UPN
UNIVERSIDAD PRIVADA DEL NORTE

FACULTAD DE INGENIERÍA

Carrera de INGENIERÍA MECATRÓNICA

“DESARROLLO DE UN SISTEMA ROBÓTICO PARA EL TRATAMIENTO FISIOTERAPÉUTICO DE MIEMBROS INFERIORES EN EL CENTRO DE REHABILITACIÓN FÍSICA HABILIS PERÚ SAC”

Tesis para optar al título profesional de:

Ingeniero Mecatrónico

Autor:

Antonio Paucar Quispe

Asesor:

Mg. Jorge Luis Contreras Cossio

https://orcid.org/0000-0001-7801-5833

Lima - Perú

2023

JURADO EVALUADOR

Jurado 1 Presidente(a)	**Néstor Bernardo Corpus Vergara**	**08467416**
	Nombre y Apellidos	N° DNI

Jurado 2	**Sergio Martinez Martinez**	**47559200**
	Nombre y Apellidos	N° DNI

Jurado 3	**Eliseo Juan Zarate Pérez**	**42415810**
	Nombre y Apellidos	Nº DNI

INFORME DE SIMILITUD

“Desarrollo de un sistema robótico para el tratamiento fisioterapéutico de miembros inferiores en el centro de rehabilitación física Habilis Perú SAC”

INFORME DE ORIGINALIDAD

17%	16%	3%	10%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	hdl.handle.net Fuente de Internet	1%
2	repositorio.unsa.edu.pe Fuente de Internet	1%
3	pdfcoffee.com Fuente de Internet	1%
4	qdoc.tips Fuente de Internet	<1%
5	idoc.pub Fuente de Internet	<1%
6	repositorio.upn.edu.pe Fuente de Internet	<1%
7	www.scribd.com Fuente de Internet	<1%
8	Submitted to Universidad Internacional de la Rioja Trabajo del estudiante	<1%

DEDICATORIA

A mis padres que me apoyan
en todo momento.

AGRADECIMIENTO

Agradezco a mi asesor y a toda la plana docente de Ingeniería Mecatrónica, por contribuir con mi aprendizaje y asesorarme durante mi formación universitaria.

Gracias a todos ellos.

Tabla de contenido

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ECUACIONES

RESUMEN

La presente investigación consiste en desarrollar un sistema robótico capaz de realizar ejercicios fisioterapéutico pasivos (también conocidos como ejercicios de rango de movimiento) en los miembros inferiores, durante las fases inicial e intermedia del proceso de rehabilitación. Se dividió el desarrollo de este sistema en 4 partes: control, electrónica, mecánica, y software (GUI).

La metodología que se siguió fue de enfoque cuantitativo, de tipo experimental y explicativo.

En los resultados de la investigación se presenta el desarrollo de las 4 partes que componen este sistema. Para la parte de control se diseñó un sistema de control de doble lazo, con apoyo del software Matlab. La parte electrónica se desarrolló mediante simulaciones en Proteus y KiCAD. La parte mecánica se desarrolló mediante cálculos y simulaciones en el software SolidWorks. Se desarrolló una interfaz gráfica en la que se pueden realizar las configuraciones y almacenar datos de los ejercicios fisioterapéuticos. De esta forma se consiguió cumplir con los objetivos específicos propuestos, es decir realizar los ejercicios fisioterapéuticos de flexión-extensión en rodilla y flexión plantar-flexión dorsal en tobillo de forma precisa, repetible y con la capacidad de almacenar datos.

Palabras clave: robótica, fisioterapia, rehabilitación, ingeniería de control, interfaz gráfica

CAPÍTULO I. INTRODUCCIÓN

Uno de los problemas de salud más graves que existe a nivel mundial, es sin duda la pérdida parcial o total de movilidad en miembros inferiores; esto debido a que afecta la movilidad de la persona afectada, lo que dificulta y retrasa su capacidad de reintegrarse en el mercado laboral y desarrollar de forma normal las actividades de la vida cotidiana. Al respecto Hurtado Floyd et al. (2012) señalan lo siguiente: "la persona con discapacidad presenta restricciones para involucrarse en situaciones vitales, debido a tres tipos de factores: personales, del entorno inmediato y del ámbito social, económico, político y físico" (pág. 229)

Por otro lado, el estado de salud de los pacientes afectados podría empeorar con el paso del tiempo, ya que generalmente adquieren otros males.

> Las personas con discapacidades con frecuencia tienen un mayor riesgo de presentar problemas de salud que se pueden prevenir. Como consecuencia de un tipo específico de discapacidad, pueden presentarse otras afecciones físicas o mentales.
>
> A algunas de estas otras afecciones también se las llama afecciones secundarias y pueden incluir lo siguiente: problemas urinarios e intestinales, fatiga, lesiones, problemas de salud mental y depresión, sobrepeso y obesidad, dolor, llagas o ulceras por presión. (Centro para el Control y la Prevención de Enfermedades, 2020, pág. 1)

Es decir, esta condición perjudica gravemente en la calidad de vida del paciente afectado.

Asimismo, los centros fisioterapéuticos se dedican a la atención y rehabilitación de pacientes con pérdida de movilidad en miembros superiores e inferiores. La cantidad

de pacientes que se atienden en un centro fisioterapéutico para estos casos generalmente es grande, ya que las causas que originan pérdida de movilidad son varias: enfermedades a los huesos, lesiones en los ligamentos, distrofias musculares, accidentes cerebrovasculares (ACV), entre otras. Del mismo modo, los tratamientos son de larga duración y requieren de varias sesiones. Todo esto limita la capacidad para atender personas, ya que muchas veces los especialistas no pueden dar abasto para atender la alta cantidad de pacientes.

Por otro lado, el desarrollo de sistemas robóticos en los últimos años se ha extendido a varios ámbitos relacionados con las necesidades y servicios de primera necesidad para los humanos, entre ellos los servicios domésticos, la medicina y la rehabilitación mediante fisioterapia.

> La rehabilitación robótica se presenta como uno de los campos de mayor interés en la actualidad, sirviendo de asistencia al trabajo arduo de los fisioterapeutas, además de que logra una mejor coordinación para los ejercicios de rehabilitación y mayor precisión en el diagnóstico de lesiones. (Araujo, Díaz, & Mata, 2017, pág. 16)

1.1. Formulación del problema

1.1.1. Problema general

- ¿De qué manera el desarrollo de un sistema robótico permitirá el tratamiento fisioterapéutico de miembros inferiores en el centro de rehabilitación física Habilis Perú SAC?

1.1.2 Problemas específicos

- ¿De qué manera el desarrollo de un sistema robótico permitirá realizar con precisión los ejercicios del tratamiento fisioterapéutico de miembros inferiores en el centro de rehabilitación física Habilis Perú SAC?
- ¿De qué manera el desarrollo de un sistema robótico permitirá repetir los ejercicios del tratamiento fisioterapéutico de miembros inferiores en el centro de rehabilitación física Habilis Perú SAC?
- ¿De qué manera el desarrollo de un sistema robótico permitirá registrar datos del tratamiento fisioterapéutico de miembros inferiores en el centro de rehabilitación física Habilis Perú SAC?

1.2. Marco teórico

1.2.1 Desarrollo de un sistema robótico

> Un robot es una máquina automática o autónoma que posee cierto grado de inteligencia, capaz de percibir su entorno y de imitar determinados comportamientos del ser humano. Los robots se utilizan para desempeñar labores riesgosas o que requieren de una fuerza, velocidad o precisión que está fuera de nuestro alcance. (Riccillo, 2012, pág. 8)

El presente sistema robótico se diseñó para realizar los ejercicios fisioterapéuticos de flexión-extensión en rodilla y flexión plantar-flexión dorsal en tobillo, de forma repetible y precisa, para las personas que requieren rehabilitación en los miembros inferiores.

Asimismo, este sistema robótico se diseñó con 2 grados de libertad (en adelante abreviado como GDL), un GDL le corresponde a la articulación de la rodilla y el otro GDL a la articulación del tobillo. Se realizó el diseño considerando que el paciente pueda

realizar los ejercicios en posición de sentado. Es decir, se trata de una máquina de tipo estacionario.

1.2.2 Tratamiento fisioterapéutico de miembros inferiores

Mediante el presente sistema robótico se realizarán de forma automática los ejercicios de flexión- extensión en la rodilla y de flexión dorsal-flexión plantar en el tobillo.

Figura 1

Posiciones de las articulaciones de rodilla y tobillo

Nota. La imagen muestra las posiciones de Flexión en tobillo + flexión plantar en tobillo (izquierda) y extensión en tobillo + flexión dorsal en tobillo (derecha). Tomado de https://riberasalud.com/povisa/rodilla/

Además, este sistema está diseñado para realizar estos ejercicios fisioterapéuticos de forma pasiva, es decir cuando el paciente en rehabilitación aún no es capaz de contraer los músculos por sí mismo.

Estos ejercicios consisten en el movimiento articular sin contracciones musculares del paciente. Todo el movimiento lo realiza el fisioterapeuta o el médico. El propósito de este ejercicio es mantener o incrementar el movimiento articular disponible. Estos ejercicios están indicados cuando la contracción muscular voluntaria es imposible, indeseable, o no lo suficientemente fuerte como para

vencer la contractura capsular. (Hoppenfeld & Murthy, 2004, pág. 21)

Al realizar estos ejercicios mediante el sistema robótico se trabajarán los movimientos rotacionales de la rodilla y el tobillo, y de esta manera se recuperará progresivamente la fortaleza de músculos y tendones. Ya que, el movimiento de esas articulaciones incentivará la interacción activa y pasiva del sistema locomotor, al ejercitarse mediante el funcionamiento combinado de músculos, tendones, articulaciones y huesos. (Hüter-Becker, Schewe, & Heipertz, 2003, pág. 95)

Partes del miembro inferior

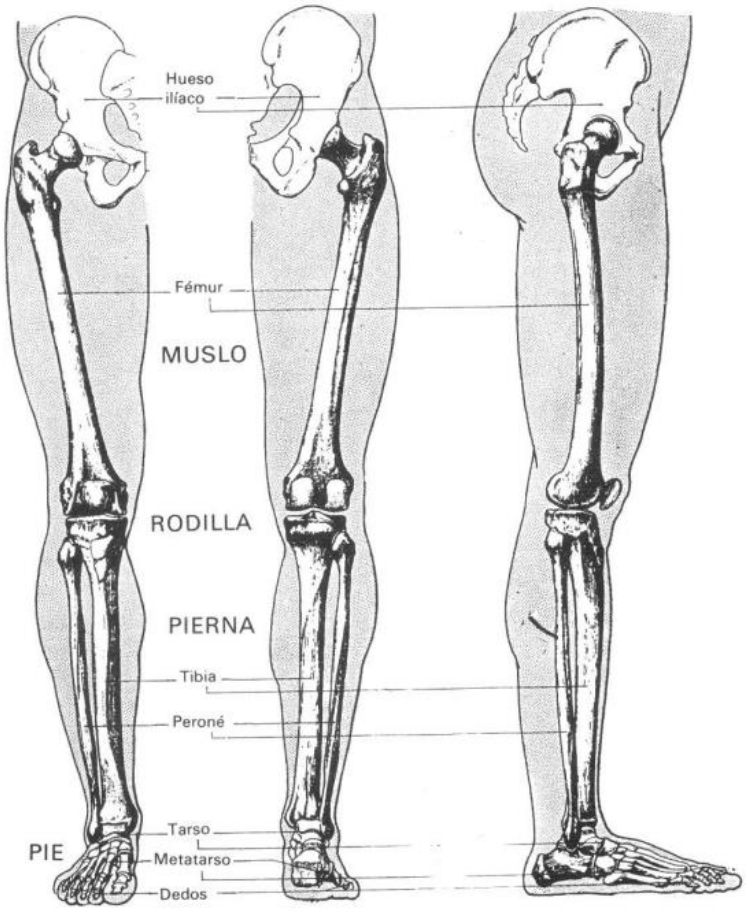
El miembro inferior puede ser dividido en dos partes: la pelvis o cintura pelviana, y la extremidad libre (compuesta por el muslo, la pierna y el pie). Al respecto González (s.f.) detalla:

> La finalidad del miembro inferior es servir de base de sustentación al cuerpo (estática) y permitir su marcha (locomoción). Se distingue en él una raíz o cintura pelviana y una extremidad libre subdividida en muslo, pierna y pie. La cintura pelviana está sólidamente articulada a la columna vertebral. Los huesos de esta cintura (coxales), junto con el extremo caudal de la columna vertebral (sacro-coxis) forman la pelvis o bacinete, armazón poco flexible que actúa de plataforma en el movimiento. (pág.2)

El presente sistema robótico se diseñó para ejercitar la extremidad libre en rodilla y tobillo, mediante actuadores (motores) accionados por un sistema de control. En la siguiente figura se puede observar las partes de la pierna.

Figura 2

Partes del miembro inferior



Nota. Tomado de Horcajada (s.f.)

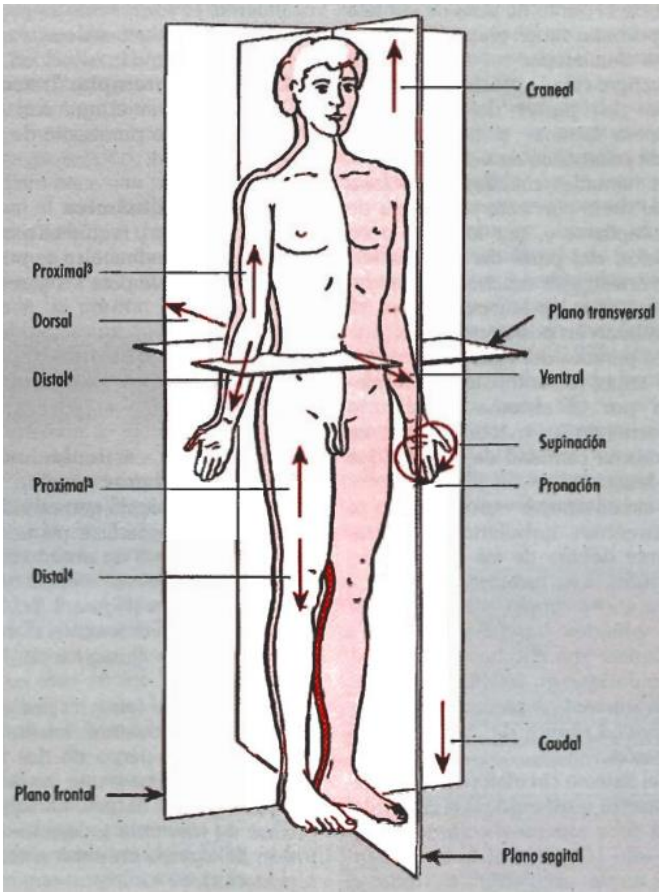
Rangos de movilidad

Para el diseño de este sistema robótico se tomaron en cuenta los movimientos de las articulaciones de rodilla y tobillo. Se tomó como plano de referencia al plano sagital, al respecto Hüter-Becker, Schewe, & Heipertz (2003) señalan: “el plano sagital (anterior-posterior) discurre de delante hacia atrás (división del cuerpo en dos mitades, una derecha y una izquierda)”(pág.27).

En la siguiente figura se muestra al plano sagital y otros planos (imaginarios) que dividen el cuerpo humano.

Figura 3

Planos anatómicos



Nota. Tomado de Hüter-Becker, Schewe, & Heipertz (2003)

- **Rango de movilidad de la rodilla**

La rodilla tiene 2 movimientos en el plano sagital, denominados como flexión y extensión. El rango de movilidad puede ser completo o funcional (lo necesario y suficiente para realizar una tarea determinada).

El rango de movilidad funcional es el movimiento que requiere una articulación específica para la realización de actividades de la vida diaria o para cualquier tarea específica del paciente (p. ej., lanzar una pelota). Para sentarse confortablemente, por ejemplo, son necesarios 90° de flexión de la rodilla. Un rango de movimiento de la rodilla desde extensión completa 0° a 90° de flexión no es completo, pero es

funcional. (Hoppenfeld & Murthy, 2004, pág. 20)

Figura 4

Rango de movimiento de la articulación de la rodilla

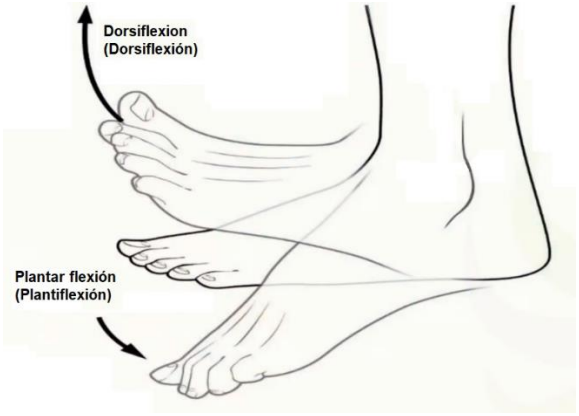
Nota. Tomado de Artur (2022)

- **Rango de movilidad del tobillo**

El tobillo tiene 2 movimientos en el plano sagital, denominados como dorsiflexión o flexión dorsal (movimiento del pie hacia arriba) y plantiflexión o flexión plantar (movimiento del pie hacia abajo).

Figura 5

Movimientos de la articulación del tobillo



Nota. Tomado de https://www.fisioterapia-online.com/glosario/flexion-plantar-o-plantiflexion

La movilidad del tobillo se desarrolla en un rango aproximado de 30-35°.

> Durante el primer rocker de la marcha, en el momento del apoyo inicial del talón, el tobillo se encuentra en posición neutra para realizar, poco después, una flexión plantar pasiva. Cuando el antepié contacta con el suelo, la rotación de avance de la tibia deja de tomar como referencia el talón para centrarse sobre el astrágalo, experimentando el tobillo un movimiento de flexión dorsal pasiva. El segundo rocker, o rocker del tobillo, se corresponde con el período de apoyo intermedio durante el cual el pie tiene una disposición plantígrada respecto del suelo. Cuando el talón despega se inicia el tercer rocker, en el que el tobillo realiza una dorsiflexión de unos 10-15º, seguida de una flexión plantar rápida de unos 10-20º. El arco de flexoextensión del tobillo durante la marcha normal se estima en unos 30-35º. (Monteagudo, Martínez, Maceira, & Borja, 2016, pág. 11)

1.3. Antecedentes

1.3.1 Antecedentes nacionales

(Chau Delgado, 2018), tuvo como objetivo realizar el diseño de una interfaz cerebro-máquina que, mediante procesamiento de señales cerebrales y aprendizaje de máquina, permite a un usuario seleccionar diversas tareas predefinidas para un manipulador robótico asistencial aprovechando el potencial relacionado a eventos conocido como "P300". La metodología que usó fue de enfoque cuantitativa, aplicativa y de diseño experimental. En sus resultados y conclusiones, presenta datos de las pruebas realizadas en ocho usuarios, utilizando 3 repeticiones por ensayo para el entrenamiento de un brazo robótico; además señala una eficiencia

de 90.17% en la etapa de entrenamiento y 82.62% en la etapa de validación, lo que es suficiente para demostrar que su sistema es viable.

(Alburqueque & Rondón, 2019)**,** tuvo como objetivo realizar el diseño e implementación de un exoesqueleto de fisioterapia para pacientes con artrosis de rodilla de la Clínica Geriátrica Militar de Chorrillos. Su trabajo se realizó bajo la metodología de enfoque cuantitativa, aplicativa y fue de diseño experimental. En sus resultados y conclusiones, consigue que su diseño ejerza una presión adecuada sobre determinados ligamentos, músculos y tendones, lo que evita inflamaciones y detiene la evolución de la artrosis.

(Mendoza Flores, 2021), tuvo como objetivo realizar el diseño y fabricación de un dispositivo robótico para la rehabilitación de extremidades inferiores en recién nacidos con mielomeningocele. Su trabajo se realizó bajo la metodología de enfoque cuantitativa, aplicativa y fue de diseño experimental. En sus resultados y conclusiones, consigue un prototipo funcional que realiza la flexión y extensión en la articulación de la rodilla utilizando una mini bomba de vacío como actuador; una limitación de dicho sistema es que no mide la contracción a tiempo real de los músculos artificiales en el modelo de pierna, además, no cuenta con una retroalimentación de la presión interna del actuador.

1.3.2 Antecedentes internacionales

(Park, 2020), tuvo como objetivo diseñar 2 dispositivos portátiles para la rehabilitación del antebrazo y la mano con el menor peso posible, haciendo uso

de actuadores neumáticos que emulan la actividad muscular de los músculos del antebrazo y la mano. Realizó su investigación bajo la metodología de enfoque cuantitativo, aplicativo y de diseño experimental. En sus resultados y conclusiones, se encontraron parámetros para una fabricación confiable, y se espera que sus dispositivos sean utilizados como alternativa frente a los habituales dispositivos de materiales rígidos, que son más pesados y costosos.

(Quinga Acosta & Sarsoza García, 2021), tuvieron como objetivo rediseñar y mejorar un prototipo robótico para rehabilitación de los dedos de las manos en personas con daño cerebral adquirido (DCA), este sistema se controla mediante un sistema en lazo cerrado y se opera mediante un software HMI. Su trabajo se realizó bajo la metodología de enfoque aplicativo y fue de diseño experimental. En sus resultados y conclusiones evalúa el funcionamiento del equipo respecto a 3 parámetros: desplazamiento, velocidad y repeticiones, y desarrolla un HMI que guarda datos de las sesiones de rehabilitación en un archivo CSV para que posteriormente pueda ser analizado por un especialista en rehabilitación física.

(Aldás Mayorga & Molina Aguiar, 2021), tuvieron como objetivo desarrollar un sistema mecatrónico para pacientes con daño cerebral adquirido (DCA) que permita la asistencia en la rehabilitación de extremidades superiores. Su trabajo se realizó bajo la metodología de enfoque aplicativo y fue de diseño experimental. En sus resultados y conclusiones, obtienen un sistema desarrollado en Arduino y con HMI desarrollado en Unity; además consideran un ángulo máximo de 180° y

realizan pruebas de repetibilidad y obtiene una variación de ±2.033mm en el eje X y ±1.85mm en el eje Y.

1.4. Objetivos

1.4.1 Objetivo general

- Desarrollar un sistema robótico para el tratamiento fisioterapéutico de miembros inferiores en el centro de rehabilitación física Habilis Perú SAC.

1.4.2 Objetivos específicos

- Desarrollar un sistema robótico que permita realizar con precisión los ejercicios del tratamiento fisioterapéutico de miembros inferiores en el centro de rehabilitación física Habilis Perú SAC.
- Desarrollar un sistema robótico que permita repetir los ejercicios del tratamiento fisioterapéutico de miembros inferiores en el centro de rehabilitación física Habilis Perú SAC.
- Desarrollar un sistema robótico que permita registrar datos del tratamiento fisioterapéutico de miembros inferiores en el centro de rehabilitación física Habilis Perú SAC.

1.5. Hipótesis

1.5.1 Hipótesis general

- El desarrollo de un sistema robótico permitiría realizar el tratamiento fisioterapéutico de miembros inferiores en el centro de rehabilitación física Habilis Perú SAC.

1.5.2 Hipótesis específicas

- Desarrollar un sistema robótico permitiría realizar con precisión los ejercicios del

tratamiento fisioterapéutico de miembros inferiores en el centro de rehabilitación física Habilis Perú SAC.

- Desarrollar un sistema robótico permitiría repetir los ejercicios del tratamiento fisioterapéutico de miembros inferiores en el centro de rehabilitación física Habilis Perú SAC.
- Desarrollar un sistema robótico permitiría registrar datos del tratamiento fisioterapéutico de miembros inferiores en el centro de rehabilitación física Habilis Perú SAC.

1.6. Justificación

En la actualidad el tratamiento de rehabilitación de miembros inferiores se lleva a cabo con la asistencia de un especialista en fisioterapia, de forma manual. Sin embargo, esta forma de realizar el tratamiento tiene algunas limitaciones e inconvenientes.

> Las técnicas de rehabilitación actuales implican la asistencia de una persona para facilitar el movimiento de las extremidades, sin embargo, un mal cálculo en la fuerza a ser usada, la falta de homogeneidad en los movimientos, e incluso el medio en el que se hacen los ejercicios pueden no ser óptimos para la persona a tratar, además, la duración del tratamiento en algunos casos es tan larga que los pacientes (o las familias de los pacientes, quienes normalmente cuidan de ellos y se encargan de llevarlos a las terapias) desisten en el camino (algunas veces incluso antes de ver algún resultado). (Arrese A. , 2015, pág. 2)

Estos inconvenientes podrían ocasionar que la situación del paciente empeore, inclusive pueden llegar a originar nuevas lesiones.

Es por ello que en la presente tesis se diseñó un sistema robótico para realizar los ejercicios de rehabilitación de miembros inferiores, con la capacidad de realizar varias

repeticiones y de manera precisa.

Se decidió desarrollar el sistema robótico para el tratamiento de miembros inferiores, debido a que es la zona crítica que soporta el peso de las personas. Este sistema robótico permitiría automatizar los ejercicios de rehabilitación física de los pacientes, y de esta forma ayudaría a recuperar la capacidad locomotora de más personas.

Por otro lado, el aspecto innovador del presente sistema robótico es que permite al usuario elegir si se realizarán los ejercicios de rehabilitación en la rodilla, en el tobillo o en ambos de forma simultánea. Otro aspecto innovador es que cuenta con una interfaz gráfica que permite visualizar datos de los ejercicios y luego, mediante un botón, exportarlos a una hoja Excel donde se pueden guardar los datos para que luego lo pueda analizar el especialista en fisioterapia y rehabilitación.

CAPÍTULO II. METODOLOGÍA

2.1. Enfoque de la investigación

La presente investigación es de enfoque cuantitativo, ya que analiza la realidad de manera objetiva, tiene un objetivo y está orientado a obtener resultados.

> El enfoque cuantitativo es secuencial y probatorio… El orden es riguroso, aunque desde luego, podemos redefinir alguna fase. Parte de una idea que va acotándose y, una vez delimitada, se derivan objetivos y preguntas de investigación, se revisa la literatura y se construye un marco o una perspectiva teórica. Se miden las variables en un determinado contexto y se extrae una serie de conclusiones… La investigación cuantitativa debe ser lo más "objetiva" posible. Los fenómenos que se observan o miden no deben ser afectados por el investigador, quien debe evitar en lo posible que sus temores, creencias, deseos y tendencias influyan en los resultados del estudio o interfieran en los procesos y que tampoco sean alterados por las tendencias de otros (Hernández, Fernández, & Baptista, 2014, pág. 4)

2.1.1 Diseño de la investigación

La presente investigación es de diseño experimental, ya que requiere la manipulación intencional de una acción para analizar sus posibles resultados (Hernández, Fernández, & Baptista, 2014, pág. 129). Asimismo, es explicativo porque busca explicar los beneficios que tendrá usar este sistema robótico en el tratamiento fisioterapéutico de personas con lesiones en los miembros inferiores.

2.2. Población

La población en esta investigación está conformada por los pacientes del centro de rehabilitación física Habilis Perú SAC con lesiones en los miembros inferiores, aquellos en fase inicial e intermedia del proceso de rehabilitación.

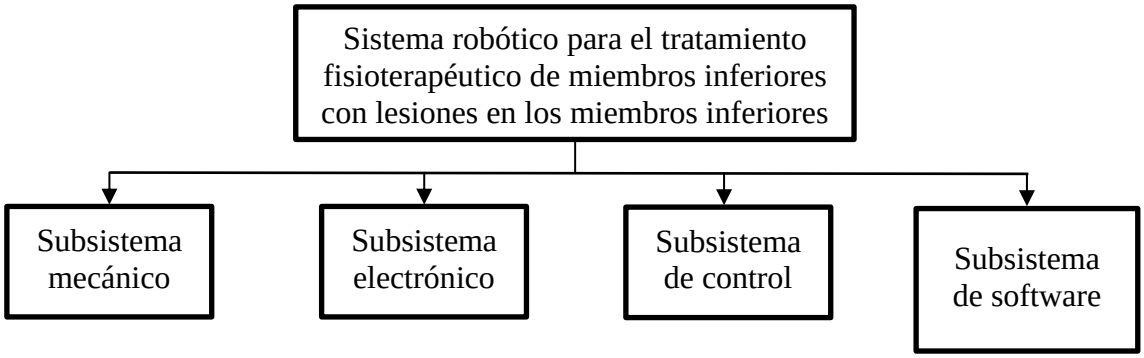
2.3. Métodos e instrumentos

2.3.1 Métodos

El presente sistema robótico está conformado por 4 subsistemas: mecánico, electrónico, de control y software.

Figura 6

Composición del sistema robótico



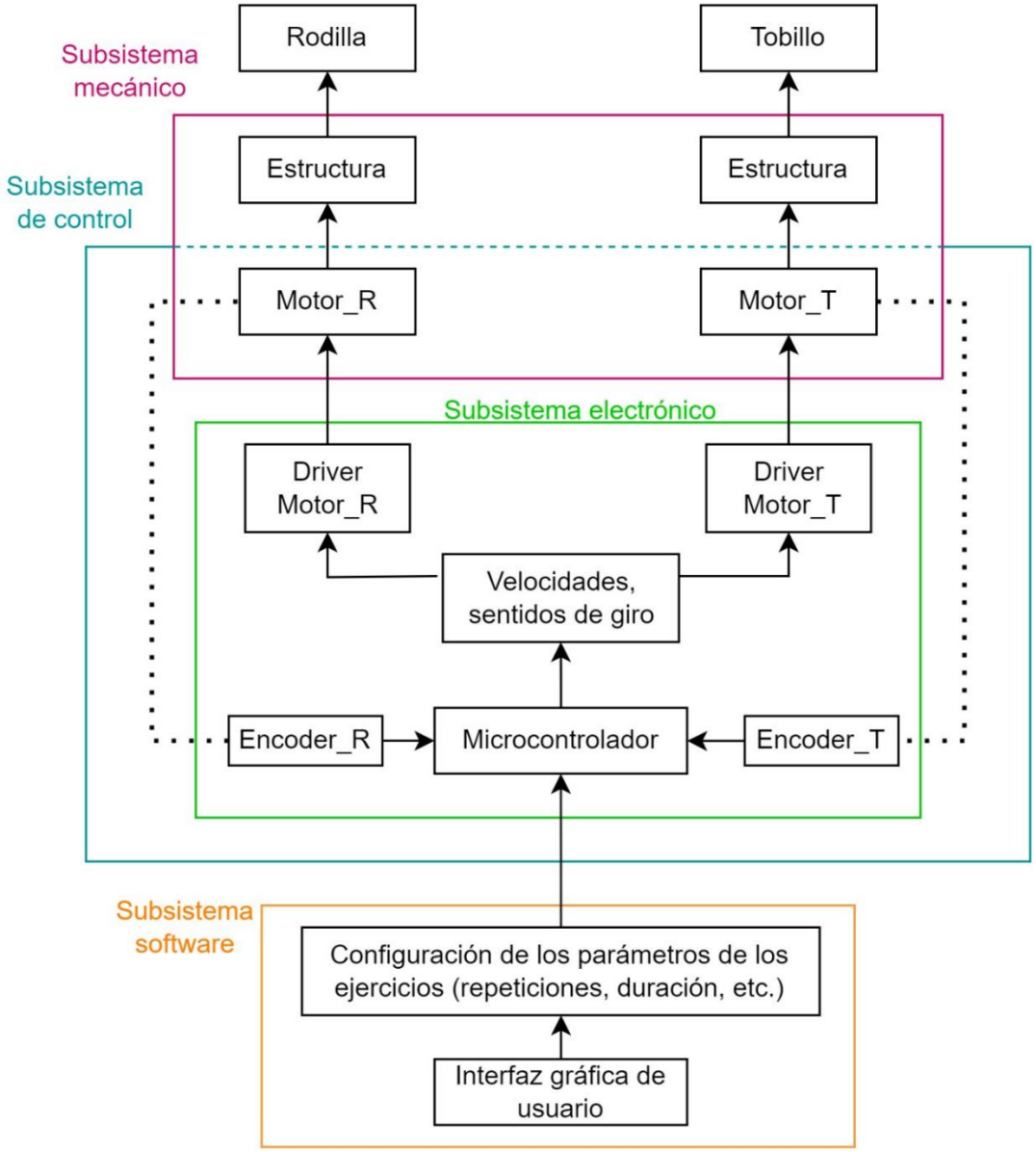
- El subsistema mecánico está conformado por la estructura metálica, los eslabones de la máquina que soportan el propio peso de la estructura y los motores.
- El subsistema electrónico está conformado por los circuitos que incluyen a la fuente conmutada, microcontrolador, encoders, drivers de los motores y otros dispositivos electrónicos.
- El subsistema de control consta del algoritmo que tiene como entrada de referencia a la trayectoria (posición en el tiempo). Este subsistema se encarga de que se realicen los ejercicios de flexión-extensión en rodilla y flexión plantar-flexión dorsal en tobillo.
- El subsistema de software consiste en la interfaz gráfica de usuario (GUI), la cual permite configurar los parámetros de los ejercicios fisioterapéuticos (duración, repeticiones, ID del paciente y selección de las articulaciones). También muestra gráficos de la performance del sistema y otros datos de cada sesión de ejercicios

realizada.

En la siguiente figura se ilustra el funcionamiento general del sistema robótico, donde se observa que los parámetros ingresados en la interfaz gráfica (subsistema software) pasan por los otros 3 subsistemas hasta actuar sobre las articulaciones de rodilla y tobillo para realizar los ejercicios fisioterapéuticos.

Figura 7

Funcionamiento general del sistema robótico



2.3.2 Instrumento

Para la realización de la presente tesis se usaron varios instrumentos de software y de hardware.

Para el diseño mecánico se usó el software Solidworks 2020. En este software se dibujaron las piezas y se hicieron los análisis de tensiones Von Misses para las piezas que soportan cargas. Por otro lado, el dimensionamiento de motores se realizó mediante la aplicación de las ecuaciones de dinámica inversa, y luego se corroboraron los valores hallados en la herramienta web “Motor Sizing Tool” de Orientalmotor (https://sizing.orientalmotor.co.jp/application/next/arm/).

Para el diseño del subsistema electrónico se usó el software KiCAD para realizar los esquemas, diagramas electrónicos y el diseño de la PCB. También se usó el software Proteus para realizar simulaciones del driver de motor.

El diseño del subsistema de control se realizó con ayuda del software Matlab. Se desarrolló la parte del lazo interno de forma física, para identificar el modelo matemático del sistema, con ayuda del Arduino Nano y Matlab. Posteriormente, también se implementó la parte del lazo cerrado para verificar que se siga la trayectoria deseada r(t), desarrollando una comunicación vía USB entre el Arduino y la interfaz desarrollada en Matlab App Designer.

El diseño del software (interfaz gráfica de usuario) se realizó mediante la herramienta App Designer de Matlab. En esta herramienta se realizó el diseño del panel gráfico y también la programación correspondiente para los botones, barras de entrada y barras de salida.

2.4. Aspectos éticos

En cuanto a lo que concierne a la originalidad, esta investigación fue elaborada en

su totalidad por el autor y no es copia de otros trabajos presentados anteriormente en ninguna institución.

CAPÍTULO III. RESULTADOS

3.1. Precisión en los ejercicios del tratamiento fisioterapéutico

Para realizar con precisión los ejercicios del tratamiento fisioterapéutico se diseñó un sistema de control retroalimentado. Este sistema tiene como propósito reducir el error, es decir la diferencia entre la entrada de referencia y la salida del sistema.

> Un sistema de control retroalimentado a menudo utiliza una función de una relación prescrita entre la salida y la entrada de referencia para controlar el proceso. A menudo, la diferencia entre la salida del proceso bajo control y la entrada de referencia se amplifica y se utiliza para controlar el proceso, de modo que la diferencia se reduzca continuamente.En general, la diferencia entre la salida deseada y la salida real es igual al error, que luego es ajustado por el controlador. La salida del controlador hace que el actuador module el proceso para reducir el error. (Dorf & Bishop, 2022)

Se realizaron el diseño y simulaciones mediante el software Matlab. Para dar mayor robustez, el sistema se diseñó con dos lazos de control: primario y secundario, o lazo interno y externo respectivamente.

Figura 8

Diagrama de bloques del sistema de control

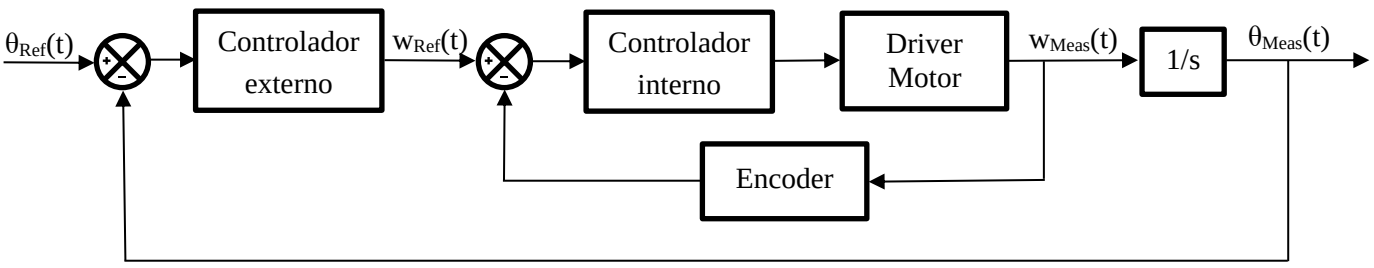


Tabla 1

Descripción de las variables de entrada y salida

Variable	Descripción	Unidades	Lazo de control
θ_{Ref} (t)	Posición angular deseada en el tiempo (ecuación de la trayectoria)	Grados sexagesimales	Lazo externo
θ_{Meas} (t)	Posición angular medida	Grados sexagesimales	Lazo externo
w_{Ref} (t)	Velocidad angular deseada	Grados sexagesimales por segundo	Lazo interno
w_{Meas} (t)	Velocidad angular medida	Grados sexagesimales por segundo	Lazo interno

3.1.1 Subsistema de control

3.1.1.1. Lazo interno

- **Identificación del modelo matemático del sistema**

Para realizar la identificación del modelo matemático del sistema se hizo uso de Arduino Nano, software Matlab y de la herramienta System Identification. La respuesta del sistema en lazo abierto para una entrada escalón se modeló como un sistema de primer orden con retardo.

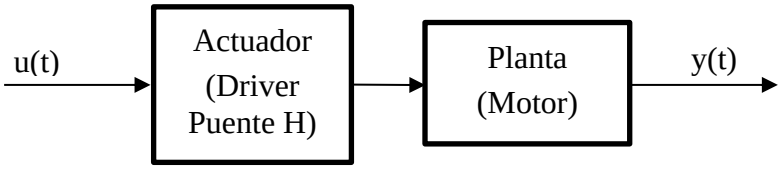
$$G(s) = \frac{K}{\tau s + 1} * e^{-L*s} \quad (1)$$

La programación para la identificación del sistema fue implementada en el Arduino Nano y en Matlab. Tiene como entrada u(t) a la señal de control en PWM (0-255), y como salida y(t) a velocidad angular en grados sexagesimales. Dentro de la programación del Arduino Nano se crearon funciones para normalizar ambas variables, ya que es necesario para poder hacer una correcta sintonización de los parámetros del

sistema (ecuación (1).

Figura 9

Diagrama de bloques de sistema en lazo abierto



También fue necesario implementar un filtro, ya que se observó que la señal proveniente del encoder era ruidosa. En este caso se implementó un filtro EMA (Exponential Moving Average) para atenuar el ruido.

Dada la secuencia de números reales: $x = x_1, x_2, x_3, \ldots$

Y la secuencia adicional: $\alpha = \alpha_1, \alpha_2, \alpha_3, \ldots$, con sus valores en el intervalo [0,1], generan la secuencia de números reales filtrada: $\delta = \delta_1, \delta_2, \delta_3, \ldots$

Mediante la ecuación:

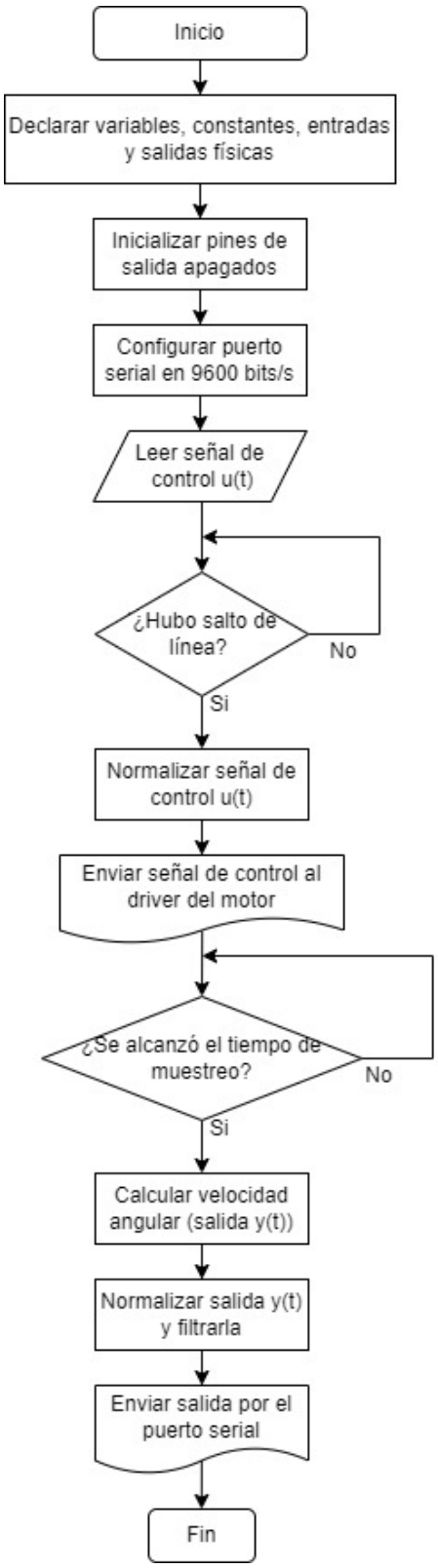
$$\delta_{n+1} = (1 - \alpha_n) * \delta_n + \alpha_n * x_{n+1}, \qquad n \geq 1 \tag{2}$$

(Klinker, 2011, pág. 98)

A continuación, se presenta el diagrama de flujo de la programación en Arduino para el sistema en lazo abierto. La programación se encuentra en el Anexo B: Programación.

Figura 10

Diagrama de flujo lazo abierto en Arduino



La programación en Matlab (Anexo B: Programación) se encarga de comunicarse

de forma serial con el Arduino, enviando la señal de control u(t) y recibiendo la señal de salida del sistema y(t), luego muestra los gráficos de las señales vs tiempo.

Figura 11

Diagrama de flujo lazo abierto Matlab

Las respuestas obtenidas para una entrada escalón al 10% se muestran a continuación.

Figura 12

Respuesta ante una estrada escalón para motor $_T$

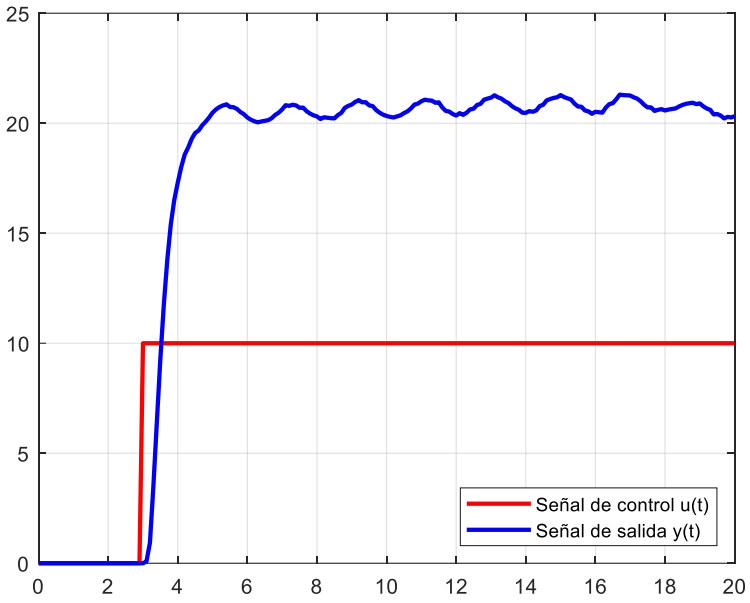
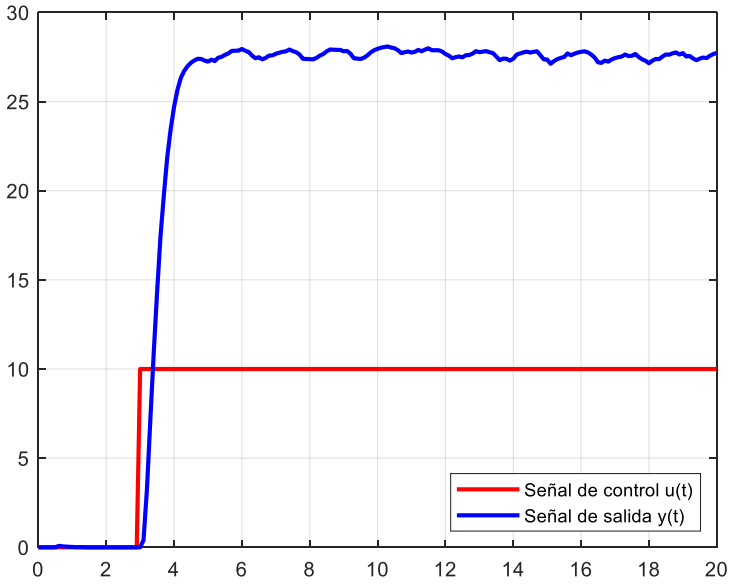


Figura 13

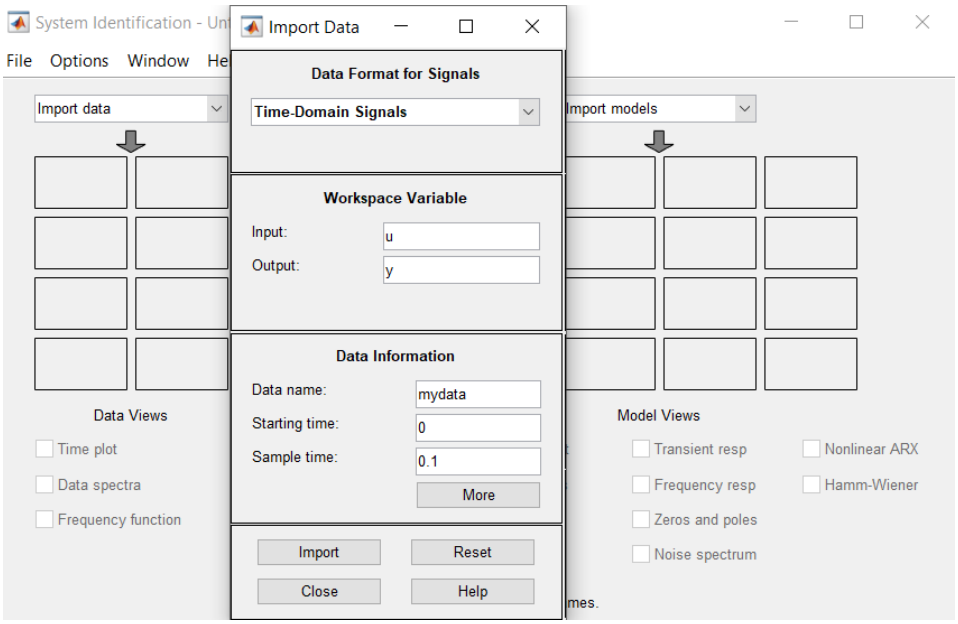
Respuesta ante una estrada escalón para motor $_R$



Luego se obtuvo los modelos dinámicos de ambos sistemas (rodilla y tobillo), mediante la herramienta System Identification de Matlab.

Figura 14

Configuración de las señales en System Identification



Mediante esta herramienta se obtuvo los parámetros indicados en la siguiente tabla:

Tabla 2

Parámetros obtenidos en la identificación de sistema

	Motor $_T$	Motor $_R$
K (Ganancia estática)	2.0695	2.7656
τ (Constante de tiempo)	0.4323	0.40595
L (Retardo)	0.2287	0.1709

Figura 15

Parámetros del modelo obtenido para Motor $_T$

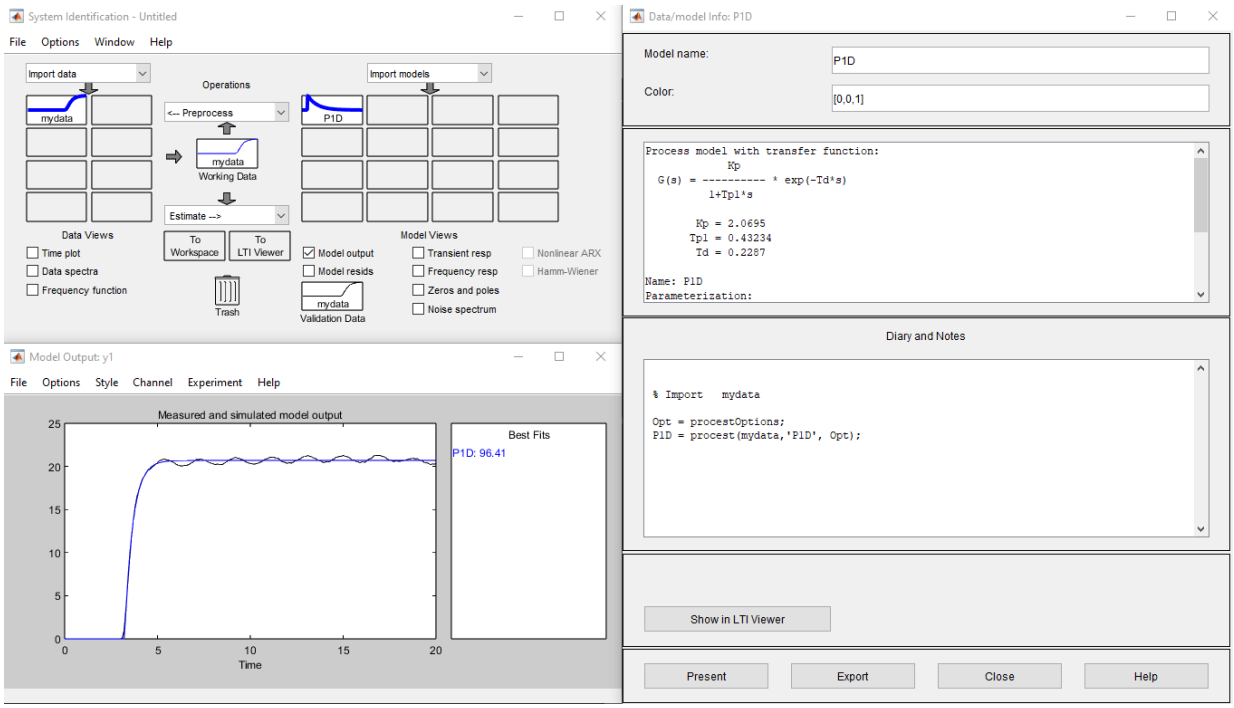
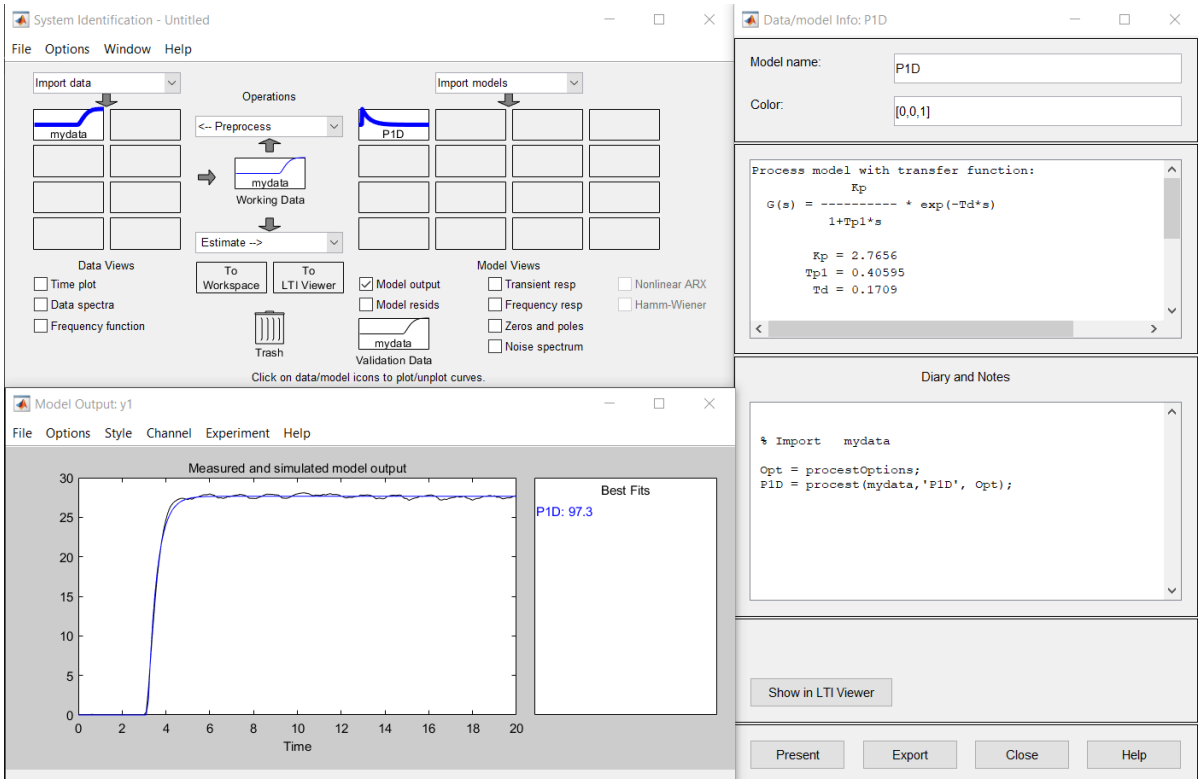


Figura 16

Parámetros del modelo obtenido para Motor $_R$



- **Selección de periodo de muestreo**

De acuerdo con la regla:

$$T_S \leq 0.1 * \tau: \tag{3}$$

Tenemos para motor $_T$:

$$T_S \leq 0.1 * 0.4323$$

$$T_S \leq 0.04323$$

Y para motor $_R$:

$$T_S \leq 0.1 * 0.4060$$

$$T_S \leq 0.04060$$

Para que el tiempo de muestreo sea múltiplo entero de 1000 ms (1 segundo), se seleccionó el valor de muestreo de 25 ms (0.025 s).

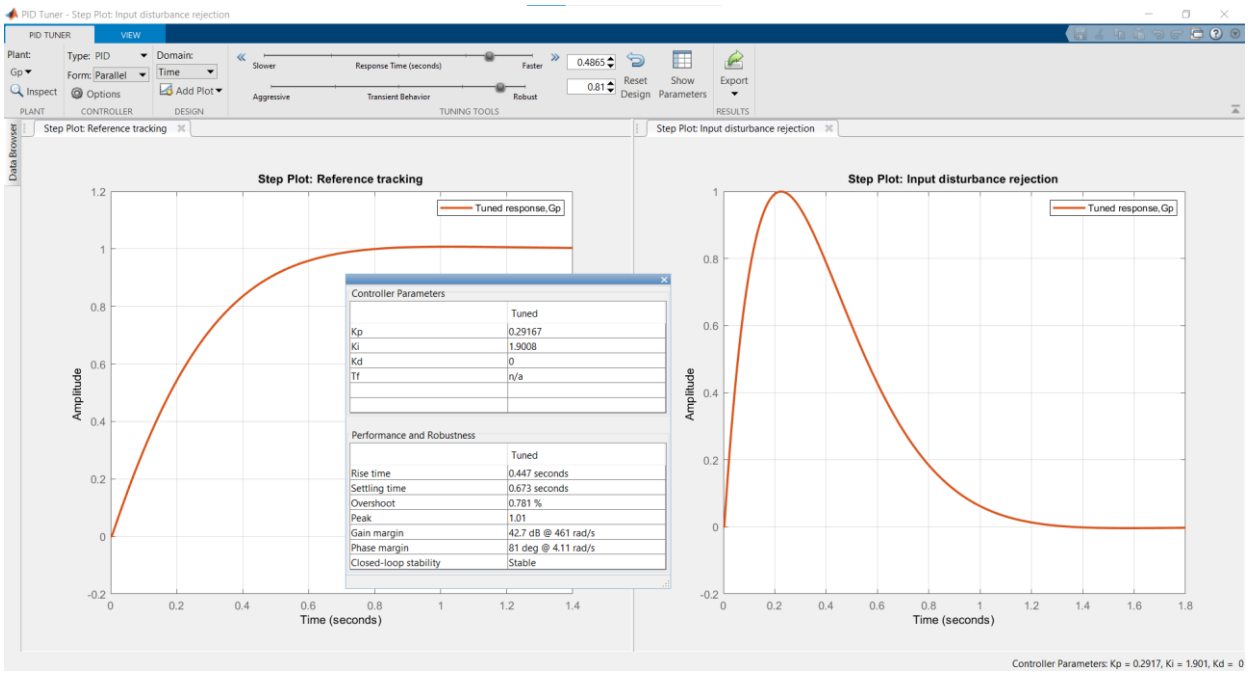
- **Sintonización de controlador PID**

Posteriormente se utilizó la herramienta PID Tuner de Matlab para estimar los parámetros del controlador PID. Se desea obtener una respuesta rápida y de preferencia sin sobreimpulso. Mediante esta herramienta se obtuvieron los siguientes valores:

$K_P = 0.2917$, $K_I = 1.9$, $K_D = 0$ para una entrada escalón.

Figura 17

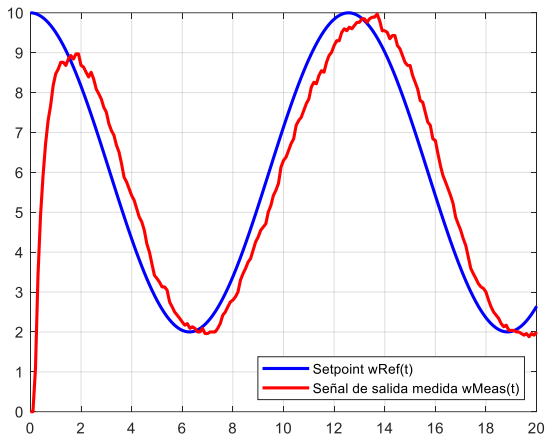
Parámetros PID obtenidos en PID Tuner



Se observó que la planta converge bien para una entrada tipo escalón. Pero el presente sistema robótico trabajará con entradas de forma senoidal, y se observó que no consigue seguir muy bien a las entradas $w_{Ref}(t)$ de ese tipo.

Figura 18

Respuestas ante señal senoidal

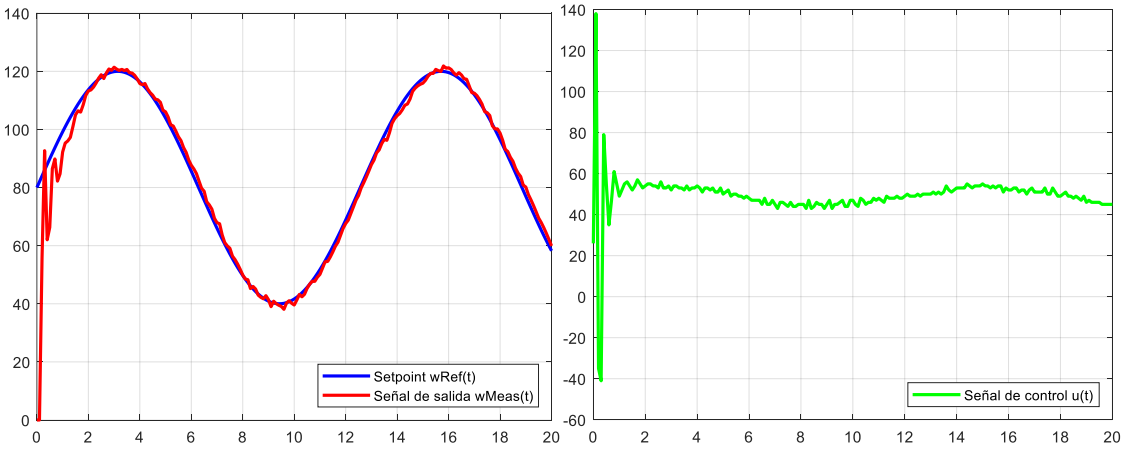


Por lo tanto, fue necesario reajustar los parámetros PID mediante prueba y error.

Luego, se obtuvo los valores K_P = 1.5, K_I = 4.6, K_D = 0.18 para Motor $_R$ y K_P = 1.8, K_I = 4.6, K_D = 0.18 para Motor $_T$. Se obtuvo las siguientes gráficas con esos valores.

Figura 19

Respuesta y señal de control u(t) ante entrada senoidal



Nota. Los parámetros K_P, K_I, K_D fueron reajustados respecto a los obtenidos anteriormente con la herramienta PID Tuner

El diagrama de flujo para la programación en Arduino (Anexo B: Programación)

del sistema en lazo cerrado, se muestra en la siguiente figura:

Figura 20

Diagrama de flujo lazo cerrado Arduino

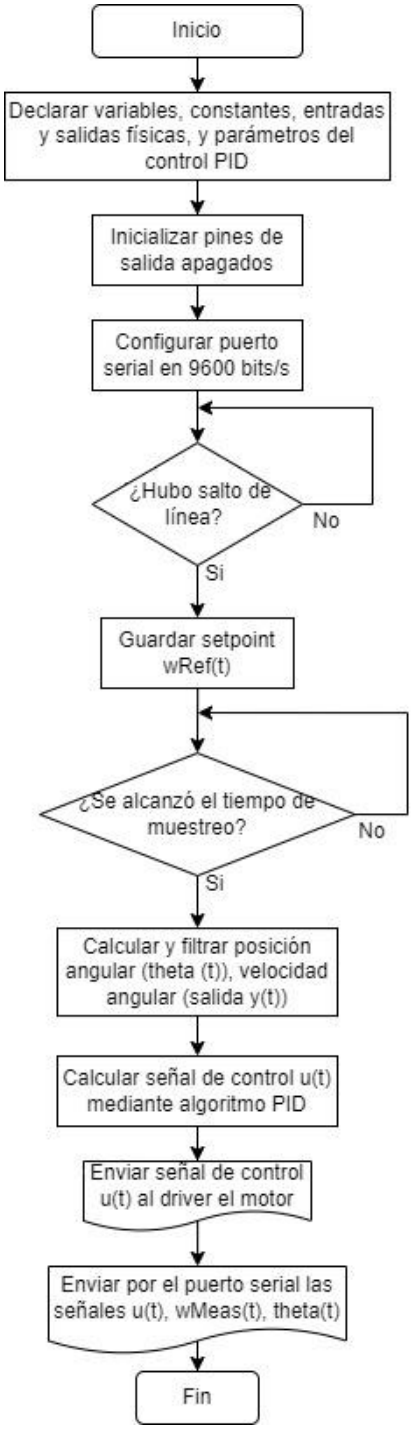
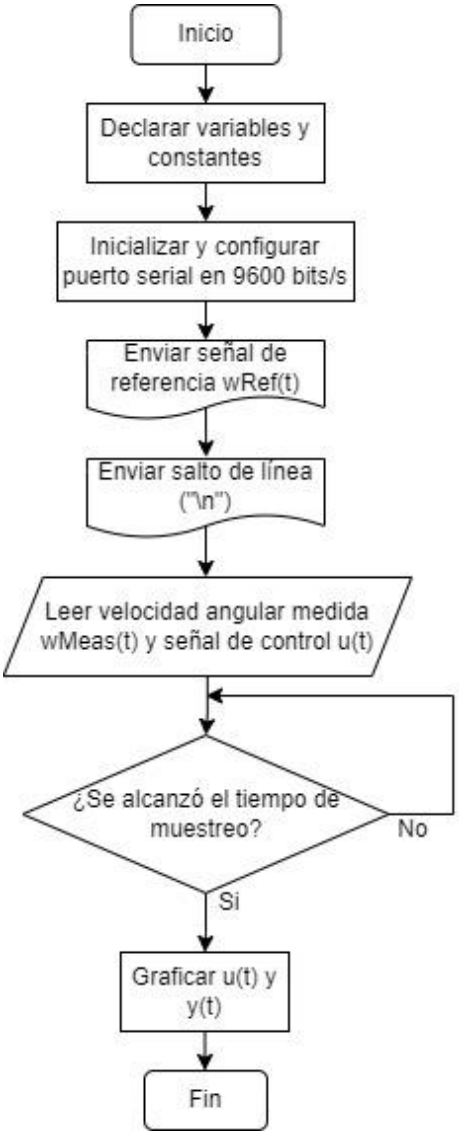


Diagrama de flujo para la programación en Matlab (Anexo B: Programación) del sistema

en lazo cerrado:

Figura 21

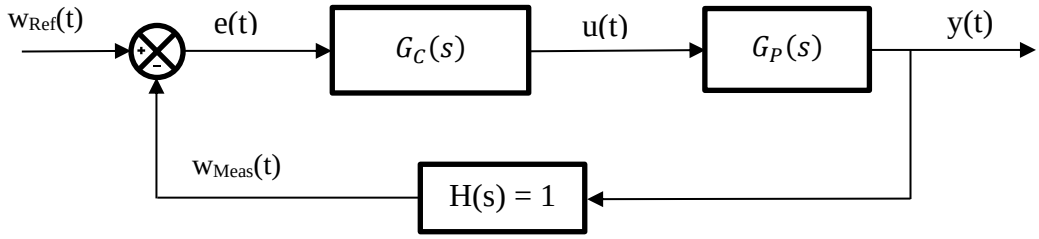
Diagrama de flujo lazo cerrado Matlab



Finalmente se obtiene el sistema de control del lazo interno, representado en el siguiente diagrama de bloques:

Figura 22

Diagrama de bloques del lazo interno



Donde:

$$G_C(s) = K_P + \frac{K_I}{s} + K_D * s = 1.50 + \frac{4.60}{s} + 0.18 * s \quad (4)$$

$$G_P(s) = \frac{K}{\tau * s + 1} * e^{-L*s} = \frac{2.0695}{0.4323 * s + 1} * e^{-0.2287*s} \quad (5)$$

- **Análisis de estabilidad del lazo interno**

Cuando se diseña un sistema de control de lazo cerrado, se debe realizar un análisis de estabilidad, ya que es necesario asegurarse que el sistema es estable para un funcionamiento correcto.

> Una condición necesaria y suficiente para que un sistema de lazo cerrado sea estable es que todos los polos de la función de transferencia del sistema tengan partes reales negativas. Es decir, un sistema es estable si todos los polos de la función de transferencia están en el lado izquierdo del plano s. (Dorf & Bishop, 2022, pág. 397)

$$T(s) = \frac{Y(s)}{w_{Ref}(s)} = \frac{G_C(s) * G_P(s)}{1 + G_C(s) * G_P(s)} \quad (6)$$

De la cual se obtiene la ecuación característica (denominador) del sistema:

$$q(s) = 1 + G_C(s) * G_P(s) = 0 \quad (7)$$

Debido a que $G_P(s)$ tiene un retardo, es necesario realizar una aproximación de Padé:

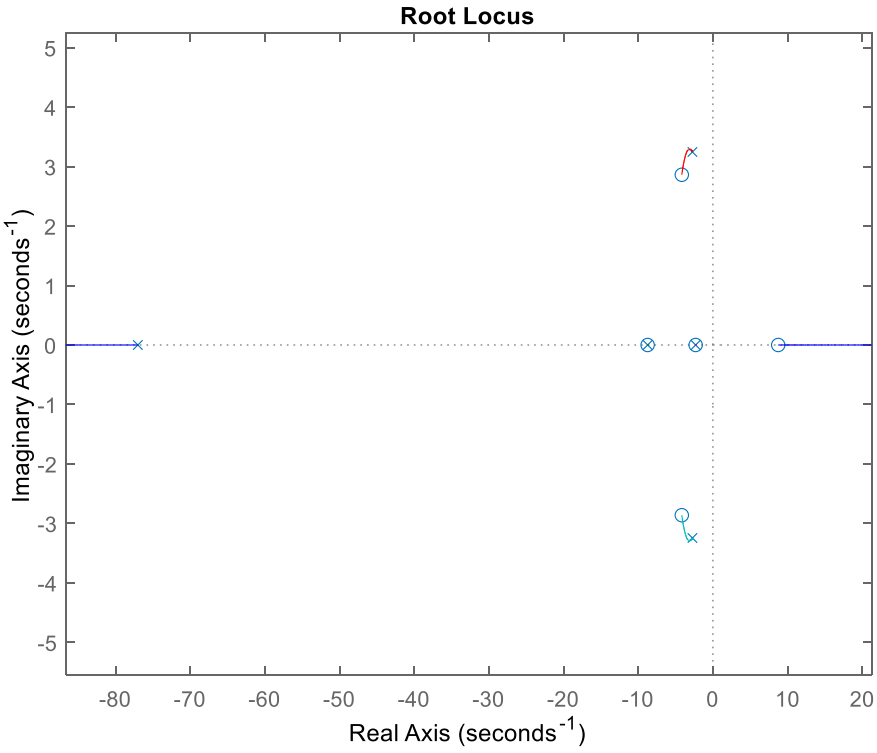
$$e^{-T\,s} = \frac{e^{\frac{-T\,s}{2}}}{e^{\frac{T\,s}{2}}} = \frac{1 - \frac{T\,s}{2} + \frac{1}{2!}\left(\frac{T\,s}{2}\right)^2 - \cdots}{1 + \frac{T\,s}{2} + \frac{1}{2!}\left(\frac{T\,s}{2}\right)^2 - \cdots} \quad (8)$$

(Pinto & Matía, 2015, pág. 57)

En Matlab se puede realizar mediante el comando “pade(Gp, orden)”. En este caso se hizo una aproximación de orden 1. La programación del script de Matlab para la ubicación de polos se encuentra en el Anexo C.

Figura 23

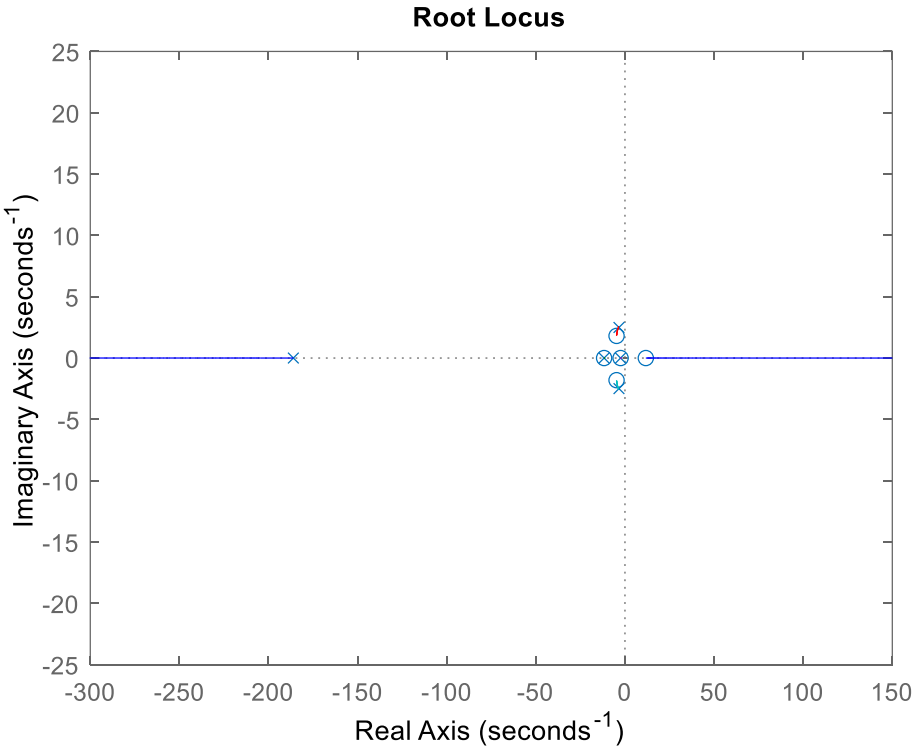
Ubicación de polos ('x') de motor $_T$



Nota. Los polos del sistema se muestran con un símbolo “x”, y los ceros con un símbolo “o”

Figura 24

Ubicación de polos ('x') de motor $_R$



Nota. Los polos del sistema se muestran con un símbolo “x”, y los ceros con un símbolo “o”

P_T =	P_R =
-77.0286 + 0.0000i -8.7451 + 0.0000i -2.7458 + 3.2461i -2.7458 - 3.2461i -2.3132 + 0.0000i	1.0e+02 * -1.8605 + 0.0000i -0.1170 + 0.0000i -0.0336 + 0.0249i -0.0336 - 0.0249i -0.0246 + 0.0000i

Se obtienen los polos y todos tienen parte real negativa, por lo tanto, se concluye que los sistemas de lazo interno son asintóticamente estables.

3.1.1.2. Lazo externo

En cuanto al lazo externo, se diseñó un controlador para la posición angular del eje del motor, mediante el análisis de estabilidad de Lyapunov.

Por otro lado, la cinemática diferencial directa de un robot está expresado por:

$$dp = J * dq \tag{9}$$

Donde: $dq = \begin{bmatrix} dq_1 \\ \vdots \\ dq_n \end{bmatrix}$, $dp = \begin{bmatrix} dx_1 \\ \vdots \\ dx_n \end{bmatrix}$ y la matriz $J = \begin{bmatrix} \frac{\partial x_1}{\partial q_1} & \cdots & \frac{\partial x_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_m}{\partial q_1} & \cdots & \frac{\partial x_m}{\partial q_n} \end{bmatrix}_{mxn}$

(Tzafestas, 2014, pág. 32)

Para el presente sistema, sería como se detalla a continuación:

$$dq = \dot{q} = \begin{bmatrix} \omega_R(t) \\ \omega_T(t) \end{bmatrix}$$

$$dp = \dot{p} = \begin{bmatrix} \dot{\theta}_R(t) \\ \dot{\theta}_T(t) \end{bmatrix}$$

$$J = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{10}$$

Se obtiene la ecuación para la cinemática directa del sistema:

$$dp = J * dq \rightarrow \begin{bmatrix} \dot{\theta}_R(t) \\ \dot{\theta}_T(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} \omega_R(t) \\ \omega_T(t) \end{bmatrix} \tag{11}$$

- **Análisis de estabilidad del lazo externo**

Se realizó el análisis de estabilidad del sistema, mediante los criterios de estabilidad de Lyapunov.

Una función de Lyapunov $V(x)$ para todo $t > t_0$, es asintóticamente estable si satisface las siguientes 4 condiciones:

- ✓ $V(x)$ es continua y tiene derivada continua (12)

- ✓ $V(0) = 0$ (13)

- ✓ $V(x)$ es continua y tiene derivada continua (14)

- ✓ $\frac{dV(x)}{dt} = \left[\frac{\partial V(x)}{\partial x}\right]^T \frac{dx}{dt} < 0$ para $x \neq 0$ (15)

(Tzafestas, 2014, pág. 146)

Se propone la siguiente función:

$$V(Error_\theta) = \frac{Error_\theta{}^T * Error_\theta}{2} \tag{16}$$

Donde, $Error_\theta$ es el vector de error para la posición angular de rodilla y tobillo.

$$Error_\theta(t) = \theta_{Ref}(t) - \theta_{Meas}(t)$$
$$= \begin{bmatrix} \theta_{Ref_R}(t) \\ \theta_{Ref_T}(t) \end{bmatrix} - \begin{bmatrix} \theta_{Meas_R}(t) \\ \theta_{Meas_T}(t) \end{bmatrix} \tag{17}$$

Luego, analizando la estabilidad de la función propuesta en (16), mediante las condiciones de estabilidad de Lyapunov:

$$V(Error_\theta) \text{ y } \frac{dV(Error_\theta)}{Error_\theta} \text{ son continuas} \tag{18}$$

$$V(Error_\theta = 0) = 0 \tag{19}$$

$$V(Error_\theta) > 0 \text{ para todo } Error_\theta \neq 0 \tag{20}$$

$$\frac{dV(Error_\theta)}{dt} = Error_\theta(t)^T * \dot{Error}_\theta(t) < 0 \tag{21}$$

Para cumplir con la cuarta condición señalada en (21), se sustituye:

$$\begin{gathered} \dot{Error}_\theta(t) = -k * Error_\theta(t) \\ Error_\theta(t) = -\begin{bmatrix} k_R & 0 \\ 0 & k_T \end{bmatrix} * \begin{bmatrix} Error_{\theta_R}(t) \\ Error_{\theta_T}(t) \end{bmatrix} \end{gathered} \tag{22}$$

Para obtener:

$$\dot{Error}_\theta(t) = -k * Error_\theta(t)^T * Error_\theta(t) < 0 \tag{23}$$

Es decir, el sistema será asintóticamente estable siempre y cuando se cumpla con la condición:

$$k > 0 \tag{24}$$

Luego, reordenando la ecuación de cinemática del sistema:

$$\begin{gathered} \dot{p} = J * \dot{q} \\ \dot{q} = J^{-1} * \dot{p} \\ \dot{q}(t) = J^{-1} * \dot{\theta}(t) \end{gathered} \tag{25}$$

Y dado el error (vector 2x1):

$$\begin{gathered} \dot{Error}_\theta(t) = \dot{\theta}_{Ref}(t) - \dot{\theta}_{Meas}(t) \\ \dot{\theta}_{Meas}(t) = \dot{\theta}_{Ref}(t) - \dot{Error}_\theta(t) \end{gathered} \tag{26}$$

Luego, ya que el sensor encoder mide la posición angular del eje:

$$\theta(t) = \theta_{Meas}(t)$$

$$\dot{\theta}(t) = \dot{\theta}_{Meas}(t)$$

Reemplazando en la ecuación de la cinemática del sistema (25) :

$$\dot{q}(t) = J^{-1} * \dot{\theta}_{Meas}(t) \tag{27}$$

, luego reemplazando (26) en (27) :

$$= J^{-1} * (\dot{\theta}_{Ref}(t) - \dot{Error}_{\theta}(t)) \tag{28}$$

Y finalmente reemplazando la sustitución realizada (22) en (28), se obtiene la ley de control:

$$\dot{q}(t) = J^{-1} * (\dot{\theta}_{Ref}(t) + k * Error_{\theta}(t)) \tag{29}$$

Donde:

$$J^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\dot{\theta}_{Ref}(t) = \begin{bmatrix} \dot{\theta}_{Ref_R}(t) \\ \dot{\theta}_{Ref_T}(t) \end{bmatrix}$$

$$\dot{q}(t) = \begin{bmatrix} w_{Ref_R} \\ w_{Ref_T} \end{bmatrix}$$

$$k = \begin{bmatrix} k_R & 0 \\ 0 & k_T \end{bmatrix}$$

$$Error_{\theta}(t) = \begin{bmatrix} Error_{\theta_R}(t) \\ Error_{\theta_T}(t) \end{bmatrix}$$

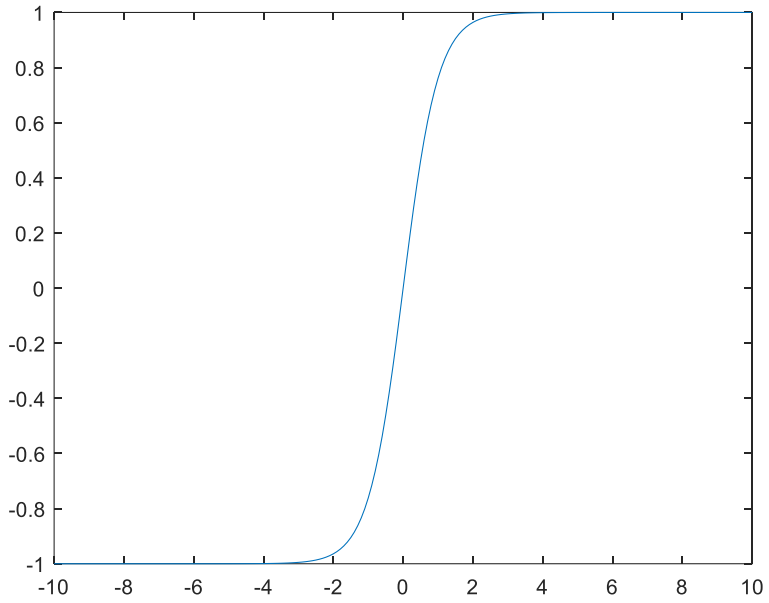
En este caso la matriz jacobiana J era un escalar igual a 1, pero como tobillo y rodilla son independientes se agrupó J en una matriz diagonal $2x2$. La expresión $\dot{h}(t)$ representa la variable controlada, en este caso la posición angular $\theta_{R,T}(t)$; y $q(t)$ es la

variable manipulada, en este caso la velocidad angular $w_{R,T}(t)$.

Por otro lado, la función tangente hiperbólica ($tanh$ en Matlab) tiene una forma acotada con magnitud igual a 1, como se muestra en la siguiente figura.

Figura 25

Función tangente hiperbólica



Debido a este motivo se usó esta función para saturar el error de theta, de la forma “tanh ($Error_\theta$)”. Finalmente, la ley de control del lazo externo en su forma final para la implementación queda como se muestra a continuación:

$$\dot{q}(t) = J^{-1} * \left(\dot{\theta}_{Ref}(t) + k * \tanh\left(Error_\theta(t)\right)\right) \quad (30)$$

3.1.2 Subsistema mecánico

El diseño del subsistema mecánico, las simulaciones y análisis de esfuerzos de las piezas se realizaron en el software SolidWorks.

En base al análisis de los antecedentes, se sugiere que el material tenga las siguientes características: alta rigidez, alta resistencia, peso liviano, resistente a la corrosión, facilidad para realizar mantenimiento y bajo costo. (Alburqueque & Rondón, 2019). Adicionalmente, también se consideró la facilidad para realizar soldadura y disipar calor, considerando la posibilidad de implementarlo en físico.

El material estructural más común para este tipo de proyectos es el aluminio, existen varias aleaciones con diferentes características. Para seleccionar al más adecuado se hizo una comparación entre las características de 3 diferentes aleaciones de aluminio.

Tabla 3

Tabla comparativa de aleaciones de aluminio

Parámetro	Aluminio 5052-O	Aluminio 6061-O	Aluminio 7075-O
Densidad (MPa)	2680	2700	2810
Límite elástico (MPa)	89.6	55.2	103
Tensión última (MPa)	193	124	228
Módulo de Young (MPa)	70300	68900	71700
Soldabilidad	Excelente	Muy Buena	Malo
Resistencia a la corrosión	Excelente	Buena	Menor
Mecanizado	Regular-Malo	Excelente	Regular
Conductividad térmica (W/m-K)	138	180	173

Aplicaciones comunes	Tubos hidráulicos, equipo marino, recipientes a presión.	Ensambles soldados, plataformas, accesorios eléctricos, sujetadores.	Utilaje industrial, equipo de escalada, utensilios de acampada, armas de fuego.

Fuente: https://www.industrialmetalsupply.com/blog/6061-vs-5052-aluminum, https://www.dekmake.com/es/6061-contra-7075-aluminio/ y datos de https://www.matweb.com/

De los datos de la tabla, se observa que la aleación 5052-O tiene las propiedades mecánicas más adecuadas. Sin embargo, dicha aleación se comercializa comúnmente en placas o láminas de metal, y no en tubos cuadrados. Además, por lo general se utiliza el aluminio 5052 en aplicaciones donde la soldabilidad y la resistencia a la corrosión son factores muy severos, como en las aplicaciones marinas. (Gabrian International Ltd., 2020)

Debido a lo descrito anteriormente, se eligió el aluminio 6061-O, ya que es más comercial y se puede encontrar en tubos cuadrados. Además, tiene las propiedades mecánicas suficientes para el presente sistema, con un factor de seguridad mucho mayor a 1, como se evidencia en la sección 3.1.2.2 Análisis de esfuerzos.

El aluminio 6061-O es un material de baja densidad, presenta alta resistencia mecánica y es resistente a la corrosión. Debido a sus propiedades mecánicas se le conoce como aluminio estructural (Gabrian International Ltd., 2020).

La estructura mecánica se diseñó y simuló en el software SolidWorks, considerando tubos de aluminio 6061-O de sección rectangular, con medidas 40 x 70mm y espesor de 3.6mm.

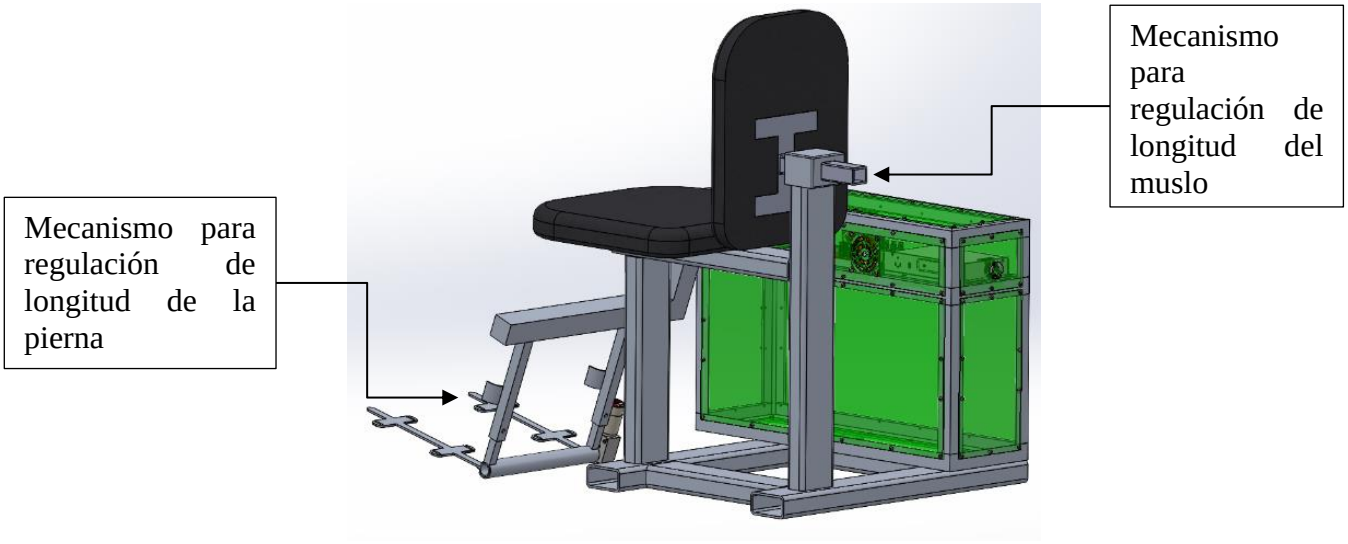
También se diseñaron dos mecanismos que permitirán regular la longitud de las secciones del muslo y la pierna. Se regulará la longitud de la sección del muslo

desplazando el espaldar, y la longitud de la pierna mediante un mecanismo tipo telescópico.

Por último, se diseñó un gabinete, el cual protegerá a los motores, la fuente de voltaje DC y los circuitos electrónicos del sistema.

Figura 26

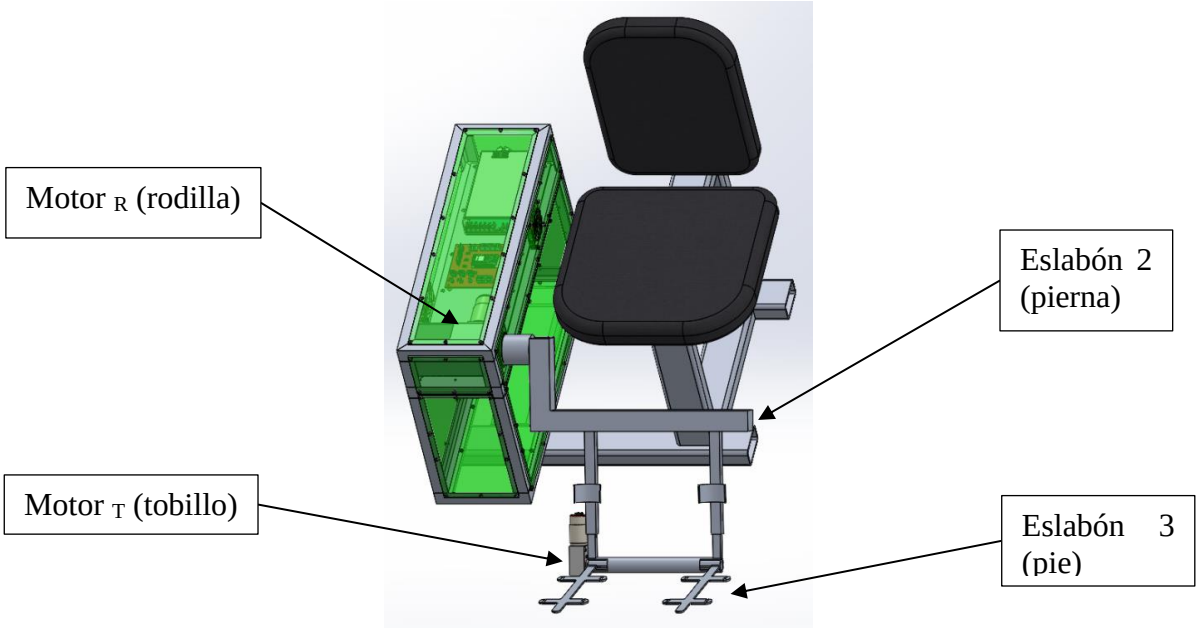
Vista posterior de la estructura



Las piezas eslabón 2 y eslabón 3, correspondientes a la pierna y el pie respectivamente, son las que serán accionadas (en rotación a su eje) por los motores de la rodilla y el tobillo (motor $_R$ y motor $_T$, respectivamente), como se muestra en la siguiente figura:

Figura 27

Vista en 3D de la estructura



3.1.2.1. Dimensionamiento de los motores

Se dimensionó los torques necesarios para cada motor aplicando las ecuaciones de la dinámica inversa mediante una programación realizada en Matlab (Anexo B: Programación). Al respecto, Lynch & Park (2017) mencionan que la dinámica inversa encuentra las fuerzas y torques en las articulaciones, correspondientes al estado del robot y las aceleraciones deseadas (pág.269). Luego, los valores hallados se corroboraron con la herramienta “Motor Sizing” de Oriental Motor.

Se usaron los datos obtenidos de SolidWorks y datos obtenidos de la investigación de antropometría del autor Paolo de Leva.

Figura 28

Propiedades físicas del eslabón 2

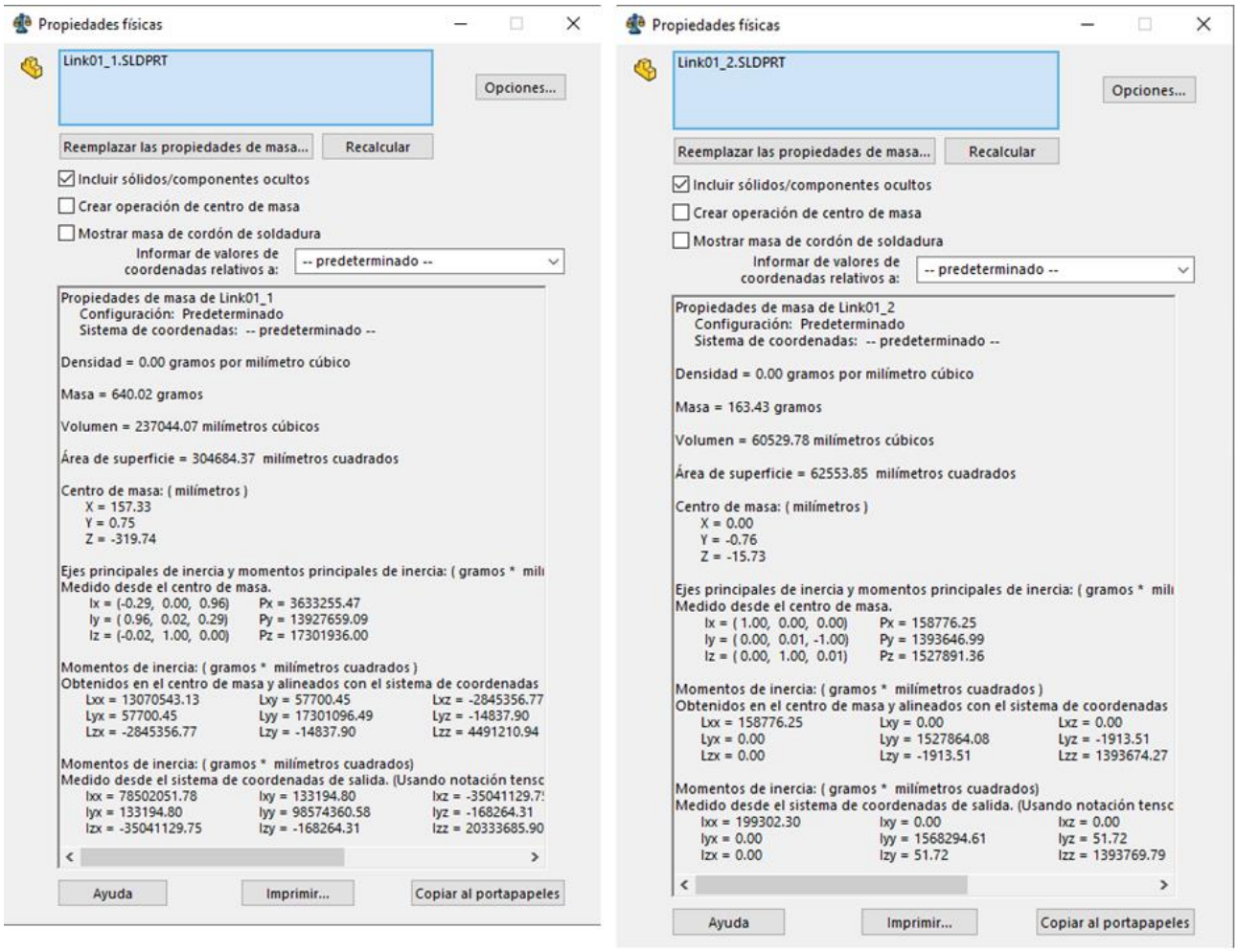
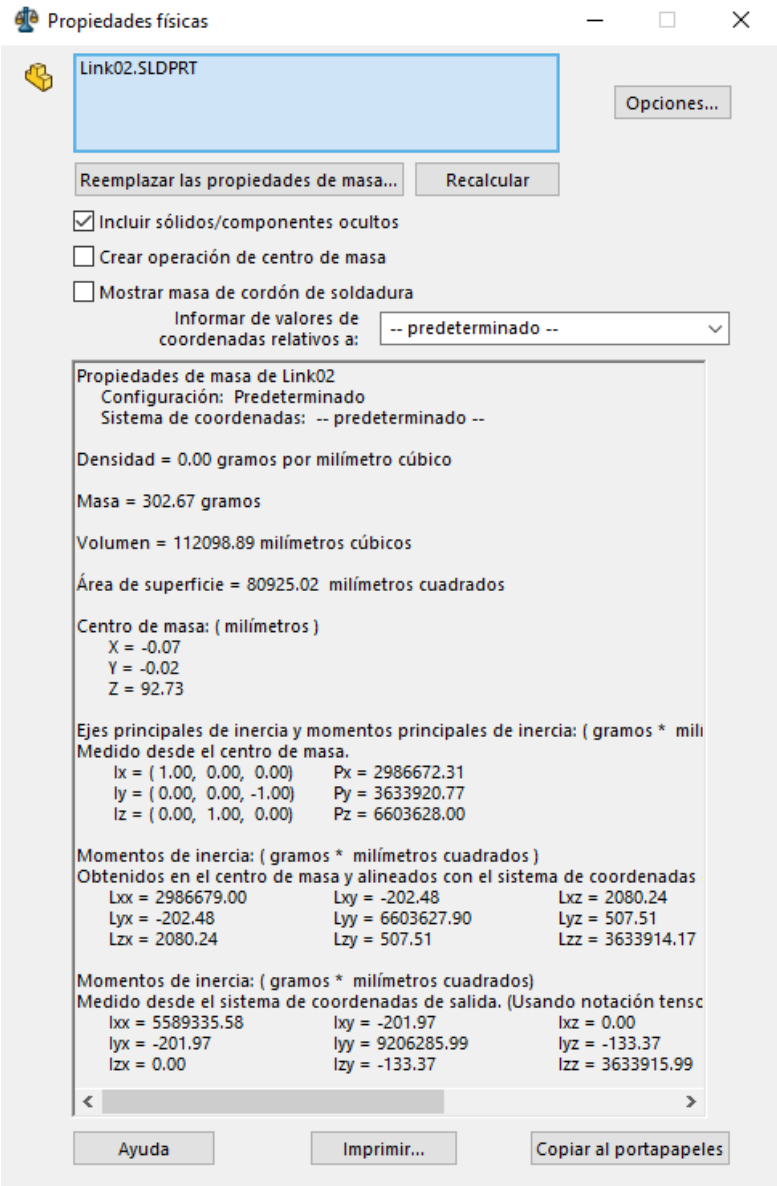


Figura 29

Propiedades físicas del eslabón 3



De acuerdo con los datos antropométricos tomados de la investigación de de Leva (1996), se incluyen los pesos para los segmentos de la pierna y el pie en las siguientes tablas:

Tabla 4

Datos antropométricos de los segmentos de la pierna y el pie

	Mujeres (peso = 61.9 kg, estatura = 1.73 m)			Hombres (peso = 73 kg, estatura = 1.74 m)		
	L (cm)	CM (cm)	m (kg)	L (cm)	CM (cm)	m (kg)
Pierna	43.2	19.08	2.98	43.4	19.35	3.16
Pie	22.8	9.15	0.80	25.8	11.39	1.00

Tabla 5

Datos antropométricos de los segmentos de la pierna y el pie (promedio)

	L (m)	CM (m)	m (kg)
Pierna	0.433	0.1921	3.07
Pie	0.243	0.1027	0.90

Con estos datos se procedió a calcular el peso total que soportará cada articulación para dimensionar los motores.

Tabla 6

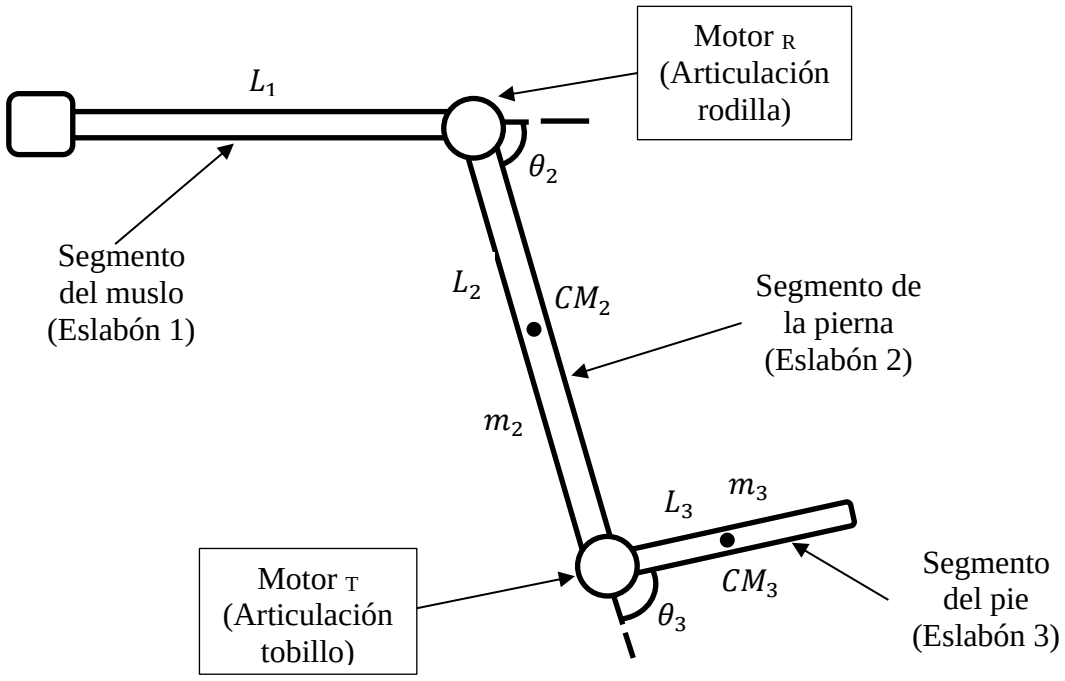
Pesos soportados en cada articulación

Articulación	Pesos que soporta (componentes)	Peso total(kg)
Motor $_R$	Eslabón 2 + Eslabón 3 + Piernas+ pies+ motor $_T$	(0.640+0.163) +0.302+2*3.07+2*0.90+0.30 = 9.345

Motor $_T$	Eslabón 3 + pies	0.302+2*0.90 = 2.102

Figura 30

Diagrama esquemático del sistema robótico



Estos datos se introdujeron en las ecuaciones de la dinámica inversa para estimar los torques necesarios en el motor $_R$ y motor $_T$, $\tau_2(t)$ y $\tau_3(t)$ respectivamente. De acuerdo con Lynch & Park (2017), los torques generados para un sistema planar de 2 GDL con juntas rotativas están dados por:

$$\begin{aligned}\tau_2 = {} & \left(m_2 L_2^2 + m_3(L_2^2 + 2 * L_2 L_3 \cos\theta_3 + L_3^2)\right)\ddot{\theta}_2 \\ & + m_3(L_2 L_3 \cos\theta_3 + L_3^2)\ddot{\theta}_3 \\ & - m_3 L_2 L_3 \sin\theta_3 \left(2 * \dot{\theta}_2\dot{\theta}_3 + \dot{\theta}_3^2\right) \\ & + (m_2 + m_3) L_2 g \cos\theta_2 + m_3 g L_3 \cos(\theta_2 + \theta_3)\end{aligned} \tag{31}$$

$$\begin{aligned}\tau_3 = {} & m_3(L_2 L_3 \cos\theta_3 + L_3^2)\ddot{\theta}_2 + m_3 L_3^2 \ddot{\theta}_3 + m_3 L_2 L_3 \dot{\theta}_2^2 \sin\theta_3 \\ & + \ m_3 g L_3 \cos(\theta_2 + \theta_3)\end{aligned} \tag{32}$$

Los cuales pueden ser agrupados en 3 componentes (para aligerar la programación): Mass matrix ($M(\theta)$), Centripetal-coriolis ($c(\theta, \dot{\theta})$) y torque gravitacional ($g(\theta)$).

$$M(\theta) = \begin{bmatrix} m_2 L_2^2 + m_3(L_2^2 + 2L_2 L_3 \cos\theta_3 + L_3^2) & m_3(L_2 L_3 \cos\theta_3 + L_3^2) \\ m_3(L_2 L_3 \cos\theta_3 + L_3^2) & m_3 L_3^2 \end{bmatrix}$$

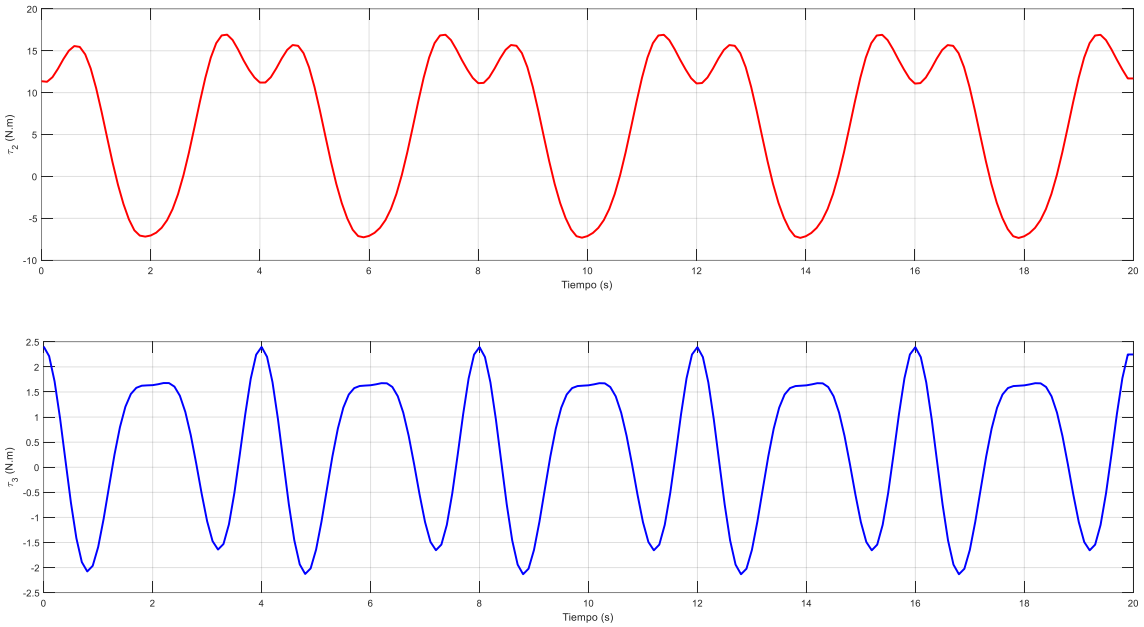
$$c(\theta, \dot{\theta}) = \begin{bmatrix} -m_3 L_2 L_3 \sin\theta_3 \left(2\dot{\theta}_2\dot{\theta}_3 + \dot{\theta}_3^2\right) \\ m_3 L_2 L_3 \dot{\theta}_2^3 \sin\theta_3 \end{bmatrix}$$

$$g(\theta) = \begin{bmatrix} (m_2 + m_3) L_2 g \cos\theta_2 + m_3 g L_3 \cos(\theta_2 + \theta_3) \\ m_3 g L_3 \cos(\theta_2 + \theta_3) \end{bmatrix}$$

Se introdujeron estas fórmulas en un script de Matlab (Anexo C), y se obtuvo las siguientes gráficas:

Figura 31

Gráficas de torque vs tiempo ejercido por los motores



Para corroborar los valores de torque hallados, se usó la herramienta Motor Sizing Tool de Oriental Motor, como se muestra en las siguientes figuras:

Figura 32

Torque del Motor $_R$ calculado mediante la herramienta Motor Sizing

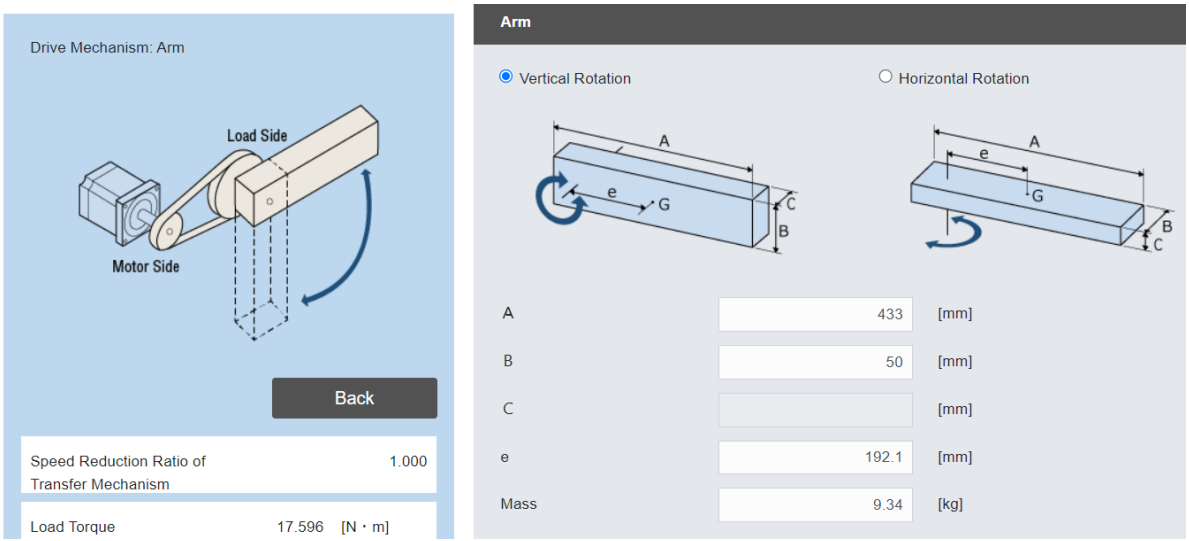
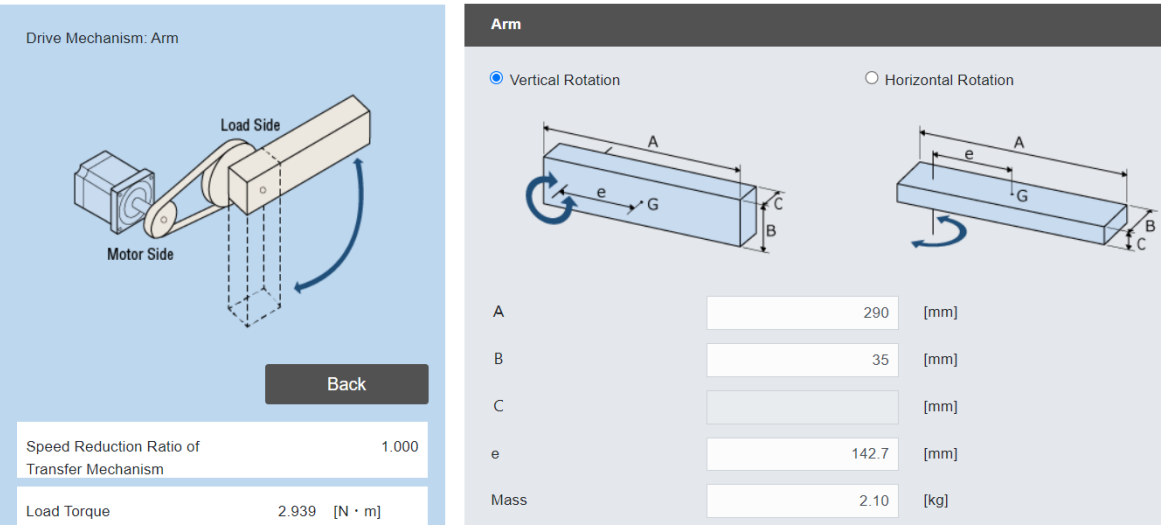


Figura 33

Torque del motor $_T$ calculado mediante la herramienta Motor Sizing



Haciendo la conversión de unidades de una magnitud *x* de newton-metro (N.m) a kilogramo-centímetro (kg.cm):

$$x\ [N.m] * \frac{1\ [kgf]}{9.807\ [N]} * \frac{100\ [cm]}{1\ [m]} = 10.1968 * x\ [kg.cm] \quad (33)$$

Y se obtuvo: $10.117\ \text{N.m} = 103.161\text{kg.cm}$ para el motor $_R$ y $1.6930\ \text{N.m} = 17.2631\text{kg.cm}$ para el motor $_T$. Además, se sobredimensionó en un 20%, para tener en cuenta la fricción y tener un margen adicional.

Tabla 7

Dimensionamiento de los motores

	Torque [N.m]	Torque [kg.cm]	Torque [kg.cm] con 20% de margen adicional
Motor $_R$	17.596	179.423	215.307
Motor $_T$	2.939	29.968	35.962

Se obtuvo el valor de 215.307 kg.cm para el motor $_R$ (rodilla), y 35.962 kg.cm para el motor$_T$ (tobillo). A partir de estos datos se realizó la búsqueda de motores que cumplan con estas características.

Tabla 8

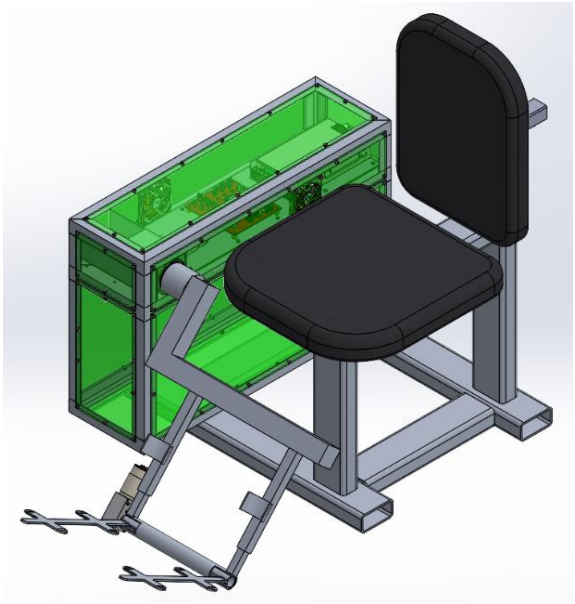
Datos técnicos de los motores seleccionados

Modelo	Torque (kg.cm)	Voltaje (V)	Corriente (A)	Velocidad (rpm)
CM42-775L4-5	253	24	2	8
CM42-775L4-1	66	24	2	29

Nota. Datos tomados de https://es.aliexpress.com/item/1005005526151531.html?spm=a2g0o.productlist.main.1.480577d3mml7Vf&algo_pvid=3f3d939b-c834-4557-9e08-

Figura 34

Vista isométrica del subsistema mecánico



3.1.2.2. Análisis de esfuerzos

Para garantizar la seguridad de la estructura, es necesario que la carga aplicada sea menor a la carga máxima que el elemento puede soportar. Esto debido a que pueden ocurrir eventos imprevistos que no se consideran en el diseño, por ejemplo: impactos, corrosión o desgaste de los materiales (Hibbeler, 2017).

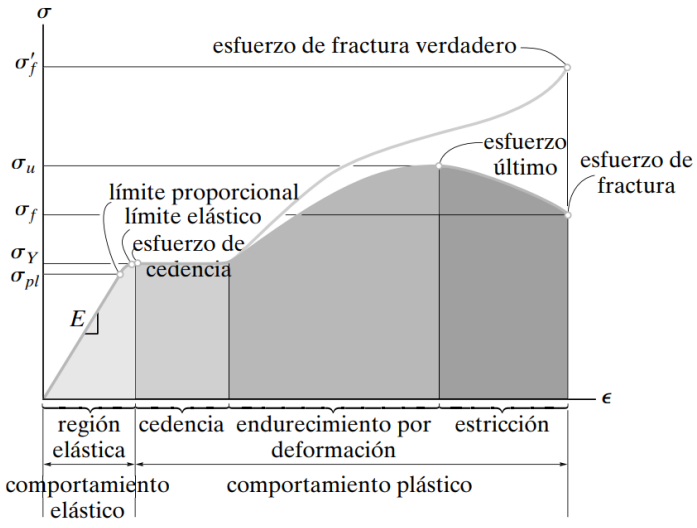
Asimismo, existe un parámetro llamado factor de seguridad que expresa cuánto más fuerte es un elemento de lo que necesita ser para una carga prevista. Al respecto Hibbeler (2017) afirma: “Un método para especificar la carga permisible en un elemento consiste en usar un número llamado factor de seguridad (F.S.), el cual es una razón de la carga de falla F_{falla} sobre la carga permisible F_{perm}” (pág.46).

$$F.S = \frac{F_{falla}}{F_{perm}} = \frac{Límite\ elástico}{Esfuerzo\ máximo\ de\ diseño} \tag{34}$$

En donde se considera que el elemento falla cuando se deforma permanentemente. Es decir, cuando sale de la región elástica y entra en la región plástica (límite elástico).

Figura 35

Diagrama de esfuerzo-deformación



Se realizó el análisis de esfuerzos y factor de seguridad (F.S) mediante el software

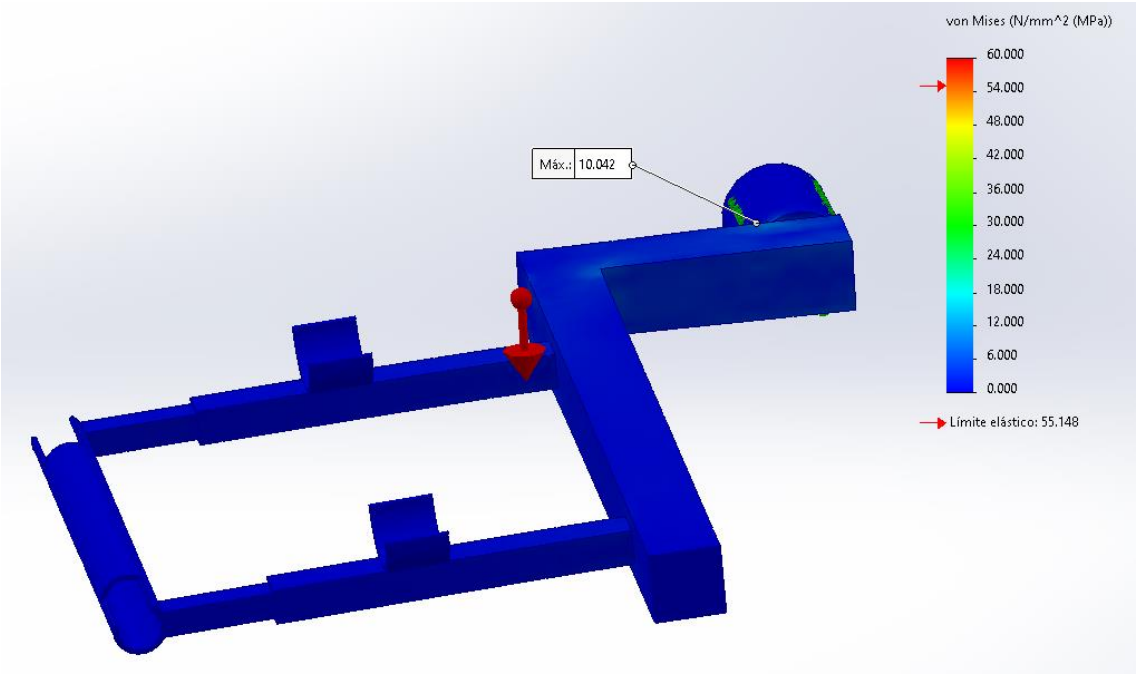
SolidWorks, para los eslabones 2 y 3, los cuales soportan el peso del segmento de la pierna y el pie respectivamente, además del propio peso de la estructura.

- **Análisis del eslabón 2**

Mediante el software SolidWorks se observa que el punto de mayor esfuerzo tiene un valor de 10.042 MPa, y el límite elástico es de 55.148 MPa.

Figura 36

Simulación esfuerzo eslabón 2



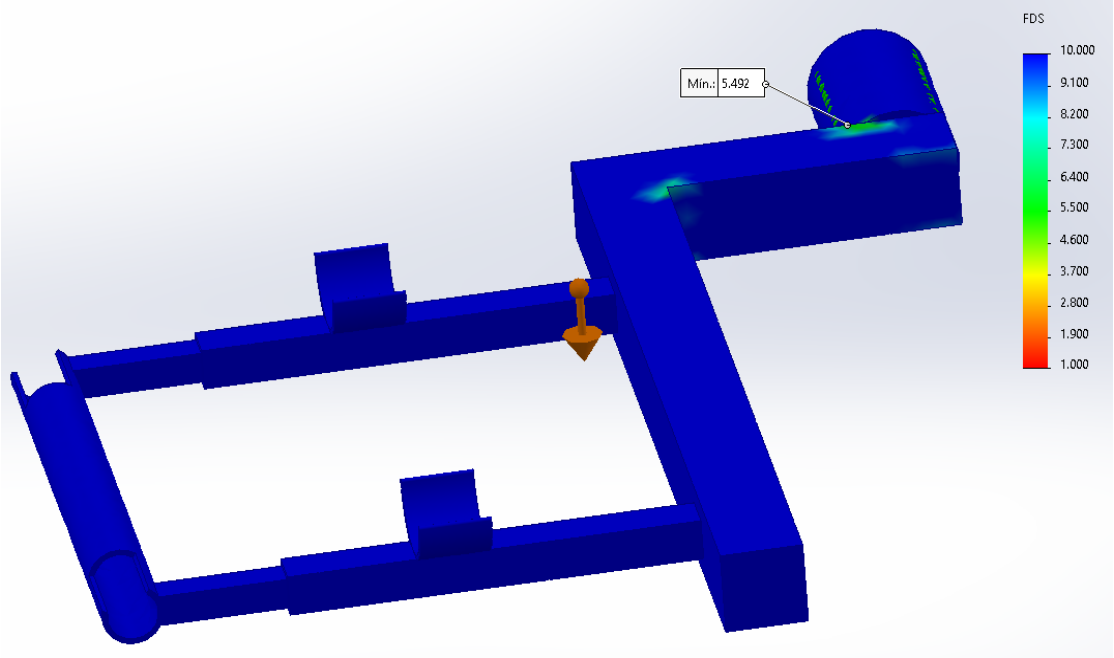
Por lo tanto, el F.S para esta pieza sería:

$$F.S = \frac{Límite\ elástico}{Esfuerzo\ máximo\ de\ diseño} = \frac{55.148}{10.042} = 5.492$$

Y esto se corroboró en el software SolidWorks, donde se obtuvo el mismo valor.

Figura 37

Simulación del factor de seguridad para el eslabón 2

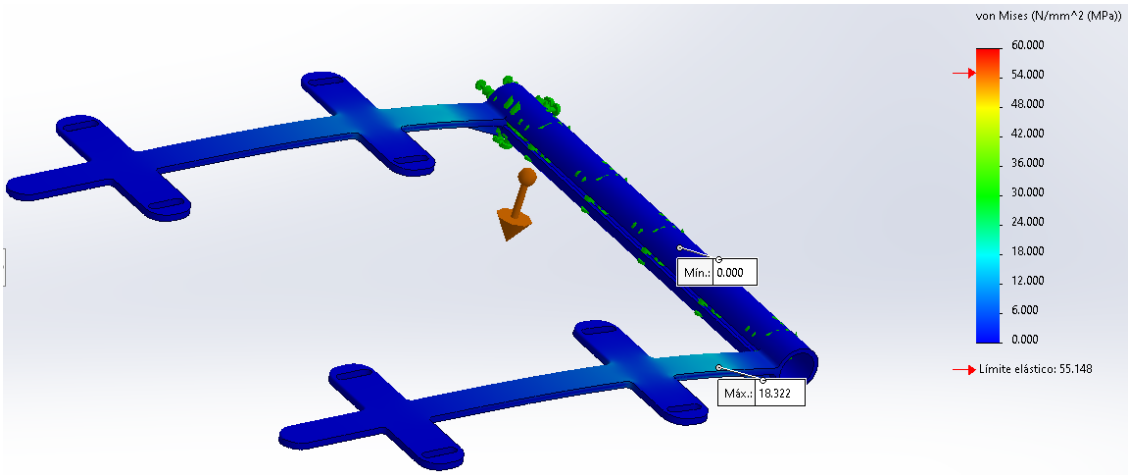


- **Análisis del eslabón 3**

Mediante el software SolidWorks se observa que el punto de mayor esfuerzo tiene un valor de 18.322 MPa, y el límite elástico es de 55.148 MPa.

Figura 38

Simulación esfuerzo eslabón 3



Por lo tanto, el F.S para esta pieza sería:

$$F.S = \frac{Límite\ elástico}{Esfuerzo\ máximo\ de\ diseño} = \frac{55.148}{18.322} = 3.010$$

Y esto se corroboró en el software, donde se obtuvo el mismo valor:

Figura 39

Simulación del factor de seguridad para el eslabón 3

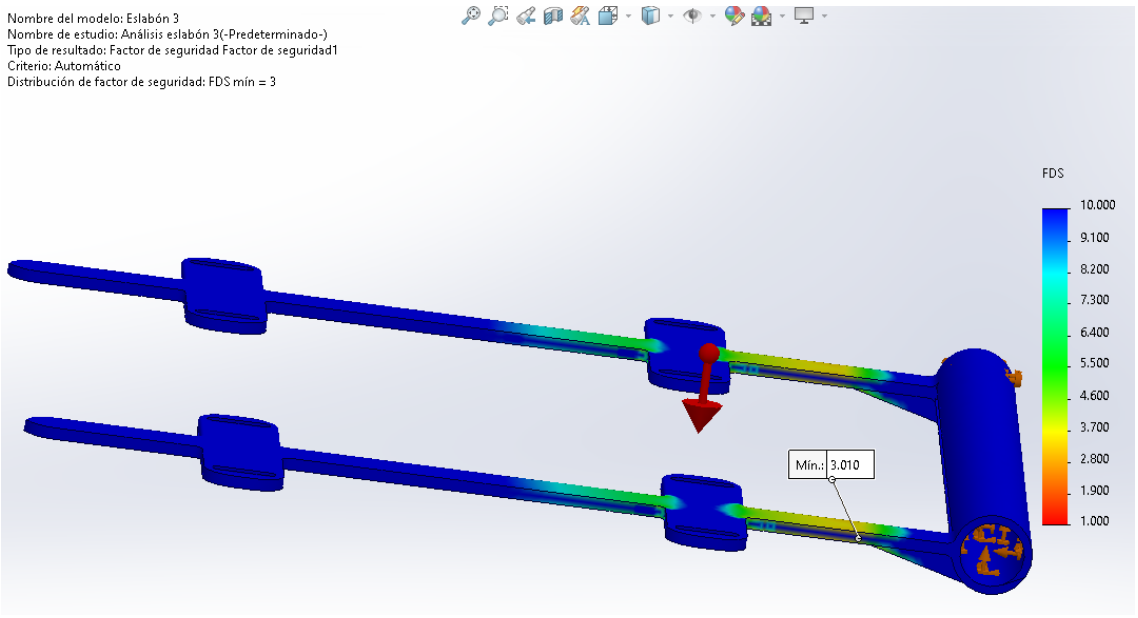
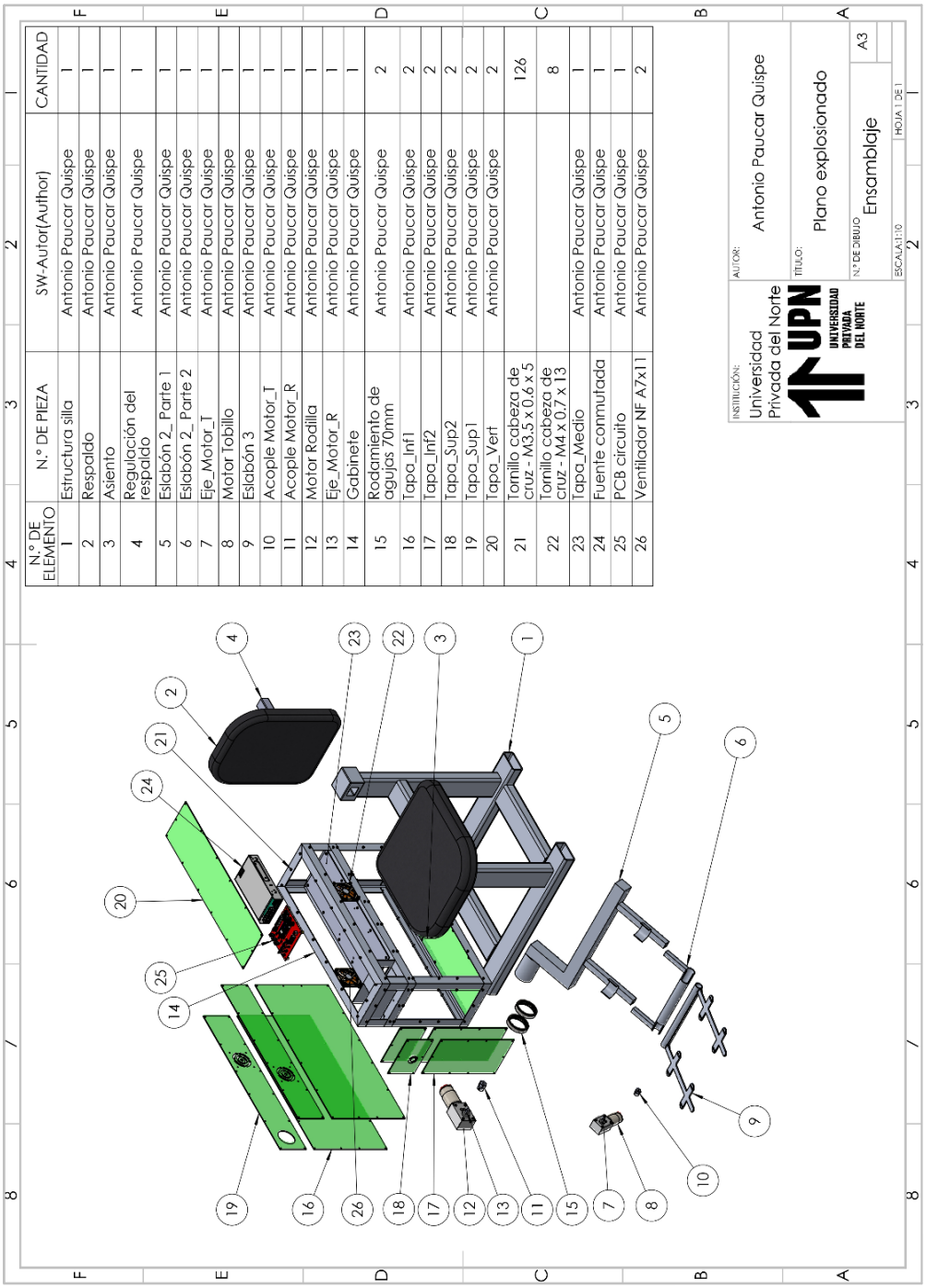


Figura 40

Plano explosionado



3.1.3 Subsistema electrónico

El subsistema electrónico corresponde a todos los dispositivos electrónicos presentes en el sistema robótico, es decir la fuente de alimentación, microcontrolador,

drivers para los motores, encoders, entre otros dispositivos electrónicos como los reguladores de voltaje, entre otros.

3.1.3.1. Selección del microcontrolador

Para la selección del microcontrolador se observaron las características de 3 microcontroladores o tarjetas de desarrollo, y se seleccionó aquel que cumple con las características técnicas necesarias y suficientes para ejecutar el algoritmo de control desarrollado. Además, también se consideró su precio, disponibilidad en el mercado local y tipo de montaje.

Tabla 9

Tabla comparativa de microcontroladores

	Deseable	Arduino Nano	DOIT Esp32 DevKit v1	PIC18F97J94
Voltaje de alimentación(V)	5 o más	6-20	5	5
Voltaje de operación(V)	5	5	3.3	2-3.6
Pines digitales	6	14	24 (GPIO)	100
Pines PWM	4	6	16	7
Pines de interrupción externa	4	2+12(mediante Librería)	32	4
Salida 3.3V	Si	Si	Si	No
Tipo de montaje	Orificio pasante	Orificio pasante	Orificio pasante	Superficial (SMD)
Precio (S/.)	Mínimo	25	55	27

Fuente: https://www.mouser.com/, https://docs.arduino.cc/, https://naylampmechatronics.com/ y https://www.microchip.com/

Se ha seleccionado Arduino Nano, debido a que cumple con las especificaciones técnicas suficientes, como se puede observar en la **Tabla *9***. Además, es de gran ayuda la librería “PinChangeInterrupt”, ya que brinda la flexibilidad de convertir cualquier pin digital en pin para interrupciones externas. Por otro lado, es la opción más económica ya que el ESP32 cuesta más del doble y el PIC18F97J94 es difícil de conseguir en el mercado local (requeriría realizar una importación de varias unidades, y habría que añadir el costo de envío y desaduanaje).

Por otro lado, el Arduino Nano tiene baja potencia de operación y un tamaño pequeño. Además, cuenta con la memoria suficiente para ejecutar el programa elaborado.

Tabla 10

Datos técnicos del Arduino Nano

Arduino Nano	
Microcontrolador	ATmega328
Voltaje de operación I/O	5 V
Corriente DC por Pin I/O	20 mA
Voltaje de Alimentación	5-20 V
Pines I/O Digitales	14
Pines PWM	6
Entradas analógicas	8
Memoria Flash	32 KB
EEPROM	1 KB
Frecuencia de Trabajo	16 MHz
Dimensiones	1.8 x 4.5 cm

Peso	7 g

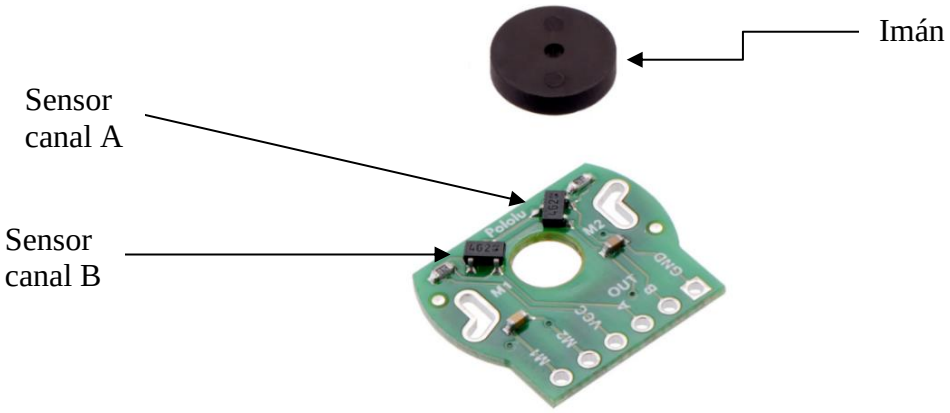
Nota. Datos tomados de https://docs.arduino.cc/

3.1.3.2. Encoder de cuadratura

El encoder de cuadratura de efecto Hall funciona mediante un imán que está acoplado al eje del motor y una PCB fijada en la carcasa con 2 sensores magnéticos de efecto Hall separados en un ángulo de 90°, los cuales generan señales cuadradas desfasadas. En la siguiente figura se puede observar la separación en la ubicación de los sensores.

Figura 41

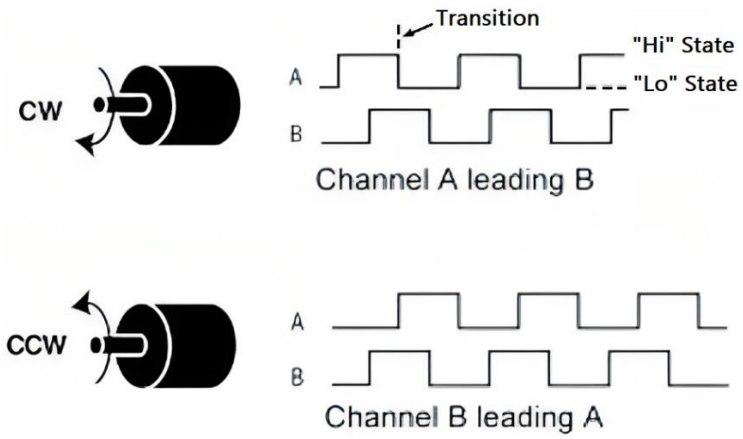
Encoder de cuadratura de efecto Hall



Debido a ese desfase en las señales, es posible identificar el sentido de giro y el ángulo recorrido por el eje, con ayuda del microcontrolador.

Figura 42

Desfase en señales A y B del encoder



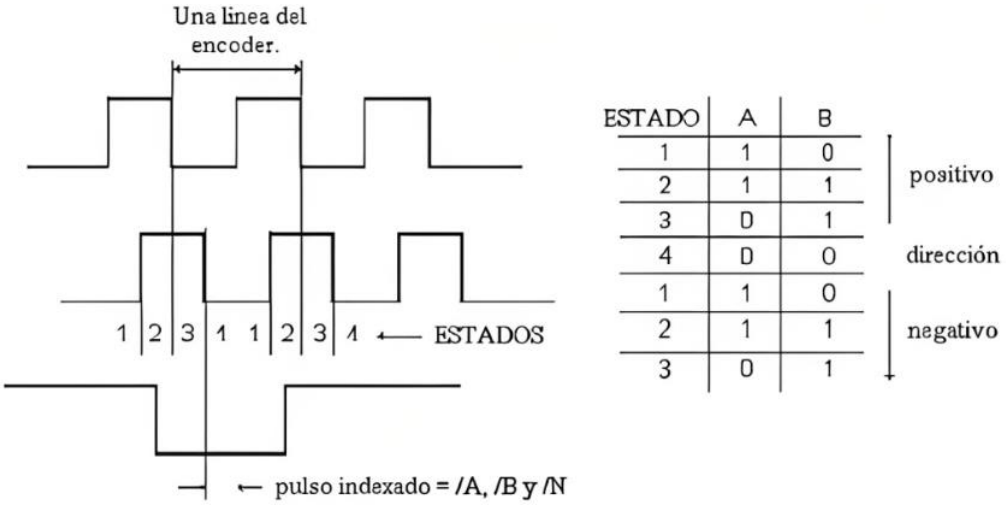
Nota. Tomado de

https://www.dynapar.com/technology/encoder_basics/quadrature_encoder/

En la siguiente figura se puede observar las secuencias en los estados dependiendo del sentido de giro del motor.

Figura 43

Secuencia de estados del encoder



Nota. Tomado de https://androminarobot.blogspot.com/2016/08/encoder-de-cuadratura-y-arduino.html

Tabla 11

Matriz de estados del encoder de cuadratura

		Estado actual			
	AB	(AB=00)	(AB=01)	(AB=10)	(AB=11)
Estado previo	(AB=00)	0	+1	-1	N.E.
	(AB=01)	-1	0	N.E.	+1
	(AB=10)	+1	N.E.	0	-1
	(AB=11)	N.E.	-1	+1	0

Nota: “N.E” significa que no existe esa secuencia.

Programación para los encoders (funciones de interrupción)

Para este sistema se usaron 2 encoders de cuadratura de 2 canales cada uno. Por lo tanto, fueron necesarias 4 entradas del Arduino Nano. Se usaron las entradas D2, D3, D4 y D5.

El microcontrolador Atmega 328P tiene los pines digitales desde el D0 hasta el D7 agrupados en el puerto D (de 8 bits). En Arduino se puede hacer la lectura de todo el puerto mediante la instrucción “PIND”. Para obtener sólo los valores de interés (las entradas D2, D3, D4, D5), se hizo una operación de multiplicación booleana AND (&) en la programación de Arduino.

Tabla 12

Bits del Puerto D

PIND							
D7	D6	D5	D4	D3	D2	D1	D0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

Reemplazando los valores de estos bits de entradas, en la matriz de estados para el encoder de cada motor:

Tabla 13

Matriz de estados del encoder de motor $_R$

		Estado actual (“act_R”)			
		0	16	32	48
Estado previo (“ prev_R”)	0	0	+1	-1	N.E.
	16	-1	0	N.E.	+1
	32	+1	N.E.	0	-1
	48	N.E.	-1	+1	0

Nota: Pines usados D4 (2^4 = 16) y D5 (2^5 = 32)

Tabla 14

Matriz de estados del encoder de motor $_T$

		Estado actual ("act_T")			
		0	4	8	12
Estado previo (" prev_T")	0	0	+1	-1	N.E.
	4	-1	0	N.E.	+1
	8	+1	N.E.	0	-1
	12	N.E.	-1	+1	0

Nota: Pines usados D4 ($2^2 = 4$) y D5($2^3 = 8$)

Para obtener la máxima precisión posible en la posición angular del eje de los motores, se realizó el conteo de ambos flancos (bajada y subida). Las funciones de interrupción para el conteo de flancos de cada motor ("enconderR" para motor $_R$, y "encoderT" para motor $_T$) son las siguientes:

```
void encoderR(void)
{
  prev_R = act_R;
  act_R = PIND & 48;

  if(prev_R==0  && act_R== 16)  {count_R+=1;}
  if(prev_R==16  && act_R== 48) {count_R+=1;}
  if(prev_R==32  && act_R== 0)  {count_R+=1;}
  if(prev_R==48  && act_R== 32) {count_R+=1;}

  if(prev_R==0 && act_R==32)  {count_R-=1;}
  if(prev_R==16 && act_R==0)  {count_R-=1;}
  if(prev_R==32 && act_R==48) {count_R-=1;}
  if(prev_R==48 && act_R==16) {count_R-=1;}
}

void encoderT(void)
{
```

```
  prev_T = act_T;
  act_T = PIND & 12;

  if(prev_T==0  && act_T== 4)  {count_T+=1;}
  if(prev_T==4  && act_T==12)  {count_T+=1;}
  if(prev_T==8  && act_T== 0)  {count_T+=1;}
  if(prev_T==12 && act_T== 8)  {count_T+=1;}

  if(prev_T==0 && act_T==8)  {count_T-=1;}
  if(prev_T==4 && act_T==0)  {count_T-=1;}
  if(prev_T==8 && act_T==12) {count_T-=1;}
  if(prev_T==12 && act_T==4) {count_T-=1;}
}
```

3.1.3.3. Drivers de los motores

Los drivers de los motores se encargarán de que los motores (motor $_R$ y motor $_T$) ejecuten las velocidades y sentidos de giro requeridos por el microcontrolador en cada instante de muestreo.

Para la selección de este componente se observaron las características de 2 drivers, y se seleccionó aquel que cumple con las especificaciones señaladas en la sección 3.1.2.1.

Tabla 15

Tabla comparativa de drivers para motor

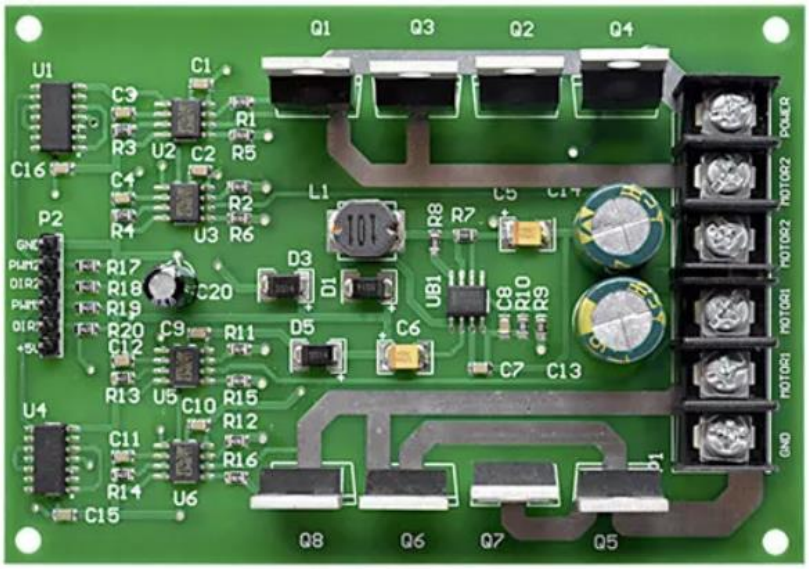
	Deseable	Módulo L298	Módulo IRF 3205x8
Corriente (A)	Superior a 2	Hasta 2	Hasta 10
Salidas Motor DC	2	2	2
Voltaje de potencia (V)	24	5-35	3-36
Voltaje lógico (V)	5-12	5	5

De los datos de la tabla anterior, se observa que el módulo IRF 3205x8 tiene las características suficientes para este sistema robótico, incluso con un margen extra de

corriente, para cuando se presenten valores pico. Por el contrario, el módulo L298 se descartó debido a que fue diseñado para ser usado con motores de baja potencia, en caso fuera usado en este proyecto estaría trabajando en el límite de su capacidad, lo que provocaría que se dañe por sobrecalentamiento. Debido a ello se optó por el módulo IRF3205x8.

Figura 44

Módulo IRF3205x8



Nota. Tomado de https://www.aliexpress.us/item/2255800841312994.html

En la siguiente tabla se detalla el funcionamiento del módulo IRF3205x8.

Tabla 16

Tabla de estados del driver IRF3205x8

	PWM1	DIR1	PWM2	DIR2
Motor_R (Dir 1)	PWM	0	X	X
Motor_R (Dir 2)	PWM	1	X	X

Motor_T (Dir 1)	X	X	PWM	0
Motor_T (Dir 2)	X	X	PWM	1

Donde: 1: Estado lógico alto (5V)

0: Estado lógico bajo (0V)

X: cualquier estado

PWM: señal PWM (0-255 en el Arduino)

Figura 45

Señales PWM para sentido de giro horario

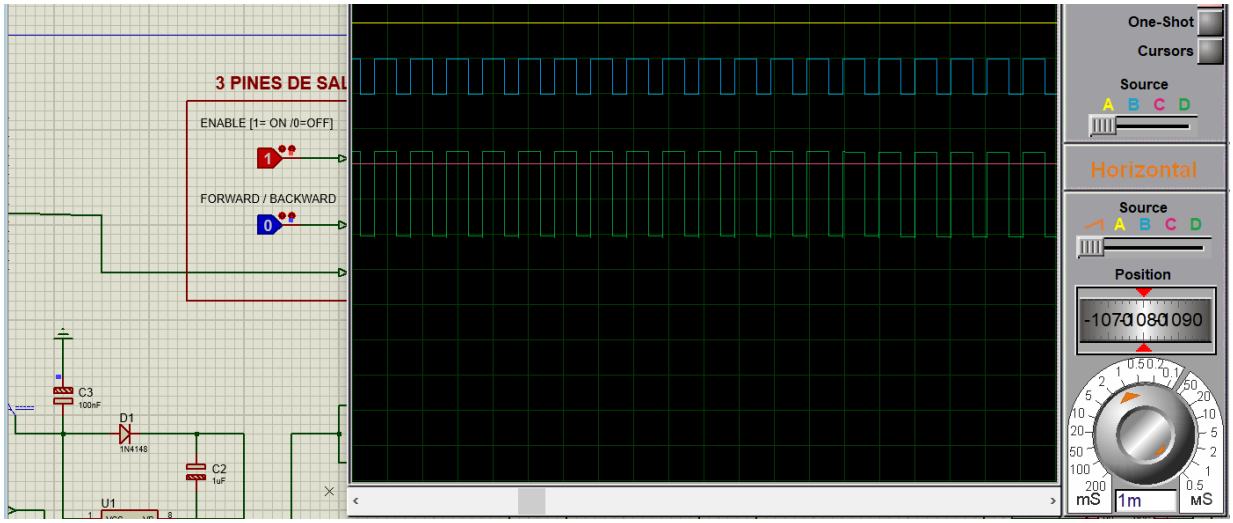
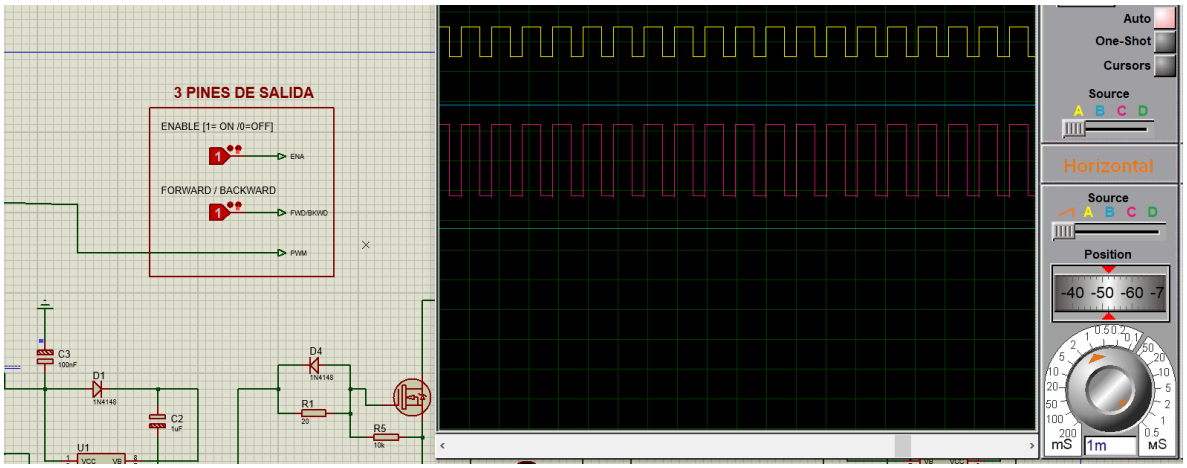


Figura 46

Señales PWM para sentido de giro antihorario



3.1.3.4. Dimensionamiento de la fuente de alimentación

Para dimensionar la fuente de alimentación es necesario conocer la corriente total que requerirá el sistema, considerando todos los componentes electrónicos. Para obtener esta información se usaron las hojas de datos (datasheets) de los componentes del sistema. Esto se presenta en la siguiente tabla:

Tabla 17

Consumo total de corriente del sistema (A)

	Consumo (A)	Cantidad	Total (A)
Arduino Nano	0.019+ (4*0.04)	1	0.179
Driver Motor	0.36	2	0.72
Motor_R	3.2	1	3.2
Motor_T	2	1	2
	Consumo total:		6.099

En base a lo calculado, se necesitaría una fuente de 6.099 A para alimentar todo el sistema. Sin embargo, aún falta considerar otros elementos electrónicos (de bajo consumo) y también un margen de seguridad, que se sugiere sea de 65% con la fuente operando a plena carga (Alburqueque & Rondón, 2019). Por ello se realiza la siguiente operación:

$$X = \frac{1 * 6.099}{0.65} = 9.38A \approx 10\text{A} \quad (35)$$

Considerando el factor de seguridad se obtiene 9.38 A, y se selecciona una fuente conmutada (Switching) con el valor comercial más próximo igual a 10 A.

Figura 47

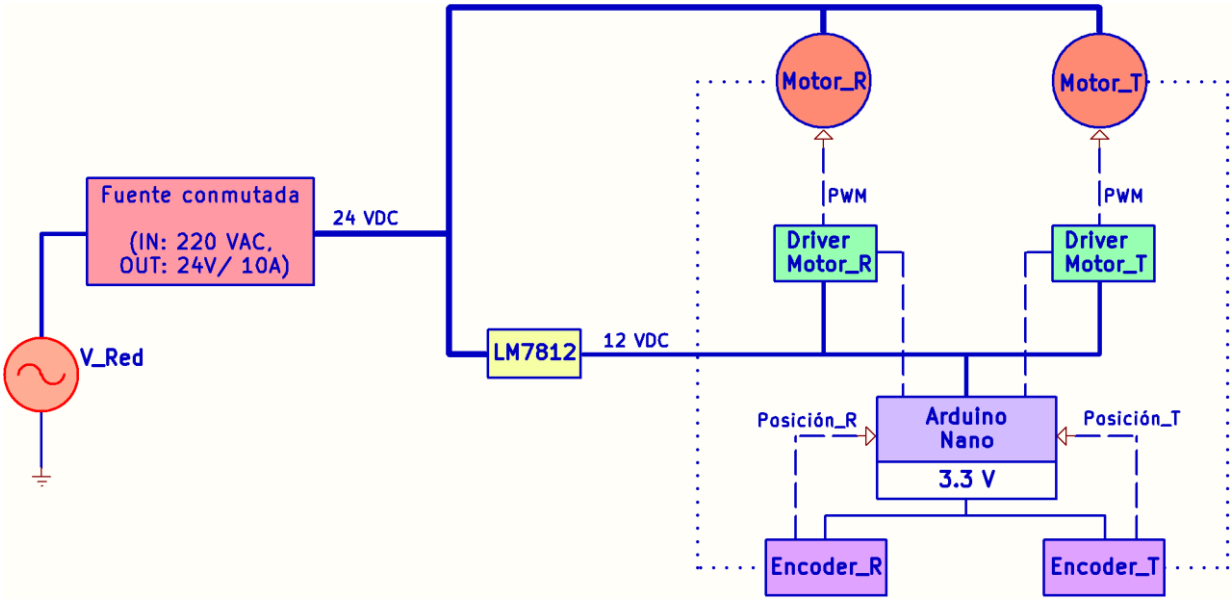
Fuente Switching de 24 V y 10 A

3.1.3.5. Diagrama del circuito electrónico

En la siguiente figura se muestra un esquema donde se ilustra las conexiones y el funcionamiento del subsistema electrónico.

Figura 48

Esquema del subsistema electrónico

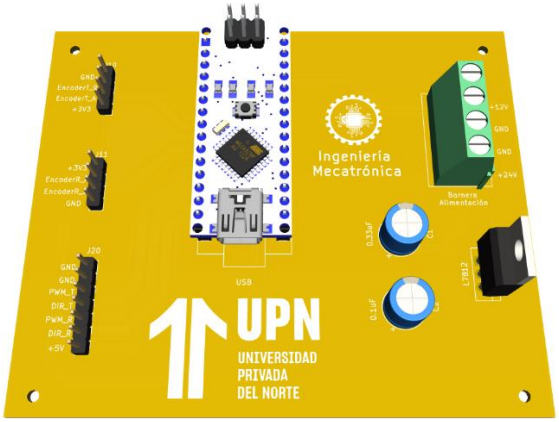


Nota. Los bloques verdes corresponden a los drivers de los motores y tienen su propia placa de circuito impreso (en adelante abreviado como PCB), mostrada en la Figura 44.

Para agrupar al resto de dispositivos se diseñó otra PCB en el software KiCAD, mostrada en la siguiente figura:

Figura 49

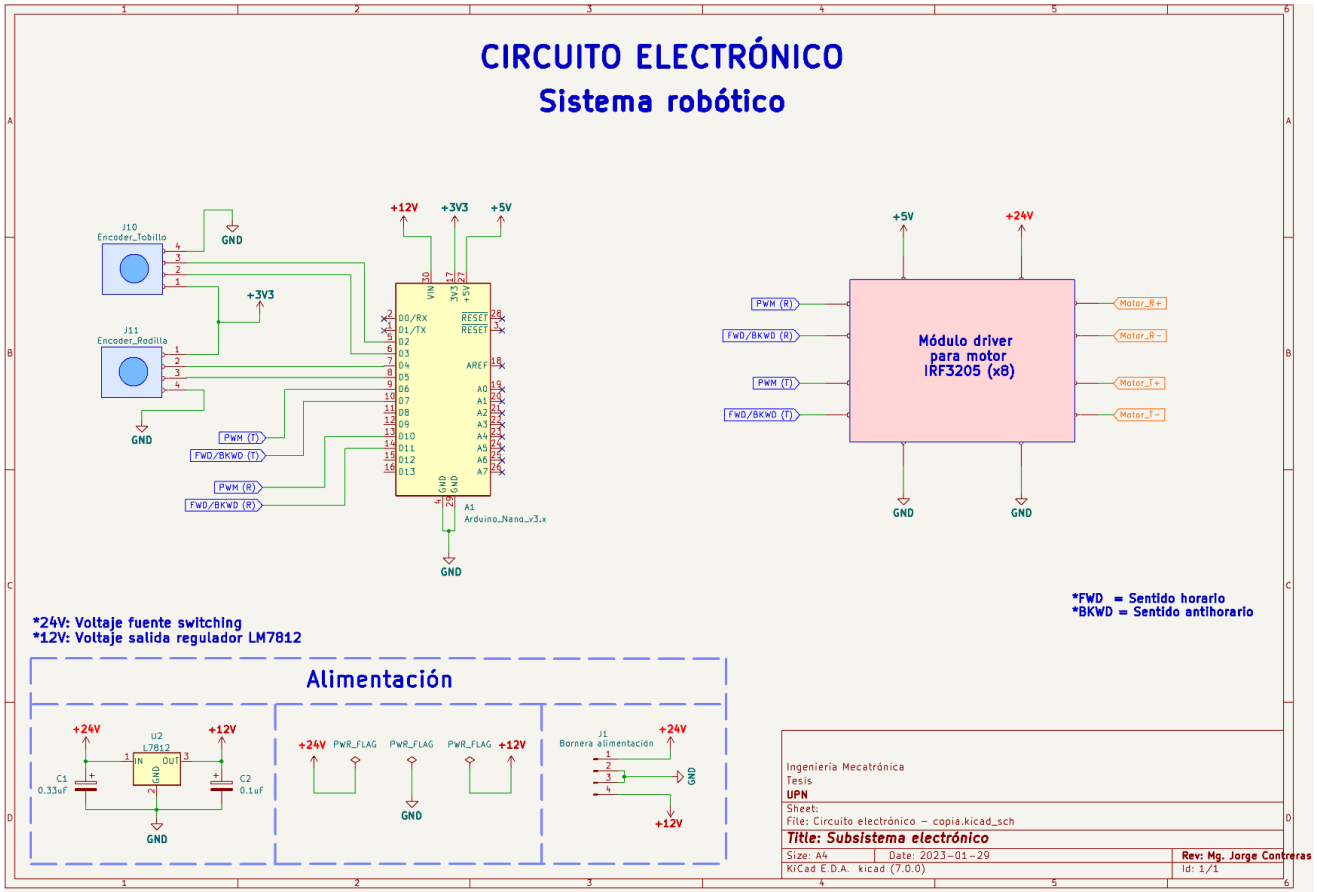
PCB modular para el microcontrolador



El esquema del circuito electrónico correspondiente a esta PCB se muestra en la siguiente figura.

Figura 50

Esquema del circuito electrónico



3.2. Repetibilidad de los ejercicios del tratamiento fisioterapéutico

Para garantizar que los ejercicios se realicen de forma repetible, se definió una entrada de referencia r(t) de forma senoidal con frecuencia y amplitud constante. Se optó por una entrada de forma senoidal debido a que tiene la característica de desacelerar al acercarse a los límites del rango de amplitud, esto daría la sensación de un movimiento más suave al acercarse a los límites de los rangos de movilidad de rodilla y tobillo.

3.2.1 Ecuaciones de la trayectoria

Tal como se describió en la sección 1.2.2, existen unos rangos de movilidad en las articulaciones de rodilla y tobillo, estos rangos se muestran en la siguiente tabla.

Tabla 18

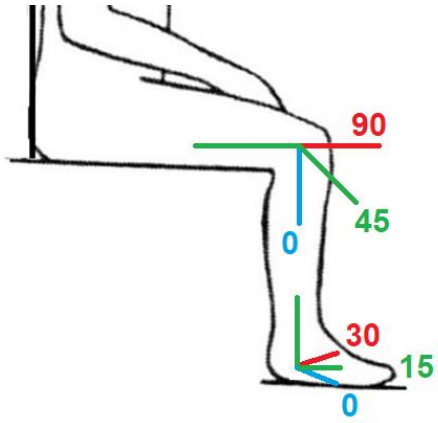
Límites de movilidad de las articulaciones

Articulación	Límite inferior (Flexión / flexión plantar)	Origen (Posición inicial)	Límite superior (Extensión / flexión dorsal)
Rodilla	0°	0°	90°
Tobillo	0°	0°	30°

Se asumió que la posición inicial de los motores es la posición de flexión en rodilla y flexión plantar en tobillo, con un ángulo igual a 0° en ambos. Esto se muestra en la siguiente figura.

Figura 51

Rangos y límites de movilidad de las articulaciones

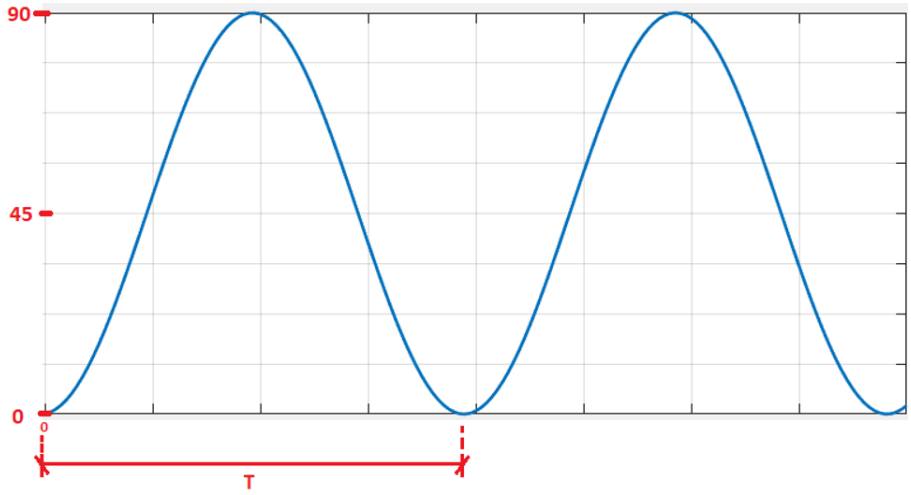


- **Trayectoria para motor $_{R}$**

De acuerdo con la Tabla 18, el motor de la articulación de la rodilla inicia en la posición de flexión con un ángulo igual a 0° y llega hasta la posición de extensión con un ángulo igual a 90°. La trayectoria de forma senoidal sería como se muestra en la siguiente figura.

Figura 52

Trayectoria del motor $_{R}$



La ecuación de la función seno con desfase tiene la siguiente forma general:

$$y(t) = C + A * \sin\left((2 * pi * \frac{1}{T} * t) + B\right) \quad (36)$$

Donde A es la amplitud, B es el desfase y C es el desplazamiento vertical.

Para realizar la trayectoria mostrada en la figura, la ecuación debe ser como se muestra a continuación:

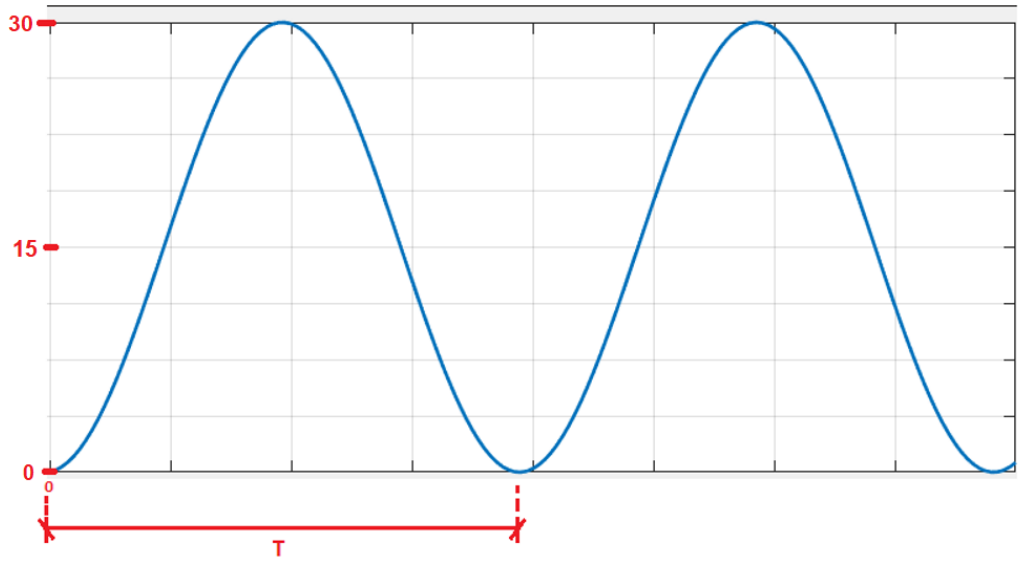
$$\theta_{Ref_R}(t) = 45 + 45 * \sin(2 * \text{pi} * \frac{1}{T} * \text{t} - \text{pi}/2) \quad (37)$$

- **Trayectoria para motor T**

De acuerdo con la Tabla 18, el motor de la articulación del tobillo inicia en la posición de flexión plantar con un ángulo de 0° y llega hasta la posición de flexión dorsal con un ángulo de 30°.

Figura 53

Trayectoria del motor T



Para realizar la trayectoria mostrada en la **Figura 53**

Trayectoria del motor T, la ecuación debe ser como se muestra a continuación:

$$\theta_{Ref_T}(t) = 15 + 15 * \sin(2 * \text{pi} * \frac{1}{T} * \text{t} - \text{pi}/2) \quad (38)$$

3.2.2 Índice de comportamiento

Dentro de la programación del sistema robótico se calculó un índice de comportamiento, el cual es una medida cuantitativa para evaluar la performance del sistema.

Debido a las características del presente sistema robótico, se eligió el índice de comportamiento IAE, ya que evalúa al sistema en la misma medida desde el primer instante hasta el último, a diferencia de otros índices que minimizan la contribución de los errores iniciales como los índices ITAE e ITSE.

> Un índice de comportamiento es una medida cuantitativa del comportamiento de un sistema y se elige de forma que resalte las especificaciones importantes del sistema.
>
> Un sistema se considera sistema de control óptimo cuando sus parámetros se ajustan de forma que el índice alcanza un valor extremo, comúnmente un valor mínimo. Para que un índice de comportamiento resulte útil, siempre debe ser un numero positivo o cero. Entonces el mejor sistema se define como aquel que minimiza este índice…. Otro criterio de comportamiento, aplicable con facilidad, es la integral de la magnitud absoluta del error, IAE la cual se describe como:

$$IAE = \int_0^T |e(t)|\ dt \quad (39)$$

(Dorf & Bishop, 2022, pág. 344)

3.3. Registro de datos del tratamiento fisioterapéutico

Para registrar los datos del tratamiento fisioterapéutico, se implementó una interfaz gráfica de usuario (GUI). Esta interfaz puede guardar los siguientes datos de las sesiones realizadas: ID del paciente, número de repeticiones (flexión-extensión) de cada sesión, duración total de cada sesión, un indicador de las articulaciones en las que se realizaron los ejercicios (rodilla y/o tobillo), y finalmente los índices de comportamiento de cada articulación.

Esta interfaz es un software ejecutable (.exe) que se implementó mediante la herramienta App Designer de Matlab. En esta interfaz se puede configurar la cantidad de repeticiones para cada sesión, duración de cada ciclo de repetición, ingresar el ID o código del paciente, y además permite seleccionar si se realizarán los ejercicios en rodilla, tobillo o ambos en simultáneo. Asimismo, cuenta con botones de marcha (“Start”) y paro (“Stop”).

Figura 54

Interfaz gráfica de usuario con comunicación USB

En la parte inferior de la interfaz gráfica se ubica una tabla donde se registran los datos de cada sesión y los índices de comportamiento de motor $_R$ y motor $_T$, los cuales cuantifican la performance del sistema. Cuanto más se aproxime el error a 0 más pequeño será la magnitud del índice.

Figura 55

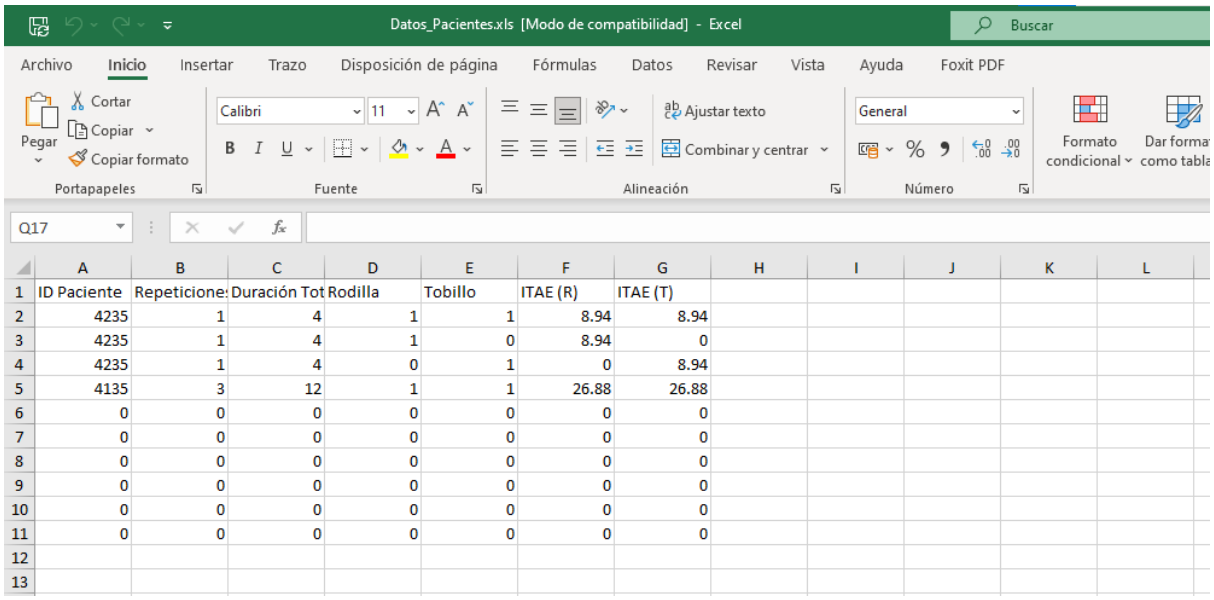
Interfaz gráfica de usuario con simulación

Además, al finalizar cada sesión se muestran las gráficas de las trayectorias vs tiempo, los errores vs tiempo y las velocidades angulares vs tiempo. Finalmente, a través de la pestaña “Opciones” se puede acceder a un documento de ayuda PDF donde se encuentran las instrucciones de uso y la información de contacto para soporte técnico.

Finalmente, debajo de la tabla se ubica un botón que al ser presionado genera un documento Excel con los datos de las sesiones realizadas.

Figura 56

Datos importados a un archivo Excel



	A	B	C	D	E	F	G	H	I	J	K	L
1	ID Paciente	Repeticione:	Duración Tot	Rodilla	Tobillo	ITAE (R)	ITAE (T)					
2	4235	1	4	1	1	8.94	8.94					
3	4235	1	4	1	0	8.94	0					
4	4235	1	4	0	1	0	8.94					
5	4135	3	12	1	1	26.88	26.88					
6	0	0	0	0	0	0	0					
7	0	0	0	0	0	0	0					
8	0	0	0	0	0	0	0					
9	0	0	0	0	0	0	0					
10	0	0	0	0	0	0	0					
11	0	0	0	0	0	0	0					
12												
13												

CAPÍTULO IV. DISCUSIÓN

5.1. Limitaciones

- Poca cantidad de antecedentes con detalles técnicos sobre diseños de tipo estacionario para la rehabilitación de miembros inferiores.
- Existieron algunas limitaciones para realizar pruebas en físico del sistema de control, debido a que no se dispone en el mercado local de motores de velocidades bajas y de encoders de efecto hall de 2 canales. Se tuvo que realizar las pruebas con encoders acoplados a motores de altas velocidades (160 RPM). Sin embargo, en la experimentación si se consiguió seguir una trayectoria de forma senoidal con un error aceptable de acuerdo con el índice IAE.
- En la computadora de la que dispongo no se pudo experimentar con el periodo de muestreo calculado, ya que no es capaz de responder a esa velocidad. Sin embargo, el sistema de control consigue funcionar muy bien con un tiempo de muestro un poco mayor (100 ms), siguiendo la entrada de referencia r(t).

5.2 Interpretación comparativa

La presente investigación se focalizó en el desarrollo de un sistema de control para controlar la posición angular en el tiempo (trayectoria) y de esa forma realizar los ejercicios de flexión- extensión en la rodilla y flexión plantar-flexión dorsal en el tobillo. Mediante el uso de este sistema se puede realizar el tratamiento fisioterapéutico de miembros inferiores durante la etapa inicial e intermedia de rehabilitación. Asimismo, Alburqueque y Rondón (2019) realizaron un proyecto similar en el que desarrollan un sistema mecánico ajustable a la pierna de los pacientes y que realiza fisioterapia mediante frotamiento y presión controlada sobre ligamentos, músculos y tendones para combatir la evolución de la artrosis. Su diseño también es para etapas inicial e intermedia de

rehabilitación.

Además, el subsistema mecánico diseñado posee un sistema de regulación para ajustar el tamaño de la estructura para diferentes pacientes según su estatura, asimismo es ergonómico para garantizar la comodidad del paciente. De la misma forma, Alburqueque y Rondón (2019) desarrollaron un sistema mecánico ajustable para pacientes de distintas características anatómicas.

Por otro lado, el subsistema electrónico diseñado está conformado por los componentes de control y transmisión de datos mediante comunicación serial via puerto USB entre la computadora y el microcontrolador. En contraste, Alburqueque, C. & Rondón, L. (2019) usan un sistema de transmisión inalámbrico (Bluetooth), mediante un módulo HC-05 y una app para smartphone Android.

Finalmente, el subsistema de control diseñado en esta investigación consiste en un sistema de control de doble lazo que consiste en un controlador PID en el lazo interno, el cual se encarga de rechazar perturbaciones, el lazo externo se diseñó mediante los criterios de estabilidad de Lyapunov y se encarga de seguir la trayectoria de referencia r(t). Por otro lado, en la investigación de Mendoza, M. (2021), se realiza la regulación de la presión interna de su actuador neumático mediante el regulador electrónico de presión de vacío, e implementó un controlador de tipo proporcional en un Arduino Mega.

CAPÍTULO V. CONCLUSIONES

- Se consiguió desarrollar un sistema robótico para realizar el tratamiento fisioterapéutico de miembros inferiores. El desarrollo de este sistema robótico se dividió en 4 partes: subsistema de control, subsistema mecánico, subsistema electrónico y subsistema de software.
- Se logró determinar que los ejercicios del tratamiento fisioterapéutico se pueden realizar de forma precisa mediante el desarrollo del sistema de control de doble lazo desarrollado. Asimismo, se demostró que ambos controladores son asintóticamente estables.
- Se logró determinar que los ejercicios del tratamiento fisioterapéutico se pueden realizar de forma repetible, demostrable mediante el índice de comportamiento IAE. Asimismo, se diseñó una entrada de referencia r(t) de forma senoidal, con frecuencia y amplitud constante, y con sus parámetros configurables mediante la interfaz gráfica.
- Se consiguió que el sistema robótico almacene datos del tratamiento fisioterapéutico, ya que la interfaz gráfica desarrollada es capaz de acumular los datos de las sesiones realizadas y luego generar un archivo Excel (.xlsx), al presionar el botón "Exportar a Hoja Excel .xls".

REFERENCIAS

Alburqueque, C. A., & Rondón, L. A. (2019). *Diseño e implementación de un exoesqueleto para fisioterapia en pacientes con artrosis de rodilla en la Clínica Geriátrica Militar de Chorrillos.* Lima: Universidad Ricardo Palma.

Ander-Egg, E. (2015). *Aprender a investigar. Nociones básicas para la investigación social.* Buenos Aires: Editorial Brujas.

Antomás, J., & Huarte del Barrio, S. (2011). Confidencialidad e historia clínica. Consideraciones ético-legales. *Anales del Sistema Sanitario de Navarra*, 73-82.

Araujo, P., Díaz, M., & Mata, V. (2017). Diseño de Dispositivo Robótico para la Rehabilitación y Diagnosis de Extremidades Inferiores. *Diseño de Dispositivos para Rehabilitación y Órtesis*, 15-42.

Arrese, A. (2015). *Sistema mecatrónico para rehabilitación de pacientes con parálisis total o parcial en miembros superiores.* Lima: PUCP.

Arrese, A. (Mayo de 2015). *Sistema mecatrónico para rehabilitación de pacientes con parálisis total o parcial en miembros superiores.* Lima: Pontificia Universidad Católica del Perú.

Artur. (2022). *Terapia de frío*. Obtenido de Terapiadefrio.com: https://www.terapiadefrio.com/blogs/terapia-de-frio-calor-tratamiento-natural-de-lesiones/ejercicios-recuperacion-lesion-rodilla

Bunge, M. (2007). *La investigación científica. Su estrategia y su filosofía.* México D.F.: Siglo XXI Editores, S.A. de C.V.

Campos y Covarrubias, G., & Lule Martínez, N. (2012). La observación, un método para el estudio de la realidad. *Xihmai*, 45-60.

Celedón Flórez, H. J. (2016). *Diseño mecatrónico de un robot exoesqueleto de extremidad superior para rehabilitación de personas con discapacidad parcial en el codo.* Bogotá: Universidad Santo Tomás.

Centro para el Control y la Prevención de Enfermedades. (16 de Septiembre de 2020). *CDC*. Obtenido de Centros para el Control y la Prevención de Enfermedades: https://www.cdc.gov/ncbddd/spanish/disabilityandhealth/relatedconditions.html

Chandler, R., Clauser, C., McConville, J., Reynolds, H., & Young, J. (1975). *Investigation of inertial properties of the human body.* Washington, D.C.: U.S. National Highway Traffic Safety Adminsitration.

De Leva, P. (1996). Adjustments to Zatsiorsky-Seluyanov's Segment Inertia Parameters. *Journal of Biomechanics*, 1223-1230.

Dorf, R. C., & Bishop, R. H. (2022). *Modern Control Systems.* Londres: Pearson Education.

Gabrian International Ltd. (2020). *Gabrian International*. Obtenido de Gabrian International Sitio web: https://www.gabrian.com/es/aluminio-6061-conozca-sus-propiedades-y-usos/

Galán Cutipa, F. (2017). *Diseño, implementación y control de un exoesqueleto para rehabilitación de extremidades superiores.* Piura: Universidad de Piura.

Góngora, L. H., Rosales, C., González, I., & Pujals, N. (2003). Articulación de la rodilla y su mecánica articular. *Laboratorios de Anticuerpos y Biomodelos Experimentales*, 100-109.

Hernández, R., Fernández, C., & Baptista, P. (2014). *Metodología de la investigación.* México D.F.: McGRAW-HILL/ INTERAMERICANA EDITORES S.A. DE C.V.

Hibbeler, R. (2017). *Mecánica de materiales.* Ciudad de México: Pearson Educación.

Hoppenfeld, S., & Murthy, V. L. (2004). *Fracturas: tratamiento y rehabilitación.* Madrid: Marbán Libros , S.L.

Horcajada, R. (s.f.). Miembro inferior: Osteología, miología y artrología. Madrid, España.

Hurtado Floyd, M., Aguilar Zambrano, J., Mora Antó, A., Sandoval Jiménez, C., Peña Solórzano, C., & León Díaz, A. (2012). Identificación de las barreras del entorno que afectan la inclusión social de las personas con discapacidad motriz de miembros inferiores. *Salud Uninorte*, 229.

Hüter-Becker, A., Schewe, H., & Heipertz, W. (2003). *Fisioterapia: descripción de las técnicas y tratamiento.* España: Paidotribo.

Klinker, F. (2011). Exponential moving average versus moving exponential average. *Math Semesterber*, 97-107.

Lynch, K. M., & Park, F. C. (2017). *MODERN ROBOTICS. Mechanics, Planning, and Control.* Cambridge: Cambridge Unirversity Press.

Marizcal Castro, A. d. (2017). *Diseño de un exoesqueleto para terapias de rehabilitación.* Baja California Sur: Tecnológico Nacional de México.

Monteagudo, M., Martínez, P., Maceira, E., & Borja, G. (2016). Anatomía funcional, biomecánica y patomecánica de la estabilidad del tobillo. *Monografías de actualización de la Sociedad Española de Medicina y Cirugía del Pie y Tobillo*, 7-16.

Pinto, E., & Matía, F. (2015). *Fundamentos de control con Matlab.* Madrid: Pearson Educación S.A.

Riccillo, M. (2012). *Robótica: Entrá al mundo de la inteligencia artificial.* Buenos Aires, Buenos Aires, Argentina: Ministerio de Educación de Argentina.

Robledo Mérida, C. (2006). *Técnicas y Proceso de Investigación Científica.* Guatemala: Editora Educativa.

Sosa Mendez, D. (2017). *Desarrollo de un exoesqueleto para rehabilitación del hombro.* Oaxaca: Universidad Tecnológica de la Mixteca.

Texas Instruments. (2018). *Bootstrap Circuitry Selection for Half-Bridge Configurations.* Dallas, Texas: Texas Instruments Incorporated.

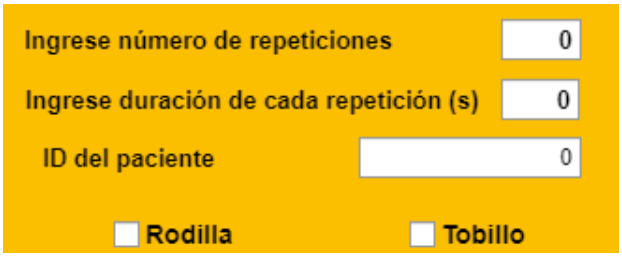
Tzafestas, S. G. (2014). *Introduction to mobile robot control.* Atenas: Elsevier Inc.

ANEXOS

Anexo A: Guía de usuario

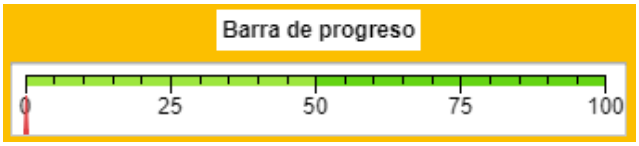
GUIA DE USUARIO

1. Abrir el software Interfaz_Din.exe (doble click en el ícono).
2. Ingrese los parámetros deseados para realizar los ejercicios fisioterapéuticos (Número de repeticiones, ID del paciente y seleccione Rodilla, Tobillo o ambas)

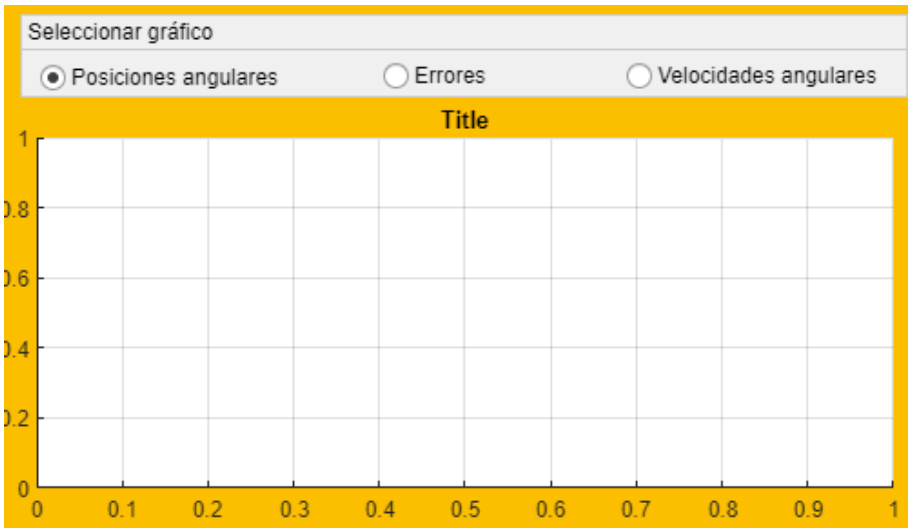


3. Presione el botón "Start" para iniciar la sesión de ejercicios. En caso desee detenerlo presione el botón "Stop". Luego deberá regresar la máquina a su posición inicial antes de volver a iniciar.

4. Puede visualizar el progreso de la sesión de ejercicios en porcentaje (0-100%), en la barra de progreso de la interfaz.



5. Al finalizar la sesión (100%), se mostrarán los gráficos de trayectoria, velocidad y errores en la ventana de gráficos, puede seleccionarlo haciendo click en los botones de la barra de selección de gráfico.



6. También se mostrarán los datos de cada sesión en la parte inferior de la interfaz. Puede exportarlos a un documento Excel haciendo click en el botón “Exportar a hoja Excel .xls”.

Información de contacto

Soporte técnico (Antonio Paucar Quispe)

Correo: paucarquispemtr@gmail.com

Teléfono: 929756221

Anexo B: Programación

- Programación de la dinámica inversa en rodilla y tobillo

```
clear all; close all; clc;
tam=get(0,'ScreenSize');

% C:\Users\pauca\OneDrive\02 Documentos\11 Tesis\Programacion\Matlab
codigo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% TIEMPO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
T = 4;  Repeticiones = 2;

ti = 0; ts =0.1; tf = T*Repeticiones;
t = ti:ts:tf;         % Vector de tiempo
N = length(t);        % # muestras

%%%%%%%%%%%%%%%%%%%%%%%% PARAMETROS DEL ROBOT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Longitud de los eslabones. L1 seria el femur, L2 seria la tibia y L3
%el pie. Las unidades son en metros(m)---------------
L1 = 0.45;
L2 = 0.4; L2 = 0.433; L3 = 0.243;
link00 = 0.25;           %distancia entre piernas

y0 = zeros(1,N); z0 = zeros(1,N);    %cadera
y1 = zeros(1,N); z1 = zeros(1,N);    %rodilla
y2 = zeros(1,N); z2 = zeros(1,N);    %tobillo
y3 = zeros(1,N); z3 = zeros(1,N);    %punta del pie (x,z)

%%%%%%%%%%%%%%%%%%%%%%%% CONDICIONES INICIALES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-Posiciones,velocidades, aceleraciones angulares (Rodilla y tobillo)
theta2 = zeros(1, N);   theta3 = zeros(1, N);
w2= zeros(1,N);     w3= zeros(1,N);
alpha2= zeros(1,N);     alpha3= zeros(1,N);
%--------------------Errores----------------------------
theta2_err= zeros(1,N);     theta3_err= zeros(1,N);
%--------------Flexion-extension en rodilla y tobillo-----------------
---
%Posiciones angulares iniciales
theta2ini = -90;    theta2(1) = theta2ini;
theta3ini = -30;     theta3(1) = theta3ini;

%Coordenadas iniciales
y1 = L1 * ones(1,N);
y2(1) = L1 + L2*cosd(theta2(1));
z2(1) = L2*sind(theta2(1));
y3(1) = L1 + L2*cosd(theta2(1)) + L3*cosd(theta2(1)+theta3(1));
z3(1) = L2*sind(theta2(1)) + L3*sind(theta2(1)+theta3(1));

%Parametros de la funcion seno (trayectoria) para flexion-extension
Aoff2  = 45; offset2 = pi/2;
Aoff3  = 15; offset3 = pi/2;
f = 1/T;
%-------------------------------------------------------------------
---

%------------------Extension en rodilla y tobillo-------------------
-
%---------Posiciones angulares deseadas--------------------------
theta2_d = -Aoff2 + Aoff2*sin(2*pi*f*t -offset2);
theta2_dp = Aoff2*2*pi*f*cos(2*pi*f*t -offset2);
```

```
theta3_d = -Aoff3 + Aoff3*sin(2*pi*f*t -offset3);
theta3_dp = Aoff3*2*pi*f*cos(2*pi*f*t -offset3);

%=======================BUCLE DE SIMULACION=====================

for k=1:N
    %-----------------Calculando errores de posicion angular --------
    theta2_err(k) = theta2_d(k) - theta2(k);
    theta3_err(k) = theta3_d(k) - theta3(k);

    theta_err= [theta2_err(k);...
                theta3_err(k)];       %vector de errores

    %---------------------- Matriz Jacobiana ---------------
    J= [1  0;...
        0  1];

    %-------------------------Parametros de control ---------
    theta23_dp = [theta2_dp(k);...
                  theta3_dp(k)];
    %Ganancias
    K = [0.9 0;...
           0 0.9];

    %------------ Accion/ley de control -------------

    w23 = pinv(J)*(theta23_dp+K*tanh(theta_err));
%     w23 = pinv(J)*(K*tanh(theta_err));

    w2(k) = w23(1);          w3(k) = w23(2);


%Aplicando las acciones de control para realizar la trayectoria

    theta2(k+1) = theta2(k)+ ts*w2(k);
    theta3(k+1) = theta3(k)+ ts*w3(k);

    y2(k+1) = L1 + L2*cosd(theta2(k+1));
    z2(k+1) = L2*sind(theta2(k+1));
    y3(k+1) = L1 +
L2*cosd(theta2(k+1))+L3*cosd(theta2(k+1)+theta3(k+1));
    z3(k+1) = L2*sind(theta2(k+1)) + L3*sind(theta2(k+1)+theta3(k+1));

end

%============================Dinámica
inversa=========================
Mass = zeros(2,2);
Torque2 = zeros(1,N); Torque3 = zeros(1,N);

m2 = 5.375;               %masa del eslabón 1 en kg
m3 = 1.202;                %masa del eslabon 2 + pie en kg
g = 9.81;              %aceleración de la gravedad en m2/s

L2=0.1921;    L3= 0.1027;

%%%%%%%%%%%%%%%%%%%corrige despues el centro de masa

%para ahorrar tiempo de ejecución (constantes)
Mass22 = m3*L3*L3;
CC11 = -m3*L2*L3;
CC21 = m3*L2*L3;


for k=1:N-1
```

```
% %%%==========Derivada aproximada Método de Euler=================
    alpha2(k) = (w2(k+1)-w2(k))/ts;
    alpha3(k) = (w3(k+1)-w3(k))/ts;

    alpha = [alpha2(k);...          %vector de aceleraciones
             alpha3(k)];

%Mass matrix
    Mass11 = m2*L2*L2 + m3*(L2*L2 + 2*L2*L3*cosd(theta3(k)) + L3*L3);
    diagM = m3*(L2*L3*cosd(theta3(k)) + L3*L3);
    Mass12 = diagM;
    Mass21 = diagM;
    Mass = [Mass11    Mass12;...
            Mass21    Mass22];

%Vector c (centripetal y Coriolis)
    CentCor11 = CC11*sind(theta3(k))*(2*w2(k)*w3(k)+w3(k)*w3(k));
    CentCor21 = CC21*w2(k)*w2(k)*sind(theta3(k));
    CentCor = [CentCor11;...
               CentCor21];

%Torque gravitacional
    grav11 = (m2+m3)*L2*g*cosd(theta2(k)) +
m3*g*L3*cosd(theta2(k)+theta3(k));
    grav21 = m3*g*L3*cosd(theta2(k)+theta3(k));
    grav = [grav11;...
            grav21];

%Vector de torques reales de cada eslabón
    Tau = Mass*alpha + CentCor + grav;

%Separando para obtener los torques en rodilla y tobillo
    Torque2(k) = Tau(1);
    Torque3(k) = Tau(2);
end

Torque2(end) = Torque2(end-1);       %un instante antes para no
obtener pico
Torque3(end) = Torque3(end-1);
% Torque2= Torque2*0.8;

%%%
```

```
scene3=figure('Name','Posiciones angulares \theta (°)');
set(scene3,'position',[tam(1) tam(2) tam(3) tam(4)]);
subplot(2,1,1)
H31 = plot(t,theta2(1:N),'r','LineWidth',2); grid on;
xlabel('Tiempo (s)'); ylabel('Posicion angular \theta_2 (°)'); %
Etiquetas
subplot(2,1,2)
H32 = plot(t,theta3(1:N),'b','LineWidth',2); grid on;
xlabel('Tiempo (s)'); ylabel('Posicion angular \theta_3 (°)');
%-------------------------------------------------------------------
----
scene4=figure('Name','Velocidades angulares (rad/s)');
set(scene4,'position',[tam(1) tam(2) tam(3) tam(4)]);
subplot(2,1,1)
H21 = plot(t,w2,'r','LineWidth',2); grid on;
xlabel('Tiempo (s)'); ylabel('Velocidad angular \omega_2 (rad/s)');
subplot(2,1,2)
H22 = plot(t,w3,'b','LineWidth',2); grid on;
xlabel('Tiempo (s)'); ylabel('Velocidad angular \omega_3 (rad/s)');
%-------------------------------------------------------------------
----
scene5=figure('Name','Aceleraciones angulares (rad/s)');
set(scene5,'position',[tam(1) tam(2) tam(3) tam(4)]);
subplot(2,1,1)
H21 = plot(t,alpha2,'r','LineWidth',2); grid on;
xlabel('Tiempo (s)'); ylabel('\alpha_2 (rad/s^{2})');
subplot(2,1,2)
H22 = plot(t,alpha3,'b','LineWidth',2); grid on;
xlabel('Tiempo (s)'); ylabel('\alpha_3 (rad/s^{2})');
%-------------------------------------------------------------------
----
scene6=figure('Name','Torque N.m');
set(scene6,'position',[tam(1) tam(2) tam(3) tam(4)]);
subplot(2,1,1)
H21 = plot(t,Torque2,'r','LineWidth',2); grid on;
xlabel('Tiempo (s)'); ylabel('\tau_2 (N.m)');
subplot(2,1,2)
H22 = plot(t,Torque3,'b','LineWidth',2); grid on;
xlabel('Tiempo (s)'); ylabel('\tau_3 (N.m)');
```

- Programación sistema en lazo abierto (Arduino):

```
////////////////////// Control Lazo Abierto ///////////////
String inputString = "";
bool stringComplete = false;
double inputU;

unsigned long lastTime = 0;
const int ts = 100;               // Sample time

float Y_f = 0.0;
float alpha_f = 0.3;
float S_f = Y_f;

const double  Umax = 255, Umin = 60;
const double  Ymax = 460, Ymin = 0;

//Entradas de puente H y encoder
const byte    in1 = 12;
```

```
const byte    in2 = 11;
const byte    ena = 10;

const byte    C1 = 3; // Entrada de la señal A del encoder (Cable
amarillo).
const byte    C2 = 2; // Entrada de la señal B del encoder (Cable
verde).

//  Variables para las interrupciones (volatiles)

volatile int  count = 0;
volatile byte prev  = 0;
volatile byte act   = 0;

// Señales en la planta (motor)

int u = 0; // Señal de control
double y = 0.0;  // Velocidad angular en rad/s.


const double constValue = 239.0438; //((1000*2*180)/R) ---> R = 1506
Resolucion encoder cuadruple
//double constValue = 238.4106;         // R = 1510 (Izquierdo)

void setup()
{
  Serial.begin(9600);
  pinMode(C1, INPUT);
  pinMode(C2, INPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);

  //Inicializa el motor apagado
  digitalWrite(in1, false);
  digitalWrite(in2, false);
  analogWrite(ena, u);

  ////////////////////////// INTERRUPCIONES ////////////////////
  attachInterrupt(digitalPinToInterrupt(C1), encoder, CHANGE);
  attachInterrupt(digitalPinToInterrupt(C2), encoder, CHANGE);

  lastTime = millis();
}

void loop() {
  //Cuando se envie la señal de control
  if (stringComplete)
  {
    inputU = inputString.toDouble();
    u = normalizeU(inputU);
    if (u>=0)
      {forward(u,in1,in2,ena);}
    else
      {backward(abs(u),in1,in2,ena);}
    inputString = "";
```

```
    stringComplete = false;
  }

  if (millis() - lastTime >= ts)
  {
     calculateY();
     Y_f = normalizeY(y); // Escalar del 0 al 100%
     S_f = (alpha_f*Y_f) + ((1-alpha_f)*S_f);
     Serial.println(S_f);
  }
}

void calculateY(void)
{
  y =(constValue*count)/(millis()-lastTime); // Calculamos velocidad
rad/s
  lastTime = millis(); // Almacenamos el tiempo actual.

  count = 0;  // Reiniciamos los pulsos.
}

void serialEvent()
{
  while (Serial.available())
  {
    char inChar = (char)Serial.read();
    inputString += inChar;
    if (inChar == '\n')
    {
      stringComplete = true;
    }
  }
}

// Encoder precisión cuádruple
void encoder(void)
{
  prev = act;
  act = PIND & 12;

  if(prev==0  && act== 4)  {count+=1;}
  if(prev==4  && act==12)  {count+=1;}
  if(prev==8  && act== 0)  {count+=1;}
  if(prev==12 && act== 8)  {count+=1;}

  if(prev==0 && act==8)  {count-=1;}
  if(prev==4 && act==0)  {count-=1;}
  if(prev==8 && act==12) {count-=1;}
  if(prev==12 && act==4) {count-=1;}
}

// Funciones envio de señal de control u(t)
void forward(int pwm, int pind1, int pind2, int analogPin)
{
  digitalWrite(pind1, LOW);
```

```
  digitalWrite(pind2, HIGH);
  analogWrite(analogPin, pwm);
  }

void backward(int pwm, int pind1, int pind2, int analogPin)
{
  digitalWrite(pind1, HIGH);
  digitalWrite(pind2, LOW);
  analogWrite(analogPin, pwm);
  }


//Funciones de normalización de las variables

double normalizeU(double u)
{
 if (u>= 0) return (u) * (Umax - Umin) / (100.0) + Umin;
 else return (u) * (-Umax + Umin) / (-100.0) - Umin;
}

double normalizeY(double y)
{
 return y*100.0/Ymax;
}
```

- Programación sistema lazo abierto (Matlab):

```
close all; clear; clc;
%Vector de tiempo
ti = 0; tf = 10; ts = 0.1;
t = ti:ts:tf;
N = length(t);

%%%%%%%%%%%%%%%%%% COMUNICACION SERIAL %%%%%%%%%%%%%%%%%%
COM = 'COM4';                    %Seleccionar puerto USB
baudrate = 9600;
delete(instrfind({'Port'},{COM}));
SerialP = serialport(COM,baudrate);              %SerialP is a serial object
configureTerminator(SerialP,'CR/LF');
%Señal de control u(t) y salida y(t)
u = zeros(1,N);
y = zeros(1,N);
disp('Recibiendo datos...');

for k = 1:N
    tic;
   %Entrada escalón
```

```
    if (k*ts) > 3
        u(k) = 10;
    else
        u(k) = 0;
    end

    %Envio de señal de control
    send = sprintf('%s',num2str(round(u(k))));   %enviar u(t) como string
    writeline(SerialP,send);
    %Lectura de señal de salida
    y(k) = readline(SerialP);
    time(k)= toc;
    while toc<ts
    end
end
writeline(SerialP,'0');
clear SerialP;
disp('Finalizado');
%%%%%%%%%%%%%%%%%%%%%%%%% PLOTEO %%%%%%%%%%%%%%%%%%%%%%%
figure(1)
plot(t,u,'r','LineWidth',2);
hold on
plot(t,y,'b','LineWidth',2);
grid on
legend('Señal de control u(t)','Señal de salida
y(t)','Location','southeast')
```

- Programación sistema lazo cerrado (Arduino):

```
#include "controlMotor.h"
#include "PinChangeInterrupt.h"

//////////////////////////// COMUNICACION SERIAL //////////////////
String inputString = "";
bool stringComplete = false;
double inputRef; // Velocidad de referencia (Setpoint)

//////////////////////////// CONTROLADOR PID //////////////////
unsigned long lastTime = 0;
const int ts = 100;  // Tiempo de muestreo
controlMotor motor(ts);
//////////////////////////Filtro EMA//////////////////////////
float Y_f_1 = 0.0;
float Y_f_2 = 0.0;
```

```
float alpha_f = 0.3;
float S_f_1 = Y_f_1;
float S_f_2 = Y_f_2;
////////////////////////// PUENTE H y Encoder //////////////////////
/*
const byte    in1 = 8;
const byte    in2 = 7;
const byte    ena = 6;

const byte    C1 = 3; // Entrada de la señal A del encoder (Cable amarillo).
const byte    C2 = 2; // Entrada de la señal B del encoder (Cable verde).
*/
const byte    in1 = 12;
const byte    in2 = 11;
const byte    ena = 10;

const byte    C1 = 5; // Entrada de la señal A del encoder (Cable amarillo).
const byte    C2 = 4; // Entrada de la señal B del encoder (Cable verde).

//Variables para las interrupciones
volatile int  count = 0;
volatile byte prev  = 0;
volatile byte act   = 0;

////////////////////// Variables Planta (Motor ///////////////

double wMeas = 0.0;      // Velocidad angular en rad/s medido con el encoder.
double wRef = 0.0;       // Velocidad angular de referencia en rad/s. (sp)
int u = 0;   //Señal de control (pwm)
double constValue = 239.0438;             //(1000*2*pi*180)/(R*pi) ---> R =
1506 Resolucion encoder cuadruple (Derecho)
const int R = 1506;
double theta = 0.0;
int memcount = 0;

void setup()
{
  ////////////////// CONFIGURACION PUERTO SERIAL ///////////////
  Serial.begin(9600);

  ////////////////// SINTONIA con PID Tuner de matlab/////////////////

  motor.setGains(1.5, 4.6, 0.18);         //nueva sintonizacion 29/03/23
1.59am
//seteado en 0.2917, 0.11, 0.01 varias pruebas realizadas


  ////////////////// Limites de señales /////////////////
  //motor.setCvLimits(255,26); // Limites de Cv (0-255), considerar zona
muerta  (Izq)
  motor.setCvLimits(255,26);      //(Derecha)
  motor.setPvLimits(900,0);   // Limites de Pv (rad/s)

  ////////////////// CONFIGURACION DE PINES ///////////////
  pinMode(C1, INPUT);
```

```
  pinMode(C2, INPUT);

  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);

  ////////////////////// MOTOR APAGADO ///////////////////////
  digitalWrite(in1, false);
  digitalWrite(in2, false);

  analogWrite(ena,u);

  ////////////////////////// INTERRUPCIONES ////////////////////
  //attachInterrupt(digitalPinToInterrupt(C1), encoder, CHANGE);
  //attachInterrupt(digitalPinToInterrupt(C2), encoder, CHANGE);

  attachPinChangeInterrupt(digitalPinToPinChangeInterrupt(C1), encoder,
CHANGE);
  attachPinChangeInterrupt(digitalPinToPinChangeInterrupt(C2), encoder,
CHANGE);

  lastTime = millis();
}

void loop() {

  ////////// SI RECIBE DATOS /////////////
  if (stringComplete)
  {
    wRef= inputString.toDouble();
    inputString = "";
    stringComplete = false;
  }

  if (millis() - lastTime >= ts)
  {
      computeTheta();
      computeW();             //Velocidad medida mediante el encoder

      u = motor.PIDController(wRef,wMeas);  // Control PID

      if (u>=0)
        {forward(u,in1,in2,ena);}
      else
        {backward(abs(u),in1,in2,ena);}
      Serial.println(wMeas);
      Serial.println(u);
      //Serial.println(theta);
  }
}

void computeW(void)
{
  Y_f_1 =(constValue*count)/(millis()-lastTime); // Calculamos velocidad
rad/s
  S_f_1 = (alpha_f*Y_f_1) + ((1-alpha_f)*S_f_1);
```

```
  wMeas = S_f_1;
  lastTime = millis(); // Almacenamos el tiempo actual.
  count = 0;  // Reiniciamos los pulsos.
}

void computeTheta(void)
{
  memcount = memcount + count;
  theta = (memcount*360.0)/R;
  }


void serialEvent() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    inputString += inChar;
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
}
/*
// Encoder precisión cuádruple
void encoder(void)
{
  prev = act;
  act = PIND & 12;

  if(prev==0  && act== 4)  {count+=1;}
  if(prev==4  && act==12)  {count+=1;}
  if(prev==8  && act== 0)  {count+=1;}
  if(prev==12 && act== 8)  {count+=1;}

  if(prev==0 && act==8)  {count-=1;}
  if(prev==4 && act==0)  {count-=1;}
  if(prev==8 && act==12) {count-=1;}
  if(prev==12 && act==4) {count-=1;}
}
*/
void encoder(void)
{
  prev = act;
  act = PIND & 48;

  if(prev==0  && act== 16)  {count+=1;}
  if(prev==16  && act==48)  {count+=1;}
  if(prev==32  && act== 0)  {count+=1;}
  if(prev==48 && act== 32)  {count+=1;}

  if(prev==0 && act==32)  {count-=1;}
  if(prev==16 && act==0)  {count-=1;}
  if(prev==32 && act==48) {count-=1;}
  if(prev==48 && act==16) {count-=1;}
}
```

```
// Funciones envio de señal de control u(t)
void forward(int cv, int pind1, int pind2, int analogPin)
{
  digitalWrite(pind1, LOW);
  digitalWrite(pind2, HIGH);
  analogWrite(analogPin, cv);
  }

void backward(int cv, int pind1, int pind2, int analogPin)
{
  digitalWrite(pind1, HIGH);
  digitalWrite(pind2, LOW);
  analogWrite(analogPin, cv);
 }
```

- Programación sistema lazo cerrado (Matlab):

```
clc; clear; close all;

%Vector de tiempo
ti = 0;
tf = 15;
ts = 0.1;
t = ti:ts:tf;
N = length(t);

wRef = zeros(1,N);
u = zeros(1,N);
wMeas= zeros(1,N);

%----------------------------Setpoint r(t)--------------------------
wRef = 4*sin(0.5*t)+5;

% for k = 1:N
%       %ENTRADA ESCALON
%
%     if (k*ts) > 3
%         wRef(k) = -6;
%     else
%         wRef(k) = 0;
%     end
% end


%%%%%%%%%%%%%%%%% COMUNICACION SERIAL %%%%%%%%%%%%%%%%%
COM = 'COM4';                    %Seleccionar puerto USB
baudrate = 9600;
delete(instrfind({'Port'},{COM}));
SerialP = serialport(COM,baudrate);              %SerialP es un "serial
object"
```

```
% configureTerminator(SerialP,'CR/LF');
disp('Recibiendo datos...');
tiempo = zeros(1,N);
for k = 1:N
    tic;
    %--------------Envio de setpoint wRef(rad/s)------------
    send = sprintf('%s',num2str(wRef(k)));   %enviar cv como string
    writeline(SerialP,send);

    %Lectura de wMeas(t )(rad/s) y u(t)
    wMeas(k) = str2double(readline(SerialP));
    u(k) = str2double(readline(SerialP));
    tiempo(k) = toc;
    while toc<ts
    end
end

writeline(SerialP,'0');
pause(0.2);
writeline(SerialP,'0');
clear SerialP;
disp('Finalizando');

%%%%%%%%%%%%%%%%%%%%%%%%% PLOTEO %%%%%%%%%%%%%%%%%%%%%%

figure(1);
plot(t,wRef,'b','LineWidth',2);grid on;
hold on;
axis([0 tf 0 10]);
plot(t,wMeas,'r','LineWidth',2);
legend('Setpoint wRef(t)','Señal de salida
wMeas(t)','Location','southeast');

figure(2);
plot(t,u,'g','LineWidth',2); grid on;
legend('Señal de control u(t)','Location','southeast');
```

- Programación en Matlab para calcular ubicación de polos del lazo primario:

```
clear; close all; clc;

Gp_T = tf(2.0695,[0.4323 1],'inputdelay',0.2287);    %motor_T
Gp_R = tf(2.0656,[0.4060 1],'inputdelay',0.1709);      %motor_R
%%%Pade aproximacion
Gp_T = pade(Gp_T,1);
Gp_R = pade(Gp_R,1);

s = tf('s');
```

```
Gc_T = 1.5 + (4.6/s) + 0.18*s;
Gc_R = 1.8 + (4.6/s) + 0.18*s;

T_T = Gc_T*Gp_T / (1 + Gc_T*Gp_T);
T_R = Gc_R*Gp_R / (1 + Gc_R*Gp_R);

%polos
P_T = pole(T_T)
P_R = pole(T_R)

%ploteo
figure(1);
rlocus(T_T);

figure(2);
rlocus(T_R);
```

- Programación Interfaz gráfica del sistema, con animación del sistema robótico en movimiento (App designer):

```
properties (Access = private)
        pausar
        tiempo          %Tiempo transcurrido
        N
        contador
        datos
        memo
        z

        t
        theta_R
        theta_T
        w_R
        w_T
        w_R_Meas
        w_T_Meas

        theta_R_d
        theta_T_d

        ErrorTheta_R
        ErrorTheta_T
    end
```

```
function startupFcn(app)

            app.pausar = false;
            app.UITable.ColumnFormat = {'char', 'numeric', 'numeric',
'logical', 'logical', 'numeric', 'numeric'};
            app.UITable.Data = cell(5,7);
```

```
            app.memo = zeros(7,20);
            app.contador = 1;
            app.z = 1;
        end
% Button pushed function: Boton_Start
        function Boton_StartButtonPushed(app, event)
%             La magia está aqui SIUUUUUUUUUUUUUUUUU
            flag_start = 1;
            app.pausar = false;

            cla(app.Axes_Simulacion,"reset")
            cla(app.Axes_Senales1,"reset")
            cla(app.Axes_Senales2,"reset")


            while app.contador <20 && flag_start ==1
                T = app.Periodo.Value;
                tf = app.NroRepeticiones.Value * T;

                ti = 0; ts = 0.1;
                app.t = ti:ts:tf;
                app.N = length(app.t);

                L1 = 0.4; L2 = 0.55; L3 = 0.30;
                y0 = zeros(1,app.N); z0 = zeros(1,app.N);   %cadera
                y1 = zeros(1,app.N); z1 = zeros(1,app.N);   %rodilla
                y2 = zeros(1,app.N); z2 = zeros(1,app.N);   %tobillo
                y3 = zeros(1,app.N); z3 = zeros(1,app.N);   %punta del pie
(x,z)

                %%%%%%%%%%%%%%%%%%%%%%%%% CONDICIONES INICIALES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %----------------------Para el control-----------------------
                %-Posiciones,velocidades, aceleraciones angulares (Rodilla y
tobillo)
                app.theta_R = zeros(1, app.N);      app.theta_T = zeros(1,
app.N);
                app.w_R = zeros(1,app.N);           app.w_T =
zeros(1,app.N);
                app.w_R_Meas = zeros(1,app.N);      app.w_T_Meas =
zeros(1,app.N);
                %--------------------Errores-----------------------------
                app.ErrorTheta_R= zeros(1,app.N);     app.ErrorTheta_T=
zeros(1,app.N);

                %----------------------Para la animacion----------------------
_
                %--------------Flexion-extension en rodilla y tobillo-------
-------------
                %--------------Posiciones angulares iniciales
                theta_Rini = -90;    app.theta_R(1) = theta_Rini;
                theta_Tini = 45;     app.theta_T(1) = theta_Tini;
                %--------------Posicion inicial del sistema
                y1 = L1 * ones(1,app.N);
                y2(1) = L1 + L2*cosd(app.theta_R(1));
```

```
                z2(1) = L2*sind(app.theta_R(1));
                y3(1) = L1 + L2*cosd(app.theta_R(1)) +
L3*cosd(app.theta_R(1)+app.theta_T(1));
                z3(1) = L2*sind(app.theta_R(1)) +
L3*sind(app.theta_R(1)+app.theta_T(1));
                %Parametros de la funcion seno (trayectoria) para flexion-
extension
                Aoff2  = 45; offset2 = pi/2;
                Aoff3  = 45; offset3 = pi/2;
                f = 1/T;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %-------------------------------------------------------
-------------
                hold(app.Axes_Simulacion,'on')
                grid(app.Axes_Simulacion,'on')
                M01L =
plot(app.Axes_Simulacion,[0,y1(1)],[0,0],'r','LineWidth',5);
                M12L =
plot(app.Axes_Simulacion,[y1(1),y2(1)],[0,z2(1)],'g','LineWidth',5);
                M23L =
plot(app.Axes_Simulacion,[y2(1),y3(1)],[z2(1),z3(1)],'b','LineWidth',5);
                axis(app.Axes_Simulacion,[0 1.5 -1 0.25]);

                %-------------------Extension en rodilla y tobillo----------
-----------
                %---------Posiciones angulares deseadas---------------------
------
                if (app.RodillaCheckBox.Value == 1) &&
(app.TobilloCheckBox.Value == 1)

                    app.theta_R_d = -Aoff2 + Aoff2*sin(2*pi*f*app.t -
offset2);
                    theta_R_dp = Aoff2*2*pi*f*cos(2*pi*f*app.t -offset2);

                    app.theta_T_d = 2*Aoff3 + Aoff3*sin(2*pi*f*app.t -
offset3);
                    theta_T_dp = Aoff3*2*pi*f*cos(2*pi*f*app.t -offset3);

                elseif (app.RodillaCheckBox.Value == 1) &&
(app.TobilloCheckBox.Value == 0)

                    app.theta_R_d = -Aoff2 + Aoff2*sin(2*pi*f*app.t -
offset2);
                    theta_R_dp = Aoff2*2*pi*f*cos(2*pi*f*app.t -offset2);

                    app.theta_T_d = theta_Tini * ones(1,app.N);
                    theta_T_dp = zeros(1,app.N);

                elseif (app.RodillaCheckBox.Value == 0) &&
(app.TobilloCheckBox.Value == 1)

                    app.theta_R_d = theta_Rini * ones(1,app.N);
                    theta_R_dp = zeros(1, app.N);
```

```
                    app.theta_T_d = 2*Aoff3 + Aoff3*sin(2*pi*f*app.t -
offset3);
                    theta_T_dp = Aoff3*2*pi*f*cos(2*pi*f*app.t -offset3);
                end

                %%%%%%%%%%%%%%%%%%%%%%%%%% BUCLE DE CONTROL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                k = 1;
                while k < app.N

                   if app.pausar == true
                      break
                   else
                   end
                   %-----------------Calculando errores de posicion angular
--------
                    app.ErrorTheta_R(k) = app.theta_R_d(k) - app.theta_R(k);
                    app.ErrorTheta_T(k) = app.theta_T_d(k) - app.theta_T(k);

                    theta_err= [app.ErrorTheta_R(k);...
                            app.ErrorTheta_T(k)];       %vector de errores

                   %---------------------- Matriz Jacobiana ---------------
                    J= [1  0;...
                        0  1];

                   %-------------------------Parametros de control ---------
                    thetaRT_dp = [theta_R_dp(k);...
                    theta_T_dp(k)];
                   %Ganancias
                    K = [0.5 0;...
                        0 0.5];

                   %------------ Accion/ley de control -------------
                    wRT = pinv(J)*(thetaRT_dp+K*tanh(theta_err));
                    app.w_R(k) = wRT(1);          app.w_T(k) = wRT(2);


                    %Aplicando las acciones de control para realizar la
trayectoria

                    app.theta_R(k+1) = app.theta_R(k)+ ts*app.w_R(k);
                    app.theta_T(k+1) = app.theta_T(k)+ ts*app.w_T(k);

                    y2(k+1) = L1 + L2*cosd(app.theta_R(k+1));
                    z2(k+1) = L2*sind(app.theta_R(k+1));
                    y3(k+1) = L1 +
L2*cosd(app.theta_R(k+1))+L3*cosd(app.theta_R(k+1)+app.theta_T(k+1));
                    z3(k+1) = L2*sind(app.theta_R(k+1)) +
L3*sind(app.theta_R(k+1)+app.theta_T(k+1));

                    app.Barra_Progreso.Value = (k/app.N)*100;

                    delete(M12L);  delete(M23L);
```

```
                    M12L =
plot(app.Axes_Simulacion,[y1(k),y2(k)],[0,z2(k)],'g','LineWidth',5);
                    M23L =
plot(app.Axes_Simulacion,[y2(k),y3(k)],[z2(k),z3(k)],'b','LineWidth',5);

                    %----------------w_Meas----------------
                    app.w_R_Meas(k+1) = (app.theta_R(k+1) -
app.theta_R(k))/ts;
                    app.w_T_Meas(k+1) = (app.theta_T(k+1) -
app.theta_T(k))/ts;
                    k = k+1;

                    pause(ts);
                end

                %---------------Indices de performance---------------------
                IAE_R = 0;
                IAE_T = 0;
                for s=1:app.N
                    IAE_R = IAE_R + abs(app.ErrorTheta_R(s))*ts;
                    IAE_T = IAE_T + abs(app.ErrorTheta_T(s))*ts;
                end

                IAE_R = IAE_R / app.NroRepeticiones.Value;
                IAE_T = IAE_T / app.NroRepeticiones.Value;

            %-------------------------------------------------------------
            %%

                if (app.RodillaCheckBox.Value == 0) &&
(app.TobilloCheckBox.Value == 1)
                    axis(app.Axes_Senales1,'off')

                    hold(app.Axes_Senales2,'on')
                    grid(app.Axes_Senales2,'on')

                    app.Axes_Senales2.Title.String = 'Posiciones angulares
\theta_T (°)';

plot(app.Axes_Senales2,app.t,app.theta_T_d,'b','LineWidth',2);

plot(app.Axes_Senales2,app.t,app.theta_T,'r','LineWidth',2);
                    xlabel(app.Axes_Senales2,'Tiempo (s)');
ylabel(app.Axes_Senales2,'Posicion angular \theta_T (°)');
                    legend(app.Axes_Senales2,'Posicion Ref','Posicion
Meas');

                elseif (app.RodillaCheckBox.Value == 1) &&
(app.TobilloCheckBox.Value == 0)
                    axis(app.Axes_Senales2,'off')

                    hold(app.Axes_Senales1,'on')
                    grid(app.Axes_Senales1,'on')
```

```
                    app.Axes_Senales1.Title.String = 'Posiciones angulares
\theta_R (°)';

plot(app.Axes_Senales1,app.t,app.theta_R_d,'b','LineWidth',2);

plot(app.Axes_Senales1,app.t,app.theta_R,'r','LineWidth',2);
                    xlabel(app.Axes_Senales1,'Tiempo (s)');
ylabel(app.Axes_Senales1,'Posicion angular \theta_R (°)');
                    legend(app.Axes_Senales1,'Posicion Ref','Posicion
Meas');

                else
                    hold(app.Axes_Senales1,'on')
                    hold(app.Axes_Senales2,'on')
                    grid(app.Axes_Senales1,'on')
                    grid(app.Axes_Senales2,'on')

                    app.Axes_Senales1.Title.String = 'Posiciones angulares
\theta_R (°)';

plot(app.Axes_Senales1,app.t,app.theta_R_d,'b','LineWidth',2);

plot(app.Axes_Senales1,app.t,app.theta_R,'r','LineWidth',2);
                    xlabel(app.Axes_Senales1,'Tiempo (s)');
ylabel(app.Axes_Senales1,'Posicion angular \theta_R (°)');
                    legend(app.Axes_Senales1,'Posicion Ref','Posicion
Meas');

                    app.Axes_Senales2.Title.String = 'Posiciones angulares
\theta_T (°)';

plot(app.Axes_Senales2,app.t,app.theta_T_d,'b','LineWidth',2);

plot(app.Axes_Senales2,app.t,app.theta_T,'r','LineWidth',2);
                    xlabel(app.Axes_Senales2,'Tiempo (s)');
ylabel(app.Axes_Senales2,'Posicion angular \theta_T (°)');
                    legend(app.Axes_Senales2,'Posicion Ref','Posicion
Meas');

                end

                Codigo_p(app.contador) = app.CodigoPaciente.Value;
                Repeticiones(app.contador) = app.NroRepeticiones.Value;
%                 time(app.contador) = tf;
                Rodilla(app.contador) = app.RodillaCheckBox.Value;
                Tobillo(app.contador) = app.TobilloCheckBox.Value;

                if IAE_R < 7 && IAE_R > 0
                    IAE_R = 9999;
                else
                    IAE_R = round(IAE_R, 2);
                end

                if IAE_T < 7 && IAE_T > 0
                    IAE_T = 9999;
```

```
                else
                    IAE_T = round(IAE_T, 2);
                end

                if app.z == app.contador
                    app.memo(:,app.z) = [Codigo_p(app.z);
Repeticiones(app.z); tf; Rodilla(app.z); Tobillo(app.z); IAE_R ; IAE_T];
                end

                app.contador = app.contador +1;
                flag_start = 0;
            end

                app.datos = app.memo;
                app.UITable.Data = app.datos';
                app.z = app.z+1;
        end
```

```
% Selection changed function: SeleccionarGrafico
        function SeleccionarGraficoSelectionChanged(app, event)
%             selectedButton = app.SeleccionarGrafico.SelectedObject;
            R1 = app.Posiciones_Button.Value;
            R2 = app.Errores_Button.Value;
            R3 = app.Velocidades_Button.Value;


            if R1 == 1
                cla(app.Axes_Senales1,"reset")
                cla(app.Axes_Senales2,"reset")
                hold(app.Axes_Senales1,'on')
                hold(app.Axes_Senales2,'on')
                grid(app.Axes_Senales1,'on')
                grid(app.Axes_Senales2,'on')

                app.Axes_Senales1.Title.String = 'Posiciones angulares
\theta_R (°)';

plot(app.Axes_Senales1,app.t,app.theta_R_d,'b','LineWidth',2);
                plot(app.Axes_Senales1,app.t,app.theta_R,'r','LineWidth',2);
                xlabel(app.Axes_Senales1,'Tiempo (s)');
ylabel(app.Axes_Senales1,'Posicion angular \theta_R (°)')
                legend(app.Axes_Senales1,'Posicion Ref','Posicion Meas');

                app.Axes_Senales2.Title.String = 'Posiciones angulares
\theta_T (°)';

plot(app.Axes_Senales2,app.t,app.theta_T_d,'b','LineWidth',2);
                plot(app.Axes_Senales2,app.t,app.theta_T,'r','LineWidth',2);
                xlabel(app.Axes_Senales2,'Tiempo (s)');
ylabel(app.Axes_Senales2,'Posicion angular \theta_T (°)')
                legend(app.Axes_Senales2,'Posicion Ref','Posicion Meas');

            elseif R2 ==1
                cla(app.Axes_Senales1,"reset")
                cla(app.Axes_Senales2,"reset")
```

```
                hold(app.Axes_Senales1,'on')
                hold(app.Axes_Senales2,'on')
                grid(app.Axes_Senales1,'on')
                grid(app.Axes_Senales2,'on')

                app.Axes_Senales1.Title.String = 'Gráfica de errores
\theta_R';

plot(app.Axes_Senales1,app.t,app.ErrorTheta_R,'r','LineWidth',2);
                xlabel(app.Axes_Senales1,'Tiempo (s)');
ylabel(app.Axes_Senales1,'Error en \theta_R (°)');

                app.Axes_Senales2.Title.String = 'Gráfica de errores
\theta_T';

plot(app.Axes_Senales2,app.t,app.ErrorTheta_T,'r','LineWidth',2);
                xlabel(app.Axes_Senales2,'Tiempo (s)');
ylabel(app.Axes_Senales2,'Error en \theta_T (°)');


            elseif R3 == 1
                cla(app.Axes_Senales1,"reset")
                cla(app.Axes_Senales2,"reset")
                hold(app.Axes_Senales1,'on')
                hold(app.Axes_Senales2,'on')
                grid(app.Axes_Senales1,'on')
                grid(app.Axes_Senales2,'on')

                app.Axes_Senales1.Title.String = 'Velocidades angulares w_R
(deg/s)';
                plot(app.Axes_Senales1,app.t,app.w_R,'b','LineWidth',2);

plot(app.Axes_Senales1,app.t,app.w_R_Meas,'r','LineWidth',2);
                xlabel(app.Axes_Senales1,'Tiempo (s)');
ylabel(app.Axes_Senales1,'Velocidad angular \omega_R (rad/s)');
                legend(app.Axes_Senales1,'Velocidad Ref','Velocidad Meas');

                app.Axes_Senales2.Title.String = 'Velocidades angulares w_T
(deg/s)';
                plot(app.Axes_Senales2,app.t,app.w_T,'b','LineWidth',2);

plot(app.Axes_Senales2,app.t,app.w_T_Meas,'r','LineWidth',2);
                xlabel(app.Axes_Senales2,'Tiempo (s)');
ylabel(app.Axes_Senales2,'Velocidad angular \omega_T (rad/s)');
                legend(app.Axes_Senales2,'Velocidad Ref','Velocidad Meas');
            end

            if (app.RodillaCheckBox.Value == 0) &&
(app.TobilloCheckBox.Value == 1)
                cla(app.Axes_Senales1,"reset")
                axis(app.Axes_Senales1,'off')

            elseif (app.RodillaCheckBox.Value == 1) &&
(app.TobilloCheckBox.Value == 0)
                cla(app.Axes_Senales2,"reset")
```

```
                axis(app.Axes_Senales2,'off')
            end
        end
```

```
% Button pushed function: Boton_Pausa
        function Boton_PausaButtonPushed(app, event)
            app.pausar = true;
        end
```

```
% Button pushed function: Exportar
        function ExportarButtonPushed(app, event)
            data = get(app.UITable, 'data');
            datax = get(app.UITable, 'ColumnName');
            filename = 'Datos_Pacientes.xls';
            xlswrite(filename,datax');
            limpiar = zeros(10,7);
            xlswrite(filename,limpiar,1,'A2:G11');
            pause(0.1);
            xlswrite(filename,data,1,'A2:G11');
        end
```

```
% Menu selected function: AyudaMenu
        function AyudaMenuSelected(app, event)
            winopen('Manual.pdf');
        end
```

```
% Menu selected function: CerrarMenu
        function CerrarMenuSelected(app, event)
            close(app.UIFigure);
        end
```

- Programación para hacer comunicación serial con la GUI de Matlab App Designer (Arduino):

```
#include "PinChangeInterrupt.h"
#include "controlMotor.h"
////////////////////////////////// COMUNICACION SERIAL
//////////////////////////
String inputString = "";
bool stringComplete = false;
const char separator = ',';
const int dataLength = 2;
double data[dataLength];

////////////////////////////////// CONTROLADOR PID ////////////////////////////
unsigned long lastTime = 0;
const int ts = 100;  //Tiempo de muestreo
controlMotor motorR(ts);
controlMotor motorT(ts);
```

```
////////////////////////////////Filtro EMA//////////////////////////////
float Y_f_R = 0.0;
float Y_f_T = 0.0;
float alpha_f = 0.30;
float S_f_R = Y_f_R;
float S_f_T = Y_f_T;
//////////////////////////// PUENTE H y Encoder ///////////////////////
const byte    in1_R = 12;
const byte    in2_R = 11;
const byte    ena_R = 10;
const byte    C1_R = 5; //encoder (Cable amarillo)
const byte    C2_R = 4; //encoder (Cable verde)

const byte    in1_T = 8;
const byte    in2_T = 7;
const byte    ena_T = 6;
const byte    C1_T = 3; //encoder (Cable amarillo)
const byte    C2_T = 2; //encoder (Cable verde)

//////////////////////////// Variables de las INTERRUPCIONES
//////////////////////
volatile int  count_R = 0;
volatile byte prev_R  = 0;
volatile byte act_R   = 0;

volatile int  count_T = 0;
volatile byte prev_T  = 0;
volatile byte act_T   = 0;

////////////////////////// VARIABLES DE CONTROL //////////////////////
double wRef_R  = 0;       // Velocidad angular de referencia en deg/s. (sp)
double wMeas_R = 0;      // Velocidad angular en deg/s medido con el encoder.
int u_R = 0;

double wRef_T = 0;
double wMeas_T = 0;
int u_T = 0;

const double R = 1506.0;
double constValue = 239.0438; //((1000*2*180)/R) ---> R = 1506 Resolucion
encoder cuadruple(Der)
//const int R_T = 1506;
//double constValue_T = 239.0438; //((1000*2*180)/R) ---> R = 1510
Resolucion encoder cuadruple(Izq)

double thetaMeas_R = 0;
int memcount_R = 0;
double thetaMeas_T = 0;
int memcount_T = 0;
```

```
void setup()
{
  ////////////////////// CONFIGURACION PUERTO SERIAL ////////////////
```

```
  Serial.begin(9600);

  ////////////////// SINTONIA con PID Tuner de matlab/////////////////
  //motor.setGains(kp,   ki,  kd);
  motorR.setGains(1.5, 4.6, 0.18);         //nueva sintonizacion 30/07/22
1.59am
  motorT.setGains(1.8, 4.6, 0.18);
//seteado en 2.1, 4.5, 0.28 varias pruebas realizadas. K= [0.8,0.8] <--
lyapunov

  ////////////////// Limites de señales /////////////////
  motorR.setCvLimits(255,26); // Limites de Cv (0-255), considerar zona
muerta
  motorR.setPvLimits(900,0);   // Limites de Pv (deg/s)

  motorT.setCvLimits(255,40); // Limites de Cv (0-255), considerar zona
muerta
  motorT.setPvLimits(900,0);   // Limites de Pv (deg/s)

  ////////////////// CONFIGURACION DE PINES ///////////////
  pinMode(C1_R, INPUT);
  pinMode(C2_R, INPUT);
  pinMode(C1_T, INPUT);
  pinMode(C2_T, INPUT);

  pinMode(in1_R, OUTPUT);
  pinMode(in2_R, OUTPUT);
  pinMode(in1_T, OUTPUT);
  pinMode(in2_T, OUTPUT);

  ////////////////// MOTOR APAGADO /////////////////////
  digitalWrite(in1_R, false);
  digitalWrite(in2_R, false);
  digitalWrite(in1_T, false);
  digitalWrite(in2_T, false);

  analogWrite(ena_R,u_R);
  analogWrite(ena_T,u_T);

  ////////////////////////// INTERRUPCIONES ////////////////////
  attachPinChangeInterrupt(digitalPinToPinChangeInterrupt(C1_R), encoderR,
CHANGE);
  attachPinChangeInterrupt(digitalPinToPinChangeInterrupt(C2_R), encoderR,
CHANGE);
  attachInterrupt(digitalPinToInterrupt(C1_T), encoderT, CHANGE);
  attachInterrupt(digitalPinToInterrupt(C2_T), encoderT, CHANGE);

  lastTime = millis();
}
```

```
void loop() {

  ////////// SI RECIBE DATOS ////////////
  if (stringComplete)
  {
```

```
    for (int i = 0; i<dataLength ; i++)
    {
      int index_separator = inputString.indexOf(separator);
      data[i] = inputString.substring(0, index_separator).toFloat();
      inputString = inputString.substring(index_separator + 1);
     }

     wRef_R = data[0];
     wRef_T = data[1];

     inputString = "";
     stringComplete = false;
  }

  if (millis() - lastTime >= ts)
  {
      Y_f_R = (constValue * count_R) / (millis() - lastTime); // Calculamos
velocidad deg/s y la filtramos
      S_f_R = ((1 - alpha_f) * S_f_R) + (alpha_f * Y_f_R);
      wMeas_R = S_f_R;

      Y_f_T = (constValue * count_T) / (millis() - lastTime); // Calculamos
velocidad deg/s y la filtramos
      S_f_T = ((1 - alpha_f) * S_f_T) + (alpha_f * Y_f_T);
      wMeas_T = S_f_T;

      lastTime = millis();
      memcount_R = memcount_R + count_R;
      thetaMeas_R = (memcount_R * 360.0) / R;
      memcount_T = memcount_T + count_T;
      thetaMeas_T = (memcount_T * 360.0) / R;

      if (thetaMeas_R >400) {thetaMeas_R = 400;};     //saturando
      if (thetaMeas_T >400) {thetaMeas_T = 400;};
      if (thetaMeas_R <0) {thetaMeas_R = 0;};
      if (thetaMeas_T <0) {thetaMeas_T = 0;};

      count_R = 0;
      count_T = 0;

      u_R = motorR.PIDController(wRef_R,wMeas_R);  // Control PID, hallar
u(k)
      u_T = motorT.PIDController(wRef_T,wMeas_T);

      if (u_R>=0) forward(u_R,in1_R,in2_R,ena_R); else
backward(abs(u_R),in1_R,in2_R,ena_R);
      if (u_T>=0) forward(u_T,in1_T,in2_T,ena_T); else
backward(abs(u_T),in1_T,in2_T,ena_T);

      /*----Envio serial a Matlab-----*/

      Serial.println(wMeas_R);
      Serial.println(u_R);
      Serial.println(thetaMeas_R);
      Serial.println(wMeas_T);
```

```
      Serial.println(u_T);
      Serial.println(thetaMeas_T);
  }
}
```

```
void serialEvent() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    inputString += inChar;
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
}
```

```
////////////////////// Encoder precisión cuádruple/////////////////////
void encoderR(void)
{
  prev_R = act_R;
  act_R = PIND & 48;

  if(prev_R==0  && act_R== 16)  {count_R+=1;}
  if(prev_R==16  && act_R== 48) {count_R+=1;}
  if(prev_R==32  && act_R== 0)  {count_R+=1;}
  if(prev_R==48  && act_R== 32) {count_R+=1;}

  if(prev_R==0 && act_R==32)  {count_R-=1;}
  if(prev_R==16 && act_R==0)  {count_R-=1;}
  if(prev_R==32 && act_R==48) {count_R-=1;}
  if(prev_R==48 && act_R==16) {count_R-=1;}
}

void encoderT(void)
{
  prev_T = act_T;
  act_T = PIND & 12;

  if(prev_T==0  && act_T== 4)  {count_T+=1;}
  if(prev_T==4  && act_T==12)  {count_T+=1;}
  if(prev_T==8  && act_T== 0)  {count_T+=1;}
  if(prev_T==12 && act_T== 8)  {count_T+=1;}

  if(prev_T==0 && act_T==8)  {count_T-=1;}
  if(prev_T==4 && act_T==0)  {count_T-=1;}
  if(prev_T==8 && act_T==12) {count_T-=1;}
  if(prev_T==12 && act_T==4) {count_T-=1;}
}
```

```
// Funciones envio de señal de control u(t)
void forward(int cv, int pind1, int pind2, int analogPin)
{
  digitalWrite(pind1, LOW);
  digitalWrite(pind2, HIGH);
  analogWrite(analogPin, cv);
```

```
  }

void backward(int cv, int pind1, int pind2, int analogPin)
{
  digitalWrite(pind1, HIGH);
  digitalWrite(pind2, LOW);
  analogWrite(analogPin, cv);
  }
```

- Programación interfaz gráfica con comunicación serial (App Designer)

```
properties (Access = private)
        Stop
        N
        COM
        contador
        datos
        memo
        z

        t
        Error_Theta_R
        Error_Theta_T
        thetaRef_R
        thetaRef_T
        thetaMeas_R
        thetaMeas_T
        wRef_R
        wRef_T
        wMeas_R
        wMeas_T

        SerialP

        R1
        R2
        R3
    end
```

```
% Code that executes after component creation
        function startupFcn(app)
            app.Stop = false;
            app.UITable.ColumnFormat = {'char', 'numeric', 'numeric',
'logical', 'logical', 'numeric', 'numeric'};
            app.UITable.Data = cell(5,7);
            app.memo = zeros(7,20);
            app.contador = 1;
            app.z = 1;
            app.R1 = 0;
            app.R2 = 0;
            app.R3 = 0;
```

```
        end
% Button pushed function: Boton_Start
        function Boton_StartButtonPushed(app, event)
%             La magia está aqui SIUUUUUUUUUUUUUUUUU
            flag_start = 1;
            app.Stop = false;
            cla(app.Axes_Senales2, "reset")
            cla(app.Axes_Senales1, "reset")

            %%%%%%%%%%%%%%% INICIALIZA COMUNICACION SERIAL
%%%%%%%%%%%%%%%%%%
            ports = serialportlist;
            if isempty(ports)
                ports ='NONE';
                clc
                disp('No se ha encontrado CONEXIÓN con el Dispositivo de
Control');
            else
                app.COM =ports(2);                  %Seleccionar puerto USB
            end

            baudrate = 9600;
            delete(instrfind({'Port'},{app.COM}));
            app.SerialP = serialport(app.COM,baudrate);
%SerialP is a serial object



            app.Puerto_USB.Value = app.COM;     %Muestra el puerto USB
conectado en el indicador

            while app.contador <20 && flag_start ==1
                T = app.Periodo.Value;

                ti = 0; ts = 0.1; tf = app.NroRepeticiones.Value * T;
                app.t = ti:ts:tf;        % Vector de tiempo
                app.N = length(app.t);       % Nro iteraciones

    %             thetaRef_R = zeros(1,app.N);  thetaRef_T = zeros(1,N);
%Posiciones angulares (sp)
                app.thetaMeas_R = zeros(1, app.N); app.thetaMeas_T =
zeros(1, app.N); %Variable a recibir de arduino
                app.wRef_R = zeros(1, app.N);       app.wRef_T = zeros(1,
app.N);             %Velocidades angulares (cv)
                app.wMeas_R = zeros(1, app.N);      app.wMeas_T = zeros(1,
app.N);      %Variables a recibir


                u_R = zeros(1,app.N);              u_T = zeros(1,app.N);
%pwm
                app.Error_Theta_R = zeros(1,app.N); app.Error_Theta_T =
zeros(1,app.N);      %Errores

                %--------------------Trayectoria angular(t) deseada (sp)----
-----------------
```

```
                A_R = 200; A_T = 200;    f = 1/T;
                offset = pi/2;          %90 grados de desfase

                %-------------------Extension en rodilla y tobillo----------
-----------
                %---------Posiciones angulares deseadas--------------------
------
                if (app.RodillaCheckBox.Value == 1) &&
(app.TobilloCheckBox.Value == 1)

                    app.thetaRef_R = A_R + A_R*sin(2*pi*f*app.t-offset);
                    thetaRef_R_p = A_R*2*pi*f*cos(2*pi*f*app.t-offset);

                    app.thetaRef_T = A_T + A_T*sin(2*pi*f*app.t-offset);
                    thetaRef_T_p = A_T*2*pi*f*cos(2*pi*f*app.t-offset);

                elseif (app.RodillaCheckBox.Value == 1) &&
(app.TobilloCheckBox.Value == 0)

                    app.thetaRef_R = A_R + A_R*sin(2*pi*f*app.t-offset);
                    thetaRef_R_p = A_T*2*pi*f*cos(2*pi*f*app.t-offset);

                    app.thetaRef_T = zeros(1, app.N);
                    thetaRef_T_p = zeros(1, app.N);
                elseif (app.RodillaCheckBox.Value == 0) &&
(app.TobilloCheckBox.Value == 1)

                    app.thetaRef_R = zeros(1, app.N);
                    thetaRef_R_p = zeros(1, app.N);

                    app.thetaRef_T = A_T + A_T*sin(2*pi*f*app.t-offset);
                    thetaRef_T_p = A_T*2*pi*f*cos(2*pi*f*app.t-offset);
                end

                %%%%%%%%%%%%%%%%%%%%%% BUCLE DE CONTROL %%%%%%%%%%%%%%%%%%%
                tiempo= zeros(1, app.N);
                k = 1;

                while k < app.N
                    if app.Stop == true
                      break
                   else
                   end
                    tic;
                    % a) Calculando errores de posición angular
                    app.Error_Theta_R(k) = app.thetaRef_R(k) -
app.thetaMeas_R(k);
                    app.Error_Theta_T(k) = app.thetaRef_T(k) -
app.thetaMeas_T(k);

                    ErrorTheta = [app.Error_Theta_R(k);...
                                  app.Error_Theta_T(k)];       %vector de
errores
```

```
                    % b) Matriz Jacobiana
                    J= [1  0;...
                        0  1];

                    % c) Parámetros de control
                        %ganancias

                    K = [0.30 0;...
                        0 0.42];                % aceptable

                    theta_ref_p = [thetaRef_R_p(k);...
                                   thetaRef_T_p(k)];

                    % d) Ley de control
                    w_RT = J*(theta_ref_p+K*tanh(ErrorTheta));

                    % e) Separando las acciones de control para rodilla y
tobillo
                    app.wRef_R(k) = w_RT(1);
                    app.wRef_T(k) = w_RT(2);

                    %-----------Envio de cv (pwm)
                    formatSpec = '%d,%d';
                    data0       = app.wRef_R(k); data1 = app.wRef_T(k);
                    send        = sprintf(formatSpec,data0,data1); %enviar
cv
                    writeline(app.SerialP,send);

                    %----------Lectura de theta_Meas(t )(rad/s) y u(t)
                    app.wMeas_R(k)     = str2double(readline(app.SerialP));
                    u_R(k)         = str2double(readline(app.SerialP));
%pwm
                    app.thetaMeas_R(k) = str2double(readline(app.SerialP));
                    app.wMeas_T(k)     = str2double(readline(app.SerialP));
                    u_T(k)         = str2double(readline(app.SerialP));
%pwm
                    app.thetaMeas_T(k) = str2double(readline(app.SerialP));
                    tiempo(k)= toc;
                    app.Barra_Progreso.Value = (k/app.N)*100;
                    k = k+1;
                    while toc<ts
                    end
                end
                writeline(app.SerialP,'0,0');
                pause(0.1);
                writeline(app.SerialP,'0,0');
%                 delete(instrfind({'Port'},{app.COM}));
                app.SerialP = [];
%                 clear app.SerialP;


                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                app.Error_Theta_R = (100/A_R/2)*(app.thetaRef_R -
app.thetaMeas_R);
```

```
                app.Error_Theta_T = (100/A_T/2)*(app.thetaRef_T -
app.thetaMeas_T);

                %----------------------Indices de performance-------------
                IAE_R = 0;
                IAE_T = 0;
                for s = 1:app.N
                    IAE_R = IAE_R + abs(app.Error_Theta_R(s))*ts;
                    IAE_T = IAE_T + abs(app.Error_Theta_T(s))*ts;
                end
                IAE_R = IAE_R / (app.NroRepeticiones.Value*10);
                IAE_T = IAE_T / (app.NroRepeticiones.Value*10);

                if (app.RodillaCheckBox.Value == 0) &&
(app.TobilloCheckBox.Value == 1)
                    axis(app.Axes_Senales1,'off')

                    hold(app.Axes_Senales2,'on')
                    grid(app.Axes_Senales2,'on')

                    app.Axes_Senales2.Title.String = 'Posiciones angulares
\theta_T (°)';

plot(app.Axes_Senales2,app.t,app.thetaRef_T,'b','LineWidth',2);

plot(app.Axes_Senales2,app.t,app.thetaMeas_T,'r','LineWidth',2);
                    xlabel(app.Axes_Senales2,'Tiempo (s)');
ylabel(app.Axes_Senales2,'Posicion angular \theta_T (°)');
                    legend(app.Axes_Senales2,'Posicion Ref','Posicion
Meas');
                elseif (app.RodillaCheckBox.Value == 1) &&
(app.TobilloCheckBox.Value == 0)
                    axis(app.Axes_Senales2,'off')

                    hold(app.Axes_Senales1,'on')
                    grid(app.Axes_Senales1,'on')

                    app.Axes_Senales1.Title.String = 'Posiciones angulares
\theta_R (°)';

plot(app.Axes_Senales1,app.t,app.thetaRef_R,'b','LineWidth',2);

plot(app.Axes_Senales1,app.t,app.thetaMeas_R,'r','LineWidth',2);
                    xlabel(app.Axes_Senales1,'Tiempo (s)');
ylabel(app.Axes_Senales1,'Posicion angular \theta_R (°)');
                    legend(app.Axes_Senales1,'Posicion Ref','Posicion
Meas');
                else
                    hold(app.Axes_Senales1,'on')
                    hold(app.Axes_Senales2,'on')
                    grid(app.Axes_Senales1,'on')
                    grid(app.Axes_Senales2,'on')

                    app.Axes_Senales1.Title.String = 'Posiciones angulares
\theta_R (°)';
```

```
plot(app.Axes_Senales1,app.t,app.thetaRef_R,'b','LineWidth',2);

plot(app.Axes_Senales1,app.t,app.thetaMeas_R,'r','LineWidth',2);
                    xlabel(app.Axes_Senales1,'Tiempo (s)');
ylabel(app.Axes_Senales1,'Posicion angular \theta_R (°)');
                    legend(app.Axes_Senales1,'Posicion Ref','Posicion
Meas');

                    app.Axes_Senales2.Title.String = 'Posiciones angulares
\theta_T (°)';

plot(app.Axes_Senales2,app.t,app.thetaRef_T,'b','LineWidth',2);

plot(app.Axes_Senales2,app.t,app.thetaMeas_T,'r','LineWidth',2);
                    xlabel(app.Axes_Senales2,'Tiempo (s)');
ylabel(app.Axes_Senales2,'Posicion angular \theta_T (°)');
                    legend(app.Axes_Senales2,'Posicion Ref','Posicion
Meas');
                end

                Codigo_p(app.contador) = app.CodigoPaciente.Value;
                Repeticiones(app.contador) = app.NroRepeticiones.Value;
    %                 time(app.contador) = tf;
                Rodilla(app.contador) = app.RodillaCheckBox.Value;
                Tobillo(app.contador) = app.TobilloCheckBox.Value;

                if IAE_R > 7 && IAE_R > 0
                        IAE_R = 9999;
                else
                    IAE_R = round(IAE_R, 2);
                end

                if IAE_T > 7 && IAE_T > 0
                    IAE_T = 9999;
                else
                    IAE_T = round(IAE_T, 2);
                end

                if app.z == app.contador
                    app.memo(:,app.z) = [Codigo_p(app.z);
Repeticiones(app.z); tf; Rodilla(app.z); Tobillo(app.z); IAE_R ; IAE_T];
                end

                app.contador = app.contador +1;
                flag_start = 0;

            end

                app.datos = app.memo;
                app.UITable.Data = app.datos';
                app.z = app.z+1;

        end
```

```
% Selection changed function: SeleccionarGrafico
        function SeleccionarGraficoSelectionChanged(app, event)
%             selectedButton = app.SeleccionarGrafico.SelectedObject;

            app.R1 = app.Posiciones_Button.Value;
            app.R2 = app.Errores_Button.Value;
            app.R3 = app.Velocidades_Button.Value;

            if app.R1 == 1
                cla(app.Axes_Senales1,"reset")
                cla(app.Axes_Senales2,"reset")
                hold(app.Axes_Senales1,'on')
                hold(app.Axes_Senales2,'on')
                grid(app.Axes_Senales1,'on')
                grid(app.Axes_Senales2,'on')

                app.Axes_Senales1.Title.String = 'Posiciones angulares
\theta_R (°)';

plot(app.Axes_Senales1,app.t,app.thetaRef_R,'b','LineWidth',2);

plot(app.Axes_Senales1,app.t,app.thetaMeas_R,'r','LineWidth',2);
                xlabel(app.Axes_Senales1,'Tiempo (s)');
ylabel(app.Axes_Senales1,'Posicion angular \theta_R (°)')
                legend(app.Axes_Senales1,'Posicion Ref','Posicion Meas');

                app.Axes_Senales2.Title.String = 'Posiciones angulares
\theta_T (°)';

plot(app.Axes_Senales2,app.t,app.thetaRef_T,'b','LineWidth',2);

plot(app.Axes_Senales2,app.t,app.thetaMeas_T,'r','LineWidth',2);
                xlabel(app.Axes_Senales2,'Tiempo (s)');
ylabel(app.Axes_Senales2,'Posicion angular \theta_T (°)')
                legend(app.Axes_Senales2,'Posicion Ref','Posicion Meas');

            elseif app.R2 ==1
                cla(app.Axes_Senales1,"reset")
                cla(app.Axes_Senales2,"reset")
                hold(app.Axes_Senales1,'on')
                hold(app.Axes_Senales2,'on')
                grid(app.Axes_Senales1,'on')
                grid(app.Axes_Senales2,'on')

                app.Axes_Senales1.Title.String = 'Gráfica de errores
\theta_R (%)';

plot(app.Axes_Senales1,app.t,app.Error_Theta_R,'r','LineWidth',2);
                xlabel(app.Axes_Senales1,'Tiempo (s)');
ylabel(app.Axes_Senales1,'Error en \theta_R (°)');

                app.Axes_Senales2.Title.String = 'Gráfica de errores
\theta_T (%)';

plot(app.Axes_Senales2,app.t,app.Error_Theta_T,'r','LineWidth',2);
```

```
                xlabel(app.Axes_Senales2,'Tiempo (s)');
ylabel(app.Axes_Senales2,'Error en \theta_T (°)');


            elseif app.R3 == 1
                cla(app.Axes_Senales1,"reset")
                cla(app.Axes_Senales2,"reset")
                hold(app.Axes_Senales1,'on')
                hold(app.Axes_Senales2,'on')
                grid(app.Axes_Senales1,'on')
                grid(app.Axes_Senales2,'on')

                app.Axes_Senales1.Title.String = 'Velocidades angulares w_R
(deg/s)';
                plot(app.Axes_Senales1,app.t,app.wRef_R,'b','LineWidth',2);
                plot(app.Axes_Senales1,app.t,app.wMeas_R,'r','LineWidth',2);
                xlabel(app.Axes_Senales1,'Tiempo (s)');
ylabel(app.Axes_Senales1,'Velocidad angular \omega_R (rad/s)');
                legend(app.Axes_Senales1,'Velocidad Ref','Velocidad Meas');

                app.Axes_Senales2.Title.String = 'Velocidades angulares w_T
(deg/s)';
                plot(app.Axes_Senales2,app.t,app.wRef_T,'b','LineWidth',2);
                plot(app.Axes_Senales2,app.t,app.wMeas_T,'r','LineWidth',2);
                xlabel(app.Axes_Senales2,'Tiempo (s)');
ylabel(app.Axes_Senales2,'Velocidad angular \omega_T (rad/s)');
                legend(app.Axes_Senales2,'Velocidad Ref','Velocidad Meas');
            end

            if (app.RodillaCheckBox.Value == 0) &&
(app.TobilloCheckBox.Value == 1)
                cla(app.Axes_Senales1,"reset")
                axis(app.Axes_Senales1,'off')

            elseif (app.RodillaCheckBox.Value == 1) &&
(app.TobilloCheckBox.Value == 0)
                cla(app.Axes_Senales2,"reset")
                axis(app.Axes_Senales2,'off')
            end
        end
```

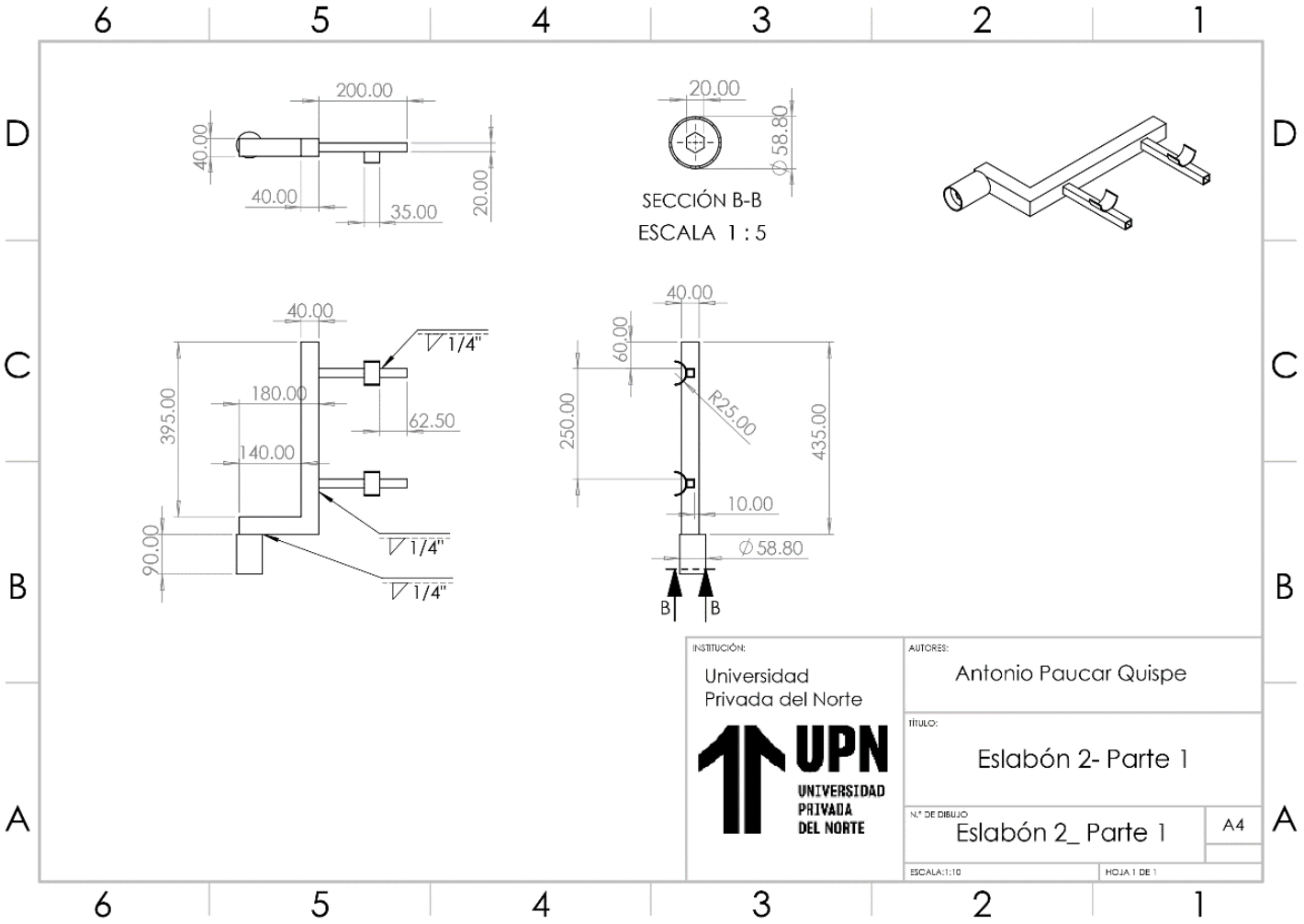
```
% Button pushed function: Boton_Pausa
        function Boton_PausaButtonPushed(app, event)
            app.Stop = true;
            writeline(app.SerialP,'0,0');
            pause(0.1);
            writeline(app.SerialP,'0,0');
        end
```

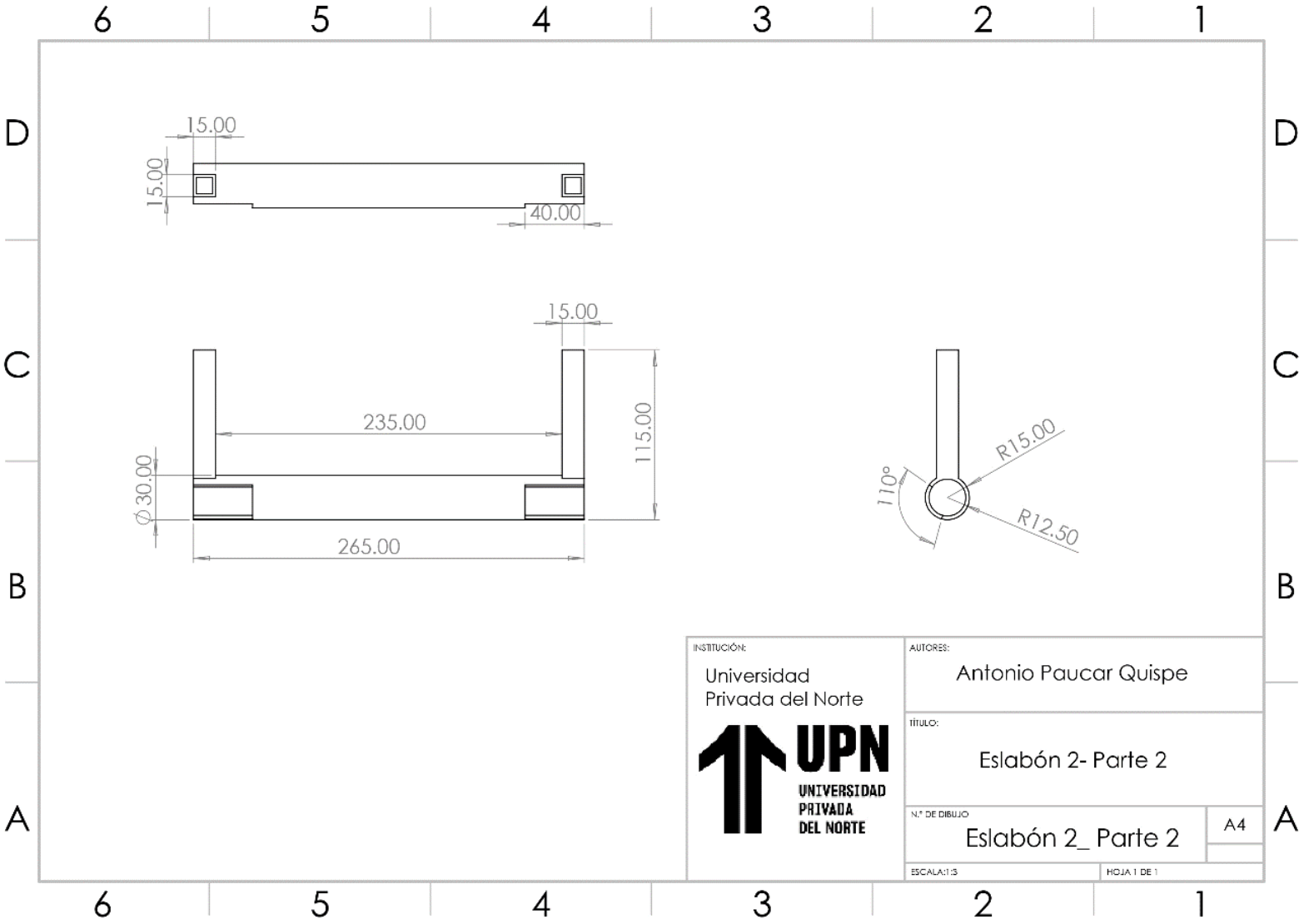
```
% Button pushed function: Exportar
        function ExportarPushed(app, event)
            data = get(app.UITable, 'data');
            datax = get(app.UITable, 'ColumnName');
            filename = 'Datos_Pacientes_Real.xls';
            xlswrite(filename,datax');
```

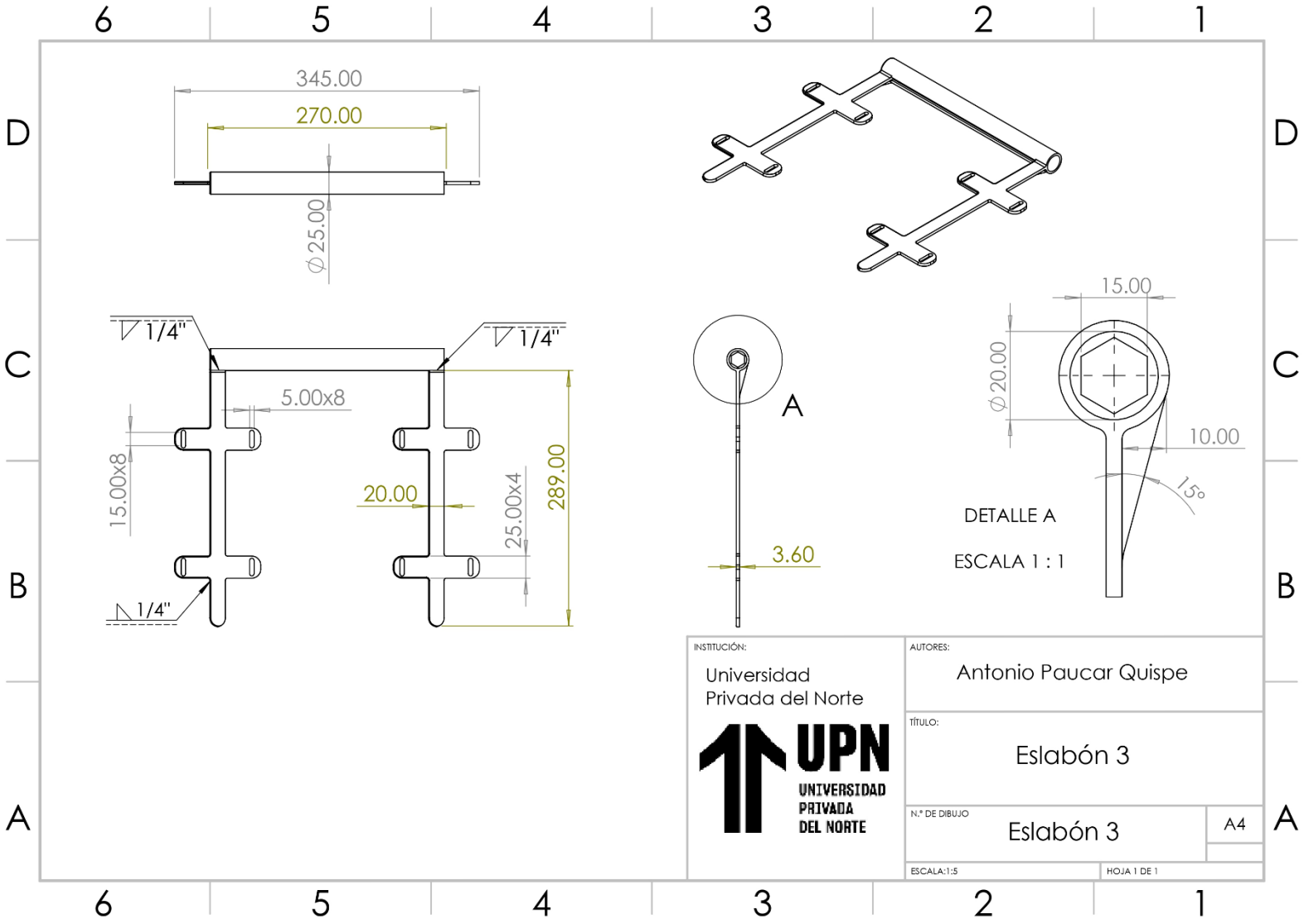
```
            limpiar = zeros(20,7);
            xlswrite(filename,limpiar,1,'A2:G11');
            pause(0.1);
            xlswrite(filename,data,1,'A2:G11');
        end
```

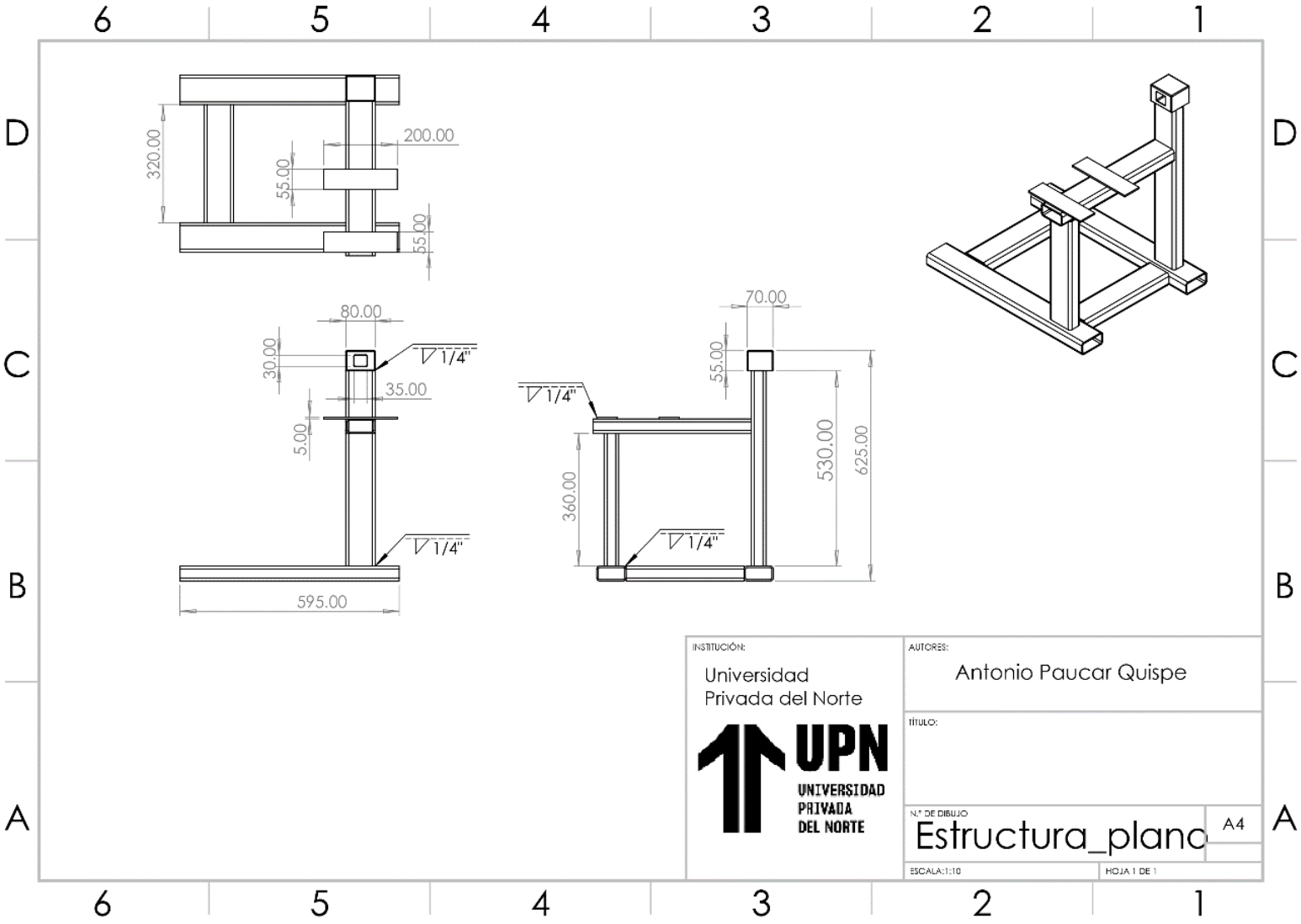
```
% Menu selected function: CerrarMenu
        function CerrarMenuSelected(app, event)
            close(app.UIFigure);
        end
```

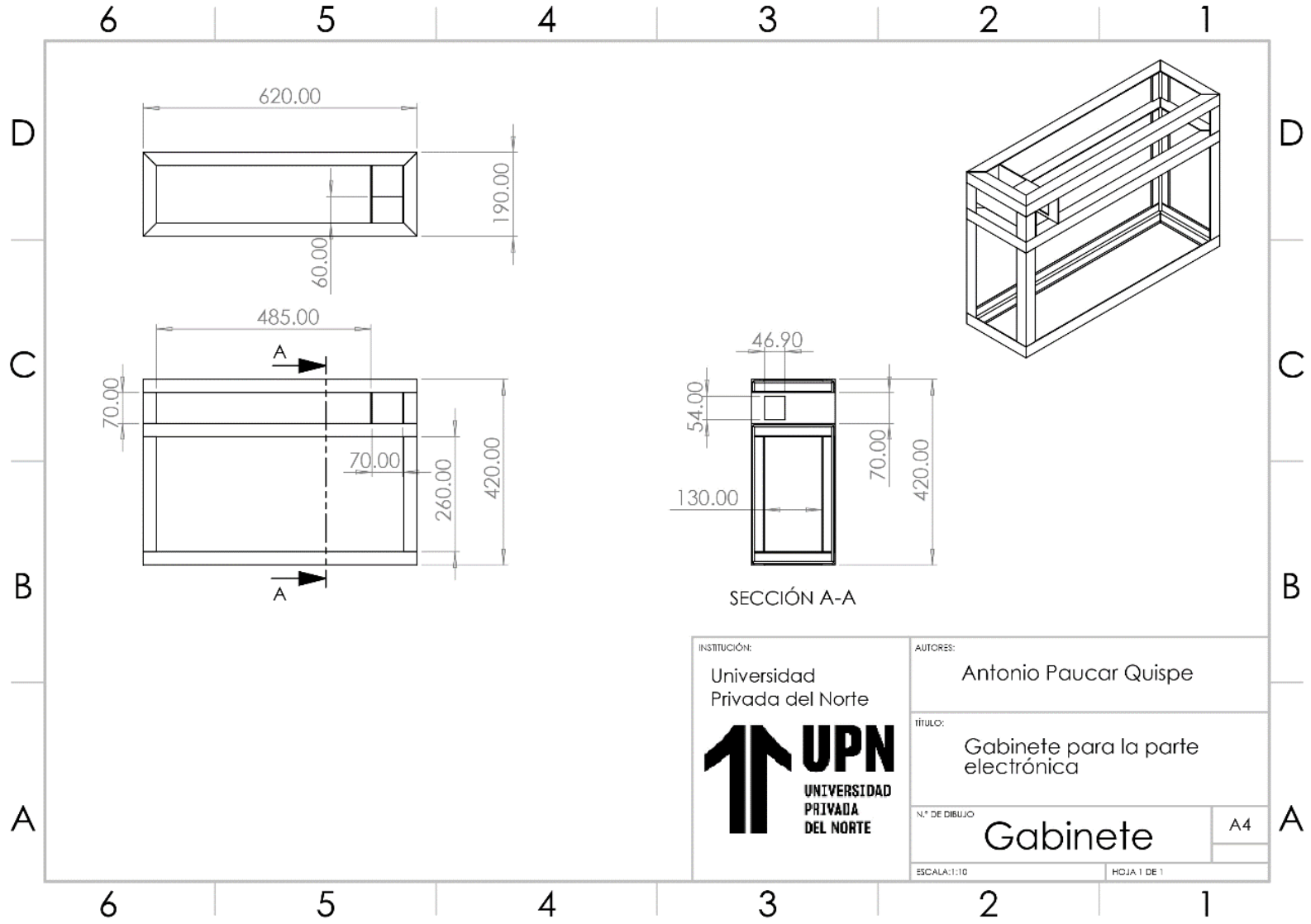
Anexo C: Planos mecánicos

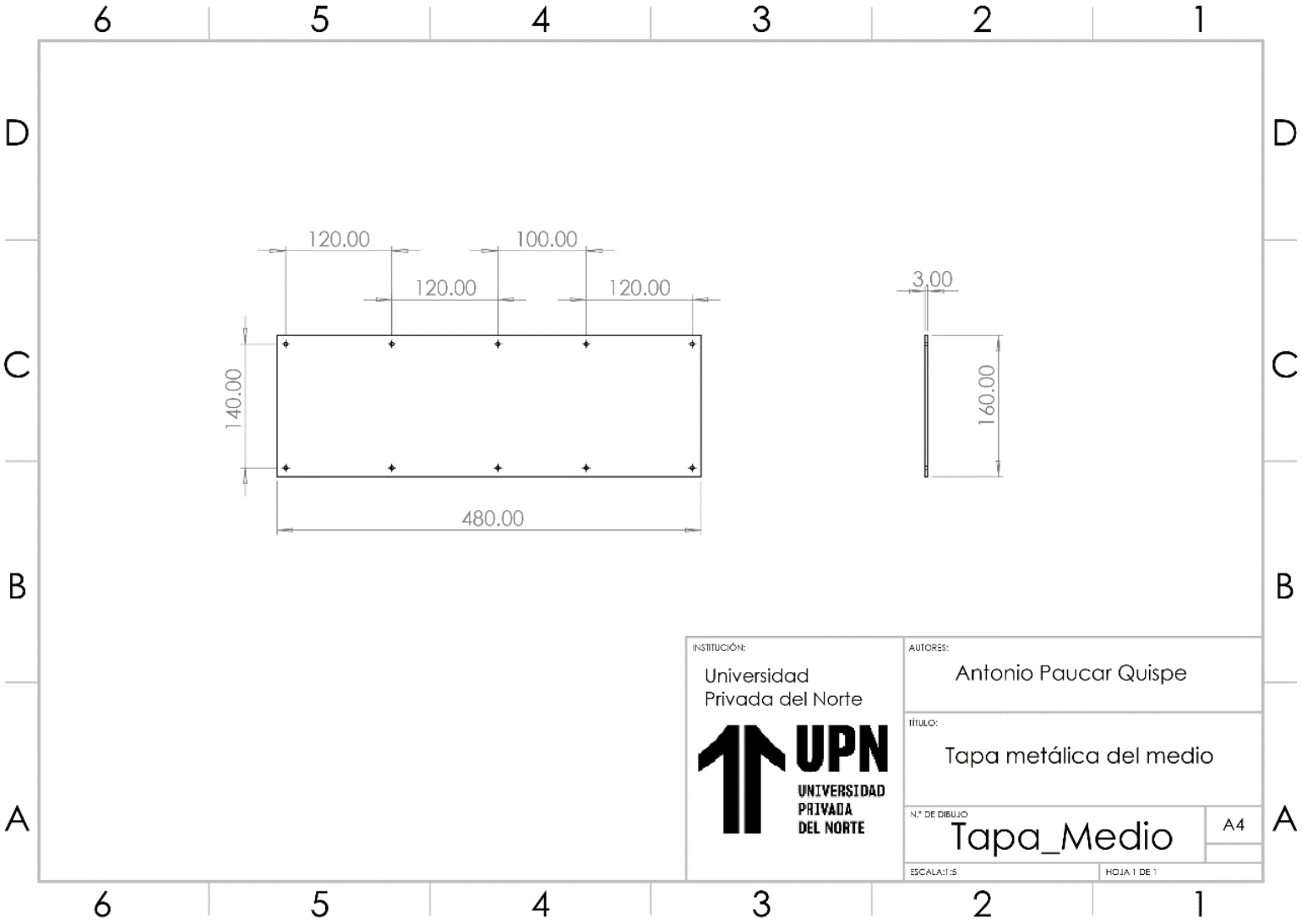



15.00
15.00
40.00
15.00
235.00
115.00
Ø30.00
265.00
110°
R15.00
R12.50
INSTITUCIÓN:
Universidad Privada del Norte
AUTORES:
Antonio Paucar Quispe
TÍTULO:
Eslabón 2- Parte 2
N.º DE DIBUJO
Eslabón 2_ Parte 2
A4
ESCALA:1:5
HOJA 1 DE 1


345.00
270.00
Ø25.00
1/4"
1/4"
5.00x8
15.00x8
20.00
25.00x4
289.00
1/4"
A
3.60
15.00
Ø20.00
10.00
15°
DETALLE A
ESCALA 1 : 1
INSTITUCIÓN:
Universidad Privada del Norte
UPN UNIVERSIDAD PRIVADA DEL NORTE
AUTORES:
Antonio Paucar Quispe
TÍTULO:
Eslabón 3
N.º DE DIBUJO
Eslabón 3
A4
ESCALA:1:5
HOJA 1 DE 1


6
5
4
3
2
1
D
C
B
A
320.00
200.00
55.00
55.00
80.00
30.00
1/4"
35.00
5.00
1/4"
595.00
70.00
55.00
1/4"
530.00
625.00
360.00
1/4"
INSTITUCIÓN:
Universidad Privada del Norte
UPN UNIVERSIDAD PRIVADA DEL NORTE
AUTORES:
Antonio Paucar Quispe
TÍTULO:
N.º DE DIBUJO
Estructura_plano
A4
ESCALA:1:10
HOJA 1 DE 1


620.00
190.00
60.00
485.00
A
70.00
70.00
260.00
420.00
A
46.90
54.00
70.00
420.00
130.00
SECCIÓN A-A
INSTITUCIÓN:
Universidad Privada del Norte
UPN UNIVERSIDAD PRIVADA DEL NORTE
AUTORES:
Antonio Paucar Quispe
TÍTULO:
Gabinete para la parte electrónica
N.º DE DIBUJO
Gabinete
A4
ESCALA:1:10
HOJA 1 DE 1


6
5
4
3
2
1
D
C
B
A
120.00
100.00
120.00
120.00
140.00
480.00
3.00
160.00
INSTITUCIÓN:
Universidad Privada del Norte
UPN UNIVERSIDAD PRIVADA DEL NORTE
AUTORES:
Antonio Paucar Quispe
TÍTULO:
Tapa metálica del medio
N.° DE DIBUJO
Tapa_Medio
A4
ESCALA:1:5
HOJA 1 DE 1

Anexo D: Datasheets

Componente	Link
Arduino Nano	https://docs.arduino.cc/static/70488a44662b48155f3def728bcf6454/A000005-datasheet.pdf
LM7805 Regulador de voltaje 5V	https://www.st.com/content/ccc/resource/technical/document/datasheet/41/4f/b3/b0/12/d4/47/88/CD00000444.pdf/files/CD00000444.pdf/jcr:content/translations/en.CD00000444.pdf
Módulo 3205x8 Driver para motor	https://www.amazon.com/dp/B094W7KKL8/ref=sspa_dk_detail_0?psc=1&content-id=amzn1.sym.0d1092dc-81bb-493f-8769-d5c802257e94&s=industrial&sp_csd=d2lkZ2V0TmFtZT1zcF9kZXRhaWwy
IRF3205	https://pdf1.alldatasheet.es/datasheet-pdf/view/68131/IRF/IRF3205.html
Aluminio 6061	https://www.gabrian.com/wp-content/uploads/2018/09/6061-Aluminum-Alloy-Properties-1.pdf