

TESIS CARRERA DE GRADO EN INGENIERÍA
MECÁNICA

DISEÑO, IMPLEMENTACIÓN Y CONTROL DE UN
BANCO DE PRUEBAS PARA EL CONTROL DE
VIBRACIONES DE UN COLIMADOR OSCILANTE

Marten Trapp
Estudiante

Mg. Ing. Santiago Pincin
Director

Ing. Juan Carlos García
Co-director

Miembros del Jurado
Ing. Matías Marticorena
Ing. Marcelo Javier Castelao Caruana

26 de Julio de 2023

Departamento de Termohidráulica – Gerencia de Ingeniería Nuclear
Centro Atómico Bariloche

Instituto Balseiro
Universidad Nacional de Cuyo
Comisión Nacional de Energía Atómica
Argentina

(Biblioteca Leo Falicov CAB-IB)

A mi familia

A mis amigos

A todos los que me acompañaron en el proceso

Resumen

En el marco del proyecto del Laboratorio Argentino de Haces de Neutrones (LAHN), se desarrolló un prototipo de colimador oscilante para el difractómetro multipropósito ANDES. Este colimador genera vibraciones debido a sus oscilaciones y a su masa considerable, lo cual podría afectar la electrónica de detección y al detector. Para abordar este problema, se implementó un actuador inercial en el prototipo, el cual busca compensar las fuerzas inerciales del colimador mediante la aceleración de su masa.

En este proyecto integrador se buscó diseñar, implementar y controlar un banco de pruebas para controlar las vibraciones del prototipo mencionado. Esto implicó verificar los componentes críticos del dispositivo, diseñar el banco de pruebas, realizar un modelado dinámico del conjunto, instrumentar físicamente el conjunto y programar el microcontrolador y la interfaz necesaria. Luego de solucionar tanto inconvenientes físicos como de programación se finalizó realizando mediciones sobre el conjunto.

Se realizaron cálculos analíticos para verificar los componentes críticos del prototipo, considerando un perfil de velocidades trapezoidal en el movimiento. Se utilizó un diseño 3D para medir las distancias y determinar las masas de los conjuntos, y se calcularon los factores de seguridad siguiendo los catálogos de fabricantes.

En relación al diseño del banco de pruebas, se consideró una frecuencia natural para la estructura soporte final del prototipo y se determinó la rigidez necesaria de la misma. Se buscó diseñar el banco con la misma rigidez y utilizando materiales similares, pero una revisión del mismo llevó a corregir su rigidez teórica.

En el análisis del conjunto del banco de pruebas con el prototipo colimador oscilante montado, se evaluó la reducción de vibraciones transmitidas al banco tanto sin el movimiento del actuador inercial como con el movimiento en contrafase del mismo. Se realizó un modelado del sistema con 3 grados de libertad y se calculó la respuesta dinámica del sistema.

Finalizando se obtuvieron mediante los ensayos experimentales algunos resultados congruentes con el modelo dinámico. Se destaca que se pudo verificar una disminución de las oscilaciones al compensar las fuerzas con la actuación del actuador inercial.

Palabras clave: VIBRACIONES, ACTUADOR INERCIAL, MICROCONTROLADOR, QT CREATOR

Abstract

Within the framework of the Argentine Neutron Beam Laboratory (LAHN) project, an oscillating collimator prototype was developed for the multipurpose diffractometer ANDES. This collimator generates vibrations due to its oscillations and considerable mass, which could affect the detection electronics. To address this issue, an inertial actuator was implemented in the prototype, which aims to compensate for the collimator's inertial forces by accelerating its mass.

In this integrative project, the goal was to design, implement, and control a test bench to manage the vibrations of the mentioned prototype. This involved verifying critical components of the device, designing the test bench, performing dynamic modeling of the system, physically instrumenting the assembly, and programming the microcontroller and necessary interface. Both physical and programming challenges had to be addressed, but ultimately, measurements were taken on the assembly.

Analytical calculations were carried out to verify the critical components of the prototype, considering a trapezoidal velocity profile in the motion. A 3D design was used to measure distances and determine the masses of the assemblies, and safety factors were calculated following manufacturers catalogs.

Regarding the design of the test bench, a natural frequency was considered for the final support structure of the prototype, and the necessary stiffness of it was determined. The goal was to design the bench with the same stiffness and using similar materials, but a review of it led to correcting its theoretical stiffness.

In the analysis of the test bench assembly with the mounted oscillating collimator prototype, the reduction of vibrations transmitted to the bench was evaluated both without the motion of the inertial actuator and with the counter-phase motion of the actuator. A 3-degree-of-freedom system modeling was performed, and the dynamic response of the system was calculated.

Finally, experimental tests yielded results consistent with the dynamic model. The most notable finding was the verification of reduced oscillations by compensating the forces through the action of the inertial actuator.

Keywords: VIBRATIONS, INERTIAL ACTUATOR, MICROCONTROLLER, QT CREATOR

Índice de contenidos

Resumen	v
Abstract	vii
Índice de contenidos	ix
Índice de figuras	xiii
Índice de tablas	xvii
Índice de símbolos	xix
1. Introducción	1
1.1. Contexto general	1
1.2. Prototipo alpha del Colimador Oscilante	1
1.3. Objetivos	4
2. Verificación analítica de componentes principales del prototipo	7
2.1. Introducción	7
2.2. Metodología	9
2.2.1. Tornillo de husillo de bolas	9
2.2.2. Guías lineales	13
2.2.3. Eje pivote y rodamientos	17
2.3. Resultados	23
2.3.1. Tornillos de husillos de bola	23
2.3.2. Guías lineales	24
2.3.3. Eje pivote y rodamientos	26
2.4. Conclusión	27
3. Diseño del banco de pruebas	29
3.1. Introducción	29
3.2. Diseño	30
3.3. Revisión del diseño	33

4. Modelado dinámico del conjunto prototipo más banco de pruebas	35
4.1. Introducción	35
4.2. Metodología	36
4.3. Resultados	40
4.4. Conclusiones	44
5. Instrumentación del prototipo	47
5.1. Introducción	47
5.2. Hardware	47
5.3. Software	51
5.3.1. Programa del microcontrolador	51
5.3.2. Programa de la interfaz	59
5.4. Conclusión	63
6. Ensayos experimentales	65
6.1. Introducción	65
6.2. Metodología	67
6.3. Resultados	69
6.4. Conclusión	74
7. Conclusiones generales y trabajos a futuro	75
7.1. Conclusiones	75
7.2. Trabajos a futuro	77
Bibliografía	79
Agradecimientos	81
A. Práctica Profesional Supervisada y actividades de proyecto y diseño	83
A.1. Actividades Relacionadas a la Práctica Profesional Supervisada	83
A.2. Actividades de proyecto y diseño	84
B. Planos del banco de pruebas diseñado	85
C. Código correspondiente al modelado dinámico	89
D. Plano de la placa electrónica diseñada y fabricada	99
E. Códigos utilizados para programar el microprocesador STM32	101
F. Códigos utilizados para la programación de la interfaz	131
F.1. Colimador-Com-serie-BP-QT-V7.pro	131

F.2. comandos.h	132
F.3. comunicacion.h	134
F.4. mainwindow.h	140
F.5. comunicacion.cpp	142
F.6. main.cpp	151
F.7. mainwindow.cpp	151
G. Código utilizado para el análisis de los datos experimentales	163

Índice de figuras

1.1. Esquema explicativo del funcionamiento de un difractómetro.	2
1.2. Prototipo de colimador oscilante diseñado.	3
1.3. Esquema de los movimientos de los componentes del colimador oscilante.	3
2.1. Componentes críticos identificados del prototipo colimador oscilante. .	8
2.2. Perfil de velocidad trapezoidal que se busca en los movimientos lineales del colimador y del actuador.	9
2.3. Esquema de fuerza que debe soportar el tornillo del husillo de bolas recirculantes.	11
2.4. Distancias medidas entre el centro de masa del conjunto móvil del colimador y el eje de la guía lineal y el centro de los carros A y B sobre los cuales se desplaza.	13
2.5. Distancias medidas entre el centro de masa del conjunto móvil del actuador y el eje de la guía lineal y el centro de los carros A y B sobre los cuales se desplaza.	14
2.6. Reacciones que debe soportar el conjunto guía lineal y los carros sobre los cuales desliza el colimador.	14
2.7. Reacciones que debe soportar el conjunto guía lineal y los carros sobre los cuales desliza el colimador.	15
2.8. Propiedades de las guías lineales utilizadas.	16
2.9. Factor de carga para el cálculo de la vida en servicio de cada guía. . . .	17
2.10. Distancias medidas entre el centro de masa del conjunto rotante y la sección analizada del eje pivote.	17
2.11. Diagramas esquemáticos de corte y momento flector del eje pivote. . . .	18
2.12. Croquis del eje pivote con los esfuerzos trasladados a la sección analizada.	18
2.13. Factor de concentración de tensiones para un eje acanalado sometido a flexión y/o tensión.	20
2.14. Esfuerzos trasladados a la sección media entre los rodamientos.	22
2.15. Factores de seguridad recomendados por THK-TBR para los tornillos de husillos de bolas recirculantes.	24

2.16. Valores recomendados por HIWIN para los factores de seguridad estáticos de las guías lineales.	25
2.17. Factores de seguridad recomendados para rodillos de bolas en distintas situaciones de carga.	27
3.1. Estructura soporte del colimador oscilante y el detector del difractor ANDES.	30
3.2. Constante elástica equivalente para N vigas empotradas en ambos extremos.	32
3.3. Banco de pruebas diseñado con sus medidas principales.	32
3.4. Banco de pruebas diseñado con el prototipo colimador oscilante montado.	33
3.5. Deflexión del banco de pruebas como pórtico empotrado en el suelo.	34
4.1. Esquema plano del banco de pruebas en conjunto con el prototipo colimador oscilante montado.	37
4.2. Modelo de 3 GDL del banco de pruebas con el prototipo colimador oscilante montado.	37
4.3. Perfil de fuerza aplicada al colimador y al actuador en un periodo.	38
4.4. Desplazamiento absoluto de m_1 sin accionar el actuador.	41
4.5. Desplazamiento absoluto de m_1 accionando el actuador.	41
4.6. Comparación del desplazamiento absoluto de m_1 sin y con el accionamiento del actuador.	42
4.7. Aceleraciones de m_1 sin y con el accionamiento del actuador.	42
4.8. Perfil de fuerza aplicada al colimador y al actuador en un periodo con un desplazamiento de 80 mm del actuador.	43
4.9. Comparación del desplazamiento absoluto de m_1 sin y con el accionamiento del actuador para un recorrido de 80mm de este.	44
4.10. Aceleraciones de m_1 sin y con el accionamiento del actuador para un recorrido de 80 mm de este.	44
5.1. Esquema de interconexión entre los componentes que forman el hardware.	48
5.2. Prototipo inicial de la placa electrónica.	49
5.3. Placa electrónica fabricada para la instrumentación del prototipo colimador oscilante.	50
5.4. Soporte de electrónica.	50
5.5. Fijación del acelerómetro en la estructura.	51
5.6. Diagrama de flujo del código programado para el microcontrolador.	52
5.7. Diagrama de flujo del código programado para la función BucleCtrl.	56
5.8. Diagrama de flujo del código programado para la función Homefunc.	58

5.9. Diagrama de flujo del código para un solo motor programado para la función <code>CodigoG</code>	59
5.10. Interfaz diseñada para el control de los movimientos y lecturas por parte del usuario.	60
5.11. Diagrama de secuencias del código de la interfaz.	62
5.12. Interfaz diseñada para control de los movimientos por el usuario en funcionamiento.	63
6.1. Disposición experimental del conjunto banco de pruebas con colimador oscilante instrumentado.	66
6.2. Arreglo experimental para la medición estática de la rigidez del banco de pruebas.	67
6.3. Desplazamiento vs fuerza aplicada obtenidos para la determinación de la rigidez del banco de pruebas.	69
6.4. Aceleraciones medidas en la plataforma del banco de pruebas ante un impulso inicial en el tiempo aproximado 1,1s.	70
6.5. Detalle de las aceleraciones medidas en la plataforma del banco de pruebas ante un impulso inicial en el tiempo aproximado 1,1s.	70
6.6. Periodograma obtenido del banco de pruebas ante un impulso inicial.	71
6.7. Perfil de aceleración medido con el movimiento del colimador sin el actuador.	72
6.8. Perfil de aceleración medido con el movimiento del colimador más el actuador en contrafase.	72
6.9. Perfil de desplazamiento obtenido con el movimiento del colimador sin el actuador.	73
6.10. Perfil de desplazamiento obtenido con el movimiento del colimador más el actuador en contrafase.	73

Índice de tablas

2.1. Coeficientes de corrección para el cálculo de la resistencia a la fatiga corregida.	21
2.2. Fuerzas axiales que deben soportar los tornillos durante un ciclo de movimiento.	23
2.3. Factores de seguridad de los tornillos de husillo de bolas obtenidos ante carga de tracción/compresión y ante pandeo para el conjunto móvil del colimador y el actuador inercial.	24
2.4. Distancias medidas entre los centros de gravedad de los conjuntos móviles y los ejes de las guías lineales como así también los centros de los carros.	25
2.5. Reacciones obtenidas que deben soportar los carros de las guías lineales de cada conjunto móvil	25
2.6. Factores de seguridad de las guías lineales ante carga y momento obtenidos para el conjunto móvil del colimador y el actuador inercial.	25

Índice de símbolos

- a : aceleración.
- a_{anillo} : espesor del alojamiento del anillo de retención.
- a_{kt} : coeficiente para el cálculo de la sensibilidad a la muesca.
- A_y : aceleración medida con el acelerómetro.
- A_{yf} : aceleración medida con el acelerómetro y filtrada.
- C : capacidad de carga dinámica.
- C_{carga} : coeficiente de corrección a la fatiga por tipo de carga.
- C_{conf} : coeficiente de corrección a la fatiga por confiabilidad.
- CM : centro de masa.
- C_{sup} : coeficiente de corrección a la fatiga por terminación superficial.
- $C_{tamaño}$: coeficiente de corrección a la fatiga por tamaño.
- C_{temp} : coeficiente de corrección a la fatiga por temperatura.
- C_0 : capacidad de carga estática.
- d : diámetro menor.
- D : diámetro mayor.
- E : módulo de elasticidad.
- f : fuerza de fricción de las guías lineales sin carga.
- $F_{a1}, F_{a2}, F_{a3}, F_{a4}, F_{a5}, F_{a6}$: fuerzas de tracción involucradas en el movimiento con perfil de velocidad trapezoidal.
- FFT: fast Fourier transform.
- f_n : frecuencia natural.

- f_{SL} : factor de seguridad a cargas simples.
- f_{SM} : factor de seguridad al momento.
- $FS_{trac/comp}$: factor de seguridad ante los esfuerzos de tracción y compresión.
- FS_{pandeo} : factor de seguridad ante el pandeo.
- f_w : factor de carga.
- F_x : fuerza ejercida en la dirección del eje x.
- F_1, F_2, F_3 : fuerzas involucradas en el modelado dinámico.
- g : aceleración gravitatoria.
- I : momento de inercia polar.
- J : momento de inercia.
- K : rigidez o constante elástica.
- k_t : factor de concentración de tensiones corregido por sensibilidad a la muesca.
- k_t : factor de concentración de tensiones.
- L : longitud de desplazamiento.
- l_a : distancia entre los puntos de montaje de los tornillos de los husillos de bolas.
- $L_{columnas}$: longitud de las columnas del banco de pruebas.
- l_h : vida de servicio.
- l_x : distancia entre el centro de masa y un punto a lo largo del eje x.
- l_y : distancia entre el centro de masa y un punto a lo largo del eje y.
- l_z : distancia entre el centro de masa y un punto a lo largo del eje z.
- m : masa.
- M_{neto} : momento flector neto.
- M_R : momento torsor sobre las guías lineales.
- M_{R0} : momento torsor admisible de las guías lineales.
- M_x : componente en dirección al eje x del momento flector.

- M_y : componente en dirección al eje y del momento flector.
- M_z : componente en dirección al eje z del momento flector.
- N_f : factor de seguridad a la fatiga.
- N_{max} : velocidad máxima de rotación.
- P : peso.
- P_{crit} : carga crítica para el pandeo.
- Ph : paso de los tornillos de los husillos de bolas.
- P_0 : carga estática equivalente en los rodamientos.
- P_1 : carga admisible al pandeo de los tornillos de los husillos de bolas.
- P_2 : carga admisible a la tracción/compresión de los tornillos de los husillos de bolas.
- q : sensibilidad a la muesca.
- R : fuerzas reactivas que ejerce un elemento.
- r_{anillo} : radio de encuentro en el alojamiento del anillo de retención.
- S'_e : límite de resistencia a la fatiga.
- S_f : límite de resistencia a la fatiga corregido.
- S_{ut} : tensión última.
- s_0 : factor de seguridad de los rodamientos a carga estática.
- T : período.
- t_{anillo} : profundidad del alojamiento del anillo de retención.
- $ta\%T$: tiempo de aceleración como porcentaje del período.
- T_{max} : torque máximo que debe entregar un motor.
- V_{max} : velocidad máxima lineal.
- α : aceleración angular.
- η_1 : corrección de la carga admisible por pandeo por modo de montaje.
- μ : coeficiente de fricción.

- ϕ : matriz de modos normalizados.
- f : vector de frecuencias.
- $\sigma_{comp, trac, adm}$: tensión admisible a la tracción/compresión de los tornillos de los husillos de bolas.
- σ'_a : tensión alterna.
- σ'_m : tensión media.
- ξ : factor de amortiguamiento viscoso.

Capítulo 1

Introducción

“No existe una segunda oportunidad para una buena primera impresión”

— Oscar Wilde

1.1. Contexto general

El reactor argentino RA10, el cual se encuentra en etapa de construcción al momento de escribir esta tesis, tiene por objetivo, entre otros, brindar facilidades a distintas instalaciones que lo rodean. Una de estas instalaciones es el Laboratorio Argentino de Haces de Neutrones (LAHN) [1]. Este es un centro de investigación orientado al desarrollo y aplicación de técnicas neutrónicas en diversas disciplinas, con fines académicos, científicos y tecnológicos.

Una de las principales facilidades que brindará este laboratorio será el escáner de tensiones o difractómetro multipropósito ANDES (por sus siglas en inglés: Advanced Non-Destructive Evaluation of Stress). Este difractómetro permitirá realizar experimentos de difracción de naturaleza no destructiva sobre distintos tipos de piezas mecánicas y muestras. La intención es que sea utilizado por entes públicos y privados para realizar ensayos no destructivos de componentes críticos de alta tecnología.[2]

Particularmente, dentro de este difractómetro, un componente funcional es el colimador oscilante en el cual se basa este proyecto. Un prototipo alpha de este colimador fue diseñado por el Mg. Ing. Santiago Pincin, director de este proyecto integrador.

1.2. Prototipo alpha del Colimador Oscilante

Un colimador de haces de neutrones es un dispositivo que mediante una serie de láminas absorbentes de neutrones permite obtener un haz paralelo a partir de un haz

difuso. En la Figura 1.1 se puede observar un esquema del funcionamiento de la óptica del haz monocromático proveniente del difractor ANDES. En la figura se observa un haz de neutrones con una cierta longitud de onda, que provienen del monocromador del difractor ANDES. Este haz pasa a través de un colimador y, una vez colimado, impacta sobre la muestra. Dependiendo de la micro estructura de la muestra este haz se difractará en distintas direcciones con distintas intensidades. Para detectar en mayor medida los neutrones difractados provenientes de la muestra, se tiene una estructura capaz de posicionarse angularmente alrededor de la muestra. Esta estructura soporta al colimador oscilante y al detector. [3]

Para poder cumplir la función de colimación, y evitar zonas en desuso del detector, se requiere que el colimador oscile con una frecuencia de entre 0,1 Hz y 1 Hz, con un perfil de velocidades trapezoidal y un movimiento semiparabólico con una longitud de 30 mm. Dado que el colimador tiene una masa de aproximadamente 50 kg su oscilación genera fuerzas inerciales durante los momentos de aceleración y desaceleración que deben ser soportadas por la estructura. Estas fuerzas son capaces de generar vibraciones que en caso de llegar al detector pueden afectar su medición y son indeseables para su electrónica.

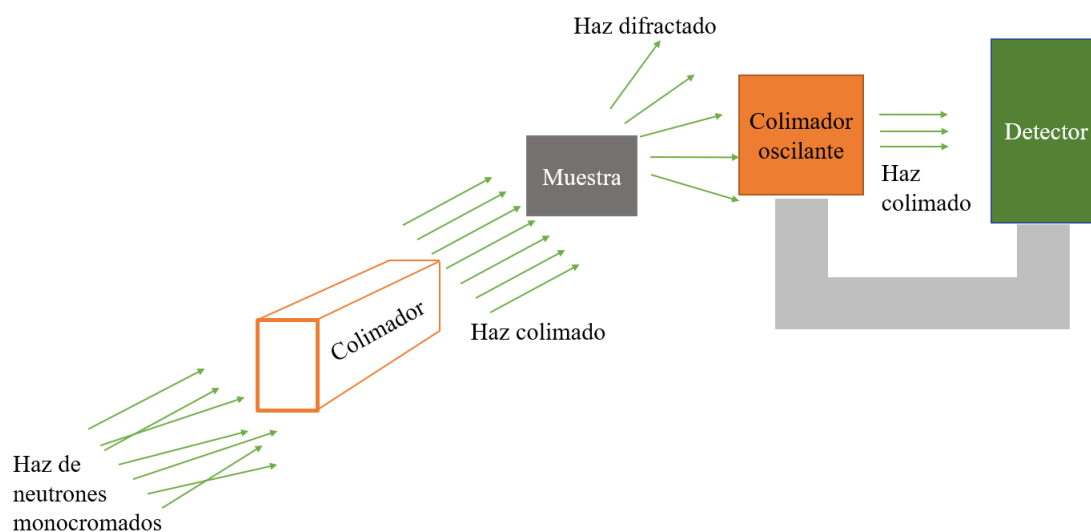


Figura 1.1: Esquema explicativo del funcionamiento de un difractor.

El prototipo de colimador oscilante diseñado por el Mg. Ing. Santiago Pincin se muestra en la Figura 1.2. En este prototipo se propone amortiguar las vibraciones producidas por el movimiento del colimador mediante un actuador inercial con un sistema de control. Un actuador inercial es un dispositivo que realiza una fuerza mediante la aceleración de una masa. Por ser inercial no puede generar una fuerza continua, pero dado que las vibraciones se producen también por fuerzas inerciales del colimador, se pretenden contrarrestar estas fuerzas con fuerzas de la misma naturaleza. Cabe aclarar que este prototipo se fabricó por el taller de Termohidráulica del Centro Atómico Ba-

rilоче durante el primer semestre de este trabajo y luego de este Proyecto Integrador va a ser probado y puesto en marcha en el LAHN.

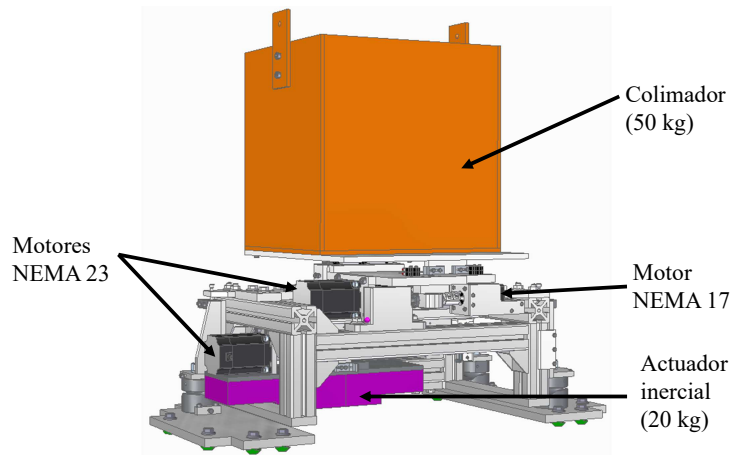


Figura 1.2: Prototipo de colimador oscilante diseñado.

Entrando en los detalles de diseño del prototipo se pueden resaltar los mecanismos actuadores y las masas involucradas, que se indican en la Figura 1.2. La masa del actuador inercial es movida por un motor paso a paso NEMA 23 a través de un husillo de bolas recirculantes. Esta masa se mueve únicamente de forma lineal. Por otro lado, el movimiento semiparabólico del colimador se consigue superponiendo un movimiento lineal y un movimiento angular oscilatorio alrededor del eje pivote. El movimiento lineal del colimador también se consigue a través de un motor paso a paso NEMA 23 y un husillo de bolas recirculantes. El movimiento angular se obtiene mediante un motor paso a paso NEMA 17 y un tornillo de potencia. En la Figura 1.3 se esquematizan los movimientos descritos.

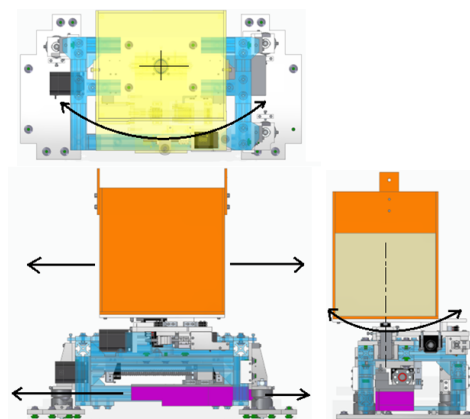


Figura 1.3: Esquema de los movimientos de los componentes del colimador oscilante.

1.3. Objetivos

El proyecto integrador que se expone en esta tesis tiene la característica de abarcar múltiples áreas de la ingeniería mecánica. Los objetivos principales de este proyectos son:

- Diseñar e implementar un banco de pruebas para ensayar el mecanismo de control de vibraciones del colimador oscilante de ANDES.
- Diseñar e implementar la electrónica del control de vibraciones, basándose en un actuador inercial.
- Realizar el modelado del control de vibraciones, y posteriormente implementarlo en un sistema embebido.

Sin embargo, para alcanzar estos objetivos globales se pueden definir objetivos más específicos y puntuales. A continuación se detallan estos objetivos específicos:

- Verificar analíticamente los componentes críticos del prototipo colimador oscilante: Esto refiere a llevar a cabo un análisis exhaustivo utilizando diversos métodos para evaluar las fuerzas a las que se someten los componentes más exigidos del prototipo. Mediante cálculos analíticos, se obtendrán los coeficientes de seguridad que determinarán la capacidad de resistencia del diseño frente a estas fuerzas.
- Diseñar la estructura del banco de pruebas: Esto comprende la realización, desde el diseño conceptual hasta el diseño de detalle y los planos para la fabricación, de un banco de pruebas al cual se le pueda montar el prototipo de colimador oscilante y permita realizar las mediciones necesarias para el control requerido. Este banco se diseñará con el objetivo de que cuente con una determinada rigidez que permita realizar las mediciones requeridas.
- Modelado dinámico del prototipo en conjunto con el banco de pruebas: Esto implica modelar la dinámica de movimiento del colimador oscilante con el actuador inercial y el banco de pruebas para obtener analítica o numéricamente los desplazamientos y aceleraciones esperados en los ensayos experimentales futuros.
- Instrumentación del dispositivo: Esto conlleva el diseño y armado de la plaqueta electrónica para el control de los motores paso a paso, la lectura de los sensores y otros indicadores y la comunicación con la computadora. Además, también involucra la programación del microcontrolador contenido en la plaqueta y de la interfase con el usuario de la computadora para el control del dispositivo.

- Ensayos experimentales: Esto implica implementar todo lo desarrollado anteriormente en un sistema embebido y medir los desplazamientos con y sin el sistema de control de vibraciones para su comprobación.

Esta tesis estará organizada de tal manera que cada objetivo anteriormente descrito comprende un capítulo de la misma.

Capítulo 2

Verificación analítica de componentes principales del prototipo

“El hombre de ciencia ha aprendido a creer en la justificación, no por la fe, sino por la verificación”

— Thomas Henry Huxley, 1866

2.1. Introducción

El proceso de diseño mecánico implica desde la generación de conceptos y selección de los mismos hasta el desarrollo de la ingeniería de detalle de cada pieza para su fabricación. Durante el proceso de ingeniería de detalle se buscan determinar las dimensiones, acabados superficiales y materiales de cada pieza que forma parte del producto diseñado, como también las interfases entre estas. En esta misma etapa también se realiza la selección y verificación de componentes comerciales.

Dado que usualmente un diseño de una máquina o mecanismo cuenta con una gran cantidad de piezas, no es práctico calcular analíticamente la resistencia de cada uno de los componentes ya que, utilizando criterios de buena práctica ingenieril, experiencias anteriores o códigos y estándares ingenieriles, se puede justificar su elección. Sin embargo, dentro de cualquier máquina o mecanismo, es posible identificar componentes críticos, ya sea debido a su importancia en la funcionalidad del dispositivo o a su ubicación crucial en el flujo de fuerzas. En este capítulo se buscará verificar analíticamente que cada componente crítico cumpla con los criterios de aceptación establecidos según el modo de falla mecánico establecido para este.

En la Figura [2.1](#) se exponen los componentes críticos identificados del prototipo

colimador oscilante. Se verificarán los tornillos de los husillos de bola del colimador y del actuador inercial utilizando como criterio de falla el pandeo y la fluencia de los componentes rotantes por las fuerzas de compresión y tracción. Luego se verificará la resistencia a la fluencia de las guías lineales del colimador y del actuador inercial con los momentos y esfuerzos lineales que estas deben absorber. Para finalizar se verificará el eje pivote a fatiga y sus correspondientes rodamientos a carga estática. Para todas las verificaciones se reportará como resultado final el factor de seguridad que tienen los componentes a los modos de falla verificados, estableciendo como criterio de aceptación que este se encuentre dentro de los límites especificados por el fabricante. En caso de tratarse de elementos de fabricación propia, como el eje pivote, se considerará como criterio de aceptación que el factor de seguridad sea mayor a 3,1 siguiendo el criterio de Pugsley. Este es un criterio para determinar factores de seguridad basándose tanto en los riesgos económicos y sobre las personas que produciría la falla del componente, como también considerando la calidad de los materiales, mantenimiento e inspección; el control sobre la carga aplicada a la parte; y el nivel análisis de esfuerzos, información experimental o experiencia con dispositivos similares que se tiene. Para mayor información se recomienda consultar el capítulo 1 de la bibliografía [4]. El valor de factor de seguridad de 3,1 se obtuvo considerando una calidad de los materiales y conocimiento de esfuerzos regular, un control sobre las cargas aplicadas pobre, un impacto económico serio y un impacto sobre las personas poco serio.

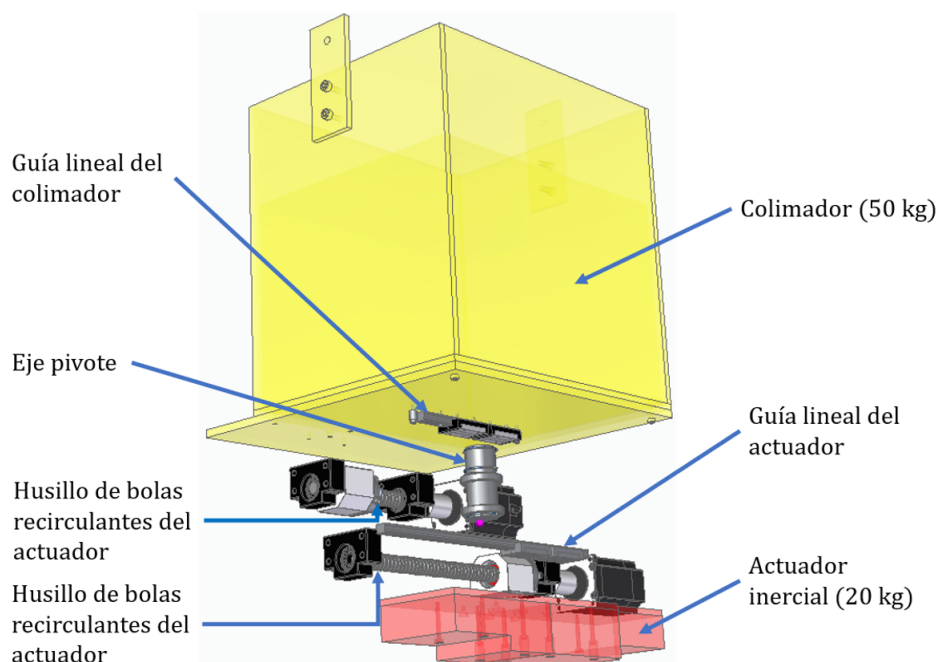


Figura 2.1: Componentes críticos identificados del prototipo colimador oscilante.

2.2. Metodología

2.2.1. Tornillo de husillo de bolas

Para la verificación de los tornillos de husillos de bolas del colimador y del actuador inercial se siguió el catálogo de husillo de bolas recirculantes [5]. Con el objetivo de ser lo más conservativos posible se decidió considerar la frecuencia máxima de trabajo que es 1 Hz, esto da un período mínimo de trabajo de $T = 1$ s. Además, dado que se busca obtener una velocidad constante la mayor parte del tiempo, se consideró un movimiento con un perfil de velocidades trapezoidal como el que se muestra en la Figura 2.2. Se ajustó iterativamente el tiempo de aceleración como un porcentaje del período ($ta_{\%T}$), es decir, se varió x siendo $ta_{\%T} = x\%T$. Se determinó un x mínimo de 1 para no sobre exigir los motores y un x máximo de 25 ya que con este se tiene un perfil de velocidades triangular. Todo esto considerando una velocidad máxima de los motores de 2000 RPM siendo la velocidad máxima teórica de los motores NEMA 23 utilizados de 2400 RPM. Se consideró además que los tornillos solo absorben cargas axiales y momentos torsores, pero no cargas radiales ni momentos flectores.

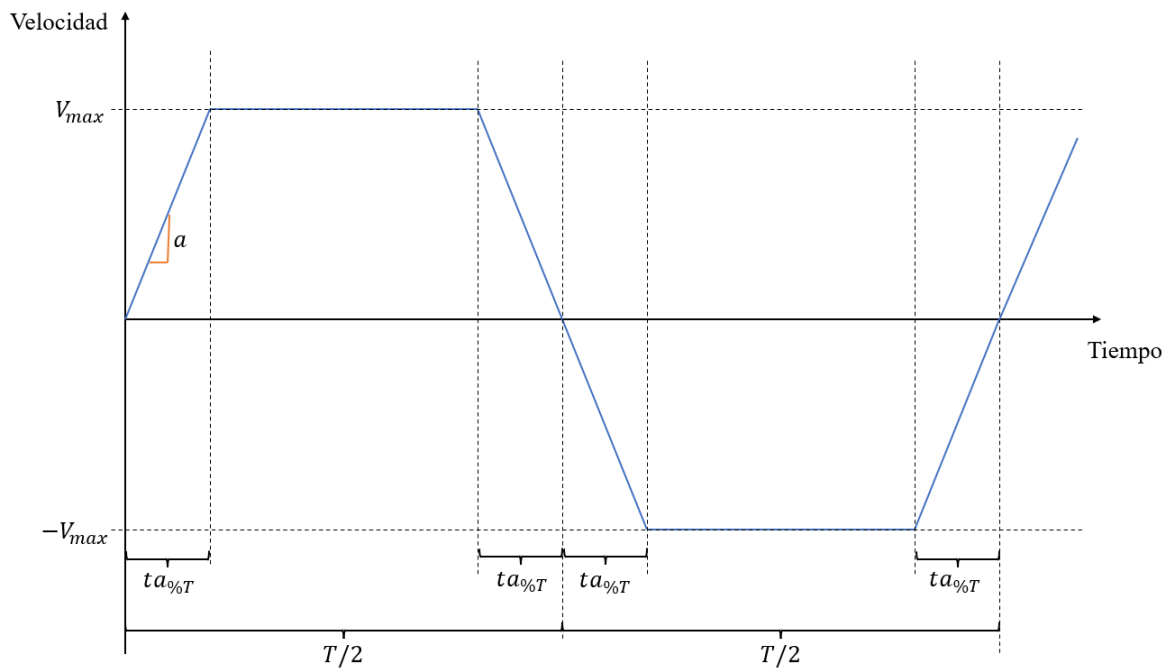


Figura 2.2: Perfil de velocidad trapezoidal que se busca en los movimientos lineales del colimador y del actuador.

Mediante la ecuación 2.1 se tiene la longitud recorrida en función de la velocidad máxima alcanzada (V_{max}), el período (T) y el tiempo de aceleración como porcentaje del período ($ta_{\%T}$).

$$L = V_{max} \left(\frac{T}{2} - 2ta_{\%T} \right) + 2 \frac{V_{max}}{2} ta_{\%T} \quad (2.1)$$

De esta ecuación se puede despejar directamente la velocidad máxima que se obtendrá como se muestra en la ecuación 2.2.

$$V_{max} = \frac{L}{T/2 - ta_{\%T}} \quad (2.2)$$

Luego, siendo la aceleración dada por $a = V_{max}/ta_{\%T}$ se la puede calcular mediante la ecuación 2.3 en función de la longitud de desplazamiento, el período y el tiempo de aceleración.

$$a = \frac{L}{(T/2 - ta_{\%T})ta_{\%T}} \quad (2.3)$$

Los husillos de bola seleccionados para el prototipo cuentan con las siguientes características:

- Paso: $Ph = 5 \text{ mm}$
- Diámetro exterior: $D = 16 \text{ mm}$
- Diámetro interior: $d = 12,8 \text{ mm}$

Con estos datos se obtiene en primer lugar la velocidad angular máxima del motor como muestra la ecuación 2.4. Análogamente, mediante la ecuación 2.5 se calculó la aceleración angular máxima.

$$N_{max} = \frac{V_{max}}{Ph} \cdot 60 \frac{s}{min} \text{ en } [RPM]. \quad (2.4)$$

$$\alpha = \frac{2\pi \cdot a}{Ph} \text{ en } \left[\frac{1}{s^2}\right]. \quad (2.5)$$

Se procedió con el cálculo, mediante la ecuación 2.6, de la carga admisible al pandeo siguiendo lo desarrollado en [5] página 398. Donde $\eta_1 = 2$ ya que el modo de montaje es un montaje fijo en un extremo y con soporte en el otro; el modulo de elasticidad $E = 206 \text{ GPa}$ por ser el eje de acero; l_a corresponde a la distancia entre los puntos de apoyo; e $I = \pi/64 \cdot d^4$ corresponde al momento de inercia polar del tornillo. Notar que ya el propio catálogo impone un factor de seguridad de 0,5 a esta carga admisible.

$$P_1 = \frac{\eta_1 \pi^2 EI}{l_a^2} \cdot 0,5 \quad (2.6)$$

Luego se calculó la carga admisible a la tracción/compresión mediante la ecuación 2.7 siguiendo la misma bibliografía. En esta fórmula $\sigma_{comp, trac, adm} = 147 \text{ MPa}$ definido por el catálogo. La capacidad de carga estática básica de un husillo de bolas recirculantes suele ser equivalente a la carga admisible de compresión/tracción y se define como una carga estática con una dirección y una magnitud constante en la que la suma de la

deformación permanente del elemento giratorio y la de la ranura en el área de contacto bajo el esfuerzo máximo equivale a 0,0001 veces el diámetro del elemento basculante.

$$P_2 = \sigma_{comp, trac, adm} \cdot \frac{\pi}{4} \cdot d^2 \quad (2.7)$$

Continuando se procedió a calcular las fuerzas axiales que debe soportar el tornillo como muestra la Figura 2.3 siguiendo la página 40 del catálogo [5] durante todo un ciclo de movimiento. Cabe aclarar que se considera que el tornillo únicamente soporta cargas axiales, soportando los esfuerzos restantes las guías lineales.

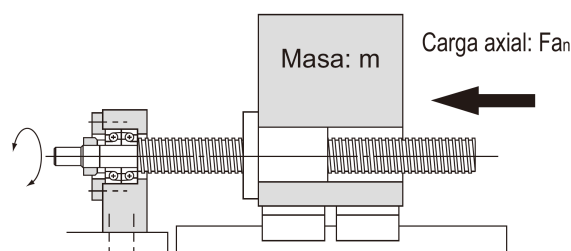


Figura 2.3: Esquema de fuerza que debe soportar el tornillo del husillo de bolas recirculantes.

Para esto se consideró una fricción de las guías lineales igual a la de un rodamiento $\mu = 0,005$ según [6] y una resistencia de la superficie de la guía sin carga de $f = 4 N$ según [7]. Entonces mediante las siguientes ecuaciones se calculó respectivamente la fuerza durante la aceleración 2.8, la fuerza durante la velocidad positiva constante 2.9, la fuerza durante la desaceleración 2.10, la fuerza durante la aceleración en sentido contrario 2.11, la fuerza durante la velocidad constante negativa 2.12 y la fuerza durante la desaceleración en sentido contrario 2.13.

$$Fa_1 = \mu \cdot m \cdot g + f + m \cdot a \quad (2.8)$$

$$Fa_2 = \mu \cdot m \cdot g + f \quad (2.9)$$

$$Fa_3 = \mu \cdot m \cdot g + f - m \cdot a \quad (2.10)$$

$$Fa_4 = (-\mu)m \cdot g - f - m \cdot a \quad (2.11)$$

$$Fa_5 = -\mu \cdot m \cdot g - f \quad (2.12)$$

$$Fa_6 = -\mu \cdot m \cdot g - f + m \cdot a \quad (2.13)$$

Una vez obtenidas las fuerzas que debe soportar el tornillo del husillo de bolas a

tracción y compresión durante toda la dinámica de un ciclo se observa fácilmente de las fórmulas que la mayor fuerza se da en los momentos de aceleración de la masa. Conservativamente se calcularon con esta fuerza los factores de seguridad a la deformación de los elementos rotantes y la pista por tracción/compresión y al pandeo siendo estos dados por las ecuaciones 2.14 y 2.15 respectivamente.

$$FS_{trac/comp} = \frac{P_2}{Fa_1} \quad (2.14)$$

$$FS_{pandeo} = \frac{P_1}{Fa_1} \quad (2.15)$$

Habiendo verificado la resistencia del tornillo del husillo de bolas se procedió calculando el torque requerido sobre este para la dinámica deseada. En particular se calculó el torque máximo requerido sobre este que corresponde al torque requerido para la aceleración de la masa. Este se obtuvo siguiendo la ecuación 2.16 con el momento de inercia del tornillo $J = \frac{m_{tornillo} \cdot D^2}{8}$ donde $m_{tornillo}$ corresponde a la masa del tornillo. Además, al tratarse de un husillo de bolas recirculantes se consideró el coeficiente de fricción de la tuerca igual a la de un rodamiento, entonces, nuevamente de [6] se consideró $\mu_{tuerca} = 0,005$. Este torque, en conjunto con la curva característica del motor paso a paso permitirá determinar el rango de desplazamientos y tiempos de aceleración como porcentaje del período que se encuentran dentro de las limitaciones del motor.

$$T_{max} = \frac{Fa_1 \cdot d}{2} \cdot \frac{\mu_{tuerca} \cdot \pi \cdot d + Ph}{\pi \cdot d - \mu_{tuerca} \cdot Ph} + J \cdot \alpha \quad (2.16)$$

Para finalizar se calculó la rigidez axial del conjunto husillo de bolas. Esta rigidez no debe verificar ningún límite ya que no es parte del requerimiento, pero si se utilizará para el modelado dinámico del conjunto desarrollado en el Capítulo 4. La rigidez total se la calculó mediante la ecuación 2.17 donde la rigidez de la tuerca (K_N) se la calculó siguiendo la ecuación 2.18 y la rigidez del tornillo (K_s) mediante la ecuación 2.19 con el valor de rigidez en las tablas de especificación (K) y la capacidad de carga dinámica básica (C_a) de [8].

$$K_{total} = \frac{K_N \cdot K_s}{K_N + K_s} \quad (2.17)$$

$$K_N = K \cdot \left(\frac{Fa_2}{0,3 \cdot C_a} \right)^{1/3} \cdot 0,8 \quad (2.18)$$

$$K_s = \frac{\pi \cdot d_1^2 \cdot E}{4 \cdot l_a} \quad (2.19)$$

2.2.2. Guías lineales

Para la verificación de las guías lineales se utilizó como elemento de medición el modelo 3D del prototipo colimador oscilante desarrollado en el programa SolidEdge en su versión estudiantil [9]. De este diseño se obtuvo la masa de cada conjunto móvil (colimador y actuador), y la posición de su centro de masa. Con esto se pudieron medir las distancias entre los centros de masa y los ejes de las guías, lo cual permitió obtener los brazos de palanca y hacer un diagrama de reacciones que debe soportar cada guía lineal. En la Figura 2.4 y en la Figura 2.5 se pueden observar las distancias medidas entre el centro de masa del conjunto móvil y los centros de los carros de las guías lineales del colimador y del actuador respectivamente.

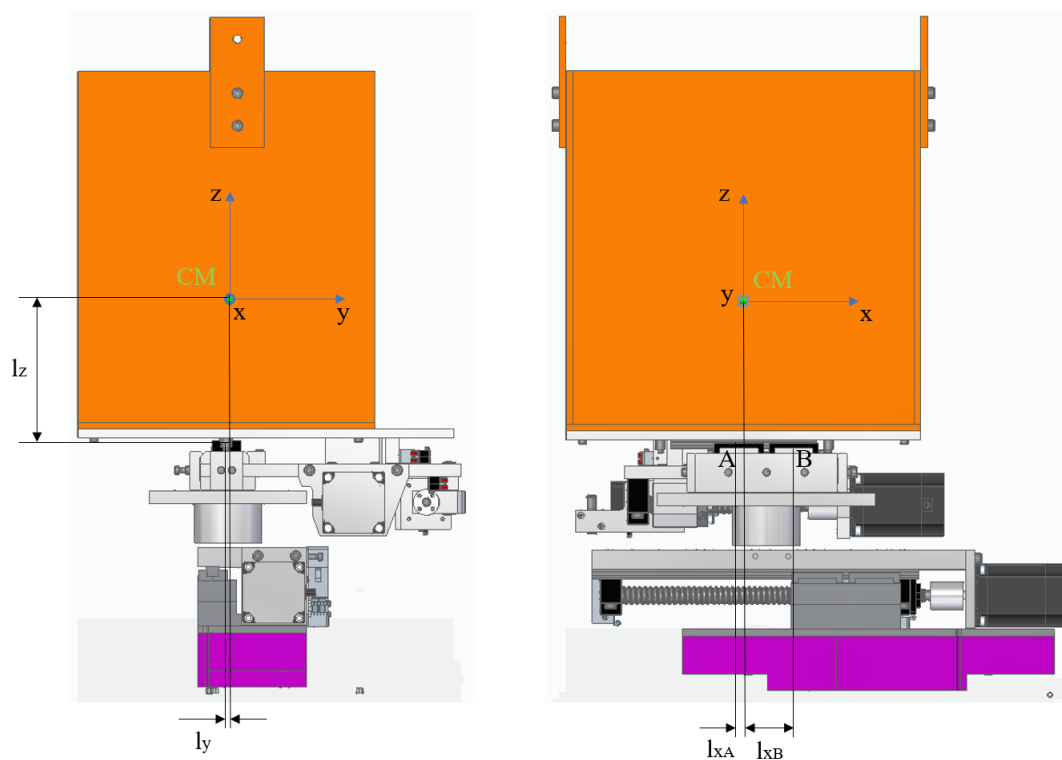


Figura 2.4: Distancias medidas entre el centro de masa del conjunto móvil del colimador y el eje de la guía lineal y el centro de los carros A y B sobre los cuales se desplaza.

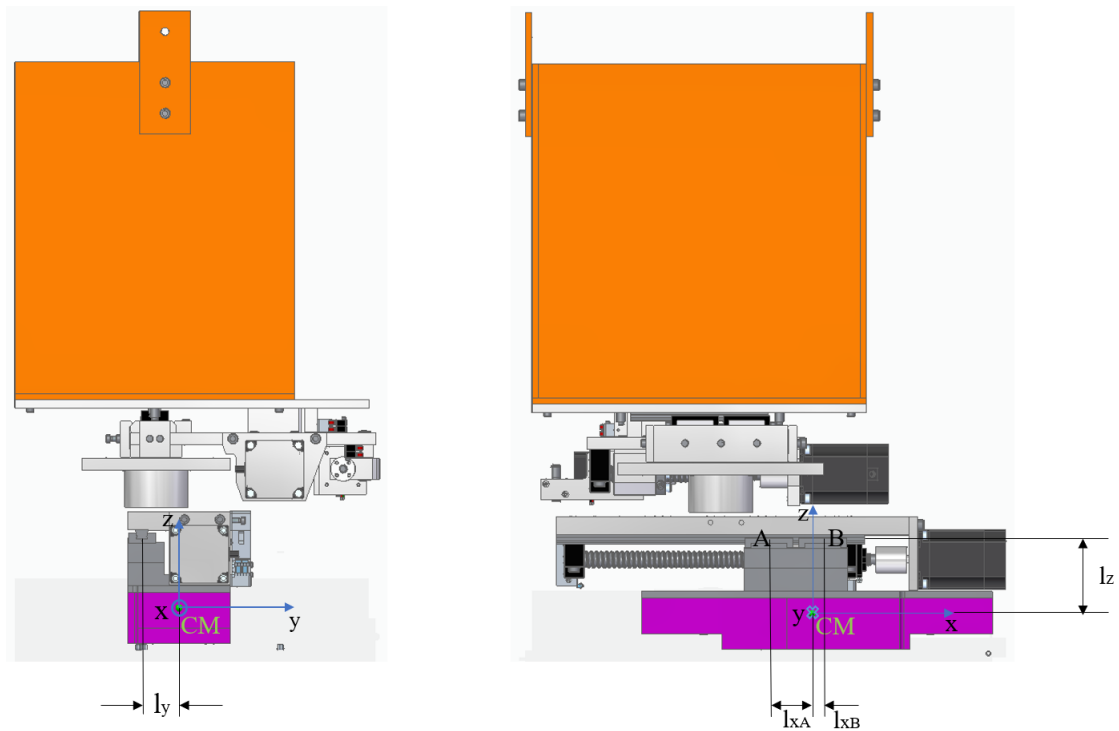


Figura 2.5: Distancias medidas entre el centro de masa del conjunto móvil del actuador y el eje de la guía lineal y el centro de los carros A y B sobre los cuales se desplaza.

Una vez medidas estas distancias se graficaron los diagramas de reacciones para cada guía lineal con sus respectivos carros. En la Figura 2.6 se expone el diagrama graficado para la guía del colimador mientras que en la Figura 2.7 se expone el correspondiente para la guía lineal del actuador. Cabe resaltar que como el diseño del prototipo colimador oscilante comprende dos carros por cada guía lineal, no se consideraron los momentos flectores absorbiendo estos con la combinación de fuerzas normales y laterales entre los dos carros. Además, cabe aclarar que no se consideraron fuerzas laterales que deban absorber los carros.

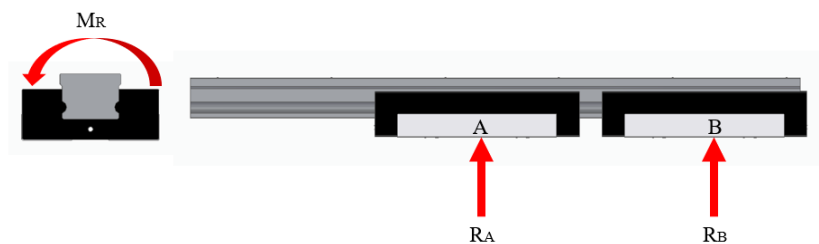


Figura 2.6: Reacciones que debe soportar el conjunto guía lineal y los carros sobre los cuales desliza el colimador.

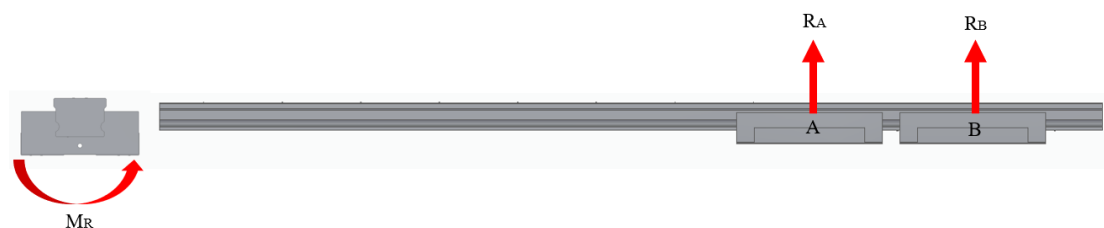


Figura 2.7: Reacciones que debe soportar el conjunto guía lineal y los carros sobre los cuales desliza el colimador.

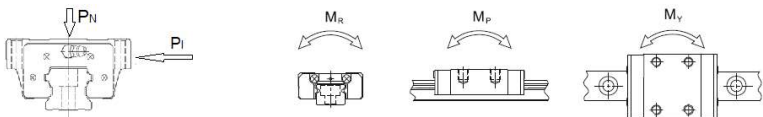
Estas reacciones se calcularon resolviendo el sistema de ecuaciones dado por las ecuaciones 2.20 (sumatoria de fuerzas), 2.21 (sumatoria de momentos respecto a A) y 2.22 (sumatoria de momentos respecto al eje de la guía). Se puede destacar que, siendo conservativos, para el cálculo de las reacciones normales se consideró la fuerza de inercia máxima como constante, es decir, se consideró la peor situación a la cual se encuentran sometidas las guías. Además se consideró aceleración positiva y negativa de igual magnitud. Cabe aclarar también que se consideró conservativamente que ambos carros soportan el total del momento torsor.

$$R_A + R_B = P \quad (2.20)$$

$$R_B \cdot (l_{xA} + l_{xB}) = P \cdot l_{xA} - m \cdot a \cdot l_z \quad (2.21)$$

$$M_R = P \cdot l_y \quad (2.22)$$

Con estos diagramas de reacciones de fuerza y momento que deben soportar las guías lineales se compararon las fuerzas máximas calculadas con los esfuerzos máximos admisibles según el fabricante [7] mostrados en la Figura 2.8. De esta se obtuvo más específicamente la capacidad de carga estática $C_0 = 5,88 \text{ kN}$, la capacidad de carga dinámica $C = 3,72 \text{ kN}$ y el momento torsor admisible $M_{0R} = 38,22 \text{ Nm}$ por las guías lineales HIWIN MGNR12H seleccionadas durante el diseño.



Modelo N°.	Dimensiones (mm)			Dimensiones del Patin (mm)								Dimensiones del Rail (mm)								Tornillo de Montaje (mm)	Capacidad de Carga Dinámica C(kN)	Capacidad de Carga Estática C ₀ (kN)	Máximo Momento Estático			Peso		
	H	H ₁	N	W	B	B ₁	C	L ₁	L	G	G ₀	Mx1	H ₂	W _R	H ₂	D	h	d	P				E	M _R	M _p	M _y	Patin	Rail
				N-m			N-m			N-m			kg	kg/m														
MGN 5C	6	1.5	3.5	12	8	2	-	9.6	16	-	0.8	M2x1.5	1	5	3.6	3.6	0.8	2.4	15	5	M2x6	0.54	0.84	2	1.3	1.3	0.008	0.15
MGN 7C	8	1.5	5	17	12	2.5	8	13.5	22.5	-	Ø1.2	M2x2.5	1.5	7	4.8	4.2	2.3	2.4	15	5	M2x6	0.98	1.24	4.70	2.84	2.84	0.010	0.22
MGN 7H																						1.37	1.96	7.64	4.80	4.80	0.015	
MGN 9C	10	2	5.5	20	15	2.5	10	18.9	28.9	-	Ø1.4	M3x3	1.8	9	6.5	6	3.5	3.5	20	7.5	M3x8	1.86	2.55	11.76	7.35	7.35	0.016	0.38
MGN 9H																						2.55	4.02	19.60	18.62	18.62	0.026	
MGN 12C	13	3	7.5	27	20	3.5	15	21.7	24.7	-	Ø2	M3x3.5	2.5	12	8	6	4.5	3.5	25	10	M3x8	2.84	2.02	26.48	12.72	12.72	0.034	0.65
MGN 12H																						3.72	5.88	38.22	36.26	36.26	0.054	
MGN 15C	16	4	8.5	32	25	3.5	20	26.7	42.1	4.5	M3	M3x4	3	15	10	6	4.5	3.5	40	15	M3x10	4.61	5.59	45.08	21.56	21.56	0.059	1.06
MGN 15H																						6.37	9.11	73.50	57.82	57.82	0.092	

Nota : 1 kgf = 9.81 N

Figura 2.8: Propiedades de las guías lineales utilizadas.

Teniendo la fuerza normal máxima R_{max} y el momento torsor M_R que debe soportar cada guía se calculó el factor de seguridad estático para cargas simples siguiendo la ecuación 2.23 y el factor de seguridad para un momento siguiendo la ecuación 2.24 para ambas guías.

$$f_{SL} = \frac{C_0}{R_{max}} \quad (2.23)$$

$$f_{SM} = \frac{M_{0R}}{M_R} \quad (2.24)$$

Para finalizar con la verificación de las guías lineales se calculó la vida de servicio de cada guía siguiendo la ecuación 2.25 donde el factor de carga $f_w = 1,5$ según la tabla que se expone en la Figura 2.9.

$$l_h = \left(\frac{C}{f_w \cdot R_{max}} \right)^3 \cdot 50km \cdot \frac{1}{2 \cdot L \cdot 1Hz} \quad (2.25)$$

Serie MG		
Condiciones	Velocidad	f_w
Sin impactos ni vibraciones	$V \leq 15$ m/min	1 ~ 1.5
Carga normal	$15 \text{ m/min} < V \leq 60$ m/min	1.5 ~ 2.0
Con impactos y vibraciones	$V > 60$ m/min	2.0 ~ 3.5

Figura 2.9: Factor de carga para el cálculo de la vida en servicio de cada guía.

2.2.3. Eje pivote y rodamientos

Para la verificación del eje pivote y sus rodamientos se utilizó nuevamente el modelo 3D del prototipo colimador oscilante desarrollado en SolidEdge en su versión estudiantil [9]. De este diseño se obtuvo la masa del conjunto rotante, y la posición de su centro de masa. Con esto se midió la distancia entre el centro de masa y el eje del pivote en la sección más solicitada. Luego, con lo obtenido se calculó el momento flector que debe soportar el eje en esta sección y así las tensiones máximas, afectadas por la concentración de tensiones causadas por el alojamiento del anillo de retención. Con estas tensiones máximas se consideró una tensión alterna y una tensión media debido a la variación de la fuerza de inercia, y se procedió a calcular el factor de seguridad para vida infinita siguiendo la bibliografía [6].

En la Figura 2.10 se muestran las medidas realizadas en el diseño 3D. Se consideró que la sección más solicitada se ubica en el espacio entre el bastidor y la base del conjunto rotante donde también se encuentra el anillo de retención.

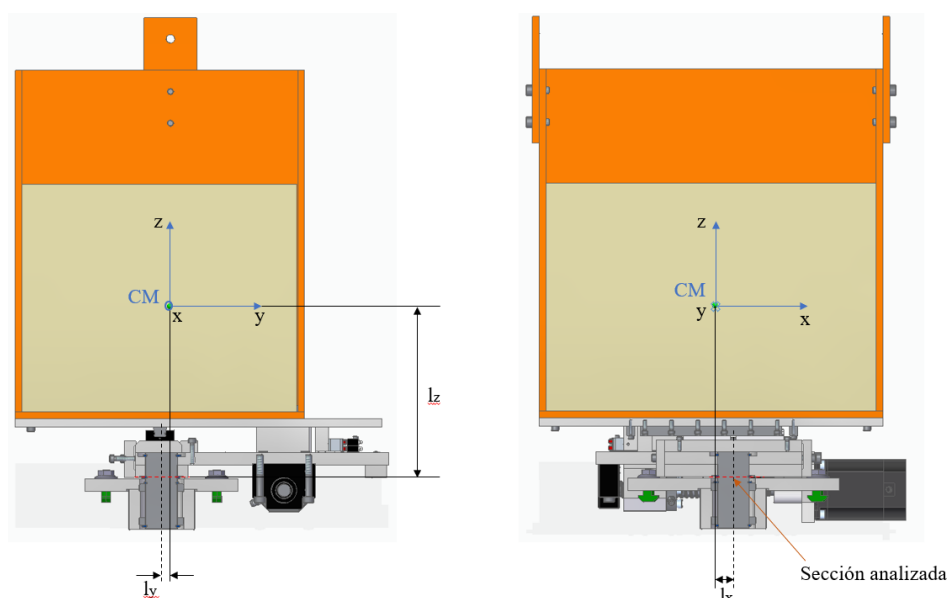


Figura 2.10: Distancias medidas entre el centro de masa del conjunto rotante y la sección analizada del eje pivote.

Una vez medidas estas distancias se determinó la sección más solicitada analizando los diagramas de corte y momento esquemáticos que se muestran en la Figura 2.11 y

considerando el concentrador de tensiones más cercano.

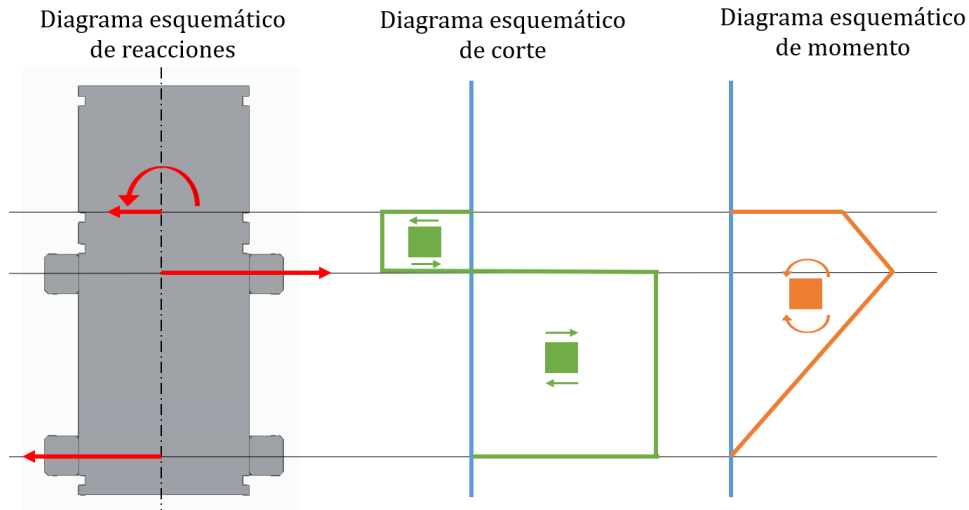


Figura 2.11: Diagramas esquemáticos de corte y momento flector del eje pivote.

Luego se trasladaron los esfuerzos a la sección más solicitada como una combinación entre fuerzas y momentos. Esto se expone mejor en el croquis de la Figura 2.12. Cabe aclarar que, siguiendo la bibliografía [6], la fuerza F_x no se la considera en el cálculo de verificación del eje.

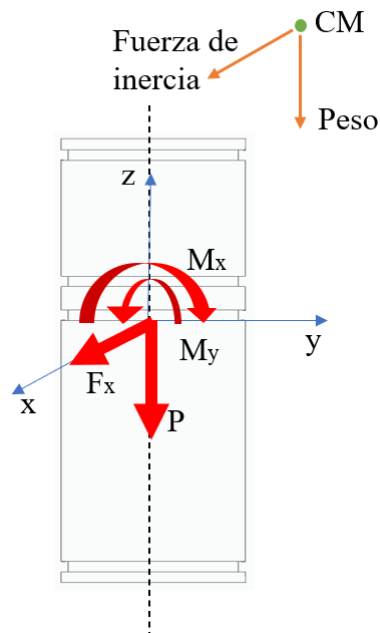


Figura 2.12: Croquis del eje pivote con los esfuerzos trasladados a la sección analizada.

Mediante las ecuaciones 2.26 se obtuvo el momento flector M_x . Como ya se mencionó anteriormente, debido a que la fuerza inercial es cíclica se la consideró como una fuerza alternante con una amplitud igual a la fuerza inercial máxima obtenida. Así, mediante la ecuación 2.27 se calculó el momento flector $M_{y,max}$ máximo y mediante la

ecuación 2.28 el mínimo ($M_{y,min}$). De estas ecuaciones se debe resaltar que la magnitud de estos momentos no es igual ya que se consideraron los brazos de palanca en los desplazamientos máximos ($l_{x,max}$ y $l_{x,min}$) del colimador y estos no son simétricos con el eje pivote.

$$M_x = P \cdot l_y \quad (2.26)$$

$$M_{y,max} = P \cdot l_{x,max} + m \cdot a \cdot lz \quad (2.27)$$

$$M_{y,min} = -(P \cdot l_{x,min}) - m \cdot a \cdot lz \quad (2.28)$$

Con estos momentos calculados se calcularon los momentos netos máximos y mínimos mediante las ecuaciones 2.29 y 2.30 respectivamente.

$$M_{neto,max} = \sqrt{M_x^2 + M_{y,max}^2} \quad (2.29)$$

$$M_{neto,min} = \sqrt{M_x^2 + M_{y,min}^2} \quad (2.30)$$

Con estos momentos se calculó luego el momento neto medio y el momento neto alterno siguiendo las ecuaciones 2.31 y 2.32 respectivamente.

$$M_{neto,medio} = \frac{M_{neto,max} - M_{neto,min}}{2} \quad (2.31)$$

$$M_{neto,alterno} = M_{neto,max} - M_{neto,medio} \quad (2.32)$$

Considerando las características del eje pivote se determinó su tensión última y su factor de concentración de tensiones en la sección analizada. Siendo el material del eje pivote un acero 1020 se tiene del catálogo [10] que este tiene una tensión última de $S_{ut} = 380 \text{ MPa}$. Además, del diseño 3D se obtuvieron las características dimensionales del alojamiento para el anillo de retención. Sobre este se tiene entonces que cuenta con una profundidad de $t_{anillo} = 0,7 \text{ mm}$, un espesor de $a_{anillo} = 1,6 \text{ mm}$ y un redondeo en sus vértices internos de $r_{anillo} = 0,1 \text{ mm}$. También se tiene con esto que siendo el diámetro mayor del eje de $D = 30 \text{ mm}$ queda un diámetro menor de la sección más solicitada de $d = 28,6 \text{ mm}$. Con estas dimensiones se obtienen los factores $\frac{a_{anillo}}{t_{anillo}} = 2,29$ y $\frac{r_{anillo}}{t_{anillo}} = 0,14$. Con estos factores y siguiendo la bibliografía [11] se obtuvo de la figura A-15-16 de la misma el factor de concentración de tensiones $k_t = 4,5$. La figura mencionada se expone en la Figura 2.13.

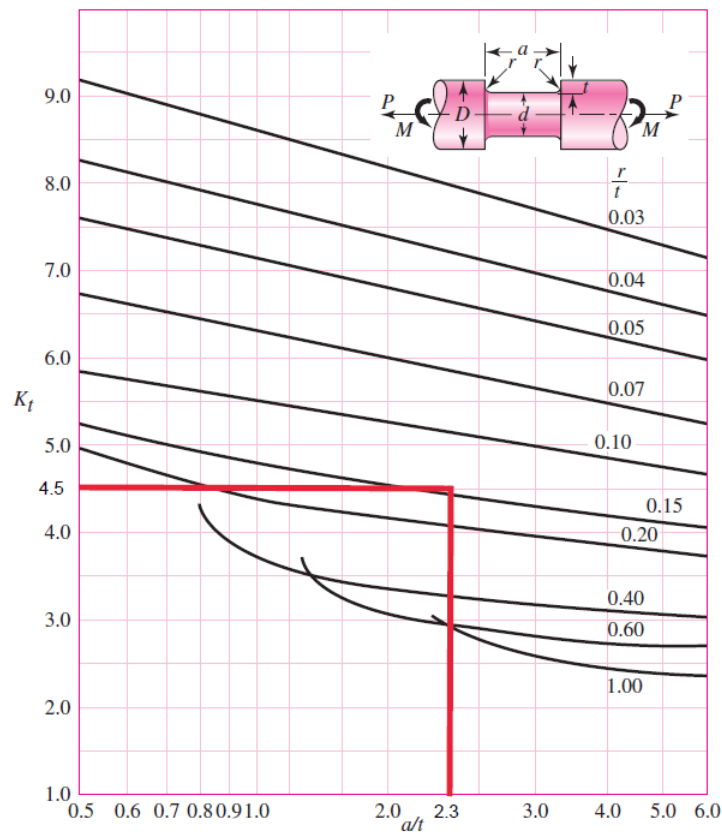


Figura 2.13: Factor de concentración de tensiones para un eje acanalado sometido a flexión y/o tensión.

Dado que el material del eje es dúctil se corrigió este factor de concentración de tensiones con la sensibilidad del material a la muesca. Esta sensibilidad se la calculó mediante la ecuación 2.33 donde a_{kt} viene dado por la ecuación 2.34 con S_{ut} en [kpsi] y a_{kt} en [pulg^{0,5}]. Luego, mediante la ecuación 2.35 se obtuvo el factor de concentración de tensiones corregido k_f .

$$q = \frac{1}{1 + \sqrt{\frac{a_{kt}}{r}}} \quad (2.33)$$

$$a_{kt} = 0,246 - 3,08 \cdot 10^{-3} \cdot S_{ut} + 1,51 \cdot 10^{-5} \cdot S_{ut}^2 - 2,67 \cdot 10^{-8} \cdot S_{ut}^3 \quad (2.34)$$

$$k_f = 1 + q \cdot (k_t - 1) \quad (2.35)$$

Con este factor de concentración de tensiones corregido obtenido se calculó mediante las ecuaciones 2.36 y 2.37 la tensión media y alterna de Von Mises siguiendo el capítulo 6 de la bibliografía [6].

$$\sigma'_m = \sqrt{\left(k_f \cdot \frac{32 \cdot M_{neto,medio}}{\pi \cdot d^3} - k_f \cdot \frac{4 \cdot P}{\pi \cdot d^2}\right)^2} \quad (2.36)$$

$$\sigma'_a = k_f \cdot \frac{32 \cdot M_{neto,alternativo}}{\pi \cdot d^3} \quad (2.37)$$

Para continuar con la verificación del eje pivote a fatiga se procedió a calcular el límite de resistencia a la fatiga corregida. Para esto se calculó primero el límite de resistencia a la fatiga para vida infinita como $S'_e = 0,5 \cdot S_{ut}$. Luego, aplicando los factores de corrección mostrados en la Tabla 2.1 se obtuvo el límite de resistencia a la fatiga corregido para vida infinita mediante la ecuación 2.38.

Factor de corrección	Valor	Justificación
C_{carga}	1	Por tratarse de una carga de flexión
$C_{tamaño}$	0,86	para $8mm < d \leq 250mm$ entonces $C_{tamaño} = 1,189 \cdot d^{-0,097}$
C_{sup}	0,98	$C_{sup} = A \cdot S_{ut}^b$ con $A = 4,51$ y $b = -0,265$ por ser maquinado
C_{temp}	1	Por trabajar a menos de $450^\circ C$
C_{conf}	0,814	Para tener una confiabilidad del 99 %

Tabla 2.1: Coeficientes de corrección para el cálculo de la resistencia a la fatiga corregida.

$$S_f = C_{carga} \cdot C_{tamaño} \cdot C_{sup} \cdot C_{temp} \cdot C_{conf} \cdot S'_e \quad (2.38)$$

Para finalizar la verificación del eje pivote, con la resistencia a la fatiga corregida para vida infinita y considerando la línea modificada de Goodman se calculó el factor de seguridad mediante la ecuación 2.39.

$$N_f = \frac{1}{\frac{\sigma'_a}{S_f} + \frac{\sigma'_m}{S_{ut}}} \quad (2.39)$$

Para la verificación de los rodamientos se trasladaron las fuerzas y los momentos a la sección del eje equidistante a ambos rodamientos y se calcularon las reacciones radiales que estos deben soportar. Además, conservativamente se consideró que ambos rodamientos deben soportar toda la carga axial dada por el peso del conjunto rotante. Se verificaron los rodamientos a carga estática ya que estos no realizan rotaciones completas y la frecuencia es menor a 10 Hz.

En la Figura 2.14 se observan las fuerzas trasladadas a la distancia medida entre los rodamientos. De este traslado de fuerzas se obtuvo $M'_x = M_x$, $M'_y = M_{y,max} + m \cdot a \cdot d$ y $F'_x = m \cdot a$.

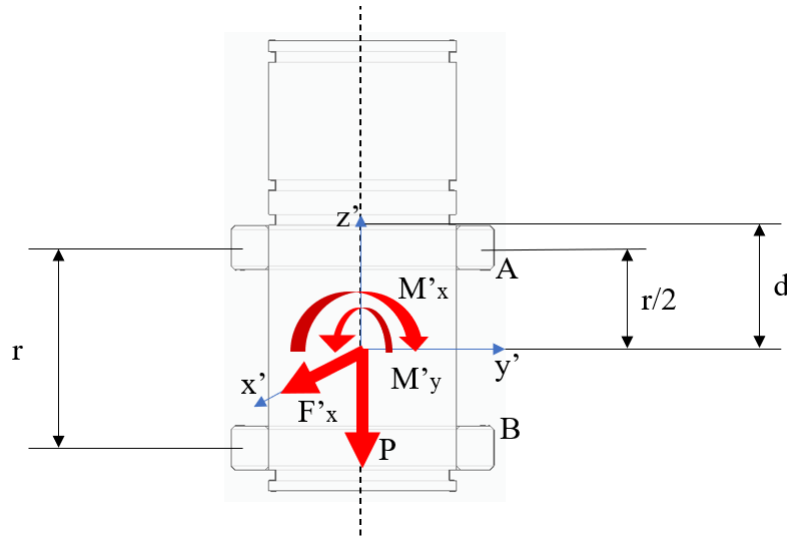


Figura 2.14: Esfuerzos trasladados a la sección media entre los rodamientos.

Mediante las ecuaciones 2.40, 2.41 y 2.42 se calcularon las componentes de las reacciones de cada rodamiento siendo $R_{By} = -R_{Ay}$. Además, como ya se mencionó, al considerar que cada rodamiento debe soportar la totalidad de la carga axial se tiene que $R_{Az} = R_{Bz} = P$.

$$R_{Ax} = \left(\frac{M'_y}{2 \cdot \frac{r}{2}} + \frac{F'_x}{2} \right) \quad (2.40)$$

$$R_{Ay} = \frac{M'_x}{2 \cdot \frac{r}{2}} \quad (2.41)$$

$$R_{Bx} = \left(\frac{-M'_y}{2 \cdot \frac{r}{2}} + \frac{F'_x}{2} \right) \quad (2.42)$$

Teniendo las componentes de las reacciones de cada rodamiento se calcularon mediante las ecuaciones 2.43 y 2.44 las reacciones radiales totales que debe soportar cada rodamiento.

$$R_A = \sqrt{R_{Ax}^2 + R_{Ay}^2} \quad (2.43)$$

$$R_B = \sqrt{R_{Bx}^2 + R_{By}^2} \quad (2.44)$$

Del catálogo [12] se obtuvieron las características de los rodamientos seleccionados. Estos son rodamientos de bolas SKF 61806 - 2RS1 con capacidad de carga estática $C_0 = 2900 \text{ N}$. Se verificará únicamente el rodamiento más solicitado, ya que así se verifica automáticamente el otro. Siguiendo este criterio y como $0,6 \cdot R_A + 0,5 \cdot R_{Az} < R_A$ se consideró la carga estática equivalente $P_0 = R_A$. Entonces, mediante la ecuación 2.45 se obtuvo el factor de seguridad del rodamiento más solicitado.

$$s_0 = C_0/P_0 \quad (2.45)$$

2.3. Resultados

2.3.1. Tornillos de husillos de bola

De los requerimientos detallados en 1.2 se tiene que la longitud de desplazamiento del colimador debe ser $L = 30 \text{ mm}$. Para el actuador inercial se consideró la misma longitud ya que para mayor longitud el motor del actuador perdía pasos debido a distintos problemas ocurridos. Para más detalles de estos se puede ver el Capítulo 6. Entonces, siguiendo la metodología desarrollada en 2.2.1 y considerando un tiempo de aceleración del 2% del período ($ta_{\%T} = 0,02 \text{ s}$) se obtuvieron los siguientes parámetros dinámicos de la componente lineal del movimiento del colimador y del actuador inercial.

- $V_{max} = 0,0625 \frac{m}{s}$
- $a = 3,125 \frac{m}{s^2}$
- $N_{max} = 750 \text{ RPM}$
- $\alpha = 3927 \frac{1}{s^2}$

Del modelo 3D se obtuvo una longitud del tornillo del colimador de $l_a(\text{colimador}) = 104,4 \text{ mm}$ y una longitud del tornillo del actuador inercial de $l_a(\text{actuador}) = 239,5 \text{ mm}$. Luego, siguiendo las ecuaciones 2.6 y 2.7 se obtuvo para el tornillo del colimador una carga máxima admisible por pandeo de $P_1(\text{colimador}) = 247,7 \text{ kN}$. Análogamente se obtuvo para el tornillo del actuador inercial una carga máxima admisible por pandeo de $P_1(\text{actuador}) = 47 \text{ kN}$. Además se obtuvo para ambos tornillos una carga máxima admisible por tracción/compresión de $P_2 = 19 \text{ kN}$.

Prosiguiendo se obtuvo del modelo 3D que el conjunto móvil del colimador tiene una masa de $m(\text{colimador}) = 53,4 \text{ kg}$, mientras que la masa del actuador inercial es de $m(\text{actuador}) = 20 \text{ kg}$. Entonces con estas masas se calcularon las fuerzas axiales que debe soportar el tornillo durante un ciclo. Los resultados para estas fuerzas obtenidos se muestran en la Tabla 2.2.

Conjunto móvil	Fa_1	Fa_2	Fa_3	Fa_4	Fa_5	Fa_6
Colimador	173,5 N	6,6 N	-160,3 N	-173,5 N	-6,6 N	160,3 N
Actuador	67,5 N	5 N	-57,5 N	-67,5 N	-5 N	57,5 N

Tabla 2.2: Fuerzas axiales que deben soportar los tornillos durante un ciclo de movimiento.

Con estos valores obtenidos para las fuerzas axiales que debe soportar el tornillo se obtuvieron mediante las ecuaciones 2.15 y 2.14 los factores de seguridad ante el pandeo y la tracción/compresión expuestos en la Tabla 2.3. Se puede observar que estos factores de seguridad son muy altos, particularmente mucho mayores a los especificados por el fabricante en el catálogo [5] expuestos en la Figura 2.15.

Conjunto móvil	$FS_{trac/comp}$	FS_{pandeo}
Colimador	109	1427
Actuador	281	697

Tabla 2.3: Factores de seguridad de los tornillos de husillo de bolas obtenidos ante carga de tracción/compresión y ante pandeo para el conjunto móvil del colimador y el actuador inercial.

Máquina que utiliza el sistema LM	Condiciones de carga	Límite más bajo de f_s
Maquinaria industrial general	Sin vibración ni impacto	1,0 a 3,5
	Con vibración o impacto	2,0 a 5,0
Máquina-herramienta	Sin vibración ni impacto	1,0 a 4,0
	Con vibración o impacto	2,5 a 7,0

Figura 2.15: Factores de seguridad recomendados por THK-TBR para los tornillos de husillos de bolas recirculantes.

Con respecto al torque máximo que deben entregar los motores para la dinámica requerida de cada conjunto móvil se obtuvo un torque máximo para el motor del colimador de $T_{max}(colimador) = 0,2 Nm$ y para el motor del actuador de $T_{max}(actuador) = 0,11 Nm$.

Para finalizar se obtuvo una rigidez del tornillo del husillo de bolas del colimador de $K_{total}(colimador) = 396 \frac{MN}{m}$ y una rigidez del tornillo del husillo de bolas del actuador de $K_{total}(actuador) = 111 \frac{MN}{m}$.

2.3.2. Guías lineales

Siguiendo lo desarrollado en 2.2.2 y nuevamente basándose en el diseño 3D se obtuvieron las distancias entre los centros de gravedad de cada conjunto móvil y los centros de los carros y ejes de las guías como muestran las Figuras 2.4 y 2.5. Los resultados de estas mediciones para cada conjunto móvil se exponen en la Tabla 2.4.

Conjunto móvil	l_{xA}	l_{xB}	l_y	l_z
Colimador	3,4 mm	46,6 mm	2,7 mm	133,4 mm
Actuador	41,8 mm	10,2 mm	36,2 mm	69,7 mm

Tabla 2.4: Distancias medidas entre los centros de gravedad de los conjuntos móviles y los ejes de las guías lineales como así también los centros de los carros.

Luego, considerando las aceleraciones y masas de cada conjunto móvil obtenidas en la sección 2.2.1 se obtuvieron las reacciones que debe soportar cada carro de las guías lineales. Los resultados de estas reacciones se muestran en la Tabla 2.5 donde las reacciones primadas corresponden a las obtenidas con una aceleración negativa.

Conjunto móvil	R_A	R'_A	R_B	R'_B	M_R
Colimador	933,3 N	42,8 N	-409,6 N	480,8 N	1,41 Nm
Actuador	349,5 N	16 N	-153,4 N	180 N	0,53 Nm

Tabla 2.5: Reacciones obtenidas que deben soportar los carros de las guías lineales de cada conjunto móvil

Considerando entonces la reacción máxima para cada guía y el momento torsor obtenido se obtuvieron los factores de seguridad estáticos y al momento torsor mostrados en la Tabla 2.6. Se puede observar que los factores de seguridad obtenidos se encuentran todos por encima de los valores recomendados por el fabricante, los cuales se pueden observar en la Figura 2.16 obtenida del catálogo [7].

Conjunto móvil	f_{SL}	f_{SM}
Colimador	6,3	27
Actuador	16	72

Tabla 2.6: Factores de seguridad de las guías lineales ante carga y momento obtenidos para el conjunto móvil del colimador y el actuador inercial.

Condiciones de Carga	f_{SL}, f_{SM} (Min.)
Carga Normal	1.0-3.0
Con Impactos / Vibraciones	3.0-5.0

Figura 2.16: Valores recomendados por HIWIN para los factores de seguridad estáticos de las guías lineales.

Para finalizar se obtuvo una vida de servicio de la guía del colimador de $l_h(\text{colimador}) = 4343$ hs y de la guía lineal del actuador inercial de $l_h(\text{actuador}) = 24801$ hs. Cabe aclarar que al tratarse de un prototipo, este no tiene requerimientos de vida útil, sin embargo se consideró beneficioso obtener estos parámetros.

2.3.3. Eje pivote y rodamientos

Siguiendo lo desarrollado en la sección 2.2.3 y nuevamente basándose en el diseño 3D se midieron las distancias entre el centro de gravedad del conjunto rotante y la sección analizada como muestra la Figuras 2.10. Además también se obtuvo la masa del conjunto rotante siendo esta $m(\text{conjunto rotante}) = 57,5 \text{ kg}$. Los resultados obtenidos en estas mediciones son los siguientes:

- $l_{x,max} = 19,8 \text{ mm}$
- $l_{x,min} = 17,4 \text{ mm}$
- $l_y = 7,7 \text{ mm}$
- $l_z = 168,2 \text{ mm}$

Con estas distancias medidas se obtuvieron los siguientes resultados de los momentos trasladados al centro de la sección analizada:

- $M_x = 4,3 \text{ Nm}$
- $M_{y,max} = 39,3 \text{ Nm}$
- $M_{y,min} = -37,9 \text{ Nm}$

Con estos resultados se obtuvo entonces un momento flector neto máximo de $M_{neto,max} = 39,5 \text{ Nm}$ y un momento flector neto mínimo de $M_{neto,min} = 38,1 \text{ Nm}$. Esto a su vez dio como resultado un momento flector neto medio de $M_{neto,medio} = 0,7 \text{ Nm}$ y un momento flector neto alterno de $M_{neto,alterno} = 38,8 \text{ Nm}$.

Luego, conociendo el momento flector medio y alterno que debe soportar el eje pivote se calcularon las tensiones máximas producidas en la sección analizada sobre este. Así, de las ecuaciones 2.36 y 2.37 se obtuvo la tensión media $\sigma'_m = 1,3 \text{ MPa}$ y la tensión alterna $\sigma'_a = 37,5 \text{ MPa}$.

Para finalizar con la verificación del eje pivote y utilizando los factores de corrección analizados en la Tabla 2.1 se obtuvo un límite de resistencia a la fatiga corregido para una vida infinita de $S_f = 129,9 \text{ MPa}$. Este límite permitió obtener mediante la línea de Goodman modificada un coeficiente de seguridad del eje pivote ante la fatiga a vida infinita de $N_f = 3,4$. Se observa que este factor de seguridad es mayor al límite de 3,1 establecido mediante el criterio de Pugsley. Sin embargo, al estar cercano al límite se analizó adicionalmente con mayor profundidad el diseño y se concluyó que reemplazando los dos anillos de retención intermedios con sus respectivos alojamientos por un anillo separador se evita tener un concentrador de tensiones en la sección más solicitada y se llega a tener un factor de seguridad a la fatiga a vida infinita de 7,6.

Por otro lado, durante la verificación de los rodamientos con las fuerzas y los momentos calculados anteriormente se obtuvieron los momentos y las fuerzas trasladadas a la sección del eje equidistante a ambos rodamientos. Estas cargas son $M'_x = 4,3 Nm$, $M'_y = 42,5 Nm$, $F'_x = m \cdot a = 167 N$ y $P = 563,9 N$.

Una vez calculados estos esfuerzos se obtuvieron las componentes de las reacciones que deben soportar los rodamientos. Estas resultaron tener los siguientes valores:

- $R_{Ax} = 1413 N$
- $R_{Ay} = -R_{By} = 135,9 N$
- $R_{Bx} = -1246 N$
- $R_{Az} = R_{Bz} = 563,9 N$

Con estos valores se obtuvieron las reacciones radiales netas que debe soportar cada rodamiento y sus valores son $R_A = 1419,5 N$ y $R_B = 1253,3 N$. Con estas fuerzas se obtuvo un coeficiente de seguridad para el rodamiento más solicitado de $s_0 = 2$ lo cual se encuentra dentro de los factores de seguridad recomendados por el fabricante según la tabla expuesta en la Figura 2.17 obtenida de [12].

Valores orientativos para el factor de seguridad estática s_0 , para cargas continuas y/u ocasionales, rodamientos de bolas				
Certeza del nivel de carga	Movimiento continuo			Movimiento poco frecuente
	Aceptación de deformación permanente			
	Sí	Algunas	No	Aceptación de deformación permanente
	Sí			Sí
Certeza alta Por ejemplo, carga por gravedad y sin vibración.	0,5	1	2	0,4
Certeza baja Por ejemplo, pico de carga.	$\geq 1,5$	$\geq 1,5$	≥ 2	≥ 1

Figura 2.17: Factores de seguridad recomendados para rodillos de bolas en distintas situaciones de carga.

2.4. Conclusión

En este capítulo se verificaron analíticamente los componentes críticos del prototipo colimador oscilante para una dinámica de movimiento con perfil de velocidades trapezoidal, un período de 1 s, un tiempo de aceleración del 2% del período y una longitud de desplazamiento tanto del colimador como del actuador de 30mm. Los componentes críticos verificados son el tornillo del husillo de bolas del colimador, el tornillo

del husillo de bolas del actuador inercial, la guía lineal del colimador, la guía lineal del actuador, el eje pivote y sus rodamientos. En todos los casos se utilizó el diseño 3D desarrollado en el programa SolidEdge para medir las diferentes distancias y obtener las masas de los distintos conjuntos. Para calcular los factores de seguridad se siguieron los distintos catálogos de fabricantes de los elementos comerciales. Además, se calcularon los esfuerzos a los que están sometidos los elementos a verificar mediante mecánica de sólidos utilizando diagramas de cuerpo libre y traslación de fuerzas y momentos.

Se pudo verificar que todos los factores de seguridad obtenidos se encontraron por encima de los límites especificados por los fabricantes. Para el caso del eje pivote se obtuvo un factor de seguridad de 3,4, el cual es mayor al límite de 3,1 establecido mediante el criterio de Pugsley. Sin embargo, al estar cercano al límite se analizó adicionalmente con mayor profundidad el diseño y se concluyó que reemplazando los dos anillos de retención intermedios con sus respectivos alojamientos por un anillo separador se evita tener un concentrador de tensiones en la sección más solicitada y se llega a tener un factor de seguridad a la fatiga a vida infinita de 7,6. Con esta conclusión se recomienda entonces, en caso de volver a fabricar el eje, reemplazar los anillos de retención intermedios por un anillo separador.

Capítulo 3

Diseño del banco de pruebas

“Todo es difícil antes de hacerse fácil”

— Johann Wolfgang von Goethe

3.1. Introducción

Cómo ya se mencionó en el Capítulo 1, en el difractor multipropósito ANDES correspondiente al LAHN, el colimador oscilante es un componente funcional fundamental. Este se encuentra ubicado en una estructura que es capaz de ubicarse angularmente alrededor de la muestra. También se destacó que esta estructura es compartida con el detector, un dispositivo electrónico sensible a las vibraciones y, por esta razón, se buscan minimizar las vibraciones que el colimador pueda generar en esta estructura compartida. En la Figura 3.1 se muestra la estructura mencionada en la cual se colocará finalmente el colimador oscilante.

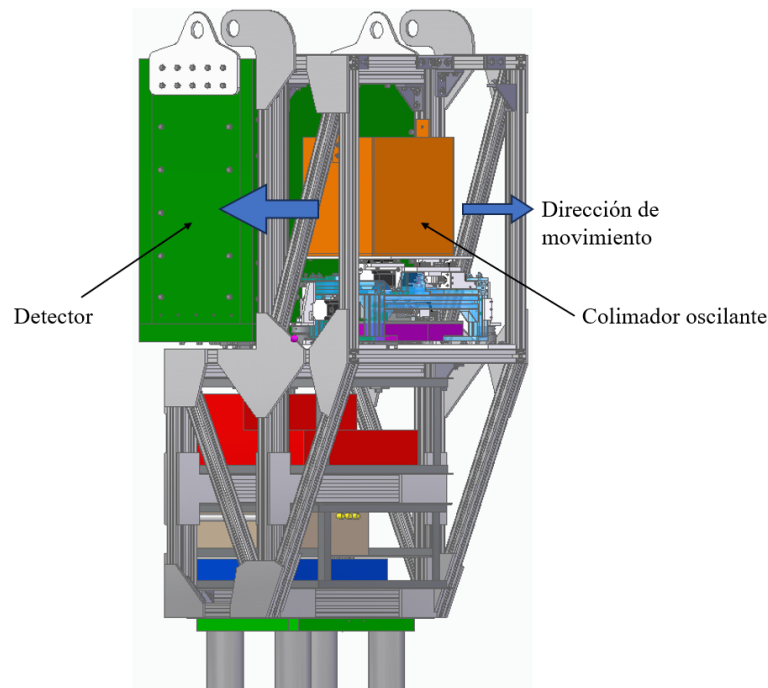


Figura 3.1: Estructura soporte del colimador oscilante y el detector del difractor ANDES.

El objetivo del banco de pruebas para este trabajo es proporcionar un soporte para el prototipo colimador oscilante que permita medir los desplazamientos o aceleraciones que las oscilaciones generan. Estas mediciones se utilizarán luego para un posterior análisis del funcionamiento del sistema de amortiguación y como retroalimentación para el control del mismo.

3.2. Diseño

Como se puede observar en la Figura 3.1 la estructura soporte del colimador en el ANDES se compone principalmente de perfiles extrudados de aluminio y placas de aluminio de refuerzo. Además se distingue que la misma se encuentra muy rigidizada mediante diagonales y refuerzos en la dirección transversal a la dirección de desplazamiento principal del colimador. Sin embargo, se encuentra menos rigidizada en la dirección del movimiento principal.

Para simular mejor el comportamiento de la estructura soporte se decidió fabricar el banco de ensayos utilizando perfiles extrudados de aluminio T-slot 40x40. Con el objetivo de ser conservativos en cuanto a la estimación de la rigidez del bastidor real, y desconociendo el valor real de este, la base del diseño del banco fue considerar una rigidez en el sentido del movimiento tal que sea igual a la rigidez que tendría todo el conjunto mostrado en la Figura 3.1 con una frecuencia natural de 5Hz. Esta frecuencia fue un requerimiento de entrada para este trabajo.

Se comenzó diseñando una plataforma con las mismas dimensiones que la plataforma final a la cual se anclará el colimador. Esta plataforma será soportada por cuatro columnas con las cuales se definió, mediante su longitud, la constante de rigidez del banco de pruebas deseada. Se buscó rigidizar mediante diagonales la estructura en dirección transversal tal como sucede en la estructura soporte final, y para evitar ruido de oscilaciones en otras direcciones que no sean las del movimiento principal. Además, se reforzaron todos los vértices para simular mejor la condición de doble empotramiento de las columnas. La base y la plataforma se la diseñó simétricamente y considerando el sistema de fijación del colimador oscilante.

Del diseño 3D de la estructura soporte del colimador oscilante y el detector del difractor ANDES se obtuvo la masa de todo este conjunto sin considerar el prototipo. Esta medición dio como resultado una masa de $m_{estructura-ANDES} = 415 \text{ kg}$. Con esta masa y considerando la frecuencia natural asumida para esta estructura de $f_n = 5 \text{ Hz}$ se obtuvo mediante la ecuación 3.1 la rigidez buscada que debe tener el banco de pruebas en la dirección del movimiento principal.

$$K = (2 \cdot \pi \cdot f_n)^2 \cdot m_{estructura-ANDES} = 409,6 \frac{kN}{m} \quad (3.1)$$

Para lograr que el banco de ensayos diseñado tenga esta rigidez en el sentido del movimiento principal se ajustó el largo de sus columnas mediante la ecuación 3.2 basándose en la rigidez equivalente mostrada en la Figura 3.2 obtenida de [13]. Por otro lado, del catálogo [14] se obtuvo el momento de inercia de los perfiles de aluminio $I_{T-slot} = 9,1 \text{ cm}^4$ y el módulo de elasticidad $E_{T-slot} = 70000 \frac{N}{mm^2}$. Con lo anterior se obtuvo entonces un largo de las columnas del banco de pruebas de $L_{columnas} = 907 \text{ mm}$. Cabe aclarar que esta longitud corresponde a la longitud de las columnas para la cual no tienen contacto con ningún refuerzo en el sentido del movimiento principal. Esta longitud se muestra en la Figura 3.3.

$$L_{columnas} = \sqrt[3]{\frac{12 \cdot E_{T-slot} \cdot I_s}{K}} \text{ con } I_s = 4 \cdot I_{T-slot} \quad (3.2)$$

Para finalizar el diseño se verificaron las columnas al pandeo. Para esto se calculó la carga crítica de pandeo para una columna con la ecuación 3.3. Con esta carga crítica y considerando que cada columna soportará un cuarto del peso del prototipo colimador oscilante junto a la plataforma del banco de pruebas, de masa total suspendida $m_{suspendida} = 109,8 \text{ kg}$, se obtuvo un coeficiente de seguridad al pandeo de $FS_{pandeo} = \frac{P_{crit}}{m_{suspendida} \cdot g} = 70$. Al ser este coeficiente mucho mayor a 3,1 se tiene por el criterio de Pugsley que el banco verifica al pandeo.

$$P_{crit} = \frac{\pi^2 \cdot E \cdot I}{4 \cdot L_{columnas}^2} = 19100 \text{ N} \quad (3.3)$$

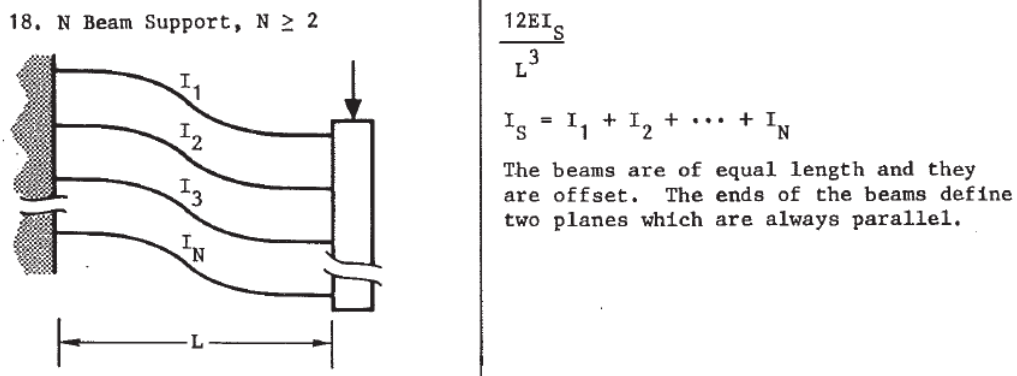


Figura 3.2: Constante elástica equivalente para N vigas empotradas en ambos extremos.

Con todo el desarrollo anterior se obtuvo el banco de pruebas que se muestra en la Figura 3.3 con sus dimensiones principales. En el Anexo B se exponen los planos de este mismo con los cuales se fabricó finalmente.

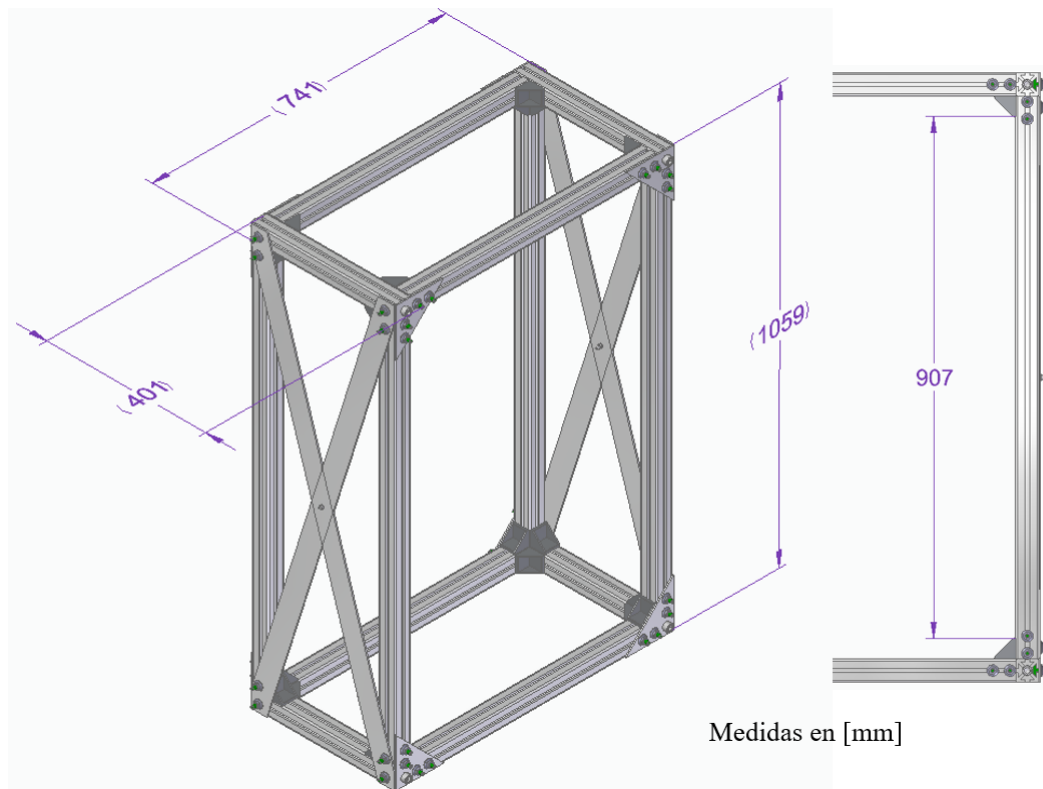


Figura 3.3: Banco de pruebas diseñado con sus medidas principales.

Dado que al momento de contar con el banco de pruebas y gran parte del prototipo colimador oscilante fabricado no se contaba con los Kinetic Coupling con los cuales se fijará el colimador oscilante al bastidor final, se diseñaron placas de acoplamiento entre el prototipo y el banco de pruebas. En la Figura 3.4 se observa el modelo 3D del banco de pruebas con el prototipo colimador oscilante montado.

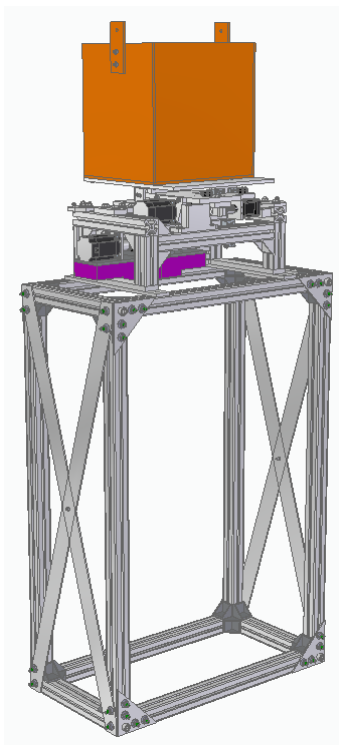


Figura 3.4: Banco de pruebas diseñado con el prototipo colimador oscilante montado.

3.3. Revisión del diseño

Luego de observar que la rigidez medida experimentalmente como se detalla en el Capítulo 6 no coincide con la cual se diseñó el banco de pruebas se decidió analizar el banco con el programa de elementos finitos MEFI [15].

Para esto se modeló el banco de pruebas como un pórtico simplemente apoyado en los vértices de la base y con el doble de área y momento de inercia en las barras que las barras de aluminio. Este modelo se expone en la Figura 3.5. A este modelo se le aplicó una fuerza lateral de 1 N en la plataforma y se obtuvo un desplazamiento lateral de esta de $5,6 \mu m$. De esta manera se obtuvo fácilmente que la rigidez del banco teórica corregida es $K_{corregida} = \frac{1N}{5,6 \mu m} = 178,6 \frac{kN}{m}$. Cabe aclarar que para este modelo se consideró como longitud de las barras la distancia entre los nodos centrales del banco de pruebas.

Dada la rigidez corregida obtenida se considerará de acá en adelante esta rigidez como la rigidez teórica del banco de pruebas diseñado. Con esta rigidez se tiene, invirtiendo la ecuación 3.1, que la frecuencia natural del soporte final del prototipo colimador oscilante sería de 3,3 Hz. Se observa que, considerando la rigidez corregida, se mantiene aproximadamente semejante la frecuencia natural del banco de pruebas respecto del soporte final. Como ventaja se tiene que se obtendrán mayores desplazamientos, lo cual permitirá poder tomar mejores mediciones. Además esta frecuencia es todavía mayor a la frecuencia de trabajo, lo cual es deseable para que la estructura no

entre en resonancia.

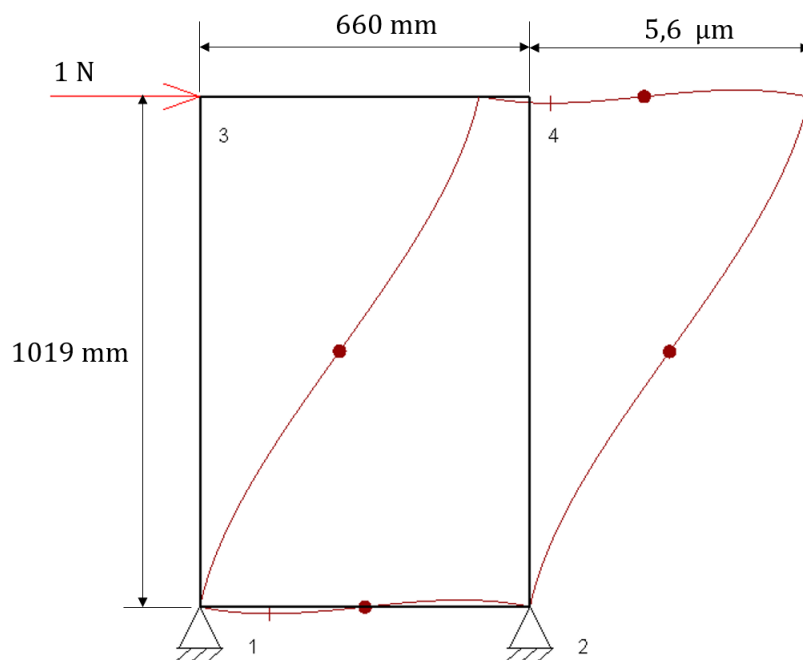


Figura 3.5: Deflexión del banco de pruebas como pórtico empotrado en el suelo.

Capítulo 4

Modelado dinámico del conjunto prototipo más banco de pruebas

“Nunca se puede predecir un acontecimiento físico con una precisión absoluta.”

— Max Planck

4.1. Introducción

El modelado dinámico de una estructura es fundamental en el campo de la ingeniería. Esta disciplina se encarga de comprender y predecir el comportamiento de una estructura en movimiento o sometida a cargas dinámicas. Para esto se utiliza un enfoque analítico que se basa en la aplicación de principios físicos y matemáticos para describir y simular el movimiento y las respuestas vibratorias de las estructuras.

El proceso de modelado dinámico implica la construcción de un modelo matemático que represente fielmente las propiedades físicas y el comportamiento dinámico de la estructura. Esto implica definir las características geométricas, las propiedades de los materiales y las condiciones de contorno que influyen en el comportamiento dinámico de la estructura.

Existen diferentes enfoques para el modelado dinámico de estructuras. Se las puede modelar utilizando métodos que permiten discretizar la estructura en elementos más pequeños y resolver las ecuaciones de movimiento resultantes para obtener las respuestas dinámicas de interés. Alguno de estos métodos son el método de los elementos finitos, el método de los elementos de contorno y el método de las diferencias finitas.

Por otro lado, utilizando un enfoque más tradicional, se puede realizar el modelado dinámico de estructuras reduciendo el problema en unos pocos grados de libertad y resolviendo las ecuaciones dinámicas para cada uno de estos y sus relaciones. Esto se

logra simplificando el problema mediante importantes suposiciones. La gran desventaja de este método es que estas suposiciones restan precisión al método y se pueden omitir efectos no predecidos en el análisis conceptual. Sin embargo, este método resulta eficaz para obtener una predicción rápida de la respuesta dinámica de una estructura y tomar diferentes decisiones en función a esta.

En este capítulo se detallará el modelado dinámico realizado para la estructura formada por el prototipo colimador oscilante montado sobre el banco de pruebas diseñado. El diseño de este se detalla en el Capítulo 3. Se consideraron, en primer lugar, las mismas condiciones de movimiento del prototipo colimador oscilante consideradas para la verificación de los componentes principales del mismo en el Capítulo 2. Estas condiciones de movimiento implican un desplazamiento del actuador igual al del colimador. Se trabajó con estas, ya que es el máximo desplazamiento del actuador que fue posible de obtener físicamente con una frecuencia de 1 Hz debido a inconvenientes con el tornillo del husillo de bolas del actuador. Más precisamente, al momento del montaje se encontró que este presentaba cierta curvatura, lo que implicó un aumento de esfuerzos aplicados a este y en consecuencia un mayor torque requerido al motor que lo acciona. Por esta razón se obtuvo un desplazamiento máximo del actuador de 30mm.

Por otro lado, considerando una igualdad del producto masa por desplazamiento entre el colimador y el actuador, el desplazamiento teórico del actuador debería ser de 80 mm. A pesar de no haberse podido alcanzar este desplazamiento físicamente se analizó en este capítulo los resultados que expone el modelo dinámico para este caso.

4.2. Metodología

En la Figura 4.1 se muestra un esquema bidimensional del conjunto banco de pruebas con colimador oscilante y sus dimensiones y masas principales involucradas. Para modelar este conjunto se realizaron ciertas consideraciones. En primer lugar se consideró un empotramiento del banco de ensayos al suelo. Luego se consideró a la estructura del prototipo de colimador oscilante como totalmente rígida, esto ya que presenta perfiles robustos y cortos que además se encuentran rigidizados entre si por refuerzos. Debido al diseño del banco de ensayo y a las características de movimiento del colimador y el actuador se consideró todo el movimiento en una única dirección horizontal, despreciando los momentos aplicados resultantes, los huelgos y el grado de libertad en torsión.

Con las consideraciones realizadas mencionadas anteriormente se modeló la estructura como un sistema con tres grados de libertad como el que se muestra en la Figura 4.2. En este sistema la masa $m_1 = 36,4 \text{ kg}$ representa la masa de la plataforma del banco de ensayo junto con la estructura del sistema colimador oscilante sin las masas

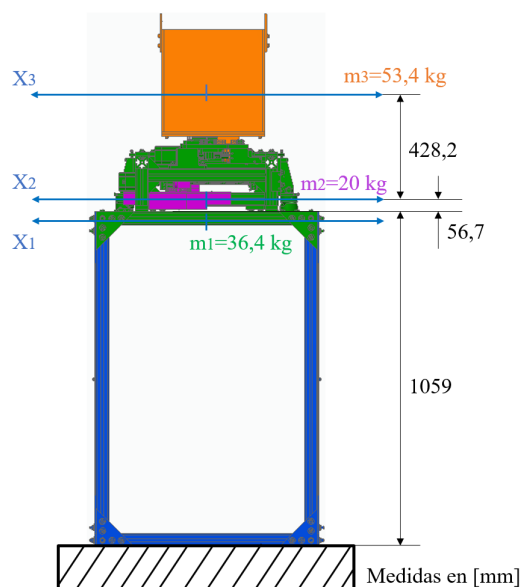


Figura 4.1: Esquema plano del banco de pruebas en conjunto con el prototipo colimador oscilante montado.

móviles (base). La masa $m_2 = 20 \text{ kg}$ representa la masa del conjunto móvil del actuador y la masa $m_3 = 53,4 \text{ kg}$ representa la masa del conjunto móvil del colimador. Para todo el sistema se consideró un coeficiente de amortiguamiento $\xi = 5\%$ dado que se tienen uniones atornilladas en todos los casos siguiendo la bibliografía [16]. La constante elástica $K_1 = 178,6 \frac{kN}{m}$ representa la rigidez del banco de pruebas obtenida en la revisión del diseño de este en el Capítulo 3. Por otro lado, se obtuvieron las constantes elásticas $K_2 = 111 \frac{MN}{m}$ y $K_3 = 396 \frac{MN}{m}$ que representan los tornillos de potencia siguiendo lo desarrollado en el Capítulo 2. Dado estos valores altos de las rigideces de los tornillos se propone para un trabajo a futuro evaluar también la rigidez de la estructura bastidor del prototipo de colimador oscilante.

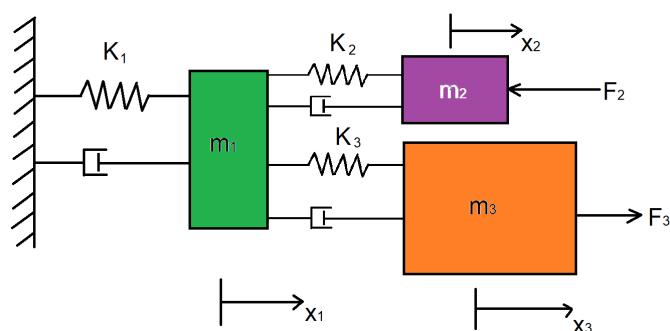


Figura 4.2: Modelo de 3 GDL del banco de pruebas con el prototipo colimador oscilante montado.

Las fuerzas que actúan sobre las masas 2 y 3, fruto de sus respectivas aceleraciones, deben ser soportadas por la masa 1 a través de los tornillos de los husillos de bola. Dado que para el funcionamiento del sistema se requiere un movimiento osci-

lante del colimador con velocidad constante la mayor parte del tiempo, se estableció un movimiento de este con un perfil de velocidades trapezoidal de 1 Hz y un tiempo de aceleración de 2% del periodo tal como se consideró en el Capítulo 2. Las fuerzas requeridas para mantener este movimiento se calcularon durante la verificación de los componentes principales en el Capítulo 2. Los perfiles de la fuerza aplicada al colimador (F_3) y al actuador (F_2) en un periodo se pueden observar en la figura 4.3. En la figura se puede observar un pequeño escalón en el perfil de las fuerzas al momento del cambio de sentido de movimiento, esto se debe a que durante la desaceleración la fricción involucrada ayuda, mientras que durante la aceleración esta fricción se opone a la fuerza actuada. Esto se puede observar en las ecuaciones 2.10 y 2.8 desarrolladas en el Capítulo 2. Se trabajó en primer lugar con este perfil de fuerzas ya que, como se mencionó anteriormente, corresponde al máximo desplazamiento del actuador posible de obtener físicamente con una frecuencia de 1 Hz, debido a inconvenientes con el tornillo del mismo.

Cabe destacar que los perfiles de las fuerzas mostradas no consideran el posible desfase entre ellas ni la dinámica de los motores y la electrónica. El desfase entre estas fuerzas provocaría un adelanto o retraso en la compensación entre estas, aumentando la fuerza aplicada a m_1 . Este caso no se analizó en este trabajo pero se obtendrían desplazamientos de m_1 mayores a los obtenidos sin desfase pero igualmente menores a los obtenidos sin el accionamiento del actuador. Por otro lado, si se considerase la dinámica del motor, los perfiles de las fuerzas se suavizarían y esto aumentaría el tiempo de respuesta pero haría al sistema menos sensible a los desfases mencionados anteriormente.

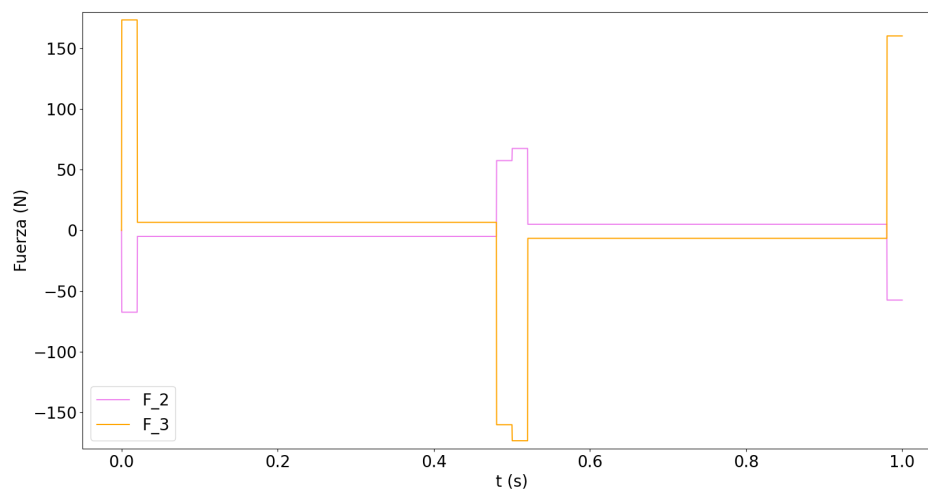


Figura 4.3: Perfil de fuerza aplicada al colimador y al actuador en un periodo.

Teniendo entonces así todos los datos necesarios se realizó un código en lenguaje

Python utilizando el programa Spyder (Anaconda 3)[17] para calcular la dinámica del sistema modelado. Se consideró el sistema como un sistema forzado arbitrariamente donde las fuerzas aplicadas a m_2 y m_3 vienen dadas por el vector de fuerzas $F = (0, F_2, F_3)^T$ donde F_2 y F_3 vienen dados por los perfiles mostrados en la Figura 4.3. Para poder realizar una comparación entre la respuesta sin el accionamiento del actuador y con el accionamiento del mismo simplemente se realizó el mismo cálculo sin considerar F_2 y luego sí considerando esta.

Para calcular entonces la respuesta dinámica del sistema se trabajó utilizando superposición modal y el método numérico Newmark. El código desarrollado se expone en el Anexo C.

Superposición modal

La superposición modal se basa en la suposición de que la respuesta de un sistema a una excitación externa se puede expresar como una combinación lineal de las respuestas modales del sistema. [16]

Cuando una estructura se somete a una carga dinámica, sus modos de vibración naturales son los patrones fundamentales de movimiento que se presentan en el sistema. Cada modo (ϕ_j) de vibración tiene una frecuencia natural asociada y una forma de vibración característica. La superposición modal aprovecha estas características para simplificar el análisis dinámico. Entonces, el análisis modal consiste en determinar las frecuencias naturales, los modos de vibración y los coeficientes de participación modal del sistema. Una vez que se han obtenido estos valores, es posible expresar la respuesta dinámica del sistema a una carga o excitación externa como una suma ponderada de las respuestas modales.

Así, teniendo el sistema amortiguado correspondiente al modelo planteado como muestra la ecuación matricial 4.1 y aplicando el concepto de superposición modal como muestra la ecuación 4.2 se llega a la ecuación matricial 4.3 cuyas matrices son diagonales. Es decir, se llega a un sistema de ecuaciones desacopladas. Cabe aclarar que en estas ecuaciones \mathbf{M} es la matriz de masas, \mathbf{C} es la matriz de amortiguamiento y \mathbf{K} es la matriz de rigideces. Además, $\mathbb{M} = \Phi^T \cdot \mathbf{M} \cdot \Phi$, $\mathbb{C} = \Phi^T \cdot \mathbf{C} \cdot \Phi$, $\mathbb{K} = \Phi^T \cdot \mathbf{K} \cdot \Phi$ y $\mathbb{F} = \Phi^T \cdot \mathbf{F}$.

$$\mathbf{M} \ddot{u}(t) + \mathbf{C} \cdot \dot{u}(t) + \mathbf{K} \cdot u(t) = \mathbf{F}(t) \quad (4.1)$$

$$u(t) = \sum_{j=1}^3 \phi_j \cdot q_j(t) = \Phi \cdot q(t) \quad (4.2)$$

$$\mathbb{M} \ddot{q}(t) + \mathbb{C} \cdot \dot{q}(t) + \mathbb{K} \cdot q(t) = \mathbb{F}(t) \quad (4.3)$$

Método Newmark

Habiendo obtenido entonces el sistema desacoplado se resolvió este utilizando el método de Newmark. Este es un algoritmo utilizado para la solución numérica de ecuaciones diferenciales en el contexto del análisis estructural y la mecánica de materiales. Este método se emplea comúnmente en la dinámica estructural para calcular las respuestas temporales de sistemas estructurales sujetos a cargas dinámicas. [16]

En este algoritmo el tiempo continuo se divide en pequeños intervalos de tiempo discretos, denotados como pasos de tiempo. Cada paso de tiempo se representa por un valor de tiempo t y $t + \Delta t$, donde Δt es el tamaño del paso de tiempo. Luego, el método de integración de Newmark utiliza un esquema de integración temporal para calcular los desplazamientos y velocidades de los nodos en cada paso de tiempo. Se basa en una expansión en series de Taylor y utiliza dos parámetros, β y γ , para controlar la precisión y la estabilidad del método. Entre cada paso se realizan cálculos preliminares para obtener una estimación de los desplazamientos y velocidades en el siguiente paso de tiempo. Luego se corrigen los valores estimados obtenidos en el paso de predicción utilizando información adicional, como las aceleraciones del sistema y las fuerzas externas aplicadas. Una vez que se obtienen los desplazamientos, velocidades y aceleraciones en el paso de corrección, se actualizan estas variables para el siguiente paso de tiempo. Los pasos de predicción, corrección y actualización se repiten para cada paso de tiempo hasta que se alcance el tiempo final deseado.

4.3. Resultados

En primer lugar se obtuvo la matriz Φ de modos normalizados donde cada columna corresponde a un modo normalizado. Los modos normalizados son sencillamente los modos dividido su masa modal correspondiente.

$$\Phi = \begin{pmatrix} -0,134899 & 0,0954178 & 0,0130468 \\ 0,0485257 & 0,0954458 & -0,196304 \\ 0,0737577 & 0,0954387 & 0,0646362 \end{pmatrix} \quad (4.4)$$

Las frecuencias naturales obtenidas para estos modos son $f = \begin{pmatrix} 729,0 \\ 6,4 \\ 387,2 \end{pmatrix}$ Hz.

Se puede observar que para el primer modo, con frecuencia de $6,4 \text{ Hz}$, se tiene que todas las masas oscilan en fase y con igual amplitud. En el segundo modo, con frecuencia de $387,2 \text{ Hz}$, la masa 2 oscila en contrafase con las otras masas. En este modo la masa 1 tiene poca amplitud frente a las otras. Por último, en el tercer modo con frecuencia de $729,0 \text{ Hz}$, la masa 2 y 3 oscilan en fase entre sí y en contrafase con

la masa 1 que oscila con mayor amplitud que las otras.

Luego se graficó la posición absoluta de la masa 1 sin accionamiento del actuador y con accionamiento del mismo durante 2 periodos. Los resultados obtenidos se pueden observar en la Figura 4.4 y la Figura 4.5 respectivamente. En los mismos gráficos se muestran en línea punteada los perfiles de las fuerzas actuantes en cada masa. Se puede observar cómo el desplazamiento de la base tiene sus máximos debido al aumento de las fuerzas al cambiar de dirección el movimiento.

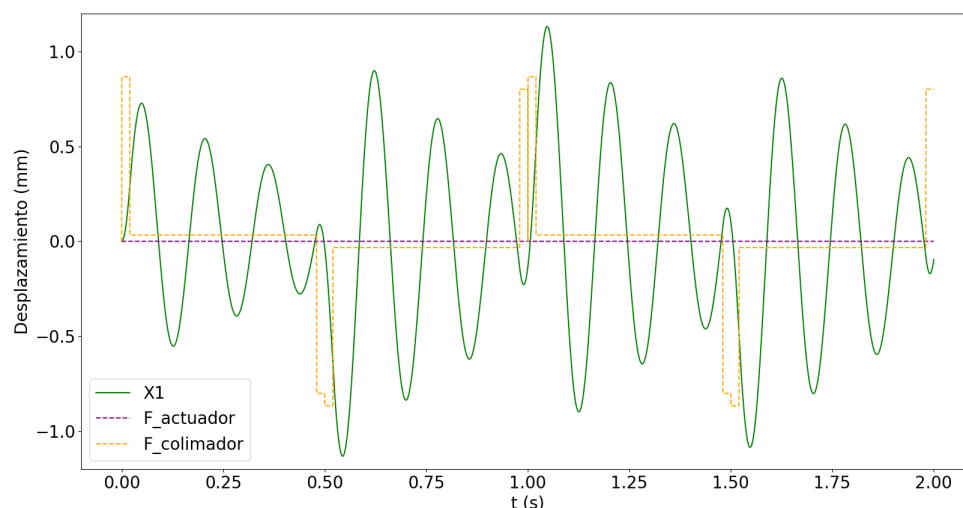


Figura 4.4: Desplazamiento absoluto de m_1 sin accionar el actuador.

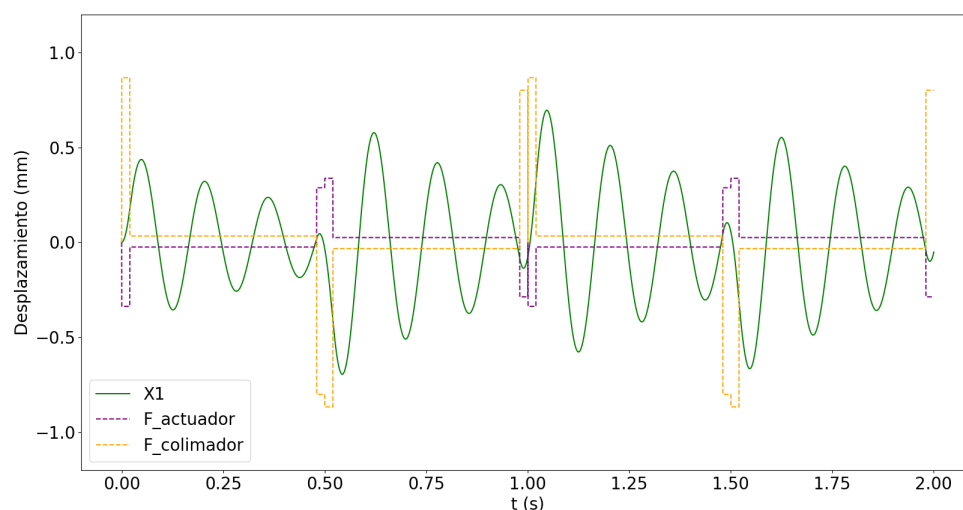


Figura 4.5: Desplazamiento absoluto de m_1 accionando el actuador.

De las figuras anteriores se puede observar una disminución del desplazamiento absoluto de la masa 1 que es justamente lo buscado con el accionamiento del actuador. Para comparar mejor el desplazamiento de m_1 antes y después de accionar el actuador se graficaron estos en conjunto en la Figura 4.6. Sin el accionamiento del actuador se obtuvo un desplazamiento máximo de m_1 de 1,13 mm, mientras que con el accionamiento del actuador se obtuvo un desplazamiento máximo de 0,70 mm. Con estos

resultados se puede observar que el mecanismo de reducción de vibraciones reduce un 38 % el valor máximo de desplazamiento.

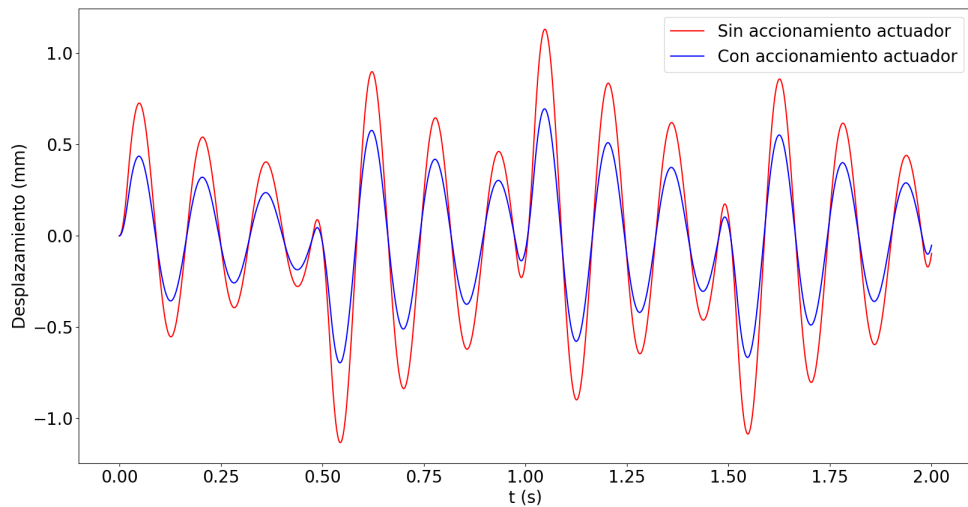


Figura 4.6: Comparación del desplazamiento absoluto de m_1 sin y con el accionamiento del actuador.

Por otro lado, se calcularon las aceleraciones de m_1 antes y después de accionar el actuador. Se graficaron estas en conjunto en la Figura 4.7. Sin el accionamiento del actuador se obtuvo una aceleración máxima de m_1 de $2,89 \frac{m}{s^2}$, mientras que con el accionamiento del actuador se obtuvo una aceleración máxima de $1,98 \frac{m}{s^2}$. Con estos resultados se puede observar que el mecanismo de reducción de vibraciones reduce un 31,5 % el valor máximo de aceleración.

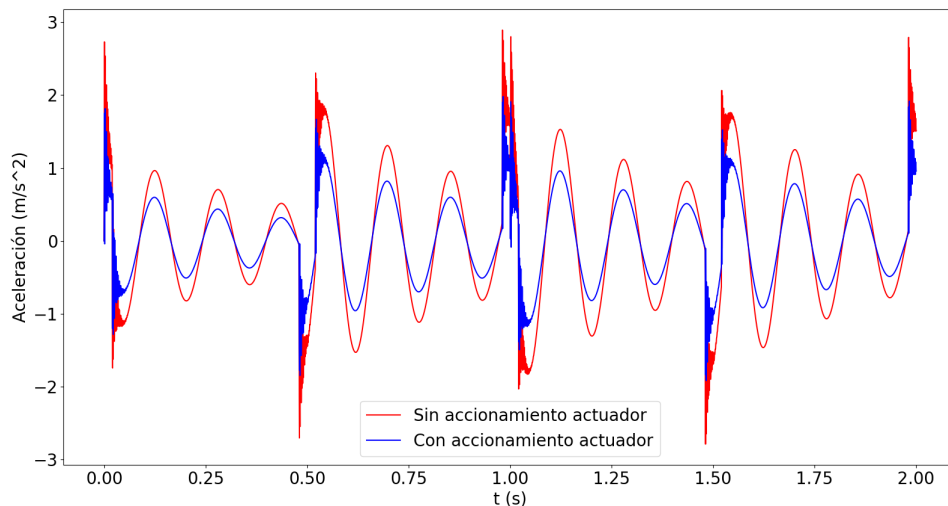


Figura 4.7: Aceleraciones de m_1 sin y con el accionamiento del actuador.

Como se puede observar, los resultados obtenidos no muestran una gran disminución de los desplazamientos y las aceleraciones con el accionamiento del actuador. Esto se debe a que se consideró un desplazamiento de 30 mm tanto para el actuador como para el colimador ya que durante las pruebas experimentales descritas en el Capítulo

6 no se pudo imponer un mayor desplazamiento que el mencionado debido a diferentes inconvenientes. Sin embargo, se analizó la respuesta de este mismo modelo dinámico considerando un desplazamiento del actuador de 80 mm. Se tomó esta distancia para tener una equivalencia de masa por desplazamiento entre el colimador y el actuador inercial.

Considerando entonces este desplazamiento de 80 mm del actuador se obtuvo el perfil de fuerzas expuesto en la Figura 4.8. En esta figura ya se puede observar claramente como, con un mayor recorrido, el actuador inercial puede compensar mediante las fuerzas inerciales que genera las fuerzas inerciales del colimador oscilante.

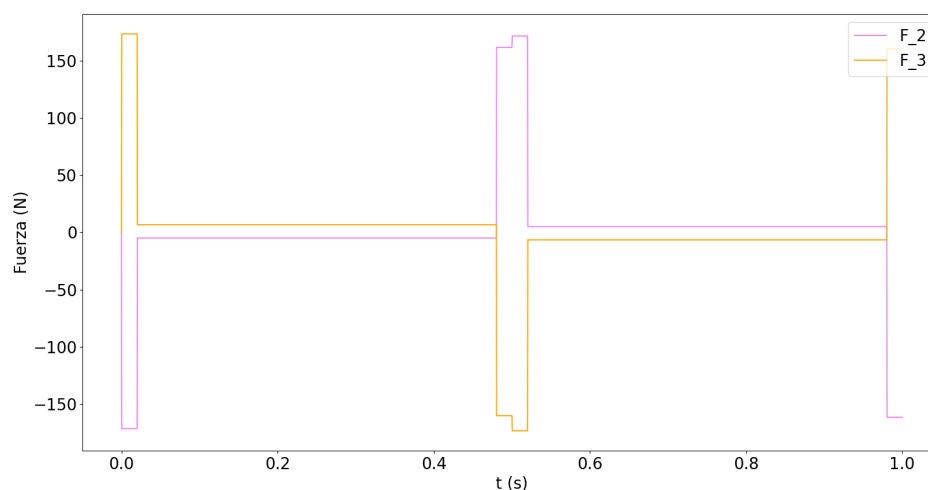


Figura 4.8: Perfil de fuerza aplicada al colimador y al actuador en un periodo con un desplazamiento de 80 mm del actuador.

En la Figura 4.9 se pueden observar los desplazamientos de la base sin y con el accionamiento del actuador para este caso, mientras que en la Figura 4.10 se comparan las aceleraciones de la base sin y con el accionamiento del actuador. De estas figuras se observa un desplazamiento máximo sin el accionamiento del actuador de $1,13 \text{ mm}$, mientras que con el accionamiento del actuador se tiene un desplazamiento máximo de $24 \mu\text{m}$. Esto da una disminución del desplazamiento máximo de la base de 98% . Con respecto a las aceleraciones se tiene una aceleración máxima sin el accionamiento del actuador de $2,89 \frac{\text{m}}{\text{s}^2}$, mientras que con el accionamiento del actuador se tiene una aceleración máxima de la base de $0,83 \frac{\text{m}}{\text{s}^2}$. Esto da una disminución de la aceleración máxima de la base de $71,3\%$. Esta fuerte disminución de las aceleraciones representa una fuerte disminución de las fuerzas transmitidas a la base, lo cual es lo que se busca al implementar el actuador inercial.

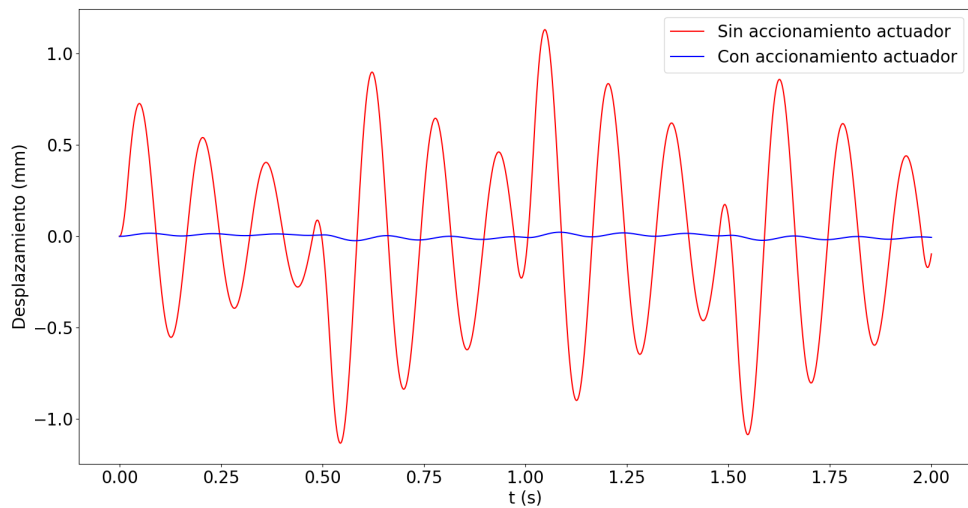


Figura 4.9: Comparación del desplazamiento absoluto de m_1 sin y con el accionamiento del actuador para un recorrido de 80mm de este.

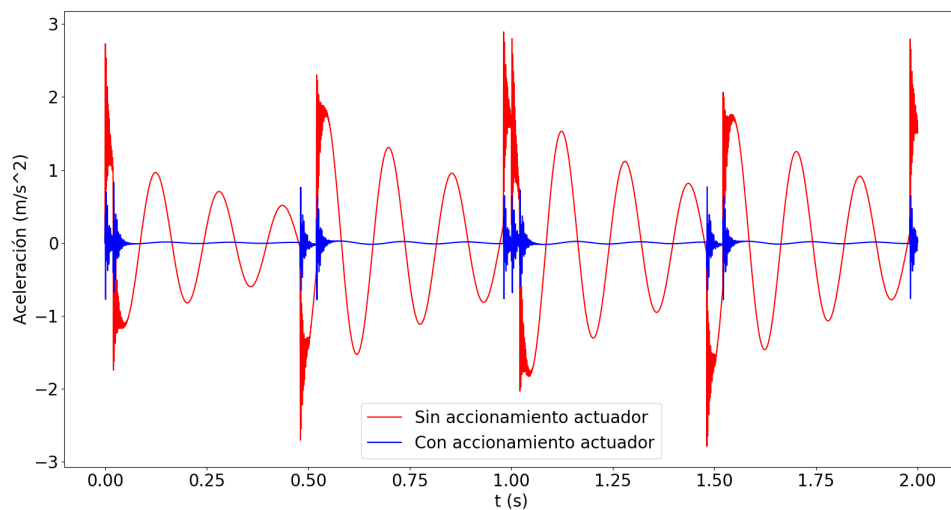


Figura 4.10: Aceleraciones de m_1 sin y con el accionamiento del actuador para un recorrido de 80 mm de este.

4.4. Conclusiones

En este capítulo se analizó la reducción de vibraciones que el prototipo colimador oscilante transfiere al banco de pruebas sin y con el actuador inercial moviéndose en contrafase. En primer lugar se realizó un modelado del sistema con 3 grados de libertad constando de tres masas con desplazamiento en una única dirección. Se modelaron las fuerzas internas del sistema que le realizan las masas a la base como fuerzas externas aplicadas al colimador y al actuador. La respuesta dinámica del sistema se calculó utilizando el desacoplamiento modal del sistema y el método numérico Newmark.

Como resultado se obtuvo una disminución del desplazamiento máximo de la base de 38 % siendo el desplazamiento máximo remanente de 0,70 mm. Además se obtuvo una

disminución de la aceleración máxima de la base de 31,5 % siendo la aceleración máxima remanente de $1,98 \frac{m}{s^2}$. Mostrando estos resultados se observa una baja disminución de los desplazamientos y las aceleraciones con el accionamiento del actuador, debido a que se consideró un desplazamiento de 30 mm para ambos conjuntos móviles. Luego se analizó la respuesta de este mismo modelo dinámico considerando un desplazamiento del actuador de 80 mm. Se tomó esta distancia para tener una equivalencia de masa por desplazamiento entre el colimador y el actuador inercial. En este segundo análisis se obtuvo una disminución del 98 % del desplazamiento máximo y una disminución del 71,3 % de la aceleración máxima. Así se obtuvo un desplazamiento máximo remanente de $24 \mu m$ y una aceleración remanente de $0,83 \frac{m}{s^2}$.

Esta alta disminución del desplazamiento de la base considerando un recorrido de 80 mm del actuador podría inducir a que simplemente sincronizando los movimientos se puede llegar a obtener un control de las vibraciones aceptable sin necesidad de realizar un control a lazo cerrado. Sin embargo para confirmar esto se requieren más estudios. Cabe aclarar que debido a problemas físicos que se explican en el Capítulo 6 no se pudo imponer esta longitud de desplazamiento del actuador de 80mm siendo la longitud máxima alcanzada los 30mm analizados en primer lugar.

Capítulo 5

Instrumentación del prototipo

“Existen dos tipos de lenguajes de programación: por un lado, aquellos de los que la gente se queja todo el rato; por otro, los que nadie utiliza.”

— Bjarne Stroustrup

5.1. Introducción

Para que el prototipo de colimador oscilante funcione, que sus motores se muevan y que se puedan realizar las mediciones correspondientes para verificar la disminución de vibraciones, es necesaria la instrumentación del mismo. Esta instrumentación abarca desde el cableado de los componentes electrónicos del dispositivo, el armado de una placa electrónica que permita conectar el microcontrolador con los controladores de los motores, hasta la programación de los microcontroladores involucrados y de la interface con el usuario. En primer lugar se detallará la selección, armado e interconexión del hardware, para luego explicar el software desarrollado para cada caso.

5.2. Hardware

Como ya se mencionó en capítulos anteriores, los actuadores del prototipo de colimador oscilante son dos motores paso a paso NEMA 23 y un motor paso a paso NEMA 17. Este último es el encargado de actuar el tornillo de potencia que hace pivotar el conjunto rotante alrededor del eje pivote, mientras que los otros son los encargados de actuar los tornillos de los husillos de bola que producen el movimiento lineal del actuador inercial y del colimador. En la Figura 5.1 se muestra esquemáticamente la interconexión entre los distintos componentes que forman el hardware utilizado que se describe más detalladamente a lo largo de esta sección.

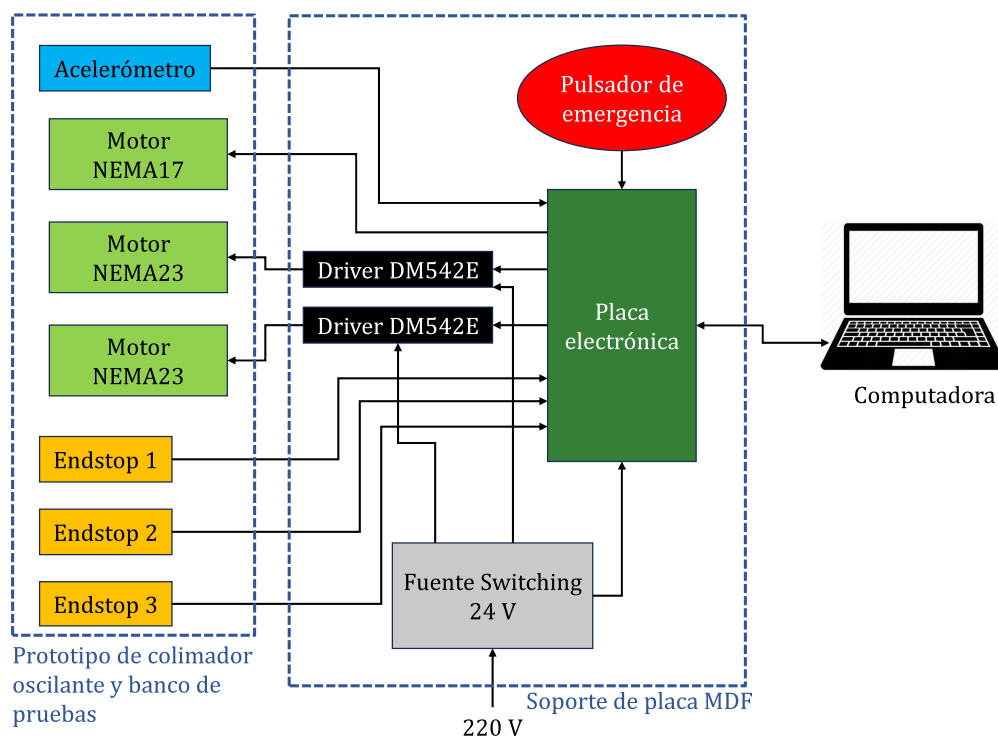


Figura 5.1: Esquema de interconexión entre los componentes que forman el hardware.

Para controlar estos motores se requirió como interfase entre estos y un microcontrolador los controladores correspondientes para cada motor. Para ambos motores NEMA 23 se utilizaron los Driver DM542E de Leadshine mientras que para el motor NEMA 17 se utilizó el Driver DRV8825. Luego, para controlar los movimientos de los motores, se seleccionó una placa de desarrollo conocida como "bluepill", que tiene un microcontrolador STM32f103C8T6. Este microcontrolador permite además la lectura de los finales de carrera de cada movimiento y una comunicación tipo serie con la computadora. Estos finales de carrera cumplen tanto una función de seguridad como permitir la calibración de la posición de las masas.

Para poder implementar los componentes mencionados se requiere de una placa electrónica que los vincule. Para esto se realizó en primer lugar un prototipo de la misma utilizando una protoboard. Este prototipo se muestra en la Figura 5.2. Así se programaron, en primer lugar, códigos básicos que permitieran manejar salidas y leer entradas del microcontrolador, para luego implementar movimientos simples a los motores hasta llegar a un código final completo el cual se desarrollará en la siguiente sección.

Luego de verificar el funcionamiento del prototipo, se procedió a diseñar y fabricar una placa electrónica en una placa universal preperforada. Esta placa se expone en la Figura 5.3 detallando sus componentes principales. Entre estos se destacan la BluePill

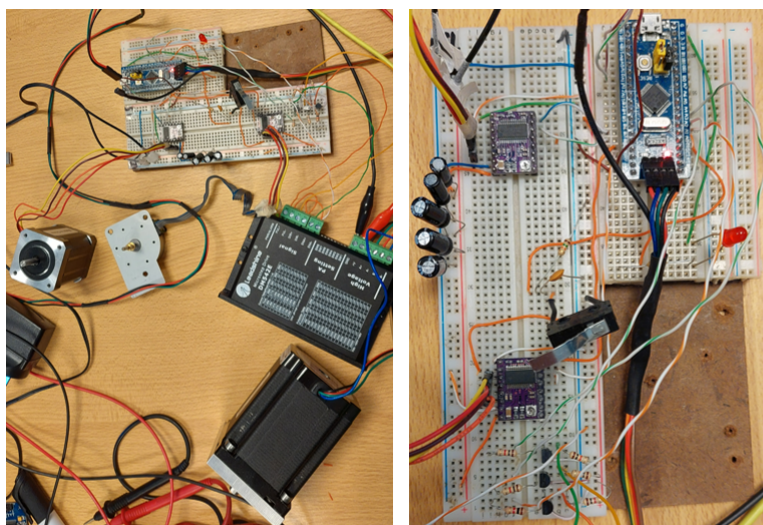


Figura 5.2: Prototipo inicial de la placa electrónica.

con el microcontrolador, las borneras de salida para conectar los controladores de los motores NEMA 23 y las borneras de entrada para conectar los finales de carrera. También se muestra el controlador DRV8825 para controlar el motor paso a paso NEMA 17; la entrada para la lectura de un acelerómetro y las salidas para la comunicación en serie y protocolo Ethernet. Además, cabe destacar que la placa es alimentada por 24V de corriente continua ya que esta será la tensión de alimentación de los controladores DM542E y así se evita la necesidad de utilizar varias fuentes. Dentro de la placa se encuentra un Stepdown que transforma los 24V a 12V. Este alimenta al motor NEMA17 y mediante un regulador 7805 y un regulador LM317 se transforman estos 12V en 5V y 3.3V respectivamente. Los 3.3V alimentan los finales de carrera y la salida Ethernet, mientras que los 5V alimentan los leds, la BluePill, los transistores para ajustar la señal para los controladores DM542E, el controlador DRV8825, entre otros. En el Anexo D se adjunta el plano eléctrico de esta placa electrónica.

Para mantener todo unido se montó esta placa electrónica, los controladores de los motores, un módulo de Ethernet ENC28J60, un pulsador de corte por emergencia y una fuente switching de 24V sobre un soporte de placa MDF. En la Figura 5.4 se puede observar el soporte descrito indicando cada componente. Se conectaron todos los componentes entre sí y mediante una bornera se conectaron los motores NEMA23 y la alimentación de 220V a esta electrónica. Cabe aclarar que se conectaron tanto los finales de carrera como el pulsador de emergencia como normal cerrado, detectando una apertura del circuito si se pulsan. Esto último para mantener la seguridad ante una desconexión o el corte de un cable.

Por otro lado, para la medición de aceleraciones se seleccionó un acelerómetro ADXL335. Este es un acelerómetro analógico de tres ejes que ofrece una medición de aceleración en tiempo real. Puede medir la aceleración en tres ejes: X, Y y Z; y

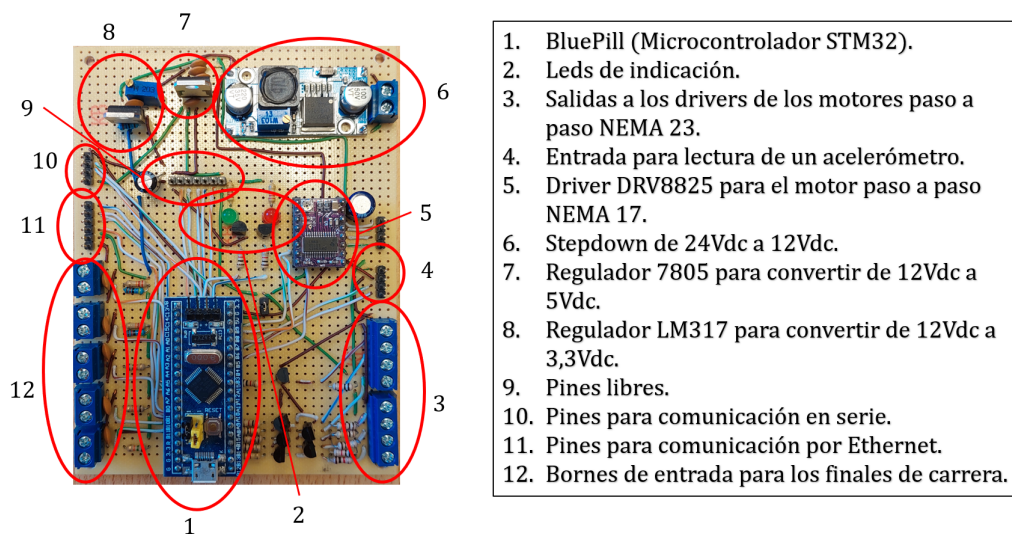


Figura 5.3: Placa electrónica fabricada para la instrumentación del prototipo colimador oscilante.

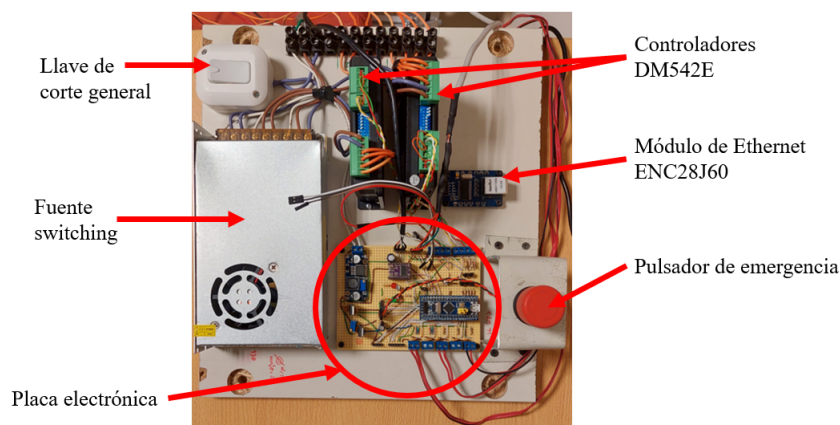


Figura 5.4: Soporte de electrónica.

proporciona una salida analógica proporcional a la aceleración medida en cada uno de estos ejes. Este acelerómetro tiene un rango de medición ajustable que permite seleccionar la escala adecuada para su aplicación. Puede medir aceleraciones de $\pm 3 g$, $\pm 5 g$ o $\pm 10 g$, lo que permite adaptarlo a diferentes requerimientos. Las tensiones de sus salidas analógicas permiten una fácil lectura mediante el microcontrolador evitando comunicaciones bloqueantes que afecten a la dinámica del dispositivo. Además, el PCB del acelerómetro cuenta con un filtro antialias integrado con frecuencia de corte de 50 Hz.[18]

A este acelerómetro se lo montó en primer lugar en el punto medio de uno de los laterales de la plataforma del banco de pruebas para así disminuir el ruido generado por las aceleraciones perpendiculares al movimiento principal y poder medir mejor las aceleraciones en el sentido del movimiento principal. En la Figura 5.5 se muestra la ubicación del acelerómetro y el modo de montaje. Este se montó a la estructura mediante un soporte diseñado específicamente e impreso en 3D con plástico PLA, al

cual se le pegó con pegamento epoxy de dos componentes el acelerómetro. Esto último se realizó para obtener una unión lo más rígida posible y evitar así mediciones erróneas o con ruido intermedio por la dinámica de la fijación.

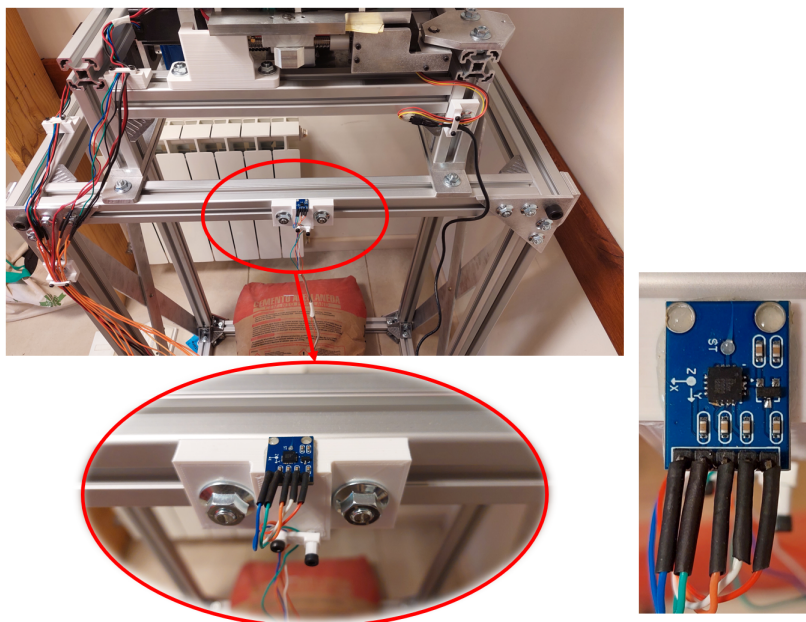


Figura 5.5: Fijación del acelerómetro en la estructura.

5.3. Software

5.3.1. Programa del microcontrolador

Tal como se mencionó en la sección anterior, para el control de los actuadores del prototipo colimador oscilante se programó un microcontrolador STM32 en lenguaje C++ utilizando el programa Arduino 1.8.19. Luego de un largo proceso de programación de distintos códigos yendo de lo general a lo particular, versión a versión, se llegó al programa final. El código de este programa se adjunta en el Anexo E.

El programa tiene por objetivo controlar los motores para que realicen una rutina donde el movimiento lineal del colimador y el movimiento angular del mismo se actúen en fase y el actuador inercial se actúe en contrafase. Para poder realizar estos movimientos también se requiere de un proceso de posicionamiento inicial denominado "homing", o calibración inicial de la posición mediante un final de carrera, que tiene por objetivo fijar el origen de los movimientos para cada motor. Por seguridad se deben leer pulsadores de fin de carrera y el pulsador de emergencia. Además, debe ser capaz de leer y procesar mensajes de comando y enviar mensajes con información por puerto serie, como también leer el acelerómetro a través de pines con entrada analógica. También debe interpretar y responder ante mensajes de código G preestablecido. Como nomenclatura se mencionó motor 1 al motor que acciona el actuador inercial,

motor 2 al que acciona el movimiento lineal del colimador y motor 3 al que acciona el movimiento pivotante del colimador.

Para evitar funciones bloqueantes que impidan la interrupción de procesos se programó una máquina de estados para los distintos estados del motor en la cual se verifica por interrupción cíclica el estado establecido, como el estado de los interruptores de fin de carrera y del pulsador de emergencia para actuar en consecuencia. Sin embargo, la función de homing y la correspondiente al código G se las definió como funciones bloqueantes ya que implican muchos movimientos que se buscan evitar interrumpir y que sobrecargan el bucle de evaluación de estados. En estas funciones se realizan igualmente las verificación de los factores de seguridad durante cada movimiento.

Un diagrama de flujo del programa final alcanzado se muestra en la Figura 5.6. Para este programa se utilizó las librería AccelStepper para el control de los motores paso a paso.

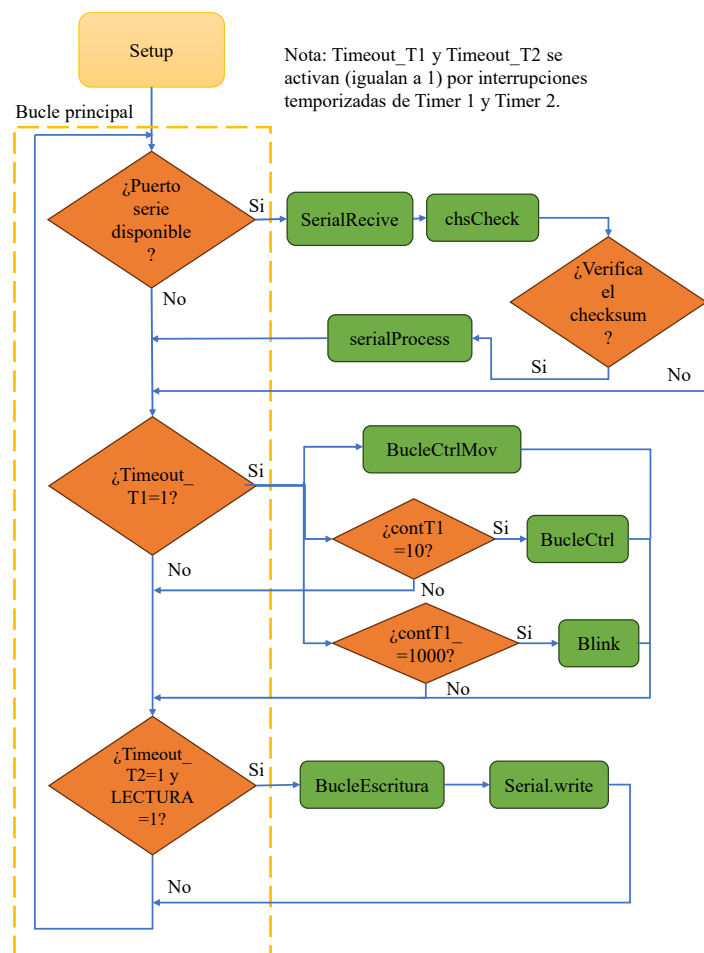


Figura 5.6: Diagrama de flujo del código programado para el microcontrolador.

El diagrama de flujo anterior pretende mostrar esquemáticamente el funcionamiento

del código escrito con el cual se programó el microcontrolador. Este programa contiene una etapa de inicialización o Setup la cual se realiza solo al inicio del programa. En esta función se definen, en primer lugar, la función de cada pin utilizado, es decir, se determina si es un pin de lectura o escritura. Luego también se inicializa la comunicación en serie con un baudrate de 115200. A los motores se les establece una velocidad máxima lineal de $2 \frac{mm}{s}$ por seguridad, aunque esta velocidad no se aplicaría en ningún momento. También se deshabilitan todos los motores y se apaga el led indicador de la BluePill. Finalmente, se activa la salida para el funcionamiento del pulsador de emergencia, se calculan las variables de la rutina con la función CalculateVariables y se inicializan las interrupciones cíclicas con los timer TIM1 y TIM2. Como factores pre-determinados se estableció para la rutina un período de 1 segundo, un desplazamiento de 30 mm de los movimientos lineales y de 3 mm para el actuador del pivoteo, y un tiempo de aceleración del 2 % del período para todos los motores. La interrupción cíclica correspondiente al timer TIM1 es fija con un período de $100 \mu s$, mientras que la interrupción cíclica correspondiente al timer TIM2 es en función del período de lectura deseado. Este período de lectura se recibe como parte del mensaje que inicia la lectura del acelerómetro. Sin embargo se inicializa arbitrariamente en la función recién descrita con $10 ms$.

Bucle principal

Luego de finalizar con la inicialización del microcontrolador se pasa a su bucle principal en el cual se realizan tres verificaciones de condición. La primer condición evalúa si se recibió un caracter mediante comunicación serie. Si es así se pasa el mensaje a la función SerialRecive en la cual se lo almacena en un buffer y se lo pasa a la función chsCheck. Esta función verifica el Checksum recibido con un cálculo local del mismo. Este Checksum se lo calcula como la suma de todos los componentes del mensaje menos el caracter de inicio, de fin (HEAD y TAIL) y el propio chequeo. Si el Checksum recibido es igual al local se verifica el mensaje y se pasa el buffer a la función serialProcess. En esta función se procesa el mensaje dependiendo de su objetivo. Si es un mensaje que establece algún estado simplemente se cambia el estado de la variable correspondiente. Si el mensaje es para cambiar alguna variable de la rutina se sobrescribe la variable correspondiente y se recalculan todas las variables llamando a la función CalculateVariables que utiliza las ecuaciones 2.2 y 2.3 desarrolladas en el Capítulo 2. Si el mensaje es para seguir algún código G o realizar el homing se llama a la función CodigoG o Homefunc respectivamente. Por último, si el mensaje indica el comienzo de la lectura del acelerómetro se establece el período de lectura y el número de muestras, se re-inicializa el timer TIM2 con el período de interrupción requerido y se establece el contador de lecturas cont.lect a 0 y la variable LECTURA a 1. Finalmente

se vuelve al bucle principal.

La segunda condición que se evalúa en el bucle principal es el estado de la variable `Timeout_T1` que se activa en cada interrupción del timer TIM1 con un período de $100 \mu s$. Si esta variable está activada se llama a la función `BucleCtrlMov`. Esta función evalúa el estado de los motores y si estos deben realizar un nuevo paso se manda la señal correspondiente a los controladores. También se realizan dos evaluaciones de condición más. Si la variable `contT1 = 10` se reinicializa esta variable a cero y se llama a la función `BucleCtrl` que actúa en función del último estado establecido. Por último, si la variable `contT1_ = 1000` se reinicializa esta variable a cero y se llama a la función `Blink`. Esta función simplemente tiene una lógica de evaluaciones que prende o apaga el led indicador de la BluePill y así lo hace parpadear con una frecuencia de 1 Hz. Finalmente se vuelve al bucle principal.

La última condición evaluada en el bucle principal es el estado de la variable `Timeout_T2` y la variable `LECTURA`. Si ambas variables se encuentran habilitadas se llama a la función `BucleEscritura`. Esta función evalúa primero si se llegó a la cantidad de lecturas requeridas. Si es así se sobrescriben las variables `LECTURA` y `cont_lect` con 0. Si no es así, se lee el acelerómetro y se guardan sus valores en el buffer `buff_rec` de dos componentes `uint8_t`. Luego se escribe el buffer `buff_out` con 4 componentes `uint8_t` donde en los primeros dos se copian los valores del buffer `buff_rec` en el mismo orden y en los últimos se registra la velocidad del motor del colimador, separando los valores devueltos por la función `myStepper2.speed` en los 8bits más significativos y menos significativos copiándolos al buffer en este orden. Luego se suma 1 a la variable `cont_lect` y se envía el buffer `buff_out` mediante el puerto serie con la función `Serial.write`.

A continuación se describirá el funcionamiento de las funciones principales mencionadas pero no descritas anteriormente. Estas funciones son `BucleCtrl`, `Homefunc` y `CodigoG`.

Función bucle de control

En primer lugar se detallará entonces el funcionamiento de la función `BucleCtrl`. En la Figura 5.7 se observa un diagrama de flujo de la misma. Cabe aclarar que en primer lugar se define globalmente una enumeración con todos los estados definidos, estos son: `Alarma`, `Alarma1`, `Espera`, `Rutina`, `Rutina1`, `Stop`, `Disable` y `Enable`. Para una correcta comprensión de esta función es necesario tener siempre en cuenta que esta función se llama por interrupción cíclica con un período de $1 ms$. Esto quiere decir que las verificaciones realizadas actúan como bucles si no se cambia la variable estado. Además es necesario que el tiempo de ejecución de esta función sumado al tiempo de ejecución a las demás funciones involucradas en el bucle principal debe ser menor al período de la interrupción más rápida. Es por esta razón también que se decidió

quitar las funciones que realizan el homing y responden al código G de estos bucles y trabajarlas como funciones bloqueantes y no con una máquina de estados, además de que se busca evitar interrupciones a estos movimientos particulares para evitar un mal funcionamiento de los mismos.

Volviendo específicamente a la función `BucleCtrl` se observa cómo en primer lugar se verifica el estado del pulsador de emergencia si el estado momentáneo no es `Disable` ni `Espera`. Si entonces no se está en estos estados y el pulsador se encuentra activado se define directamente el estado `Disable`. Si no se encuentra activado se regresa a la rama principal. Luego se verifica si se encuentra en estado `Alarma`. Si es así se frenan todos los motores entrando a un bucle donde se verifica si terminaron de frenar. Al finalizar con el frenado se define el estado `Alarma1` y se vuelve a la rama principal. Posteriormente se verifica si se encuentra el estado `Alarma1`. De ser así se realizan una serie de verificaciones que resumidamente verifican que final de carrera se activó y en que posición actual se encuentra el motor correspondiente a dicho final de carrera. Esto último se realiza ya que los finales de carrera de cada eje se encuentran conectados en serie, con lo que no se cuenta con una diferenciación posible para saber en que sentido corregir. Entonces si la posición actual es menor a cero se mueve el motor 1 mm en sentido positivo, mientras que si la posición actual es mayor a cero se lo mueve 1 mm en sentido negativo y se vuelve a la rama principal. Si ya ningún final de carrera se encuentra activo se define el estado `Disable`. Luego, en la rama principal se verifica el estado `Espera`, pero si este se encuentra activo no se realiza ninguna acción y simplemente se continúa. Continuando, le sigue la verificación del estado `Enable`. Si este se encuentra activo se habilitan todos los motores y se define el estado `Espera` antes de retornar a la rama principal. Algo análogo a lo anterior sucede luego, al verificar el estado `Disable`, solamente que en este caso se deshabilitan los motores. Posteriormente se verifica el estado `Stop`. Si se encuentra en este estado se comanda el frenado de todos los motores y se ingresa a un bucle donde se verifica si terminaron de frenar todos. Cuando ya todos se encuentran frenados se define el estado `espera` y se retorna a la rama principal. Llegando al final se verifica el estado `Rutina` y si este estado se encuentra activo se aplican los parámetros establecidos para la rutina de cada motor y se define el estado `Rutina1` previo al regreso a la rama principal. Por último se verifica el estado `Rutina1`. Si este se encuentra activo se continúa con una serie de verificaciones donde la primera verifica el estado de los pulsadores de fin de carrera. Si en el proceso de rutina se toca algún fin de carrera se define el estado `Alarma` y se sale de la función. De no ser así se hace una verificación para cada motor de si este llegó a su destino. Si llegó a su destino se redefine su destino como su posición actual con signo contrario. Una vez verificadas las posiciones de todos los motores se sale de la función.

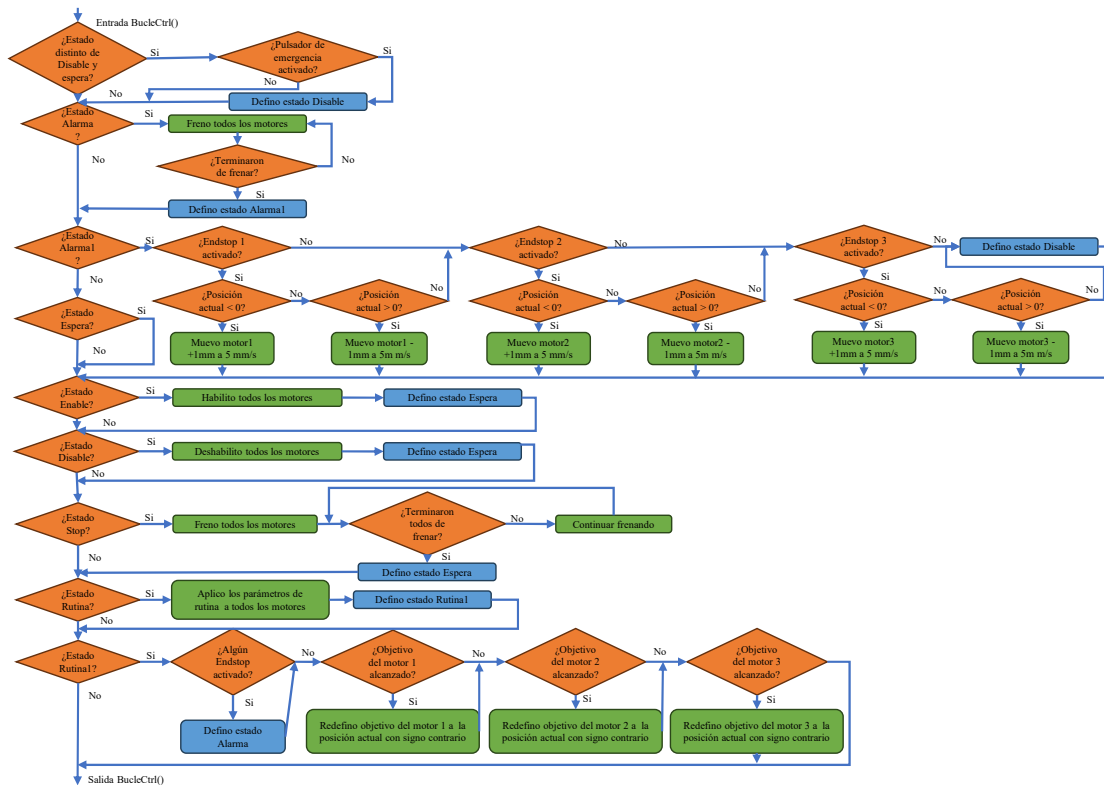


Figura 5.7: Diagrama de flujo del código programado para la función BucleCtrl.

Función de calibración inicial de la posición

En segundo lugar se detalla la función Homefunc. Un diagrama de flujo de esta función se observa en la Figura 5.8. En este diagrama se puede observar que al llamar a la función primero se prende el led indicador de la BluePill, esto es para que el led se mantenga encendido mientras se está realizando el proceso de homing. Análogamente se observa que en todos los casos, antes de salir de la función se vuelve a apagar este led. Antes de comenzar con el proceso de posicionamiento se verifica que los motores estén habilitados, si esto no ocurre simplemente se sale de la función posterior al apagado del led indicador. Esto es para que no se bloquee el programa en la función sin poder hacer el homing por encontrarse los motores deshabilitados. Si los motores se encuentran habilitados lo que se realiza es el frenado de los mismos, esto por si se llama a la función estando los motores en movimiento. En este proceso se verifica que no se toquen los pulsadores de finales de carrera. En caso de suceder esto se frenan todos los motores y se establece el estado Alarma antes de salir de la función. En caso de no suceder esto se procede a realizar el homing del motor 1. Entonces en primer lugar se establece la velocidad y la aceleración para alcanzar el final de carrera. Luego se entra a un bucle donde se verifica si el final de carrera correspondiente a este motor se activa, si no es

así se prosigue actuando el motor hasta que llegue al final de carrera. Al llegar al final de carrera se frenan todos los motores para luego definir la velocidad con la cual se soltará el final de carrera. Nuevamente se entra a un bucle donde se verifica si el final de carrera correspondiente a este motor se desactiva, si no es así se prosigue actuando el motor hasta que se desactive el final de carrera. Una vez desactivado el final de carrera se definen los pasos y la velocidad para llegar al centro de los movimientos requeridos. Mientras no se realizaron la cantidad de pasos establecidos se mueve el motor con un previo chequeo de activación de finales de carrera. Si algún final de carrera se activa en este proceso se frenan todos los motores y se establece el estado Alarma para luego salir de la función posterior apagado del led indicador. Cuando se alcanzan los pasos establecidos se frenan todos los motores verificando nuevamente en el proceso la activación de algún final de carrera. Una vez frenado el motor se establece la posición alcanzada como 0 y se pasa a realizar el mismo proceso con el motor 2 y luego con el motor 3. Cabe aclarar que durante todos los movimientos de los motores dentro de esta función se verifica el estado del pulsador de emergencia y, en caso de activarse, se frenan todos los motores y se establece el estado Alarma para luego salir de la función apagando el led indicador. Esto último no se incluyó en el diagrama por falta de espacio.

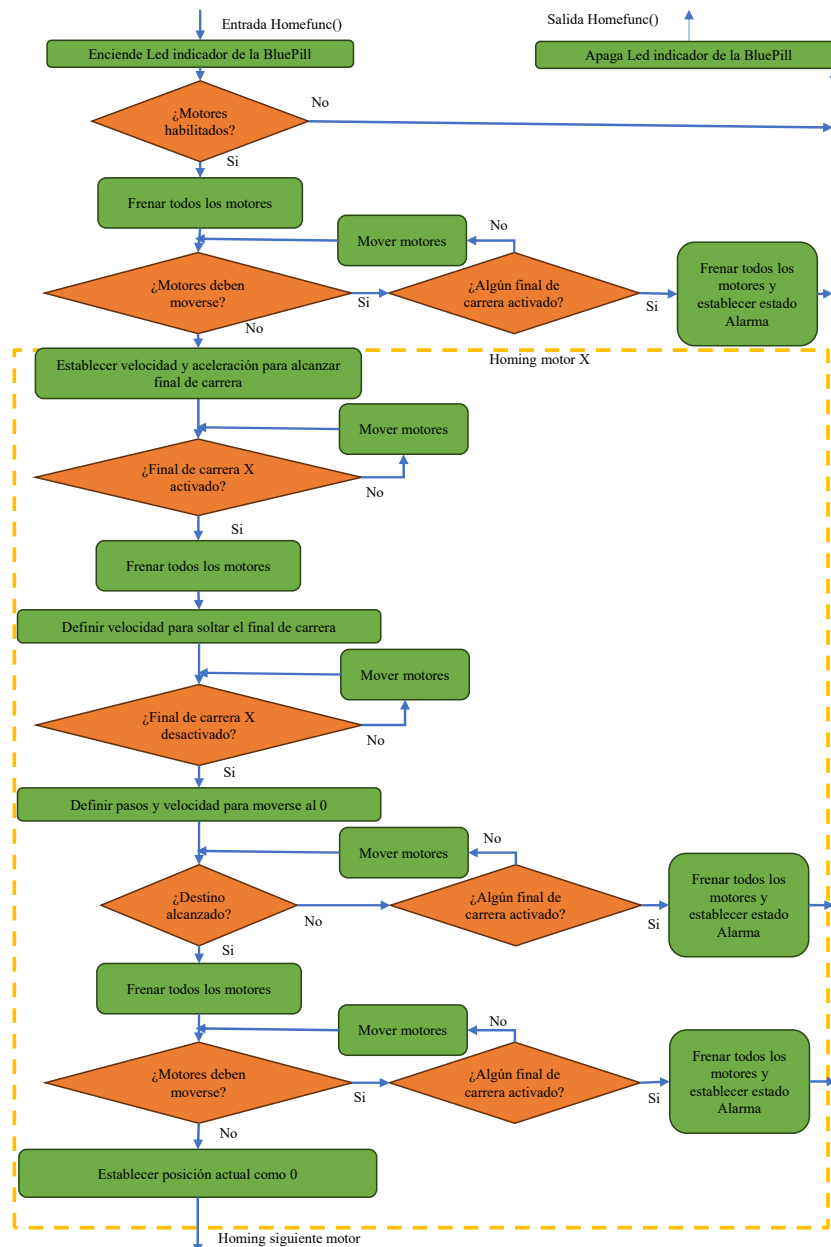


Figura 5.8: Diagrama de flujo del código programado para la función Homefunc.

Función del código G

Por último se procederá a detallar la lógica del código de la función CódigoG. En la Figura 5.9 se expone un diagrama de flujo de este código para un solo motor siendo exactamente igual para los tres motores. La principal diferencia de esta función frente a las anteriores es que tiene como argumento el mensaje recibido que la llamó mediante la función serialProcess. Una vez llamada esta función se verifica en primer lugar si el comando corresponde al motor seleccionado. Si es así se lee el mensaje obtenido como argumento y se lo decodifica guardando el objetivo y la velocidad para alcanzarlo que el mensaje define. Luego se le define al motor dicho objetivo y velocidad para una

vez definidos estos parámetros entrar a un bucle donde se verifica si el motor se debe seguir moviendo o no. Mientras el motor se debe seguir moviendo se verifica el estado de los pulsadores de final de carrera y del pulsador de emergencia. Si alguno de estos se encuentra activo se ordena frenar todos los motores y moverlos hasta que frenen por completo. Cuando el motor ya no se mueve se pasa al mismo proceso con los otros dos motores.

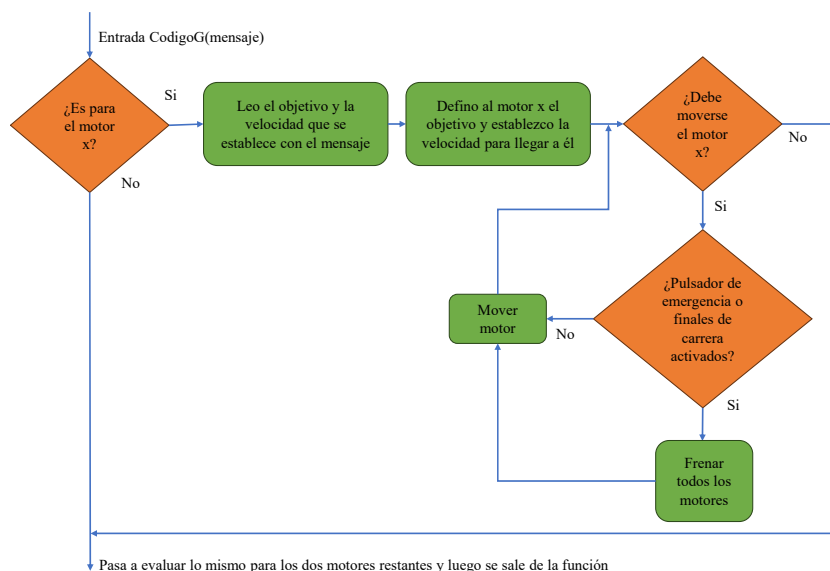


Figura 5.9: Diagrama de flujo del código para un solo motor programado para la función CodigoG.

5.3.2. Programa de la interfaz

Para poder establecer una comunicación activa de escritura y lectura con el microcontrolador programado como se describió en la sección anterior, se programó una interfaz con el usuario utilizando el conjunto de librerías de Qt Creator y el entorno de programación QtCreator 4.5.0. Se utilizó el conjunto de librerías de Qt en su versión opensource 5.10 [19]. El código de este programa se lo escribió en el lenguaje C++. Se seleccionó este programa ya que es un programa con versión gratuita, opensource y multiplataforma que permite realizar interfases dinámicas tipo Scada.

El programa fue dividido en distintos archivos encabezado (.h) y archivos fuente (.cpp). En los encabezados se definen las funciones que se utilizarán como también las variables y las señales, esto tanto como variable pública o privada. Luego, en los archivos fuente se define la lógica de cada función.

El programa debe tener las siguientes características básicas:

- Poder enviar mensajes de comandos al microcontrolador para que este ejecute los movimientos deseados.

- Poder leer datos enviados por el microcontrolador y guardarlos en un paquete auxiliar.
- Imprimir los datos de este paquete auxiliar en un archivo de texto para su posterior análisis.

El programa escrito cuenta con tres archivos de encabezado y tres archivos fuente. Estos archivos se adjuntan en el Anexo F. En el encabezado comandos.h se definen los paquetes de mensajes enviados y recibidos, como también el paquete auxiliar que guarda los datos para imprimirlos una vez finalizada la lectura de datos. El encabezado mainwindow.h define las funciones y las variables utilizadas para programar el Scada de interfaz. Por último, en el encabezado comunicacion.h se definen las funciones y variables utilizadas para la comunicación por puerto serie, como también el bucle encargado del proceso de lectura. Como característica particular del programa se tiene que este trabaja con dos hilos en paralelo. En un hilo se realizan todas las funciones correspondientes al Scada, mientras que en otro hilo se realizan las funciones correspondientes a la lectura de datos. Cabe aclarar que se definieron longitudes de mensajes fijos para facilitar y estandarizar el procesamiento de los mismos. Así se definieron los mensajes enviados con una longitud de 10 bytes y los mensajes recibidos con una longitud de 4 bytes.

En la Figura 5.10 se expone la interfaz diseñada indicando la funcionalidad de sus elementos.

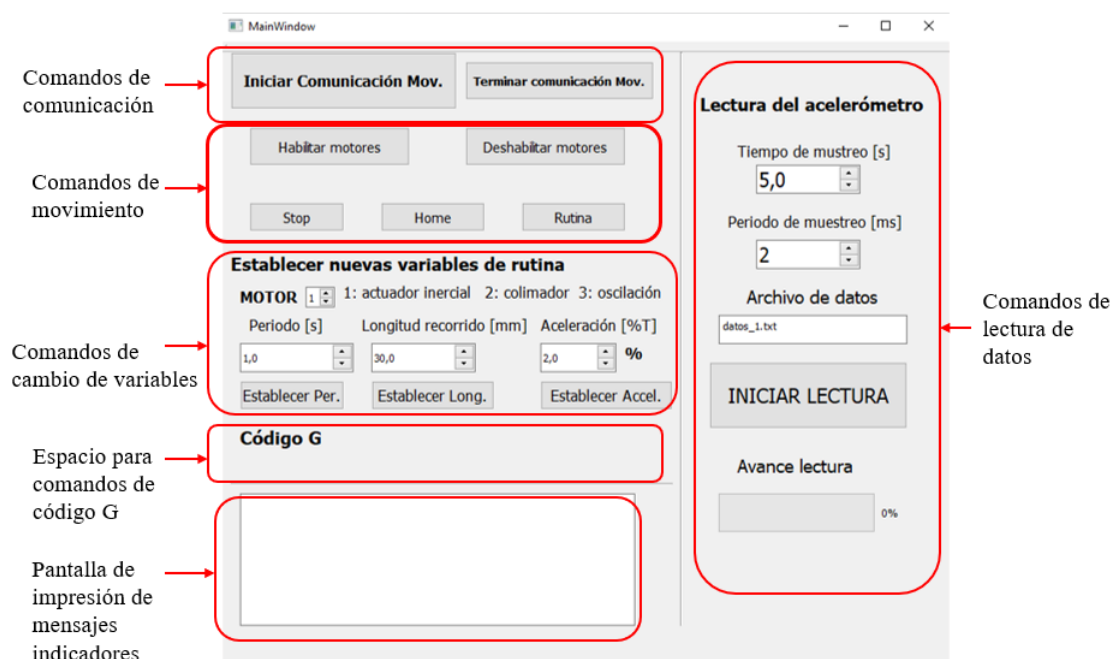


Figura 5.10: Interfaz diseñada para el control de los movimientos y lecturas por parte del usuario.

Los comandos de comunicación inician o terminan la comunicación por puerto serie a través del hilo correspondiente al Scada. Además estos comandos habilitan los comandos correspondientes a los motores al iniciar la comunicación y los deshabilitan al finalizarla. Los comandos de movimiento envían los mensajes que establecen los distintos tipos de movimiento según lo descrito en la sección anterior. Los comandos de cambio de variables permiten establecer nuevos valores de período, longitud de recorrido y porcentaje de aceleración para la rutina de cada motor. Estas se las debe establecer una por una. También se definió un espacio para los comandos de código G que lamentablemente no se llegaron a programar por falta de tiempo y priorizando otras actividades como la medición.

Del lado izquierdo de la pantalla se tienen los comandos de lectura de datos. Con estos se pueden establecer en primer lugar el tiempo y el período de muestreo que se desea como el nombre del archivo donde se quieren imprimir los datos. Luego, mediante el pulsador de iniciar lectura se cierra el puerto serie en el hilo del Scada y se pasa al hilo correspondiente a la lectura de datos. En este hilo se abre el puerto serie y se envía en primer lugar el mensaje que inicializa la lectura del acelerómetro en el microcontrolador y establece el valor del período de muestreo. Luego se inicia el bucle de lectura que se corta al llegar al número de muestras dado por el tiempo y período de muestreo, o se fuerza su finalización pulsando el botón terminar. Dentro de este bucle se leen los datos que envía el microcontrolador y se los almacena en el paquete auxiliar destinado a tal fin.

Adicionalmente se tiene una barra de avance que muestra el estado de la lectura en porcentaje de las muestras leídas frente a las objetivo. Cuando se termina el bucle se imprime los datos almacenados en el archivo deseado y se vuelve al hilo del Scada previo cierre del puerto serie en el hilo de lectura y reabrir el puerto serie en el hilo del Scada. Al volver al hilo del Scada se vuelve a estar disponible para ejecutar cualquier comando nuevamente.

En la Figura 5.12 se muestra nuevamente la interfaz diseñada, en este caso en situación de lectura. Por otro lado, en la Figura 5.11 se esquematiza un diagrama de las secuencias descritas anteriormente.

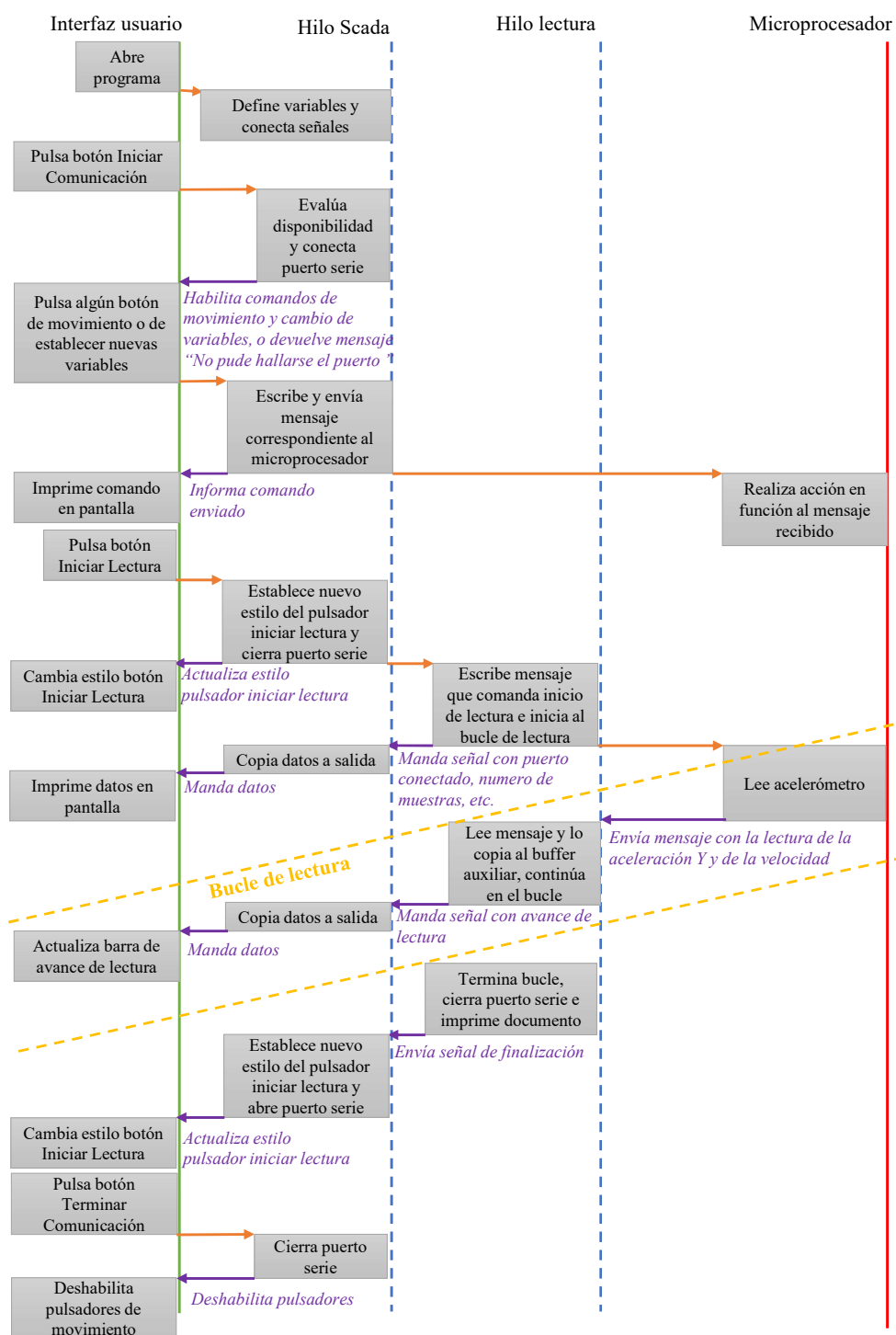


Figura 5.11: Diagrama de secuencias del código de la interfaz.

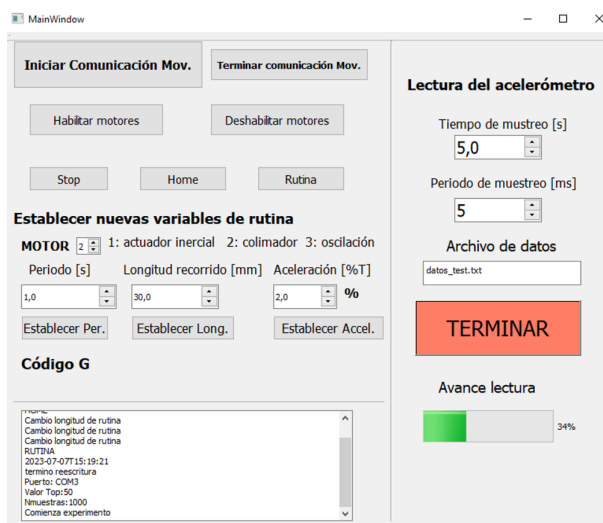


Figura 5.12: Interfaz diseñada para control de los movimientos por el usuario en funcionamiento.

5.4. Conclusión

En este capítulo se describió la instrumentación del prototipo colimador oscilante y el banco de ensayos. Se llegó a un conjunto de hardware y software funcionales con una interfaz que permite al usuario comandar el dispositivo pudiendo realizar un proceso de posicionamiento y un proceso de rutina a la cual se le puede ajustar las distintas variables. Además permite la medición de un acelerómetro analógico montado en la plataforma del banco de ensayos con el objetivo de poder realizar distintos ensayos experimentales y un futuro control activo del actuador inercial.

Como trabajos a futuro queda pendiente programar, por un lado, la lógica del código G en la interfaz para aprovechar la lógica programada en el microcontrolador. Por otro lado, queda pendiente diseñar y programar el control activo del actuador inercial dentro de los programas actuales.

Capítulo 6

Ensayos experimentales

“En teoría, no existe diferencia entre teoría y práctica; en la práctica sí la hay ”

— L. A. Van De Snepscheut

6.1. Introducción

Para culminar con este proyecto integrador se realizaron ensayos experimentales agrupando todo lo desarrollado en los capítulos anteriores. Se montó el prototipo colimador oscilante sobre el banco de pruebas y se lo instrumentó como se describió en el Capítulo 5. En la Figura 6.1 se puede observar la disposición del conjunto montado con el cual se realizaron los ensayos.

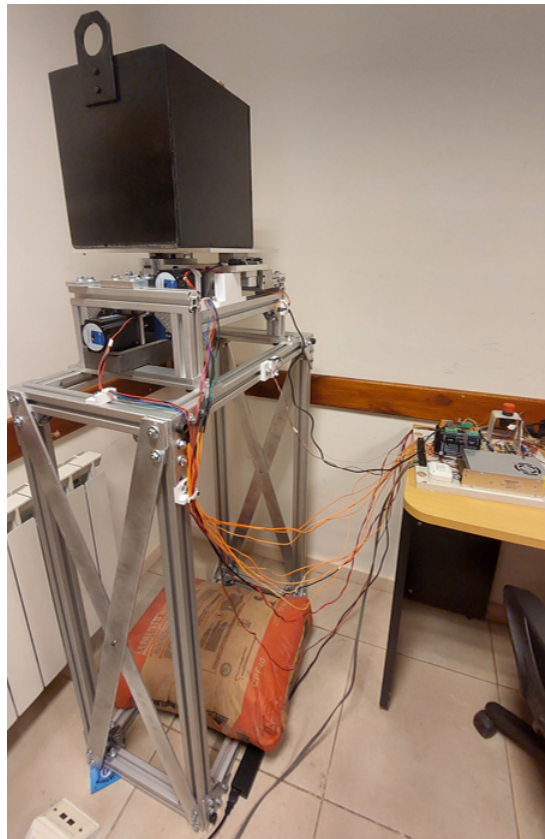


Figura 6.1: Disposición experimental del conjunto banco de pruebas con colimador oscilante instrumentado.

En primer lugar se decidió medir estáticamente la rigidez del dispositivo. Luego se midió la respuesta del sistema ante un impulso para poder determinar la frecuencia natural del mismo. Por último se midieron y compararon las aceleraciones y desplazamientos de la plataforma del banco de pruebas con la oscilación del colimador sin y con la actuación del actuador en contrafase.

Por falta de tiempo no se llegaron a realizar ensayos más exhaustivos y profundos sobre el sistema, al igual que tampoco se alcanzó a implementar un sistema de control activo. La falta de tiempo responde a varios factores. Por un lado se demoró más de lo planificado la fabricación del prototipo colimador oscilante y se tuvieron que resolver muchos problemas técnicos durante su montaje. Por otro lado, se demoró en llegar a un programa capaz de leer el acelerómetro implementado sin modificar la dinámica de los movimientos.

En cuanto a los problemas técnicos mencionados se destacan los siguientes:

- Falta de rectitud del tornillo correspondiente al husillo de bolas del actuador inercial: al momento del montaje se encontró que este tornillo presentaba cierta curvatura. Esto implicó un aumento de esfuerzos aplicados a este y en consecuencia un mayor torque requerido al motor que lo acciona. Por esta razón es que se debió disminuir el desplazamiento del actuador inercial de 80 mm (el despla-

miento teórico requerido según la conservación de masa por desplazamiento) a 30 mm.

- Juego entre el eje pivote y la base de oscilación: Al momento de ensamblar estos componentes se identificó un excesivo juego entre el eje pivote y la base de oscilación a la cual el eje debería estar firmemente fijado. Se intentó minimizar esto introduciendo láminas de bronce en el huelgo, algo que minimizó el juego pero no lo eliminó por completo. Este juego produce no linealidades en el sistema, lo cual aleja la realidad del modelo.

6.2. Metodología

En primer lugar se buscó determinar la constante de rigidez del banco de pruebas de manera estática. Para esto se traccionó el banco de pruebas desde su plataforma con diferentes fuerzas medidas con un dinamómetro. Además se posicionó firmemente un comparador centesimal para medir el desplazamiento de la plataforma a la misma altura a la cual se aplica la fuerza. En la Figura 6.2 se expone el arreglo experimental mencionado.

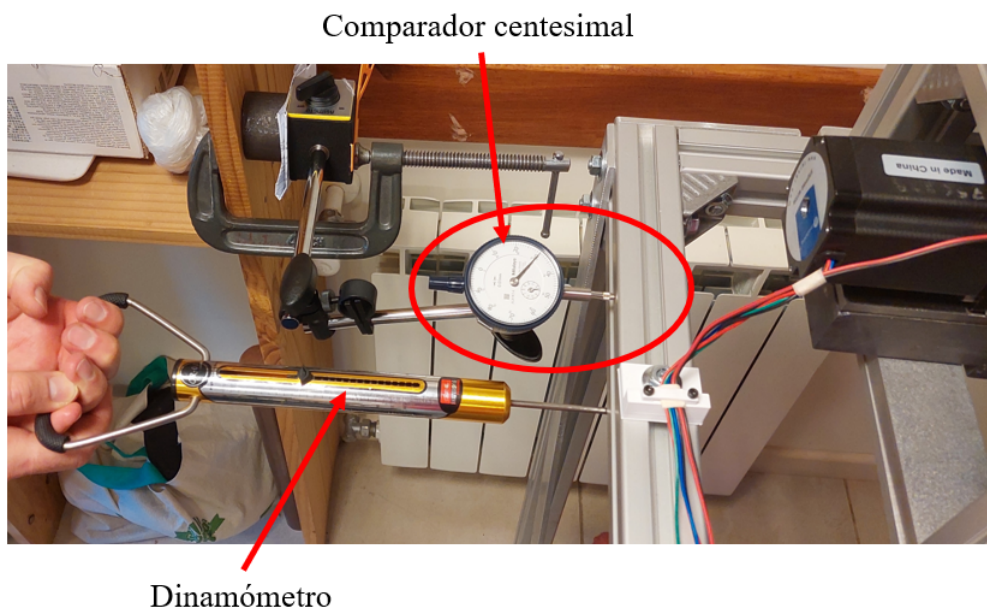


Figura 6.2: Arreglo experimental para la medición estática de la rigidez del banco de pruebas.

Se midió 5 veces el desplazamiento traccionando primero con una fuerza de 2 ± 1 kgf, luego con 4 ± 1 kgf, luego con 5 ± 1 kgf, luego 7 ± 1 kgf y finalizando con 10 ± 1 kgf. A estas mediciones se las promedió y se calculó un desvío estándar de las mismas para cada fuerza aplicada. Posteriormente se realizó una regresión lineal de los 4 promedios obtenidos y mediante su pendiente se obtuvo la rigidez del banco de pruebas.

Para calcular la respuesta del sistema ante el impulso se utilizó la instrumentación montada y sin iniciar movimiento alguno de los motores se suministró un impulso mediante un golpe en la plataforma del banco de pruebas en la dirección del movimiento principal. Este golpe se lo suministró instantáneamente luego de iniciar la lectura del acelerómetro. Así se obtuvo un documento con los datos de la aceleración medida con una frecuencia de muestreo de 500 Hz durante 5 segundos.

Cabe aclarar que para normalizar las señales se consideró la constante de conversión de $100 \frac{\text{cuentas}}{\frac{\text{m}}{\text{s}^2}}$ obtenida de la hoja de datos del acelerómetro [18].

A estas mediciones se las analizó con un código programado en Octave el cual se adjunta en el Anexo G. Con este código se graficaron las aceleraciones sin filtrar y filtradas con un filtro pasa alto de primer orden con frecuencia de corte de 3 Hz y un filtro pasa bajo de primer orden con frecuencia de corte de 50 Hz. Superponiendo estas dos gráficas se verificó que la señal filtrada representa correctamente las características de la gráfica sin filtrar. Además, con estas mediciones, se obtuvo un periodograma con el cual se analizó el espectro de frecuencias de la curva sin filtrar y así se obtuvo la frecuencia natural del dispositivo.

Para finalizar se midieron las aceleraciones haciendo oscilar el colimador oscilante con y sin el accionamiento del actuador, pero siempre sin actuar el movimiento angular del colimador. Para esto se estableció en primer lugar un recorrido de 0 mm para el motor correspondiente al actuador del movimiento angular (motor 3). Se realizó lo mismo con el motor 1 para desactivar el actuador inercial. Moviendo únicamente el colimador se midieron entonces las aceleraciones de la plataforma del banco de pruebas con un período de 2 ms y un tiempo de lectura de 5 s. Luego se activó el actuador moviéndolo en contrafase con el colimador y se midieron las aceleraciones con el mismo período y tiempo de lectura. Cabe aclarar que siempre se midieron estos con una frecuencia de los movimientos de 1 Hz. Además, para ambos desplazamientos se determinó una longitud de desplazamiento de 20 mm ya que, aplicando una longitud de 30 mm como la inicialmente propuesta, aparecía un desfase entre ambos movimientos debido a la dinámica de la electrónica del microcontrolador.

A estas mediciones se las analizó con el mismo código programado en Octave con los mismos filtros indicados anteriormente. Superponiendo estas dos gráficas se verificó que la señal filtrada representa correctamente las características de la gráfica sin filtrar. Los datos filtrados se los integraron 2 veces para obtener así el perfil de desplazamiento de la plataforma del banco de pruebas. Para finalizar se compararon los valores máximos de aceleración y desplazamiento buscando observar una disminución de estos al activar el movimiento del actuador en contrafase.

6.3. Resultados

Los promedios de los datos obtenidos durante las mediciones de la rigidez del banco de pruebas se exponen en la Figura 6.3. En la misma figura se muestra la regresión lineal obtenida con sus parámetros. De la pendiente de esta regresión se obtuvo una rigidez del banco de pruebas de $168 \pm 5 \frac{N}{mm}$. Como se puede observar esta rigidez es levemente menor a la rigidez teórica de $178,6 \frac{kN}{m}$ obtenida durante la revisión del diseño del banco de pruebas. Esto se puede deber a que el banco no se encontró apoyado correctamente en los vértices de la base como se modeló mediante elementos finitos.

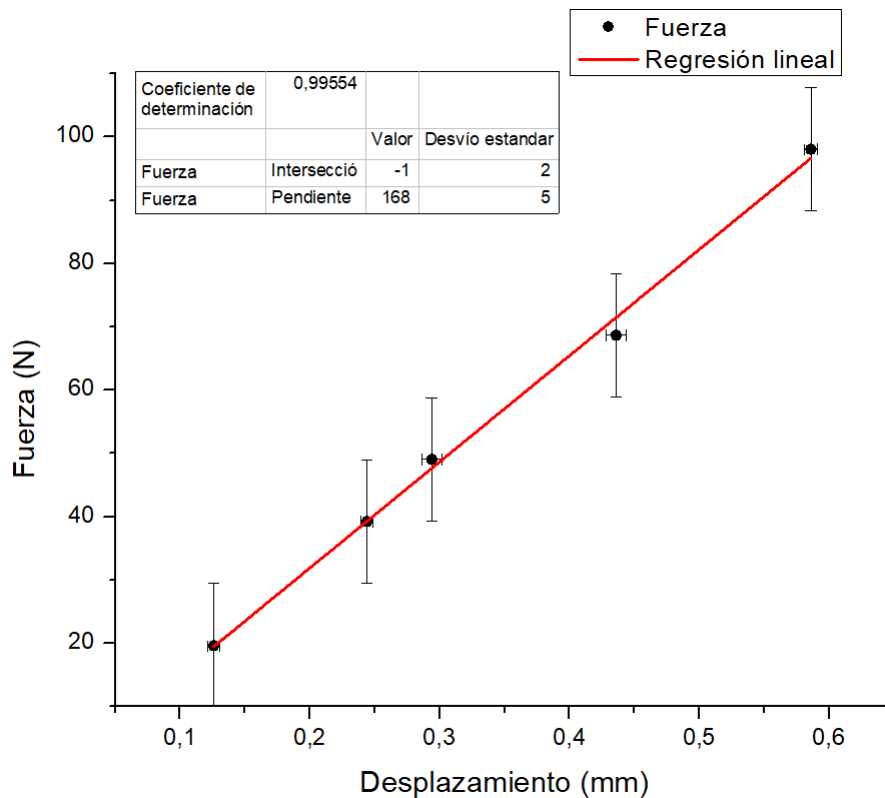


Figura 6.3: Desplazamiento vs fuerza aplicada obtenidos para la determinación de la rigidez del banco de pruebas.

Por otro lado, en la Figura 6.4 se exponen la curva de aceleración (A_y) medida y su correspondiente curva filtrada ($A_{y,f}$). En la Figura 6.5 se observa un detalle de la misma. Como se puede observar la curva filtrada sigue aceptablemente la curva sin filtrar. Se puede observar que en los datos medidos se encuentra una gran cantidad de ruido. Se intuye que este ruido puede corresponder a ruido eléctrico dado por la cercanía de la fuente switching y los motores.

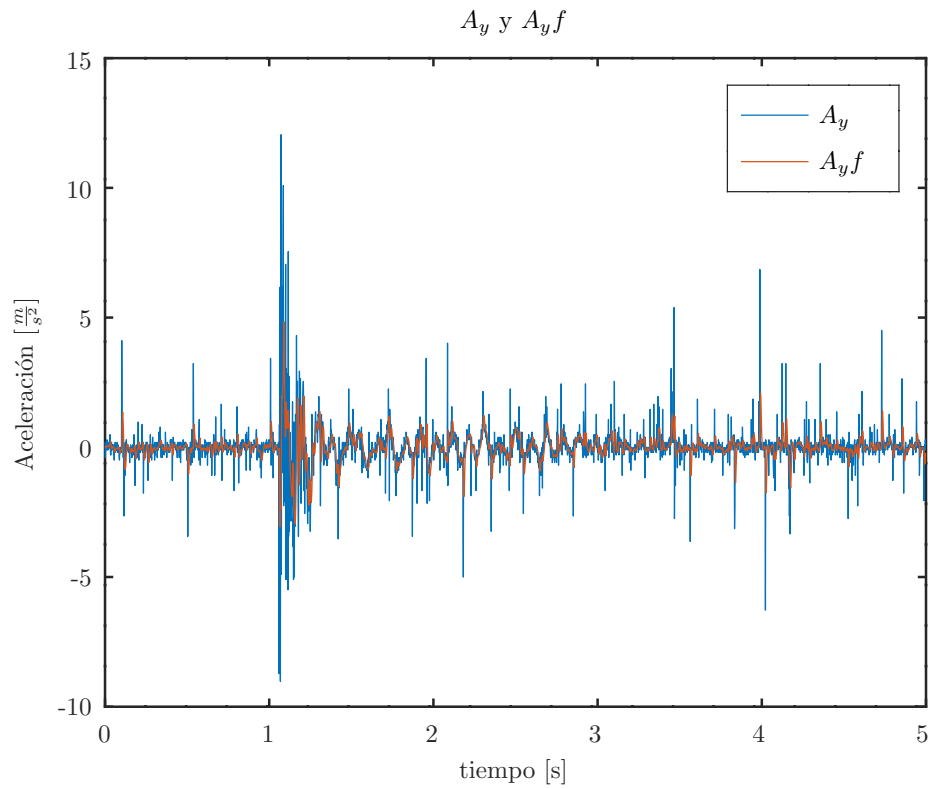


Figura 6.4: Aceleraciones medidas en la plataforma del banco de pruebas ante un impulso inicial en el tiempo aproximado 1,1s.

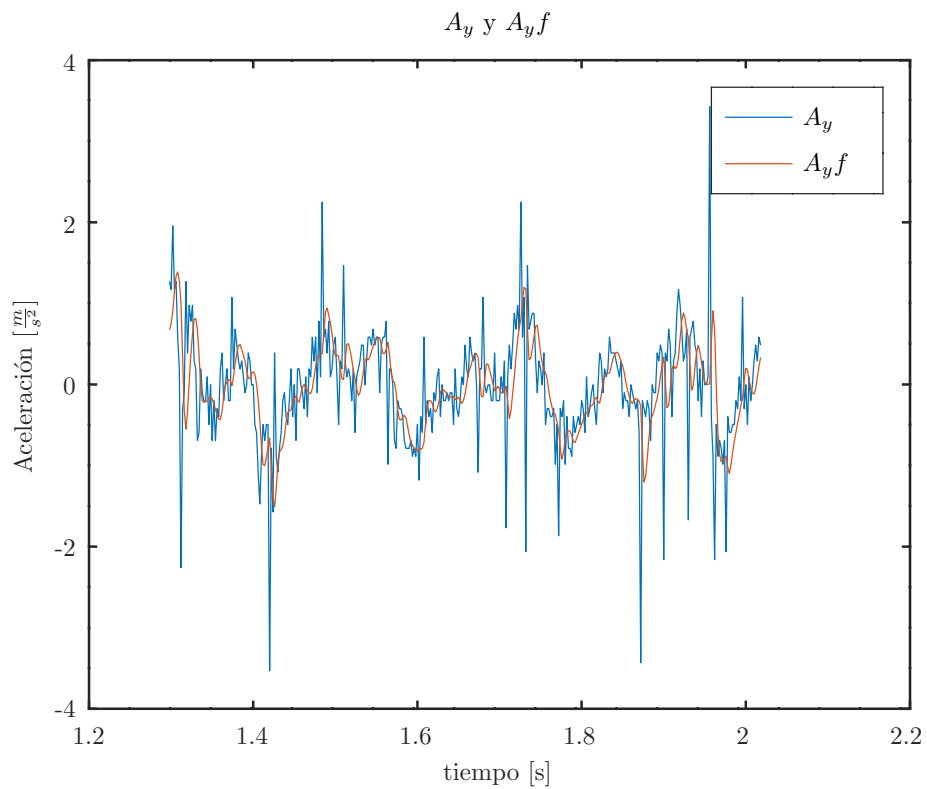


Figura 6.5: Detalle de las aceleraciones medidas en la plataforma del banco de pruebas ante un impulso inicial en el tiempo aproximado 1,1s.

Al analizar el periodograma de la curva original se obtuvo el gráfico mostrado en la Figura 6.6. De este espectro se observan dos picos, el primero a una frecuencia de 4,7 Hz y el segundo a una frecuencia de 10,5 Hz. Se concluyó entonces que la primer frecuencia natural del sistema es 4,7 Hz, mientras que la frecuencia de 10,5 Hz puede deberse a frecuencias transversales o torsionales.

Se puede apreciar entonces que la primer frecuencia natural obtenida es menor a la calculada en el modelado dinámico del sistema en el Capítulo 4. Esto se debe a que en el modelo se consideró todas uniones rígidas entre las columnas y un empotramiento del banco al suelo, condiciones que no se cumplen en la realidad. Con esto se sobrestimó la rigidez del sistema en el modelo y se justifica que la frecuencia natural medida sea menor que la obtenida con el modelo.

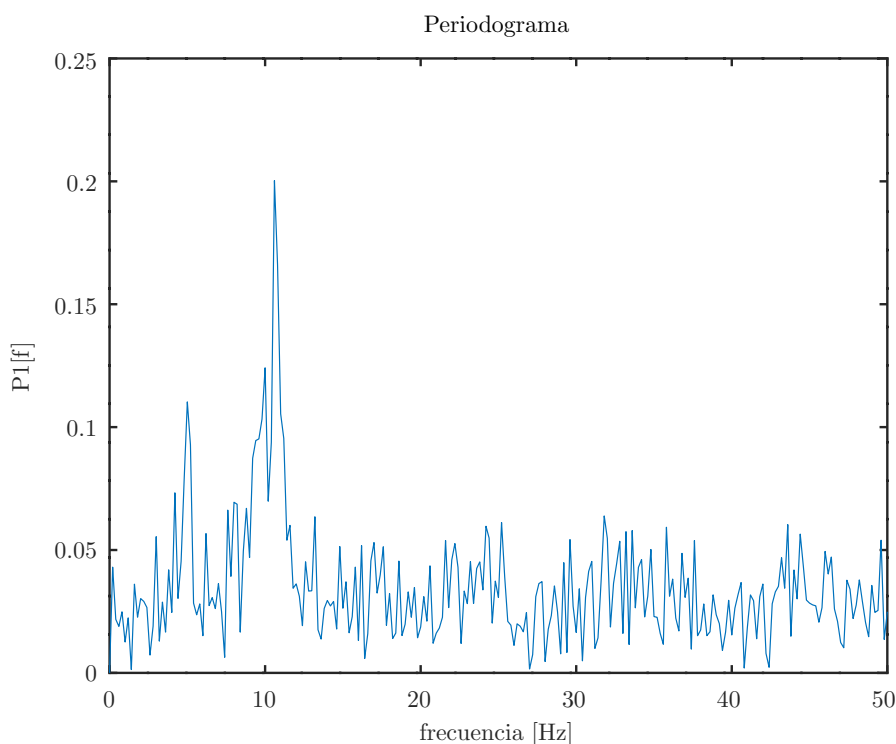


Figura 6.6: Periodograma obtenido del banco de pruebas ante un impulso inicial.

Luego se obtuvo el perfil de aceleraciones con y sin filtro y el perfil de desplazamientos filtrado de la plataforma del banco de pruebas actuando únicamente el colimador. En la Figura 6.7 se expone justamente el perfil de aceleraciones obtenido, mientras que en la Figura 6.9 se expone el perfil de desplazamientos obtenido. De estos gráficos se obtuvo una aceleración máxima de $2,3 \frac{m}{s^2}$ y un desplazamiento máximo de $1,4 \text{ mm}$.

Para finalizar se obtuvo el perfil de aceleraciones con y sin filtro y el perfil de desplazamientos filtrado de la plataforma del banco de pruebas actuando el colimador y el actuador en contrafase. En la Figura 6.8 se expone justamente el perfil de aceleraciones obtenido, mientras que en la Figura 6.10 se expone el perfil de desplazamientos obteni-

do. De estos gráficos se obtuvo una aceleración máxima de $2,3 \frac{m}{s^2}$ y un desplazamiento máximo de $0,76 \text{ mm}$.

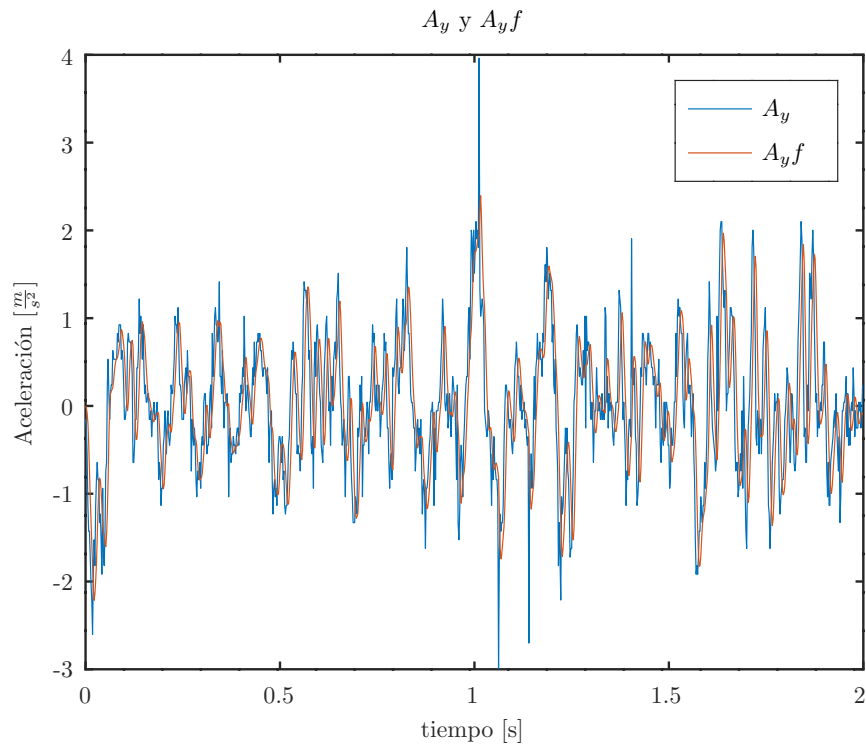


Figura 6.7: Perfil de aceleración medido con el movimiento del colimador sin el actuador.

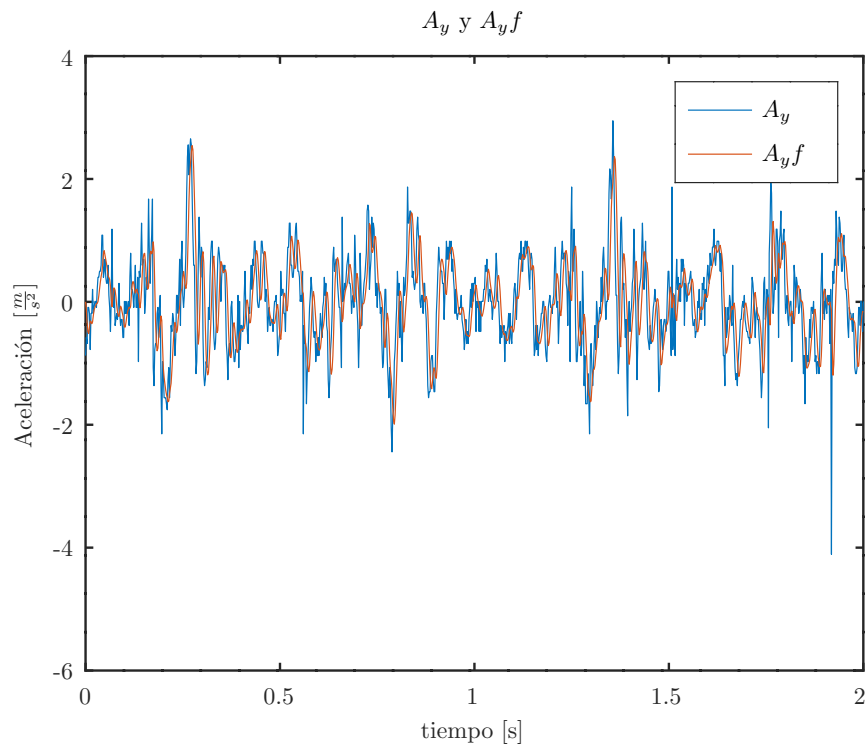


Figura 6.8: Perfil de aceleración medido con el movimiento del colimador más el actuador en contrafase.

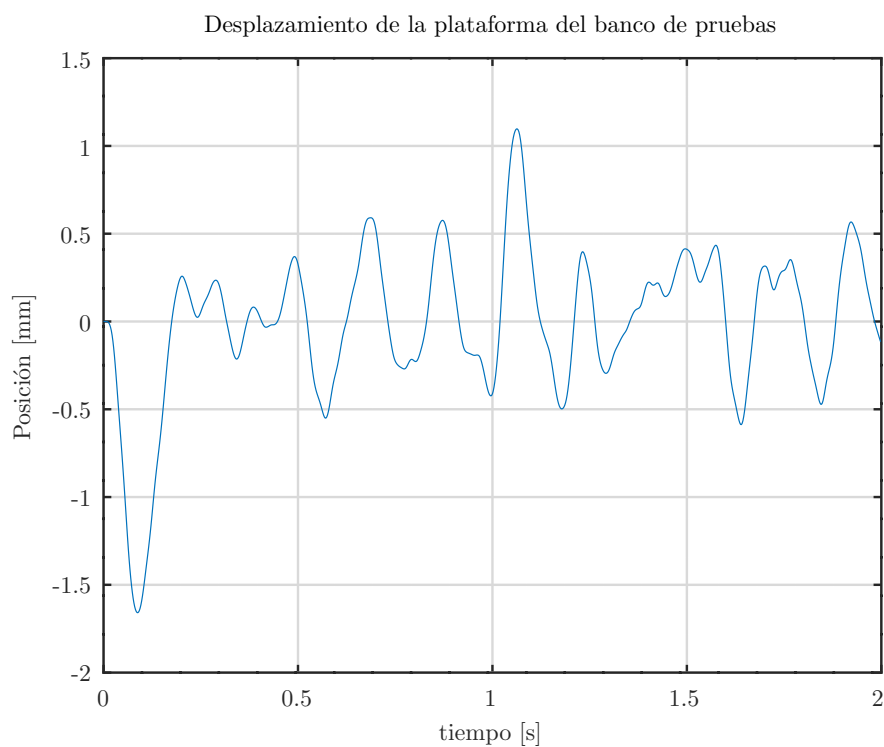


Figura 6.9: Perfil de desplazamiento obtenido con el movimiento del colimador sin el actuador.

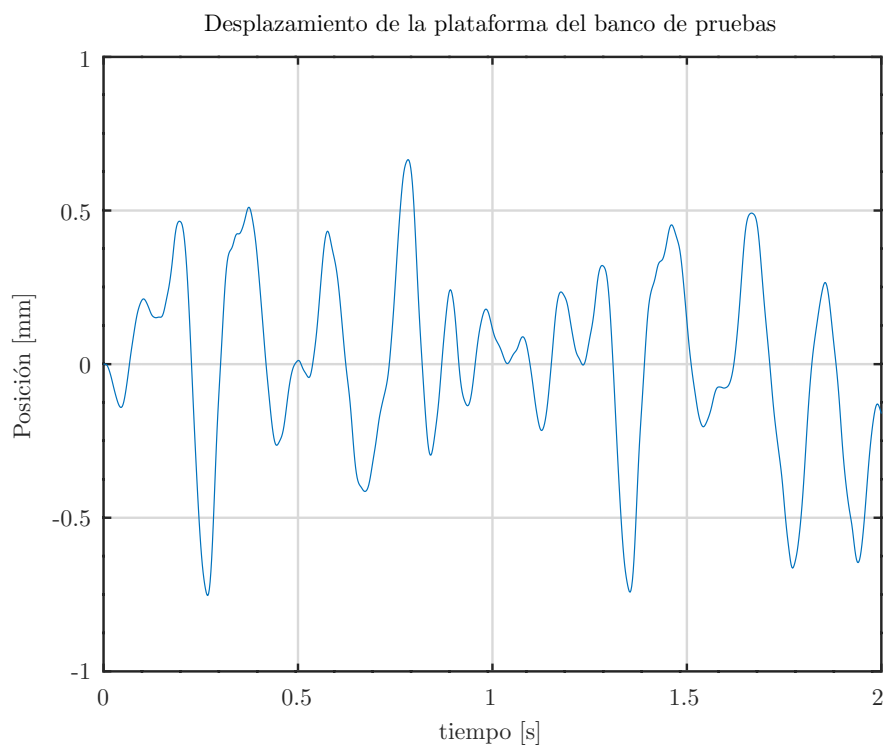


Figura 6.10: Perfil de desplazamiento obtenido con el movimiento del colimador más el actuador en contrafase.

De los valores obtenidos no se observa una disminución significativa de las acelera-

ciones medidas. Esto se puede deber a la gran cantidad de ruido en la medición de las mismas. Sin embargo, si se observa una disminución del 46 % en los desplazamientos máximos comparable con la disminución del 38 %, resultado del modelo del Capítulo 4, considerando el ruido en las mediciones y el método de filtrado. Además los desplazamientos medidos también son similares a los obtenidos con el modelo bajando de 1,13 mm a 0,70 mm en el modelo y de 1,4 mm a 0,76 mm en las mediciones. Esto muestra que es posible amortiguar las vibraciones que produce el movimiento del colimador mediante la actuación del actuador inercial en contrafase.

6.4. Conclusión

De los ensayos experimentales realizados se obtuvo la rigidez del banco de pruebas, su primer frecuencia natural y el perfil de aceleraciones y desplazamientos de la plataforma del banco de ensayos al actuar el colimador, y comparando los resultados con y sin el accionamiento del actuador inercial en contrafase.

Con la metodología propuesta para calcular la rigidez estática del banco de pruebas se obtuvo una rigidez del banco de pruebas de $168 \pm 5 \frac{N}{mm}$. Esta rigidez es levemente menor a la rigidez teórica obtenida durante la revisión del diseño del banco de pruebas. Esto se puede deber a que el banco no se encontró apoyado correctamente en los vértices de la base como se modeló mediante elementos finitos.

Continuando se obtuvo una frecuencia natural del banco de pruebas de 4,7 Hz. Esta frecuencia es levemente menor a la obtenida en el modelado, esto se debe a que en el modelo se consideró todas uniones rígidas entre las columnas y un empotramiento del banco al suelo, condiciones que no se cumplen en la realidad. Con esto se sobrestimó la rigidez del sistema en el modelo y se justifica que la frecuencia natural medida sea menor que la obtenida con el modelo.

Para finalizar, no se observó una disminución significativa de las aceleraciones medidas. Esto se puede deber a la gran cantidad de ruido en la medición de las mismas. Sin embargo, si se observó una disminución del 46 % en los desplazamientos máximos comparable con la disminución del 38 % resultado del modelo. Además los desplazamientos medidos también son similares en valor a los obtenidos con el modelo. Esto muestra que es posible amortiguar las vibraciones que produce el movimiento del colimador mediante la actuación del actuador inercial en contrafase.

Como tareas a futuro se propone realizar ensayos más exhaustivos, para luego poder implementar un sistema de control activo del dispositivo.

Capítulo 7

Conclusiones generales y trabajos a futuro

“Es más divertido llegar a una conclusión que justificarla.”

— Malcolm Forbes

7.1. Conclusiones

En el marco del proyecto del Laboratorio Argentino de Haces de Neutrones (LAHN) se desarrolló un prototipo del colimador oscilante para el difractor multipropósito ANDES. Las oscilaciones del colimador requeridas para su funcionamiento, junto a su considerable masa, producen fuerzas inerciales capaces de producir vibraciones en la estructura que lo soporta, las cuales pueden afectar a la electrónica de detección y al detector. Para controlar y disminuir estas vibraciones se le implementó al prototipo un actuador inercial el cual, mediante la aceleración de su masa, busca compensar las fuerzas inerciales del colimador.

En este proyecto integrador se propuso diseñar, implementar y controlar un banco de pruebas para el control de vibraciones del prototipo mencionado. En primer lugar se verificaron los componentes críticos del dispositivo. Luego se diseñó el banco de pruebas desde su concepto hasta los planos de fabricación. Para poder anticipar ciertas características dinámicas del banco de pruebas con el prototipo montado como conjunto se realizó un modelado dinámico del mismo. Posteriormente se procedió a la instrumentación del conjunto, esto implicó la instrumentación física y la programación del microcontrolador y la interfaz necesaria. Finalizando se realizaron algunos ensayos experimentales con el dispositivo.

Para la verificación analítica de los componentes críticos del prototipo colimador oscilante se realizaron cálculos considerando un perfil de velocidades trapezoidal en el

movimiento. Los componentes críticos que se verificaron incluyen el tornillo del husillo de bolas del colimador, el tornillo del husillo de bolas del actuador inercial, las guías lineales del colimador y del actuador, el eje pivote y sus rodamientos. Los factores de seguridad se calcularon siguiendo los catálogos de fabricantes de los componentes comerciales y se determinaron los esfuerzos a los que están sometidos los elementos mediante mecánica de sólidos, utilizando diagramas de cuerpo libre y traslación de fuerzas y momentos. Se encontró que todos los factores de seguridad obtenidos cumplen con los criterios de aceptación para cada caso. Para el peor caso de factor de seguridad, se propuso una mejora para evitar concentración de tensiones.

Para el diseño del banco de pruebas se consideró una frecuencia natural de 5 Hz para la estructura soporte del colimador oscilante y el detector. Con esto y la masa de este conjunto se determinó la rigidez que tendría esta estructura. Con ella se determinó el largo que las columnas del banco de pruebas deben tener para que el banco tenga la misma rigidez en la dirección del movimiento principal del colimador y el actuador, y rigidizando la estructura en el sentido transversal. Al medir experimentalmente que la rigidez obtenida es menor a la estimada se revisó el diseño del banco utilizando la técnica de elementos finitos y se determinó una rigidez teórica corregida de este.

En el análisis del conjunto del banco de pruebas con el prototipo colimador oscilante montado, se evaluó la reducción de vibraciones transmitidas al banco de pruebas tanto sin el movimiento del actuador inercial como con el movimiento en contrafase del actuador. La respuesta dinámica del sistema se calculó utilizando el desacoplamiento modal y el método numérico Newmark.

Se probaron dos configuraciones de compensación de actuador inercial, siendo más prometedora la configuración en la cual el actuador inercial realizaba una mayor fuerza. Los mejores resultados fueron una reducción de 98 % en el desplazamiento máximo, y un 71,3 % en la aceleración máxima. Esta alta disminución del desplazamiento máximo podría inducir a que simplemente sincronizando los movimientos se puede llegar a obtener un control de las vibraciones aceptable sin necesidad de realizar un control a lazo cerrado. Sin embargo, para confirmar esto se requieren más estudios. Es importante mencionar que debido a limitaciones físicas, no fue posible alcanzar el desplazamiento del actuador de 80 mm a una frecuencia de 1 Hz, y la longitud máxima alcanzada fue de los 30 mm analizados inicialmente.

Durante la instrumentación del dispositivo se llegó a un conjunto de hardware y software funcionales con una interfaz que permite al usuario comandar el dispositivo pudiendo realizar un proceso de posicionamiento y un proceso de rutina a la cual se le pueden ajustar las distintas variables. Además permite la medición de un acelerómetro analógico montado en la plataforma del banco de ensayos con el objetivo de poder realizar distintos ensayos experimentales y un futuro control activo del actuador inercial.

Con los ensayos finalmente realizados se obtuvo la rigidez del banco de pruebas, su primer frecuencia natural y el perfil de aceleraciones y desplazamientos de la plataforma del banco de pruebas al actuar el colimador, y comparando los resultados con y sin el accionamiento del actuador inercial en contrafase. Se obtuvo una frecuencia natural del banco de pruebas de 4,7 Hz y una rigidez de $168 \pm 5 \frac{kN}{m}$.

Para finalizar, mediante la medición de las aceleraciones y desplazamientos durante dos ciclos de rutina se observó una disminución del 46 % de los desplazamientos máximos de la plataforma del banco de pruebas obteniendo valores comparables con los obtenidos en el modelado dinámico. Esto muestra que es posible amortiguar las vibraciones que produce el movimiento del colimador mediante la actuación del actuador inercial.

7.2. Trabajos a futuro

Quedan entonces como trabajos a futuro:

- Cambiar el tornillo del husillo de bolas que transmite movimiento al actuador: dado que este se encontró doblado al momento del ensamblaje, su no linealidad produce esfuerzos mayores a los esperados durante su recorrido. Esto aumentó el torque requerido al motor y disminuyó la longitud de desplazamiento posible a una frecuencia de 1 Hz. Con su reemplazo por uno equivalente o reemplazando el conjunto husillo de bolas por un tornillo de potencia se podrán realizar mayores desplazamientos y así proporcionar un mejor control de las vibraciones.
- Fijar firmemente el eje pivot en la base pivotante: dado que la interface entre el eje pivote y la base pivotante contó con un ajuste muy holgado se tuvieron juegos no deseados durante los movimientos que afectan a la dinámica del dispositivo. Estos huelgos también impidieron que el eje funcione verdaderamente como un pivote rozando la placa pivotante con su sistema de actuación. Por esto se propone fabricar nuevamente otro eje con un diámetro mayor y un mejor ajuste. Aprovechando esto se recomienda reemplazar los dos anillos de retención intermedios con sus alojamientos correspondientes por un anillo separador y así evitar la concentración de tensiones en la sección más solicitada.
- Programar la lógica para implementar el código G: para aprovechar la lógica programada en el microcontrolador se propone programar la lógica correspondiente en el código de la interfaz para poder controlar los movimientos del prototipo arbitrariamente mediante este código para cualquier ensayo o proceso que lo requiera.

- En caso de ser necesario diseñar el control activo de las vibraciones del colimador oscilante: para completar los objetivos iniciales se propone realizar ensayos exhaustivos del sistema para así poder modelar la planta y mediante la lectura del acelerómetro poder implementar un control activo con retroalimentación. Esto permitirá disminuir con mayor eficacia las vibraciones producidas por la oscilación del colimador mediante la actuación del actuador inercial.

Bibliografía

- [1] CNEA. Reactor multipropósito ra-10, 2023. URL <https://www.argentina.gob.ar/cnea/ra10>, 07/07/2023. 1
- [2] CNEA, 2023. URL <https://www.lahn.cnea.gov.ar/>, 07/07/2023. 1
- [3] Monteros, L. Descripción general del difractor andes. Inf. Téc. MD-LAHN-001/0200 Rev.: 0, CNEA, 2018. 2
- [4] Hamrock, B. J., Jacobson, B. O., Schmid, S. R., Hernández, A. E. G. Elementos de máquinas. 1^a ed^{ón}. Mexico: McGraw-Hill, 2000. 8
- [5] THK. Husillos de bolas, Catálogo general. URL <https://www.thk.com/catalog/?lang=es>, 26/07/2023. 9, 10, 11, 24
- [6] Norton, R. L. Diseño de máquinas: un enfoque integrado. 4^a ed^{ón}. Mexico: Pearson Education, 2011. 11, 12, 17, 18, 20
- [7] Gaes-Sistemas-Mecánicos. Guías Lineales, Información Técnica. URL <https://grupogaes.com/tienda/movimiento-lineal/guias-lineales/guias-lineales-serie-hg/>, 26/07/2023. 11, 15, 25
- [8] Lishui City Youright Precision Machinery Co., L. Ball Screw Single Nut Size Specification, SFU. URL <http://www.cnyouright.com>, 26/07/2023. 12
- [9] Simens, 2022. URL <https://solidedge.siemens.com/en/>, 26/06/2023. 13, 17
- [10] ArcelorMittal-Acindar. Productos para la Industria, catálogo de productos, información técnica. URL <https://www.acindar.com.ar/wp-content/uploads/2018/11/Catalogo-de-productos-para-la-industria.pdf>, 26/07/2023. 19
- [11] Budynas, R. G. DISEÑO EN INGENIERÍA MECÁNICA DE SHIGLEY. 8^a ed^{ón}. Mexico: McGraw-Hill, 2008. 19
- [12] SKF. Rodamientos, 2019. PUB BU/P1 17000/1 ES. 22, 27
- [13] Blevins, R. D. Formulas for Dynamics, Acoustics and Vibration. Wiley Series in Acoustics Noise and Vibration. Nashville, TN: John Wiley & Sons, 2015. 31

-
- [14] Bosch Rexroth, A. Elementos básicos de mecánica, 13.0. URL <http://docs.lineartec.com.ar/docs/sistemasestructurales.pdf>, 26/07/2023. 31
- [15] UniversidadPolitécnicodeCartagena. URL <https://www.upct.es/goe/software/mefi.php>, 25/06/2023. 33
- [16] Rao, S. S. Mechanical Vibrations in SI units. 6^a ed^{ón}. London, England: Pearson Education, 2017. 37, 39, 40
- [17] Spyder. URL <https://www.spyder-ide.org/>, 26/06/2023. 39
- [18] Analog Devices, I. Preliminary Technical Data ADXL335. URL <https://pdf1.alldatasheet.com/datasheet-pdf/download/250056/AD/ADXL335.html>, 26/07/2023. 50, 68
- [19] QT. URL <https://www.qt.io/product/development-tools>, 26/06/2023. 59

Agradecimientos

Quiero agradecer a la Comisión Nacional de Energía Atómica y al Instituto Balseiro por permitirme estudiar en esta institución y realizar este proyecto.

También quiero agradecer a mi director Santiago Pincin y a mi co-director Juan Carlos García por haberme guiado y ayudado incondicionalmente durante este trabajo. No solo son excelentes directores y profesionales, si no también excelentes personas.

Quiero agradecer también a los técnicos del taller de Termohidráulica y a los técnicos del Laboratorio de Ingeniería por la fabricación de las piezas y la provisión de materiales y herramientas, pero sobre todo, por su amabilidad y predisposición.

Me gustaría agradecer especialmente a mi mamá, Tania, a mi papá, Ingmar y a mi hermana, Tasia, por estar siempre presentes y apoyarme incondicionalmente en todo momento.

Por último gracias a todas aquellas personas que me acompañaron en esta etapa y de una forma u otra me ayudaron a llegar a donde estoy.

Apéndice A

Práctica Profesional Supervisada y actividades de proyecto y diseño

“La práctica es un maestro excepcional.”

— Cayo Plinio

A.1. Actividades Relacionadas a la Práctica Profesional Supervisada

- Estudio del sistema mecánico tratado, su función y las problemáticas asociadas.
- Verificación mecánica de componentes solicitados a esfuerzos dinámicos alternos, entre ellos las guías lineales, tornillos de bolas recirculantes, rodamientos y ejes.
- Diseño de un banco de pruebas para la evaluación del desempeño dinámico del sistema mecánico en cuestión.
- Modelado dinámico simplificado del conjunto. Resolución temporal del mismo por medio de algún método numérico. Análisis de la mejora en el desempeño por medio de compensación de vibraciones.
- Armado de las partes mecánicas y eléctricas del dispositivo prototipo Alpha.
- Instrumentación del dispositivo: esto incluye la realización de una electrónica que comanda el dispositivo, el cableado, y la programación de la misma junto con una interface de comunicación por computadora.
- Ensayos básicos experimentales del dispositivo, que incluye el análisis de resultados ante una entrada impulsiva, y en funcionamiento con y sin actuación de compensación de vibraciones.

A.2. Actividades de proyecto y diseño

Las actividades de Proyecto y Diseño que el alumno ha realizado en la tesis “Diseño, implementación y control de un banco de pruebas para el control de vibraciones de un colimador oscilante” tienen una base fuerte de ciencias básicas e ingeniería que fueron adquiridas durante la carrera. Las tareas realizadas fueron:

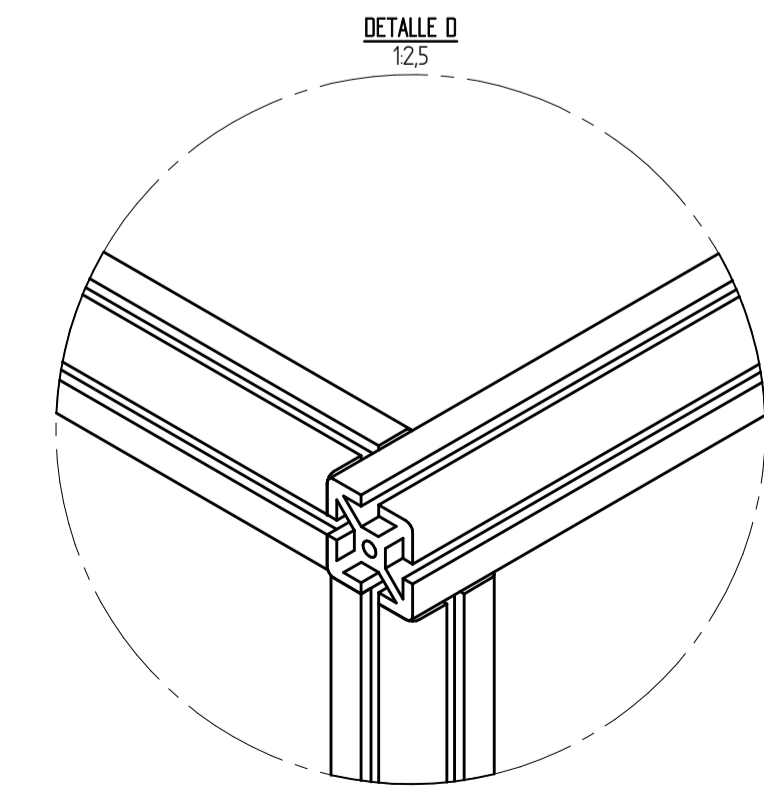
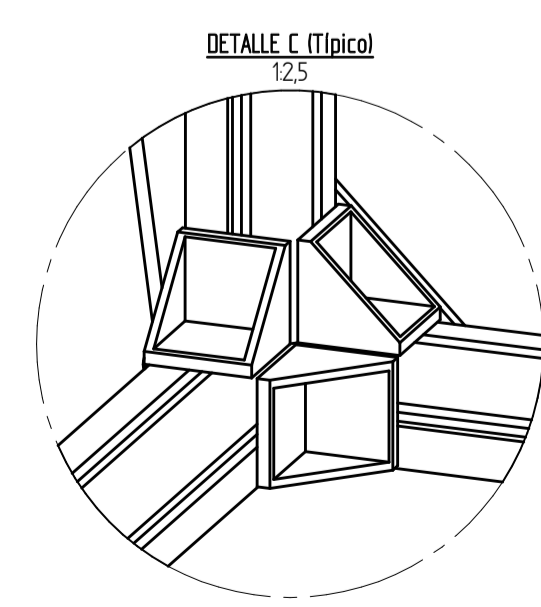
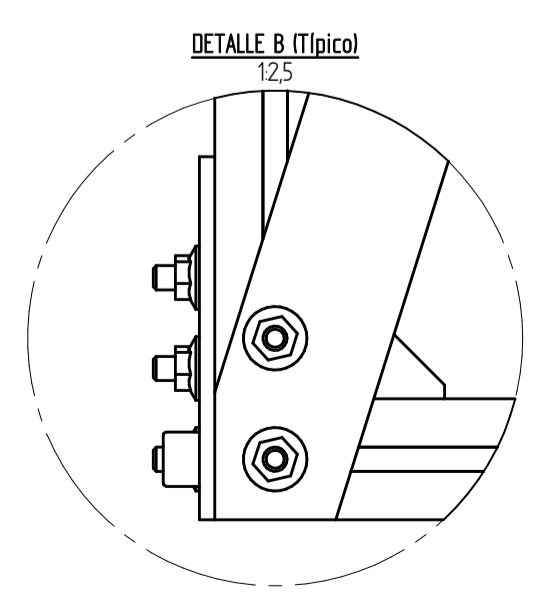
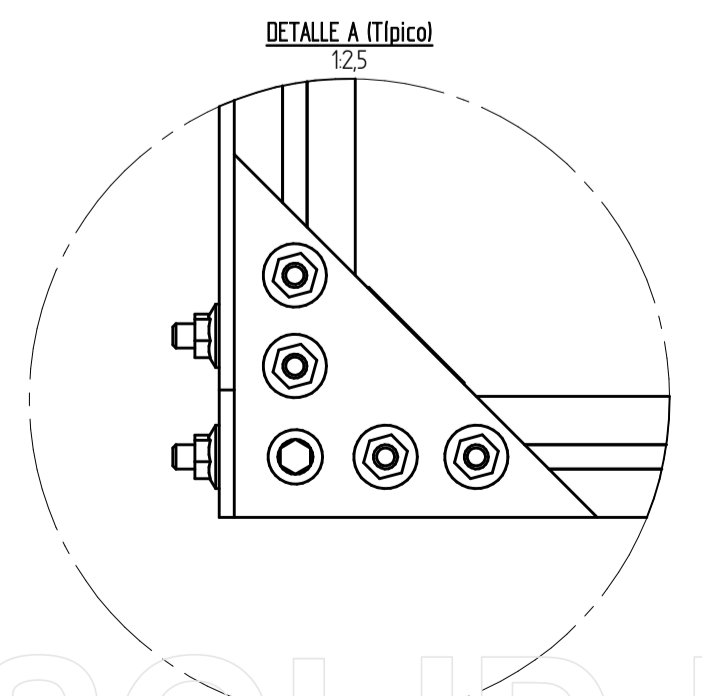
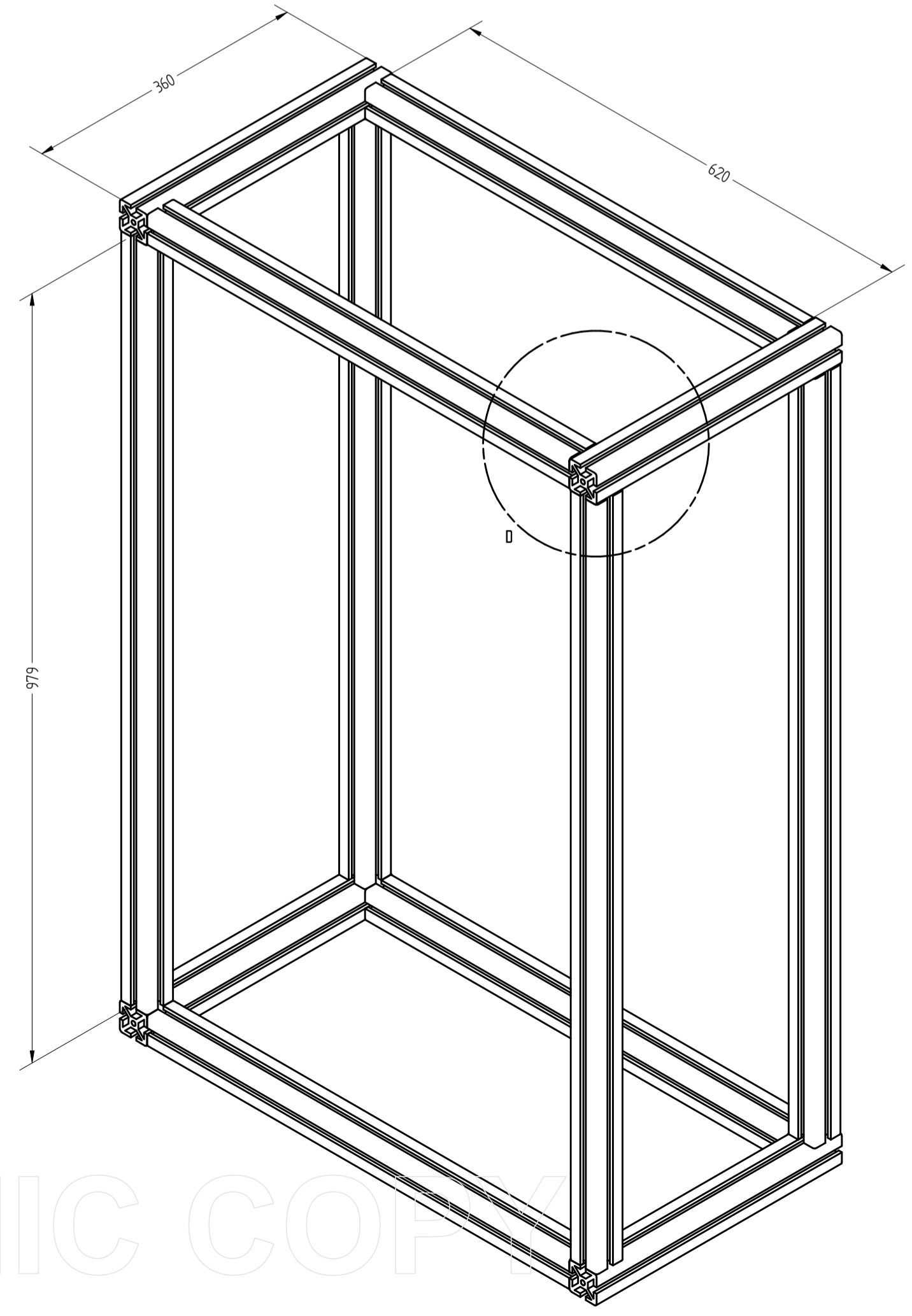
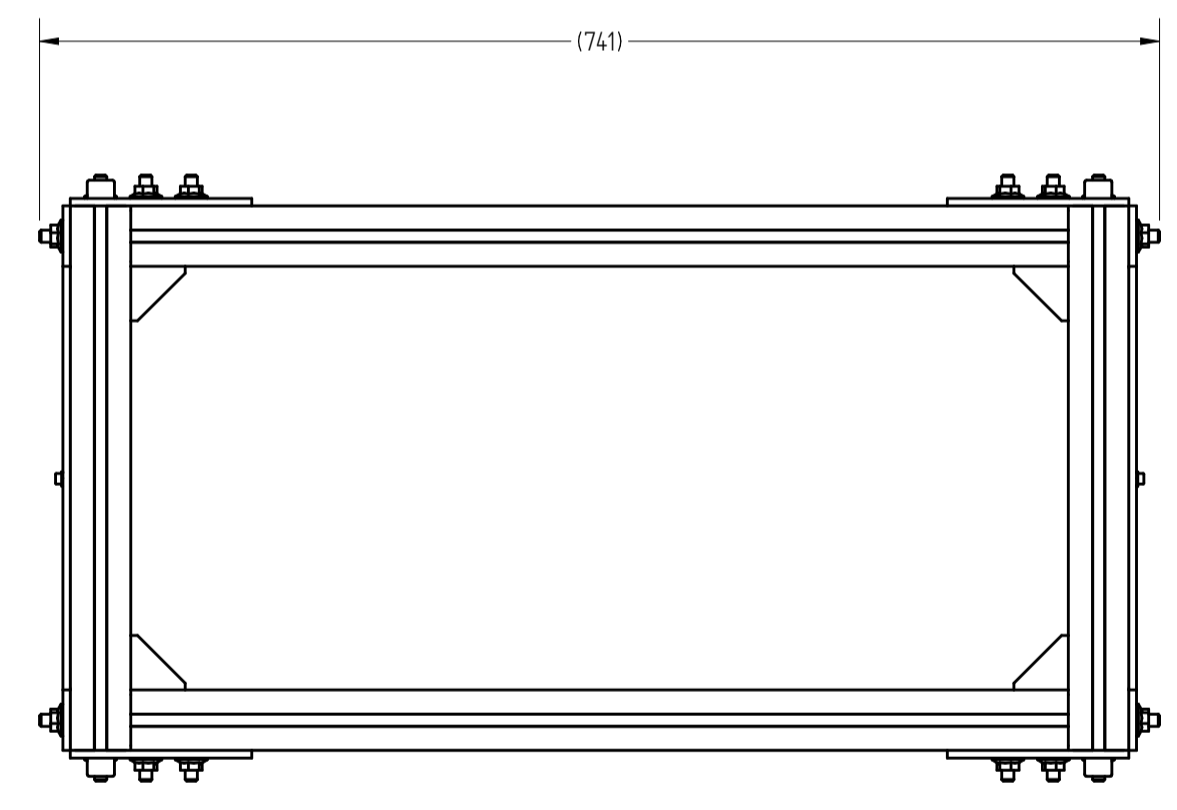
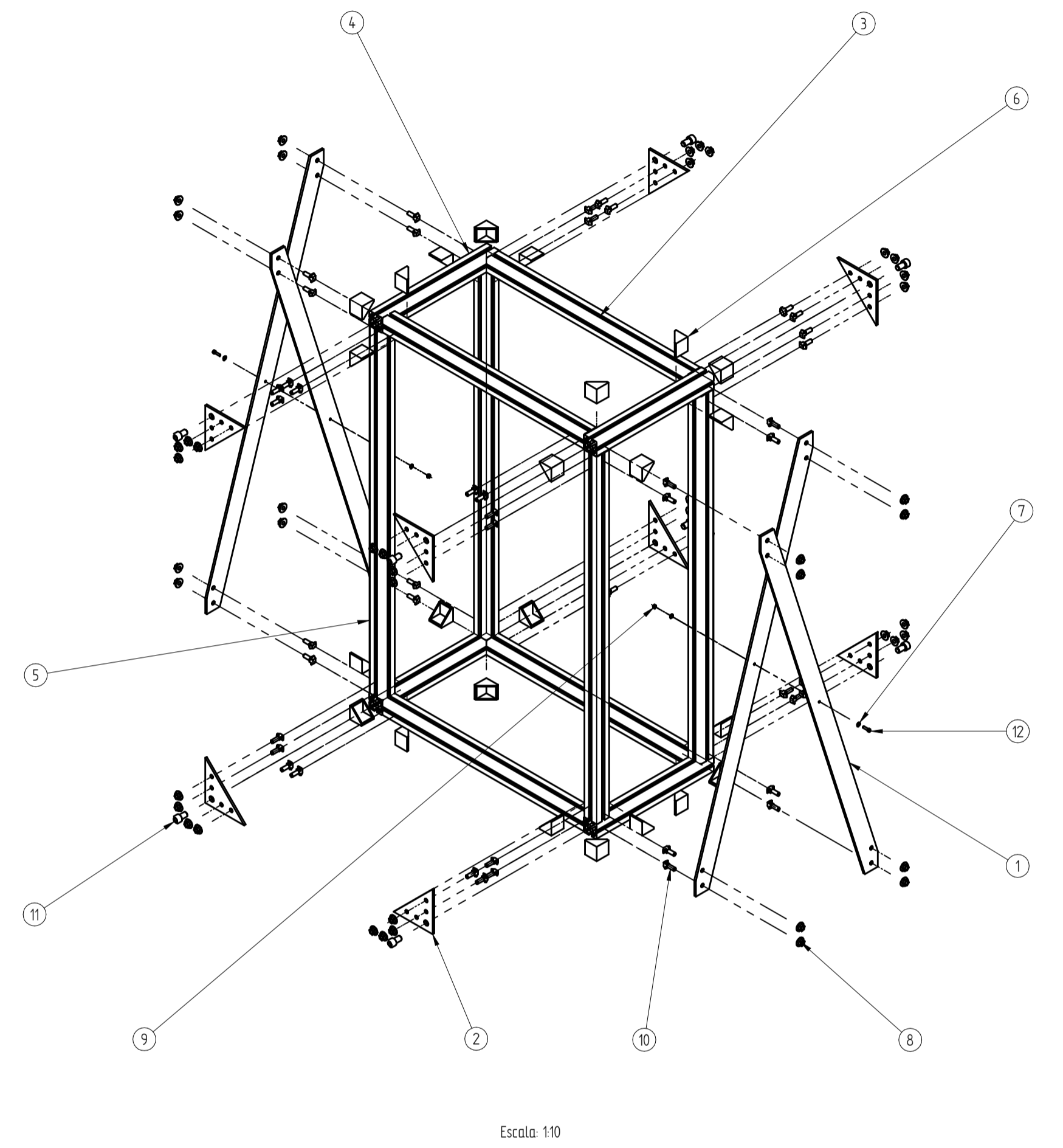
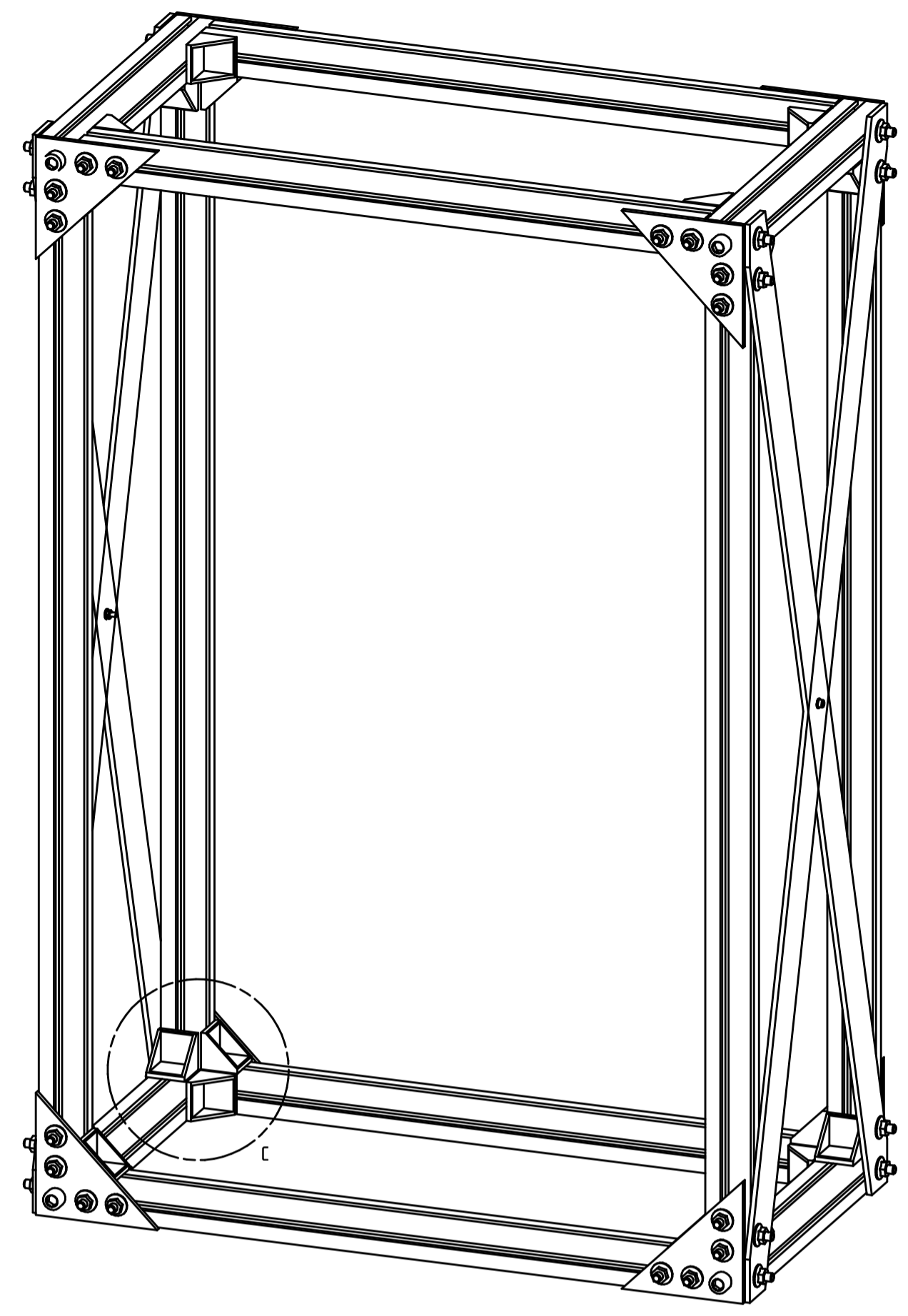
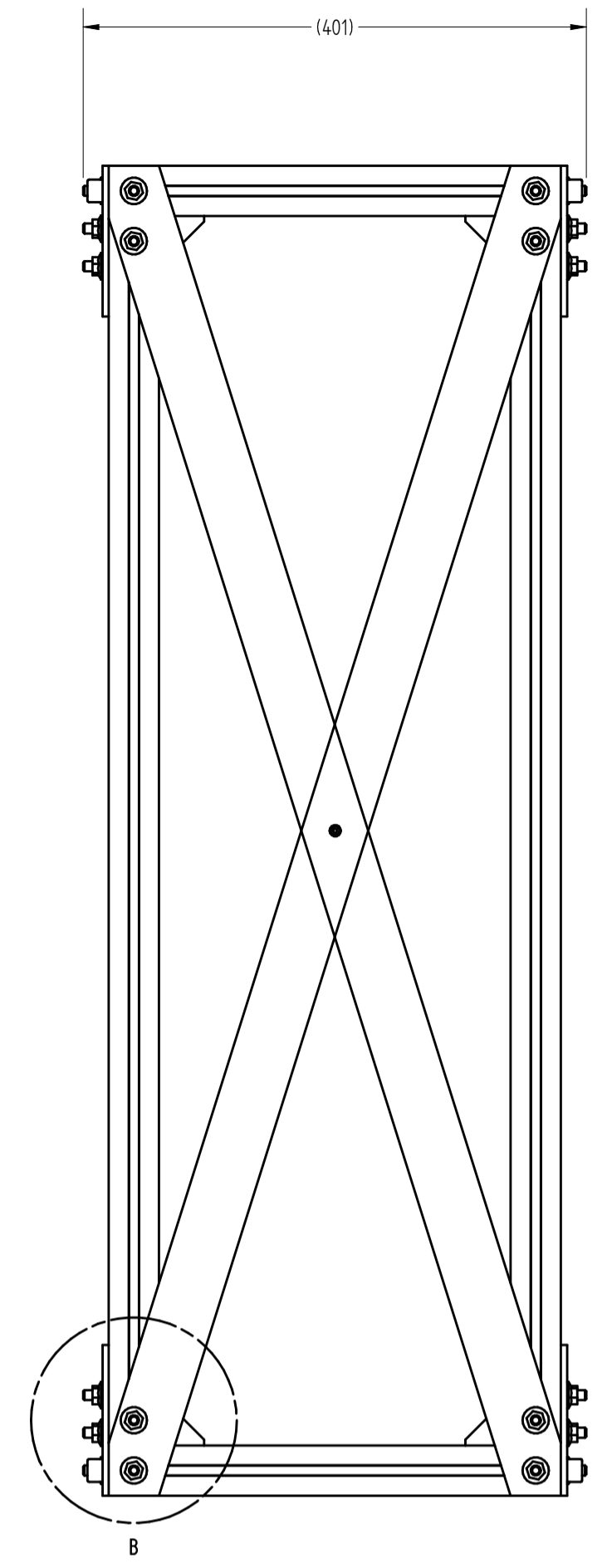
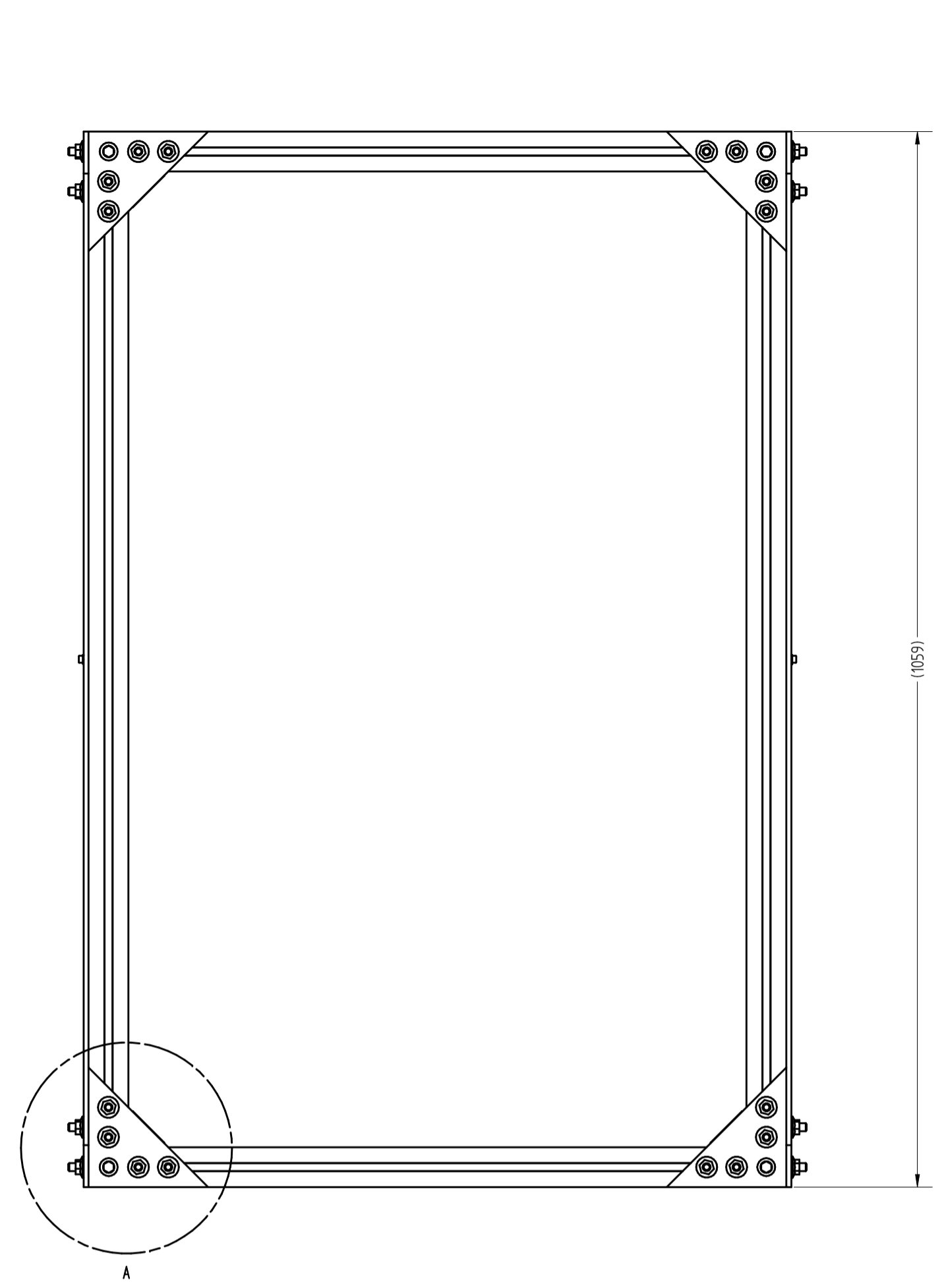
- Se verificó mecánicamente los componentes asociados al prototipo Alpha de colimador oscilante, y se muestra en el Capítulo 2.
- Se realizó el diseño mecánico de un banco de ensayos para el colimador oscilante, junto con la documentación de ingeniería hasta su materialización. Esto se indica en el Capítulo 3.
- Se realizó el modelado numérico-analítico, y se resolvió la respuesta temporal del sistema ante un movimiento trapezoidal propuesto como entrada, con y sin actuación de compensación de vibraciones. En el Capítulo 4 se indica este modelo y sus resultados.
- En el Capítulo 5 se indica la instrumentación realizada, junto con su programación asociada.
- Por último, se realizó ensayos experimentales básicos para comprender la dinámica del dispositivo, y verificar fallas y/o problemas existentes en el mismo. Esto último se muestra en el Capítulo 6.

Apéndice B

Planos del banco de pruebas diseñado

En este apéndice se exponen los planos correspondientes al banco de pruebas diseñado. Los mismos se identificaron con el código PL-TH_ME-18-256-0 suministrado por un sistema de códigos de la división de Termohidráulica del Centro Atómico Bariloche.

RUGOSIDAD Ra(μm) ISO 13021	ARISTAS (ISO 13715)		TOLERANCIAS PARA DIMENSIONES SIN INDICACIONES INDIVIDUALES			
	3.2	-0.3	ISO 2768.1	ISO 19920	ISO 2768.2	
			LINEALES	Grado: m	Grado: N/A	Grado: K
			BOSQUES Y RADIOS	Grado: m	Grado: N/A	Grado: K
			ANGULOS	Grado: m	Grado: N/A	Grado: K
			RELT. PLA. PAR.	Grado: N/A	Grado: N/A	Grado: K
TOLERANCIAS GEOMETRICAS SEGUN NORMA ISO 8015						



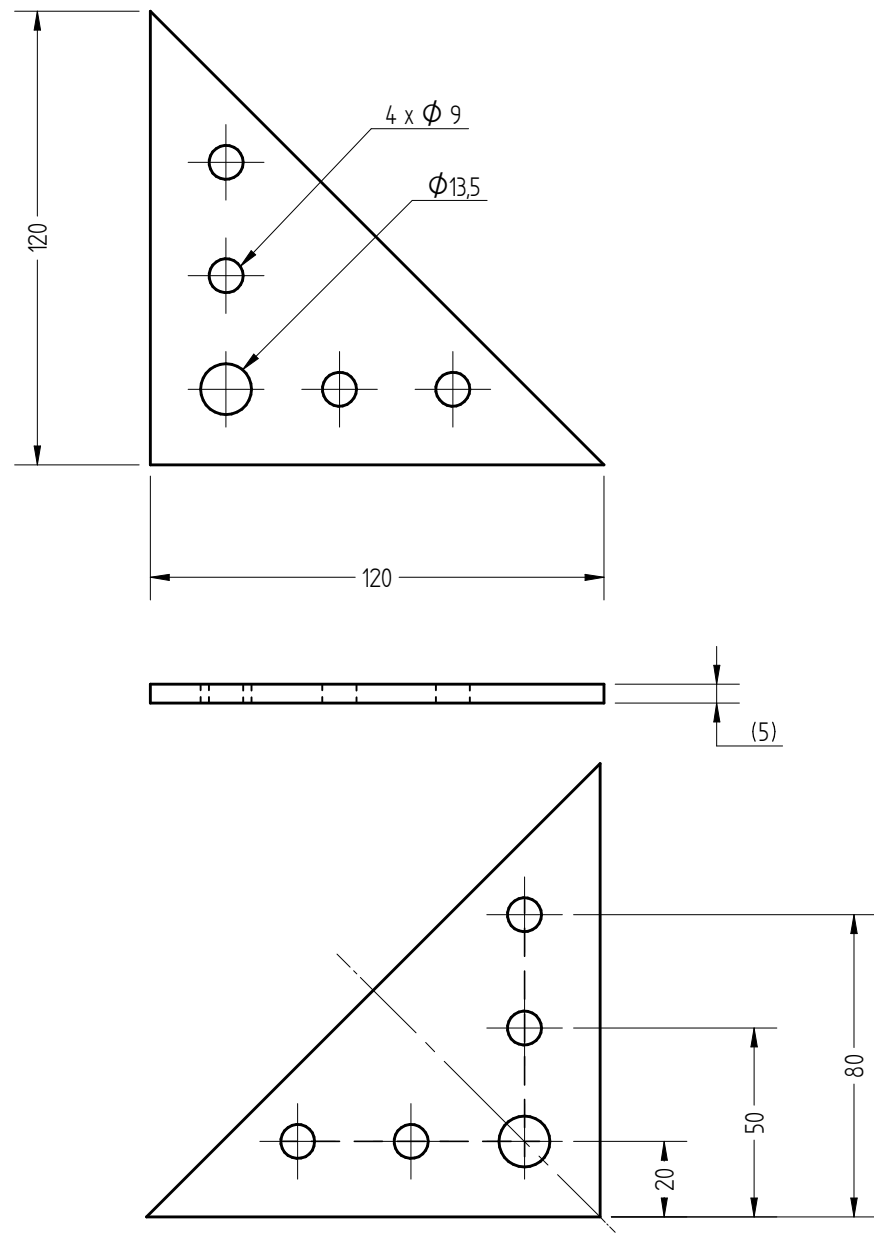
NOTA 1 Extremos de los perfiles fresados.

Pos.	Nombre	Cont.	Material	Referencia
12	Tornillo cilíndrico con hexágono interior M4x16	2	Acero de bajo o medio carbono clase 4.8	DIN 912
11	Tornillo cilíndrico con hexágono interior M12x20	8	Acero de bajo o medio carbono clase 4.8	DIN 912
10	Tornillo de martillo con grifería M8x25	48	Acero de bajo o medio carbono clase 4.8	DIN 188
9	Tuerca hexagonal M4	2	Acero de bajo o medio carbono clase 4.8	DIN 934
8	Tuerca hexagonal con collar biselado M8	48	Acero de bajo o medio carbono clase 4.8	DIN 934
7	Arandela plana M4	4	Acero al carbono	DIN 125
6	Juego escuadra 40x40 y material de fijación (2xFS)	24		Código Bosch 3 842 529 383
5	40x40 T-slot	4	Aluminio, 6061-T6	Largo = 979
4	40x40 T-slot	4	Aluminio, 6061-T6	Largo = 360; Roscar extremos M12x20
3	40x40 T-slot	4	Aluminio, 6061-T6	Largo = 620
2	Refuerzo esquinero	8	Aluminio, 6061-T6	Hoja 2
1	Diagonales	4	Aluminio, 6061-T6	Hoja 3

Calidad: E. Ruz Nocini Aprobó: A. Calerff Distribución: S. Pizarro Eje: J. C. Gracia Firma: 15 Estado del documento: Borrador Fecha: A partir de fecha de aprobación Liberó: E. Ruz Nocini	Fecha: _____ Firma: _____ Títulos: ANDES - ALPHA SS06-03 Oscilador - Banco de ensayo A1-H	 Código N°: Hoja 1 de 3 PL-TH_ME-18-256-0 Rev. N°: 0 Documento Base: -
---	---	--

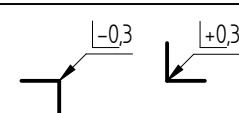
SOLID EDGE ACADÉMICO COPY

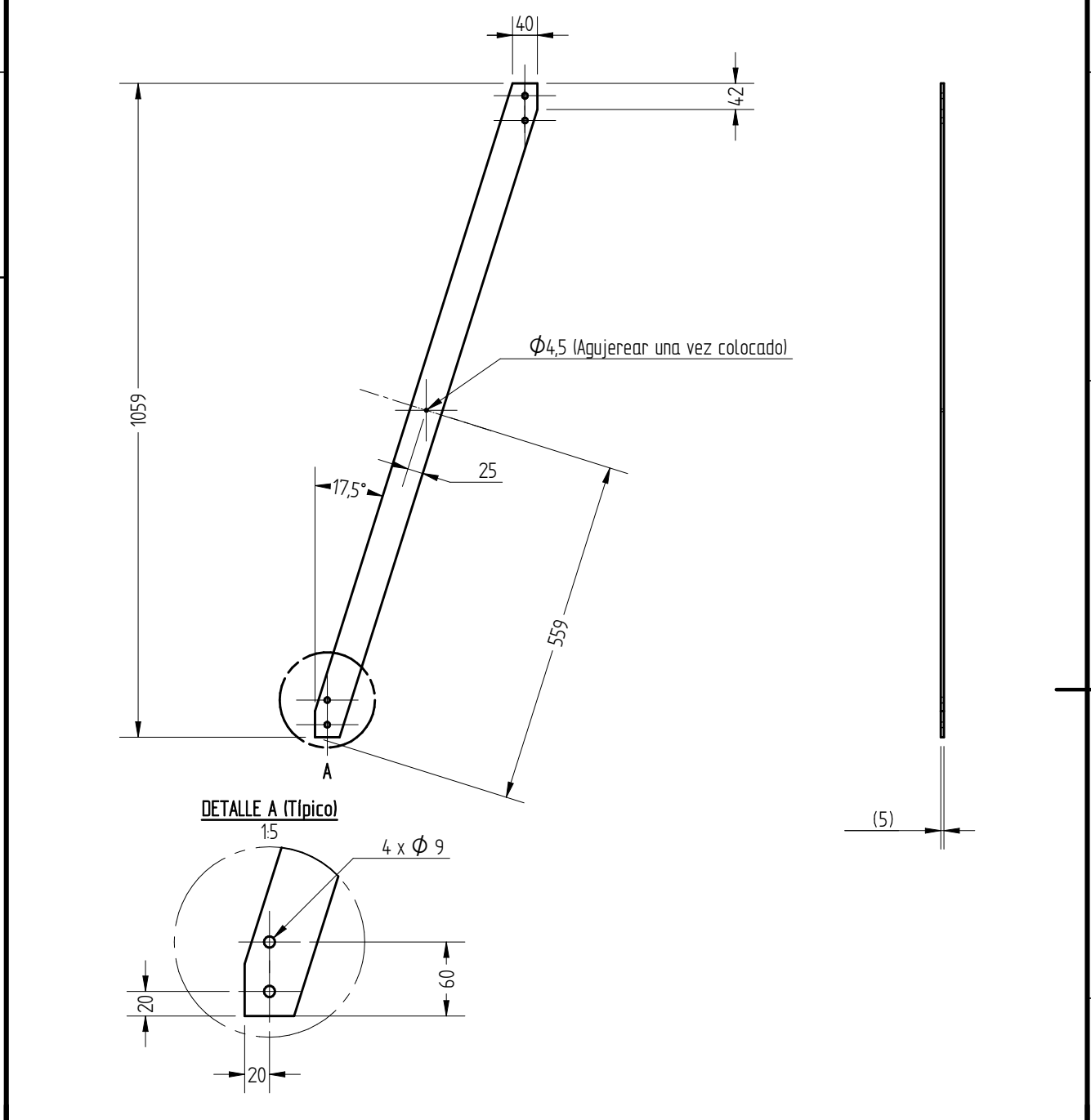
1	2	3	4
RUGOSIDAD Ra[μm] (ISO 1302)	ARISTAS (ISO 13715)	TOLERANCIAS PARA DIMENSIONES SIN INDICACIONES INDIVIDUALES	
3,2 		ISO 2768.1	ISO 13920
		LINEALES Grado: m	Grado: N/A
		BISELES Y RADIOS Grado: m	Grado: K
		ANGULOS Grado: m	Grado: N/A
TOLERANCIAS GEOMETRICAS SEGUN NORMA: ISO 8015		RECT, PLA, PAR	Grado: N/A






ISO-44-V 210 X 297
 FO-EN_GIN-TH_ME-010 rev3
 ESTE DOCUMENTO ES PROPIEDAD DE LA COMISION NACIONAL DE ENERGIA ATOMICA. SU REPRODUCCION O UTILIZACION TOTAL O PARCIAL DE SU CONTENIDO, NO ESTA PERMITIDA, SALVO AUTORIZACION EXPRESA POR ESCRITO. TODOS LOS DERECHOS SE HAN RESERVADO, ESPECIALMENTE EN EL CASO DE PATENTES O REGISTROS DE INVENCIÓN. ESTE DOCUMENTO DEBE SER DESTRUIDO UNA VEZ QUE SU UTILIZACION NO SEA NECESARIA, A MENOS QUE LA RETENCION DEL MISMO SEA OBLIGATORIA POR MANDATO ESCRITO DE LA COMISION NACIONAL DE ENERGIA ATOMICA.

1	Refuerzo esquinero	1	Aluminio, 6061-T6	Hoja 2
Pos.	Nombre	Cant.	Material	Referencia
Calidad:	E. Ruiz Nicolini	Fecha	Firma	
Aprobó:	A. Coleff	Proy. M. Trapp		
DISTRIBUCIÓN		Dib. M. Trapp		
Copiar:		Rev. S. Pincin		
Firma:		Rev. J. C. Gracia		
ESTADO DEL DOCUMENTO:		Esc: 1:2	TITULOS:	Código N°: Hoja 2 de 3
BORRADOR			ANDES - ALPHA SS06-03	PL-TH_ME-18-256-0
Fecha: A partir de fecha de aprobación		Oscilador - Banco de ensayo		Rev. N° 0
Liberó: E. Ruiz Nicolini		A4-V		Documento Base

1	2	3	4
RUGOSIDAD Ra[μm] (ISO 1302)	ARISTAS (ISO 13715)	TOLERANCIAS PARA DIMENSIONES SIN INDICACIONES INDIVIDUALES	
3,2		ISO 2768.1	ISO 13920
		LINEALES	Grado: m
		BISELES Y RADIOS	Grado: m
		ANGULOS	Grado: m
TOLERANCIAS GEOMETRICAS SEGUN NORMA: ISO 8015		RECT, PLA, PAR	Grado: N/A



1	Diagonales	1	Aluminio, 6061-T6	Hoja 3
Pos.	Nombre	Cant.	Material	Referencia
Calidad:	E. Ruiz Nicolini	Fecha:	Firma:	  
Aprobó:	A. Coleff	Proy.:	M. Trapp	
DISTRIBUCIÓN	Consultar listado de documentos	Dib.:	M. Trapp	
Copia:		Rev.:	S. Pincin	
Firma:		Rev.:	J. C. Gracia	
ESTADO DEL DOCUMENTO:		Esc.:	1:10	TÍTULOS:
BORRADOR		ANDES - ALPHA SS06-03		Código N°:
Fecha:	A partir de fecha de aprobación	Oscilador - Banco de ensayo		Hoja 3 de 3
Liberó:	E. Ruiz Nicolini	A4-V		PL-TH_ME-18-256-0
				Rev. N°
				0
				Documento Base

ISO-44-V 210 X 297
 FO-EN_EIN-TH_ME-010 rev3
 ESTE DOCUMENTO ES PROPIEDAD DE LA COMISIÓN NACIONAL DE ENERGÍA ATÓMICA. SU REPRODUCCIÓN O UTILIZACIÓN TOTAL O PARCIAL DE SU CONTENIDO, NO ESTÁ PERMITIDA, SALVO AUTORIZACIÓN EXPRESA POR ESCRITO. TODOS LOS DERECHOS SE HAN RESERVADO, ESPECIALMENTE EN EL CASO DE PATENTES O REGISTROS DE INVENCIÓN. ESTE DOCUMENTO DEBE SER DESTROYED UNA VEZ QUE SU UTILIZACIÓN NO SEA NECESARIA, A MENOS QUE LA RETENCIÓN DEL MISMO SEA OBLIGATORIA POR MANDATO ESCRITO DE LA COMISIÓN NACIONAL DE ENERGÍA ATÓMICA.

Apéndice C

Código correspondiente al modelado dinámico

En este apéndice se exponen el código utilizado para los cálculos correspondientes al modelado dinámico. Este código se lo redactó en lenguaje Python y se lo corrió en el programa Spyder. A continuación se expone el código mencionado:

```
import sys
import time
import scipy
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt
from scipy.fft import fft, fftfreq
from scipy.signal import blackman
from scipy import signal

parameters = {'font.size' : 20}
plt.rcParams.update(parameters)

# %%

## inicializo las variables
m.1=36.4
m.2=20
m.3=53.4
k1=178600
```

```

k2=111000000
k3=396000000

K=np.matrix([[k1+k2+k3, -k2, -k3],[-k2, k2, 0],[-k3, 0, k3
]])

M=np.matrix([[m_1, 0, 0],[0, m_2, 0],[0, 0, m_3]])

k1p=(M**(-1))@K

lamda=abs(np.linalg.eigvals(k1p))

eig_vec=(np.linalg.eig(k1p))[1]

mu_0 = (np.dot( np.dot( np.transpose(eig_vec[:,0]) ,M) ,(
    eig_vec[:,0]) )).item()
mu_1 = (np.dot( np.dot( np.transpose(eig_vec[:,1]) ,M) ,(
    eig_vec[:,1]) )).item()
mu_2 = (np.dot( np.dot( np.transpose(eig_vec[:,2]) ,M) ,(
    eig_vec[:,2]) )).item()

PHI = np.matrix( [ [ 1/mu_0**0.5 *eig_vec[0,0],1/mu_1
    **0.5 *eig_vec[0,1], 1/mu_2**0.5 *eig_vec[0,2]] , [ 1/
    mu_0**0.5 *eig_vec[1,0] ,1/mu_1**0.5 *eig_vec[1,1], 1/
    mu_2**0.5 *eig_vec[1,2]], [ 1/mu_0**0.5 *eig_vec[2,0]
    ,1/mu_1**0.5 *eig_vec[2,1] , 1/mu_2**0.5 *eig_vec[2,2]]
] )

w=lamda**(0.5)

f=w/2/np.pi

```

```

## calculo la matriz C teniendo la matriz C diagonal
xi=0.05
M_dig = PHI.transpose().dot(M).dot(PHI)
C_dig= np.matrix([[2*M_dig[0,0]*xi*w[0],0,0],[0,2*M_dig
    [1,1]*xi*w[1],0],[0,0,2*M_dig[2,2]*xi*w[2]]])
##C=PHI.dot(C_dig).dot(PHI.transpose())

fm=3000
per=2
L=per*fm
## tiempo en segundos

t=np.linspace(0,per,num=L)
x0=np.array([0,0,0])
v0=np.array([0,0,0])
F_1=np.zeros((3,L))
F_2=np.zeros((3,L))

# %%
# Escribo el perfil de fuerzas calculado considerando un
tiempo de aceleracion del 2\% del periodo.
aux1=0
for k in range(1,L):
    t_aux=t[k-1]-aux1
    if (t_aux<1):
        if (t_aux<0.02):
            F_2[0,k]=0
            F_2[1,k]=-67.5
            F_2[2,k]=173.5
        if (0.02<t_aux<0.48):
            F_2[0,k]=-6.6+5
            F_2[1,k]=-5
            F_2[2,k]=6.6
        if (0.48<t_aux<0.5):
            F_2[0,k]=0
            F_2[1,k]=57.5
            F_2[2,k]=-160.3
        if (0.5<t_aux<0.52):

```

```

        F_2 [0 , k]=0
        F_2 [1 , k]=67.5
        F_2 [2 , k]=-173.5
    if (0.52 < t_aux < 0.98) :
        F_2 [0 , k]=0
        F_2 [1 , k]=5
        F_2 [2 , k]=-6.6
    if (0.98 < t_aux < 1) :
        F_2 [0 , k]=0
        F_2 [1 , k]=-57.5
        F_2 [2 , k]=160.3
else :
    aux1=aux1+1

## Fuerzas con un desplazamiento del actuador de 80 mm
    # if (t_aux < 0.02) :
    #     F_2 [0 , k] = -173.5 + 171.6
    #     F_2 [1 , k] = -171.6
    #     F_2 [2 , k] = 173.5
    # if (0.02 < t_aux < 0.48) :
    #     F_2 [0 , k] = -6.6 + 5
    #     F_2 [1 , k] = -5
    #     F_2 [2 , k] = 6.6
    # if (0.48 < t_aux < 0.5) :
    #     F_2 [0 , k] = 160.3 - 161.7
    #     F_2 [1 , k] = 161.7
    #     F_2 [2 , k] = -160.3
    # if (0.5 < t_aux < 0.52) :
    #     F_2 [0 , k] = 173.5 - 171.6
    #     F_2 [1 , k] = 171.6
    #     F_2 [2 , k] = -173.5
    # if (0.52 < t_aux < 0.98) :
    #     F_2 [0 , k] = 6.6 - 5
    #     F_2 [1 , k] = 5
    #     F_2 [2 , k] = -6.6
    # if (0.98 < t_aux < 1) :
    #     F_2 [0 , k] = -160.3 + 161.7

```

```
#      F_2[1,k]=-161.7
#      F_2[2,k]=160.3
```

```
aux2=0
for k in range(1,L):
    t_aux=t[k-1]-aux2
    if(t_aux<1):
        if(t_aux<0.02):
            F_1[0,k]=0
            F_1[1,k]=0
            F_1[2,k]=173.5
        if(0.02<t_aux<0.48):
            F_1[0,k]=0
            F_1[1,k]=0
            F_1[2,k]=6.6
        if(0.48<t_aux<0.5):
            F_1[0,k]=0
            F_1[1,k]=0
            F_1[2,k]=-160.3
        if(0.5<t_aux<0.52):
            F_1[0,k]=0
            F_1[1,k]=0
            F_1[2,k]=-173.5
        if(0.52<t_aux<0.98):
            F_1[0,k]=0
            F_1[1,k]=0
            F_1[2,k]=-6.6
        if(0.98<t_aux<1):
            F_1[0,k]=0
            F_1[1,k]=0
            F_1[2,k]=160.3
    else:
        aux2=aux2+1
```

```
F_base=np.array(F_2[0,:])
F_actuador=np.array(F_2[1,:])
F_colimador=np.array(F_2[2,:])
```

```

plt.figure()
# plt.plot(t, F_base)
# plt.plot(t, F_base, label='F_1')
plt.plot(t, F_actuador, color='violet', label='F_2')
plt.plot(t, F_colimador, color='orange', label='F_3')
plt.xlabel('t (s)')
plt.ylabel('Fuerza (N)')
plt.legend()

###
# Implemento el metodo Newmark
def Newmark(M,K,C,PHI,x0,v0,F_,fm,L, gamma=1/2,bta=1/4):

    M_PHI = PHI.transpose().dot(M).dot(PHI)
    ##print("M_PHI={}".format(M_PHI))
    K_PHI = PHI.transpose().dot(K).dot(PHI)
    ##print("K_PHI={}".format(K_PHI))
    C_PHI = C ##PHI.transpose().dot(C).dot(PHI)
    ##print("C_PHI={}".format(C_PHI))

    F_PHI = PHI.transpose().dot(F_)

    q_0 = PHI.transpose().dot(M).dot(x0)
    dq_0 = PHI.transpose().dot(M).dot(v0)

    F_PHI_0 = F_PHI[:,0]

    ddq_0 = np.linalg.inv(M)@(F_PHI_0 - C_PHI.dot(dq_0.
        transpose()) - K_PHI.dot(q_0.transpose())) ###
        elimine el dq_0.transpose()
    dt=1/fm

    a1 = 1/bta/dt**2 *M_PHI + gamma/bta/dt * C_PHI
    a2 = 1/bta/dt *M_PHI + (gamma/bta-1) * C_PHI
    a3 = (1/2/bta-1) * M_PHI + dt*(gamma/2/bta-1) * C_PHI

    Km = K_PHI + a1

    u = np.zeros((3,L))

```

```

du    = np.zeros((3,L))
ddu   = np.zeros((3,L))

q     = np.zeros((3,L))
dq    = np.zeros((3,L))
ddq   = np.zeros((3,L))

q[:,0] = q_0
dq[:,0] = dq_0
ddq[:,0] = ddq_0.transpose()

for i in range(1,L):
    Pm = PHI.transpose().dot(F_[:,i]).reshape((1,3)) +
        a1.dot(q[:,i-1]) + a2.dot(dq[:,i-1]) + a3.dot(
            ddq[:,i-1])
    q[:,i] = np.transpose(np.linalg.inv(Km).dot(Pm.
        transpose()))

    dq[:,i] = np.transpose( gamma/bta/dt * (q[:,i]-q[:,
        i-1]) + (1-gamma/bta) * dq[:,i-1] + dt * (1-gamma
        /2/bta) * ddq[:,i-1] )
    ddq[:,i] = np.transpose( 1/bta/dt**2 * (q[:,i]-q[:,
        i-1]) - 1/bta/dt * dq[:,i-1] - (1/2/bta-1)
        * ddq[:,i-1])

    u[:,i] = PHI.dot(q[:,i])
    du[:,i] = PHI.dot(dq[:,i])
    ddu[:,i] = PHI.dot(ddq[:,i])
return u,du,ddu

```

#Grafico la posicion absoluta de la base y la relativa a esta del colimador y el actuador

```

X1, V1, A1= Newmark(M,K,C_dig,PHI,x0,v0,F_1,fm,L, gamma
    =1/2,bta=1/4)
plt.figure()
plt.plot(t,X1[0,:]*1000, 'g', label='X1')
plt.plot(t,F_1[1,:]*5e-3, color=' #800080',label='
    F_actuador',linestyle='—')

```

```

plt.plot(t, F_1[2, :]*5e-3, color='orange', label='
    F_colimador', linestyle='—')
plt.xlabel('t (s)')
plt.ylabel('Desplazamiento (mm)')
plt.legend()

X2, V2, A2= Newmark(M, K, C_dig, PHI, x0, v0, F_2, fm, L, gamma
    =1/2, bta=1/4)
plt.figure()
plt.plot(t, X2[0, :]*1000, 'g', label='X1')
plt.plot(t, F_2[1, :]*5e-3, color='#800080', label='
    F_actuador', linestyle='—')
plt.plot(t, F_2[2, :]*5e-3, color='orange', label='
    F_colimador', linestyle='—')
plt.xlabel('t (s)')
plt.ylabel('Desplazamiento (mm)')
plt.legend()

%%
# Grafico la posicion y aceleracion de la base con y sin
    actuar el actuador
plt.figure()
plt.plot(t, X1[0, :]*1000, 'r', label='Sin accionamiento
    actuador')
plt.plot(t, X2[0, :]*1000, 'b', label='Con accionamiento
    actuador')
plt.xlabel('t (s)')
plt.ylabel('Desplazamiento (mm)')
plt.legend()

plt.figure()
plt.plot(t, A1[0, :], 'r', label='Sin accionamiento
    actuador')
plt.plot(t, A2[0, :], 'b', label='Con accionamiento
    actuador')
plt.xlabel('t (s)')
plt.ylabel('Aceleracion (m/s^2)')

```

```
plt.legend()
```

```
###
```

```
# Calculo los valres maximos y rms de los desplazamientos
```

```
Des_max1=np.max(np.abs(X1[0,:]))
```

```
Des_max2=np.max(np.abs(X2[0,:]))
```

```
Des_rms1=np.std(X1[0,:])
```

```
Des_rms2=np.std(X2[0,:])
```

```
# Calculo los valores maximos y rms delas aceleraciones
```

```
ac_max1=np.max(np.abs(A1[0,:]))
```

```
ac_max2=np.max(np.abs(A2[0,:]))
```

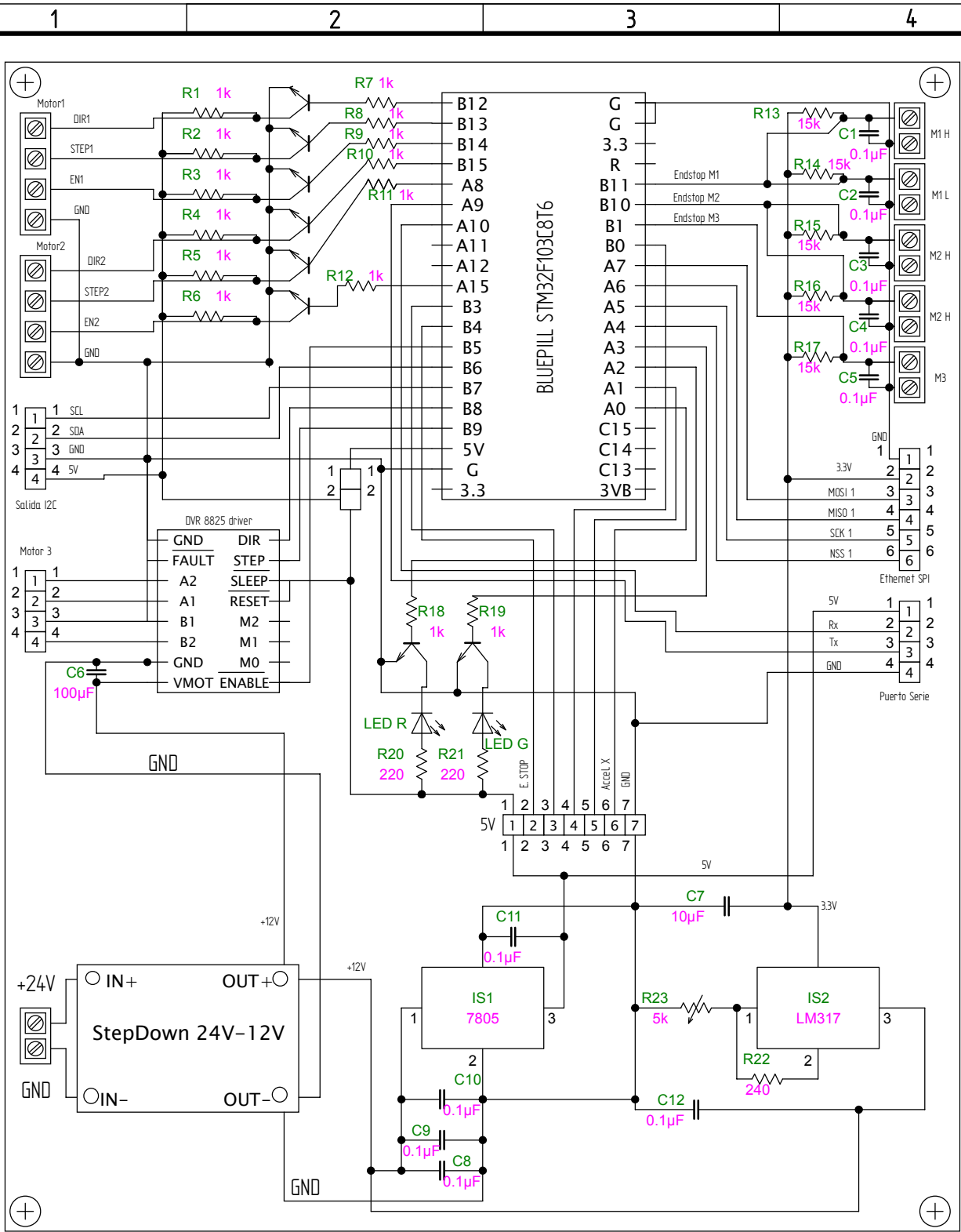
```
ac_rms1=np.std(A1[0,:])
```

```
ac_rms2=np.std(A2[0,:])
```


Apéndice D


Plano de la placa electrónica diseñada y fabricada

En este apéndice se expone el plano esquemático correspondiente a la placa electrónica diseñada. Este plano no cuenta con un código ya que aún no se le adjudicó uno por el sistema de códigos del departamento de Termohidráulica del Centro Atómico Bariloche.



FO-EN_GIN-TH_ME-010 rev3
 ESTE DOCUMENTO ES PROPIEDAD DE LA COMISIÓN NACIONAL DE ENERGÍA ATÓMICA. SU REPRODUCCIÓN O UTILIZACIÓN, TOTAL O PARCIAL, DE SU CONTENIDO, NO ESTÁ PERMITIDA SIN LA AUTORIZACIÓN EXPRESA POR ESCRITO. TODOS LOS DERECHOS SE HAN RESERVADOS, ESPECIALMENTE EN EL CASO DE PATENTES O REGISTROS DE INVENCIÓN. PENDIENTES ESTE DOCUMENTO DEBE SER DESTRUIDO UNA VEZ QUE SU UTILIZACIÓN NO SEA NECESARIA, A MENOS QUE LA RETENCIÓN DEL MISMO SEA OBLIGATORIA POR MANDATO ESCRITO DE LA COMISIÓN NACIONAL DE ENERGÍA ATÓMICA.

Calidad: E. Ruiz Nicolini
 Aprobó: S. Pincin
DISTRIBUCIÓN
 Consultar listado de documentos
 Copia:
 Firma:
ESTADO DEL DOCUMENTO:
BORRADOR
 Fecha: A partir de fecha de aprobación
 Liberó: E. Ruiz Nicolini

Proy. M. Trapp
 Dib. M. Trapp
 Rev. S. Pincin
 Rev. -
 Esc: 1:1

 A4-V

Fecha
 Firma

TÍTULOS:
 LAHN-ANDES
Alpha SS06 Colimador oscilante - Placa electrónica



Código N°: Hoja 1 de 1
 Rev. N° 0
 Documento Base

Apéndice E

Códigos utilizados para programar el microprocesador STM32

En este apéndice se expone el código utilizado para programar el microcontrolador STM32. Este código se lo redactó en lenguaje C++ utilizando el programa Arduino. A continuación se expone el código mencionado:

```
#include <AccelStepper.h>

const double StepsPermm = 40; // define conversation constant
    steps/mm para motores grandes
const double Stepspermmch = 100; // define conversation
    constant steps/mm para motores chicos
#define HEAD ':' // define start message character
#define TAIL '\n' // define finish message character
#define enablePin1 PB14 // define pins to enable and
    disable the steppers
#define enablePin2 PA15
#define enablePin3 PB5
#define PanicButton PB4
#define accelX_pin PA0
#define accelY_pin PA1
// #define RedPin PA3 // define indicator led pins.
// #define GreenPin PA2

volatile uint16_t timeout_T1;
uint16_t contT1 = 0;
uint16_t contT1_ = 0;
```

```

volatile uint16_t timeout_T2;
volatile uint16_t LECTURA;

//bool HOME = 0;

// Define period time, displacement length and porcentage
  acceleration for all steppers
/*double Tper1 = 1; // en seg
double L1 = 20; // en mm
double PorcAcel1 = 0.02; // en % del periodo

double Tper2 = 1; // en seg
double L2 = 20; // en mm
double PorcAcel2 = 0.02; // en % del periodo

double Tper3 = 1; // en seg
double L3 = 2; // en mm
double PorcAcel3 = 0.02; // en % del periodo
*/

double Tper1 = 1; // en seg
double L1 = 30; // en mm
double PorcAcel1 = 0.02; // en % del periodo

double Tper2 = 1; // en seg
double L2 = 30; // en mm
double PorcAcel2 = 0.02; // en % del periodo

double Tper3 = 1; // en seg
double L3 = 0; // en mm
double PorcAcel3 = 0.02; // en % del periodo

double maxSpeed1 = L1 / (Tper1 / 2 - Tper1*PorcAcel1) *
  StepsPermm;
double accel1 = L1 / (Tper1*PorcAcel1) / (Tper1 / 2 - Tper1*
  PorcAcel1) * StepsPermm;
double moveto1 = L1 * StepsPermm;

```

```

double maxSpeed2 = L2 / (Tper2 / 2 - Tper2*PorcAcel2) *
    StepsPermm;
double accel2 = L2 / (Tper2*PorcAcel2) / (Tper2 / 2 - Tper2*
    PorcAcel2) * StepsPermm;
double moveto2 = L2 * StepsPermm;

double maxSpeed3 = L3 / (Tper3 / 2 - Tper3*PorcAcel3) *
    Stepspermmch;
double accel3 = L3 / (Tper3*PorcAcel3) / (Tper3 / 2 - Tper3*
    PorcAcel3) * Stepspermmch;
double moveto3 = L3 * Stepspermmch;

// Define variables to use in prosses of messages
char mensaje[10] ;
uint8_t buff_out [4];
char entrada ;
int Entrada_SIZE = 0;
int8_t chsl = 0;

// Define lecture variables
volatile uint16_t top_timer = 0; // tiempo de muestreo en
    segundos
volatile uint32_t cont_lect = 0; // contador utilizado en
    el bucle de lectura
volatile uint32_t NMUESTRAS = 0; // Numero de muestras

// Define pin connections
const int dirPin1 = PB12;
const int stepPin1 = PB13;
const int dirPin2 = PB15;
const int stepPin2 = PA8;
const int dirPin3 = PB8;
const int stepPin3 = PB9;
int endstop1 = PB11;
int endstop2 = PB10;
int endstop3 = PB1;

/*

```

```

char swTxBuffer[sizeSwTxBuffer];
char swRxBuffer[sizeSwRxBuffer];
uint8_t buff_rec[2];
*/

// Define motor interface type
#define motorInterfaceType 1

// Creates the steppers instances
AccelStepper myStepper1(motorInterfaceType, stepPin1, dirPin1)
    ;
AccelStepper myStepper2(motorInterfaceType, stepPin2, dirPin2)
    ;
AccelStepper myStepper3(motorInterfaceType, stepPin3, dirPin3)
    ;

// Activate the timer TIM1 and TIM2
HardwareTimer *Tim1;
HardwareTimer *Tim2;

void TIM1_OV_interrupt(void) {
    timeout_T1 = 1;
}

void TIM2_OV_interrupt(void) {
    timeout_T2 = 1;
}

void tim1_init_arduino() {
    Tim1 = new HardwareTimer(TIM1);

    //72*10^6/9/800 frecuencia del timer en Hz
    // en este caso son 10kHz y periodo de 100us
    Tim1->setOverflow(800); //100us

    //1MHz freq, 1us tick;
    // lo que va entre parentesis es el divisor de la frecuencia
    del timer,

```

```

//en este caso la frec del timer es 72 MHz entonces 72/72=1
  MHz
Tim1->setPrescaleFactor(9);

Tim1->attachInterrupt(TIM1_OV_interrupt);
Tim1->refresh();
}

void tim2_init_arduino(uint16_t ovfl) {
  if (ovfl > 10000) {
    ovfl = 10000;
  }
  if (ovfl < 10) {
    ovfl = 10;
  }
  Tim2 = new HardwareTimer(TIM2);
  Tim2->setOverflow(ovfl); //72*10^6/72/80 frecuencia del
    timer en Hz // en este caso son 100kHz y periodo de 10us
  Tim2->setPrescaleFactor(7200); //1MHz frec, 100us tick; //
    lo que va entre parentesis es el divisor de la frecuencia
    del timer, en este caso la frec del timer es 72 MHz
    entonces 72/72=1MHz
  Tim2->attachInterrupt(TIM2_OV_interrupt);
  Tim2->refresh();
}

void tim2_reinit_arduino(uint16_t ovfl) {
  if (ovfl > 10000) { //10000
    ovfl = 10000;
  }
  if (ovfl < 10) {
    ovfl = 10;
  }
  Tim2->setOverflow(ovfl); //72*10^6/72/80 frecuencia del
    timer en Hz // en este caso son 100kHz y periodo de 10us
  Tim2->setPrescaleFactor(7200); //1MHz frec, 100us tick; //
    lo que va entre parentesis es el divisor de la frecuencia
    del timer, en este caso la frec del timer es 72 MHz
    entonces 72/72=1MHz
}

```

```

    Tim2->refresh ();
}

// define the states to use in the interrupt loop
enum Estados {Alarma, Alarma1, Espera, Rutina, Rutina1, Stop,
    Disable, Enable};
Estados estado;

/*
void sw_init(){
    sw.setTxBuffer(swTxBuffer, sizeSwTxBuffer);
    sw.setRxBuffer(swRxBuffer, sizeSwRxBuffer);
    sw.setDelay_us(5); //400khz
    //sw.setDelay_us(3); //670khz
    sw.setTimeout_ms(100);
}*/

void serialRecieve()
{
    static uint8_t index = 0;
    if (entrada != HEAD && index == 0)
    {
        //Serial.println("El mensaje debe comenzar con :");
        return;
    }
    // Saving the recibing chars in the buffer
    mensaje[index] = entrada;
    index++;

    if ( entrada == TAIL )
    {
        Entrada_SIZE = index;
        index = 0;
        serialProcess();
        if (chsCheck(mensaje)) {
            //serialProcess();
        }
        //clear buffer
        uint8_t i;

```

```
    for (i = 0; i < Entrada_SIZE; i++) {
        mensaje[i] = 0;
    }
}

}

bool chsCheck (char* comdo) {
    chsl = 0;
    chsl = static_cast<int8_t>(comdo[1]) + static_cast<int8_t>(
        comdo[2]) + static_cast<int8_t>(comdo[3]) + static_cast<
        int8_t>(comdo[4]) +
        static_cast<int8_t>(comdo[5]) + static_cast<int8_t>(
            comdo[6]) + static_cast<int8_t>(comdo[7]);

    if ((static_cast<int8_t>(comdo[8])) != chsl) {
        //Serial.println("NO DA EL CHECKSUM");
        //digitalWrite(LED_BUILTIN, LOW);
        return false;
    }
    else {
        //digitalWrite(LED_BUILTIN, HIGH);
        return true;
    }
}

uint16_t saturate_uin16(uint16_t input, uint16_t miin,
    uint16_t maax){
    if(input<miin)return miin;
    if(input>maax)return maax;
    return input;
}

int16_t saturate_in16(int16_t input, int16_t miin, int16_t
    maax){
    if(input<miin)return miin;
    if(input>maax)return maax;
    return input;
}
```

```

}

void serialProcess()
{
    // Set the state in function to the message recived
    if (mensaje [1] == 'H')
    {
        Homefunc();
    }
    else if (mensaje [1] == 'R')
    {
        estado = Rutina;
    }
    else if (mensaje [1] == 'S')
    {
        estado = Stop;
    }
    else if (mensaje [1] == 'E')
    {
        estado = Enable;
    }
    else if (mensaje [1] == 'D')
    {
        estado = Disable;
    }
    else if (mensaje [1] == 'P' && ((mensaje[2]<<8)|mensaje [3])
        == 1) {
        //1=inercial, 2=colimador, 3 pivot
        uint16_t aux=uint16_t((mensaje[4]<<8)|mensaje [5]);
        aux=saturate_uin16(aux,10, 100);
        Tper1=((double)aux)*0.1;    //en s, de 0.1 a 10seg
        CalculateVariables();
    }
    else if (mensaje [1] == 'P' && ((mensaje[2]<<8)|mensaje [3])
        == 2) {
        //1=inercial, 2=colimador, 3 pivot
        uint16_t aux=uint16_t((mensaje[4]<<8)|mensaje [5]);
        aux=saturate_uin16(aux,10, 100);
        Tper2=((double)aux)*0.1;    //en s, de 0.1 a 10seg
    }
}

```

```

    CalculateVariables();
}
else if (mensaje [1] == 'P' && ((mensaje[2]<<8)|mensaje [3])
    == 3) {
    //1=inericial, 2=colimador, 3 pivot
    uint16_t aux=uint16_t((mensaje[4]<<8)|mensaje [5]);
    aux=saturate_uin16(aux,10, 100);
    Tper3=((double)aux)*0.1;    //en s, de 0.1 a 10seg
    CalculateVariables();
}
else if (mensaje [1] == 'L' && ((mensaje[2]<<8)|mensaje [3])
    == 1) {
    //1=inericial, 2=colimador, 3 pivot
    uint16_t aux=uint16_t((mensaje[4]<<8)|mensaje [5]);
    aux=saturate_uin16(aux,0, 1800);
    L1=((double)aux)*0.1;    //en mm
    CalculateVariables();
}
else if (mensaje [1] == 'L' && ((mensaje[2]<<8)|mensaje [3])
    == 2) {
    //1=inericial, 2=colimador, 3 pivot
    uint16_t aux=uint16_t((mensaje[4]<<8)|mensaje [5]);
    aux=saturate_uin16(aux,0, 300);
    L2=((double)aux)*0.1;    //en mm
    CalculateVariables();
}
else if (mensaje [1] == 'L' && ((mensaje[2]<<8)|mensaje [3])
    == 3) {
    //1=inericial, 2=colimador, 3 pivot
    uint16_t aux=uint16_t((mensaje[4]<<8)|mensaje [5]);
    aux=saturate_uin16(aux,0, 150);
    L3=((double)aux)*0.1;    //en mm
    CalculateVariables();
}
else if (mensaje [1] == 'A' && ((mensaje[2]<<8)|mensaje [3])
    == 1) {
    //% de aceleracion. Viene 1 2 o 3 %, hay que dividir por
    cien.
    //Y ademas viene multiplicada por cien por resolucio

```

```

    uint16_t aux=uint16_t((mensaje[4]<<8)|mensaje[5]);
    aux=saturate_uin16(aux,0.2*100, 25*100);
    PorcAcel1=((double)aux)*0.01*0.01;
    CalculateVariables();
}
else if (mensaje [1] == 'A' && ((mensaje[2]<<8)|mensaje[3])
    == 2) {
    uint16_t aux=uint16_t((mensaje[4]<<8)|mensaje[5]);
    aux=saturate_uin16(aux,0.2*100, 25*100);
    PorcAcel2=((double)aux)*0.01*0.01;
    CalculateVariables();
}
else if (mensaje [1] == 'A' && ((mensaje[2]<<8)|mensaje[3])
    == 3) {
    uint16_t aux=uint16_t((mensaje[4]<<8)|mensaje[5]);
    aux=saturate_uin16(aux,0.2*100, 25*100);
    PorcAcel3=((double)aux)*0.01*0.01;
    CalculateVariables();
}
else if (mensaje[1] == 'G' && mensaje[2] == '1') {
   CodigoG();
}
if (mensaje[1] == 'L') {
    //MPU_init();
    top_timer =((static_cast<uint16_t>(mensaje[2])) << 8) | (
        mensaje[3]);
    NMUESTRAS =((static_cast<uint16_t>(mensaje[4])) << 8) | (
        mensaje[5]);
    tim2_reinit_arduino(top_timer); //mando el overflow para
        el timer
    Tim2->resume();
    cont_lect = 0;
    LECTURA = 1;
}
else {
    //Serial.println("El mensaje no tiene funcion asociada");
}
}

```

```

void CalculateVariables () {
    maxSpeed1 = L1 / (Tper1 / 2 - Tper1 * PorcAcel1) *
        StepsPermm;
    accel1 = L1 / (Tper1 * PorcAcel1) / (Tper1 / 2 - Tper1 *
        PorcAcel1) * StepsPermm;
    moveto1 = L1 * StepsPermm;

    maxSpeed2 = L2 / (Tper2 / 2 - Tper2 * PorcAcel2) *
        StepsPermm;
    accel2 = L2 / (Tper2 * PorcAcel2) / (Tper2 / 2 - Tper2 *
        PorcAcel2) * StepsPermm;
    moveto2 = L2 * StepsPermm;

    maxSpeed3 = L3 / (Tper3 / 2 - Tper3 * PorcAcel3) *
        Stepspermmch;
    accel3 = L3 / (Tper3 * PorcAcel3) / (Tper3 / 2 - Tper3 *
        PorcAcel3) * Stepspermmch;
    moveto3 = L3 * Stepspermmch;
}

voidCodigoG () {
    //Serial.println("Entro al Gcode");
    //moving the stepper 1 whith G code
    if (mensaje[3] == 'X') {
        double objetivo1 = (mensaje[4] - '0') * ((mensaje[5] - '0'
            ) * 100 + (mensaje[6] - '0') * 10 + (mensaje[7] - '0')
            + (mensaje[9] - '0') * 0.1 + (mensaje[10] - '0') *
            0.01); //posicion en mm
        double velocidad1 = (mensaje[12] - '0') * 1000 + (mensaje
            [13] - '0') * 100 + (mensaje[14] - '0') * 10 + (mensaje
            [15] - '0'); // Velocidad en mm/min
        myStepper1.setMaxSpeed(velocidad1 / 60 * StepsPermm);
        myStepper1.moveTo(objetivo1 * StepsPermm);
        //Serial.println("Mueve motor 1");
        while (myStepper1.isRunning()) {
            if (digitalRead(endstop1) == HIGH || digitalRead(
                endstop2) == HIGH || digitalRead(endstop3) == HIGH ||
                digitalRead(PanicButton) == HIGH) {
                myStepper1.stop();
            }
        }
    }
}

```

```

        myStepper2.stop();
        myStepper3.stop();
    }
    myStepper1.run();
}
}

//moving the stepper 2 whith G code
if (mensaje[3] == 'Y') {
    double objetivo2 = (mensaje[4] - '0') * ((mensaje[5] - '0'
        ) * 100 + (mensaje[6] - '0') * 10 + (mensaje[7] - '0')
        + (mensaje[9] - '0') * 0.1 + (mensaje[10] - '0') *
        0.01); //posicion en mm
    double velocidad2 = (mensaje[12] - '0') * 1000 + (mensaje
        [13] - '0') * 100 + (mensaje[14] - '0') * 10 + (mensaje
        [15] - '0'); // Velocidad en mm/min
    myStepper2.setMaxSpeed(velocidad2 / 60 * StepsPermm);
    myStepper2.moveTo(objetivo2 * StepsPermm);
    //Serial.println("Mueve motor 2");
    while (myStepper2.isRunning()) {
        if (digitalRead(endstop1) == HIGH || digitalRead(
            endstop2) == HIGH || digitalRead(endstop3) == HIGH ||
            digitalRead(PanicButton) == HIGH) {
            myStepper1.stop();
            myStepper2.stop();
            myStepper3.stop();
        }
        myStepper2.run();
    }
}

//moving the stepper 3 whith G code
if (mensaje[3] == 'Z') {
    double objetivo3 = (mensaje[4] - '0') * ((mensaje[5] - '0'
        ) * 100 + (mensaje[6] - '0') * 10 + (mensaje[7] - '0')
        + (mensaje[9] - '0') * 0.1 + (mensaje[10] - '0') *
        0.01); //posicion en mm
    double velocidad3 = (mensaje[12] - '0') * 1000 + (mensaje
        [13] - '0') * 100 + (mensaje[14] - '0') * 10 + (mensaje

```

```

    [15] - '0'); // Velocidad en mm/min
myStepper3.setMaxSpeed(velocidad3 / 60 * StepsPermm);
//Serial.println("Mueve motor 3");
myStepper3.moveTo(objetivo3 * StepsPermm);
while (myStepper3.isRunning()) {
    if (digitalRead(endstop1) == HIGH || digitalRead(
        endstop2) == HIGH || digitalRead(endstop3) == HIGH ||
        digitalRead(PanicButton) == HIGH) {
        myStepper1.stop();
        myStepper2.stop();
        myStepper3.stop();
    }
    myStepper3.run();
}
}
}
}

```

```

void Homefunc() {
    digitalWrite(LED_BUILTIN, LOW);

    if (digitalRead(enablePin1) == LOW || digitalRead(enablePin2
        ) == LOW || digitalRead(enablePin3) == HIGH) {
        digitalWrite(LED_BUILTIN, HIGH);
        return;
    }
    // Freno los tres motores
    myStepper1.stop();
    myStepper2.stop();
    myStepper3.stop();
    while (myStepper1.isRunning() || myStepper2.isRunning() ||
        myStepper3.isRunning()) {
        if (digitalRead(endstop1) == HIGH || digitalRead(endstop2
            ) == HIGH || digitalRead(endstop3) == HIGH ||
            digitalRead(PanicButton) == HIGH) {
            myStepper1.stop();
            myStepper2.stop();
            myStepper3.stop();
            estado = Alarma;
            digitalWrite(LED_BUILTIN, HIGH);

```

```
    return ;
}
myStepper1.run() ;
myStepper2.run() ;
myStepper3.run() ;
}

// Home motor 1
myStepper1.setAcceleration(150 * StepsPermm); // defino la
    aceleracion para frenar al llegar al endstop
myStepper1.setSpeed(-10 * StepsPermm); // defino la
    velocidad para llegar al endstop

while (digitalRead(endstop1) == LOW) {
    if (digitalRead(PanicButton) == HIGH) {
        myStepper1.stop() ;
        myStepper2.stop() ;
        myStepper3.stop() ;
        estado = Alarma;
        digitalWrite(LED_BUILTIN, HIGH);
        return ;
    }
    myStepper1.runSpeed() ;
}
myStepper1.stop() ;

while (myStepper1.isRunning()) {
    if (digitalRead(PanicButton) == HIGH) {
        myStepper1.stop() ;
        myStepper2.stop() ;
        myStepper3.stop() ;
        estado = Alarma;
        digitalWrite(LED_BUILTIN, HIGH);
        return ;
    }
    myStepper1.run() ;
}
```

```
myStepper1.setSpeed(20 * StepsPermm); // defino la velocidad
    para soltar al endstop
```

```
while (digitalRead(endstop1) == HIGH) {
    if (digitalRead(PanicButton) == HIGH) {
        myStepper1.stop();
        myStepper2.stop();
        myStepper3.stop();
        estado = Alarma;
        digitalWrite(LED_BUILTIN, HIGH);
        return;
    }
    myStepper1.runSpeed();
}
```

```
myStepper1.move(89.25 * StepsPermm); //cero*StepsPermm);
    //89.25
myStepper1.setMaxSpeed(30 * StepsPermm); //defino la
    velocidad para llegar al cero
//Serial.println("endpoint lose");
```

```
while (myStepper1.isRunning()) {
    if (digitalRead(endstop1) == HIGH || digitalRead(endstop2)
        == HIGH || digitalRead(endstop3) == HIGH ||
        digitalRead(PanicButton) == HIGH) {
        myStepper1.stop();
        myStepper2.stop();
        myStepper3.stop();
        estado = Alarma;
        digitalWrite(LED_BUILTIN, HIGH);
        return;
    }
    myStepper1.run();
}
```

```
myStepper1.stop();
```

```
while (myStepper1.isRunning()) {
```

```

if (digitalRead(endstop1) == HIGH || digitalRead(endstop2)
    == HIGH || digitalRead(endstop3) == HIGH ||
    digitalRead(PanicButton) == HIGH) {
    myStepper1.stop();
    myStepper2.stop();
    myStepper3.stop();
    estado = Alarma;
    digitalWrite(LED_BUILTIN, HIGH);
    return;
}
myStepper1.run();
}
myStepper1.setCurrentPosition(0);

//Home motor 2

myStepper2.setAcceleration(150 * StepsPermm); // defino la
    aceleracion para frenar al llegar al endstop
myStepper2.setSpeed(-10 * StepsPermm); // defino la
    velocidad para llegar al endstop

while (digitalRead(endstop2) == LOW) {
    if (digitalRead(PanicButton) == HIGH) {
        myStepper1.stop();
        myStepper2.stop();
        myStepper3.stop();
        estado = Alarma;
        digitalWrite(LED_BUILTIN, HIGH);
        return;
    }
    myStepper2.runSpeed();
}
myStepper2.stop();

while (myStepper2.isRunning()) {
    if (digitalRead(PanicButton) == HIGH) {
        myStepper1.stop();
        myStepper2.stop();
        myStepper3.stop();
    }
}

```

```
    estado = Alarma;
    digitalWrite(LED_BUILTIN, HIGH);
    return;
}
myStepper2.run();
}

myStepper2.setSpeed(20 * StepsPermm); // defino la velocidad
    para soltar al endstop

while (digitalRead(endstop2) == HIGH) {
    if (digitalRead(PanicButton) == HIGH) {
        myStepper1.stop();
        myStepper2.stop();
        myStepper3.stop();
        estado = Alarma;
        digitalWrite(LED_BUILTIN, HIGH);
        return;
    }
    myStepper2.runSpeed();
}

myStepper2.move(19 * StepsPermm); //cero*StepsPermm);
    //89.25
myStepper2.setMaxSpeed(20 * StepsPermm); //defino la
    velocidad para llegar al cero
//Serial.println("endpoint lose");

while (myStepper2.isRunning()) {
    if (digitalRead(endstop1) == HIGH || digitalRead(endstop2)
        == HIGH || digitalRead(endstop3) == HIGH ||
        digitalRead(PanicButton) == HIGH) {
        myStepper1.stop();
        myStepper2.stop();
        myStepper3.stop();
        estado = Alarma;
        digitalWrite(LED_BUILTIN, HIGH);
        return;
    }
}
```

```

    myStepper2.run();
}

myStepper2.stop();

while (myStepper2.isRunning()) {
    if (digitalRead(endstop1) == HIGH || digitalRead(endstop2)
        == HIGH || digitalRead(endstop3) == HIGH ||
        digitalRead(PanicButton) == HIGH) {
        myStepper1.stop();
        myStepper2.stop();
        myStepper3.stop();
        estado = Alarma;
        digitalWrite(LED_BUILTIN, HIGH);
        return;
    }
    myStepper2.run();
}
myStepper2.setCurrentPosition(0);

// Home motor 3

myStepper3.setAcceleration(10 * Stepspermmch); // defino la
    aceleracion para frenar al llegar al endstop
myStepper3.setSpeed(-2 * Stepspermmch); // defino la
    velocidad para llegar al endstop

while (digitalRead(endstop3) == LOW) {
    if (digitalRead(PanicButton) == HIGH) {
        myStepper1.stop();
        myStepper2.stop();
        myStepper3.stop();
        estado = Alarma;
        digitalWrite(LED_BUILTIN, HIGH);
        return;
    }
    myStepper3.runSpeed();
}
myStepper3.stop();

```

```

while (myStepper3.isRunning()) {
  if (digitalRead(PanicButton) == HIGH) {
    myStepper1.stop();
    myStepper2.stop();
    myStepper3.stop();
    estado = Alarma;
    digitalWrite(LED_BUILTIN, HIGH);
    return;
  }
  myStepper3.run();
}

```

```

myStepper3.setSpeed(1 * Stepspermmch); // defino la
    velocidad para soltar al endstop

```

```

while (digitalRead(endstop3) == HIGH) {
  if (digitalRead(PanicButton) == HIGH) {
    myStepper1.stop();
    myStepper2.stop();
    myStepper3.stop();
    estado = Alarma;
    digitalWrite(LED_BUILTIN, HIGH);
    return;
  }
  myStepper3.runSpeed();
}

```

```

myStepper3.move(4 * Stepspermmch); //cero*StepsPermm);
    //89.25
myStepper3.setMaxSpeed(20 * Stepspermmch); //defino la
    velocidad para llegar al cero
//Serial.println("endpoint lose");

```

```

while (myStepper3.isRunning()) {
  if (digitalRead(endstop1) == HIGH || digitalRead(endstop2)
    == HIGH || digitalRead(endstop3) == HIGH ||
    digitalRead(PanicButton) == HIGH) {
    myStepper1.stop();

```

```
    myStepper2.stop();
    myStepper3.stop();
    estado = Alarma;
    digitalWrite(LED_BUILTIN, HIGH);
    return;
}
myStepper3.run();
}

myStepper3.stop();

while (myStepper3.isRunning()) {
    if (digitalRead(endstop1) == HIGH || digitalRead(endstop2)
        == HIGH || digitalRead(endstop3) == HIGH ||
        digitalRead(PanicButton) == HIGH) {
        myStepper1.stop();
        myStepper2.stop();
        myStepper3.stop();
        estado = Alarma;
        digitalWrite(LED_BUILTIN, HIGH);
        return;
    }
    myStepper3.run();
}
myStepper3.setCurrentPosition(0);

digitalWrite(LED_BUILTIN, HIGH);
}

void BucleCtrlMov () {
    if (myStepper1.isRunning()) {
        myStepper1.run();
    }
    if (myStepper2.isRunning()) {
        myStepper2.run();
    }
    if (myStepper3.isRunning()) {
        myStepper3.run();
    }
}
```

}

```
void BucleCtrl () {

    // PanicButton chek
    if (estado != Disable && estado != Espera) {
        // PanicButton chek
        if (digitalRead(PanicButton) == HIGH) {
            //Serial.println("Boton anti panico activado");
            estado = Disable;
        }
    }

    if (estado == Alarma) {
        //Serial.println("Final de carrera activado!");
        myStepper1.stop();
        myStepper2.stop();
        myStepper3.stop();
        myStepper1.setAcceleration(125000);
        myStepper2.setAcceleration(125000);
        myStepper3.setAcceleration(125000);

        if (myStepper1.isRunning() != true && myStepper2.isRunning
            () != true && myStepper3.isRunning() != true) {
            estado = Alarma1;
            //Serial.println("PARADA EMERGENCIA");
        }
    }

    if (estado == Alarma1) {
        if (digitalRead(endstop1) == HIGH ) {
            if (myStepper1.currentPosition() < 1) {
                myStepper1.move(1 * StepsPermm); //me alejo 10 mm del
                endstop
                myStepper1.setMaxSpeed(5 * StepsPermm); //defino la
                velocidad para alejarme del endstop
            }
            if (myStepper1.currentPosition() > 1) {
```

```

    myStepper1.move(-1 * StepsPermm); //me alejo 10 mm del
        endstop
    myStepper1.setMaxSpeed(5 * StepsPermm); //defino la
        velocidad para alejarme del endstop
}
}
if (digitalRead(endstop2) == HIGH ) {
    if (myStepper2.currentPosition() < 1) {
        myStepper2.move(1 * StepsPermm); //me alejo 10 mm del
            endstop
        myStepper2.setMaxSpeed(5 * StepsPermm); //defino la
            velocidad para alejarme del endstop
    }
    if (myStepper2.currentPosition() > 1) {
        myStepper2.move(-1 * StepsPermm); //me alejo 10 mm del
            endstop
        myStepper2.setMaxSpeed(5 * StepsPermm); //defino la
            velocidad para alejarme del endstop
    }
}
}
if (digitalRead(endstop3) == HIGH ) {
    if (myStepper3.currentPosition() < 1) {
        myStepper3.move(1 * Stepspermmch); //me alejo 10 mm
            del endstop
        myStepper3.setMaxSpeed(5 * Stepspermmch); //defino la
            velocidad para alejarme del endstop
    }
    if (myStepper3.currentPosition() > 1) {
        myStepper3.move(-1 * Stepspermmch); //me alejo 10 mm
            del endstop
        myStepper3.setMaxSpeed(5 * Stepspermmch); //defino la
            velocidad para alejarme del endstop
    }
}
}
if (myStepper1.isRunning() != true && myStepper2.isRunning
    () != true && myStepper3.isRunning() != true) {
    estado = Disable;
    //Serial.println("PARADA EMERGENCIA");
}
}

```



```
}

if (estado == Espera) {
}

// DEFINE THE ENABLE LOGIC
if (estado == Enable) {
    digitalWrite(enablePin1 , HIGH);
    digitalWrite(enablePin2 , HIGH);
    digitalWrite(enablePin3 , LOW);
    estado = Espera;
    //Serial.println("Motores habilitados");
}

// DEFINE THE DISABLE LOGIC
if (estado == Disable) {
    digitalWrite(enablePin1 , LOW);
    digitalWrite(enablePin2 , LOW);
    digitalWrite(enablePin3 , HIGH);
    estado = Espera;
    //Serial.println("Motores deshabilitados");
}

// DEFINE THE STOPPING LOGIC
if (estado == Stop) {
    myStepper1.setAcceleration(25000);
    myStepper2.setAcceleration(25000);
    myStepper3.setAcceleration(25000);
    while (myStepper1.isRunning() || myStepper2.isRunning() ||
        myStepper3.isRunning()) {
        myStepper1.run();
        myStepper2.run();
        myStepper3.run();
    }
    estado = Espera;
}

// DEFINE THE TRAPEZOIDAL ROUTINE LOGIC
if (estado == Rutina) {
```

```

//Serial.println("Rutina");
// Set all the parameters for stepper 1
myStepper1.setMaxSpeed(maxSpeed1); // Steps per second
myStepper1.setAcceleration(accel1); // Steps per second
    per second
myStepper1.moveTo(-moveto1 / 2); // Steps

// Set all the parameters for stepper 2
myStepper2.setMaxSpeed(maxSpeed2); // Steps per second
myStepper2.setAcceleration(accel2); // Steps per second
    per second
myStepper2.moveTo(moveto2 / 2); // Steps

// Set all the parameters for stepper 3
myStepper3.setMaxSpeed(maxSpeed3); // Steps per second
myStepper3.setAcceleration(accel3); // Steps per second
    per second
myStepper3.moveTo(moveto3 / 2); // Steps

estado = Rutina1;
//Serial.println("Rutina1");
}

if (estado == Rutina1) {
    if (digitalRead(endstop1) == HIGH || digitalRead(endstop2)
        == HIGH || digitalRead(endstop3) == HIGH) {
        estado = Alarma;
    }
    if (myStepper1.distanceToGo() == 0) {
        myStepper1.moveTo(-myStepper1.currentPosition());
    }
    //myStepper1.run();

    if (myStepper2.distanceToGo() == 0) {
        myStepper2.moveTo(-myStepper2.currentPosition());
    }
    //myStepper2.run();

```

```

    if (myStepper3.distanceToGo() == 0) {
        myStepper3.moveTo(-myStepper3.currentPosition());
    }
    //myStepper3.run();
}
}

void BucleEscritura (int N_muestras) {

    if (cont_lect > N_muestras) {
        LECTURA = 0;
        cont_lect = 0;
    }
    else {
        //Az
        //4 bytes a 400kHz 100us masomenos
        //__MPU_reg_read_bulk_sw(MPU_DIR, 0x3f, &buff_rec[0], 2);
        //Ax
        //4 bytes a 400kHz 100us masomenos
        //__MPU_reg_read_bulk_sw(MPU_DIR, 0x3b, &buff_rec[0], 2);
        //Ay
        //4 bytes a 400kHz 100us masomenos
        //__MPU_reg_read_bulk_sw(MPU_DIR, 0x3d, &buff_rec[0], 2);

        uint16_t valX=analogRead(accelX_pin);
        //uint16_t valY=analogRead(accelY_pin);

        buff_out [0] = valX>>8;
        buff_out [1] = valX & 0x00ff;

        //750rpm como maximo, 400pasos/rev -> 5000 steps/sec
        //cada 0.2ms/2 hay que llamar a run();
        int16_t aux=myStepper2.speed(); //pasos/sec 5000
        buff_out [2] = aux >> 8;
        buff_out [3] = aux & 0x00ff;
    }
    cont_lect++;
}
/*

```

```

void __MPU_reg_write_sw(uint8_t DIR, uint8_t reg, uint8_t
    value) {
    sw.begin();
    sw.beginTransmission(DIR);
    sw.write(reg); // Access the first register
    sw.write(value);
    sw.endTransmission();
}

void __MPU_reg_read_bulk_sw(uint8_t DIR, uint8_t reg, uint8_t *
    buf, uint32_t count) {
    sw.begin();
    sw.beginTransmission(DIR);
    sw.write(reg); // Access the first register
    sw.endTransmission();

    uint8_t numBytes=sw.requestFrom(DIR, (uint8_t)count);
    uint8_t i;
    for(i=0;i<numBytes;i++){
        buf[i]= sw.read();
    }
    sw.endTransmission();
}

void MPU_init(){
    __MPU_reg_write_sw(MPU_DIR,0x6B,0xff); //off
    __MPU_reg_write_sw(MPU_DIR,0x6B,0); //on
    __MPU_reg_write_sw(MPU_DIR, 0x19, 9); //200hz sampling
        1000/(1+4)
    __MPU_reg_write_sw(MPU_DIR, 0x1A, 0x04); //1khz con pasabajo

    //__MPU_reg_write_sw(MPU_DIR, 0x1D, 0x04); //1khz con
        pasabajo
}

*/

```

```
void setup() {
    // Set the modes of the pins
    pinMode(enablePin1 , OUTPUT);
    pinMode(enablePin2 , OUTPUT);
    pinMode(enablePin3 , OUTPUT);
    pinMode(endstop1 , INPUT);
    pinMode(endstop2 , INPUT);
    pinMode(endstop3 , INPUT);
    pinMode(PanicButton , INPUT_PULLUP);
    //pinMode(RedPin , OUTPUT);
    //pinMode(GreenPin , OUTPUT);
    pinMode(LED_BUILTIN , OUTPUT);
    // begin the serial communication
    Serial.begin(115200);
    // set the maximum speed
    myStepper1.setMaxSpeed(200 * StepsPermm); // Steps per
        second
    myStepper2.setMaxSpeed(200 * StepsPermm); // Steps per
        second
    myStepper3.setMaxSpeed(200 * StepsPermm); // Steps per
        second

    // Disable all the steppers to have to enable it to begin a
        movement
    digitalWrite(enablePin1 , LOW);
    digitalWrite(enablePin2 , LOW);
    digitalWrite(enablePin3 , HIGH);

    digitalWrite(LED_BUILTIN , HIGH);

    //set the PanicButton Pin at HIGH
    digitalWrite(PanicButton , HIGH);

    //Calculate the variables
    CalculateVariables();

    tim1_init_arduino();
    Tim1->resume();
}
```

```

tim2_init_arduino(100);
Tim2->resume();

//sw_init(); //soft serial_i2c init, 400kHz

//analogReadResolution(12);

// Define the begining state
estado = Espera;
}

void Blink() {
    if (digitalRead(LED_BUILTIN) == HIGH) {
        digitalWrite(LED_BUILTIN, LOW);
    }
    else {
        digitalWrite(LED_BUILTIN, HIGH);
    }
}

void loop() {

    // Read serial inputs
    if (Serial.available() > 0)
    {
        entrada = Serial.read();
        serialRecieve();
    }

    // Enter in to the Communication loop every timeout of TIM1
    if (timeout_T1 == 1)
    {
        timeout_T1 = 0;
        contT1++;
        contT1_++;
        if (contT1 == 10) {
            contT1 = 0;
            BucleCtrl();
        }
    }
}

```

```
    }  
    if (contT1_ == 1000) {  
        contT1_ = 0;  
        Blink();  
    }  
    BucleCtrlMov();  
}  
  
if (timeout_T2 == 1 && LECTURA == 1)  
{  
    timeout_T2 = 0;  
    BucleEscritura(NMUESTRAS);  
    Serial.write(&buff_out[0], 4);  
}  
}
```


Apéndice F

Códigos utilizados para la programación de la interfaz

En este apéndice se exponen los códigos utilizado la programación de la interfaz con el usuario. Este código se lo redactó en lenguaje C++ y se lo corrió en el IDE de QT. A continuación se exponen tanto los códigos de los encabezados como los de los archivos fuente:

F.1. Colimador-Com-serie-BP-QT-V7.pro

```
#
# Project created by QtCreator 2023-05-23T14:35:30
#
#-----

QT      += core gui
QT      += serialport

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = Colimador-Com-serie-BP-QT-V7
TEMPLATE = app

# The following define makes your compiler emit warnings if
# you use
# any feature of Qt which has been marked as deprecated (the
# exact warnings
```

```

# depend on your compiler). Please consult the documentation
  of the
# deprecated API in order to know how to port your code away
  from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if you use
  deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to
  a certain version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    #
  disables all the APIs deprecated before Qt 6.0.0

SOURCES += \
    main.cpp \
    mainwindow.cpp \
    comunicacion.cpp

HEADERS += \
    mainwindow.h \
    comunicacion.h \
    comandos.h

FORMS += \
    mainwindow.ui

```

F.2. comandos.h

```

#ifndef COMANDOS_H
#define COMANDOS_H

#include <stdint.h>

#define NDATOS_IN 4
#define NDATOS_OUT 10

```

```
struct pkg_env{
    uint8_t hdr;
    uint8_t CMD;
    uint16_t D1;
    uint16_t D2;
    uint16_t D3;
    int8_t chs;
    uint8_t tail;
}__attribute__((packed));
```

```
struct pkg_rec{
    //int16_t Ax;
    int16_t Ay;
    //int16_t Az;
    //int16_t gx;
    //int16_t gy;
    //int16_t gz;
    int16_t U;
    //uint8_t Endstops;
    //int8_t chs;
}__attribute__((packed));
```

```
struct log_rec{
    double N;
    double T_ide;
    //double Ax;
    double Ay;
    //double Az;
    //double gx;
    //double gy;
    //double gz;
    double U; //Velocidad
    //double Endstops;
}__attribute__((packed));
```

```
typedef struct log_rec log_rec;
typedef struct pkg_rec pkg_rec;
typedef struct pkg_env pkg_env;
```

```

typedef enum serial_state {ON,OFF,TIMEOUT,SINC,NOSINC}
    serial_state;
typedef enum comando{ENABLE='E', DISABLE='D', STOP='S', HOME='
    H',
    RUTINA='R', LECTURA='L'}comando;

#endif // COMANDOS.H

```

F.3. comunicacion.h

```

#ifndef COMUNICACION_H
#define COMUNICACION_H

#include <QThread>
#include <QtCore>
#include <QtSerialPort/QtSerialPort>
#include <fstream>
#include <time.h>
#include "comandos.h"
#include <iomanip> // std::setprecision

extern serial_state SerialState;

class comunicacion:public QThread{
    Q_OBJECT

signals:
    void send(float);
    void debug(QString);

private:
    double dt;
    double t_exp;
    uint16_t Extra;
    unsigned int NMUESTRAS;

```

```
    unsigned int index;
    double t_ide;
    log_rec * log;

    uint8_t buffer_env [NDATOS_OUT]={0}; //10 bytes@115200 =
        0.9 ms
    uint8_t buffer_rec [NDATOS_IN]={0}; //4 bytes@115200 =
        0.35 ms //esto me limita a una frecuencia maxima de
        muestreo de 2800Hz

    uint8_t res [2]={0};

    bool quit;
    QElapsedTimer timer;
    std::string name_log;
    QString nameCOM;
    comando comand;

    //void delay(double t);
    void imprimir(bool encabezado);
    void inicializar_estructuras();
    void inicializar_micro(double Tmuestreo, QSerialPort *
        serial);
    void procesar_recepcion();
    void procesar_envio();
    int lectura_datos(QSerialPort *serial, int timeout);
    void copiar_log();

    uint16_t check_multiple(uint16_t n);

public:

    pkg_rec pkg_r;
    pkg_env pkg_e;

    //Apagado del thread
    QMutex mutex_apagado;
    bool exit=false;
```

```

//Mainwindow

void escritura(QSerialPort *serial , int timeout);

int Serial_open(const QString &nombre, qint32 baud,
    QSerialPort *serial);
void Serial_close(QSerialPort *serial);
void Serial_send(uint8_t * buffer, uint8_t size ,
    QSerialPort *serial , int timeout);
int Serial_read(uint8_t * buffer, uint8_t Ndatos,
    QSerialPort *serial , int timeout, bool &
    timeout_ocurred);
void start_comunic(void);

// Aca determino el tiempo de experimento, periodo de
// muestreo y el nombre del archivo para guardar los
// datos
comunicacion(QObject *parent=NULL, double t_experim=10.0,
    double DT=10, std::string nombre="log.txt"):
    QThread(parent), dt(DT), t_exp(t_experim), name_log(
        nombre)
{
    index=0;
    NMUESTRAS=t_exp/DT*1000;
    exit=false;
    log= new log_rec [NMUESTRAS];
    inicializar_estructuras();
    name_log=strdup(nombre.c_str());
    Extra=0;
}

uint16_t reescribir(int t_experim=3.0, int DT=5.0, std::
    string nombre="log.txt",
        const QString & N.COM="COM15",
        comando com=LECTURA, uint8_t extra
        =0b00100000) //cambiar COM segun
        corresponda
{

```

```

//extra=0b00100000 means 25fps, +-2g, X axe

dt=DT; // en ms
t_exp=t_experim;
name_log=nombre;
index=0;
NMUESTRAS=static_cast<unsigned int>(int(t_exp/DT
    *1000));
//multiplico por 1000 ms/s para cancelar unidades
//NMUESTRAS=check_multiple(NMUESTRAS);
comand=com;

qDebug()<<"NMUESTRAS: -"<<NMUESTRAS<<endl;

Extra=extra;

exit=false;
delete [] log;
log= new log_rec [NMUESTRAS];

inicializar_estructuras();
nameCOM=N_COM;

debug("termino reescritura");

return NMUESTRAS;
}

virtual ~comunicacion() {}

void run() override{

    this->setTerminationEnabled(true);

    //si hago que el serial viva aca y este thread sea
    el
    bloqueante, no hay problema
    QSerialPort * serial_thread=new QSerialPort;

```

```

if( Serial_open (nameCOM,115200 ,serial_thread) == -1
){
    debug("No abrio puerto: "+nameCOM);
    return;
}

serial_thread->close();
serial_thread->open(QIODevice::ReadWrite);

debug(" Puerto: "+nameCOM);
quit=false;
exit=false;

static bool timer_iniciado=false;
if(!timer_iniciado){
    timer.start();
    timer_iniciado=true;
}
else
    timer.restart();

index=0;

//Escribo periodo de muestreo al micro
inicializar_micro(dt,serial_thread);

uint16_t n_paquetes=NMUESTRAS;

double t_ide_init=1e-9*timer.nsecsElapsed();

while(quit==false && index<n_paquetes){
    ////////////bucle controlado//////////

    if(index==0){
        t_ide=0.0;
    }else if (index==1){

```



```

        t_ide=dt/1000;
        t_ide_init=1e-9*timer.nsecsElapsed();
    } else {
        t_ide=1e-9*timer.nsecsElapsed()-t_ide_init+
            dt/1000;
    }

    //leo el buffer
    lectura_datos(serial_thread,100);
    ++index; //esto hace correr el log

    mutex_apagado.lock();
    if(exit) quit=true;
    mutex_apagado.unlock();

    if((uint16_t(1.0*index/n_paquetes))%10==0)
        emit send(1.0*index/n_paquetes);

    //debug("otra muestra");

}
//Escribo cuantas muestras quiero leer

/* if(quit){
    pkg_e.hdr=': ';
    pkg_e.CMD=comand;
    pkg_e.D1=t_exp;
    pkg_e.D2=NMUESTRAS;
    pkg_e.D3=0;
    pkg_e.chs=0;
    pkg_e.chs+=static_cast<int8_t>(pkg_e.CMD);
    pkg_e.chs+=static_cast<int8_t>(((pkg_e.D1)>>8);
    pkg_e.chs+=static_cast<int8_t>(((pkg_e.D1)& 0
        x00ff));
    pkg_e.chs+=static_cast<int8_t>(((pkg_e.D2)>>8);
    pkg_e.chs+=static_cast<int8_t>(((pkg_e.D2)& 0
        x00ff));
    pkg_e.chs+=static_cast<int8_t>(((pkg_e.D3)>>8);

```

```

        pkg_e.chs+=static_cast<int8_t>((pkg_e.D3)& 0
            x00ff);;
        pkg_e.tail='\n';
        escritura(&serial_thread,10);
    }*/
    serial_thread->close();
    delete serial_thread;

    imprimir(true); //encabezado si/no

    return;
}

};

#endif // COMUNICACION_H

```

F.4. mainwindow.h

```

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QTimer>
#include <stdint.h>
#include <QElapsedTimer>
#include <QtCore>
#include <QTextEdit>
#include "comunicacion.h"
#include "comandos.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow

```

```
{
    Q_OBJECT

signals:
    void stop(bool);

public:
    explicit MainWindow(QWidget *parent = 0);
    static QTextEdit * DebugLine;
    ~MainWindow();

public slots:
    void experiment_finished(void);
    void recieve(float value);
    void Debug(QString);

private slots:
    void on_Iniciar_clicked();

    void on_Enable_clicked();

    void on_Disable_clicked();

    void on_Stop_clicked();

    void on_Home_clicked();

    void on_Rutina_clicked();

    //uint8_t calculaChecksum (uint8_t* dato, uint largo);

    void on_Terminar_clicked();

    void on_leer_clicked();

    void on_new_variables_clicked();

    void on_new_variables2_clicked();
```

```

    void on_new_variables3_clicked ();

private:
    Ui::MainWindow *ui;
    comunicacion COM;
    QSerialPort Serial;
    QTimer *timer1;
    serial_state SerialState;
    comando comdo;
    QString name_COM;
    uint32_t index;
    double DT;

    QString style_start , style_stop;

    //QString style_start , style_stop;

    int Serial_open(const QString &nombre, qint32 baud);
    QStringList verificar_COM(uint max_COM);
    void habilitar_pantalla(bool state);
    //void guardar_log();

};

#endif // MAINWINDOW.H

```

F.5. comunicacion.cpp

```

#include "comunicacion.h"

/*void comunicacion::delay(double t){
    struct timespec nano;

    nano.tv_sec=0;//sec;
    nano.tv_nsec=t*1e9;
    nanosleep(&nano,NULL);
}

```

```
    }*/  
  
    void comunicacion::inicializar_micro(double Tmuestreo,  
        QSerialPort *serial){  
  
        if(serial->clear(QSerialPort::AllDirections));  
        else{  
            char a;  
            while(serial->read(&a,1));  
        }  
  
        if (Tmuestreo > 1000) {  
            Tmuestreo = 1000;  
        }  
        if (Tmuestreo < 1) {  
            Tmuestreo = 1;  
        }  
  
        uint16_t N_top_timer=Tmuestreo*10;  
  
        debug(" Valor Top:"+QString::number(N_top_timer));  
        debug(" Nmuestras:"+QString::number(NMUESTRAS));  
  
        pkg_e.hdr=(uint8_t) ': ' ;  
        pkg_e.CMD=LECTURA;  
        pkg_e.D1=N_top_timer;  
        pkg_e.D2=NMUESTRAS;  
        pkg_e.D3=0;  
        pkg_e.chs=0;  
        pkg_e.chs+=static_cast<int8_t>(pkg_e.CMD);  
        pkg_e.chs+=static_cast<int8_t>((pkg_e.D1)>>8);  
        pkg_e.chs+=static_cast<int8_t>((pkg_e.D1)& 0x00ff);  
        pkg_e.chs+=static_cast<int8_t>((pkg_e.D2)>>8);  
        pkg_e.chs+=static_cast<int8_t>((pkg_e.D2)& 0x00ff);  
        pkg_e.chs+=static_cast<int8_t>((pkg_e.D3)>>8);  
        pkg_e.chs+=static_cast<int8_t>((pkg_e.D3)& 0x00ff);  
        pkg_e.tail='\n';  
  
        debug(" Comienza experimento");
```

```

    escritura (serial ,30);
}

void comunicacion::imprimir(bool encabezado=false) {

    debug("Guardando archivo: " +QString::fromStdString(
        name_log));

    std::ofstream myfile;
    myfile.open(name_log); //por default es output

    if(encabezado){
        myfile<<"# t_ide [s] -Ay[m/s2] -Vy-"<<std::endl;
    }
    for(unsigned int i=0;i<NMUESTRAS;++i){
        myfile<< log[i].N<<" "<<
            log[i].T_ide<<" "<<
            //log[i].Ax<<" "<<
            log[i].Ay<<" "<<
            //log[i].Az<<" "<<
            //log[i].gx<<" "<<
            //log[i].gy<<" "<<
            //log[i].gz<<" "<<
            log[i].U<<" "<<
            //std::setprecision(3)<< log[i].Endstops
            <<
            std::endl;
    }
    myfile.close();

    debug("-----");
}

/*uint16_t comunicacion::check_multiple(uint16_t n){
    if(n%3==0)
        return n;
    if((n-1)%3==0)
        return n-1;
}

```

```
        if ((n-2)%3==0)
            return n-2;
        return n;
    }*/

int comunicacion::Serial_open(const QString &nombre, qint32
    baud, QSerialPort *serial)
{
    serial->setPortName(nombre); //puerto
    if (!serial->open(QIODevice::ReadWrite)) {
        return -1;
    }
    serial->setBaudRate(baud);
    if (serial->error() != QSerialPort::NoError) {
        qDebug() << "Error: -" << serial->error() << endl;
        return -1;
    }
    return 0;
}

void comunicacion::Serial_close(QSerialPort *serial)
{
    serial->close();
}

void comunicacion::Serial_send(uint8_t * buffer, uint8_t
    size, QSerialPort *serial, int timeout)
{
    const QByteArray requestData = QByteArray((char*) buffer
        , size);
    serial->write(requestData);
    serial->waitForBytesWritten(timeout); //2ms espera
    //qDebug() << "envio mensaje" << endl;
}

int comunicacion::Serial_read(uint8_t * buffer, uint8_t
    Ndatos, QSerialPort *serial, int timeout, bool &
    timeout_ocurred){
    uint16_t cont;
```

```

cont=serial->read((char*) buffer ,Ndatos);
bool no_timeout_ocurred =serial->waitForReadyRead(
    timeout); //msecs de espera

if(no_timeout_ocurred){
    timeout_ocurred=false;
    //qDebug()<<"leo mensaje"<<": "<<cont<<endl;
}
else {
    timeout_ocurred=true;
    //qDebug()<<"timeout"<<endl;
}
return cont;
}

int comunicacion::lectura_datos(QSerialPort *serial ,int
timeout)
{
    //leo bloqueante hasta que tengo lo que quiero , periodo
    de muestreo lo fija el micro
    bool timeout_occured=false;
    //uint16_t contador_timeout=0;
    int retu=Serial_read(&buffer_rec [0] ,NDATOS_IN,serial ,
        timeout ,timeout_occured); //leo buffer4
    //qDebug()<<retu<<endl;
    //if(timeout_occured) contador_timeout++;
    while(retu<NDATOS_IN){
        retu+=Serial_read(&buffer_rec [retu] ,NDATOS_IN-retu ,
            serial ,timeout ,timeout_occured); //leo buffer
        //mutex_apagado.lock ();
        if(exit==true) {
            retu=NDATOS_IN;
            qDebug()<<" Break"<<endl;
            break;
        }
        //mutex_apagado.unlock ();

        //if(timeout_occured) contador_timeout++;

```



```

        //if (contador_timeout>2) {
            // qDebug()<<"timeout"<<endl;
            //break;
        //}
        qDebug()<<retu<<" -"<<serial->portName()<<endl;
        //qDebug()<<buffer_rec[retu]<<endl;
    }

    // Aca tengo que swipear los datos porque la serie manda
    // de a 8 bytes y no de a 16 bytes
    //pkg_r.Ax=static_cast<int16_t>((int16_t(buffer_rec
        [0]<<8))|(int16_t(buffer_rec[1])));
    pkg_r.Ay=static_cast<int16_t>((int16_t(buffer_rec[0]<<8)
        )|(int16_t(buffer_rec[1])));
    //pkg_r.Az=static_cast<int16_t>((int16_t(buffer_rec
        [4]<<8))|(int16_t(buffer_rec[5])));
    //pkg_r.gx=static_cast<int16_t>((int16_t(buffer_rec
        [6]<<8))|(int16_t(buffer_rec[7])));
    //pkg_r.gy=static_cast<int16_t>((int16_t(buffer_rec
        [8]<<8))|(int16_t(buffer_rec[9])));
    //pkg_r.gz=static_cast<int16_t>((int16_t(buffer_rec
        [10]<<8))|(int16_t(buffer_rec[11])));
    pkg_r.U=(buffer_rec[2]<<8)|buffer_rec[3];
    //pkg_r.Endstops=buffer_rec[14];
    //pkg_r.chs=buffer_rec[15];

    /*pkg_r.Ax=((int16_t)(buffer_rec[0]<<8))|(buffer_rec[1])
        ;
    pkg_r.Ay=100;
    pkg_r.Az=16000;
    pkg_r.gx=100;
    pkg_r.gy=200;
    pkg_r.gz=300;*/

    //checksum local
    /* uint8_t chs_l=static_cast<uint8_t>(buffer_rec[0])+

```

```

        static_cast<uint8_t>(buffer_rec [1])+static_cast<
            uint8_t>(buffer_rec [2])+
        static_cast<uint8_t>(buffer_rec [3])+static_cast<
            uint8_t>(buffer_rec [4])+
        static_cast<uint8_t>(buffer_rec [5])+static_cast<
            uint8_t>(buffer_rec [6])+
        static_cast<uint8_t>(buffer_rec [7])+static_cast<
            uint8_t>(buffer_rec [8])+
        static_cast<uint8_t>(buffer_rec [9])+static_cast<
            uint8_t>(buffer_rec [10])+
        static_cast<uint8_t>(buffer_rec [11])+static_cast
            <uint8_t>(buffer_rec [12])+
        static_cast<uint8_t>(buffer_rec [13])+static_cast
            <uint8_t>(buffer_rec [14]);

    if (chs_l!=static_cast<uint8_t>(pkg_r.chs)){
        debug("No dio el CHS");
    }*/

    //qDebug()<<retu<<endl;
    copiar_log();
    return retu;
}

void comunicacion::copiar_log()
{
    //qDebug()<<"bef:"<<pkg_r.Ax;
    log[index].N=index;
    log[index].T_ide=t_ide;
    //log[index].Ax=pkg_r.Ax/17500.0*9.8;
    log[index].Ay=pkg_r.Ay;
    //log[index].Az=pkg_r.Az/17500.0*9.8;
    //log[index].gx=pkg_r.gx;
    //log[index].gy=pkg_r.gy;
    //log[index].gz=pkg_r.gz;
    log[index].U=pkg_r.U;
    //log[index].Endstops=pkg_r.Endstops;

    //qDebug()<<"aft:"<<log[index].Ax<<endl;

```

```

    //qDebug ()<<"Copie log"<<endl;
}

void comunicacion::inicializar_estructuras () { // Inicializo
    estructuras de envio , recibo y almacenamiento

    //pkg_r .Ax=14;
    pkg_r .Ay=14;
    //pkg_r .Az=14;
    //pkg_r .gx=14;
    //pkg_r .gy=14;
    //pkg_r .gz=14;
    pkg_r .U=0;
    //pkg_r .Endstops=0;
    //pkg_r .chs=0;

    pkg_e .hdr=': ';
    pkg_e .CMD=0;
    pkg_e .D1=0;
    pkg_e .D2=0;
    pkg_e .D3=0;
    pkg_e .chs=0;
    pkg_e .chs+=static_cast<int8_t >(pkg_e .CMD) ;
    pkg_e .chs+=static_cast<int8_t >((pkg_e .D1)>>8);
    pkg_e .chs+=static_cast<int8_t >((pkg_e .D1)& 0x00ff) ;
    pkg_e .chs+=static_cast<int8_t >((pkg_e .D2)>>8);
    pkg_e .chs+=static_cast<int8_t >((pkg_e .D2)& 0x00ff) ;
    pkg_e .chs+=static_cast<int8_t >((pkg_e .D3)>>8);
    pkg_e .chs+=static_cast<int8_t >((pkg_e .D3)& 0x00ff) ;;
    pkg_e .tail='\n' ;

    for (uint i=0;i<NMUESTRAS;++i)
    {
        log [ i ] .N=14;
        log [ i ] .T_ide=14;
        //log [ i ] .Ax=14;
        log [ i ] .Ay=14;
        //log [ i ] .Az=14;
        //log [ i ] .gx=14;
    }
}

```

```

        //log[i].gy=14;
        //log[i].gz=14;
        log[i].U=0;
        //log[i].Endstops=0;
    }
}

void comunicacion::escritura(QSerialPort *serial ,int timeout
)
{

    // Aca tengo que swipear los datos porque
    //la serie manda de a 8 bytes y no de a 16 bytes
    buffer_env[0]=pkg_e.hdr;
    buffer_env[1]=pkg_e.CMD;
    buffer_env[2]=(pkg_e.D1 >> 8);
    buffer_env[3]=(pkg_e.D1 & 0x00ff);
    buffer_env[4]=(pkg_e.D2 >> 8);
    buffer_env[5]=(pkg_e.D2 & 0x00ff);
    buffer_env[6]=(pkg_e.D3 >> 8);
    buffer_env[7]=(pkg_e.D3 & 0x00ff);
    buffer_env[8]=pkg_e.chs;
    buffer_env[9]=pkg_e.tail;

    Serial_send(&buffer_env[0],NDATOS_OUT,serial ,timeout);

    qDebug()<<" Escribo mensaje"<<endl;

    /*
    //leo bloqueante hasta que tengo lo que quiero, periodo
    de muestreo lo fija el micro
    int retu=Serial_read(&res[0],2,serial ,timeout); //leo
    buffer
    while(retu<2){
        retu+=Serial_read(&res[retu],2-retu ,serial ,timeout);
    }
    */
}

```

```
        //leo buffer
    }

    qDebug()<<static_cast<int>(res[0])<<' '<<static_cast<int>
        >(res[1])<<' '<<static_cast<int>(res[0]-res[1])<<endl
        ;
*/
}
```

F.6. main.cpp

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

F.7. mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QMessageBox>
#include <QTimer>
#include <QTime>
#include <QDate>

QTextEdit * MainWindow::DebugLine = 0;
```

```

double Tiempo_muestreo=4.0; //Tiempo de muestreo en
    segundos
double Periodo_muestreo=0.005; //Periodo de muestreo en
    segundos

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent), ui(new Ui::MainWindow),
    COM(NULL, Tiempo_muestreo, Periodo_muestreo, "log.txt")
{
    ui->setupUi(this);

    DT=ui->tiempo_lect->value();

    SerialState=OFF;

    style_start="font:-63-20pt-\"Ubuntu\";\nbackground-color
        :-qlineargradient"
        "(spread:pad,-x1:0,-y1:0,-x2:1,-y2:0,-stop:1
        -rgba(170,-255,-127,-255));",
    style_stop="font:-63-20pt-\"Ubuntu\";\n
        nbackground-color:-qlineargradient"
        "(spread:pad,-x1:0,-y1:0,-x2:1,-y2:0,
        -stop:1-rgba(255,-125,-100,-255));
        ";

    //setupAccel();

    connect(&COM, SIGNAL(finished()),
        this, SLOT(experiment_finished()));

    connect(&COM, SIGNAL(send(float)),
        this, SLOT(recieve(float)));

    connect(&COM, SIGNAL(debug(QString)),
        this, SLOT(Debug(QString)));
}

MainWindow::~MainWindow()
{

```

```
Serial.close();
SerialState=OFF;
ui->Iniciar->setEnabled(1);
habilitar_pantalla(false);
COM.wait();
COM.deleteLater();
delete ui;

/* if (ui->check_out_log->isChecked() && ui->log_sesion->
    toPlainText() != NULL) {
    guardar_log();
} */
}

int MainWindow::Serial_open(const QString & nombre, qint32
    baud)
{
    Serial.setPortName(nombre);
    if (!Serial.open(QIODevice::ReadWrite)) {
        return -1;
    }
    Serial.setBaudRate(QSerialPort::Baud115200);
    return 0;
}

void MainWindow::habilitar_pantalla(bool state)
{
    ui->Enable->setEnabled(state);
    ui->Disable->setEnabled(state);
    ui->Home->setEnabled(state);
    ui->Rutina->setEnabled(state);
    ui->Stop->setEnabled(state);
    ui->Terminar->setEnabled(state);
    ui->new_variables->setEnabled(state);
    ui->new_variables2->setEnabled(state);
    ui->new_variables3->setEnabled(state);
}
```

```

    ui->per_rutina->setEnabled(state);
    ui->acel_rutina->setEnabled(state);
    ui->long_rutina->setEnabled(state);
    ui->motor_change->setEnabled(state);
}

/*uint8_t MainWindow::calculaChecksum (uint8_t* dato, uint
    largo){ //En largo definir el largo del mensaje menos el
    hdr y menos el chs
    uint8_t chsl=0;
    for(int i=0; i<largo-1; ++i){
        chsl+=*(dato+i);
    }
    return chsl;
}*/

```

```

void MainWindow::on_Iniciar_clicked()
{
    name_COM="COM3";
    if(Serial_open(name_COM, 115200) == 0 ){
        habilitar_pantalla(true);
        /*ui->tiempo_lect->setEnabled(true);
        ui->Tiempo_datos->setEnabled(true);
        ui->leer->setEnabled(true);
        ui->avance->setEnabled(true);
        ui->log_datos->setEnabled(true);*/
        SerialState=ON;
    }
    else{
        QMessageBox alarma;
        alarma.setText("No puede hallarse el puerto");
        alarma.exec();
        SerialState=OFF;
    }
}

```

```

void MainWindow::on_Enable_clicked()

```



```

{
    COM. pkg_e . hdr = ' : ' ;
    COM. pkg_e . CMD = ENABLE ;
    COM. pkg_e . D1 = 0.0 ;
    COM. pkg_e . D2 = 0.0 ;
    COM. pkg_e . D3 = 0.0 ;
    COM. pkg_e . chs = 0 ;
    COM. pkg_e . chs += static_cast < int8_t > ( COM. pkg_e . CMD ) ;
    COM. pkg_e . chs += static_cast < int8_t > ( ( COM. pkg_e . D1 ) >> 8 ) ;
    COM. pkg_e . chs += static_cast < int8_t > ( ( COM. pkg_e . D1 ) & 0
        x00ff ) ;
    COM. pkg_e . chs += static_cast < int8_t > ( ( COM. pkg_e . D2 ) >> 8 ) ;
    COM. pkg_e . chs += static_cast < int8_t > ( ( COM. pkg_e . D2 ) & 0
        x00ff ) ;
    COM. pkg_e . chs += static_cast < int8_t > ( ( COM. pkg_e . D3 ) >> 8 ) ;
    COM. pkg_e . chs += static_cast < int8_t > ( ( COM. pkg_e . D3 ) & 0
        x00ff ) ;
    COM. pkg_e . tail = '\n' ;

    Debug ( " HABILITAR - MOTORES " ) ;
    COM. escritura ( & Serial , 2 ) ;
}

```

```

void MainWindow :: on_Disable_clicked ( )

```

```

{
    COM. pkg_e . hdr = ' : ' ;
    COM. pkg_e . CMD = DISABLE ;
    COM. pkg_e . D1 = 0.0 ;
    COM. pkg_e . D2 = 0.0 ;
    COM. pkg_e . D3 = 0.0 ;
    COM. pkg_e . chs = 0 ;
    COM. pkg_e . chs += static_cast < int8_t > ( COM. pkg_e . CMD ) ;
    COM. pkg_e . chs += static_cast < int8_t > ( ( COM. pkg_e . D1 ) >> 8 ) ;
    COM. pkg_e . chs += static_cast < int8_t > ( ( COM. pkg_e . D1 ) & 0
        x00ff ) ;
    COM. pkg_e . chs += static_cast < int8_t > ( ( COM. pkg_e . D2 ) >> 8 ) ;
    COM. pkg_e . chs += static_cast < int8_t > ( ( COM. pkg_e . D2 ) & 0
        x00ff ) ;
    COM. pkg_e . chs += static_cast < int8_t > ( ( COM. pkg_e . D3 ) >> 8 ) ;
}

```

```

    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D3)& 0
        x00ff);
    COM.pkg_e.tail='\n';

    Debug("DESHABILITAR MOTORES");
    COM.escritura(&Serial,2);
}

void MainWindow::on_Stop_clicked()
{
    COM.pkg_e.hdr=': ';
    COM.pkg_e.CMD=STOP;
    COM.pkg_e.D1=0.0;
    COM.pkg_e.D2=0.0;
    COM.pkg_e.D3=0.0;
    COM.pkg_e.chs=0;
    COM.pkg_e.chs+=static_cast<int8_t>(COM.pkg_e.CMD);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D1)>>8);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D1)& 0
        x00ff);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D2)>>8);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D2)& 0
        x00ff);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D3)>>8);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D3)& 0
        x00ff);
    COM.pkg_e.tail='\n';

    Debug("STOP");
    COM.escritura(&Serial,2);
}

void MainWindow::on_Home_clicked()
{
    COM.pkg_e.hdr=': ';
    COM.pkg_e.CMD=HOME;
    COM.pkg_e.D1=0.0;
    COM.pkg_e.D2=0.0;
    COM.pkg_e.D3=0.0;

```

```
    COM.pkg_e.chs=0;
    COM.pkg_e.chs+=static_cast<int8_t>(COM.pkg_e.CMD);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D1)>>8);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D1)& 0
        x00ff);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D2)>>8);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D2)& 0
        x00ff);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D3)>>8);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D3)& 0
        x00ff);
    COM.pkg_e.tail='\n';

    Debug("HOME");
    COM.escritura(&Serial,2);
}
```

```
void MainWindow::on_Rutina_clicked()
```

```
{
    COM.pkg_e.hdr=': ';
    COM.pkg_e.CMD=RUTINA;
    COM.pkg_e.D1=0.0;
    COM.pkg_e.D2=0.0;
    COM.pkg_e.D3=0.0;
    COM.pkg_e.chs=0;
    COM.pkg_e.chs+=static_cast<int8_t>(COM.pkg_e.CMD);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D1)>>8);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D1)& 0
        x00ff);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D2)>>8);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D2)& 0
        x00ff);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D3)>>8);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D3)& 0
        x00ff);
    COM.pkg_e.tail='\n';

    Debug("RUTINA");
    COM.escritura(&Serial,2);
}
```

```

}

void MainWindow::on_Terminar_clicked()
{
    Serial.close();
    SerialState=OFF;
    ui->Iniciar->setEnabled(1);
    habilitar_pantalla(false);
}

void MainWindow::on_leer_clicked()
{
    if(ui->leer->text()=="TERMINAR"){
        ui->leer->setText("LEER");
        ui->leer->setStyleSheet(style_start); //Verde

        COM.mutex_apagado.lock();
        COM.exit=true;
        Debug("Exit forzoso/n");
        COM.mutex_apagado.unlock();
        Serial_open(name_COM, 115200);
        return;
    }
    else{
        ui->leer->setText("TERMINAR");
        ui->leer->setStyleSheet(style_stop); //Rojo

        Debug(QDateTime::currentDateTime().
            toString(Qt::DateFormat::ISODate));

        QString aux=(ui->log_datos->toPlainText());
        std::string nombre_log=aux.toUtf8().constData();

        double t_total=ui->Tiempo_datos->value();
        DT=ui->tiempo_lect->value();
    }
}

```

```
        Serial.close();

        COM.reescribir(t_total,DT,nombre_log,"COM3", LECTURA
            ,0); //cambiar COM segun donde se conecta

        ui->leer->setEnabled(1);

        COM.start(QThread::TimeCriticalPriority);
    }

}

void MainWindow::recieve(float porcentaje){
    ui->avance->setValue(porcentaje*100);
}

void MainWindow::experiment_finished(void){
    ui->leer->setText("LEER");
    ui->leer->setStyleSheet(style_start); //Verde.
    ui->avance->setValue(0);
    Serial_open(name_COM, 115200);
}

void MainWindow::Debug(QString data){
    ui->DebugLine->append(data);
}

void MainWindow::on_new_variables_clicked()
{
    uint16_t motor=static_cast<uint16_t>(ui->motor_change->
        value());
    uint16_t periodo=static_cast<uint16_t>(ui->per_rutina->
        value()); //nuevo periodo en s

    COM.pkg_e.hdr=': ';
    COM.pkg_e.CMD='P';
    COM.pkg_e.D1=motor;
    COM.pkg_e.D2=periodo*10; //paso el periodo en s,
```

```

    dividir por 10 luego
    COM. pkg_e.D3=0.0;
    COM. pkg_e.chs=0;
    COM. pkg_e.chs+=static_cast<int8_t>(COM. pkg_e.CMD) ;
    COM. pkg_e.chs+=static_cast<int8_t>((COM. pkg_e.D1)>>8);
    COM. pkg_e.chs+=static_cast<int8_t>((COM. pkg_e.D1)& 0
        x00ff) ;
    COM. pkg_e.chs+=static_cast<int8_t>((COM. pkg_e.D2)>>8);
    COM. pkg_e.chs+=static_cast<int8_t>((COM. pkg_e.D2)& 0
        x00ff) ;
    COM. pkg_e.chs+=static_cast<int8_t>((COM. pkg_e.D3)>>8);
    COM. pkg_e.chs+=static_cast<int8_t>((COM. pkg_e.D3)& 0
        x00ff) ;
    COM. pkg_e.tail='\n' ;

    COM. escritura(&Serial ,2) ;

    Debug(" Cambio periodo de rutina" ) ;
}

void MainWindow::on_new_variables2_clicked ()
{
    uint16_t motor=static_cast<uint16_t>(ui->motor_change->
        value ()); //1,2,3
    uint16_t longitud=static_cast<uint16_t>(ui->long_rutina
        ->value ()); //en mm

    COM. pkg_e.hdr=': ' ;
    COM. pkg_e.CMD='L' ;
    COM. pkg_e.D1=motor; //1,2,3
    COM. pkg_e.D2=longitud*10; //mm
    COM. pkg_e.D3=0.0;
    COM. pkg_e.chs=0;
    COM. pkg_e.chs+=static_cast<int8_t>(COM. pkg_e.CMD) ;
    COM. pkg_e.chs+=static_cast<int8_t>((COM. pkg_e.D1)>>8);
    COM. pkg_e.chs+=static_cast<int8_t>((COM. pkg_e.D1)& 0
        x00ff) ;
    COM. pkg_e.chs+=static_cast<int8_t>((COM. pkg_e.D2)>>8);
    COM. pkg_e.chs+=static_cast<int8_t>((COM. pkg_e.D2)& 0

```

```

        x00ff);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D3)>>8);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D3)& 0
        x00ff);
    COM.pkg_e.tail='\n';

    COM.escritura(&Serial,2);

    Debug("Cambio longitud de rutina");
}
void MainWindow::on_new_variables3_clicked()
{
    uint16_t motor=static_cast<uint16_t>(ui->motor_change->
        value());
    uint16_t acel=static_cast<uint16_t>(ui->acel_rutina->
        value()); //en % del periodo

    COM.pkg_e.hdr=': ';
    COM.pkg_e.CMD='A';
    COM.pkg_e.D1=motor;
    COM.pkg_e.D2=acel*100; //mas resolucion
    COM.pkg_e.D3=0.0;
    COM.pkg_e.chs=0;
    COM.pkg_e.chs+=static_cast<int8_t>(COM.pkg_e.CMD);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D1)>>8);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D1)& 0
        x00ff);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D2)>>8);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D2)& 0
        x00ff);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D3)>>8);
    COM.pkg_e.chs+=static_cast<int8_t>((COM.pkg_e.D3)& 0
        x00ff);
    COM.pkg_e.tail='\n';

    COM.escritura(&Serial,2);

    Debug("Cambio aceleracion de rutina");
}

```


Apéndice G

Código utilizado para el análisis de los datos experimentales

En este apéndice se expone el código utilizado para el análisis de los datos experimentales medidos. Este código se lo redactó en el programa Octave. A continuación se expone el código mencionado:

```
clc
clear all
close all

pkg load control;
pkg load signal;
graphics_toolkit fltk;

%ok, con golpe al medio de los 5s 5ms
##dt=0.005;
##dat=load("datos_solo.txt");

##%ok, con golpe al medio de los 5s 5ms
##dt=0.005;
##dat=load("datos_solo_golpe.txt");

%ok, con golpe al medio de los 5s 2ms
##dat=load("datos_solo_golpe2.txt");
##dt=0.002;

%ok, con golpe al medio de los 5s 2ms
##dat=load("datos_quieto.txt");
```

```
##dt=0.002;
%mean_hor=490.55;

%%ok, con golpe al medio de los 5s 2ms
##dat=load("datos_quieto_vertical.txt");
##dt=0.002;
##mean_vert=590.04;
endtime=5;

#ok, con golpe al medio de los 5s 2ms
dat=load("datos_solo_colimador_2.txt");
dt=0.002;
endtime=2;
##MaxAmpliRMS = 4.3128e-04
##MaxAmpliABS = 1.3779e-03
##MaxAmpliABS_A = 2.3049
##MaxAmpliRMS_A = 0.7160

##dat=load("datos_colimador_actuador_1.txt");
##dt=0.002;
##endtime=5;
##MaxAmpliRMS = 3.0017e-04
##MaxAmpliABS = 7.5866e-04
##MaxAmpliABS_A = 2.2667
##MaxAmpliRMS_A = 0.6255

##
##dat=load("datos_colimador11_actuador_25.txt");
##dt=0.002;
##endtime=5;
##MaxAmpliRMS = 3.4880e-04
##MaxAmpliABS = 8.4379e-04

##dat=load("datos_colimador_solo_11.txt");
##dt=0.002;
##endtime=5;
####MaxAmpliRMS = 4.4563e-04
```

```

#####MaxAmpliABS = 1.0520e-03

Numero=dat (: ,1) ;
t_id=dat (: ,2) ;
t=0:dt:endtime-dt ;
t=t' ;
Ay=dat (: ,3) ;
Ay_crudo=Ay ;

##figure
##plot (t ,Ay)

Ay=Ay-mean(Ay) ;
Vmotor_y=dat (: ,4) ;

g_int=tf ([1] , [1 0]) ;
g_int_z=c2d (g_int , dt) ;

%%filter sections_CC

f_min=3;
f_max=50;

f_min=f_min*2*pi ;
f_max=f_max*2*pi ;
s = tf ('s') ;
r=s/f_max ;
G_BUT_PB=1/(r^2+sqrt (2) *r+1) ;

j=f_min/s ;
G_BUT_PA=1/(j^2+sqrt (2) *j+1) ;
G_BUT_PA=minreal (G_BUT_PA) ;
G_int=1/s ;
GPAz=c2d (G_BUT_PA, dt , 'zoh') ;
GPBz=c2d (G_BUT_PB, dt , 'zoh') ;
Gz=GPAz*GPBz ;

Ay=Ay-mean(Ay) ;
Ay*=9.8/100 ;

```

```

Ay_f=lsim (GPBz,Ay) ;

Vy=lsim ( g_int_z*GPAz, Ay_f) ;
Py=lsim ( g_int_z ,Vy) ;

MaxAmpliRMS=rms (Py)
MaxAmpliABS=(max(Py)-min(Py))/2
MaxAmpliABS_A=(max(Ay_f)-min(Ay_f))/2
MaxAmpliRMS_A=rms ( Ay_f)
##
##figure
##plot(t,[Ay Ay_f]);
####title('Ay Ayf');
####legend('Ay','Ayf');
##title('$A_y$ y $A_yf$',"fontsize",10)
##ylabel('Aceleración [ $\frac{m}{s^2}$ ]',"fontsize",10)
##xlabel('tiempo [s]',"fontsize",10)
##oj=legend('$A_y$', '$A_yf$');
##set(gca,"linewidth",1,"fontsize",10)
##set(oj,"fontsize",10)
####xlim([f1 f2])
##
##print -depslatexstandalone fig_AyAyf
##system("pdflatex fig_AyAyf");
##delete("fig_AyAyf-inc-eps-converted-to.pdf")
##delete("fig_AyAyf.aux")
##delete("fig_AyAyf.tex")
##delete("fig_AyAyf-inc.eps")
##delete("fig_AyAyf.log")
##
####figure
####plot(t(650:1010),[Ay(650:1010) Ay_f(650:1010)]);
#####title('Ay Ayf');
#####legend('Ay','Ayf');
####title('$A_y$ y $A_yf$',"fontsize",10)
####ylabel('Aceleración [ $\frac{m}{s^2}$ ]',"fontsize",10)
####xlabel('tiempo [s]',"fontsize",10)
####oj=legend('$A_y$', '$A_yf$');

```

```

#####set(gca, "linewidth", 1, "fontsize", 10)
#####set(oj, "fontsize", 10)
#####xlim ([f1 f2])
#####
#####print -depslatexstandalone fig_Ay_Ayf_zoom
#####system ("pdflatex fig_Ay_Ayf_zoom");
#####delete ("fig_Ay_Ayf_zoom-inc-eps-converted-to.pdf")
#####delete ("fig_Ay_Ayf_zoom.aux")
#####delete ("fig_Ay_Ayf_zoom.tex")
#####delete ("fig_Ay_Ayf_zoom-inc.eps")
#####delete ("fig_Ay_Ayf_zoom.log")
##
##
#####figure
#####plot(t(500:1010), Ay_f(500:1010));
##
##
#####t_exp=t(544:1650)-1.086;
#####Ay_f_exp=Ay_f(544:1650);
#####
#####figure
#####plot(t_exp,[Ay_f_exp 5*exp(-(0.1*10.5*6.28*t_exp))]);
#####title('Ay e^{0.05t}');
#####legend('Ay', 'e^{(0.05t)}');
#####
#####title('$A_{yf}$ y $5e^{\{-0.05*10.5*t\}}$', "fontsize", 10)
#####ylabel('Aceleracion [ $\frac{m}{s^2}$ ]', "fontsize", 10)
#####xlabel('tiempo [s]', "fontsize", 10)
#####oj=legend('$A_{yf}$', '$5e^{\{-0.05*10.5*t\}}$');
#####set(gca, "linewidth", 1, "fontsize", 10)
#####set(oj, "fontsize", 10)
#####xlim ([f1 f2])
#####
#####print -depslatexstandalone fig_Ay_exp
#####system ("pdflatex fig_Ay_exp");
#####delete ("fig_Ay_exp-inc-eps-converted-to.pdf")
#####delete ("fig_Ay_exp.aux")
#####delete ("fig_Ay_exp.tex")
#####delete ("fig_Ay_exp-inc.eps")

```

```

#####delete ("fig_Ay_exp.log")
##
#####
#####figure
#####plot(t, Vmotor_y);
#####title('vel_mot')
#####
#####figure
#####plot(t(2:end), diff(t_id));
#####
#####figure
#####plot(t, Vy);
#####title('Vy')
#####
#####figure
#####plot(t, Py);
#####title('Py')
#####
##
##Ndatos=size(Ay,1);
##
##u=Ay;
##T=dt;
##L=Ndatos;
##Y=fft(u);
##P2=abs(Y/L);
##P1=P2(1:L/2+1);
##P1(2:end-1) = 2*P1(2:end-1);
##FS=1/T;
##f = FS/L*(0:(L/2));
##
#####
#####
#####
#####figure()
#####plot(f,P1);
#####title('Espectro de densidad de potencia en frecuencia
A');
#####ylabel('P1(f)');

```

```

#####xlabel('f (Hz) ');
####
#####title('Espectro de densidad de potencia en frecuencia ',
    fontsize",10)
#####ylabel('P1[f]',"fontsize",10)
#####xlabel('frecuencia [Hz]',"fontsize",10)
#####set(gca,"linewidth",1,"fontsize",10)
#####set(oj,"fontsize",10)
#####xlim([f1 f2])
####
#####print -depslatexstandalone fig_FFT
#####system("pdflatex fig_FFT");
#####delete("fig_FFT-inc-eps-converted-to.pdf")
#####delete("fig_FFT.aux")
#####delete("fig_FFT.tex")
#####delete("fig_FFT-inc.eps")
#####delete("fig_FFT.log")
##
####
####
#####figure
#####plot(t,[Ay_f Vy*100 Py*1000]);
#####grid on
#####title('Ay Vy Py')
#####legend('Ay-f','100 Vy','1000 Py')
#####xlabel('tiempo (s)')
#####ylabel('A [m/s^2], V*100[m/s], V*1000 [m]')
####
##
##figure
##plot(t, Py);
##grid on
#####title('Ay Vy Py')
#####legend('Ay-f','100 Vy','1000 Py')
#####xlabel('tiempo (s)')
#####ylabel('A [m/s^2], V*100[m/s], V*1000 [m]')
##
##title('Desplazamiento de la plataforma del banco de
    pruebas',"fontsize",10)

```

```
##ylabel('Posicion [m]',"fontsize",10)
##xlabel('tiempo [s]',"fontsize",10)
##set(gca,"linewidth",1,"fontsize",10)
##set(oj,"fontsize",10)
####xlim ([f1 f2])
##
##print -depslatexstandalone fig_pos
##system ("pdflatex fig_pos");
##delete ("fig_pos-inc-eps-converted-to.pdf")
##delete ("fig_pos.aux")
##delete ("fig_pos.tex")
##delete ("fig_pos-inc.eps")
##delete ("fig_pos.log")
```