{114.9571^2 - 4(2)(284.5969)}}{2(2)}$$

$$C_1 \approx 2.59mm$$

$$C_2 \approx 54.89mm$$

La distancia entre ejes para *RA* es $C_2 = 54.89mm$.

La relación de transmisión se define como el cociente de los dientes de las poleas. La relación de transmisión *DEC*:

$$\frac{63}{16} \approx 3.9375{:}\,1$$

La relación de transmisión *RA*:

$$\frac{69}{16} \approx 4.31{:}\,1$$

4. Desarrollo del sistema electrónico (Arduino, controladores):

Para el control de motores paso a paso se empleó controladores de motor ULN2003. Los valores de *RA* y *DEC* se miden en ángulos. Para conseguir que el motor mueva la

estructura al ángulo deseado se necesita programar el driver para que ejecute la cantidad de pasos suficientes. Se multiplica la relación de transmisión por los pasos totales del motor, luego se multiplica ese valor por el cociente del ángulo deseado entre 360°.

Ilustración 1 - código drivers A

```
// ================== CONSTANTES Y VARIABLES ==================
const float pasosMotor = 4096.0;       // pasos por vuelta del 28BYJ-48
const float relacionRA = 69.0 / 20.0;  // relación de reducción eje RA
const float relacionDEC = 63.0 / 16.0; // relación de reducción eje DEC

const float pasosRevRA = pasosMotor * relacionRA;
const float pasosRevDEC = pasosMotor * relacionDEC;
```

```
// ================== FUNCIONES ==================
void moverRA(float nuevaRA) {
  long targetSteps = (long)((nuevaRA / 360.0) * pasosRevRA);
  stepperRA.moveTo(targetSteps);
  currentPosRA = targetSteps;
}
```

5. Desarrollo del circuito:

Revisar anexo 5.

6. Integración con software (Stellarium y Python):

Se utilizó el *plugin* de Control Remoto que está incluido en el software planetario Stellarium para obtener los datos de *RA/DEC* del objeto celeste seleccionado. Esto conecta la PC a un server con Stellarium. Luego con un *script* de *Python* se obtienen las coordenadas astronómicas y se envían al Arduino por un puerto serial *COM*. Existen dos versiones del programa. El primero, utiliza el *plugin* de Control Remoto previamente explicado. El segundo, utiliza el plugin de Control de Telescopio que también viene integrado en el software Stellarium. Se utilizó el protocolo de

comunicación serial de Telescopio Meade LX200 que se emplea para comandar y controlar telescopios Meade de forma remota. Y parte del código se encarga de reflejar los valores RA/DEC del objeto seleccionado en *Stellarium* en la pantalla *LCD*.

7. Control PID del Sistema de Posicionamiento

    1. Objetivo del Control

El objetivo del sistema de control es garantizar que los ejes de Ascensión Recta (*RA)* y Declinación (*DEC*) de la montura ecuatorial alcancen de forma rápida, estable y precisa las coordenadas astronómicas solicitadas por el usuario o por el software de control astronómico.

Dado que el sistema utiliza motores paso a paso sin sensores de retroalimentación física, se implementa un control basado en la posición estimada por conteo de pasos, conocido como control en lazo abierto con realimentación lógica.

    2. Selección del Tipo de Control

Inicialmente se evaluó la implementación de un controlador *PID* completo (Proporcional–Integral–Derivativo). Sin embargo, debido a las características del sistema —motores paso a paso, cargas mecánicas suaves y ausencia de sensores— se determinó que el término derivativo podía introducir ruido e inestabilidad. Por esta razón, se adoptó un control PI (Proporcional–Integral), el cual ofrece un equilibrio adecuado entre: rapidez de respuesta, precisión en la posición final, estabilidad mecánica, simplicidad computacional.

    3. Modelo de Control Implementado

El control PI se implementa de forma digital dentro del microcontrolador Arduino, actuando sobre la velocidad máxima de cada motor mediante la librería AccelStepper.

El error de posición se define como:

$$e(t) = \theta objetivo - \theta actual$$

Donde:

- $\theta objetivo$ es la posición angular deseada (convertida a pasos).
- $\theta actual$ es la posición actual del motor obtenida por conteo de pasos.

El controlador *PI* se expresa como:

$$u(t) = Kp \cdot |e(t)| + Ki \cdot \int |e(t)| dt$$

Donde:

- Kp es la ganancia proporcional.
- Ki es la ganancia integral.
- $u(t)$ corresponde a la velocidad máxima asignada al motor.

El uso del valor absoluto permite que la dirección del movimiento sea gestionada por la función moveTo() de la librería *AccelStepper*, evitando inversiones innecesarias del signo de la velocidad.

4. Integración con *AccelStepper*

La señal de control generada por el *PI* no actúa directamente sobre la posición, sino que modifica dinámicamente la velocidad máxima permitida del motor.

Este enfoque ofrece varias ventajas:

- Movimientos rápidos cuando el error es grande.
- Desaceleración progresiva al acercarse al objetivo.
- Eliminación de oscilaciones cerca del punto final.
- Protección mecánica del sistema.

La aceleración es gestionada internamente por *AccelStepper,* lo que garantiza transiciones suaves y reduce esfuerzos sobre engranajes y ejes.

5. Consideraciones de Estabilidad

Para evitar la acumulación excesiva del término integral (fenómeno conocido como integral windup), se implementó una saturación del valor integral:

$$\int e(t)dt \in [-Imax, Imax]$$

Esto asegura estabilidad incluso ante errores grandes o movimientos prolongados.

Además, se establecieron límites mínimos y máximos de velocidad para asegurar que:

- El motor no se detenga prematuramente.
- No se excedan las capacidades físicas del motor.

6. Ajuste de Parámetros

Los valores de las ganancias se ajustaron experimentalmente mediante pruebas de movimiento real del sistema:

Parámetro	Valor

Kp	0.06
Ki	0.00002

Estos valores permiten un desplazamiento rápido durante el posicionamiento inicial (GOTO) y una aproximación suave al punto objetivo, logrando una precisión adecuada para observación astronómica.

7. Diagrama de Bloques del Control

Coordenadas RA/DEC

↓

Conversión a pasos

↓

Cálculo de error

↓

Control PI

↓

Velocidad dinámica

↓

AccelStepper

↓

Motor paso a paso

↓

Posición estimada

↺ (realimentación por conteo de pasos)

8. Ventajas del Control Propuesto

- No requiere sensores adicionales.
- Fácil implementación en microcontroladores de bajo costo.
- Estable y confiable para sistemas de seguimiento astronómico.
- Compatible con protocolos ASCOM / LX200.
- Adecuado para proyectos académicos y prototipos funcionales.

9. Conclusión del Apartado

El controlador *PI* implementado permite que la montura ecuatorial robótica alcance las posiciones solicitadas con precisión y suavidad, demostrando que es posible lograr un control efectivo de posicionamiento astronómico utilizando motores paso a paso sin sensores, siempre que se realice una adecuada gestión de velocidad y aceleración.

4. Capítulo 4.0 Análisis e interpretación de los resultados

4.1. Análisis e interpretación de los resultados

- Comparación entre coordenadas reales y simuladas y Precisión en el seguimiento del objeto.

El eje DEC parece que se desvía menos de 3 grados. El eje RA parece desviarse unos 10 grados.

- Tiempo de respuesta.

El tiempo de respuesta parece ser de forma inmediata. Al seleccionar un objeto celeste en Stellarium se actualiza la pantalla *LCD* instantáneamente y se empiezan a mover los motores.

- Estabilidad del sistema.

La estructura mecánica parece tener deslizamientos en la polea y correa dentada que afecta a la precisión del seguimiento.

4.2. Prueba de hipótesis

Al proponer un prototipo de montura ecuatorial robótica con seguimiento automático se mejorará la eficiencia para localizar y seguir objetos celestes, en comparación al ajuste manual. Al utilizar pulsos que se envían a los motores se asegura que se mueva la estructura a la posición de ángulos deseado, sin embargo, un método manual puede incluir falta de precisión y más tiempo en mover la estructura.

5. Capítulo 5.0 Propuesta de la investigación

5.1. Introducción de la propuesta

“Se propone el desarrollo de una montura ecuatorial robótica de bajo costo para seguimiento automatizado de satélites, que pueda ser reproducida por estudiantes, radioaficionados y radioastrónomos para mejorar la eficiencia en las prácticas.”

5.2. Justificación de la propuesta

Esta propuesta será útil para los estudiantes universitarios que estén interesados en radioastronomía para implementar un sistema de seguimiento de satélites de bajo costo. También puede ser empleado para radioaficionados en las competencias de fin de semana.

5.3. Objetivos de la propuesta

Objetivo General:

- *Diseñar y construir una montura robótica funcional para seguimiento satelital.*

Objetivos Específicos:

- *Desarrollar el sistema de mecánico para accionar la montura ecuatorial.*
- *Implementar el Arduino para controlar los motores paso a paso para apuntar la montura hacia las coordenadas seleccionadas, comprobando la precisión.*
- *Reflejar las coordenadas astronómicas RA/DEC en la pantalla LCD.*

5.4. Metas a alcanzar

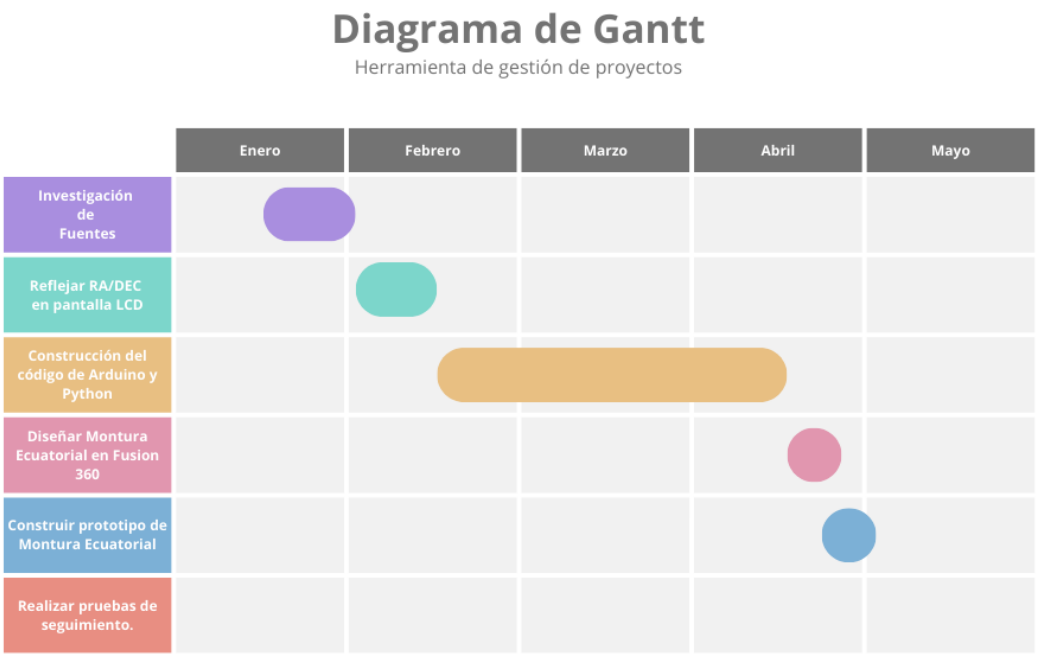
- Diseñar prototipo en *CAD*.
- Ensamblar mecánicamente.
- Ejecutar 3 pruebas exitosas de seguimiento.

5.5. Beneficios de la propuesta

Esta propuesta busca ofrecer una solución para mejorar la eficiencia en el seguimiento de satélites en las competiciones de radioaficionados. También servirá como un instrumento para los estudiantes de la Facultad de Ingeniería (telecomunicaciones y mecatrónica) para poder estudiar e investigar métodos para la comunicación satelital.

5.6. Cronograma de actividades

Tabla 1 - Cronograma



5.7. Presupuesto

Tabla 2 - Presupuesto

Componentes	Cantidad	Precio Unitario (USD)	Precio Total (USD)
Arduino	1	$3.50	$3.50
Pantalla LCD	1	$4.00	$4.00
Motor paso a paso 28BYJ-48 + ULN2003	2	$2.50	$5.00
Fuente de alimentación 5V	1	$3.00	$3.00
Impresión 3D (PLA, prototipos)		$10.00	$10.00
Protoboard y cables	1	$3.00	$3.00
Tornillería, ejes, soportes		$5.00	$5.00
Total			$33.50

5.8. Diseño de la propuesta

Ilustración 2 - Primer diseño con engranajes impresos en 3D.

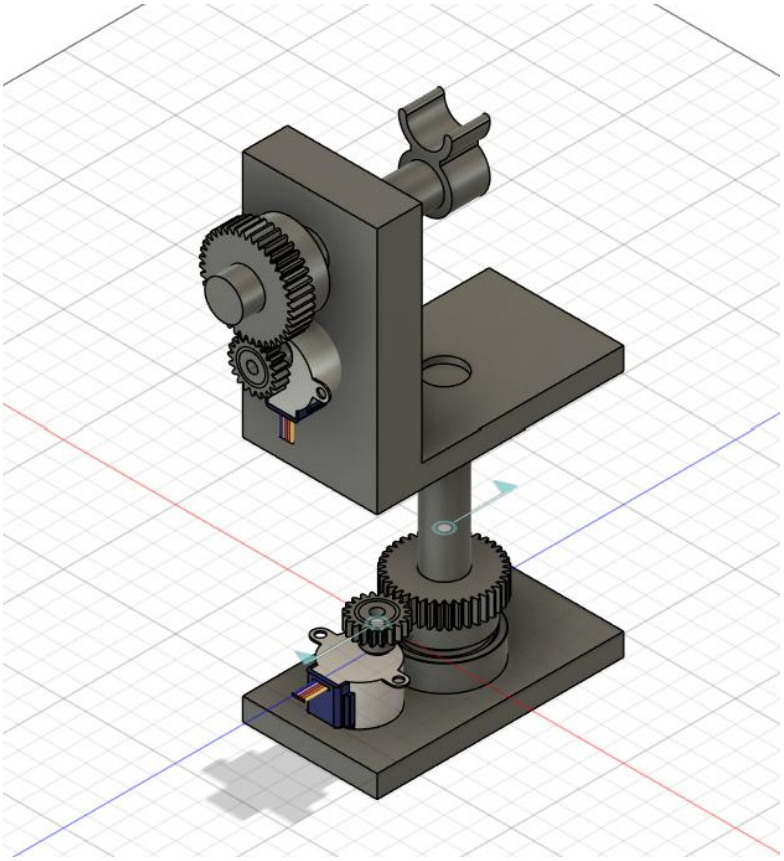
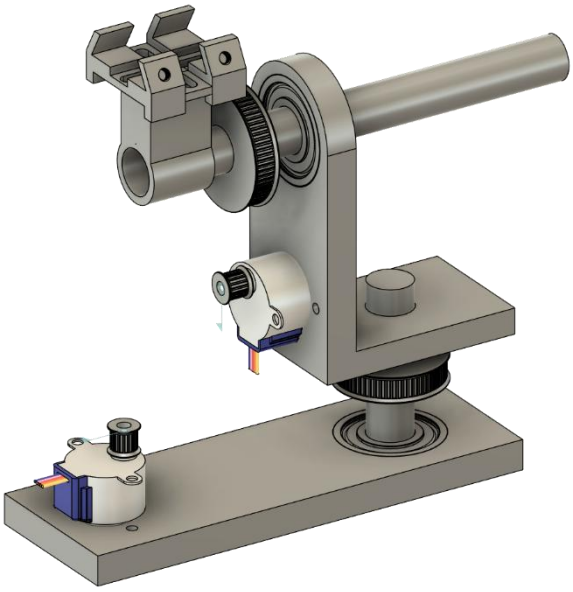
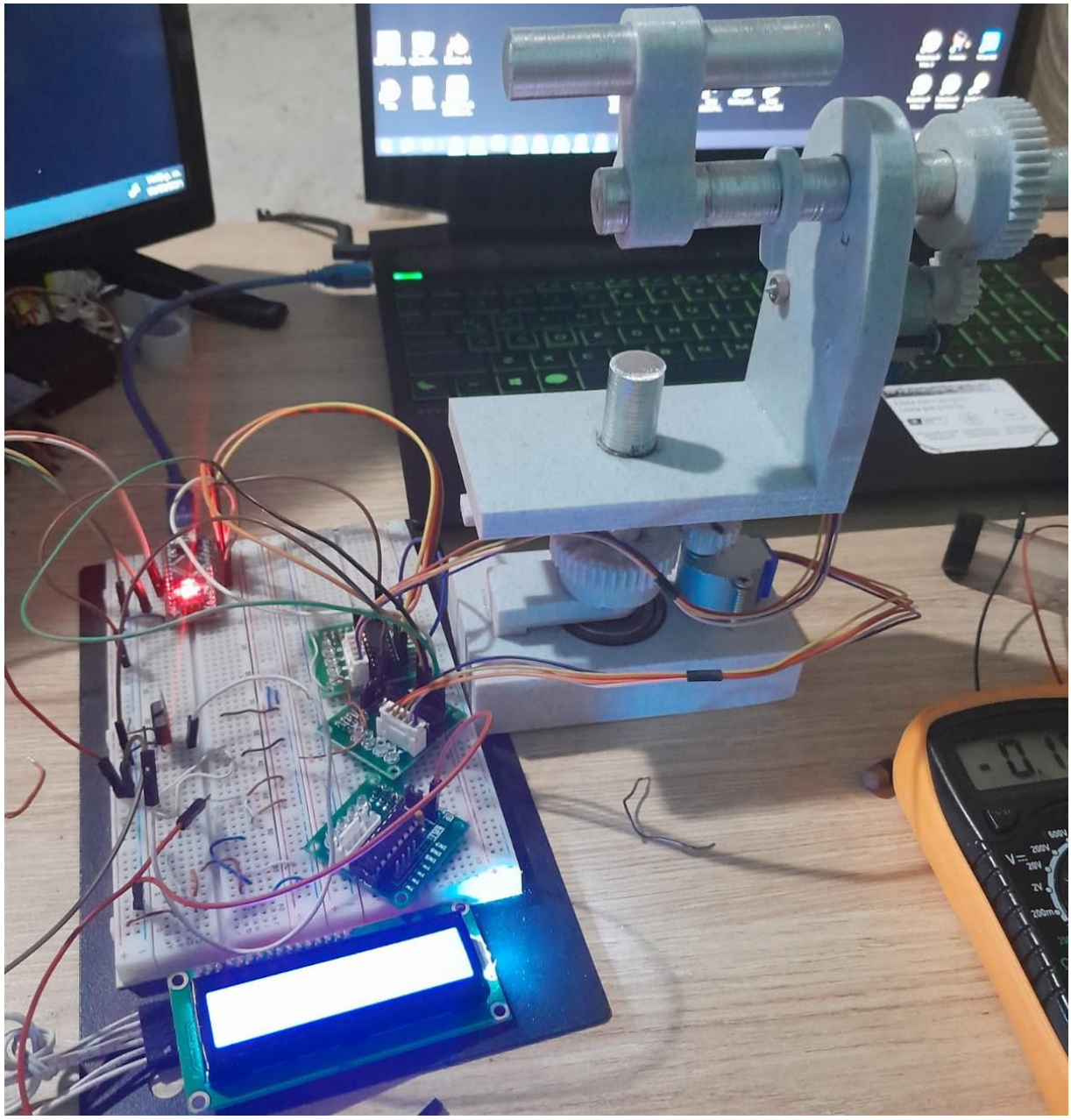


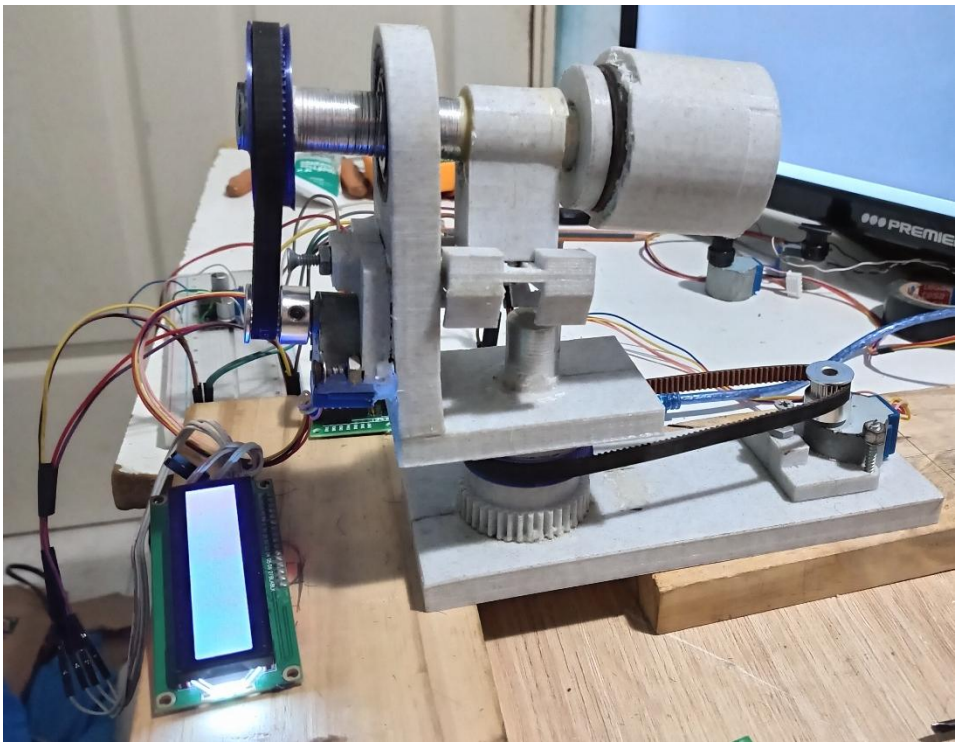
Ilustración 3 - Diseño de la montura ecuatorial robótica realizada en fusión 360.

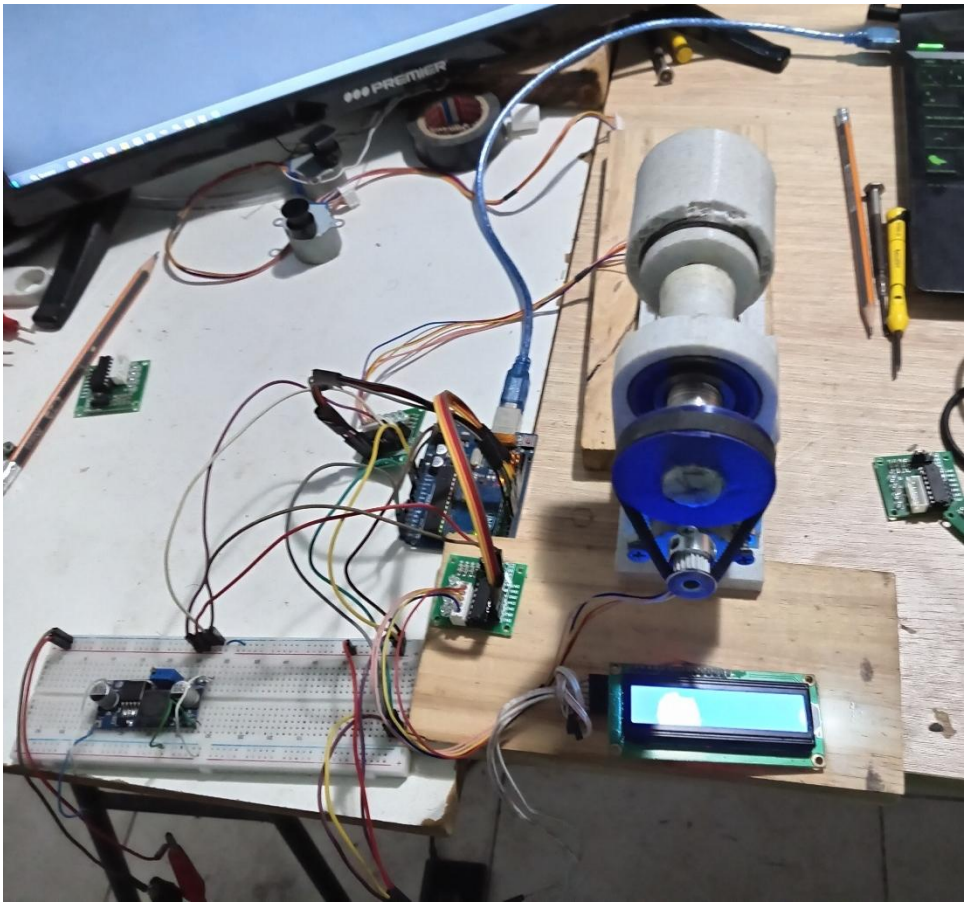


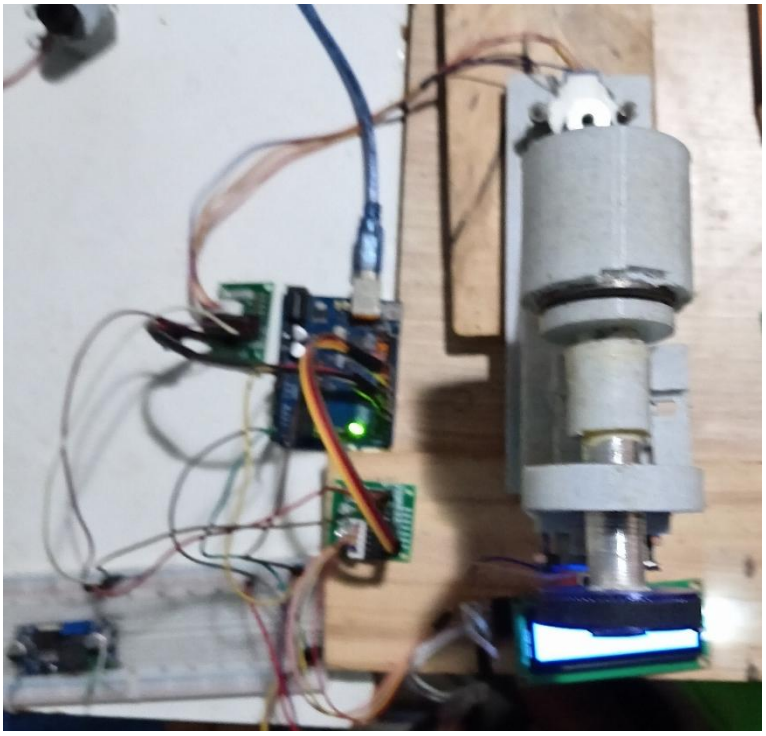
Consultar anexos para ver el archivo de fusión 360 y los modelos STL del diseño de la propuesta.

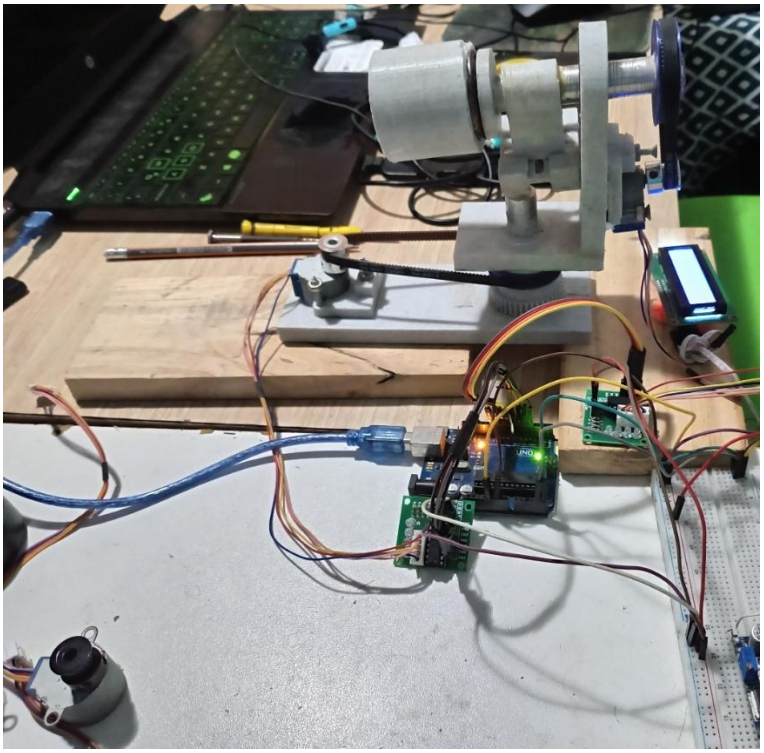
Ilustración 4 - Montura ecuatorial robótica ensamblado






PREMIER





6. Conclusiones

En la elaboración de este prototipo de proyecto de montura ecuatorial robótica se logra ofrecer una solución de seguimiento automática en vez de utilizar la configuración manual para localizar objetos celestes. El prototipo puede mejorar en términos de mecánica, en aspectos como aumentar la carga que puede soportar los motores y evitar deslizamientos.

Esto es una propuesta para que radioaficionados pueden emplear en su estudio y competencias relacionados a la astronomía, asi como los estudiantes de la facultad de ingeniería pueden utilizar esta solución para estudiar métodos de comunicación satelital.

7. Recomendaciones

Para mejorar el diseño de la montura para evitar el deslizamiento en las poleas y correas sería recomendable cambiar la polea grande impresa con impresora 3D con una polea de metal de aluminio.

También se consideró que se podrá cambiar los motores *28BYJ-48* a motores *NEMA17* que son más potentes y pueden soportar mayores cargas.

Añadir sensores a los motores *NEMA-17* para fijar un punto de reseteo. Añadir el control derivativo al sistema PI. No pude agregar el sistema *PID* a la montura, solo un sistema proporcional-integral.

Reforzar los motores con una estructura, tornillos y pegamento.

Montar a un trípode que pueda alinear el prototipo paralelo al ecuador terrestre.

8. Bibliografía

Epsilon Photo. (s.f.). *Configurare Stellarium.* https://epsilonphoto.weebly.com/configurare-stellarium.html

Meade Instruments. (s.f.). *Serie LX200*. https://www.meade.com.mx/index.php/serie-lx200

Company Seven. (s.f.). *Meade LX200 Command Set.* http://www.company7.com/library/meade/LX200CommandSet.pdf

International Business Machines Corporation. (27 de agosto de 2024). *Serial communication*. https://www.ibm.com/docs/es/aix/7.2?topic=communications-serial-communication

Acedo Sánchez, J. (2003). *Control avanzado de procesos: teoría y práctica:* (ed.). Ediciones Díaz de Santos. https://elibro.net/es/ereader/ulatina/97533?page=212

Ascensión recta. (27 de agosto de 2025). En Wikipedia. https://es.wikipedia.org/w/index.php?title=Ascensi%C3%B3n_recta&oldid=169195596

Declinación (astronomía). (27 de agosto de 2025). En Wikipedia. https://es.wikipedia.org/w/index.php?title=Declinaci%C3%B3n_(astronom%C3%ADa)&oldid=169199818

Equatorial mount. (29 de enero de 2025). En Wikipedia. https://en.wikipedia.org/w/index.php?title=Equatorial_mount&oldid=1272680764

Danilo Muñoz Jaramillo. (s.f.). *Motor Paso a Paso 28BYJ-48 con Arduino junto al driver ULN2003 y el L298.* https://programarfacil.com/blog/arduino-blog/motor-paso-a-paso-uln2003-l298n/

Electrónica Caribe. (s.f.). *Módulo para motor paso ULN2003*. https://electronicacaribe.com/product/modulo-para-motor-paso-uln2003/

Orient Display. (s.f.). *¿Qué es una pantalla LCD?* https://www.orientdisplay.com/es/knowledge-base/lcd-basics/what-is-lcd-liquid-crystal-display/

Electrónica PTY. (s.f.). *Interface serial I2C para pantalla LC*D. http://www.electronicapty.com/tienda/pantallas-lcds/interface-serial-i2c-para-pantalla-lcd-detail

Hjd1964. (s.f.). *OnStep* [Repositorio GitHub]. GitHub. https://github.com/hjd1964/OnStep

9. Anexos

9.1. Anexo 1: Código fuente del microcontrolador (Arduino UNO) - *plugin* Control Remoto

```
#include <AccelStepper.h>
#include <LiquidCrystal_I2C.h>

//LCD
LiquidCrystal_I2C lcd(0x27, 16, 2);

//MOTORES
// 28BYJ-48 + ULN2003 (HALF STEP)
AccelStepper stepperRA(AccelStepper::HALF4WIRE, 8, 10, 9, 11);
AccelStepper stepperDEC(AccelStepper::HALF4WIRE, 4, 6, 5, 7);

// MECÁNICA
const float pasosMotor = 4096.0;

const float relacionRA  = 69.0 / 16.0;
const float relacionDEC = 63.0 / 16.0;

const float pasosRevRA  = pasosMotor * relacionRA;
const float pasosRevDEC = pasosMotor * relacionDEC;
const float factorRA  = 1.000;
const float factorDEC = 1.000;




// PI
float Kp = 0.060;
float Ki = 0.00002;

float integralRA = 0;
float integralDEC = 0;

// VARIABLES
float raDecimal = 0.0;
float decDecimal = 0.0;

long targetRAsteps  = 0;
long targetDECsteps = 0;
```

```
bool moveCommand = false;


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

  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Esperando RA/DEC");

  stepperRA.setMaxSpeed(900);
  stepperRA.setAcceleration(900);

  stepperDEC.setMaxSpeed(900);
  stepperDEC.setAcceleration(900);

}


void loop() {

  //  RECEPCIÓN SERIAL DE PYTHON
  if (Serial.available()) {
    String data = Serial.readStringUntil('#');

    if (data.startsWith(":Sr")) {
      raDecimal = data.substring(3).toFloat();
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("RA:");
      lcd.print(raDecimal, 6);
    }

    if (data.startsWith(":Sd")) {
      decDecimal = data.substring(3).toFloat();
      lcd.setCursor(0, 1);
      lcd.print("DEC:");
      lcd.print(decDecimal, 6);
    }

    if (data.startsWith(":MS")) {
      targetRAsteps  = (raDecimal / 360.0) * pasosRevRA  * factorRA;
      targetDECsteps = (decDecimal / 360.0) * pasosRevDEC * factorDEC;
```

```
      stepperRA.moveTo(targetRAsteps);
      stepperDEC.moveTo(targetDECsteps);

      moveCommand = true;
    }
  }

  // CONTROL PI + ACCELSTEPPER
  if (moveCommand) {

    // PI RA
    long errorRA = targetRAsteps - stepperRA.currentPosition();
    integralRA += errorRA;
    integralRA = constrain(integralRA, -800, 800);

    float speedRA = Kp * abs(errorRA) + Ki * abs(integralRA);
    speedRA = constrain(speedRA, 900, 900);
    stepperRA.setMaxSpeed(speedRA);

    // PI DEC
    long errorDEC = targetDECsteps - stepperDEC.currentPosition();
    integralDEC += errorDEC;
    integralDEC = constrain(integralDEC, -800, 800);

    float speedDEC = Kp * abs(errorDEC) + Ki * abs(integralDEC);
    speedDEC = constrain(speedDEC, 250, 900);
    stepperDEC.setMaxSpeed(speedDEC);

    stepperRA.run();
    stepperDEC.run();

    // FIN DE MOVIMIENTO
    if (!stepperRA.isRunning() && !stepperDEC.isRunning()) {
      moveCommand = false;
      integralRA = 0;
      integralDEC = 0;
    }
  }
}
```

9.2. Anexo 2: Código fuente del microcontrolador (Arduino UNO) - *plugin* Control de Telescopio

```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

String inputString = "";
String currentRA = "00:00:00";      // LX200
String currentDEC = "+00*00:00";    // LX200
String targetRA = "00:00:00";
String targetDEC = "+00*00:00";

// ======== CONVERTIR RA HH:MM:SS → DECIMAL =========
float RAtoDecimal(String ra) {
  int hh = ra.substring(0, 2).toInt();
  int mm = ra.substring(3, 5).toInt();
  int ss = ra.substring(6, 8).toInt();
  return hh + mm / 60.0 + ss / 3600.0;
}

// ======== CONVERTIR DEC ±DD*MM:SS → DECIMAL ========
float DECtoDecimal(String dec) {
  int sign = (dec.charAt(0) == '-') ? -1 : 1;
  int dd = dec.substring(1, 3).toInt();
  int mm = dec.substring(4, 6).toInt();
  int ss = dec.substring(7, 9).toInt();
  return sign * (dd + mm / 60.0 + ss / 3600.0);
}

void setup() {
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();

  lcd.setCursor(0, 0);
  lcd.print("RA: 0.0");
  lcd.setCursor(0, 1);
  lcd.print("DEC:0.0");
}

void loop() {
```

```
  if (Serial.available()) {
    char inChar = Serial.read();
    inputString += inChar;

    if (inChar == '#') {
      processCommand(inputString);
      inputString = "";
    }
  }
}

void updateLCDdecimal() {
  float raDec  = RAtoDecimal(currentRA);
  float decDec = DECtoDecimal(currentDEC);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("RA:");
  lcd.print(raDec, 10);     // 6 decimales

  lcd.setCursor(0, 1);
  lcd.print("DEC:");
  lcd.print(decDec, 10);    // 6 decimales
}

// Procesa comandos LX200
void processCommand(String command) {

  // ===== RA actual =====
  if (command == ":GR#") {
    Serial.print(currentRA); Serial.print("#");
    updateLCDdecimal();
  }

  // ===== DEC actual =====
  else if (command == ":GD#") {
    Serial.print(currentDEC); Serial.print("#");
    updateLCDdecimal();
  }

  // ===== T-RA objetivo =====
  else if (command == ":Gr#") {
    Serial.print(targetRA); Serial.print("#");
    updateLCDdecimal();
  }
```

```
  // ===== T-DEC objetivo =====
  else if (command == ":Gd#") {
    Serial.print(targetDEC); Serial.print("#");
    updateLCDdecimal();
  }

  // ===== Set RA objetivo =====
  else if (command.startsWith(":Sr")) {
    targetRA = command.substring(3, command.length() - 1);
    Serial.print("1#");
  }

  // ===== Set DEC objetivo =====
  else if (command.startsWith(":Sd")) {
    targetDEC = command.substring(3, command.length() - 1);
    Serial.print("1#");
  }

  // ===== Simular goto =====
  else if (command == ":MS#") {
    currentRA  = targetRA;
    currentDEC = targetDEC;
    Serial.print("0#");
    updateLCDdecimal();
  }

  else {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Cmd no valido");
  }
}
```

9.3. Anexo 3: *Script* en *Python* para adquisición de coordenadas desde *Stellarium*

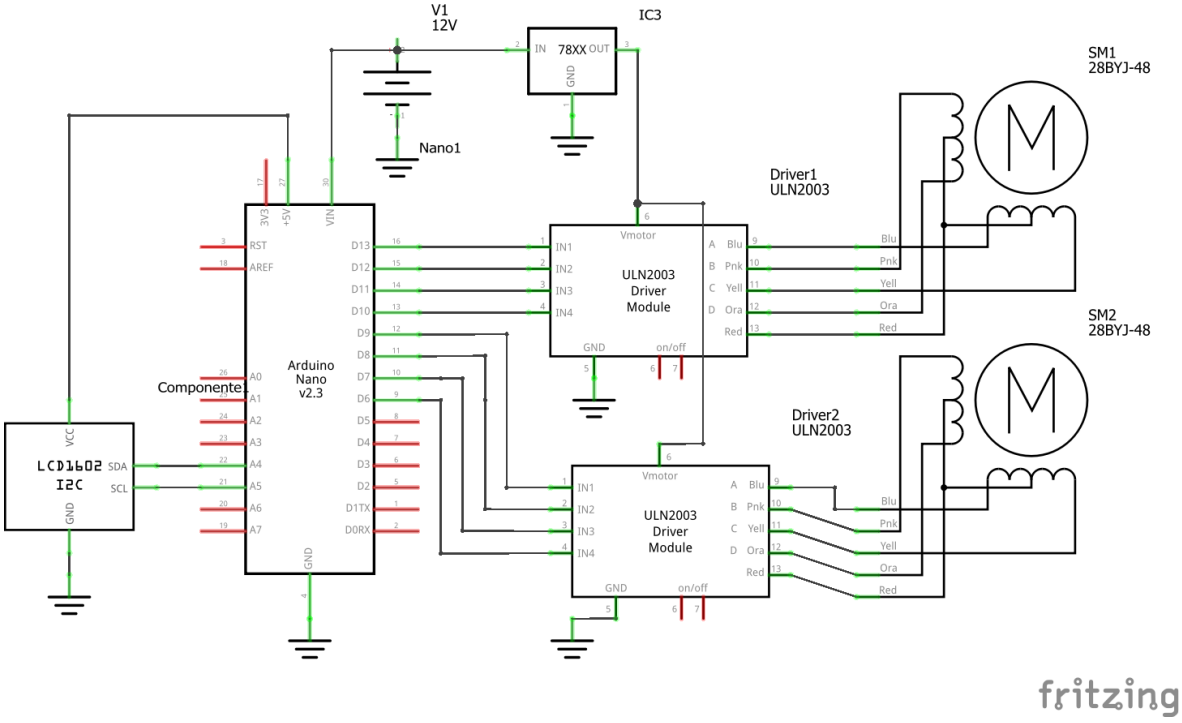
```python
import requests
import serial
import time

# Convertir RA en formato horas-minutos-segundos o decimal a grados decimales
def hms_to_decimal(ra_hms):
    try:
        return float(ra_hms)  # Ya está en grados decimales
    except ValueError:
        pass

    ra_hms = ra_hms.replace("h", ":").replace("m", ":").replace("s", "")
    ra_parts = ra_hms.split(':')
    if len(ra_parts) != 3:
        raise ValueError(f"Formato de RA inesperado: {ra_hms}")
    h, m, s = map(float, ra_parts)
    ra_decimal = (h + m / 60 + s / 3600) * 15  # Convertimos a grados
    return ra_decimal

# Convertir DEC en formato dms ("-53*10:21") o decimal ("-53.178") a grados decimales
def dms_to_decimal(dec_dms):
    try:
        return float(dec_dms)
    except ValueError:
        pass

    sign = -1 if dec_dms.strip()[0] == '-' else 1
    dec_dms = dec_dms.replace('*', ':').strip("+-")
    dec_parts = dec_dms.split(':')

    if len(dec_parts) != 3:
        raise ValueError(f"Formato de DEC inesperado: {dec_dms}")

    degrees, minutes, seconds = map(float, dec_parts)
    dec_decimal = sign * (degrees + minutes / 60 + seconds / 3600)
    return dec_decimal

# Configuración de conexión
stellarium_url = "http://localhost:8090/api/objects/info?format=json"
serial_port = "COM3" # Cambiar según el puerto de tu Arduino
baud_rate = 9600

# Conectar al Arduino
try:
    ser = serial.Serial(serial_port, baud_rate, timeout=1)
    time.sleep(2)  # Esperar que Arduino reinicie
except Exception as e:
    print("Error abriendo puerto serial:", e)
    exit()
```

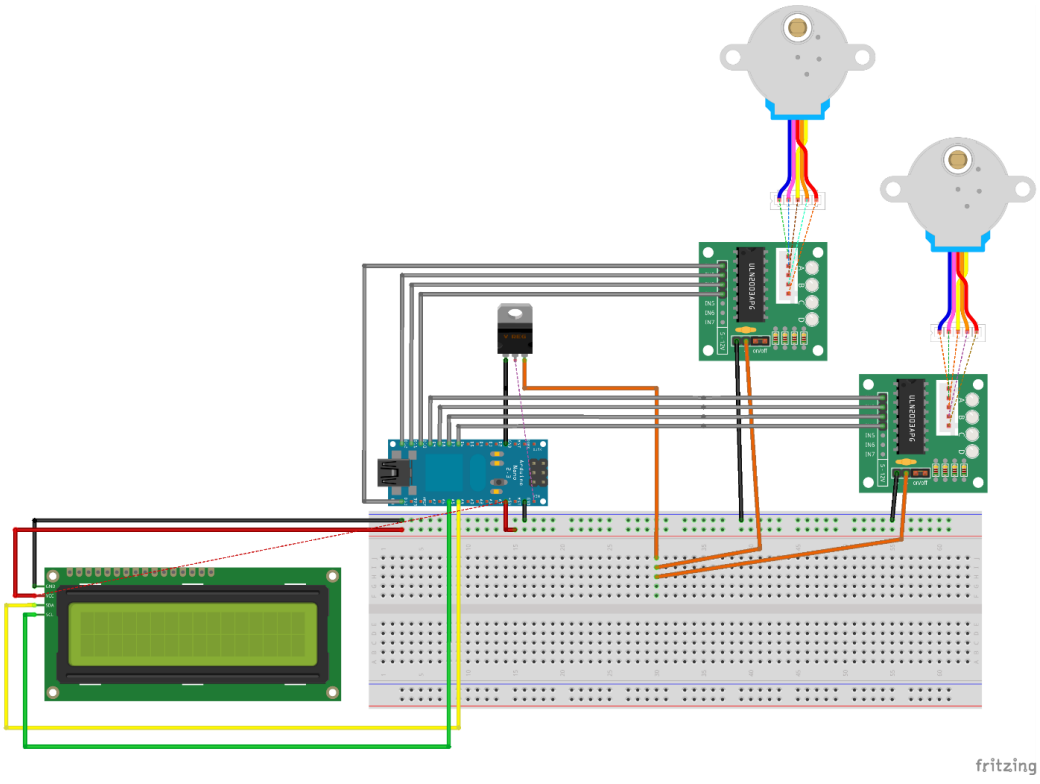
```
print("Iniciando seguimiento automático...")

while True:
    try:
        response = requests.get(stellarium_url)
        data = response.json()

        if not data.get("found", False):
            print("No hay objeto seleccionado.")
            time.sleep(1)
            continue

        # Obtener RA y DEC
        ra_value = str(data["ra"])          # Puede ser decimal o hms
        dec_value = str(data["dec"])        # Puede ser decimal o dms

        # Convertir a decimal
        ra_decimal = hms_to_decimal(ra_value)
        dec_decimal = dms_to_decimal(dec_value)

        # Mostrar
        print(f"RA en decimal: {ra_decimal:.9f}")
        print(f"DEC en decimal: {dec_decimal:.9f}")

        # Enviar a Arduino
        ser.write(f":Sr{ra_decimal:.9f}#".encode())
        time.sleep(0.1)
        ser.write(f":Sd{dec_decimal:.9f}#".encode())
        time.sleep(0.1)
        ser.write(":MS#".encode())
        time.sleep(1)

    except Exception as e:
        print("Error en conexión:", e)
        time.sleep(2)
```

9.4. Anexo 4: *Script Python* para automatización de teclado

```python
import pyautogui
import time

# Esperar unos segundos para que el usuario active Stellarium
time.sleep(5)

# Repetir cada 2 segundos: simular teclas para "objeto actual" y "rotar
hacia él"
while True:
    # Simular presionar la tecla para "Objeto actual" (barra espaciadora
por defecto)
    pyautogui.press('space')

    # Esperar un poco
    time.sleep(0.5)

    # Simular presionar la tecla para "rotar el telescopio hacia el objeto"
    pyautogui.hotkey('ctrl', '1')  # Ctrl+1 es la tecla por defecto

    # Esperar antes de repetir
    time.sleep(2)
```

9.5. Anexo 5: Diagrama de conexión electrónica del sistema





Circuito elaborado con el *software Fritzing*. Los motores 28BYJ-48 funcionan con un regulador 5V, pero el driver es compatible con 5V-12V. Para mi prototipo utilicé una batería 9V y una fuente de 12V.

9.6. Anexo 6: Archivos STL y de fusión 360 del diseño de propuesta

Enlace: https://github.com/monk-hol/Montura-Ecuatorial-Robotica-tesis-ULAT.git

9.7. Anexo 7: Carta de revisión del profesor de español

Panamá, 10 de marzo de 2026

Señores

UNIVERSIDAD LATINA DE PANAMÁ

E. S. D.

Autoridades

*La (el) suscrita (o) notifica haber revisado por solicitud del estudiante**, JUAN CARLOS CASTILLO JAMES,** con cédula de identidad personal número **8-1055-463,** el Proyecto de investigación **"DESARROLLAR UNA MONTURA ECUATORIAL ROBÓTICA PARA EL SEGUIMIENTO DE UNA COMUNICACIÓN SATELITAL",** el cual cumple satisfactoriamente con todos los requisitos formales de ortografía y de redacción exigidos por el idioma español.*

Atentamente,

Gloria P. de Ochys
C.I.P. 9-68-965
Corrección y Estilo

***NOTA:** Este es un formato de carta para él o la profesor (a) de español que le revise el proyecto final de graduación.*