ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"MODELADO, DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE CONTROL DE POSICIÓN PARA PLATAFORMA DE TIRO CON GIRO-ESTABILIZACIÓN EN EMBARCACIONES MEDIANTE TÉCNICAS DE CONTROL DIFUSO"

TRABAJO DE TITULACIÓN

Previo a la obtención del título de:

MAGISTER EN AUTOMATIZACIÓN Y CONTROL INDUSTRIAL

RONALD DAVID SOLIS MESA

GUAYAQUIL – ECUADOR

AÑO: 2017

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"MODELADO, DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE CONTROL DE POSICIÓN PARA PLATAFORMA DE TIRO CON GIRO-ESTABILIZACIÓN EN EMBARCACIONES MEDIANTE TÉCNICAS DE CONTROL DIFUSO"

TRABAJO DE TITULACIÓN

Previo a la obtención del título de:

MAGISTER EN AUTOMATIZACIÓN Y CONTROL INDUSTRIAL

RONALD DAVID SOLIS MESA

GUAYAQUIL – ECUADOR

AÑO: 2017

AGRADECIMIENTOS

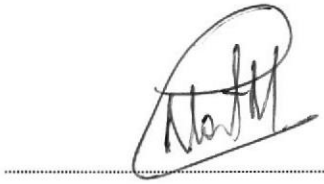
Mis más sinceros agradecimientos a Dios, mi familia y mi esposa ya que sin su apoyo este trabajo no sería posible, por su paciencia y compañía que supieron otorgarme durante el trabajo de titulación.

Agradezco también a mis colegas y a mi director de trabajo de titulación por sus conocimientos compartidos durante la implementación del proyecto los cuales forjaron la base para el desarrollo y finalización del trabajo.

DEDICATORIA

El presente proyecto lo dedico a mi esposa y a mi hijo que está en el cielo y ahora es un angelito que siempre estará en nuestros corazones.

TRIBUNAL DE EVALUACIÓN



Ph.D. César Martín

SUBDECANO DE FIEC

M.Sc. Carlos Valdivieso

DIRECTOR DE TRABAJO DE TITULACIÓN

Ph.D. Douglas Plaza

MIEMBRO PRINCIPAL DEL TRIBUNAL

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, me corresponde exclusivamente; y doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Ronald David Solis Mesa

RESUMEN

El presente trabajo nace por la necesidad de las fuerzas armadas del Ecuador de buscar un mejor control sobre sus espacios marinos, protección de sus recursos y persecución de actividades delictivas [1],como se puede observar en las lanchas adquiridas en Ecuador durante los últimos años, las cuales poseen varios sistemas de navegación y servicios de alta tecnología [2]; sin embargo, el riesgo de estos actos es alto debido a que no se menciona nada sobre mecanismos autónomos de defensa como plataformas de tiro con giro-estabilización como lo ha hecho Colombia [3] con el fin de evitar pérdidas humanas durante el fuego cruzado [5]; por ende es difícil ver a un lado y no pensar en el desarrollo local como una alternativa a nuestros problemas defensivos. En este trabajo se explicará el proceso de cómo modelar, diseñar e implementar un sistema de control de posición para plataforma de tiro con giro-estabilización en embarcaciones mediante técnicas de control difuso.

Se empezará con la justificación del tema en detalle, las teorías de la lógica difusa y cómo diseñar un controlador para mecanismos en ambientes marinos[4], los pasos para la implementación de una plataforma de tres ejes así como el análisis de su función de transferencia dado sus características eléctricas y estructurales, se explicará el software desarrollado para el cálculo más rápido de los diversos modelos matemáticos solo ingresando sus parámetros, y se mostrará los códigos para el movimiento de los diferentes mecanismos. Por último, una presentación de los resultados de la plataforma implementada y funcional capaz de reaccionar a cambios en su superficie y compensarlos. Por todo lo expuesto, el desarrollo de este tema puede incentivar a un mejor desarrollo industrial naval en Ecuador y generar la forma de patrullar sus costas con un mínimo de pérdidas humanas.

ÍNDICE GENERAL

CAPÍTULO 1

1. ANTECEDENTES

En países latinoamericanos como Colombia, debido a la elevada tasa de conflictos, se han visto en la necesidad de implementar y desarrollar políticas referentes a la defensa nacional para así mantener el orden público y la seguridad ciudadana. Gracias a ello su industria militar tuvo un gran desarrollo a tal punto de crear empresas e industrias públicas al mismo tiempo que impulsaron al sector privado a ser proveedores de materiales tanto para las Fuerzas Armadas como para el mercado internacional.

Tras este gran desarrollo, Colombia ha ido aumentando su tecnología, importando de otros países para luego fabricar su propia tecnología basada en sus necesidades y así no depender de terceros.

Un notorio ejemplo es el caso de la empresa Cotecmar, quien invierte recursos para obtener nuevos programas de investigación y desarrollo para la industria naval, además de generar nuevos procesos referentes al manejo de materiales para la construcción naval, al de corrosión marina y al sistema de calidad para la gestión de la industria. Todo esto le ha permitido ofrecer diferentes e innovadores tipos de embarcaciones tales como los Patrulleros de Zona Marítima [3].

Con esto se puede observar que existe una necesidad en nuestro país en cuanto a la defensa naval, debido a que muchos países latinoamericanos tienen un gran avance y desarrollo en esa área, sin considerar que Ecuador es un país costero y posee un amplio mar territorial que debe ser protegido tanto nacional como internacionalmente, y al mismo tiempo poder tener un cambio en nuestra matriz productiva.

1.1 Descripción del problema

La falta de sistemas de defensa con giro-estabilización en embarcaciones de vigilancia ecuatorianas aumenta la posibilidad de pérdidas humanas durante el fuego cruzado, ya que sin los mismos el artillero debe enfrentarse directamente y tratar de mantener su puntería frente a los movimientos marinos, la mayoría de los enfrentamientos son con narcotraficantes que usan las costas ecuatorianas como punto de paso a Colombia y hacia países vecinos [6], por ello es necesario dotar con las herramientas adecuadas para cumplir con esta labor, como sistemas de navegación, comunicación, defensa, etc. Sin embargo, estos sistemas deben trabajar en suelo marino donde no existe estabilidad debido a los movimientos de las olas del mar y podrían generar fallas en los mismos, sobre todo en sistemas de defensa como el sistema de tiro donde se debe apuntar y calcular la trayectoria deseada según un ángulo de inclinación, donde es necesario que la posición referencial del suelo sea fija para que los cálculos sean correctos, sin embargo esto no ocurre y el tiro puede fallar, por ello es importante que los sistemas posean plataformas con giro-estabilización para la corrección de estas perturbaciones.

1.2 Solución Propuesta

La propuesta de esta tesis consiste en desarrollar una plataforma con giro-estabilización que permita mantener el ángulo de inclinación pese a las perturbaciones de las olas.

1.3 Objetivo General

Modelar, diseñar e implementar un controlador difuso para plataforma con giro-estabilización de posición para perturbaciones ondulatorias generadas por las mareas durante la navegación con el fin de mantener una posición determinada pese a estos eventos. Además, dar la capacidad al usuario de establecer un modelo matemático de la planta de forma dinámica a través de un software que permita cambiar parámetros de la planta.

1.4 Objetivos Específicos

- Implementar un módulo didáctico para giro-estabilización.
- Implementar un módulo para simulación de movimiento ondulatorio para pruebas del controlador
- Encontrar el modelo matemático de la plataforma y desarrollar un software para obtenerlo frente a cambios de su tamaño o tipo de material a utilizar para su construcción.
- Determinar las reglas lingüísticas para el control de posición del módulo didáctico y diseño del controlador difuso base.

1.5 Metodología

Consiste en encontrar el modelo matemático de la plataforma con ayuda de un software para obtener varios modelos frente a cambios de su tamaño, tipo de material, etc. Luego se debe determinar las reglas lingüísticas para el control de posición del módulo didáctico y diseño del controlador difuso base. Luego, implementar un módulo didáctico para giro-estabilización que permita realizar pruebas para establecer una comparación entre el diseño base y un modelo de control clásico y así poder observar las ventajas y/o desventajas del uso de controladores difusos.

1.6 Alcance del trabajo

El trabajo consta de un prototipo tipo plataforma para colocar un objeto de disparo desde embarcaciones o sistemas con inestabilidad en su base el cual será simulado con un módulo que genere un movimiento ondulatorio. La plataforma modificará su posición según cambie la posición de la base con la ayuda de sensores giroscópicos, mientras que el servomotor y la carga se moverán utilizando estrategias de control difuso.

Además, se brindará al usuario un software que le permita cambiar los parámetros según sus cálculos y análisis con el objetivo de permitir el desarrollo de nuevos modelos de una forma más rápida.

CAPÍTULO 2

2. MARCO TEÓRICO

Este capítulo detalla todos los fundamentos teóricos que se emplean a lo largo del presente documento, empezando por una introducción al tema central que es la lógica difusa, luego se describe el modelamiento de la plataforma inercial implementada junto con todas sus partes fabricadas y por último una descripción de todos los módulos y dispositivos empleados en el desarrollo físico de la plataforma.

2.1 Introducción a la lógica difusa y controladores difusos.

En esta sección se presenta una introducción a la lógica difusa en sistemas de control, el cual es un preámbulo de toda la información detallada en el presente documento, también se detalla cada una de las etapas de un controlador difuso y el modelo a utilizar.

2.1.1 Lógica difusa

El término "conjuntos difusos" fue introducido en el año de 1965 en el artículo "Fuzzy Sets" por el matemático e ingeniero Lotfi A. Zadeh, en el que explica que los diferentes tipos de objetos que se encuentran en el mundo físico real no poseen de manera determinada una definición de pertenencia [7]. Sin embargo, estos conceptos no se aplicaron hasta el año 1974 donde Ebrahim Mamdani los puso en práctica para poder realizar el primer controlador difuso orientado a la regulación de un motor de vapor y 11 años después Takagi y Sugeno aportaron un nuevo método a la teoría del control difuso llamada Takagi-Sugeno-Kang (TSK), como alternativa para el método Mamdani.

La lógica difusa es una herramienta matemática que nos permite asociar la manera en cómo los seres humanos procesamos y adquirimos información, y así emitir juicios sobre dicha información tales como: "La

sopa esta salada y el café esta dulce" o "aquel auto va muy rápido comparado al de la otra calle" tal cual como lo haría el cerebro humano.

Esta ambigüedad de información está definida por los conjuntos difusos que son una extensión de la teoría clásica de conjuntos, donde un elemento solo posee dos posibilidades: pertenecer o no a un conjunto. Con el uso de conjuntos difusos lo que se busca es modelar la ambigüedad con la que se percibe una variable [8].

Por tal razón, la aplicación de la lógica difusa se la usa principalmente para modelos de sistemas no lineales donde lo que se busca lograr es emular el pensamiento humano para poder tomar una decisión a partir de una información no específica [7].

A esto se debe su gran éxito en diferentes áreas como el control industrial, la electrónica, inteligencia artificial, medicina, etc.

2.1.2 Controlador difuso aplicado

Un controlador lógico difuso (CLD), aplicas los principios de lógica difusa permitiendo la conversión de las estrategias de control lingüístico, basadas en conocimiento experto, en una estrategia de control automático. En la actualidad existen diversas metodologías para crear el control de un sistema, en el desarrollo de este trabajo se emplea el método propuesto por C.C. Lee empleando las siguientes etapas:

- Fusificación
- Base de conocimiento
- Lógica de decisiones
- Defusificación

a) Fusificación

El objetivo de esta etapa es convertir los parámetros físicos en variables difusas y asignarle a cada uno un grado de pertenencia en cada conjunto difuso que se vaya a evaluar por medio de las funciones de membresía

asociadas a dichos conjuntos. Primero, se trata de clasificar el medio a evaluar asignando etiquetas para cada variable difusa; luego, se seleccionan funciones de membresía que servirán para darle un valor numérico a cada etiqueta. Luego, dichas funciones se encargarán de mapear los datos para determinar el nivel de pertenencia [9]. Por ejemplo, en la Figura 2.1 se observan las funciones de membresía que se van a utilizar en el presente trabajo.

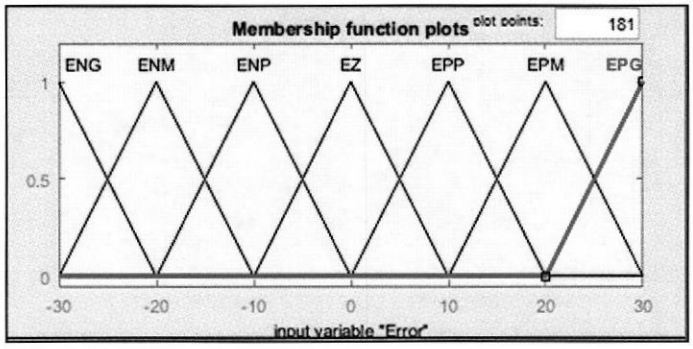


Figura 2.1: Funciones de membresía

b) Base de conocimiento

Es obtenida mediante la experiencia desarrollada por un operador y de conocimientos sobre Ingeniería de control, y depende del proceso que se quiera controlar y los requerimientos de diseño. Está compuesta de dos partes, una base de datos y una base de reglas de control difuso (utiliza variables lingüísticas). La base de reglas también se puede obtener a partir de métodos de optimización.

- **Base de datos**

Incluye discretizar y normalizar los universos de entrada y salida, la definición de subconjuntos (partición de los universos y funciones de pertenencia) y satisfacer la propiedad de completitud.

- Discretización.

Este proceso genera niveles cuantificados, ya que los datos se procesan en forma digital, cada uno de los cuales representa un elemento genérico en un universo de discurso.

- Normalización.

Esta normalización del universo discreto puede ser lineal o no.

- Partición de los universos.

Es el número de etiquetas que toma una variable lingüística; por ejemplo, la variable "temperatura de la sala", puede tomar los valores "muy baja", "baja", "media" y "alta". El número de términos está determinado por los requerimientos del sistema a controlar y la calidad de control.

- Funciones de pertenencia.

Las funciones de pertenencia o membresía, representan gráficamente la relación que tienen los elementos de un subconjunto difuso, dentro de un universo de discurso, con el grado de pertenencia al conjunto en cuestión. La representación de estas figuras puede tomar diferentes formas, siendo las más utilizadas las de forma triangular y trapezoidal. La utilización de alguna de ellas puede ser de manera arbitraria, dependiendo de la aplicación en particular.

- Completitud.

Esta propiedad indica que el algoritmo es capaz de inferir una acción correcta para cada estado del proceso.

- **Base de reglas**

La estrategia de control, que se deriva de la experiencia es expresada con el uso de algoritmos difusos. Las reglas de control que conforman el algoritmo difuso pueden ser definidas usando los siguientes criterios:

- Selección de las variables.

Las variables de entrada son seleccionadas basándose en la experiencia y en conocimientos de Ingeniería y el cambio de error (derivada del error). La importancia de usar la variación de error se puede ilustrar con un caso común, como el hecho de cruza una avenida, para lo cual no solo consideramos la distancia entre un auto y la persona que va cruzar la

calle, sino también consideramos muy importante la rapidez con la que el auto se desplaza.

- Origen y obtención de las reglas de control.

- La experiencia y los conocimientos en Ingeniería de control. Es la que más se utiliza.

- Usando métodos de optimización.

En un controlador lógico difuso, su operación dinámica se caracteriza por un grupo de reglas, compuestas por variables lingüísticas, basada en conocimiento experto es como de la forma:

IF (un conjunto de condiciones es satisfecho) THEN (un conjunto de consecuentes que pueden inferir).

Donde los antecedentes y los consecuentes de las reglas IF-Then son asociados con conceptos difusos (términos lingüísticos), formando lo que es conocido como declaraciones condicionales difusas en donde el antecedente es una condición sobre la base del estado de las variables del proceso y el consecuente es una acción de control para el sistema a controlar (proceso).

- Tipos de reglas de control.

Lee C. C., menciona que existen dos tipos de reglas usadas en el diseño de un CLD: las reglas de control de evaluación de estado y las reglas de evaluación de objeto.

Las reglas de estado son las más usadas en los sistemas de múltiples entradas y una salida (MISO), se estructuran de la siguiente manera:

R1 : si 'x_1' es A1 ,y 'x_2' es B1 entonces 'y' es C1.

R2 : si 'x_1' es A2 ,....., y 'x_m' es B2 entonces 'y' es C2.

R3 : si 'x_1' es A3 ,....., y 'x_m' es B3 entonces 'y' es C3.

Rn : si 'x_1' es An ,....., y 'x_m' es Bn entonces 'y' es Cn.

En cambio, en las reglas de objeto está involucrada la evaluación de los estados, y el resultado de la acción de control. Aplicado en control difuso predictivo.

c) Lógica de decisiones.

Mediante las funciones de implicación difusa y los mecanismos de inferencia, un CLD puede imitar a un operador experto.

- **Inferencia difusa**

La inferencia difusa es la encargada de distinguir las reglas aplicadas a cada evento, esto es realizado por el método MAX/MIN, determinando los valores de la variable lingüística de salida. A continuación, se presenta los pasos aplicados en este método.

- Agregación.
- Composición.

Asumiendo el control de un proceso se ha obtenido las siguientes variables lingüísticas de entrada: Error de posición y la derivada del error, mientras que la salida es la posición del servomotor.

Se presenta la siguiente base de reglas con los valores de membresías de las etiquetas que compone los antecedentes de cada una de las reglas.

d) Defusificación

Es un mapeo del espacio de acciones del control difuso. Esta etapa está definida sobre un Universo de discurso de salida, sobre los valores precisos que conforman el espacio de acciones de control no difuso. Este proceso conlleva vital importancia debido a que en las aplicaciones prácticas es requerido variable numérica. Las metodologías usadas son las siguientes:

- **Método del criterio máximo**

Es el punto de la distribución donde la posibilidad de la acción de control toma el valor máximo.

- **Método del promedio máximo**

Método del promedio máximo es el más sencillo entre los métodos de defusificación, el máximo grado de membresía, por ejemplo, sea un x=43 hasta x=55 (según la Figura 2.2), El promedio es 49, este es el valor de salida de la defusificación por el método del promedio máximo [10] y está dado por la siguiente ecuación:

$$MPM = \frac{x_{max1} + x_{max2}}{2} \tag{2.1}$$

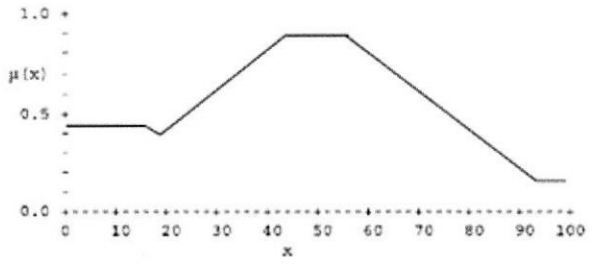


Figura 2.2: Método promedio máximo [10]

2.2 Modelamiento matemático de una plataforma inercial implementada.

Para modelar la planta se dividió en 3 brazos con su respectivo servomotor y eje rotacional los cuales hemos llamado según su posición en la planta.

El primer sistema o también llamado brazo 1 interno es el que sostendrá la carga que se desea mantener estable como por ejemplo en este prototipo un apuntador láser, las dimensiones y la ubicación del servomotor en el espacio se pueden observar en la Figura 2.3. Este sistema nos permite estabilizar el movimiento de arriba hacia abajo y viceversa (TILT).

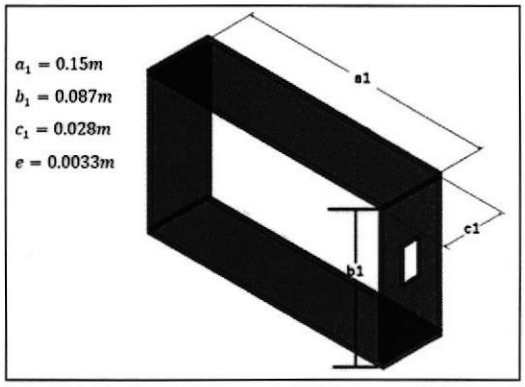


Figura 2.3: Brazo 1 (interno)

El segundo sistema o también llamado brazo 2 medio es el que sostendrá el primer sistema, las dimensiones y la ubicación del servomotor en el espacio se pueden observar en la Figura 2.4. Este sistema nos permite estabilizar el movimiento de izquierda a derecha y viceversa (PAN).

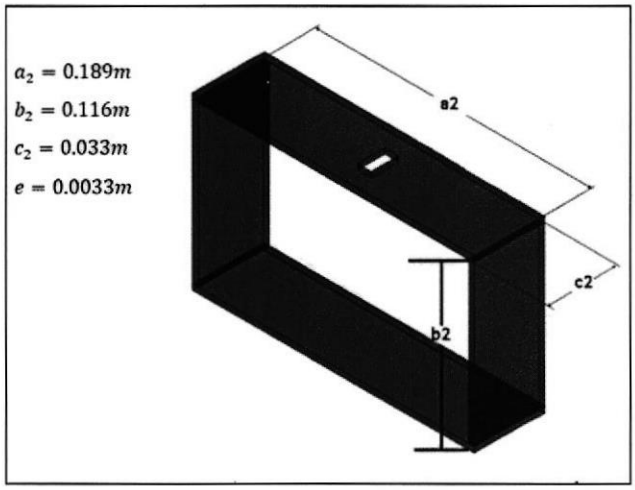


Figura 2.4: Brazo 2 (medio)

El tercer sistema o también llamado brazo 3 externo es el que sostendrá el sistema 1 y 2, las dimensiones y la ubicación del servomotor en el espacio se pueden observar en la Figura 2.5. Este sistema nos permite estabilizar el movimiento de rotación roll.

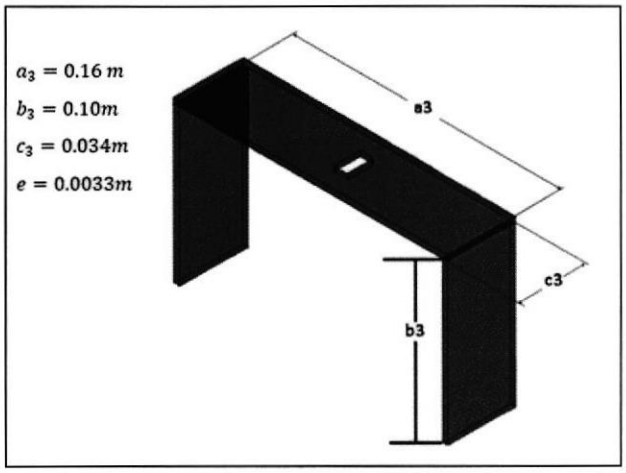


Figura 2.5: Brazo 3 (externo)

2.2.1 Modelamiento del motor DC para servomecanismo (HS311).

Para el cálculo matemático se hará referencia a los voltajes y variables mostradas en la Figura 2.6 y la Figura 2.8 para el modelo sin carga y con carga respectivamente.

Modelamiento matemático del motor sin carga.

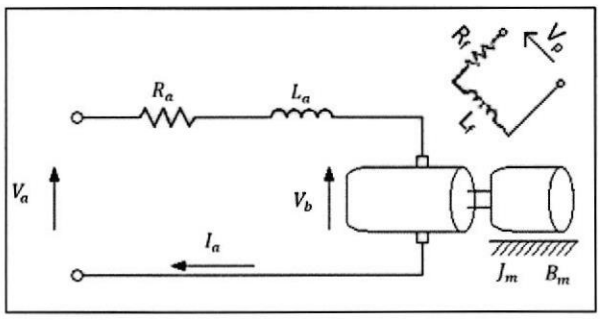


Figura 2.6: Diagrama del servomotor sin carga

Flujo Magnético:

$$\emptyset = K_f i_f \tag{2.2}$$

Torque Servomotor:

$$\tau_m = K_1 i_a \emptyset \tag{2.3}$$

Donde K_1 es la constante del motor, si la corriente de campo es constante entonces:

$$\tau_m = (K_1 K_f) i_a \tag{2.4}$$

Si $(K_1 K_f) = K_a$, entonces:

$$\tau_m = K_a i_a \tag{2.5}$$

Cuando la armadura rota, un voltaje es inducido llamado fuerza contra-electromotriz y depende de la velocidad angular $(\dot{\theta}_m)$

$$v_b = k\emptyset\dot{\theta}_m \tag{2.6}$$

Si

$$K_b = k\emptyset \tag{2.7}$$

Aplicando Kirchhoff en el diagrama mostrado en la Figura 2.6:

$$v_a = L_a \frac{di_a}{dt} + R_a i_a + v_b \tag{2.8}$$

Aplicando la transformada de Laplace y despejando I_a entonces:

$$I_a = \frac{V_a - V_b}{L_a S + R_a} \tag{2.9}$$

Reemplazando la ecuación (2.7) en (2.6) y aplicando Laplace:

$$V_b = K_b S\theta_m \tag{2.10}$$

Y luego reemplazando (2.10) en (2.9):

$$I_a = \frac{V_a - K_b S\theta_m}{L_a S + R_a} \tag{2.11}$$

La ecuación del torque del servomotor es con respecto a la posición:

$$\tau_m = J_m\ddot{\theta}_m + B_m\dot{\theta}_m \tag{2.12}$$

Donde J_m: Momento de inercia del servomotor

B_m: Coeficiente de fricción de viscosidad del servomotor

De la ecuación (2.5) y (2.12) tenemos:

$$K_a i_a = J_m\ddot{\theta}_m + B_m\dot{\theta}_m \tag{2.13}$$

En términos de Laplace y despejando I_a:

$$I_a = \frac{J_m S^2 \theta_m + B_m S \theta_m}{K_a} \tag{2.14}$$

Combinando (2.9) y (2.14) entonces:

$$K_a V_a - K_a K_b S \theta_m = (L_a S + R_a)(J_m S + B_m) S \theta_m \tag{2.15}$$

Despejando:

$$V_a = \frac{(L_a S + R_a)(J_m S + B_m) S \theta_m + K_a K_b S \theta_m}{K_a} \tag{2.16}$$

Modelo matemático del motor dc sin carga con entrada de voltaje y salida de posición angular.

$$\frac{\theta_m}{V_a} = \frac{K_a}{[(L_a S + R_a)(J_m S + B_m) + K_a K_b] S} \tag{2.17}$$

Por lo tanto, el diagrama de bloques del motor DC sin carga queda representado en la Figura 2.7

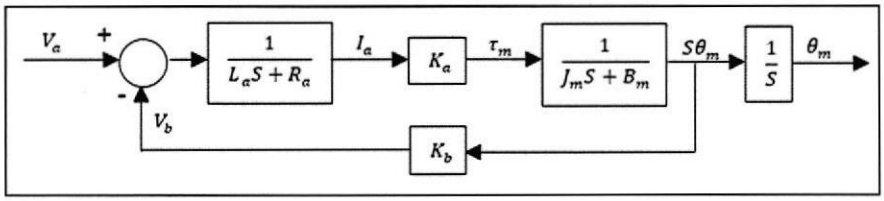


Figura 2.7 Diagrama de bloques del servomotor DC

Modelamiento matemático del motor dc con tren de engranaje y carga:

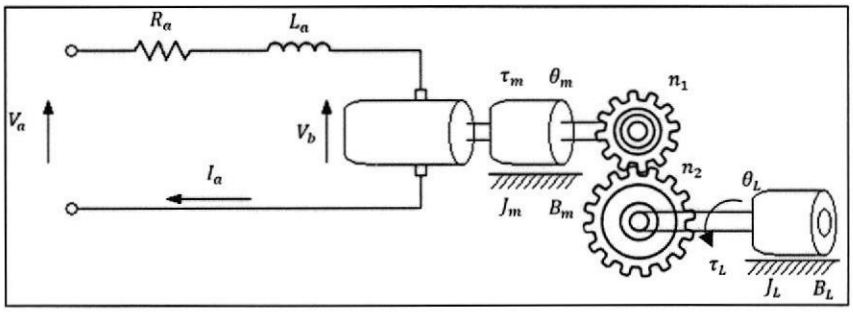


Figura 2.8 Diagrama del servomotor con tren de engranaje y carga

Para ello debemos recordar las siguientes relaciones:

$$\theta_L = \frac{n_1}{n_2}\theta_m \tag{2.18}$$

$$\tau_L = \frac{n_2}{n_1}\tau_m \tag{2.19}$$

Y sabemos la relación de torque con su posición angular tenemos:

$$J_L\theta_L S^2 + B_L\theta_L S = \tau_L \tag{2.20}$$

Ahora si referimos el torque de la carga al eje del motor:

$$J_L\frac{n_1}{n_2}\theta_m S^2 + B_L\frac{n_1}{n_2}\theta_m S = \frac{n_2}{n_1}\tau_m \tag{2.21}$$

Despejando τ_m y reduciendo a $n = \frac{n_1}{n_2}$:

$$(J_L S + B_L)n^2\theta_m S = \tau_m \tag{2.22}$$

Por lo tanto, se puede agregar al diagrama de bloques del servomotor DC el tren de engranajes y queda de la forma mostrada en la Figura 2.9

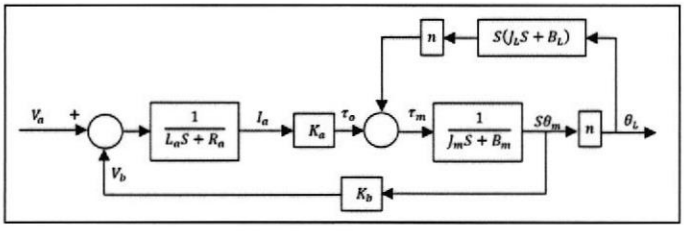


Figura 2.9 Diagrama de bloques del servomotor con tren de engranajes

Analizando la carga se obtiene su representación en la Figura 2.10

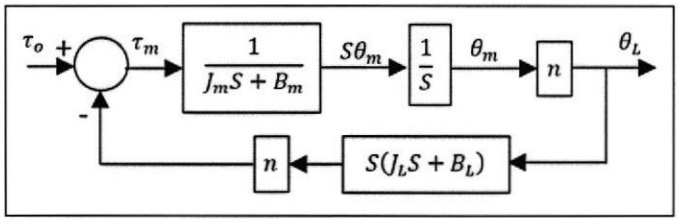


Figura 2.10 Diagrama de bloques de la carga

La relación entre la posición de la carga θ_L y el torque τ_0:

$$\frac{\theta_L}{\tau_0} = \frac{\frac{n}{(J_m S + B_m)S}}{1 + \frac{n}{(J_m S + B_m)S} * nS(J_L S + B_L)} \tag{2.23}$$

$$\frac{\theta_L}{\tau_0} = \frac{n}{[(J_m + n^2 J_L)S + (B_m + n^2 B_L)]S} \tag{2.24}$$

Donde:

$$\mathrm{n} = \frac{n_1}{n_2} \tag{2.25}$$

n_1: Número de dientes de la entrada del engrane (servomotor).

n_2: Número de dientes de la salida del engrane (carga).

n: Relación de engranes.

J_L: Momento de inercia de la carga.

B_L: Viscosidad de fricción de la carga.

$n^2 J_L$: Momento de inercia referida al servomotor.

$n^2 B_L$: Viscosidad referida al servomotor.

Se puede definir:

$$J_0 = J_m + n^2 \mathrm{J}_L \tag{2.26}$$

$$B_0 = B_m + n^2 B_L \tag{2.27}$$

Si reemplazamos (2.26) y (2.27) en (2.24) tendremos:

$$\frac{\theta_L}{\tau_0} = \frac{n}{(J_0 S + B_0)S} \tag{2.28}$$

Por lo tanto, el diagrama de bloques queda indicado en la Figura 2.11

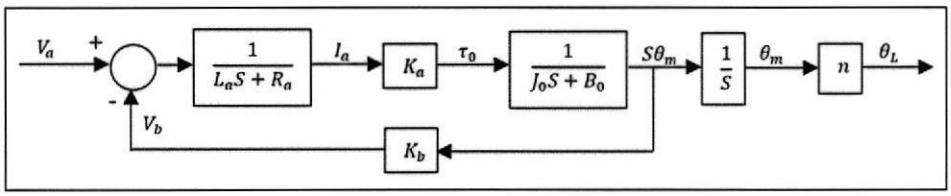


Figura 2.11 Diagrama de bloques del servomotor con carga

Con el diagrama anterior y lógica de bloques podemos obtener el modelo matemático el cual queda generalizado para un servomotor HS-311 y similar de carga variable debido a cambios en su inercia:

$$\frac{\theta_L}{V_a} = \frac{K_a\, n}{[(L_a S + R_a)(J_o S + B_o) + K_a K_b]S} \tag{2.29}$$

2.2.2 Modelamiento de la carga en el servomotor G1

La inercia que observará el servomotor G1 será la del brazo interno y la inercia de la carga que se colocará; el sistema se lo analizará girando en el vacío como se aprecia en la Figura 2.12, para encontrar la función de esta inercia se hará uso del teorema de los ejes paralelos y se considerará las paredes del brazo como un paralelepípedo de espesor $'e'$.

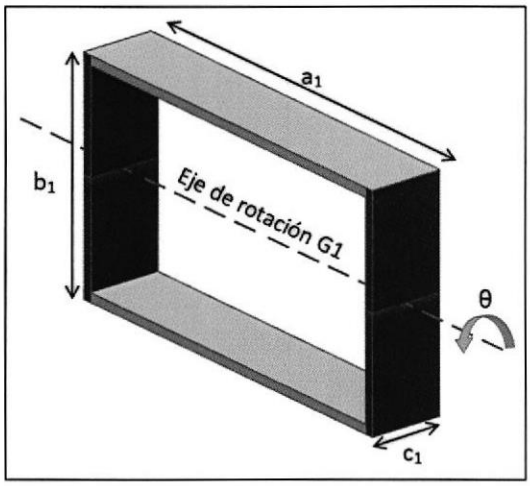


Figura 2.12 Brazo interno con eje de rotación θ

Sea un paralelepípedo con masa M_{pp}, centrado en su eje de rotación y con dimensiones tales como se muestran en la Figura 2.13, su inercia viene dada por la siguiente expresión:

$$\frac{M_{pp}}{12}(e^2 + c_1^2) \tag{2.30}$$

Según el teorema de ejes paralelos, la inercia del paralelepípedo anterior desplazado $\frac{b_1}{2}$ unidades desde el eje de rotación como se indica en la Figura 2.14 queda expresada de la siguiente manera:

$$\frac{M_{pp}}{12}(e^2 + c_1^2) + M_{pp}\left(\frac{b_1}{2}\right)^2 \tag{2.31}$$

Debido a que el brazo interno presenta dos paralelepípedos paralelos a su eje de rotación, la ecuación (2.31) se duplica para poder representar la inercia de un cuerpo presentado en la Figura 2.15

$$\frac{M_{pp}}{6}(e^2 + c_1^2) + \frac{M_{pp}\, b_1^2}{2} \tag{2.32}$$

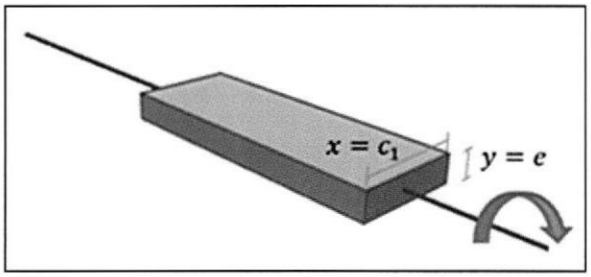


Figura 2.13 Inercia de un paralelepípedo con centro en el eje de rotación

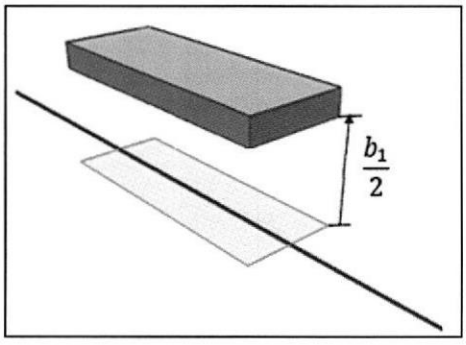


Figura 2.14 Inercia de un paralelepípedo paralelo al eje de rotación

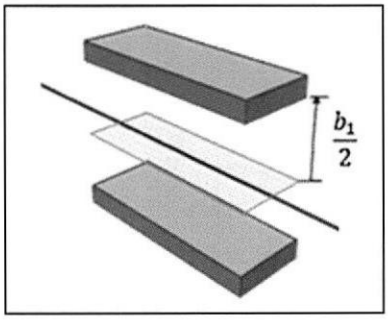


Figura 2.15 Inercia de dos paralelepípedos paralelos al eje de rotación

Para el análisis de los paralelepípedos transversales al eje de rotación con masa M_{pt} representados en la Figura 2.16 tenemos lo siguiente:

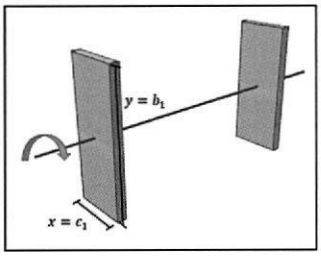


Figura 2.16 Inercia de un paralelepípedo transversal al eje de rotación

Como son dos elementos idénticos, la ecuación de su inercia se duplica, quedando de la siguiente manera:

$$\frac{M_{pt}}{6}(b_1^2 + c_1^2) \tag{2.33}$$

Sumando las ecuaciones (2.32) y (2.33) obtendremos la inercia total del brazo interno al girar en el vacío, sin la inercia que presenta la carga.

$$\frac{M_{pp}}{6}(e^2 + c_1^2) + \frac{M_{pp}\, b_1^2}{2} + \frac{M_{pt}}{6}(b_1^2 + c_1^2) \tag{2.34}$$

Considerando que la densidad del material de los brazos es $'\rho'$ y representando sus masas en función de la densidad y dimensiones obtenemos:

$$\frac{\rho(a_1 c_1 e)}{6}(c_1^2 + e^2) + \frac{\rho(a_1 c_1 e) b_1^2}{2} + \frac{\rho(b_1 c_1 e)}{6}(b_1^2 + c_1^2) \tag{2.35}$$

La carga que se colocará en el brazo interno viene representada en la Figura 2.17, que está compuesto por un paralelepípedo y un cilindro sólido

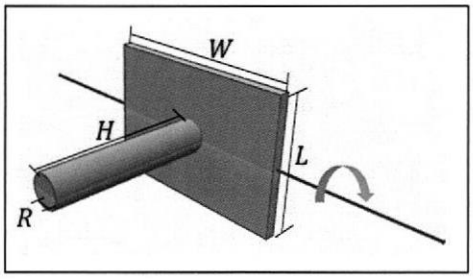


Figura 2.17 Carga del brazo interno

Partiendo de la ecuación (2.30) obtenemos la inercia que presenta el paralelepípedo:

$$\frac{\rho(eLW)}{12}(e^2 + L^2) \tag{2.36}$$

La inercia que tiene el cilindro viene representada por:

$$\rho(\pi R^2 H)\left(\frac{R^2}{4} + \frac{H^2}{3}\right) \tag{2.37}$$

Sumando las ecuaciones (2.36) y (2.37) obtenemos la inercia total de la carga J_l:

$$J_l = \frac{\rho(eLW)}{12}(e^2 + \mathrm{L}^2) + \rho(\pi R^2 H)\left(\frac{R^2}{4} + \frac{H^2}{3}\right) \tag{2.38}$$

En conclusión, la inercia I_{G_1} generada por el brazo interno y la carga que se coloque viene dada por las ecuaciones (2.35) y (2.38).

$$\begin{aligned} J_{G1} = {} & \frac{\rho(a_1 c_1 e)}{6}(c_1^2 + e^2) + \frac{\rho(a_1 c_1 e) b_1^2}{2} + \frac{\rho(b_1 c_1 e)}{6}(b_1^2 + c_1^2) \\ & + \frac{\rho(eLW)}{12}(e^2 + \mathrm{L}^2) + \rho(\pi R^2 H)\left(\frac{R^2}{4} + \frac{H^2}{3}\right) \end{aligned} \tag{2.39}$$

2.2.3 Modelamiento de la carga en el servomotor G2

Para encontrar la inercia que observa el servomotor G2 primero se realizará un proceso similar al planteado en la sección 2.2.2 para encontrar la inercia del brazo2 (medio), se considerará la densidad del brazo medio como $'\rho'$ y el eje de rotación será $'\psi'$ como se muestra en la Figura 2.18.

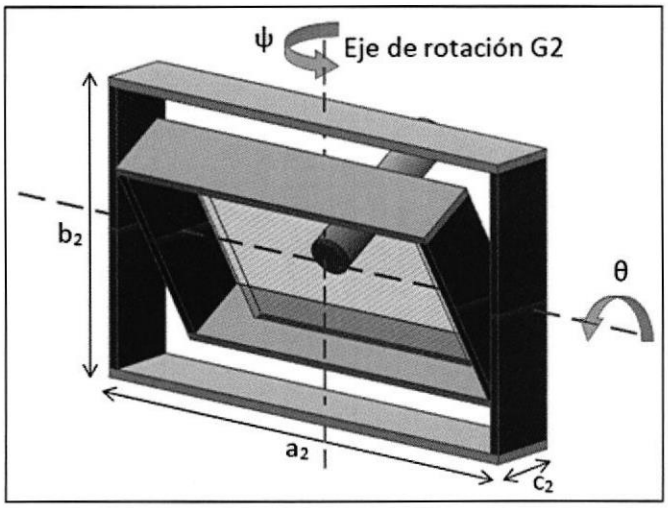


Figura 2.18: Brazo medio con eje de rotación ψ

Reemplazando el valor de las dimensiones y el valor de la masa en unidades de densidad y volumen en la ecuación (2.33), la inercia presentada por los paralelepípedos superior e inferior del brazo medio es:

$$\frac{\rho(a_2 c_2 e)}{6}(a_2^2 + c_2^2) \tag{2.40}$$

De igual manera se reemplaza la ecuación (2.32) con las dimensiones correspondientes para obtener la inercia presentada por los paralelepípedos laterales del brazo medio.

$$\frac{\rho(b_2 c_2 e)}{6}(e^2 + c_2^2) + \frac{\rho(b_2 c_2 e)\, a_2^2}{2} \tag{2.41}$$

Sumando las ecuaciones (2.41) y (2.40) obtendremos la inercia total que ejerce del brazo medio al girar en el vacío.

$$\frac{\rho(b_2 c_2 e)}{6}(e^2 + c_2^2) + \frac{\rho(b_2 c_2 e)\, a_2^2}{2} + \frac{\rho(a_2 c_2 e)}{6}(a_2^2 + c_2^2) \tag{2.42}$$

Adicionalmente el servomotor G2 se ve afectado por la inercia generada por el brazo interno, la carga colocada dentro del mismo, el servomotor G1 y una masa adicional ubicada de forma simétrica al servomotor G1 con referencia al eje de rotación ψ, la cual fue agregada para contrarrestar el peso del servomotor G1. Para este cálculo se analizará la inercia generada del brazo interno en las dos posiciones angulares limites (0° y 90°) representado por el ángulo θ.

Inercia del brazo interno con carga en θ = 0°.

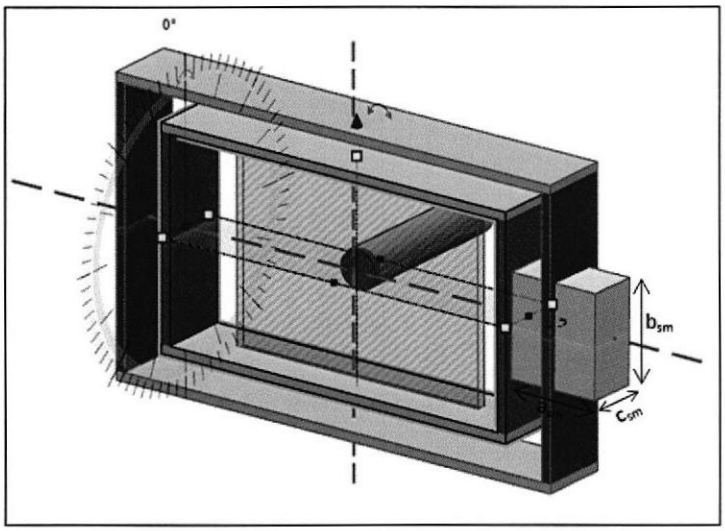


Figura 2.19: Brazo medio e interno en su posición inicial

El cálculo que ejerce el brazo interno se puede realizar a partir de la ecuación (2.42) únicamente cambiando las dimensiones correspondientes quedando de la siguiente forma:

$$\frac{\rho(b_1 c_1 e)}{6}(e^2 + c_1^2) + \frac{\rho(b_1 c_1 e)\, a_1^2}{2} + \frac{\rho(a_1 c_1 e)}{6}(a_1^2 + c_1^2) \qquad (2.43)$$

La inercia generada por la carga se obtiene a partir de la ecuación (2.38), cambiando la dimensión de la sección transversal al eje ψ, es decir reemplazar L^2 por W^2.

$$\frac{\rho(eLW)}{12}(e^2 + W^2) + \rho(\pi R^2 H)\left(\frac{R^2}{4} + \frac{H^2}{3}\right) \qquad (2.44)$$

Y, por último, la inercia generada por el servomotor G1 y la inercia que presenta la masa adicional son similares, debido a esto se duplicará la inercia del servomotor G1 la cual representará ambas inercias, las dimensiones del servomotor G1 vienen indicadas en la Figura 2.19 y cuya masa está representada por M_{SM}, según el teorema de ejes paralelos viene dado por:

$$\frac{M_{SM}}{6}\left(a_{\mathrm{SM}}^2 + c_{\mathrm{SM}}^2\right) + \frac{M_{SM}}{2}(a_1 + a_{SM})^2 \tag{2.45}$$

Por lo tanto, la inercia que siente el servomotor G2 cuando θ = 0° es igual a la suma de las ecuaciones (2.42), (2.43), (2.44) y (2.45).

$$\begin{aligned} J_{G2.1} = {} & \frac{\rho(b_2c_2e)}{6}(e^2 + c_2^2) + \frac{\rho(b_2c_2e)\,a_2^2}{2} + \frac{\rho(a_2c_2e)}{6}(a_2^2 + c_2^2) \\ & + \frac{\rho(b_1c_1e)}{6}(e^2 + c_1^2) + \frac{\rho(b_1c_1e)\,a_1^2}{2} + \frac{\rho(a_1c_1e)}{6}(a_1^2 + c_1^2) \\ & + \frac{\rho(eLW)}{12}(e^2 + W^2) + \rho(\pi R^2 H)\left(\frac{R^2}{4} + \frac{H^2}{3}\right) \\ & + \frac{M_{SM}}{6}\left(a_{\mathrm{SM}}^2 + c_{\mathrm{SM}}^2\right) + \frac{M_{SM}}{2}(a_1 + a_{SM})^2 \end{aligned} \tag{2.46}$$

Inercia del brazo interno con carga en θ = 90°.

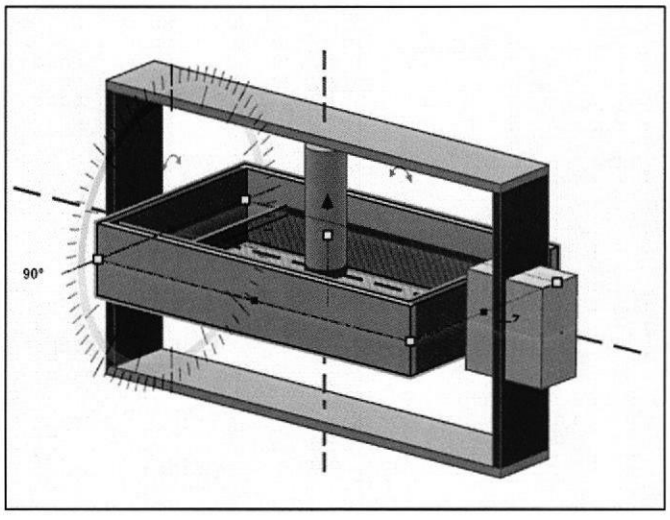


Figura 2.20: Brazo interno con desplazamiento angular de 90°

Tanto la inercia del brazo medio como la del servomotor G1 y la masa adicional se mantienen iguales, las inercias variables corresponden al brazo interno y su respectiva carga.

Los 4 paralelepípedos que forman el brazo interno generan una inercia similar a la ecuación (2.31), simplificando términos y representando la masa en términos de la densidad y volumen obtenemos la inercia total del brazo interno:

$$\frac{\rho(a_1c_1e)}{6}(e^2+a_1^2)+\frac{\rho(a_1c_1e)\,b_1^2}{2}+\frac{\rho(b_1c_1e)}{6}(e^2+b_1^2)+\frac{\rho(b_1c_1e)\,a_1^2}{2} \qquad (2.47)$$

La inercia de la carga en un ángulo de 90° como se muestra en la Figura 2.20 es la siguiente:

$$\frac{\rho(eLW)}{12}(W^2+\mathrm{L}^2)+\frac{\rho(\pi R^4H)}{2} \qquad (2.48)$$

La inercia que observa el servomotor G2 cuando θ = 90° viene dado por la adición de las ecuaciones (2.42), (2.45), (2.47) y (2.48).

$$\begin{aligned} J_{G2.2} = {} & \frac{\rho(b_2c_2e)}{6}(e^2+c_2^2)+\frac{\rho(b_2c_2e)\,a_2^2}{2}+\frac{\rho(a_2c_2e)}{6}(a_2^2+c_2^2) \\ & +\frac{M_{SM}}{6}\left(a_{\mathrm{SM}}^2+c_{\mathrm{SM}}^2\right)+\frac{M_{SM}}{2}(a_1+a_{SM})^2 \\ & +\frac{\rho(a_1c_1e)}{6}(e^2+a_1^2)+\frac{\rho(a_1c_1e)\,b_1^2}{2}+\frac{\rho(b_1c_1e)}{6}(e^2+b_1^2) \\ & +\frac{\rho(b_1c_1e)\,a_1^2}{2} \\ & +\frac{\rho(eLW)}{12}(W^2+\mathrm{L}^2)+\frac{\rho(\pi R^4H)}{2} \end{aligned} \qquad (2.49)$$

2.2.4 Modelamiento de la carga en el servomotor G3

La inercia que observa el servomotor G3 vendrá dada por la adición de las inercias que presentan los 3 paralelepípedos que conforman el brazo externo que gira en torno al eje 'ϕ' como se indica en la Figura 2.21, los brazos interno y medio, la carga del brazo interno y la inercia generada por los servomotores G1 y G2 cuyas dimensiones se muestran en la Figura 2.22.

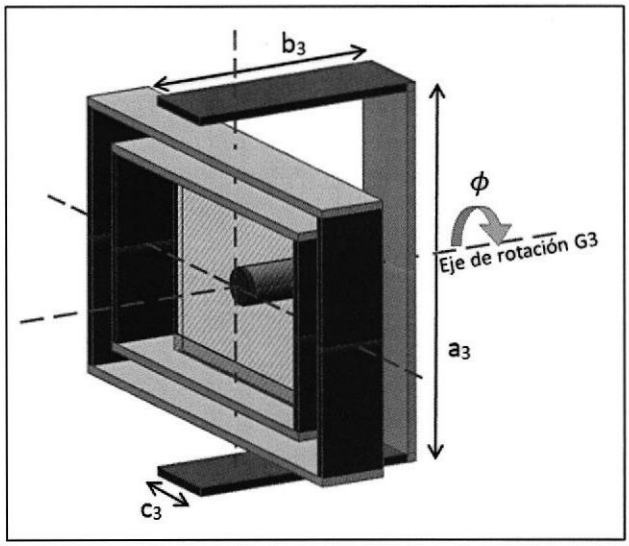


Figura 2.21: Brazo externo con eje de rotación ϕ

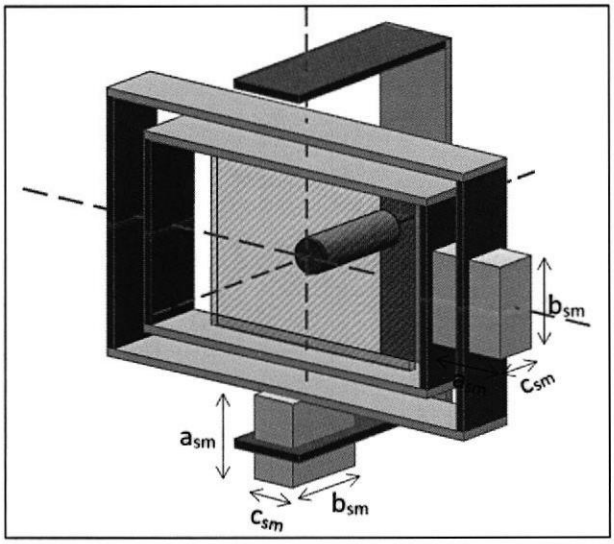


Figura 2.22: Plataforma inercial completa

Para calcular la inercia de las caras superior e inferior del brazo externo se hace uso de la ecuación (2.32) basada en el teorema de ejes paralelos.

$$\frac{\rho(b_3 c_3 e)}{6}(e^2 + c_3^2) + \frac{\rho(b_3 c_3 e)\, a_3^2}{2} \tag{2.50}$$

Para el cálculo de la inercia que presenta la única cara lateral se obtendrá a partir de la ecuación (2.33).

$$\frac{\rho(a_3 c_3 e)}{12}(a_3^2 + c_3^2) \tag{2.51}$$

La inercia presentada por el servomotor G2 se lo calcula por medio del teorema de ejes paralelos, similar a la ecuación (2.45)

$$\frac{M_{SM}}{12}\left(a_{\mathrm{SM}}^2 + c_{\mathrm{SM}}^2\right) + \frac{M_{SM}}{4}(b_2 + a_{SM})^2 \tag{2.52}$$

Sumando las ecuaciones (2.50), (2.51) y (2.52) obtendremos la inercia total del brazo externo y el servomotor G2 al girar en el vacío.

$$\frac{\rho(b_3 c_3 e)}{6}(e^2 + c_3^2) + \frac{\rho(b_3 c_3 e)\, a_3^2}{2} + \frac{\rho(a_3 c_3 e)}{12}(a_3^2 + c_3^2) + \frac{M_{SM}}{12}\left(a_{\mathrm{SM}}^2 + c_{\mathrm{SM}}^2\right) + \frac{M_{SM}}{4}(b_2 + a_{SM})^2 \tag{2.53}$$

Para el cálculo de la inercia presentada por el brazo medio, interno y su carga se presentará los 4 casos límites, es decir cuando θ=0°, θ=90°, ψ=0° y ψ=90°.

Inercia del brazo interno con carga en θ = 0° y brazo medio en ψ = 0°.

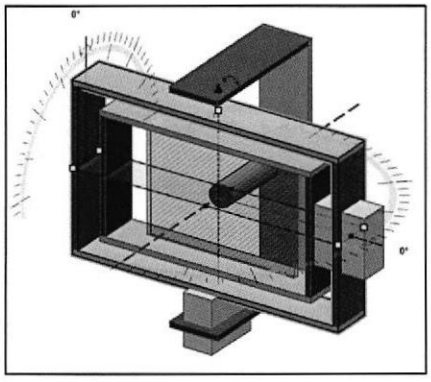


Figura 2.23: Brazo interno y medio con desplazamiento angular de 0°

La inercia del brazo interno mostrado en la Figura 2.23, junto con su carga es exactamente igual a la suma de las ecuaciones (2.47) y (2.48). La inercia presentada por el brazo medio es similar a la ecuación (2.47), quedando:

$$\frac{\rho(a_2c_2e)}{6}(e^2+a_2^2)+\frac{\rho(a_2c_2e)\,b_2^2}{2}+\frac{\rho(b_2c_2e)}{6}(e^2+b_2^2)+\frac{\rho(b_2c_2e)\,a_2^2}{2} \qquad (2.54)$$

Y para la inercia del servomotor G1 se aplica de nuevo el teorema de ejes paralelos y multiplicando por un factor de 2, debido a la inercia de la masa de contrapeso:

$$\frac{M_{SM}}{6}\left(a_{\mathrm{SM}}^2+b_{\mathrm{SM}}^2\right)+\frac{M_{SM}}{2}(a_1+a_{SM})^2 \qquad (2.55)$$

Finalmente, la inercia que observa el servomotor G3 con θ = 0° ψ = 0° equivale a la suma de las ecuaciones (2.47), (2.48), (2.53), (2.54) y (2.55).

$$\begin{aligned}
J_{G3.1} = {} & \frac{\rho(a_1c_1e)}{6}(e^2+a_1^2)+\frac{\rho(a_1c_1e)\,b_1^2}{2}+\frac{\rho(b_1c_1e)}{6}(e^2+b_1^2)+\frac{\rho(b_1c_1e)\,a_1^2}{2} \\
& +\frac{\rho(eLW)}{12}(W^2+\mathrm{L}^2)+\frac{\rho(\pi R^4H)}{2} \\
& +\frac{\rho(b_3c_3e)}{6}(e^2+c_3^2)+\frac{\rho(b_3c_3e)\,a_3^2}{2}+\frac{\rho(a_3c_3e)}{12}(a_3^2+c_3^2) \\
& +\frac{M_{SM}}{12}\left(a_{\mathrm{SM}}^2+c_{\mathrm{SM}}^2\right)+\frac{M_{SM}}{4}(b_2+a_{SM})^2 \qquad (2.56) \\
& +\frac{\rho(a_2c_2e)}{6}(e^2+a_2^2)+\frac{\rho(a_2c_2e)\,b_2^2}{2}+\frac{\rho(b_2c_2e)}{6}(e^2+b_2^2) \\
& +\frac{\rho(b_2c_2e)\,a_2^2}{2} \\
& +\frac{M_{SM}}{6}\left(a_{\mathrm{SM}}^2+b_{\mathrm{SM}}^2\right)+\frac{M_{SM}}{2}(a_1+a_{SM})^2
\end{aligned}$$

Inercia del brazo interno con carga en θ = 90° y brazo medio en ψ = 0°.

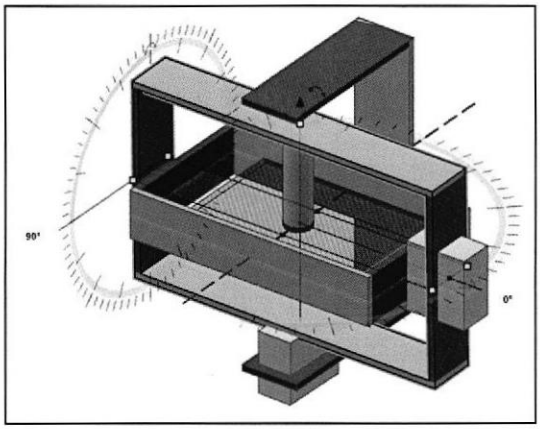


Figura 2.24: Brazo interno con desplazamiento angular de 90° y brazo medio con 0°

Para este caso tanto le inercia del servomotor G1 y la del brazo medio no cambian en comparación al cálculo anterior; en cuanto a la inercia del brazo interno y su carga es exactamente igual a las ecuaciones (2.43) y (2.44) respectivamente. Por lo tanto, la inercia que siente el servomotor G3 (Véase Figura 2.24) en estas condiciones viene dado por la adición de las ecuaciones (2.43), (2.44), (2.54) y (2.55).

$$
\begin{aligned}
J_{G3.2} = {} & \frac{\rho(b_1c_1e)}{6}(e^2+c_1^2)+\frac{\rho(b_1c_1e)\,a_1^2}{2}+\frac{\rho(a_1c_1e)}{6}(a_1^2+c_1^2) \\
& +\frac{\rho(eLW)}{12}(e^2+W^2)+\rho(\pi R^2H)\left(\frac{R^2}{4}+\frac{H^2}{3}\right) \\
& +\frac{\rho(a_1c_1e)}{6}(e^2+a_1^2)+\frac{\rho(a_1c_1e)\,b_1^2}{2}+\frac{\rho(b_1c_1e)}{6}(e^2+b_1^2)+\frac{\rho(b_1c_1e)\,a_1^2}{2} \\
& +\frac{\rho(eLW)}{12}(W^2+\mathrm{L}^2)+\frac{\rho(\pi R^4H)}{2} \\
& +\frac{\rho(b_3c_3e)}{6}(e^2+c_3^2)+\frac{\rho(b_3c_3e)\,a_3^2}{2}+\frac{\rho(a_3c_3e)}{12}(a_3^2+c_3^2)+\frac{M_{SM}}{12}\left(a_{\mathrm{SM}}^2+c_{\mathrm{SM}}^2\right) \\
& +\frac{M_{SM}}{4}(b_2+a_{SM})^2 \\
& +\frac{\rho(a_2c_2e)}{6}(e^2+a_2^2)+\frac{\rho(a_2c_2e)\,b_2^2}{2}+\frac{\rho(b_2c_2e)}{6}(e^2+b_2^2)+\frac{\rho(b_2c_2e)\,a_2^2}{2}
\end{aligned}
\tag{2.57}
$$

$$+\frac{M_{SM}}{6}\left(a_{\mathrm{SM}}^2+b_{\mathrm{SM}}^2\right)+\frac{M_{SM}}{2}(a_1+a_{SM})^2$$

Inercia del brazo interno con carga en θ = 0° y brazo medio en ψ = 90°.

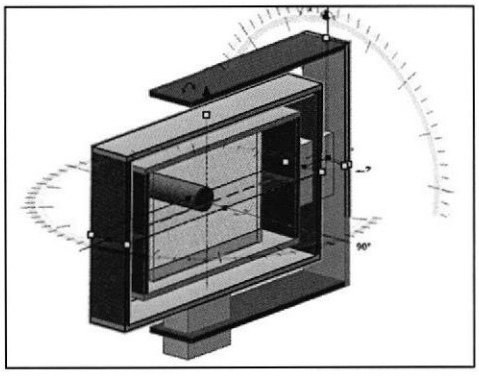


Figura 2.25: Brazo interno con desplazamiento angular de 0° y brazo medio con 90°

La inercia que genera el brazo interno junto a su carga (Véase Figura 2.25) es igual a la inercia que siente el servomotor G1 (J_{G1}), modificando las dimensiones del brazo medio en (2.35) se obtiene su inercia correspondiente quedando:

$$\frac{\rho(a_2c_2e)}{6}(c_2^2+e^2)+\frac{\rho(a_2c_2e)b_2^2}{2}+\frac{\rho(b_2c_2e)}{6}(b_2^2+c_2^2) \tag{2.58}$$

La inercia generada por el servomotor G1 y la masa de contrapeso viene dado por:

$$\frac{M_{SM}}{6}\left(b_{\mathrm{SM}}^2+c_{\mathrm{SM}}^2\right) \tag{2.59}$$

Como resultado final la inercia que siente el servomotor G3 es igual a:

$$\begin{aligned} J_{G3.3} = {} & \frac{\rho(a_1c_1e)}{6}(c_1^2+e^2)+\frac{\rho(a_1c_1e)b_1^2}{2}+\frac{\rho(b_1c_1e)}{6}(b_1^2+c_1^2) \\ & +\frac{\rho(eLW)}{12}(e^2+\mathrm{L}^2)+\rho(\pi R^2H)\left(\frac{R^2}{4}+\frac{H^2}{3}\right) \\ & +\frac{\rho(b_3c_3e)}{6}(e^2+c_3^2)+\frac{\rho(b_3c_3e)\,a_3^2}{2}+\frac{\rho(a_3c_3e)}{12}(a_3^2+c_3^2) \\ & +\frac{M_{SM}}{12}(a_{\mathrm{SM}}^2+c_{\mathrm{SM}}^2)+\frac{M_{SM}}{4}(b_2+a_{SM})^2 \end{aligned} \tag{2.60}$$

$$+\frac{\rho(a_2c_2e)}{6}(c_2^2+e^2)+\frac{\rho(a_2c_2e)b_2^2}{2}+\frac{\rho(b_2c_2e)}{6}(b_2^2+c_2^2)+\frac{M_{SM}}{6}(b_{\mathrm{SM}}^2+c_{\mathrm{SM}}^2)$$

Inercia del brazo interno con carga en θ=90° y brazo medio en ψ=90°.

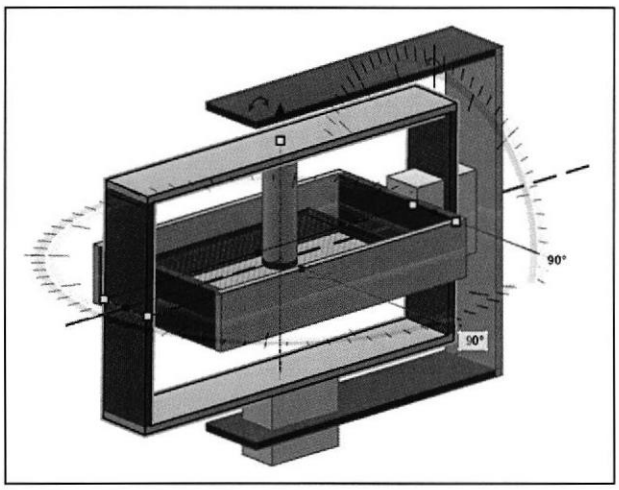


Figura 2.26: Brazo interno y medio con desplazamiento angular de 90°

Cuando ψ=90°, el eje de rotación del brazo interno coincide con el eje de rotación del brazo externo (Véase Figura 2.26), debido a esto para cualquier variación del ángulo θ y manteniéndose constante el valor de ψ en 90° la inercia de todo el sistema no se ve afectada. Por ende $J_{G3.4} = J_{G3.3}$.

2.2.5 Cálculo de los parámetros internos del servomotor.

El fabricante del servo-motor Hitec HS-311 nos brinda de forma implícita las constantes del motor que nos serán útiles para el modelo matemático obtenido en secciones anteriores y personalizar el modelo de la planta que se está implementando, a continuación, se muestran los parámetros básicos del servomotor:

Sistema de Control por Anchura de Pulso. 1,5 ms al centro

Tensión de funcionamiento 4,8V a 6 V

Velocidad a 6V 0,15 s /60 grados sin carga

Fuerza a 6V 3,53 Kg • cm

Corriente en reposo 7,7 mA

Zona Neutra 5 µs

Rango Trabajo 1100 a 1900 µs

Dimensiones 40 x 20 x 36,5 mm

Peso 43 g

Rodamiento Principal Plástico

Engranajes Plástico

Datos de operación del servo

Corriente en funcionamiento 180 mA sin carga

Corriente Máxima 800 mA

Ra 7,6 ohm

Cálculo de velocidad y aceleración angular

La velocidad angular máxima proporcionada por el fabricante nos la da en revoluciones por minuto que es alcanzada en un tiempo de 0.23s, sin embargo, nuestros cálculos son realizados con Sistema Internacional, obteniéndose:

$$w = 9590\,rpm\,x\frac{2\pi\,rad}{1\,rev}x\frac{1\,min}{60\,s} = 1004.26\,[rad/s] \tag{2.61}$$

$$\alpha = \frac{w}{t} = \frac{1004.26}{0.23\,s} = 4366.35\,[rad/s^2] \tag{2.62}$$

Cálculo de coeficiente de fricción B_0

τ : torque del motor

ω : velocidad angular del motor

$\tau = 0.38x10^{-3}[N \cdot m]$

$$B_0 = \frac{\tau}{\omega} = \frac{0.38x10^{-3}}{1004.26} = 0.37838\, x10^{-6}\left[\frac{N \cdot m \cdot s}{\text{rad}}\right] \quad (2.63)$$

Cálculo de inercia J

Se determina por J = τ / α

$$J = \frac{\tau}{\alpha} = \frac{0.38x10^{-3}}{4366.35} = 0.087029\ 10^{-6}[kg \cdot m^2] \quad (2.64)$$

Cálculo de la constante Ka

Según la ecuación (2.5), se puede despejar la constante K_a y se obtiene:

$$K_a = \frac{\tau}{i_a} = \frac{0.38x10^{-3}}{0.18} = 2.1111x10^{-3}\left[\frac{N \cdot m}{A}\right] \quad (2.65)$$

Cálculo de la constante Kb

Para el cálculo de K_b es necesario conocer el valor de la fuerza contra-electromotriz, el cual es calculado de forma estimada por la siguiente ecuación (2.66) usando los valores más óptimos, donde:

V_a: voltaje administrado al motor

R_a: Resistencia de la armadura del motor

I_a: Corriente de armadura del motor

$V_b = V_a$ – la Ra

$$e_b = 4.5 - (0.184)(7.6) = 3.1016\,[V] \quad (2.66)$$

Por ende. La constante Kb queda:

$$K_b = \frac{e_b}{\omega} = \frac{3.1016}{1004.26} = 3.08844\, x10^{-3}\left[\frac{V \cdot rad}{s}\right] \quad (2.67)$$

Cálculo de la relación n del tren de engranaje

Para calcular la relación n del tren de engranajes se debe primero analizar cómo está conformado dentro del servomotor (Véase Figura 2.27):

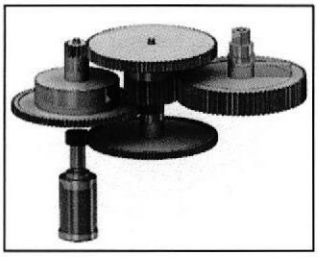


Figura 2.27: Tren de engranes HS311 [11]

Y de qué manera se relaciona cada uno de sus diámetros con su secuencia tal como se observa en la Figura 2.28:

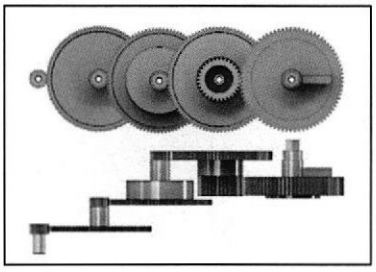


Figura 2.28: Relacion de dientes del tren de engranajes HS311 [11]

La Tabla 1 a continuación muestra el número de engrane y diámetro (cm).

Engrane	Magnitud	Unidad
n1	0.32	*cm*
n2	1.57	*cm*
n3	0.38	*cm*
n4	1.53	*cm*
n5	0.51	*cm*
n6	1.45	*cm*
n7	0.75	*cm*
n8	1.71	*cm*

Tabla 1: Diámetro de los engranes internos del servomotor

Para calcular la relación de transformación del tren de engranajes se tiene:

$$n = n_a * n_b * n_c * n_d \tag{2.68}$$

Ahora calculamos sus relaciones de 2 en 2:

$$n_a = \frac{n_1}{n_2} = \frac{0.32}{1.57} \tag{2.69}$$

$$n_b = \frac{n_3}{n_4} = \frac{0.38}{1.53} \tag{2.70}$$

$$n_c = \frac{n_5}{n_6} = \frac{0.51}{1.45} \tag{2.71}$$

$$n_d = \frac{n_7}{n_8} = \frac{0.75}{1.71} \tag{2.72}$$

$$n = \frac{0.32}{1.57} x \frac{0.38}{1.53} x \frac{0.51}{1.45} x \frac{0.75}{1.71} = \frac{320}{40977} = 0.0078 \tag{2.73}$$

Entonces n es aproximadamente:

$$n \approx \frac{1}{128}$$

2.2.6 Modelamiento especifico de los 3 brazos de la plataforma.

Para poder conocer los valores numéricos de las ecuaciones necesario remplazar las constantes de la ecuación, sin embargo, se presentas 7 casos diferentes por lo que J_{Gi} cambiará de valor según estos eventos, de tal manera que obtendremos 7 funciones de transferencia diferentes.

$$\frac{\theta_L}{V_a} = \frac{K_a\, n}{[(L_a S + R_a)(J_{Gi} S + B_o) + K_a K_b]S} \tag{2.74}$$

El listado de las constantes se muestra en la Tabla 2.

Nombre	Magnitud	Unidad
a1	0.15	*m*
b1	0.087	*m*
c1	0.028	*m*
a2	0.189	*m*
b2	0.116	*m*
c2	0.033	*m*
a3	0.16	*m*
b3	0.10	*m*
c3	0.034	*m*
e	0.0033	*m*
Msm	0.043	*Kg*
asm	0.0198	*m*

bsm	0.0398	*m*
csm	0.0363	*m*
W	0.105	*m*
L	0.085	*m*
H	0.07	*m*
R	0.0075	*m*
P_{Madera}	150	kg/m^3
R_a	7.6	Ω
K_a	2.1111×10^{-3}	$[N \cdot m/A]$
K_b	3.08844×10^{-3}	$[V \cdot rad/s]$
B_0	0.37838×10^{-6}	$[N \cdot m \cdot s/rad]$

Tabla 2: Constantes para cálculo de inercia y forma canónica del modelo

Análisis brazo 1:

$$\frac{\theta_L}{V_a} = \frac{K_a\, n}{[(L_a S + R_a)(J_{G1} S + B_o) + K_a K_b] S} \tag{2.75}$$

Para poder realizar el cálculo de la Inercia del brazo 1 se procede a reemplazar los valores de la **¡Error! No se encuentra el origen de la referencia.** en la ecuación (2.39)

$$J_{G1} = \frac{(150)(0.15 * 0.028 * 0.0033)}{6}(0.028^2 + 0.0033^2) + \frac{(150)(0.15 * 0.028 * 0.0033)(0.087)^2}{2}$$
$$+ \frac{(150)(0.087 * 0.028 * 0.0033)}{6}(0.087^2 + 0.028^2)$$
$$+ \frac{(150)(0.0033 * 0.085 * 0.105)}{12}(0.0033^2 + 0.085^2)$$
$$+ (150)(3.14 * 0.0075^2 * 0.07)\left(\frac{0.0075^2}{4} + \frac{0.07^2}{3}\right)$$

$$J_{G1} = 0.275x10^{-6} + 7.867x10^{-6} + 1.678x10^{-6} + 2.663x10^{-6} + 3.055x10^{-6}$$

$$J_{G1} = 15.538x10^{-6}\ [kg \cdot m^2]$$

Dado el valor de su inercia y la ecuación general podemos obtener:

$$\frac{\theta_L}{V_a} = \frac{(2.1111x10^{-3})\left(\frac{1}{128}\right)}{[(L_a S + 7.6)((15.538x10^{-6})S + 0.37838\, x10^{-6}) + (2.1111x10^{-3})(3.08844\, x10^{-3})]S}$$

Asumiendo que el valor de la inductancia de armadura L_a es cero, tenemos:

$$\frac{\theta_L}{V_a} = \frac{16.5x10^{-6}}{118.08x10^{-6}S^2 + 9.396\, x10^{-6}S} \tag{2.76}$$

Y obteniendo su forma canónica:

$$\frac{\theta_L}{V_a} = \frac{0.14}{S^2 + 0.08\, S} \tag{2.77}$$

Análisis brazo 2:

$$\frac{\theta_L}{V_a} = \frac{K_a\, n}{[(L_a S + R_a)(J_{G2} S + B_o) + K_a K_b] S} \tag{2.78}$$

Para el brazo 2 se tendrán dos cálculos de Inercias debido a las diferentes posiciones que puede tener su carga, las cuales serán cuando ésta esté en θ = 0° y θ = 90°.

Inercia del brazo interno con carga en θ = 0°.

Se procede a reemplazar los valores de la tabla 2 en la ecuación (2.46)

$$
\begin{aligned}
J_{G2.1} &= \frac{(150)(0.116 * 0.033 * 0.0033)}{6}(0.0033^2 + 0.033^2) \\
&+ \frac{(150)(0.116 * 0.033 * 0.0033)(0.189)^2}{2} \\
&+ \frac{(150)(0.189 * 0.033 * 0.0033)}{6}(0.189^2 + 0.033^2) \\
&+ \frac{(150)(0.087 * 0.028 * 0.0033)}{6}(0.0033^2 + 0.028^2) \\
&+ \frac{(150)(0.087 * 0.028 * 0.0033)\ (0.15^2)}{2} \\
&+ \frac{(150)(0.15 * 0.028 * 0.0033)}{6}(0.15^2 + 0.028^2) \\
&+ \frac{(150)(0.0033 * 0.085 * 0.105)}{12}(0.0033^2 + 0.105^2) \\
&+ (150)(3.14 * 0.0075^2 * 0.07)\left(\frac{0.0075^2}{4} + \frac{0.07^2}{3}\right) \\
&+ \frac{0.043}{6}(0.0198^2 + 0.0363^2) + \frac{0.043}{2}(0.15 + 0.0198)^2
\end{aligned}
$$

$$
\begin{aligned}
J_{G2.1} &= 0.347x10^{-6} + 33.843x10^{-6} + 18.940x10^{-6} + 0.159x10^{-6} + 13.565x10^{-6} \\
&+ 8.067x10^{-6} + 4.062x10^{-6} + 3.055x10^{-6} + 12.253x10^{-6} + 619.888x10^{-6}
\end{aligned}
$$

$$J_{G2.1} = 714.179x10^{-6}[kg \cdot m^2]$$

$$\frac{\theta_L}{V_a} = \frac{(2.1111x10^{-3})\left(\frac{1}{128}\right)}{[(L_aS + 7.6)((714.179x10^{-6})S + 0.37838\,x10^{-6}) + (2.1111x10^{-3})(3.08844\,x10^{-3})]S}$$

Asumiendo que el valor de la inductancia de armadura L_a es cero, tenemos:

$$\frac{\theta_L}{V_a} = \frac{16.5x10^{-6}}{5.428x10^{-3}S^2 + 9.396\,x10^{-6}S} \tag{2.79}$$

Y obteniendo su forma canónica:

$$\frac{\theta_L}{V_a} = \frac{3.04x10^{-3}}{S^2 + 1.73x10^{-3}\,S} \tag{2.80}$$

Inercia del brazo interno con carga en θ = 90°.

Se procede a reemplazar los valores de la tabla 2 en la ecuación (2.49)

$$J_{G2.2} = \frac{(150)(0.116 * 0.033 * 0.0033)}{6}((0.0033)^2 + (0.033)^2) + \frac{(150)(0.116 * 0.033 * 0.0033)\,(0.189)^2}{2} + \frac{(150)(0.189 * 0.033 * 0.0033)}{6}((0.189)^2 + (0.033)^2) + \frac{0.043}{6}((0.0198)^2 + (0.0363)^2) + \frac{0.043}{2}(0.15 + 0.0198)^2 + \frac{(150)(0.15 * 0.028 * 0.0033)}{6}((0.0033)^2 + (0.15)^2) + \frac{(150)(0.15 * 0.028 * 0.0033)\,(0.087)^2}{2} + \frac{(150)(0.087 * 0.028 * 0.0033)}{6}((0.0033)^2 + (0.087)^2) + \frac{(150)(0.087 * 0.028 * 0.0033)\,(0.15)^2}{2} + \frac{(150)(0.0033 * 0.085 * 0.105)}{12}((0.105)^2 + (0.085)^2) + \frac{(150)(3.1416 * (0.0075)^4 * 0.07)}{2}$$

$$J_{G2.2} = 0.347x10^{-6} + 33.843x10^{-6} + 18.940x10^{-6} + 12.253x10^{-6} + 619.888x10^{-6} + 7.800x10^{-6} + 7.867x10^{-6} + 1.523x10^{-6} + 13.565x10^{-6} + 6.718x10^{-6} + 0.052x10^{-6}$$

$$J_{G2.2} = 722.796x10^{-6}[kg \cdot m^2]$$

$$\frac{\theta_L}{V_a} = \frac{(2.1111x10^{-3})\left(\frac{1}{128}\right)}{[(L_aS + 7.6)((722.796x10^{-6})S + 0.37838\,x10^{-6}) + (2.1111x10^{-3})(3.08844\,x10^{-3})]S}$$

Asumiendo que el valor de la inductancia de armadura L_a es cero, tenemos:

$$\frac{\theta_L}{V_a} = \frac{16.5x10^{-6}}{5.493x10^{-3}S^2 + 9.396\,x10^{-6}S} \tag{2.81}$$

Y obteniendo su forma canónica:

$$\frac{\theta_L}{V_a} = \frac{3.00x10^{-3}}{S^2 + 1.71x10^{-3}\,S} \tag{2.82}$$

Análisis brazo 3:

$$\frac{\theta_L}{V_a} = \frac{K_a\, n}{[(L_a S + R_a)(J_{G3} S + B_o) + K_a K_b] S} \tag{2.83}$$

Para el brazo 3 se tendrán cuatro inercias las cuales se analizarán en los 4 casos límites, es decir cuando θ=0°, θ=90°, ψ=0° y ψ=90°.

Inercia de brazo interno con carga en θ = 0° y brazo medio en ψ = 0°.

$$J_{G3.1} = \frac{(150)(0.15 * 0.028 * 0.0033)}{6}((0.0033)^2 + (0.15)^2) + \frac{(150)(0.15 * 0.028 * 0.0033)\,(0.087)^2}{2}$$
$$+ \frac{(150)(0.087 * 0.028 * 0.0033)}{6}((0.0033)^2 + (0.087)^2) + \frac{(150)(0.087 * 0.028 * 0.0033)\,(0.15)^2}{2}$$
$$+ \frac{(150)(0.0033 * 0.085 * 0.105)}{12}((0.105)^2 + (0.085)^2) + \frac{(150)(3.1416 * (0.0075)^4 * 0.07)}{2}$$
$$+ \frac{(150)(0.10 * 0.034 * 0.0033)}{6}((0.0033)^2 + (0.034)^2) + \frac{(150)(0.10 * 0.034 * 0.0033)\,(0.16)^2}{2}$$
$$+ \frac{(150)(0.16 * 0.034 * 0.0033)}{12}((0.16)^2 + (0.034)^2) + \frac{0.043}{12}((0.0198)^2 + (0.0363)^2)$$
$$+ \frac{0.043}{4}(0.116 + 0.0198)^2 + \frac{(150)(0.189 * 0.033 * 0.0033)}{6}((0.0033)^2 + (0.189)^2)$$
$$+ \frac{(150)(0.189 * 0.033 * 0.0033)\,(0.116)^2}{2} + \frac{(150)(0.116 * 0.033 * 0.0033)}{6}((0.0033)^2 + (0.116)^2)$$
$$+ \frac{(150)(0.116 * 0.033 * 0.0033)\,(0.189)^2}{2} + \frac{0.043}{6}((0.0198)^2 + (0.0398)^2) + \frac{0.043}{2}(0.15 + 0.0198)^2$$

$$J_{G3.1} = 7.800x10^{-6} + 7.867x10^{-6} + 1.523x10^{-6} + 13.565x10^{-6} + 6.718x10^{-6} + 0.052x10^{-6} + 0.327x10^{-6} + 21.542x10^{-6} + 6.004x10^{-6} + 6.126x10^{-6} + 198.247x10^{-6} + 18.385x10^{-6} + 20.771x10^{-6} + 4.252x10^{-6} + 33.843x10^{-6} + 14.161x10^{-6} + 619.888x10^{-6}$$

$$J_{G3.1} = 981.071x10^{-6}[kg \cdot m^2]$$

$$\frac{\theta_L}{V_a} = \frac{(2.1111x10^{-3})\left(\frac{1}{128}\right)}{[(L_a S + 7.6)((981.071x10^{-6})S + 0.37838x10^{-6}) + (2.1111x10^{-3})(3.08844\,x10^{-3})]S}$$

Asumiendo que el valor de la inductancia de armadura L_a es cero, tenemos:

$$\frac{\theta_L}{V_a} = \frac{16.5x10^{-6}}{7.456x10^{-3}S^2 + 9.396\,x10^{-6}S} \tag{2.84}$$

obteniendo su forma canónica:

$$\frac{\theta_L}{V_a} = \frac{2.21x10^{-3}}{S^2 + 1.26x10^{-3}\,S} \tag{2.85}$$

Inercia de brazo interno con carga en θ = 90° y brazo medio en ψ = 0°.

$$J_{G3.2} = \frac{(150)(0.087 * 0.028 * 0.0033)}{6}((0.0033)^2 + (0.028)^2) + \frac{(150)(0.087 * 0.028 * 0.0033)\,(0.15)^2}{2}$$
$$+ \frac{(150)(0.15 * 0.028 * 0.0033)}{6}(0.15^2 + 0.028^2) + \frac{(150)(0.0033 * 0.085 * 0.105)}{12}((0.0033)^2 + (0.105)^2)$$
$$+ (150)(3.1416 * (0.0075)^2 * 0.07)\left(\frac{(0.0075)^2}{4} + \frac{(0.07)^2}{3}\right)$$
$$+ \frac{(150)(0.15 * 0.028 * 0.0033)}{6}((0.0033)^2 + (0.15)^2) + \frac{(150)(0.15 * 0.028 * 0.0033)\,(0.087)^2}{2}$$
$$+ \frac{(150)(0.087 * 0.028 * 0.0033)}{6}((0.0033)^2 + (0.087)^2) + \frac{(150)(0.087 * 0.028 * 0.0033)\,(0.15)^2}{2}$$
$$+ \frac{(150)(0.0033 * 0.085 * 0.105)}{12}((0.105)^2 + (0.085)^2) + \frac{(150)(3.1416 * (0.0075)^4 * 0.07)}{2}$$
$$+ \frac{(150)(0.10 * 0.034 * 0.0033)}{6}((0.0033)^2 + (0.034)^2) + \frac{(150)(0.10 * 0.034 * 0.0033)\,(0.16)^2}{2}$$
$$+ \frac{(150)(0.16 * 0.034 * 0.0033)}{12}(0.16^2 + 0.034^2) + \frac{0.043}{12}((0.0198)^2 + (0.0363)^2)$$
$$+ \frac{0.043}{4}(0.116 + 0.0198)^2 + \frac{(150)(0.189 * 0.033 * 0.0033)}{6}((0.0033)^2 + (0.189)^2)$$
$$+ \frac{(150)(0.189 * 0.033 * 0.0033)\,(0.116)^2}{2} + \frac{(150)(0.116 * 0.033 * 0.0033)}{6}((0.0033)^2 + (0.116)^2)$$
$$+ \frac{(150)(0.116 * 0.033 * 0.0033)\,(0.189)^2}{2} + \frac{0.043}{6}((0.0198)^2 + (0.0398)^2) + \frac{0.043}{2}(0.15 + 0.0198)^2$$

$$J_{G3.2} = 0.159x10^{-6} + 13.565x10^{-6} + 8.067x10^{-6} + 4.062x10^{-6} + 3.055x10^{-6}$$
$$+ 7.800x10^{-6} + 7.867x10^{-6} + 1.523x10^{-6} + 13.565x10^{-6} + 6.718x10^{-6}$$
$$+ 0.052x10^{-6} + 0.327x10^{-6} + 21.542x10^{-6} + 6.004x10^{-6} + 6.126x10^{-6}$$
$$+ 198.247x10^{-6} + 18.385x10^{-6} + 20.771x10^{-6} + 4.252x10^{-6} + 33.843x10^{-6}$$
$$+ 14.161x10^{-6} + 619.88x10^{-6}$$

$$J_{G3.2} = 1010[kg \cdot m^2]$$

$$\frac{\theta_L}{V_a} = \frac{(2.1111x10^{-3})\left(\frac{1}{128}\right)}{[(L_aS + 7.6)((1009x10^{-6})S + 0.37838\,x10^{-6}) + (2.1111x10^{-3})(3.08844\,x10^{-3})]S}$$

Asumiendo que el valor de la inductancia de armadura L_a es cero, tenemos:

$$\frac{\theta_L}{V_a} = \frac{16.5x10^{-6}}{7.676x10^{-3}S^2 + 9.396\,x10^{-6}S} \tag{2.86}$$

Y obteniendo su forma canónica:

$$\frac{\theta_L}{V_a} = \frac{2.15x10^{-3}}{S^2 + 1.22x10^{-3}\,S} \tag{2.87}$$

Inercia del brazo interno con carga en θ = 0° y brazo medio en ψ = 90°.

$$J_{G3.3} = \frac{(150)(0.15 * 0.028 * 0.0033)}{6}((0.028)^2 + (0.0033)^2) + \frac{(150)(0.15 * 0.028 * 0.0033)(0.087)^2}{2} + \frac{(150)(0.087 * 0.028 * 0.0033)}{6}((0.087)^2 + (0.028)^2) + \frac{(150)(0.0033 * 0.085 * 0.105)}{12}((0.0033)^2 + (0.085)^2) + (150)(3.1416 * (0.0075)^2 * 0.07)\left(\frac{(0.0075)^2}{4} + \frac{(0.07)^2}{3}\right) + \frac{(150)(0.10 * 0.034 * 0.0033)}{6}((0.0033)^2 + (0.034)^2) + \frac{(150)(0.10 * 0.034 * 0.0033)\ (0.16)^2}{2} + \frac{(150)(0.16 * 0.034 * 0.0033)}{12}(0.16^2 + 0.034^2) + \frac{0.043}{12}((0.0198)^2 + (0.0363)^2) + \frac{0.043}{4}(0.116 + 0.0198)^2 + \frac{(150)(0.189 * 0.033 * 0.0033)}{6}((0.033)^2 + (0.0033)^2) + \frac{(150)(0.189 * 0.033 * 0.0033)(0.116)^2}{2} + \frac{(150)(0.116 * 0.033 * 0.0033)}{6}((0.116)^2 + (0.033)^2) + \frac{0.043}{6}((0.0398)^2 + (0.0363)^2)$$

$$J_{G3.3} = 0.275x10^{-6} + 7.867x10^{-6} + 1.678x10^{-6} + 2.663x10^{-6} + 3.055x10^{-6} + 0.327x10^{-6} + 21.542x10^{-6} + 6.004x10^{-6} + 6.126x10^{-6} + 198.247x10^{-6} + 0.565x10^{-6} + 20.771x10^{-6} + 4.593x10^{-6} + 20.795x10^{-6}$$

$$J_{G3.3} = 294.508x10^{-6}[kg \cdot m^2]$$

$$\frac{\theta_L}{V_a} = \frac{(2.1111x10^{-3})\left(\frac{1}{128}\right)}{[(L_aS + 7.6)((294.508x10^{-6})S + 0.37838\,x10^{-6}) + (2.1111x10^{-3})(3.08844\,x10^{-3})]S}$$

Asumiendo que el valor de la inductancia de armadura L_a es cero, tenemos:

$$\frac{\theta_L}{V_a} = \frac{16.5x10^{-6}}{2.238x10^{-3}S^2 + 9.396\,x10^{-6}S} \tag{2.88}$$

Y obteniendo su forma canónica:

$$\frac{\theta_L}{V_a} = \frac{7.37x10^{-3}}{S^2 + 4.20x10^{-3}\,S} \tag{2.89}$$

2.3 Descripción y características de las tarjetas, sensores y actuadores a utilizar en el desarrollo del controlador y la plataforma para giro-estabilización de posición.

Las tarjetas, sensores y actuadores a utilizar durante el desarrollo del proyecto se describen a continuación:

ARDUINO MEGA 2560:

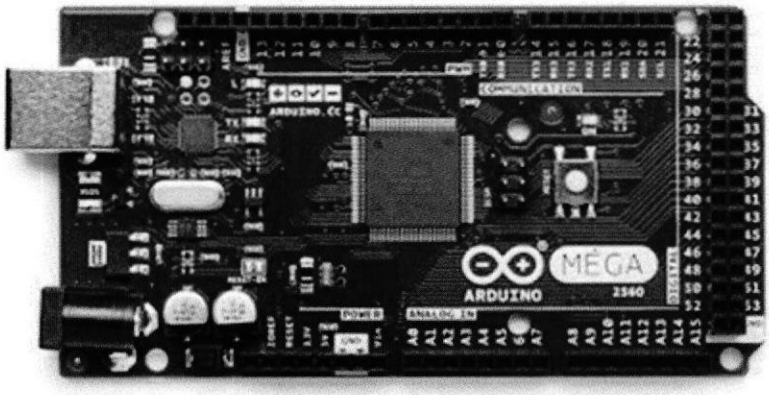


Figura 2.29: Arduino Mega 2560 [12]

Arduino es una plataforma de hardware de código abierto, basada en una sencilla placa de circuito impreso que contiene un microcontrolador de la marca "ATMEL" que cuenta con entradas y salidas, analógicas y digitales, en un entorno de desarrollo que está basado en el lenguaje de programación *processing* (Véase Figura 2.29).

Este dispositivo se lo utilizará para hacer la simulación del movimiento ondulatorio, se utilizará dos de sus entradas analógicas para poder controlar la velocidad y desplazamiento angular de dos servomotores que simularán el movimiento irregular de la base de la plataforma inercial.

Existirá otro Arduino Mega utilizado para el control de la plataforma inercial y que la carga colocada en el brazo interno no se vea afectado por los cambios generados por el movimiento ondulatorio.

MPU 6050

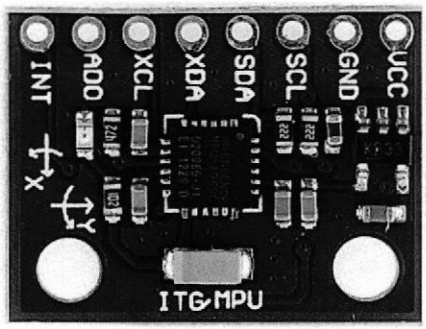


Figura 2.30: MPU 6050 [13]

Se trata de un dispositivo capaz de medir la fuerza (aceleración) y la velocidad. Genéricamente consta de un Acelerómetro y un Giroscopio. Por lo tanto: una IMU no mide ángulos. Por lo menos no directamente, requiere algunos cálculos.

El MPU-6050 (Véase Figura 2.30) es una IMU de 6DOF (6 grados de libertad). Esto significa que lleva un acelerómetro y un giroscopio, ambos de 3 ejes (3+3 = 6DOF).

La aceleración puede expresarse en 3 ejes: X, Y Z, las tres dimensiones del espacio. Por ejemplo, si mueves la IMU hacia arriba, el eje Z marcará un cierto valor. Si es hacia delante, marcará el eje X, la IMU también detecta la aceleración de la gravedad terrestre. Gracias a la gravedad terrestre puedes usar las lecturas del acelerómetro para saber cuál es el ángulo de inclinación respecto al eje X o eje Y, supongamos que la IMU esté perfectamente alineada con el suelo. Entonces, como puedes ver en la Figura 2.31, el eje Z marcará 9.8, y los otros dos ejes marcarán 0. Ahora supongamos que giramos la IMU 90 grados. Ahora es el eje X el que está perpendicular al suelo, por lo tanto, mostrará la aceleración de la gravedad.

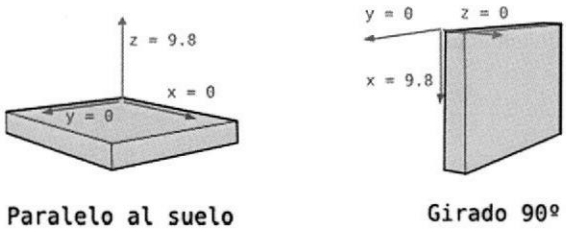


Figura 2.31: Medición de la aceleración tomando como referencia la gravedad [14]

Si sabemos que la gravedad es 9.8 m/s², y sabemos que mesura dan los tres ejes del acelerómetro, por trigonometría es posible calcular el ángulo de inclinación de la IMU. Una buena fórmula para calcular el ángulo es:

$$angulo_x = \tan^{-1}(aceleración_y / aceleración_z) \quad (2.90)$$

Dado que el ángulo se calcula a partir de la gravedad, no es posible calcular el ángulo Z con esta fórmula ni con ninguna otra. Para hacerlo se necesita otro componente: el magnetómetro, que es un tipo de brújula digital. El MPU-6050 no lleva, y por tanto nunca podrá calcular con precisión el ángulo Z. Sin embargo, para la gran mayoría de aplicaciones sólo se necesitan los ejes X e Y, lo cual hace que la plataforma de este trabajo sea controlada solo en 2 ejes de forma automática y la otra de forma manual.

El giroscopio mide la velocidad angular, si sabemos el ángulo inicial de la IMU, podemos adherirle el valor que marca el giroscopio para saber el nuevo ángulo a cada momento.

$$angulo_y = angulo_{anterior_y} + giroscopio_y * delta\_t \quad (2.91)$$

Dónde $delta\_t$ es el tiempo que transcurre cada vez que se calcula esta fórmula, $angulo_{anterior_y}$ es el ángulo calculado la última vez que se llamó esta fórmula y $giroscopio_y$ es la lectura del ángulo Y del giroscopio.

Y lo mismo pasa con los ejes X, Z. Sólo que se suele ignorar el eje Z, puesto que al no poder calcular un ángulo Z con el Acelerómetro, no se puede aplicar un Filtro Complementario para el eje Z.

El filtro de complemento o en inglés "*Complementary Filter*" es uno de los más usados por su fácil implementación, combina el ángulo calculado por el giroscopio y el ángulo calculado por el acelerómetro.

La necesidad de combinar ambas lecturas es que, si solo trabajamos con el acelerómetro, este es susceptible a las aceleraciones producto del movimiento del MPU o a fuerzas externas, pero en tiempos largos el ángulo no acumula errores.

La ecuación para calcular el ángulo usando el filtro de complemento es:

$$ángulo = 0.98(ángulo + \omega_{giroscopio}\, dt) + 0.02(ang_{acelerómetro}) \quad (2.92)$$

De esta forma el ángulo del acelerómetro está pasando por un filtro pasa bajos, amortiguando las variaciones bruscas de aceleración; y el ángulo calculado por el giroscopio tiene un filtro pasa altos teniendo gran influencia cuando hay rotaciones rápidas. Podemos probar también con otros valores diferentes a 0.98 y 0.02 pero siempre deben de sumar 1.

SERVOMOTOR HS311

Figura 2.32: Servomotor HS311 [15]

Estos elementos (Véase Figura 2.32) serán los encargados de estabilizar los brazos internos, medio y externo; trabajarán directamente con el acelerómetro ya que enviará una señal al Arduino y éste les generará una señal procesada indicándoles que ángulo deben tener con respecto a su eje de rotación de tal manera que la carga ubicada en el brazo interno no sufra un cambio de posición considerable, las características son las siguientes:

Engranajes en resina resistente

Tipo de motor: 3 Pole

Tipo de rodamiento: Ninguno

Velocidad (4.8V/6.0V): 0,19/0,15sec@60grados.

Torque kg.* cm. (4.8V/6.0V): 3.0/3.7

Tamaño en pulgadas: 1.57x0.78x1.43

Tamaño en milímetros: 39.88x19.81x36.32

Peso: 42,81g

SERVOMOTOR TR213

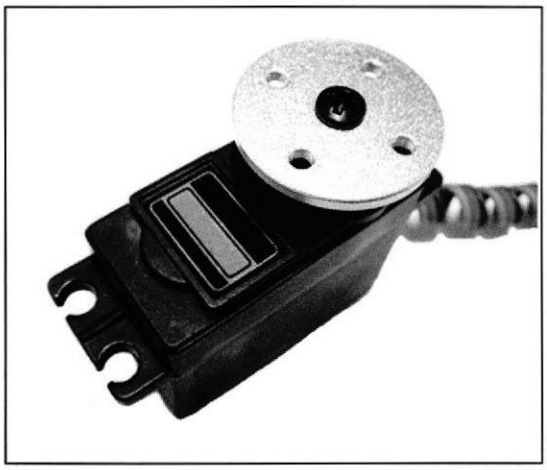


Figura 2.33: Servomotor TR213

Servomotor de alto torque (Véase Figura 2.33) para aplicaciones donde se debe mover objetos con mayores inercias, estos servomotores estarán ubicados en la plataforma de movimiento ondulatorio, posee las siguientes características:

TR213 engranaje de metal completo 13 kg de precisión de par de 0,5 grados de los servos de robot dedicado

TR213 puede controlar la rotación de 0,5 °

Peso: 63g

Tamaño: cerca de 40mmX20mmX36.5mm

Velocidad: 0,15 segundos / 60 grados (4,8 V); 0,12 seg / 60 grados (6,0 V)

Par de apriete: 13kg · cm

Precisión de control: 0.5 °

Alcance: 180 °

Temperatura de funcionamiento: 0 ~ +55 ° C

Tensión de trabajo: 4.8V-7.2V

Motor sin núcleo, estructura metálica de engranajes, cojinete de bolas doble, longitud de cable de 30 cm.

JOYSTICK ARDUINO

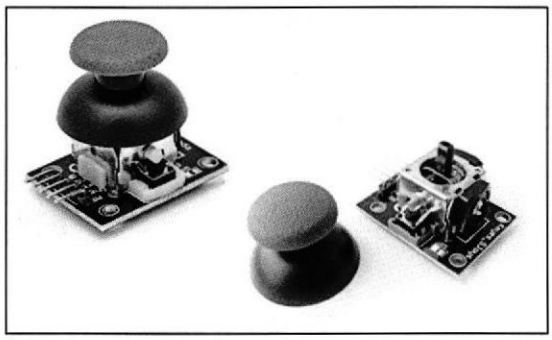


Figura 2.34: Joystick [16]

Este componente (Véase Figura 2.34) será empleado para un modo manual en que un usuario intente apuntar a un objetivo pese a que todo el sistema se está moviendo de forma ondulatoria, y de esta manera poder observar las ventajas de tener un sistema automático que pueda estabilizar una carga.

LCD

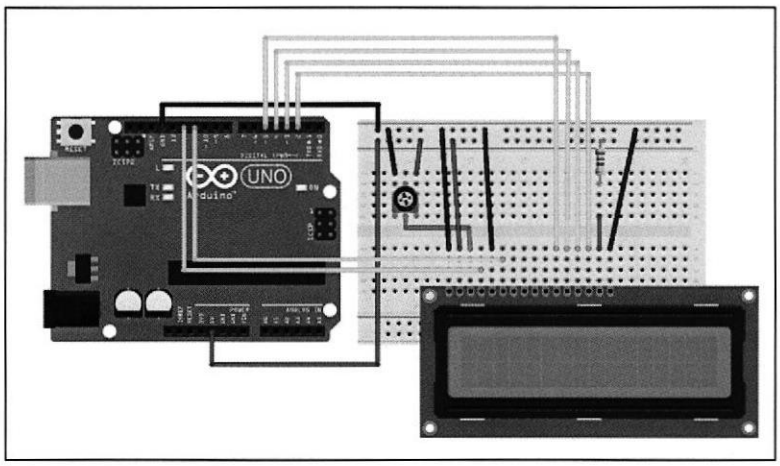


Figura 2.35: LCD [17]

Este componente electrónico (Véase Figura 2.35) será un indicador para darle al usuario información acerca de cuán rápido se está generando el movimiento ondulatorio, y cuál es su desplazamiento máximo que genera el mismo.

CAPÍTULO 3

3. DISEÑO DEL CONTROLADOR DIFUSO E INTERFAZ VÍA MATLAB-SIMULINK CON ARDUINO Y LA PLATAFORMA INERCIAL IMPLEMENTADA

En este capítulo se muestra el diseño y elaboración tanto de la plataforma inercial como el mecanismo de simulación del movimiento ondulatorio, tanto a nivel de software como de hardware.

3.1 Implementación de una plataforma de tiro con giro-estabilización.

La plataforma de tiro está conformada por tres brazos rotacionales con sus respectivos servomotores controlados por un Arduino Mega, como se muestra en la Figura 3.1, para la demostración del giro-estabilizado se colocó una carga en el brazo interno conformado por una placa rectangular y apuntador láser.

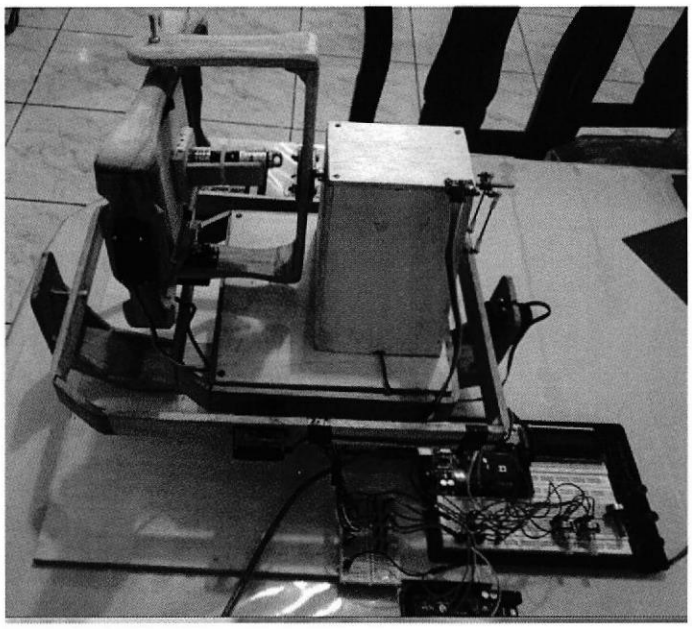


Figura 3.1: Plataforma de tiro con giro-estabilización

Debido a que el servomotor G1 genera un peso considerable, tanto para el servomotor G2 como para el G3, se ha colocado una masa de contrapeso en el extremo opuesto del brazo medio para generar una mayor estabilidad en el sistema como se aprecia en la Figura 3.2.

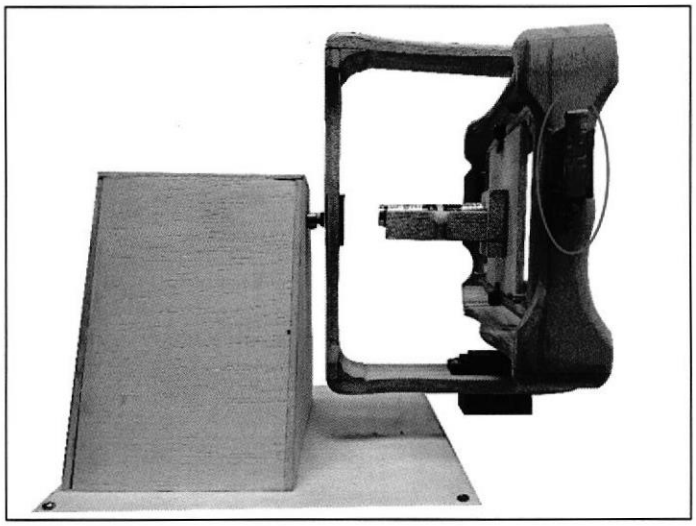


Figura 3.2: Vista Lateral de la plataforma inercial

Para la unión de los brazos rotacionales se empleó el uso de tornillos para el lado opuesto a los servomotores correspondientes, como se aprecia en la vista frontal de la Figura 3.3.

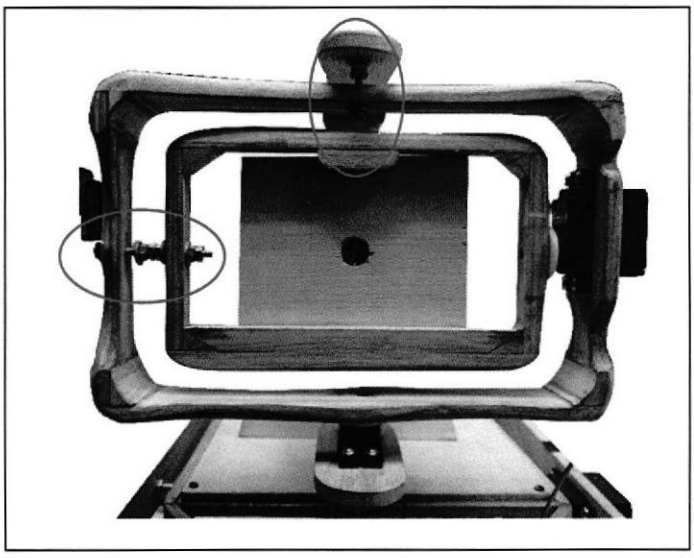


Figura 3.3: Vista Frontal de la plataforma inercial

Debido a que la plataforma de tiro posee 3 servomotores, ésta puede rotar en todos los ejes indicados en la Figura 3.4, que son: cabeceo, alabeo y guiñada. El microcontrolador les indica a los servomotores cuánto desplazamiento angular realizar basándose en las señales recibidas por el acelerómetro ubicado en la parte anterior de la plataforma.

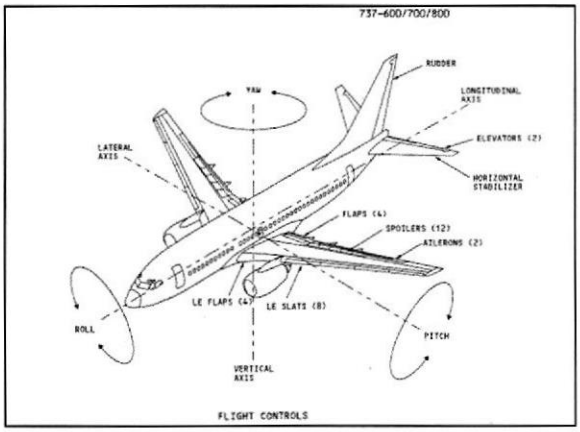


Figura 3.4: Ejes de rotación [18]

Cuando el microcontrolador detecta un cambio positivo en el eje de cabeceo (Véase Figura 3.5), inmediatamente manda una señal al servomotor relacionado a dicho eje para que realice un desplazamiento angular en sentido opuesto, pero de forma proporcional, generando un resultado ilustrado en la Figura 3.6

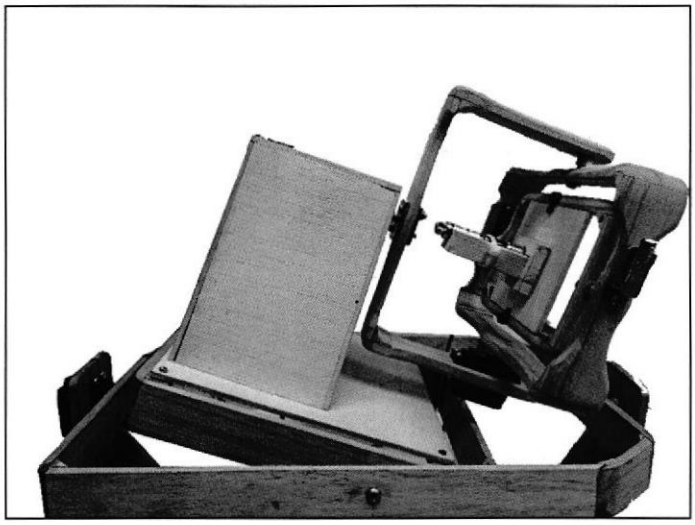


Figura 3.5: Desplazamiento de la base en el eje de cabeceo

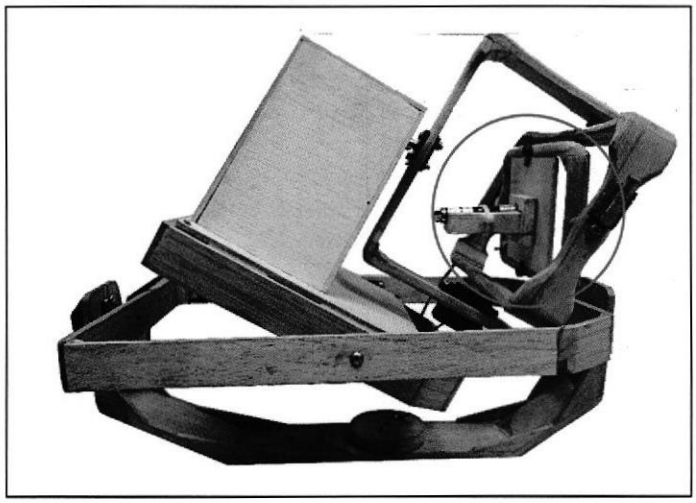


Figura 3.6: Respuesta frente a cambio en el eje de cabeceo

De manera similar el microcontrolador envía una señal como respuesta a cambios en el eje de guiñada (Véase Figura 3.7). La plataforma también realiza la giro-estabilización si detecta cambios en la posición angular en más de un eje rotacional como se muestra en la Figura 3.8.

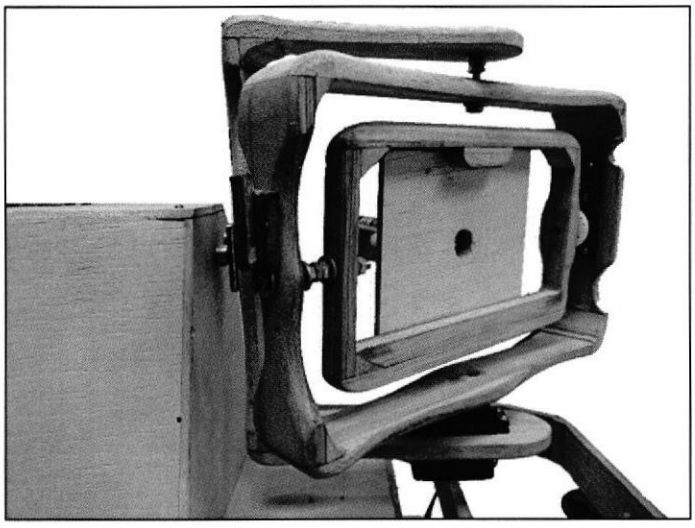


Figura 3.7: Giro de guiñada

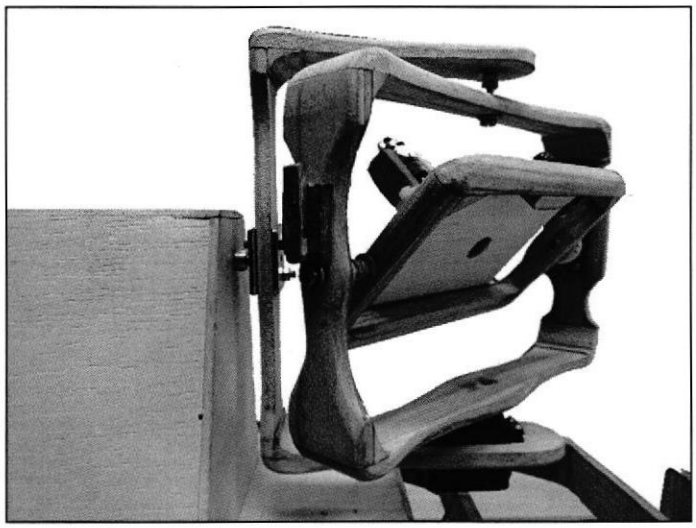


Figura 3.8: Giro de guiñada y cabeceo

Además de controlar los giros en base al acelerómetro, se adicionó un joystick para hacer giros manuales en un eje en particular y tener un mayor control del apuntador láser, las conexiones se muestran en la Figura 3.9

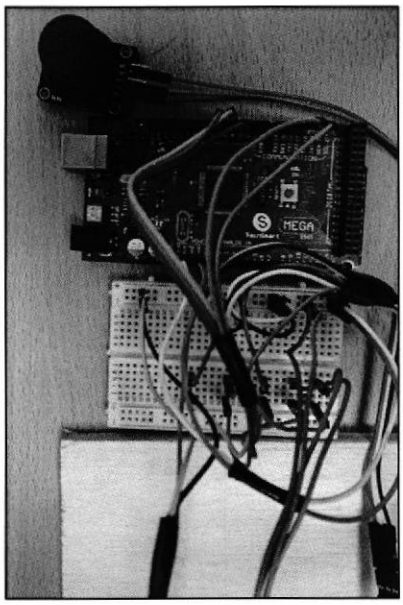


Figura 3.9: Conexiones del microcontralador con los servomotores y joystick

3.2 Implementación de un prototipo para simulación de movimiento ondulatorio de embarcaciones con 2 grados de libertad.

Según lo indicado en la sección anterior, la plataforma de tiro necesita de un prototipo para simular un movimiento ondulatorio en dos ejes (cabeceo y alabeo), el prototipo ilustrado en la Figura 3.10 presenta dos servomotores de mayor fuerza para poder soportar el peso de toda la plataforma y hacerlo girar en dos ejes diferentes de manera simultánea.

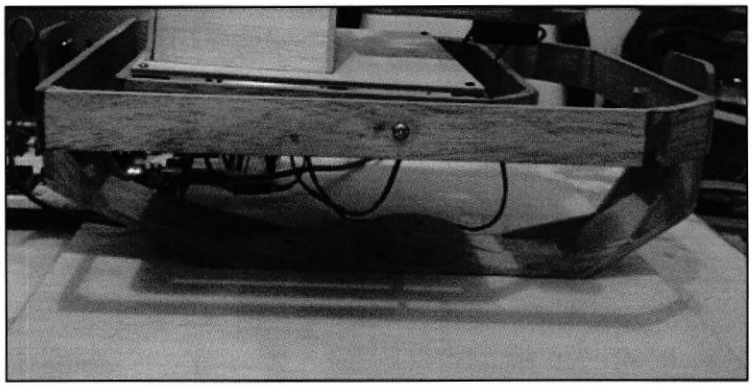


Figura 3.10: Prototipo de simulación de movimiento ondulatorio

Para la manipulación de este prototipo se hace uso de otro Arduino Mega y un par de potenciómetros, los cuales determinan la velocidad del desplazamiento de ambos ejes y el desplazamiento máximo de giro, dichos valores son presentados en una LCD para tener un mejor control del prototipo por parte del usuario (Véase Figura 3.11).

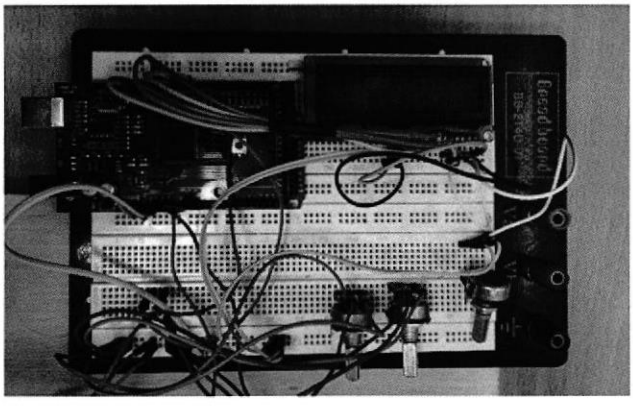


Figura 3.11: Conexiones para el control del prototipo de simulacion de movimiento ondulatorio

3.3 Desarrollo de un software para determinar los modelos matemáticos con ingreso de datos de la plataforma implementada.

La siguiente guía fue desarrollada con el software MATLAB R2015, la cual nos permite ingresar los parámetros eléctricos, mecánicos y estructurales de la plataforma con el fin que el usuario pueda obtener diferentes modelos matemáticos en función de las mismas.

3.3.1 Descripción del código fuente

El archivo que se ejecuta al inicio se denominó *"primera.m"* (Ver anexo C), el cual posee cinco funciones, de las cuales dos de ellas son funciones por defecto que sirven para crear y mostrar su ventana correspondiente, las otras tres funciones son ejecutadas cuando se selecciona uno de los botones de la ventana inicial, se detallan a continuación:

- La función *"pushbutton1_Callback"* ejecuta el próximo archivo (*"introduccion.m"*) y cierra la ventana actual.
- La función *"pushbutton2_Callback"* termina el programa cerrando todas las ventanas.
- La función *"pushbutton3_Callback"* inicializa todas las variables a utilizarse con su valor por defecto.

El siguiente archivo a ejecutarse, tal como se indica anteriormente, es *"introduccion.m"* (Ver anexo D). Este archivo contiene algunas funciones de las cuales algunas se crean por defecto sin utilidad alguna, a continuación, se detallan las funciones relevantes:

- La función *"pushbutton1_Callback"* ejecuta el próximo archivo (*"parametros_motor.m"*) y cierra la ventana actual.
- La función *"pushbutton2_Callback"* termina el programa cerrando todas las ventanas.
- La función *"pushbutton3_Callback"* ejecuta la función anterior (*"primera.m"*) y cierra la ventana actual.

El próximo archivo a ejecutarse es *"parametros_motor.m"* (Ver anexo E). Las funciones relevantes son las siguientes:

- La función *"pushbutton1_Callback"* termina el programa cerrando todas las ventanas.
- La función *"pushbutton2_Callback"* ejecuta el próximo archivo (*"tercero.m"*) y cierra la ventana actual.
- La función *"pushbutton3_Callback"* ejecuta el archivo inicial (*"primera.m"*) y cierra la ventana actual.
- La función *"pushbutton4_Callback"* convierte en valores numéricos todos los parámetros ingresados en los cuadros de texto para cálculos posteriores.
- La función *"pushbutton5_Callback"* establece todos los valores por defecto para los cálculos posteriores.
- La función *"pushbutton6_Callback"* ejecuta el archivo anterior (*"introduccion.m"*) y cierra la ventana actual.

El próximo archivo a ejecutarse es *"tercero.m"* (Ver anexo F). Las funciones relevantes son las siguientes:

- La función *"pushbutton1_Callback"* termina el programa cerrando todas las ventanas.
- La función *"pushbutton2_Callback"* ejecuta el archivo inicial (*"primera.m"*) y cierra la ventana actual.
- La función *"pushbutton3_Callback"* ejecuta el archivo anterior (*"parametros_motor.m"*) y cierra la ventana actual.
- La función *"brazos_Callback"* se ejecuta al intentar seleccionar un nuevo brazo para establecer las dimensiones físicas, mostrando la imagen del brazo seleccionado.

- La función *"listo _Callback"* asigna los valores ingresados por el usuario a las variables correspondientes al brazo seleccionado anteriormente.
- La función *"pushbutton5_Callback"* asigna los valores por defecto de todas las dimensiones de los brazos de la estructura.
- La función *"pushbutton6_Callback"* ejecuta el próximo archivo (*"resultado.m"*) y cierra la ventana actual.

El próximo y último archivo a ejecutarse es *"resultado.m"* (Ver anexo G). Las funciones relevantes son las siguientes:

- La función *"pushbutton1_Callback"* termina el programa cerrando todas las ventanas.
- La función *"pushbutton2_Callback"* ejecuta el archivo inicial (*"primera.m"*) y cierra la ventana actual.
- La función *"pushbutton3_Callback"* ejecuta el archivo anterior (*"tercero.m"*) y cierra la ventana actual.
- La función *"transfe_Callback"* se actualizan las imágenes que se presenta al usuario indicándole la función de transferencia que ha seleccionado.
- La función *"pushbutton4_Callback"* carga los datos, que fueron ingresado por el botón predeterminado de las pantallas anteriores o por los datos que el usuario ingresó por teclado, para ser calculados y obtener las funciones de transferencia de forma canónica.
- La función *"pushbutton5_Callback"* ejecuta otra función denominada *"funcionopen_system"* que abre Simullink; esto ocurre cuando el usuario selecciona el botón "*CONTROLADOR*".

3.4 Desarrollo de un software en Arduino Mega para monitoreo y control de la plataforma de movimiento ondulatorio.

Para la simulación del movimiento ondulatorio se trabajó con el microcontrolador Arduino Mega, como se muestra en la Figura 3.12, el cual envía pulsos a los servomotores, dependiendo de la posición de dos potenciómetros, para establecerlos a una velocidad y desplazamiento angular determinado, los cuales se indican mediante una LCD.

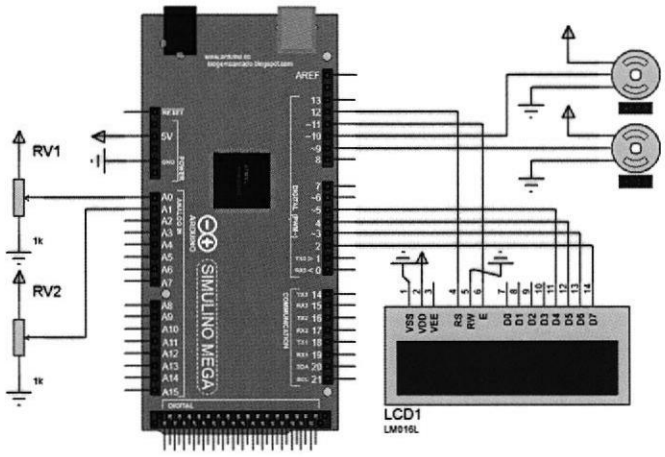


Figura 3.12: Esquemático del controlador de la plataforma de movimiento ondulatorio

La programación se la trabajó en 3 secciones: la primera es la declaración de variables y librerías a utilizar, la segunda corresponde a la inicialización de los servomotores y la LCD, la tercera sección es un lazo infinito que se encarga de actualizar los valores a mostrarse en la LCD y genera un llamado a una función que manipula la velocidad y desplazamiento máximo de los servomotores.

3.4.1 Sección 1: Declaración de librerías, constantes y variables

Las librerías a utilizarse son "Servo.h", que permite el control de la posición de los servos enviando un valor entre 0 y 180; la librería "LiquidCrystal.h" es la encargada de mostrar los mensajes en la LCD.

Debido a que se trabajará con dos servomotores, es necesario crear dos objetos de tipo "Servo" los cuales fueron denominados como "myservo" y "myservo2"; para la LCD se crea un objeto de tipo "LiquidCrystal" enviando como parámetros los pines a los cuales se va a conectar la LCD con el Arduino Mega.

Para establecer el límite máximo del desplazamiento angular se usan las constantes "angulo_ref" que será el valor de la posición inicial de los servomotores y "max_desp" que será le desplazamiento máximo que tendrá el servomotor en caso de que el potenciómetro esté en su máxima posición, es decir que si el potenciómetro está en su mínimo valor el servomotor estará en su posición inicial y no se moverá (90°±0), en cambio si el potenciómetro está en su máximo valor el servomotor se moverá entre su posición inicial más el desplazamiento máximo y entre su posición inicial menos el desplazamiento máximo (90°±45°).

La velocidad angular de los servomotores dependerá del valor de retardo que genere el otro potenciómetro, dicho retardo estará limitado por las constantes "max_ret" y "min_ret"; a mayor retardo la velocidad angular será más lenta y a menor retardo la velocidad será mayor.

La variable que controla la velocidad de los servomotores es "vel" la cual está limitada por las constantes antes mencionadas; mientras que para el desplazamiento angular dependerá de la variable "desp" la cual está limitada entre cero y la constante "max_desp".

Dado que la LCD no posee todos los valores ASCII existentes, como es el caso del signo (±), es necesario almacenar un arreglo de bytes en la memoria de la LCD que indiquen que puntos deben encenderse en un espacio de la LCD, este vector de bytes fue denominado "signo [8]" (Véase Figura 3.13).

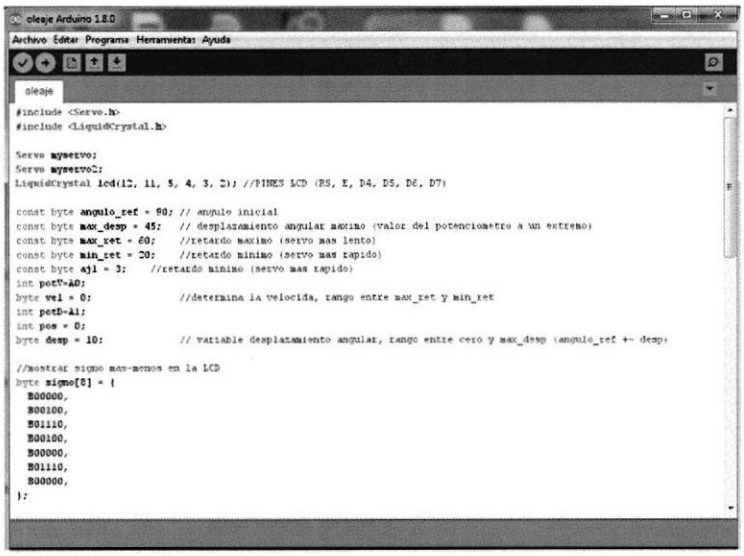


Figura 3.13: Programación Arduino, Sección 1

3.4.2 Sección 2: Configuración inicial de los objetos de tipo "servo" y "LiquidCrystal"

En esta sección se define los pines del Arduino a los que van a estar conectados los servomotores (Véase Figura 3.14), también se almacena el vector de bytes "signo" para poder mostrar (±) en la LCD mediante la función lcd.createChar(0, signo) y por último se muestra los mensajes base en la LCD, es decir los mensajes que no van a modificarse a lo largo de la ejecución.

Figura 3.14: Programación Arduino, Sección 2

3.4.3 Sección 3: Lazo Infinito y función "mover_Servo"

En esta sección se realiza la adquisición de los valores analógicos (potV y potD)que modificarán las variables "vel" y "desp", dado que la función "AnalogRead" devuelve valores entre 0 y 1023 se debe realizar una conversión para que sus valores mínimos y máximos sean las constantes predefinidas anteriormente, esta conversión se lo realiza con la función "map" que recibe 5 parámetros: el valor leído, los valores mínimo y máximo que puede tener el primer parámetro y los valores mínimo y máximo que se desea obtener (Véase Figura 3.15).

Una vez obtenido los valores de "vel" y "desp" se procede a mostrarlos en la LCD mediante la función "lcd.print".

Para realizar el giro de los servomotores se llama a la función "mover_Servo", la cual consiste en 4 ciclos que se basan en lo mismo. El primer ciclo realiza un desplazamiento de los servomotores desde su posición inicial hasta un incremento de cierta cantidad de grados determinado por la variable "desp", cada iteración realiza un desplazamiento de 1° a una velocidad determinada por la variable "vel". El segundo ciclo se encarga de hacer el desplazamiento inverso hasta llevar los servomotores a su posición inicial. El tercer ciclo realiza un desplazamiento desde su posición inicial hasta un decremento de cierta cantidad de grados determinado por la variable "desp". Y, por último, el cuarto ciclo se encarga de llevar los servomotores a su posición inicial.

Figura 3.15: Programación Arduino, Sección 3

3.5 Desarrollo de un software en Matlab usando Simulink y librerías de Arduino como interfaz para el control y adquisición de datos de la plataforma inercial.

En esta sección se realizará el análisis de cada una de las etapas para obtener y acondicionar las señales receptadas de la plataforma inercial y generar su respectiva respuesta.

3.5.1 Análisis de las etapas para adquisición y acondicionamiento de señales

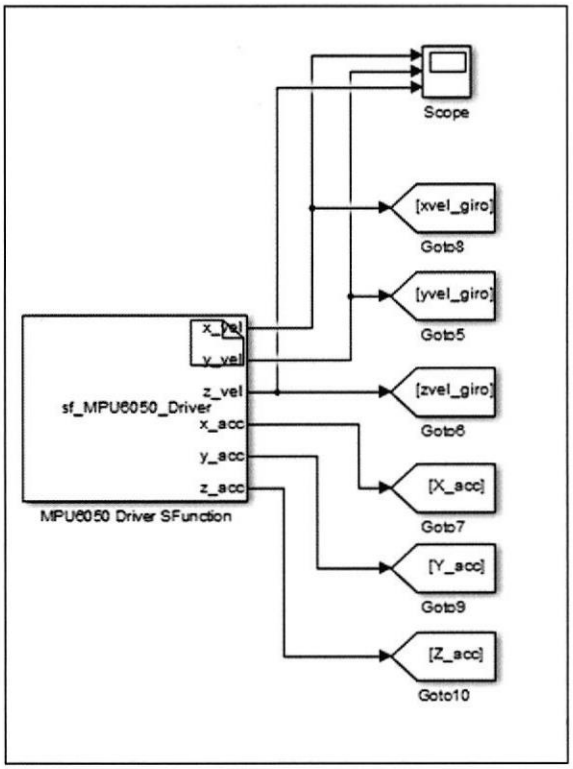


Figura 3.16: Bloque de función de lectura del MPU6050

MPU6050 Driver SFunction este bloque de función permite recibir las señales del giroscopio y acelerómetro del módulo MPU6050 de 3 ejes de movimiento desde una entrada serial del Arduino MEGA con una tasa de transferencia de 9600 baudios listo para ser tratado como 6 señales discretas las cuales servirán para calcular la posición angular del dispositivo durante las perturbaciones (Véase Figura 3.16).

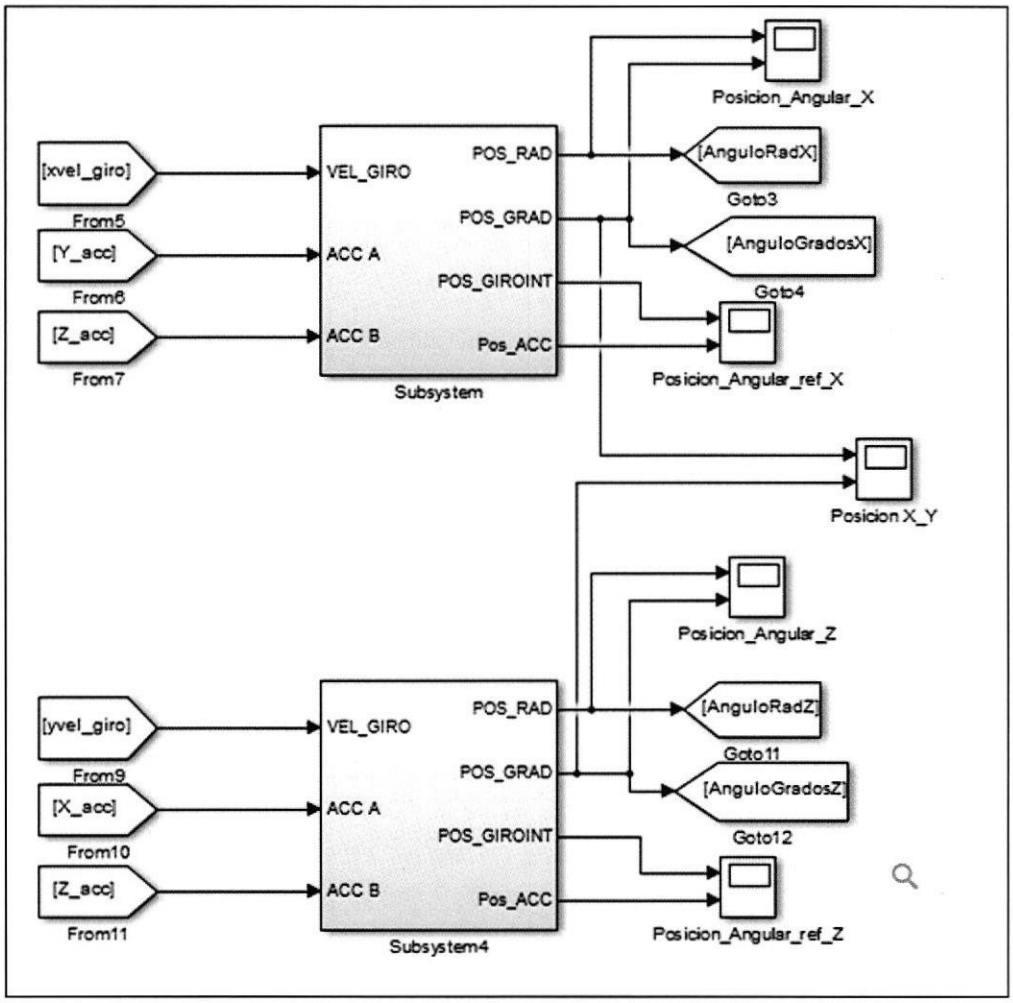


Figura 3.17: Bloque de acondicionamiento del giroscopio y acelerometro

En esta etapa los subsistemas reciben los datos en dos ejes diferentes para calcular su respectiva posición y poder visualizarlo en grados y en radianes dependiendo de qué unidades se prefiera trabajar (Véase Figura 3.17).

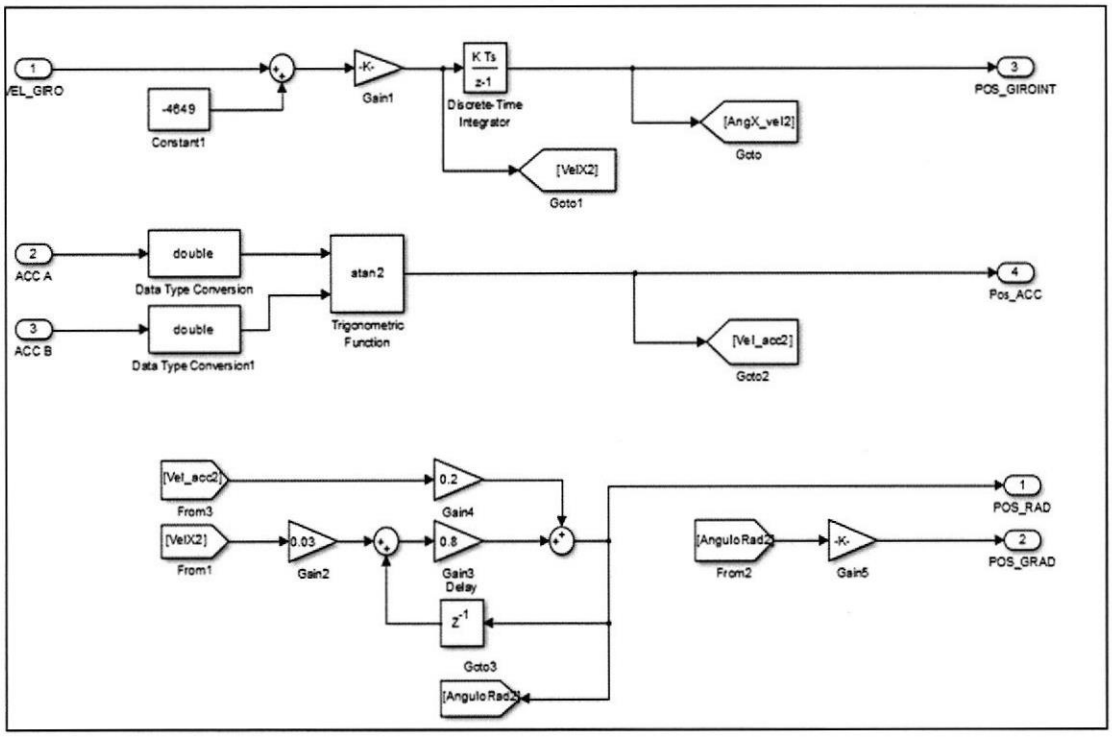


Figura 3.18: Bloque para filtrado de cálculos de posición

En la Figura 3.18 se observa el sistema para procesar los datos de velocidad y aceleración recibidos con la finalidad de obtener la posición, sin embargo el sensor presenta una leve inestabilidad lo que provoca que al integrar la señal proveniente del giroscopio se genere una acumulación en el error y la lectura indique un decremento o incremento desde el valor que se debería leer, pero pese a perturbaciones no sufre cambios drásticos, por parte de la señal del acelerómetro una perturbación como un golpe genera picos en la lectura, pero no se desalinea del valor real medido. Sin notar sus falencias y usando sus fortalezas se aplica un filtro como una participación ponderada de ambas contribuciones de posición se puede obtener una señal con poca incidencia por perturbación y estable durante el tiempo.

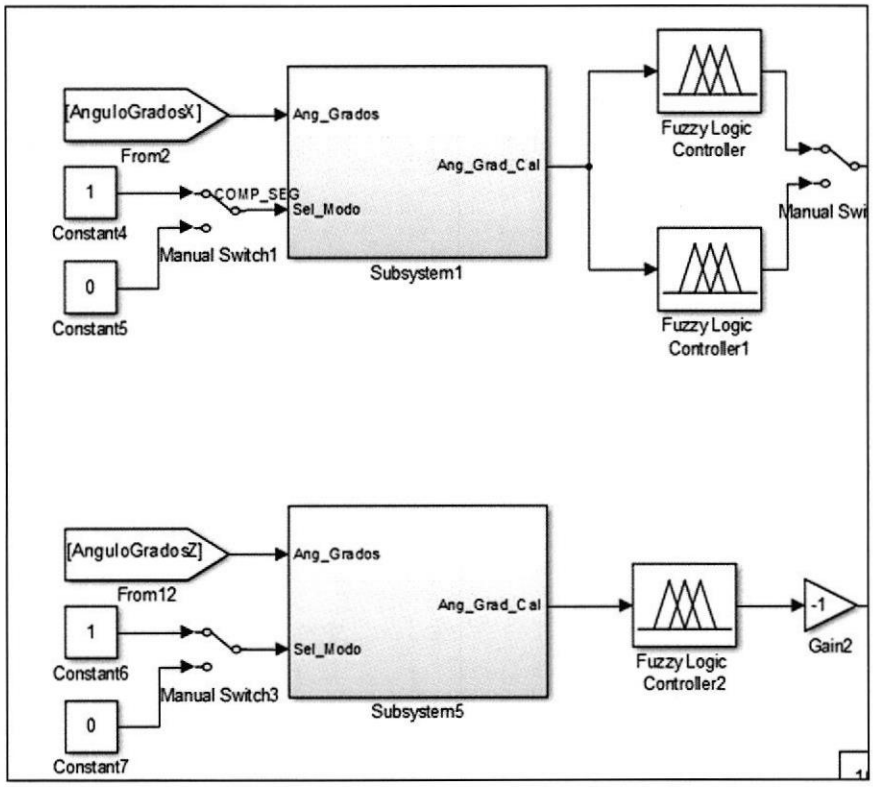


Figura 3.19: Bloque de Calibracion de posicion inical del motor

En esta etapa condicionamos la señal para el controlador de tal forma que el valor central sea 90 grados (Véase Figura 3.19).

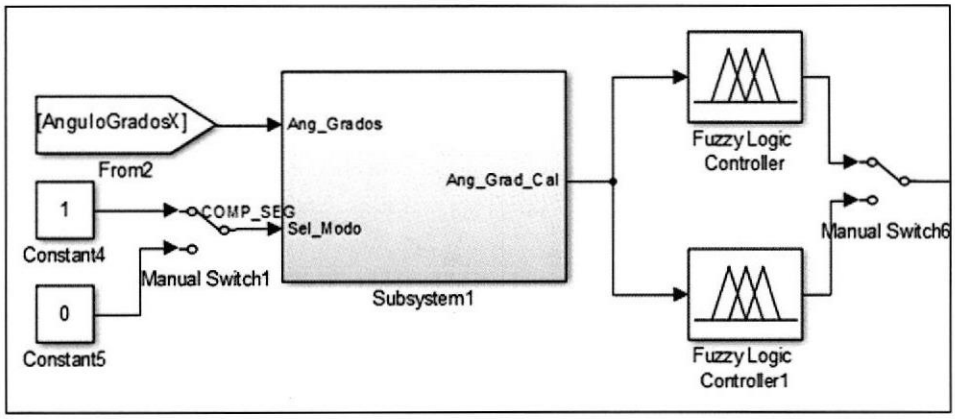


Figura 3.20: Pruebas de distintos controladores

En Figura 3.20 podemos ver la etapa de pruebas con diferentes controladores difusos de tal forma que con solo cambiar el estado del

selector podemos cambiar la forma de controlar el sistema sin necesidad de detener el proceso de control.

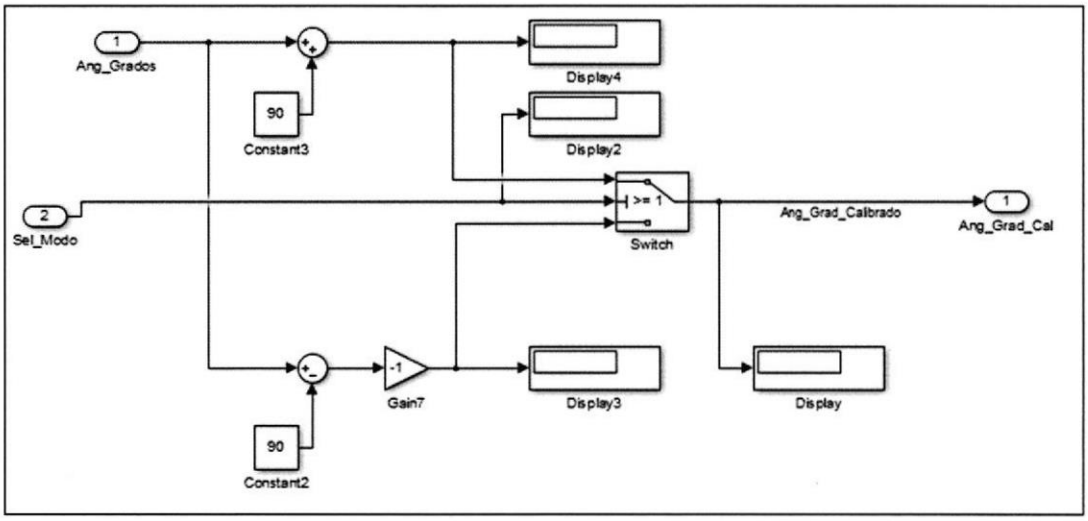


Figura 3.21: Bloque de selección de modo de seguimiento-corrector

En la Figura 3.21 se observa la selección del modo que permite cambiar el funcionamiento del sistema de control como corrector de perturbación o seguidor, en el último caso es para observar que el servomotor de corrección se mueva al mismo ángulo que está siendo perturbado.

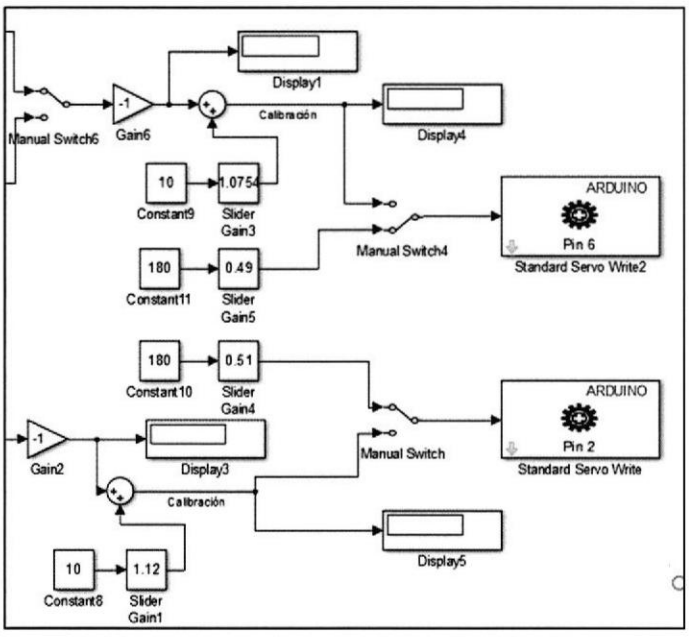


Figura 3.22: Calibracion y manejo de Servomotores

En esta fase los bloques de Arduino para servomotor reciben la señal del controlador sin embargo se deben ajustar debido a descalibraciones mecánicas e incluso variaciones con la geometría de donde se ubica el sensor (Véase Figura 3.22).

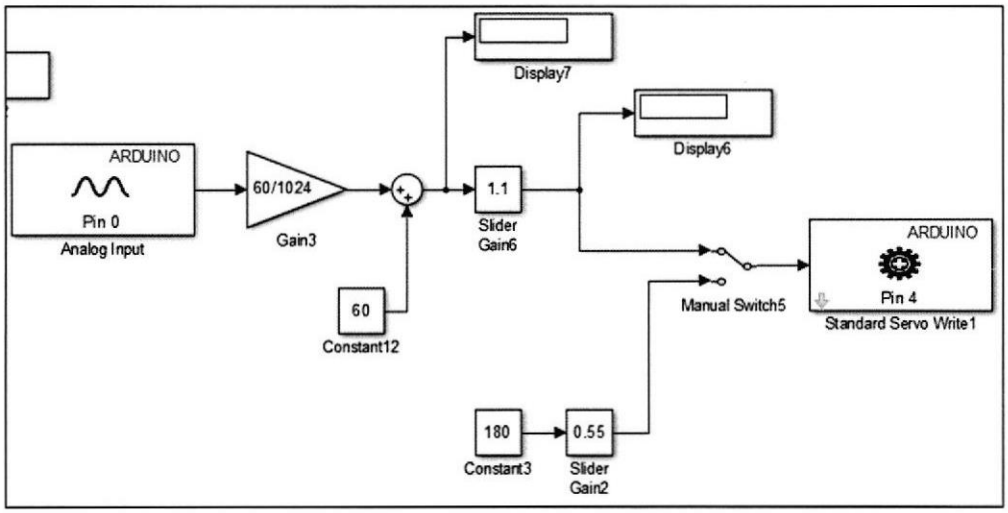


Figura 3.23: Control de modo analógico con Joystick

En la Figura 3.23 se observa el control manual a través de una entrada analógica proveniente de un Joystick que permite al usuario cambiar de izquierda a derecha la posición de tiro.

CAPÍTULO 4

4. ANÁLISIS DE LAS SIMULACIONES Y RESULTADOS

En este capítulo se muestran todos los resultados obtenidos matemáticamente tanto del sistema implementado con lógica difusa, frente a otro tipo de controlador con dimensiones distintas, y de esta manera hacer una comparativa entre qué tipo de controlador es más óptimo.

4.1 Análisis de resultados de modelos matemáticos obtenidos para diferentes dimensiones de la plataforma.

Una vez realizado el análisis teórico y desarrollado el software se puede ver los resultados del modelo matemático tanto teórico como el calculado por el software de la plataforma implementada y de modelos de plataforma que se desearía realizar por otro usuario, con el fin de mejorar o ampliar las opciones de la plataforma desarrollada en este trabajo.

4.1.1 Modelo matemático con los valores de la plataforma implementada

Debido a que el sistema posee 3 brazos se debería obtener 3 modelos matemáticos simplificados, pero 2 de ellos se encuentran relacionados entre sí, se debe analizar los cambios de posición angular de los mismos para determinar cambios en su inercia y así recalcular el modelo matemático más cercano a la realidad, por ende, para el caso del brazo1 solo existirá un modelo (Véase Figura 4.1), para el del brazo2 tendremos 2 modelos (Véase Figura 4.2 y Figura 4.3) y para el brazo3 tendremos 4 modelos (Véase Figura 4.4, Figura 4.5, Figura 4.6 y Figura 4.7), lo que nos indica 7 modelos en total:

Modelo del brazo1 (Teórico y Software):

$$\frac{\theta_L}{V_a} = \frac{0.14}{S^2 + 0.08\,S}$$

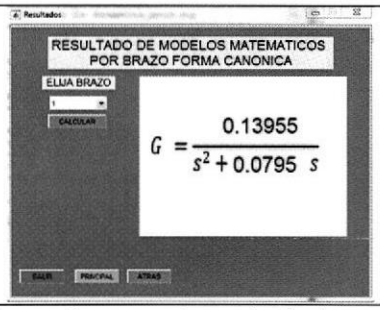


Figura 4.1: Comparación del modelo calculado brazo1 manual y calculado por Software

Modelo del brazo2 posición de la carga 0° (Teórico y Software):

$$\frac{\theta_L}{V_a} = \frac{3.04x10^{-3}}{S^2 + 1.73x10^{-3}\,S}$$

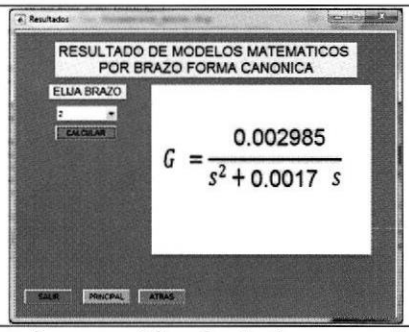


Figura 4.2: Comparación del modelo del brazo2 carga 0° calculado manual y calculado por Software.

Modelo del brazo2 posición de la carga 90° (Teórico y Software):

$$\frac{\theta_L}{V_a} = \frac{3.00x10^{-3}}{S^2 + 1.71x10^{-3}\,S}$$

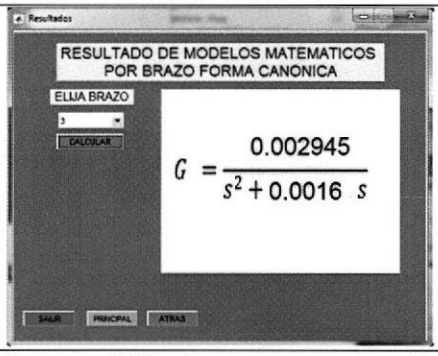


Figura 4.3: Comparación del modelo del brazo2 carga 90° cálculo manual y cálculo por Software.

Modelo del brazo3 posición del brazo2 a 0° y carga a 0° (Teórico y Software):

$$\frac{\theta_L}{V_a} = \frac{2.21x10^{-3}}{S^2 + 1.26x10^{-3}\,S}$$

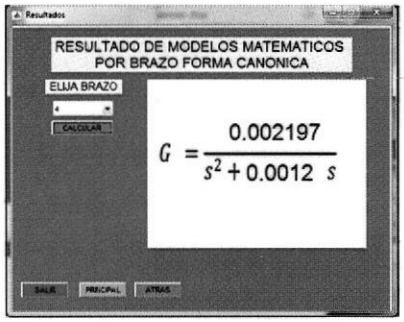


Figura 4.4: Comparación del modelo del brazo3 posición del brazo2 a 0° y carga a 0° cálculo manual y cálculo por Software.

Modelo del brazo3 posición del brazo2 a 0° y carga a 90° (Teórico y Software):

$$\frac{\theta_L}{V_a} = \frac{2.15x10^{-3}}{S^2 + 1.22x10^{-3}\,S}$$

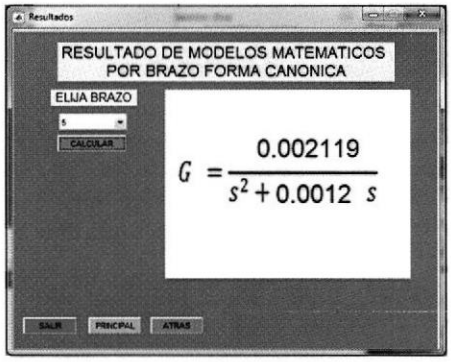


Figura 4.5: Comparación del modelo del brazo3 posición del brazo2 a 90° y carga a 0° cálculo manual y cálculo por Software.

Modelo del brazo3 posición del brazo2 a 90° y carga a 0° (Teórico y Software):

$$\frac{\theta_L}{V_a} = \frac{7.37x10^{-3}}{S^2 + 4.20x10^{-3}\,S}$$

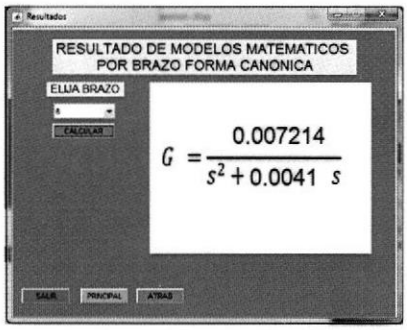


Figura 4.6: Comparación del modelo del brazo3 posición del brazo2 a 0° y carga a 0° cálculo manual y cálculo por Software.

Modelo del brazo3 posición del brazo2 a 90° y carga a 90° (Teórico y Software):

$$\frac{\theta_L}{V_a} = \frac{7.37x10^{-3}}{S^2 + 4.20x10^{-3}\,S}$$

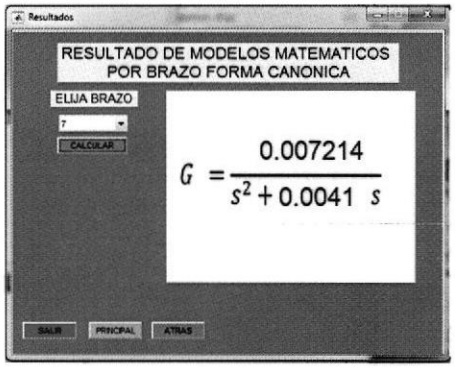


Figura 4.7: Comparación del modelo del brazo3 posición del brazo2 a 90° y carga a 0° cálculo manual y cálculo por Software.

4.1.2 Modelo matemático con los valores de la plataforma propuesta

Como en el caso anterior se obtendrá 7 modelos matemáticos para los casos ya explicados anteriormente, pero ahora solo serán resultados de una plataforma que se desee implementar en el futuro con la idea de mejorar la actual, en este ejemplo se cambia la densidad de la madera a un valor de 1180 kg/m^3, los parámetros de servomotor se mantienen constantes pero las dimensiones de la plataforma se reducen en un 20% (Véase Figura 4.8 - Figura 4.14):

Modelo del brazo1 propuesta (Teórico y Software):

$$\frac{\theta_L}{V_a} = \frac{0.03}{S^2 + 0.02\,S}$$

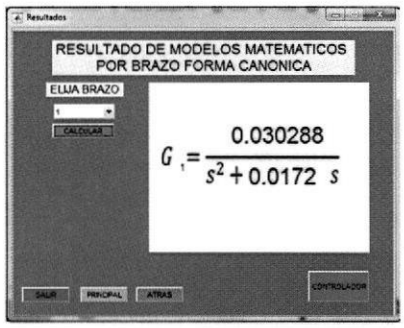


Figura 4.8: Comparación del modelo calculado brazo1 propuesta manual y calculado por Software

Modelo del brazo2 propuesta posición de la carga 0° (Teórico y Software):

$$\frac{\theta_L}{V_a} = \frac{0.40x10^{-3}}{S^2 + 0.23x10^{-3}\,S}$$

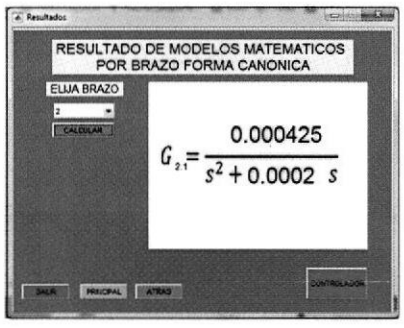


Figura 4.9: Comparación del modelo del brazo2 propuesta carga 0° calculado manual y calculado por Software.

Modelo del brazo2 propuesta posición de la carga 90° (Teórico y Software):

$$\frac{\theta_L}{V_a} = \frac{0.41x10^{-3}}{S^2 + 0.23x10^{-3}\,S}$$

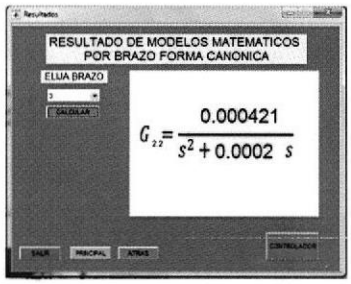


Figura 4.10: Comparación del modelo del brazo2 propuesta carga 90° cálculo manual y cálculo por Software.

Modelo del brazo3 propuesta posición del brazo2 a 0° y carga a 0° (Teórico y Software):

$$\frac{\theta_L}{V_a} = \frac{1.09x10^{-3}}{S^2 + 0.58x10^{-3}\,S}$$

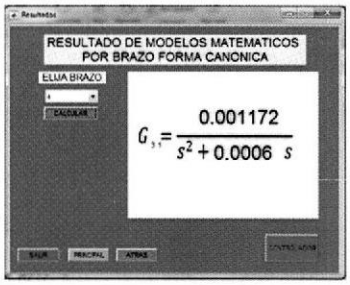


Figura 4.11: Comparación del modelo del brazo3 propuesta posición del brazo2 a 0° y carga a 0° cálculo manual y cálculo por Software.

Modelo del brazo3 propuesta posición del brazo2 a 0° y carga a 90° (Teórico y Software):

$$\frac{\theta_L}{V_a} = \frac{0.97x10^{-3}}{S^2 + 0.47x10^{-3}\,S}$$

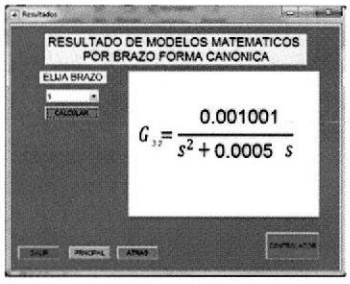


Figura 4.12: Comparación del modelo del brazo3 propuesta posición del brazo2 a 90° y carga a 0° cálculo manual y cálculo por Software.

Modelo del brazo3 propuesta, posición del brazo2 a 90° y carga a 0° (Teórico y Software):

$$\frac{\theta_L}{V_a} = \frac{3.34x10^{-3}}{S^2 + 1.88x10^{-3}\,S}$$

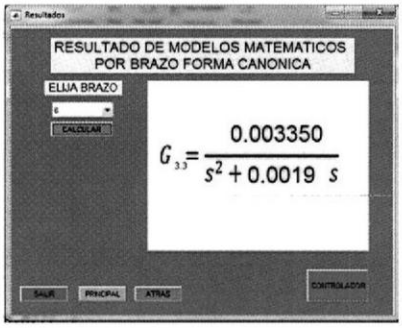


Figura 4.13: Comparación del modelo del brazo3 propuesta posición del brazo2 a 0° y carga a 0° cálculo manual y cálculo por Software.

Modelo del brazo3 propuesta, posición del brazo2 a 90° y carga a 90° (Teórico y Software):

$$\frac{\theta_L}{V_a} = \frac{3.34x10^{-3}}{S^2 + 1.88x10^{-3}\,S}$$

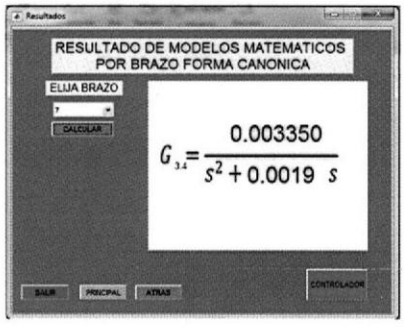


Figura 4.14: Comparación del modelo del brazo3 propuesta posición del brazo2 a 90° y carga a 0° cálculo manual y cálculo por Software.

4.2 Análisis de los diferentes controles difusos obtenidos durante la experimentación.

Para el desarrollo del controlador se probaron cambios relacionados al número de entradas al tipo de funciones de membresía y al número de reglas que se utilizaron, dentro de los más exitosos obtuvimos el primer ejemplo, el cual es un controlador mandami con 2 entradas y una salida, mientras que el segundo será un controlador sugeno de una entrada y una salida, donde el último fue el controlador seleccionado.

4.2.1 Parámetros del primer controlador Fuzzy diseñado para el control de posición de la plataforma.

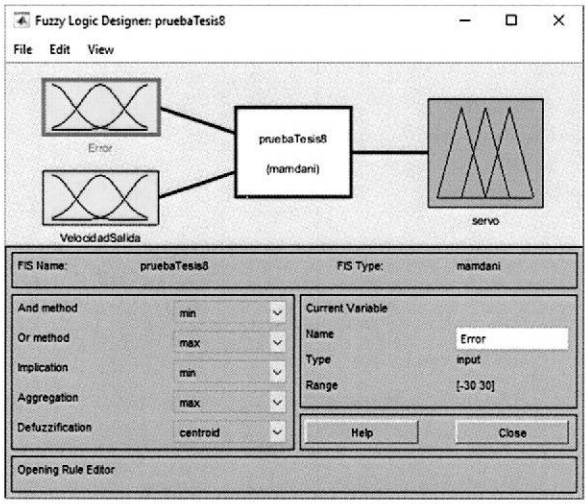


Figura 4.15: Ventana principal de la herramienta Fuzzy de Matlab

Primero debemos ingresa las 7 funciones de membresía y en este caso de tipo triangular con sus respectivos límites de operación, tanto del error y la derivada del error (Véase Figura 4.15 y Figura 4.16).

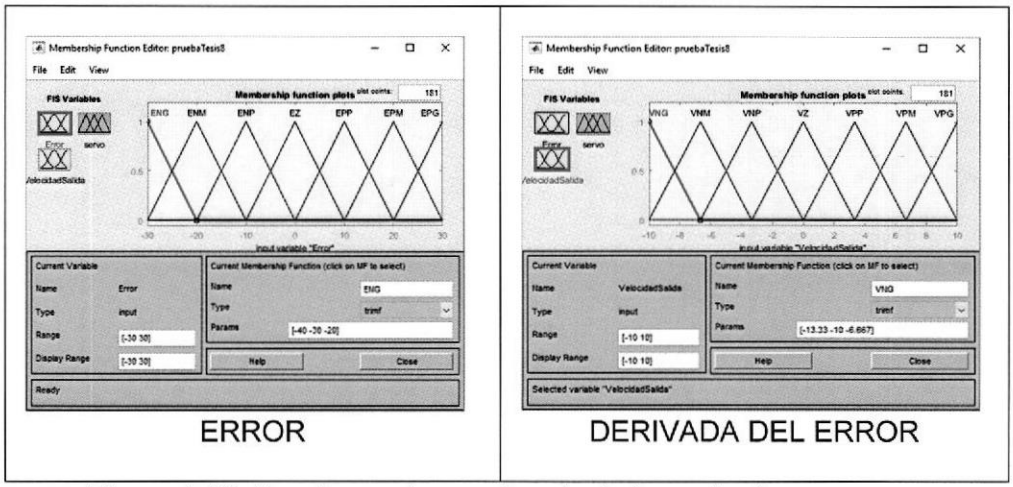


Figura 4.16: Funciones de membresía de las entradas

Luego las funciones de membresía de la salida con sus respectivos límites (Véase Figura 4.17):

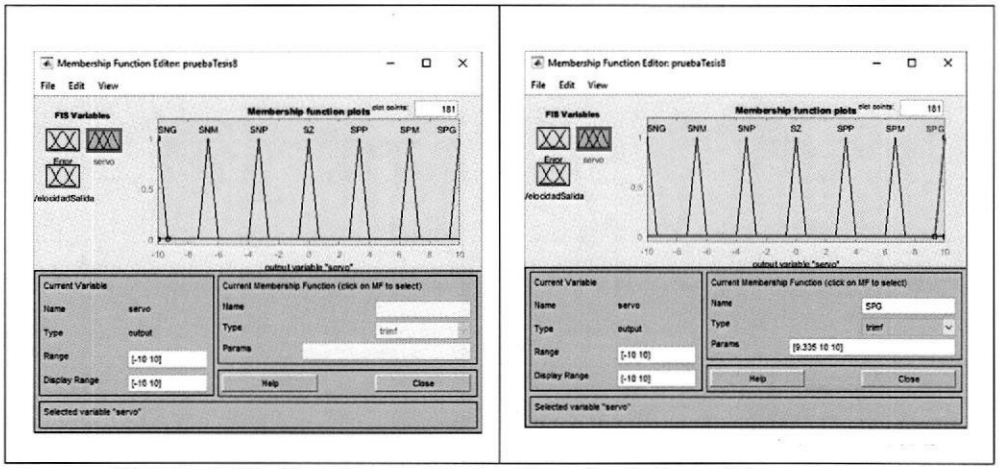


Figura 4.17: Funciones de membresía de la salida

Dado estos parámetros se deben ingresar las reglas para que el contralor difuso pueda inferir los valores que debe tomar a la salida a partir de un determinado valor en sus entradas (Véase Figura 4.18).

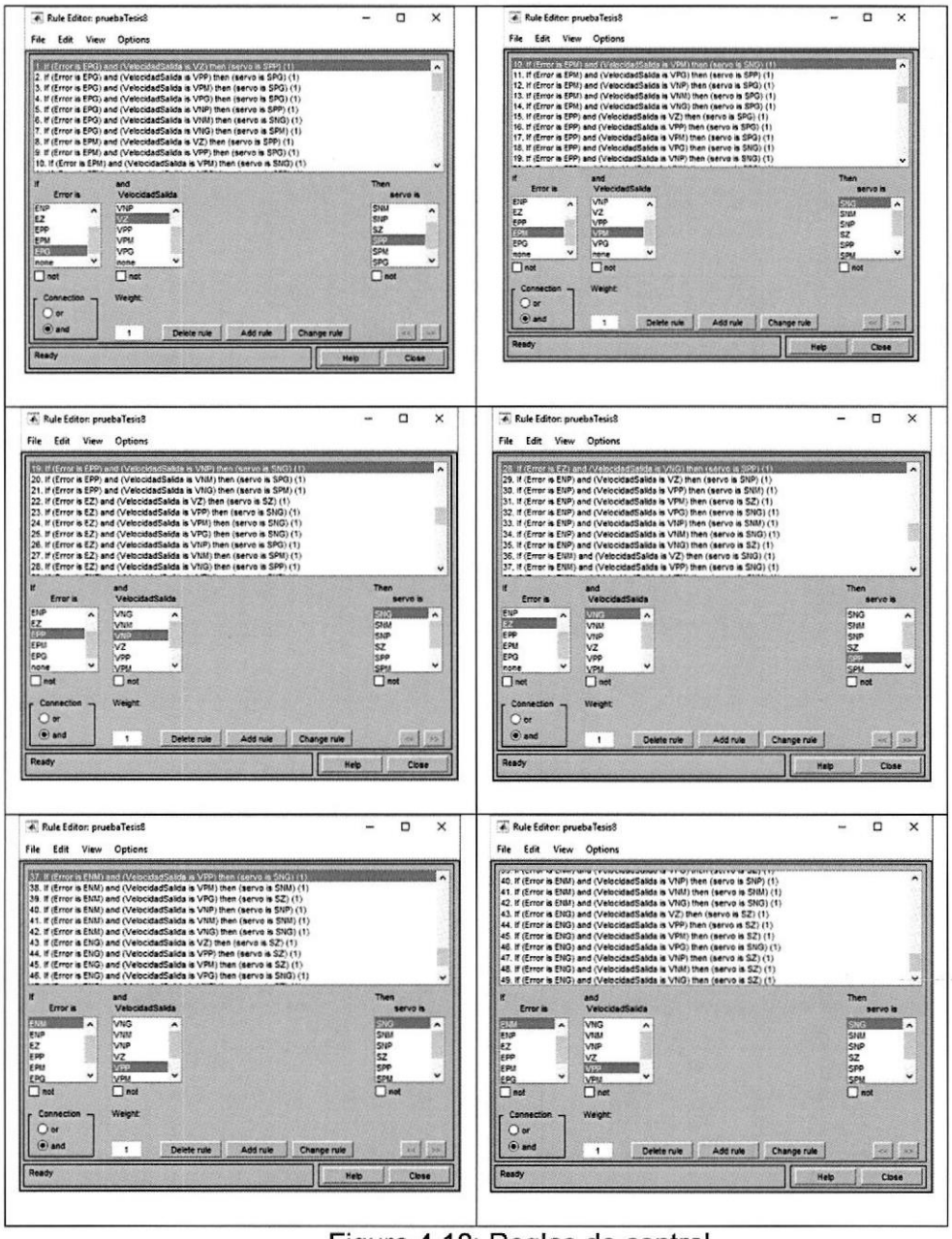


Figura 4.18: Reglas de control

También la herramienta nos permite ver la superficie de control, de tal forma que podemos relacionar las 2 entradas con la salida (Véase Figura 4.19):

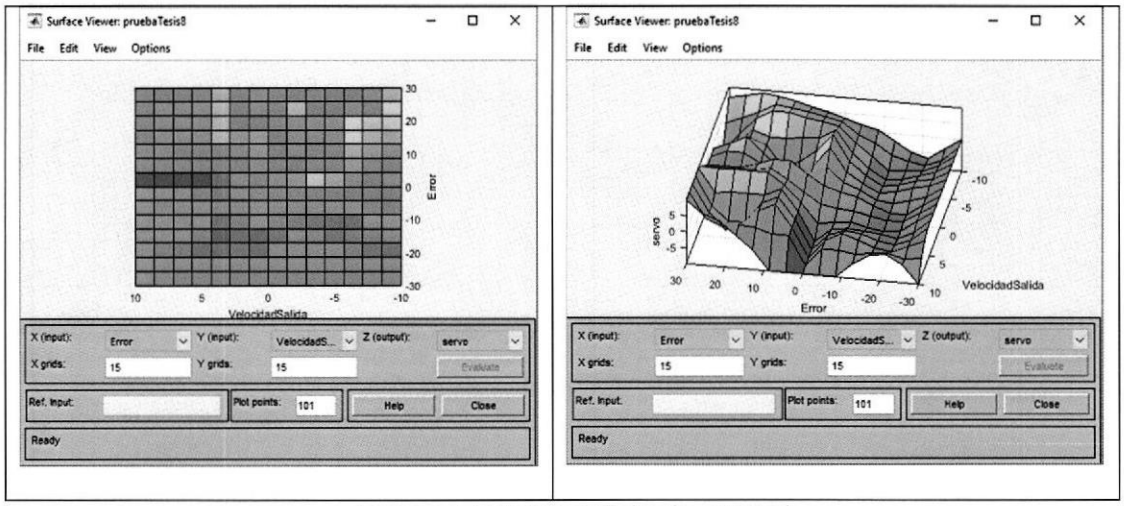


Figura 4.19: Superficie de control

Finalmente se puede simular los cambios en las entradas y ver su valor de salida (Véase Figura 4.20 - Figura 4.21):

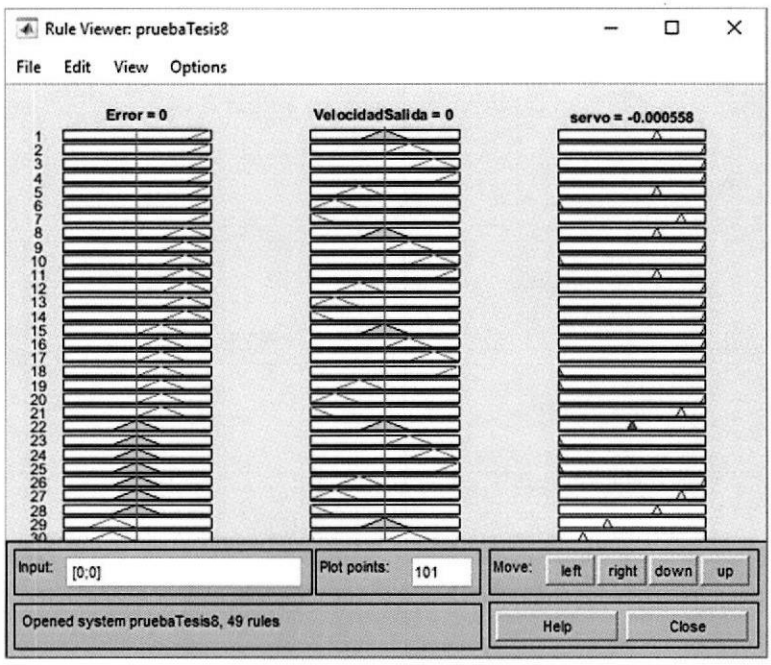


Figura 4.20: Respuesta cuando las 2 entradas son cero

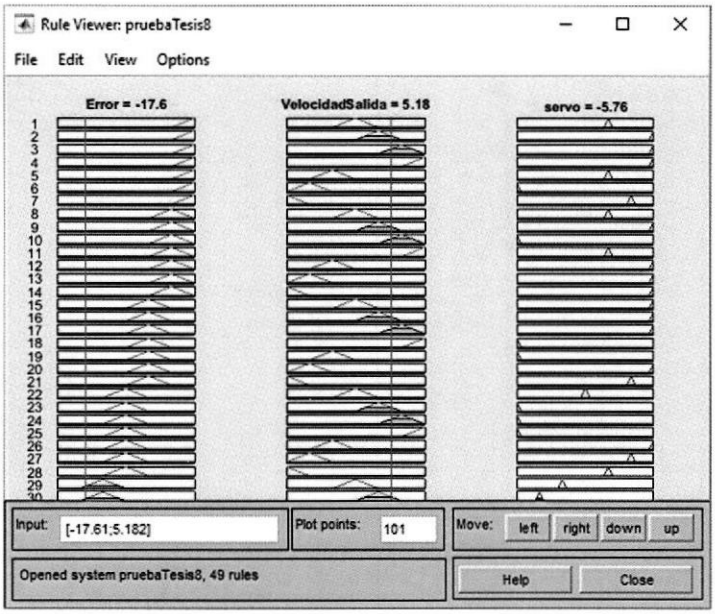


Figura 4.21: Respuesta cuando las 2 entradas son diferente de cero

4.2.2 Parámetros del segundo controlador Fuzzy diseñado para el control de posición de la plataforma.

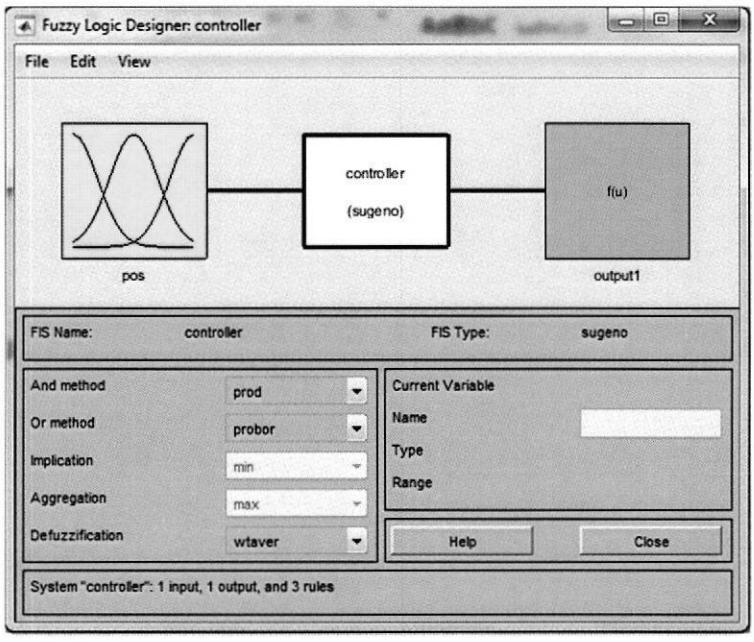


Figura 4.22: Ventana principal de la herramienta Fuzzy de Matlab

Primero debemos ingresa las 3 funciones de membresía y en este caso de tipo gaussiano con sus respectivos límites de operación, tanto del error y la derivada del error (Véase Figura 4.22 y Figura 4.23).

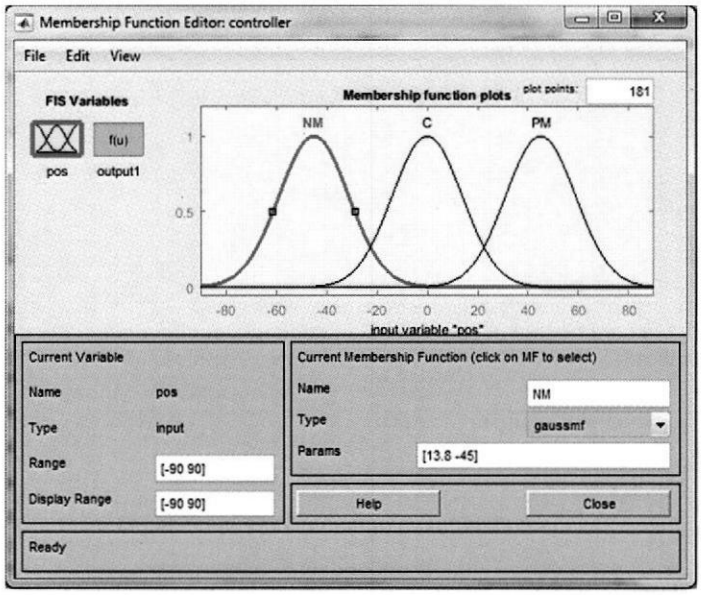


Figura 4.23: Funciones de membresía de la entrada del segundo controlador

Luego las funciones de membresía de la salida con sus respectivos límites (Véase Figura 4.24Figura 4.17):

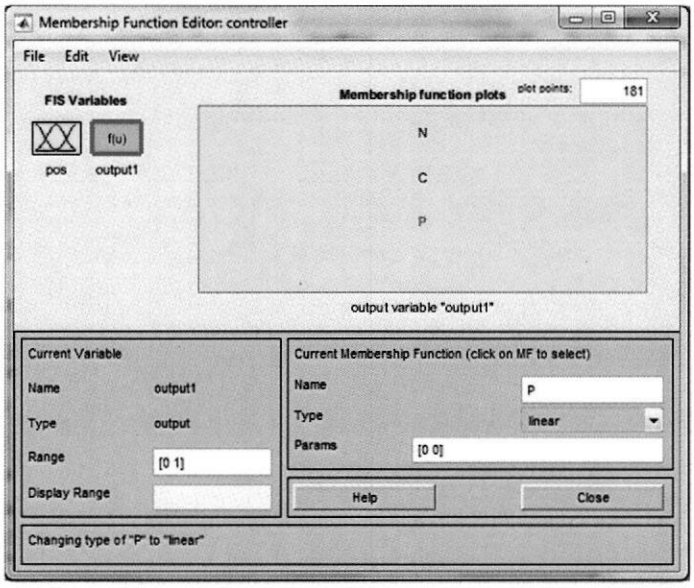


Figura 4.24: Funciones de membresía de la salida del segundo controlador

Dado estos parámetros se deben ingresar las reglas para que el contralor difuso pueda inferir los valores que debe tomar a la salida a partir de un determinado valor en sus entradas (Véase Figura 4.25).

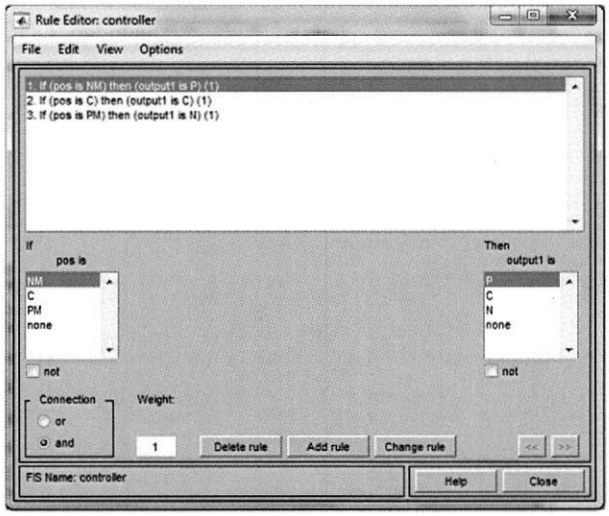


Figura 4.25: Base de reglas del segundo controlador

Finalmente se puede simular los cambios en las entradas y ver su valor de salida (Véase Figura 4.26 - Figura 4.28Figura 4.21):

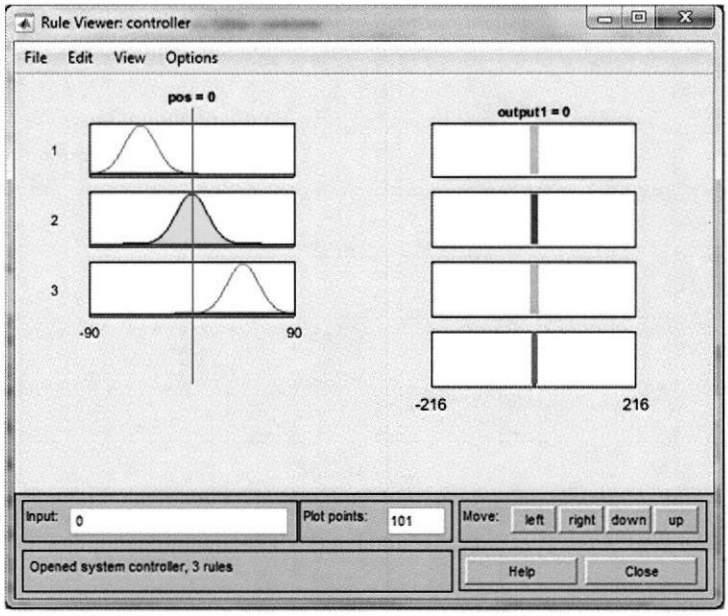


Figura 4.26: Visualizador de entradas salida del controlador punto medio

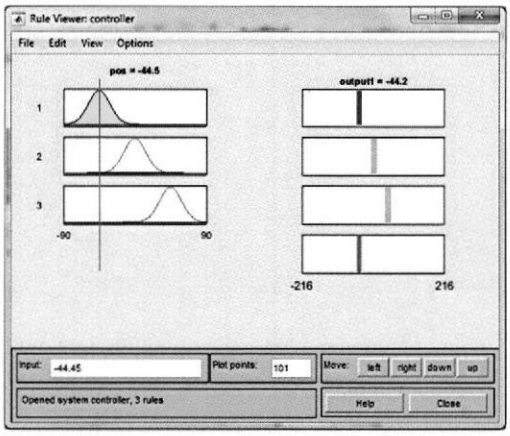


Figura 4.27: Visualizador de entradas salida del controlador referenciado izquierda

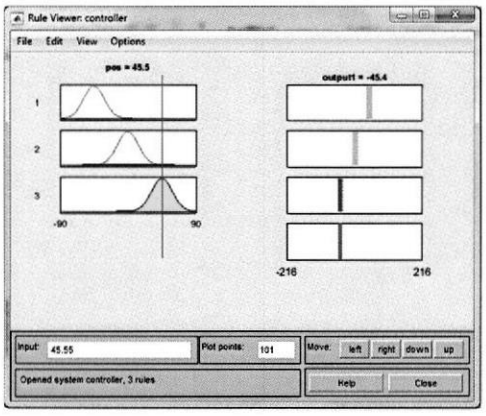


Figura 4.28: Visualizador de entradas salida del controlador referenciado derecha

También la herramienta nos permite ver la superficie de control, de tal forma que podemos relacionar las entradas con la salida (Véase Figura 4.29Figura 4.19):

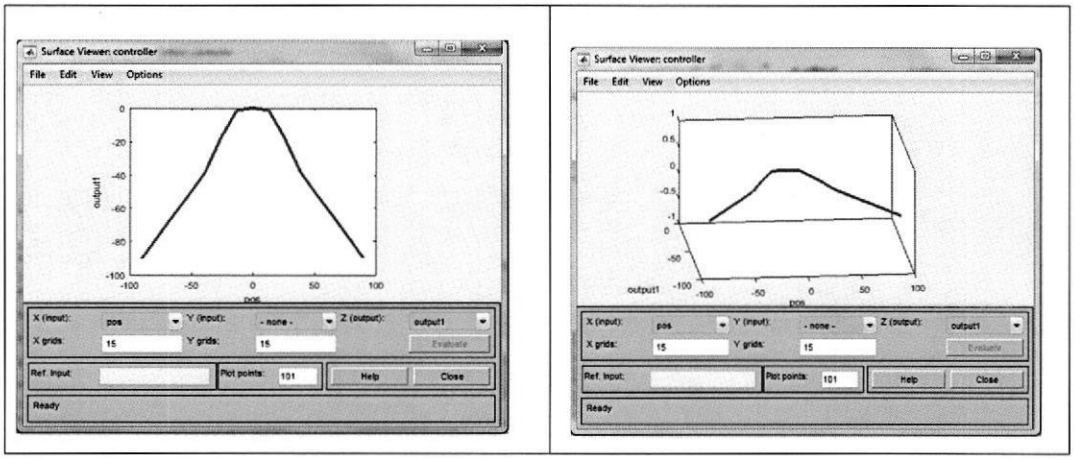


Figura 4.29: Superficie del segundo controlador

CONCLUSIONES Y RECOMENDACIONES

Se logró implementar una plataforma inercial giro-estabilizada con la capacidad de equilibrar una base con apuntador láser para emular un dispositivo de disparo, las partes se pueden mover en tres ejes, pero dos son debido a un controlador difuso desde Simulink y el otro es controlado por un joystick. También se implementó una plataforma que emula movimientos marinos que puede cambiar velocidades y ángulos de inclinación según sea la perturbación que se quiera presentar con la finalidad de poderlo llevar a embarcaciones donde se puedan apreciar.

Se desarrolló un software capaz de recibir los parámetros físicos del motor que con los cálculos y formulas obtenidas del modelo matemático se puede presentar de forma casi inmediata el nuevo modelo según los cambios que el usuario desee. Además, se programó un Arduino Mega2560 para que funcione como controlador de la plataforma de movimiento ondulatorio.

Se diseñó un controlador difuso de una entrada y una salida para el ajuste del ángulo de disparo de la base del apuntador láser, con sus respectivas reglas y funciones de membresía según la experiencia de control y de operación, se puede determinar que pese a obtener el modelo matemático este controlador no lo necesita y pese a ello puede controlar el funcionamiento del sistema.

Se recomienda mejorar las características del motor para aumentar la velocidad de reacción e incluso si algún lector lo prefiere puede modificar las dimensiones de la estructura a través del software desarrollado para disminuir los cambios inerciales de la carga.

BIBLIOGRAFÍA

[1] E. Saumeth, «Ecuador y la modernización de sus Fuerzas Armadas,» 5 Enero 2016. [En línea]. Available: http://www.infodefensa.com/latam/2016/01/05/opinion-ecuador-modernizacion-fuerzas-armadas.php.

[2] Agencia Pública de Noticias del Ecuador y Suramérica, «Astilleros Navales Ecuatorianos (Astinave) entregó lancha guardacostas "Isla Santa Cruz",» 20 Diciembre 2012. [En línea]. Available: http://www.andes.info.ec/es/actualidad/astilleros-navales-ecuatorianos-astinave-entreg%C3%B3-lancha-guardacostas-%E2%80%9Cisla-santa-cruz%E2%80%9D.

[3] E. Saumeth, «Colombia apuesta por el desarrollo de la industria local en materia de defensa,» 25 Noviembre 2015. [En línea]. Available: http://www.infodefensa.com/latam/2015/11/25/noticia-industria-defensa-colombia-potenciar-sector-hasta-tener-armada-hecha.html.

[4] Z. Kovačić y B. Stjepan, Fuzzy Controller Design Theory and Applications, Boca Raton: Taylor and Francis, 2005.

[5] A. Ferreyra, R. Fuentes y E. Sacristan, *CONTROL DIFUSO UNA ALTERNATIVA PARA APLICACIONES DE ALTA PRECISION,* Ensenada B.C.N., 1998.

[6] El Telégrafo, «64 toneladas de estupefacientes fueron incautadas en lo que va del 2016,» 16 Septiembre 2016.

[7] J. Cisneros, *Diseño de un control inteligente usando técnicas de lóica difusa, aplicado a un balastro electrónico,* Aguascalientes, 2016.

[8] A. Ancajima, *Lógica difusa y sistemas de control,* Piura, 2000.

[9] I. Jimenéz, *Control de temperatura de un horno eléctrico,* Huajuapan de León, 2012.

[10] Universidad de las Américas Puebla, 21 Julio 2009. [En línea]. Available: http://catarina.udlap.mx/u_dl_a/tales/documentos/lmt/maza_c_ac/capitulo2.pdf. [Último acceso: 9 julio 2017].

[11] C. Urrea y J. Kern, «A New Model for Analog Servo Motors,» *Canadian Journal on Automation, Control and Intelligent Systems,* vol. 2, nº 2, p. 10, 2011.

[12] Arduino, «ARDUINO MEGA 2560 REV3,» [En línea]. Available: https://store.arduino.cc/usa/arduino-mega-2560-rev3. [Último acceso: 08 Julio 2017].

[13] A. Birkett, «Bitify,» 6 Noviembre 2013. [En línea]. Available: http://blog.bitify.co.uk/2013/11/interfacing-raspberry-pi-and-mpu-6050.html.

[14] Robologs, «Tutorial de Arduino y MPU-6050,» 15 Octubre 2014. [En línea]. Available: https://robologs.net/2014/10/15/tutorial-de-arduino-y-mpu-6050/.

[15] HITEC RCD USA, Inc., «HS-311 Standard Economy Servo,» [En línea]. Available: http://hitecrcd.com/products/servos/sportservos/analog-sport-servos/hs-311-standard-economy-servo/product.

[16] L. Llamas, «CONTROLA TUS PROYECTOS CON ARDUINO Y JOYSTICK ANALÓGICO,» 8 Julio 2016. [En línea]. Available: https://www.luisllamas.es/arduino-joystick/. [Último acceso: 9 Julio 2017].

[17] Arduino, «Arduino - HelloWorld,» [En línea]. Available: https://www.arduino.cc/en/Tutorial/HelloWorld. [Último acceso: 08 Julio 2017].

[18] Instituto Nacional de Educación Tecnológica, «LOS MANDOS,» 12 Diciembre 2000. [En línea]. Available: http://www.oni.escuelas.edu.ar/2003/BUENOS_AIRES/62/tecnolog/mandos.htm.

ANEXO A

Descripción de las ventanas de la guía desarrollada

Para ingresar al software se debe presionar el botón verde INICIO, con lo cual nos lleva a una ventana que explica los pasos a seguir para obtener el modelo matemático, en el caso que se desee salir puede hacerlo con el botón rojo SALIR.

Figura 1: Ventana Principal

Luego, se presenta una ventana indicando los pasos a seguir, presionando el botón verde SIGUIENTE nos dirigimos a una nueva ventana, con el botón rojo SALIR nos saca del programa y con el botón gris atrás nos vamos a la ventana principal.

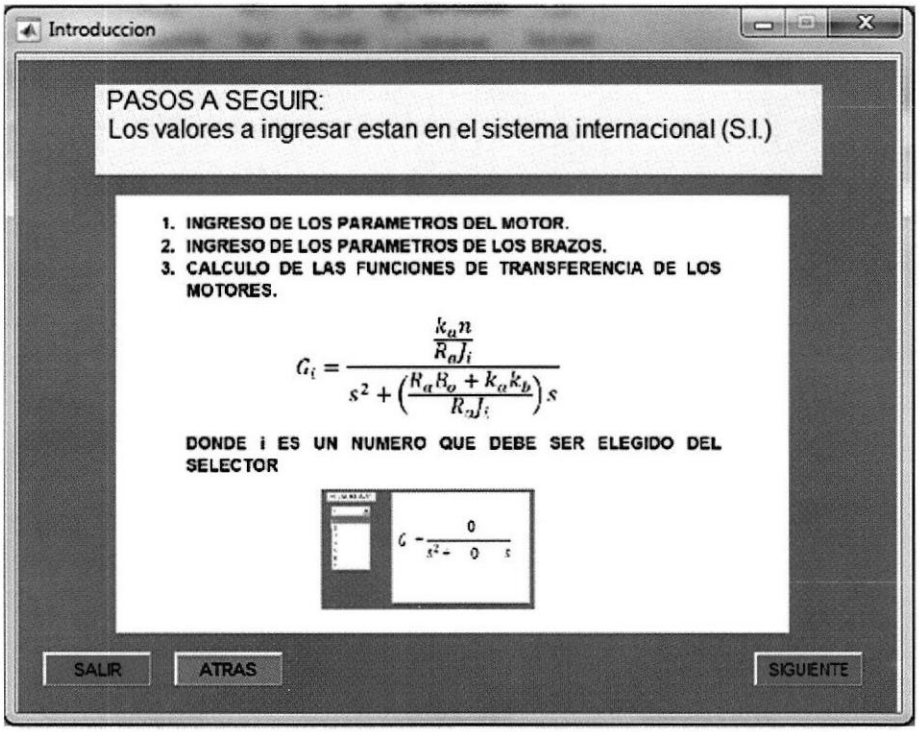


Figura 2: Ventana pasos a seguir

En la presente ventana se realiza la petición de los parámetros del motor. Al presionar el botón rojo SALIR finaliza la ejecución del programa. Al ser presionado el botón gris ATRAS retorna a la ventana pasos a seguir, si se presiona el botón naranja PRINCIPAL se procede a presentar ventana principal cerrando la ventana actual, presionando el botón verde SIGUIENTE nos dirigimos a una nueva ventana. En la esquina inferior derecha se puede apreciar un botón gris de mayor tamaño PREDETERMINADO que al ser presionado coloca los valores por defecto del circuito, estos valores se pueden apreciar en el esquema, en la parte derecha del programa se presenta las etiquetas y cuadros de texto donde el usuario puede ingresar los valores que él decida, al presionar el botón verde oscuro LISTO almacena los valores de las variables y los presentan en el circuito.

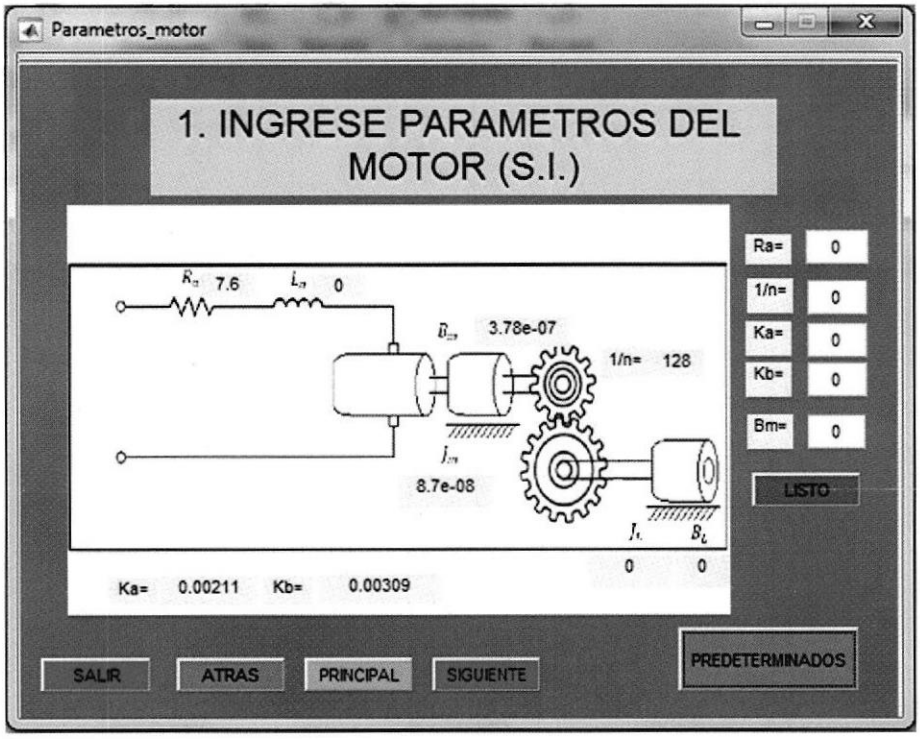


Figura 3: Ventana Parametro de motor.

Luego, se presenta una ventana la cual indica el ingreso de las dimensiones de la estructura a usar, estos datos serán usando para calcular la inercia total. En esta ventana al presionar el botón rojo SALIR se pone fin a la ejecución del programa, el botón naranja PRINCIPAL hace un salto a la ventana principal, si se presiona el botón gris ATRÁS se retrocede a la ventana Parámetro de motor. Al presionar el botón gris PREDETERMINADO los datos almacenado en el código se presentan por pantalla indicando las dimensiones de las estructuras. Si el usuario desea ingresar las dimensiones de la estructura, al finalizar la elección de cada brazo debe presionar el botón verde oscuro LISTO para almacenar los datos en las variables presentadas por pantalla. Al presionar el botón verde GENERAR G(S) nos dirigimos a una nueva ventana.

Figura 4: Ventana de Parámetros del Brazo 1

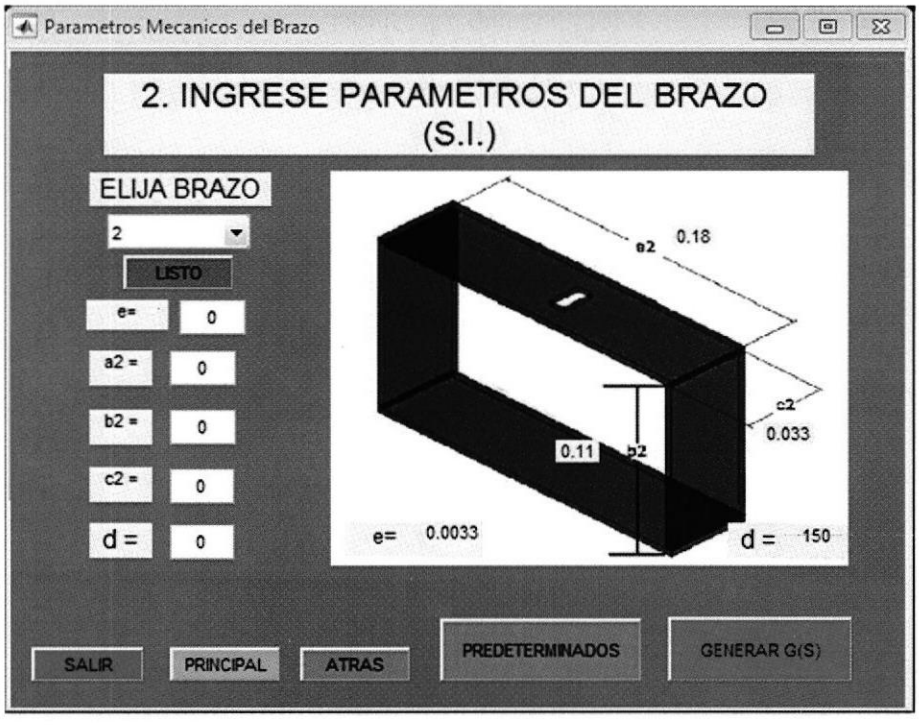


Figura 5: Ventana de Parámetros del Brazo 2

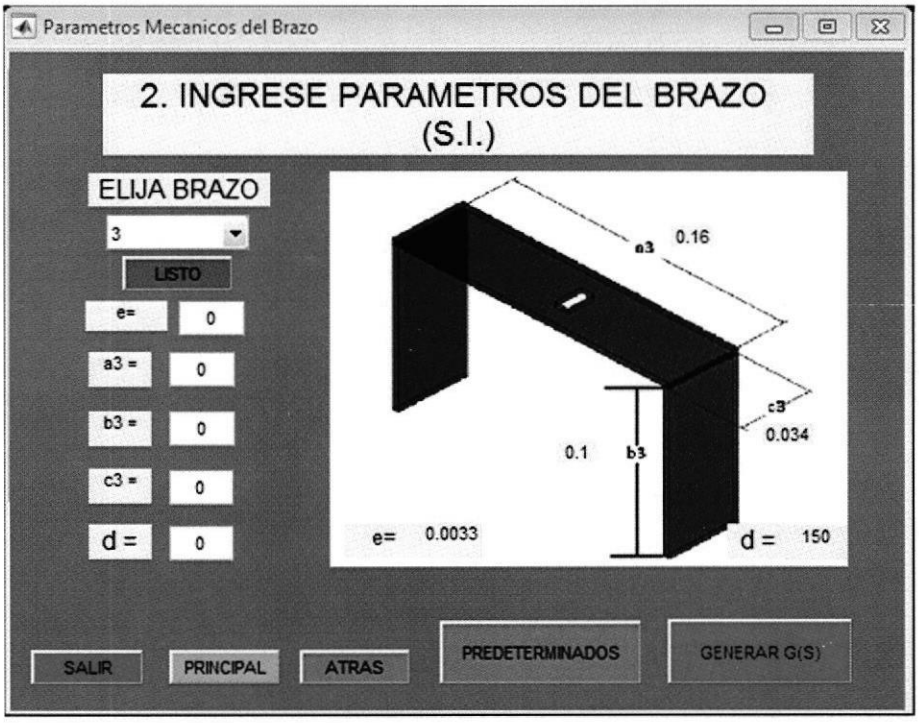


Figura 6: Ventana de Parámetros del Brazo 3

En esta ventana se procede a calcular las funciones de transferencia de cada brazo, al escoger una de las siete opciones mostrada en el menú desplegable, al presionar el botón gris CALCULAR presenta la ecuación canónica resultante indicando en el subíndice al brazo que pertenece. Al presionar el botón rojo SALIR da fin a la ejecución de programa, al presionar el botón naranja PRINCIPAL retorna a la ventana principal cerrando la actual, al presionar el botón gris ATRÁS se retorna a la ventana Parámetros.m del Brazo, si se presiona el botón verde CONTROLADOR nos lleva a la ventana de Simulink.

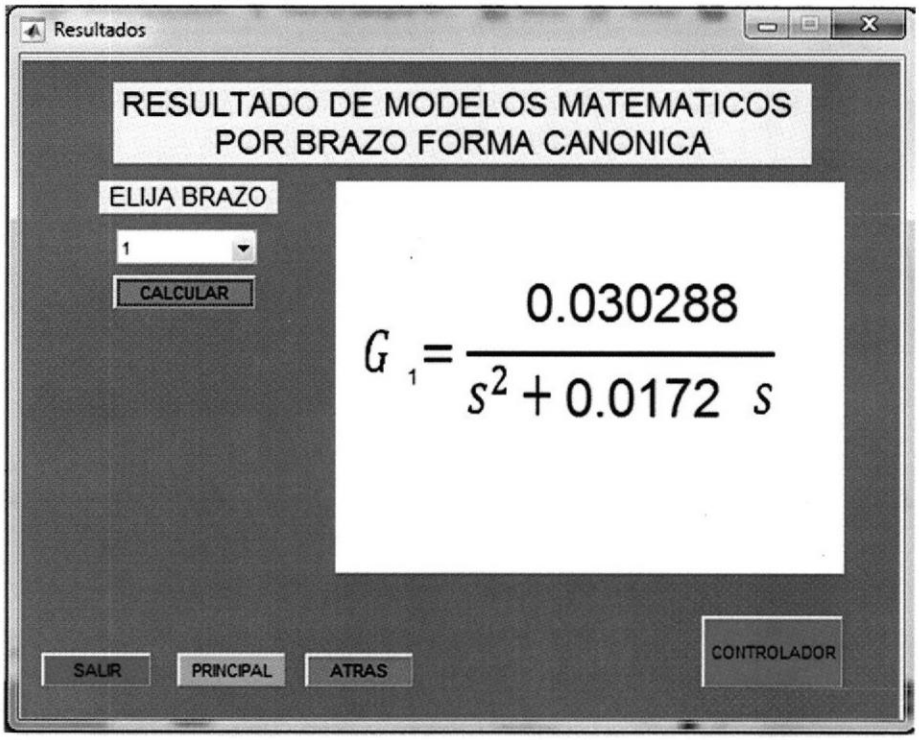


Figura 7: Ventana Funcion de transferencia G1

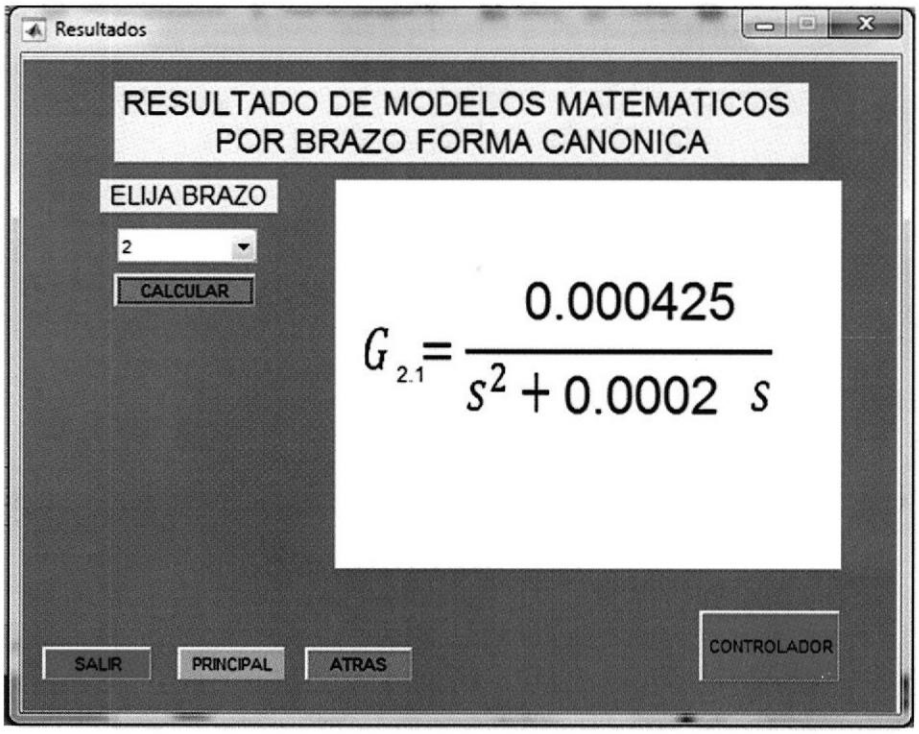


Figura 8: Ventana de Función de transferencia G2.1

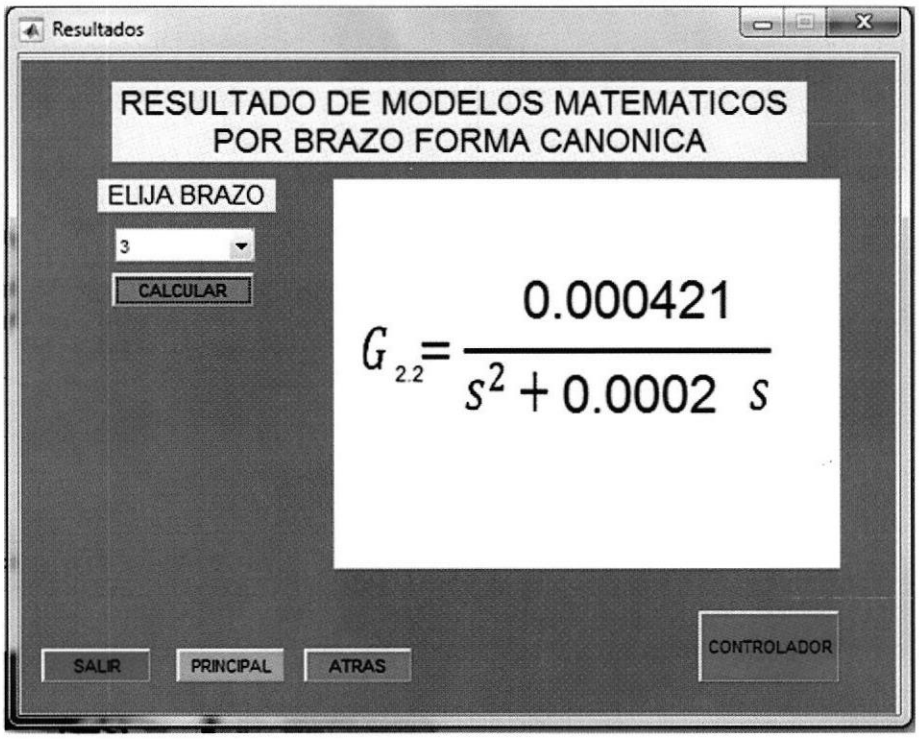


Figura 9: Ventana Función de transferencia G2.2

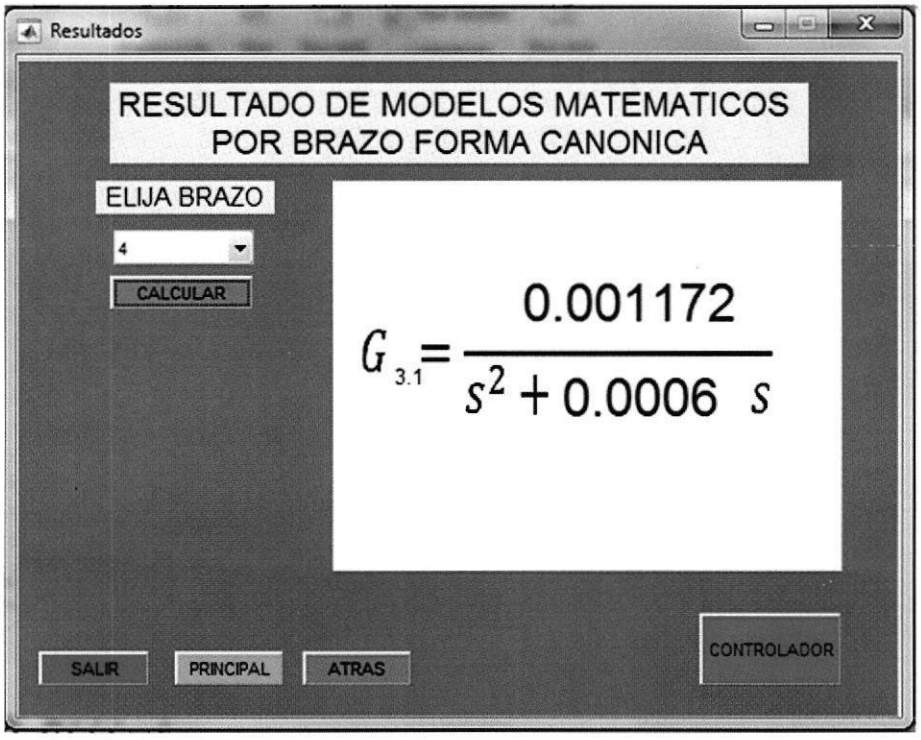


Figura 10: Ventana de Función de transferencia G3.1

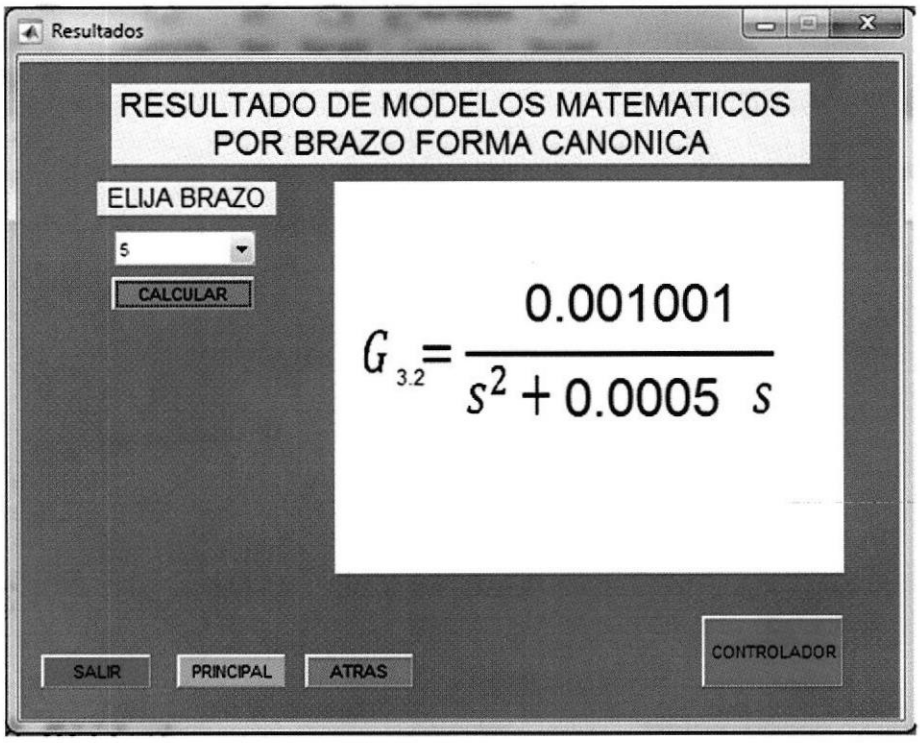


Figura 11: Ventana de Función de transferencia G3.2

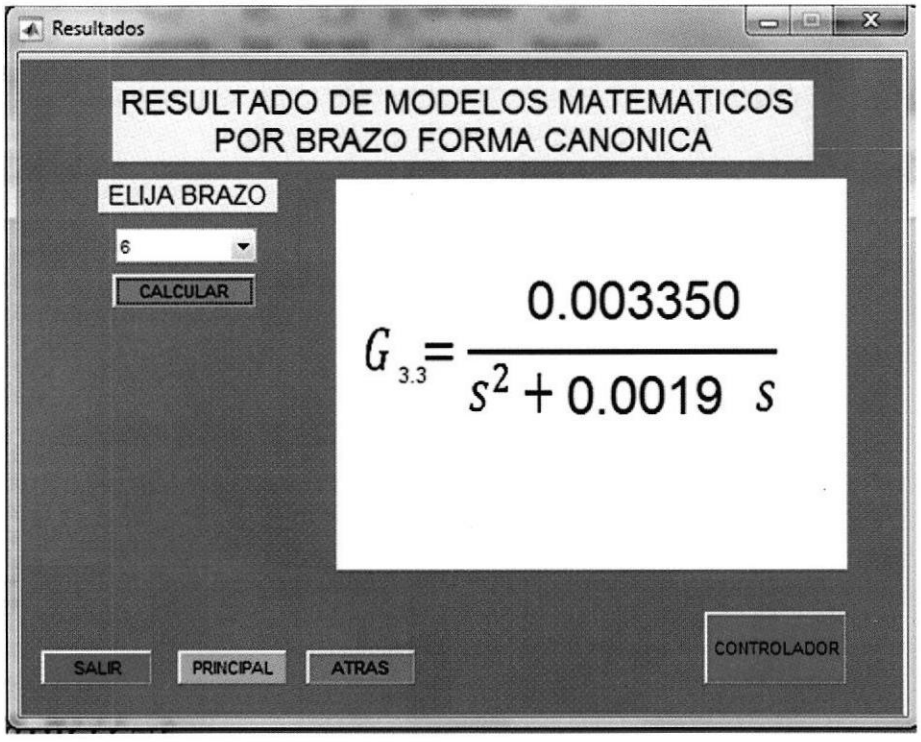


Figura 12: Ventana de Función de transferencia G3.3

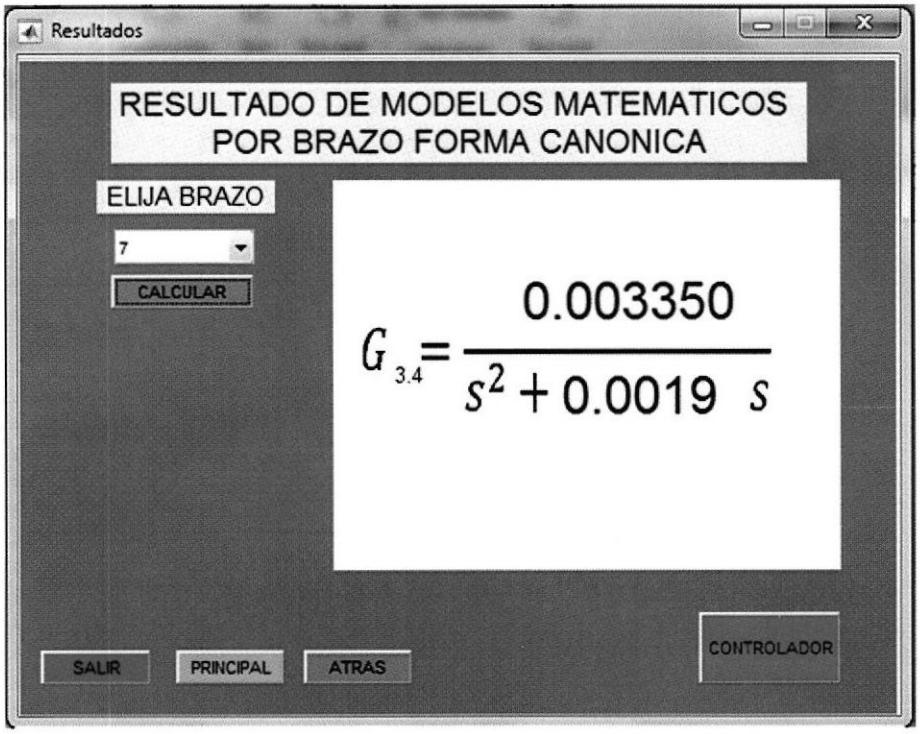


Figura 13: Ventana de Función de Transferencia G3.4

ANEXO B

Pasos para conectar Arduino con Simulink

Para poder conectar Arduino con Matlab o Simulink necesitamos primero descargar los paquetes y librerías correspondientes al Hardware de Arduino. Mathworks posee ya esta opción gratuita ingresando simplemente a la pestaña que se muestra en la Figura 14.

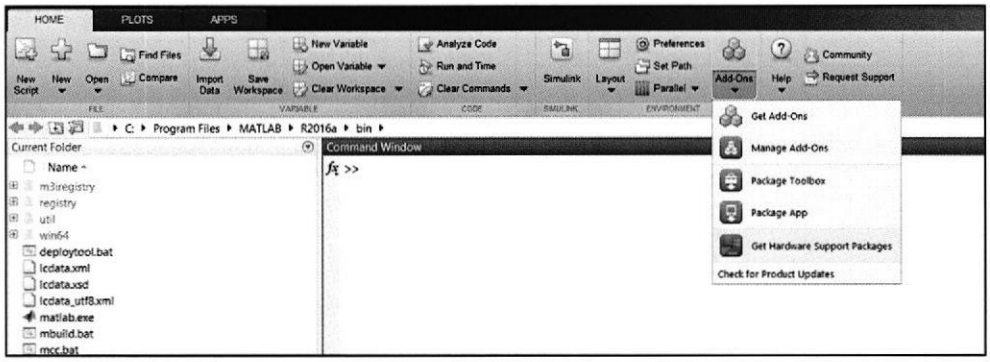


Figura 14: Descarga de paquetes y librerías de Arduino

Luego nos pedirá la opción de cómo queremos instalar el paquete, elegiremos la instalación desde Internet, como se muestra en la Figura 15.

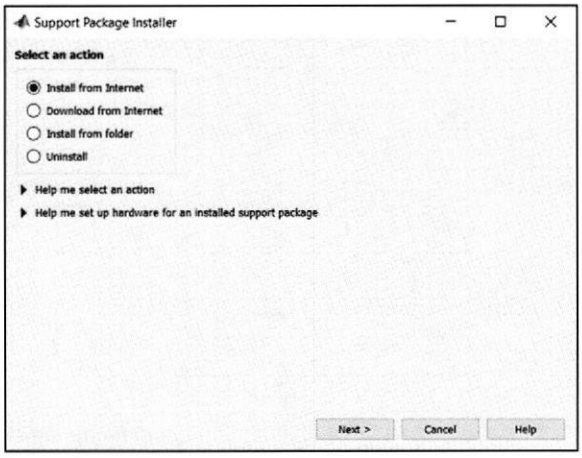


Figura 15: Bloque de selección del modo de instalación

En la Figura 16 tendremos una lista completa de los paquetes disponibles.

Elegiremos los dos que se encuentran en la pestaña Arduino, uno sirve para utilizar directamente el código en la ventana de comandos de Matlab y el otro para correr o cargar modelos de Simulink en nuestro Hardware de Arduino.

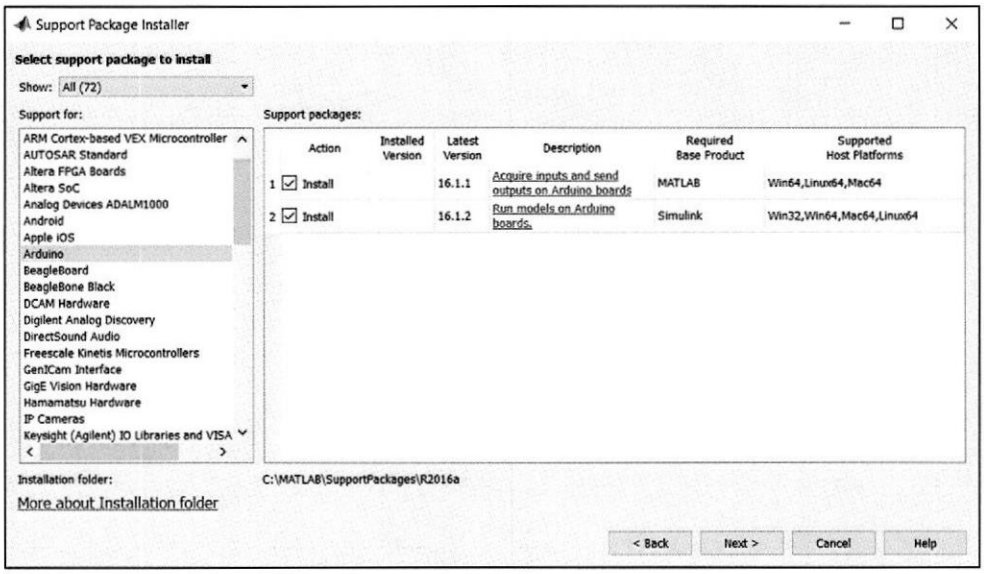


Figura 16: Listado de paquetes disponibles de Arduino

Se puede revisar la documentación en la página oficial de MathWorks para conocer si nuestra placa Arduino es soportada por el paquete. En este caso se trabajará con la placa de Arduino MEGA la cual si se encuentra en la lista.

Previo a la instalación nos pedirá que ingresemos los datos de nuestra cuenta de MathWorks (Figura 17). Podemos crear una gratuitamente en la misma ventana que se nos abre, no necesitamos una licencia personal para crear y utilizar la cuenta.

Figura 17: Registro de cuenta MathWorks para descarga

Posteriormente debemos revisar el acuerdo de licencia del paquete y dar a Siguiente (Véase Figura 18).

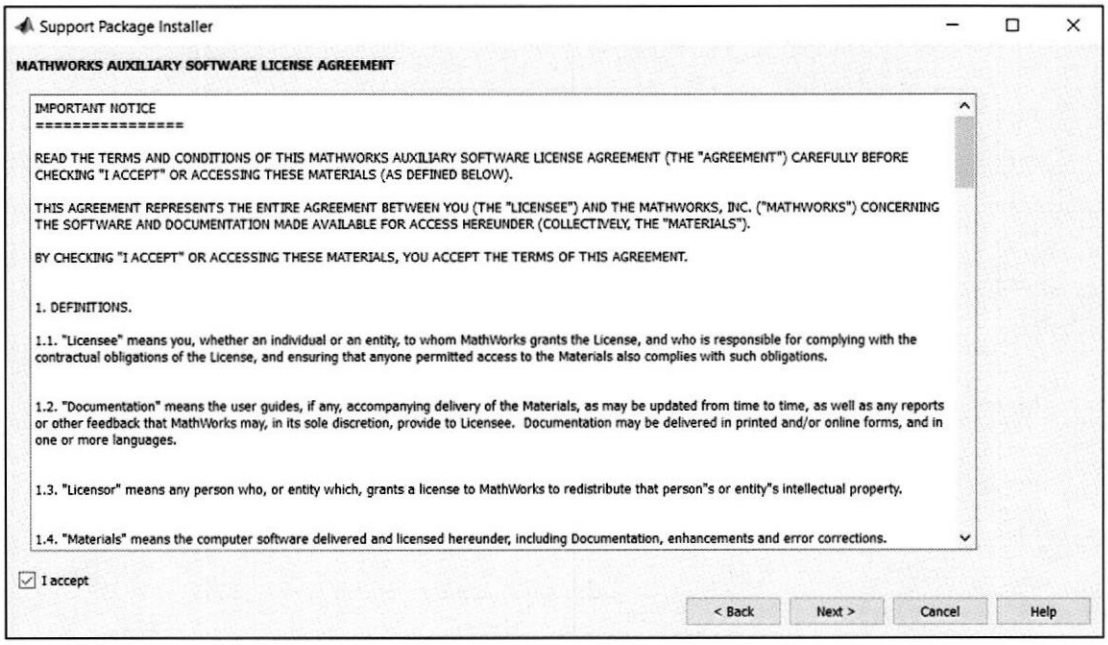


Figura 18: Bloque de especificación de paquetes a instalar

Nos encontraremos con una ventana donde se especifica los paquetes a instalar. Damos clic en instalar (Véase Figura 19) y cuando el proceso culmine nos aparecerá una ventana de confirmación como se muestra en la Figura 20.

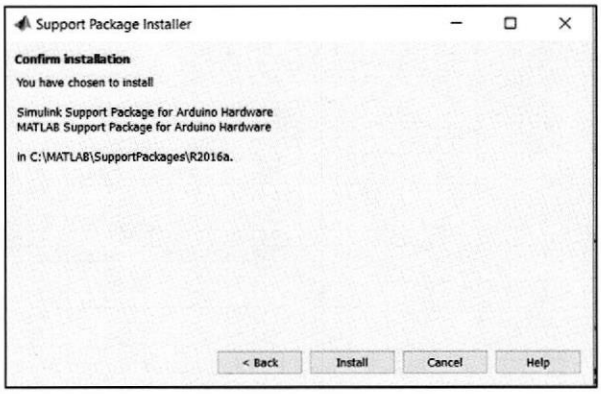


Figura 19: Bloque de confirmación de Instalación

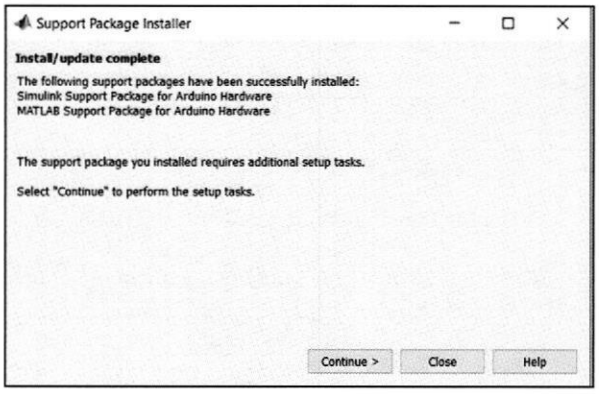


Figura 20: Bloque de instalación correcta

Teniendo ya instalado el paquete para Simulink. Podremos acceder a los objetos de la librería de Arduino como entradas, salidas y comunicación (Véase Figura 21).

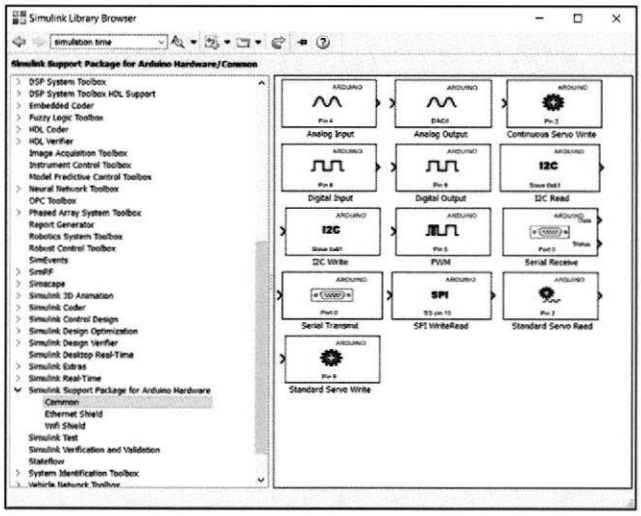


Figura 21: Bloques disponibles Hardware Arduino

Una vez creado nuestro modelo y para poder correrlo en nuestro Hardware es necesario realizar unas modificaciones en los parámetros de Simulación (Véase Figura 22 y Figura 23).

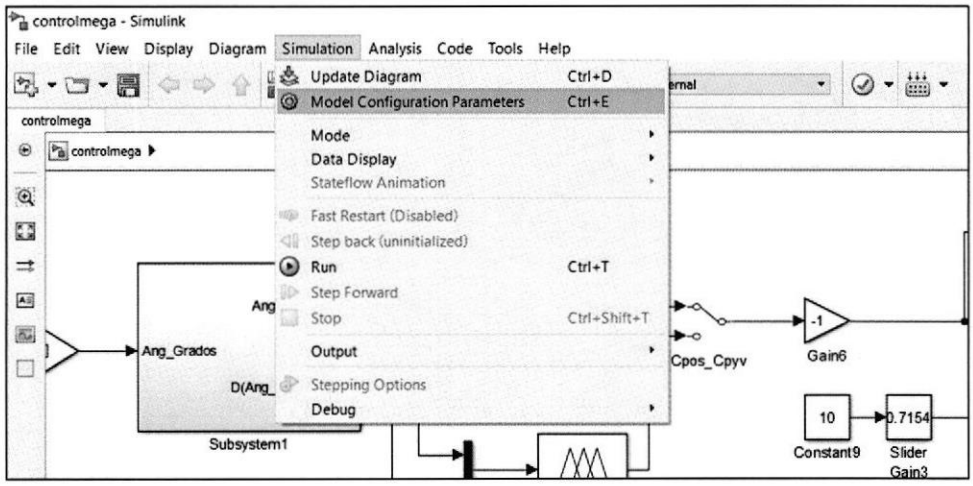


Figura 22: Configuración de parámetros del modelo

Primero colocamos un tiempo de muestreo de 30 ms y un tiempo de simulación Infinito.

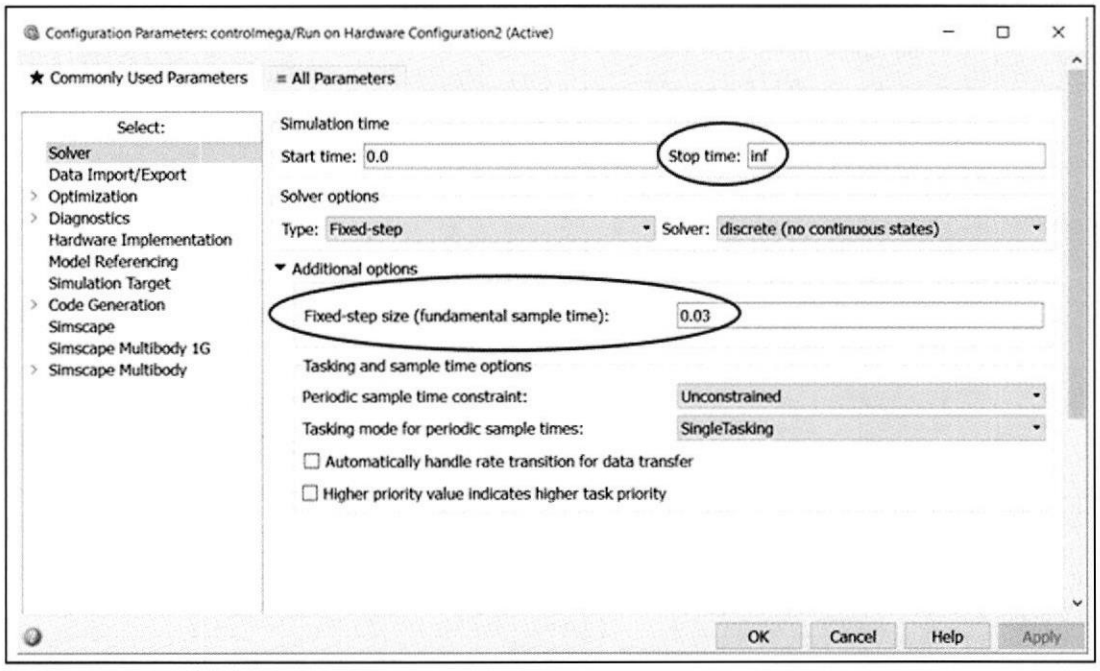


Figura 23: Configuración de los parámetros de simulación

Luego debemos seleccionar la placa con la cual vamos a trabajar. Luego de hacer esto, automáticamente nos indica qué fuente de código debemos utilizar para poder trabajar con esa placa, en este caso: ert.tlc (Véase Figura 24).

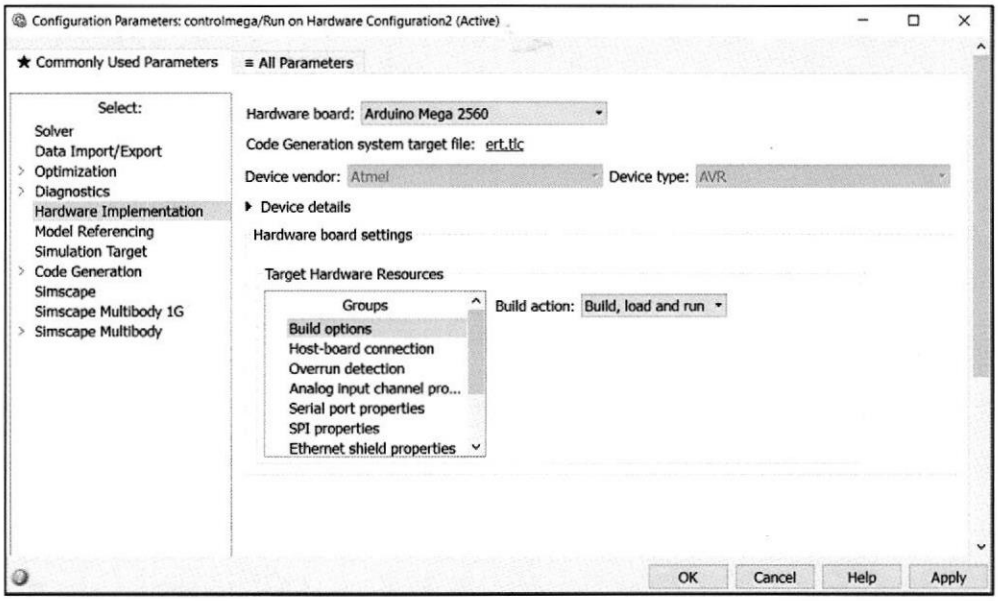


Figura 24: Selección de bloque del Hardware Arduino Mega 2560

Nos colocamos en la pestaña de generación de códigos y seleccionamos el mismo que se nos indicó en el paso anterior (Véase Figura 25).

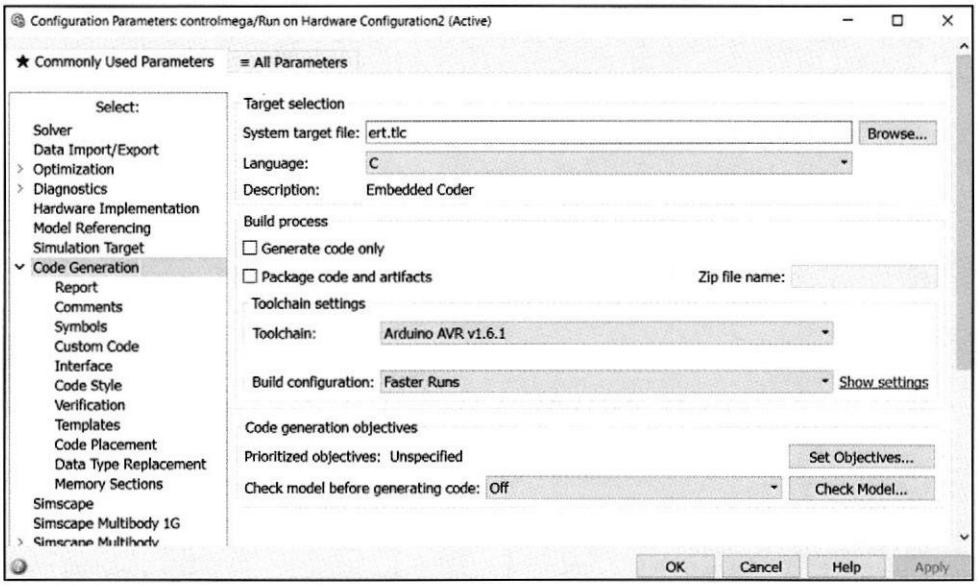


Figura 25: Selección de Lenguaje de programación

Por último, seleccionamos como tipo de señal de procesamiento EXTERNAL y podemos dar clic en RUN para comenzar a correr el programa (Véase Figura 26).

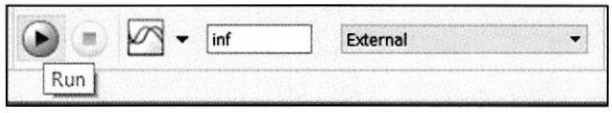


Figura 26: Inicio del proceso modo externo

Podemos ver el programa corriendo y funcionando en tiempo real (Véase Figura 27).

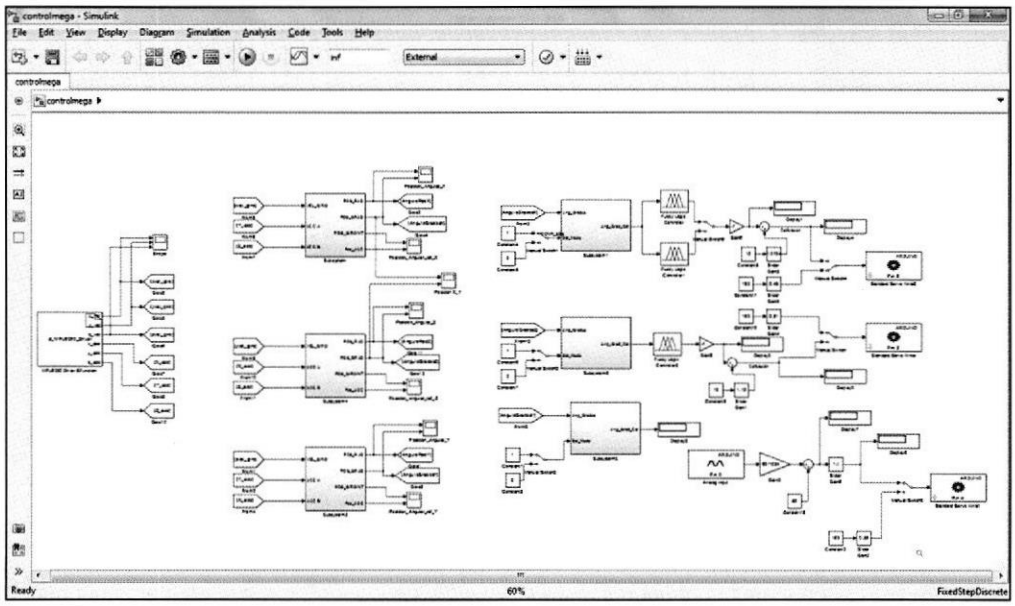


Figura 27: Bloque simulink para control difuso del sistema.

ANEXO C

Código fuente del archivo "primera.m"

En la Figura 28 y Figura 29 se presenta el código fuente del archivo "primera.m" con sus comentarios correspondientes.

```
65
66      % --- Outputs from this function are returned to the command line.
67      function varargout = PRIMERA_OutputFcn(hObject, eventdata, handles)
68      % varargout  cell array for returning output args (see VARARGOUT);
69      % hObject    handle to figure
70      % eventdata  reserved - to be defined in a future version of MATLAB
71      % handles    structure with handles and user data (see GUIDATA)
72
73      % Get default command line output from handles structure
74 -    varargout{1} = handles.output;
75
76      %_____________________________________________________________________
77      %                          BOTON INICIO
78      %_____________________________________________________________________
79      % --- Executes on button press in pushbutton1.
80      function pushbutton1_Callback(hObject, eventdata, handles)
81      % hObject    handle to pushbutton1 (see GCBO)
82      % eventdata  reserved - to be defined in a future version of MATLAB
83      % handles    structure with handles and user data (see GUIDATA)
84      %//////////winopen('segunda.fig')
85      %botok=uicontrol('String','INICIO','Callback','clear all; close all;clc; segu
86
87 -    assignin('base','a1',0);
88 -    assignin('base','b1',0);
89 -    assignin('base','c1',0);
90
91 -    assignin('base','a2',0);
92 -    assignin('base','b2',0);
93 -    assignin('base','c2',0);
94
95 -    assignin('base','a3',0);
96 -    assignin('base','b3',0);
97 -    assignin('base','c3',0);
```

Figura 28: Código Fuente "primera.m" (PARTE 1).

```
 98
 99 -   assignin('base','Jm',0);
100 -   assignin('base','N',0);
101 -   assignin('base','Bm',0);
102 -   assignin('base','JL',0);
103 -   assignin('base','Ra',0);
104 -   assignin('base','La',0);
105 -   assignin('base','K',0);
106 -   assignin('base','t1',0);
107 -   assignin('base','t2',0);
108 -   assignin('base','t3',0);
109 -   assignin('base','densidad',0);
110 -   assignin('base','e',0);
111 -   Introduccion
112 -   close(gcbf)
113
114     % --------------------------------------------------------------------
115     %function INICIO_Callback(hObject, eventdata, handles)
116     % hObject    handle to INICIO (see GCBO)
117     % eventdata  reserved - to be defined in a future version of MATLAB
118     % handles    structure with handles and user data (see GUIDATA)
119
120     %_____________________________________________________________________
121     %                           Salir
122     %_____________________________________________________________________
123     % --- Executes on button press in pushbutton2.
124     function pushbutton2_Callback(hObject, eventdata, handles)
125     % hObject    handle to pushbutton2 (see GCBO)
126     % eventdata  reserved - to be defined in a future version of MATLAB
127     % handles    structure with handles and user data (see GUIDATA)
128 -   clear all;
129 -   clc;
130 -   close(gcbf)
131
```

Figura 29: Código Fuente "primera.m" (PARTE 2).

ANEXO D

Código fuente del archivo "introduccion.m"

En la Figura 30 y Figura 31 se presenta el código fuente del archivo "introduccion.m" con sus comentarios correspondientes.

```
45
46
47      % --- Executes just before Introduccion is made visible.
48      function Introduccion_OpeningFcn(hObject, eventdata, handles, varargin)
49      % This function has no output args, see OutputFcn.
50      % hObject    handle to figure
51      % eventdata  reserved - to be defined in a future version of MATLAB
52      % handles    structure with handles and user data (see GUIDATA)
53      % varargin   command line arguments to Introduccion (see VARARGIN)
54
55 -    picture=imread('figuraINTRO1.jpg');
56 -    image(picture)
57 -    axis off
58
59      % Choose default command line output for Introduccion
60 -    handles.output = hObject;
61
62      % Update handles structure
63 -    guidata(hObject, handles);
64
65      % UIWAIT makes Introduccion wait for user response (see UIRESUME)
66      % uiwait(handles.figure1);
67
68
69      % --- Outputs from this function are returned to the command line.
70      function varargout = Introduccion_OutputFcn(hObject, eventdata, handles)
71      % varargout  cell array for returning output args (see VARARGOUT);
72      % hObject    handle to figure
73      % eventdata  reserved - to be defined in a future version of MATLAB
74      % handles    structure with handles and user data (see GUIDATA)
75
76      % Get default command line output from handles structure
77 -    varargout{1} = handles.output;
78
79      %_____Siguiente_____
80      % --- Executes on button press in pushbutton1.
81      function pushbutton1_Callback(hObject, eventdata, handles)
82      % hObject    handle to pushbutton1 (see GCBO)
83      % eventdata  reserved - to be defined in a future version of MATLAB
84      % handles    structure with handles and user data (see GUIDATA)
85 -    Parametros_motor
86 -    close(gcbf)
87
88
```

Figura 30: Código Fuente "introduccion.m" (PARTE 1).

```
111      %__________Salir_____________
112      % --- Executes on button press in pushbutton2.
113      function pushbutton2_Callback(hObject, eventdata, handles)
114      % hObject    handle to pushbutton2 (see GCBO)
115      % eventdata  reserved - to be defined in a future version of MATLAB
116      % handles    structure with handles and user data (see GUIDATA)
117 -    clear all;
118 -    clc;
119 -    close(gcbf)
120
121      %_______PRINCIPAL____________________
122      % --- Executes on button press in pushbutton3.
123      function pushbutton3_Callback(hObject, eventdata, handles)
124      % hObject    handle to pushbutton3 (see GCBO)
125      % eventdata  reserved - to be defined in a future version of MATLAB
126      % handles    structure with handles and user data (see GUIDATA)
127 -    PRIMERA
128 -    close(gcbf)
129
```

Figura 31: Código Fuente “introduccion.m” (PARTE 2).

ANEXO E

Código fuente del archivo "parametros_motor.m"

En la Figura 32, Figura 33 y Figura 34 se presenta el código fuente del archivo "parametros_motor.m" con sus comentarios correspondientes.

```
43 -    end
44      % End initialization code - DO NOT EDIT
45
46      %_____________Imagen_inicial_____________________________________________
47      % --- Executes just before Parametros_motor is made visible.
48      function Parametros_motor_OpeningFcn(hObject, eventdata, handles, varargin)
49      % This function has no output args, see OutputFcn.
50      % hObject    handle to figure
51      % eventdata  reserved - to be defined in a future version of MATLAB
52      % handles    structure with handles and user data (see GUIDATA)
53      % varargin   command line arguments to Parametros_motor (see VARARGIN)
54 -    a=imread('circuito.jpg');
55 -    image(a)
56 -    axis off
57      % Choose default command line output for Parametros_motor
58 -    handles.output = hObject;
59
60      % Update handles structure
61 -    guidata(hObject, handles);
62
63      % UIWAIT makes Parametros_motor wait for user response (see UIRESUME)
64      % uiwait(handles.figure1);
65
66
67      % --- Outputs from this function are returned to the command line.
68      function varargout = Parametros_motor_OutputFcn(hObject, eventdata, handles)
69      % varargout  cell array for returning output args (see VARARGOUT);
70      % hObject    handle to figure
71      % eventdata  reserved - to be defined in a future version of MATLAB
72      % handles    structure with handles and user data (see GUIDATA)
73
74      % Get default command line output from handles structure
75 -    varargout{1} = handles.output;
76
77      %_____________Salir_____________________________________________
78      % --- Executes on button press in pushbutton1.
79      function pushbutton1_Callback(hObject, eventdata, handles)
80      % hObject    handle to pushbutton1 (see GCBO)
81      % eventdata  reserved - to be defined in a future version of MATLAB
82      % handles    structure with handles and user data (see GUIDATA)
83 -    clear all;
84 -    clc;
85 -    close(gcbf)
86
87      %_____________SIGUIENTE_____________________________________________
88      % --- Executes on button press in pushbutton2.
89      function pushbutton2_Callback(hObject, eventdata, handles)
90      % hObject    handle to pushbutton2 (see GCBO)
91      % eventdata  reserved - to be defined in a future version of MATLAB
92      % handles    structure with handles and user data (see GUIDATA)
93 -    tercero
94 -    close(gcbf)
95
96      %_____________Principal_____________________________________________
97      % --- Executes on button press in pushbutton3.
98      function pushbutton3_Callback(hObject, eventdata, handles)
99      % hObject    handle to pushbutton3 (see GCBO)
100     % eventdata  reserved - to be defined in a future version of MATLAB
101     % handles    structure with handles and user data (see GUIDATA)
102 -    PRIMERA
103 -    close(gcbf)
```

Figura 32: Código Fuente "parametros_motor.m" (PARTE 1).

```
174     %'__________LISTO_____________________________________________________
175     % --- Executes on button press in pushbutton4.
176     function pushbutton4_Callback(hObject, eventdata, handles)
177     % hObject    handle to pushbutton4 (see GCBO)
178     % eventdata  reserved - to be defined in a future version of MATLAB
179     % handles    structure with handles and user data (see GUIDATA)
180 -   global Ra La N Ka Kb Jl  Jm  Bm  Bo
181
182 -   Ra=str2double(get(handles.edit6,'String'));
183 -   La=str2double('0');
184 -   N=str2double(get(handles.edit2,'String'));
185 -   Ka=str2double(get(handles.edit1,'String'));
186 -   Kb=str2double(get(handles.edit5,'String'));
187 -   Jm=str2double(get(handles.edit3,'String'));
188 -   Bm=str2double(get(handles.edit7,'String'));
189 -   Bo=str2double('0');
190
191
192 -   assignin('base','Ra',Ra);
193 -   assignin('base','La',La);
194 -   assignin('base','N',N);
195 -   assignin('base','Ka',Ka);
196 -   assignin('base','Kb',Kb);
197 -   assignin('base','Jm',Jm);
198 -   assignin('base','Bm',Bm);
199 -   assignin('base','Bo',Bo);
200
201
202 -   set(handles.text9,'String',evalin('base', 'Ra'));
203 -   set(handles.text8,'String',evalin('base', 'La'));
204 -   set(handles.text12,'String',evalin('base', 'N'));
205 -   set(handles.text16,'String',evalin('base', 'Ka'));
206 -   set(handles.text20,'String',evalin('base', 'Kb'));
207 -   set(handles.text13,'String',evalin('base', 'Jm'));
208 -   set(handles.text10,'String',evalin('base', 'Bm'));
209 -   set(handles.text11,'String',evalin('base', 'Bo'));
210
```

Figura 33: Código Fuente “parametros_motor.m” (PARTE 2).

```
257      %_____Predeterminados_____
258      % --- Executes on button press in pushbutton5.
259      function pushbutton5_Callback(hObject, eventdata, handles)
260      % hObject    handle to pushbutton5 (see GCBO)
261      % eventdata  reserved - to be defined in a future version of MATLAB
262      % handles    structure with handles and user data (see GUIDATA)
263 -    global  Ra La N Ka Kb J1 Jm  Bm Bo
264
265 -    Ra=str2double('7.6');
266 -    La=str2double('0');
267 -    N=str2double('128');
268 -    Ka=str2double('0.00211');
269 -    Kb=str2double('0.00309');
270 -    J1=str2double('0.0000030567');
271 -    Jm=str2double('0.000000087');
272 -    Bm=str2double('0.000000378');
273 -    Bo=str2double('0');
274
275 -    assignin('base','Ra',Ra);
276 -    assignin('base','La',La);
277 -    assignin('base','N',N);
278 -    assignin('base','Ka',Ka);
279 -    assignin('base','Kb',Kb);
280 -    assignin('base','J1',J1);
281 -    assignin('base','Jm',Jm);
282 -    assignin('base','Bm',Bm);
283 -    assignin('base','Bo',Bo);
284
285
286 -    set(handles.text9,'String', evalin('base', 'Ra'));
287 -    set(handles.text8,'String', evalin('base', 'La'));
288 -    set(handles.text12,'String',evalin('base', 'N'));
289 -    set(handles.text16,'String',evalin('base', 'Ka'));
290 -    set(handles.text20,'String',evalin('base', 'Kb'));
291 -    set(handles.text13,'String',evalin('base', 'Jm'));
292 -    set(handles.text10,'String',evalin('base', 'Bm'));
293 -    set(handles.text11,'String',evalin('base', 'Bo'));
294
```

```
317
318      %_____ATRAS_____
319      % --- Executes on button press in pushbutton6.
320      function pushbutton6_Callback(hObject, eventdata, handles)
321      % hObject    handle to pushbutton6 (see GCBO)
322      % eventdata  reserved - to be defined in a future version of MATLAB
323      % handles    structure with handles and user data (see GUIDATA)
324 -    Introduccion
325 -    close(gcbf)
326
```

Figura 34: Código Fuente “parametros_motor.m” (PARTE 3).

ANEXO F

Código fuente del archivo "tercero.m"

Desde la Figura 35 hasta la Figura 40 se presenta el código fuente del archivo "tercero.m" con sus comentarios correspondientes.

```
46      %____Pantalla incial ante de ser ejecutado el programa_____________________
47      % --- Executes just before tercero is made visible.
48      function tercero_OpeningFcn(hObject, eventdata, handles, varargin)
49      % This function has no output args, see OutputFcn.
50      % hObject    handle to figure
51      % eventdata  reserved - to be defined in a future version of MATLAB
52      % handles    structure with handles and user data (see GUIDATA)
53      % varargin   command line arguments to tercero (see VARARGIN)
54 -    global e
55 -    e=0;
56 -    w=imread('brazo1.jpg');
57 -    image(w)
58 -    axis off
59
60 -    set(handles.text4,'String','a1 =');
61 -    set(handles.text5,'String','b1 =');
62 -    set(handles.text6,'String','c1 =');
63
64 -    set(handles.text7,'String',evalin('base', 'a1'));
65 -    set(handles.text8,'String',evalin('base', 'b1'));
66 -    set(handles.text9,'String',evalin('base', 'c1'));
67 -    set(handles.text12,'String',evalin('base', 'densidad'));
68
69 -    set(handles.text15,'String','e');
70 -    set(handles.text17,'String','e');
71 -    set(handles.text16,'String',evalin('base', 'e'));
72
73
74      % Choose default command line output for tercero
75 -    handles.output = hObject;
76
77      % Update handles structure
78 -    guidata(hObject, handles);
79
```

Figura 35: Código Fuente "tercero.m" (PARTE 1).

```
77      %_____________Salir_____________
78      % --- Executes on button press in pushbutton1.
79      function pushbutton1_Callback(hObject, eventdata, handles)
80      % hObject    handle to pushbutton1 (see GCBO)
81      % eventdata  reserved - to be defined in a future version of MATLAB
82      % handles    structure with handles and user data (see GUIDATA)
83 -    clear all;
84 -    clc;
85 -    close(gcbf)
86
104     %_____________PRINCIPAL_____________
105     % --- Executes on button press in pushbutton2.
106     function pushbutton2_Callback(hObject, eventdata, handles)
107     % hObject    handle to pushbutton2 (see GCBO)
108     % eventdata  reserved - to be defined in a future version of MATLAB
109     % handles    structure with handles and user data (see GUIDATA)
110 -    PRIMERA
111 -    close(gcbf)
112
113     %_____________ATRAS_____________
114     % --- Executes on button press in pushbutton3.
115     function pushbutton3_Callback(hObject, eventdata, handles)
116     % hObject    handle to pushbutton3 (see GCBO)
117     % eventdata  reserved - to be defined in a future version of MATLAB
118     % handles    structure with handles and user data (see GUIDATA)
119 -    Parametros_motor
120 -    close(gcbf)
121
122     %_____ELEGIR_EL_BRAZO_1_2_o_3_____
123     % --- Executes on selection change in brazos.
124     function brazos_Callback(hObject, eventdata, handles)
125 -    global brazo
126 -    brazo=get(hObject,'Value');
127
128 -    switch brazo
129
130 -        case 1
131 -            opcio1=imread('brazo1.jpg');
132 -            image(opcio1)
133 -            axis off
134
135 -            set(handles.text4,'String','a1 =');
136 -            set(handles.text5,'String','b1 =');
137 -            set(handles.text6,'String','c1 =');
138 -            set(handles.text15,'String','e=');
139 -            set(handles.text17,'String','e=');
140
141 -            set(handles.text7,'String',evalin('base', 'a1'));
142 -            set(handles.text8,'String',evalin('base', 'b1'));
143 -            set(handles.text9,'String',evalin('base', 'c1'));
144 -            set(handles.text16,'String',evalin('base', 'e'));
```

Figura 36: Código Fuente “tercero.m” (PARTE 2).

```
148 -           opcio2=imread('brazo2.jpg');
149 -           image(opcio2)
150 -           axis off
151
152 -           set(handles.text4,'String','a2 =');
153 -           set(handles.text5,'String','b2 =');
154 -           set(handles.text6,'String','c2 =');
155 -           set(handles.text15,'String','e=');
156 -           set(handles.text17,'String','e=');
157
158 -           set(handles.text7,'String',evalin('base', 'a2'));
159 -           set(handles.text8,'String',evalin('base', 'b2'));
160 -           set(handles.text9,'String',evalin('base', 'c2'));
161 -           set(handles.text16,'String',evalin('base', 'e'));
162
163 -       case 3
164 -           opcio3=imread('brazo3.jpg');
165 -           image(opcio3)
166 -           axis off
167
168 -           set(handles.text4,'String','a3 =');
169 -           set(handles.text5,'String','b3 =');
170 -           set(handles.text6,'String','c3 =');
171 -           set(handles.text15,'String','e=');
172 -           set(handles.text17,'String','e=');
173
174 -           set(handles.text7,'String',evalin('base', 'a3'));
175 -           set(handles.text8,'String',evalin('base', 'b3'));
176 -           set(handles.text9,'String',evalin('base', 'c3'));
177 -           set(handles.text16,'String',evalin('base', 'e'));
178
179 -   end
263     %_____________________LISTO_____________________
264     % --- Executes on button press in listo.
265 -   function listo_Callback(hObject, eventdata, handles)
266 -   %bra=evalin('base','brazo');
267     % hObject    handle to listo (see GCBO)
268     % eventdata  reserved - to be defined in a future version of MATLAB
269     % handles    structure with handles and user data (see GUIDATA)
270 -   %bra=handles.brazo;
271
272 -   global densidad W H L R Msm asm bsm csm e
273
274 -   W=0.105;
275 -   H=0.07;
276 -   L=0.085;
277 -   R=0.0075;
278 -   Msm=0.043;
279 -   asm=0.0198;
280 -   bsm=0.0398;
281 -   csm=0.0363;
282
283 -   densidad=str2double(get(handles.edit5,'String'));
284 -   assignin('base','densidad',densidad);
285 -   set(handles.text12,'String',evalin('base', 'densidad'));
286
287 -   bra=get(handles.brazos,'Value');
288 -   switch bra
289 -       case 1
290 -           global a1 b1 c1
291
292 -           a1=str2double(get(handles.brazo1,'String'));
293 -           b1=str2double(get(handles.brazo2,'String'));
294 -           c1=str2double(get(handles.brazo3,'String'));
```

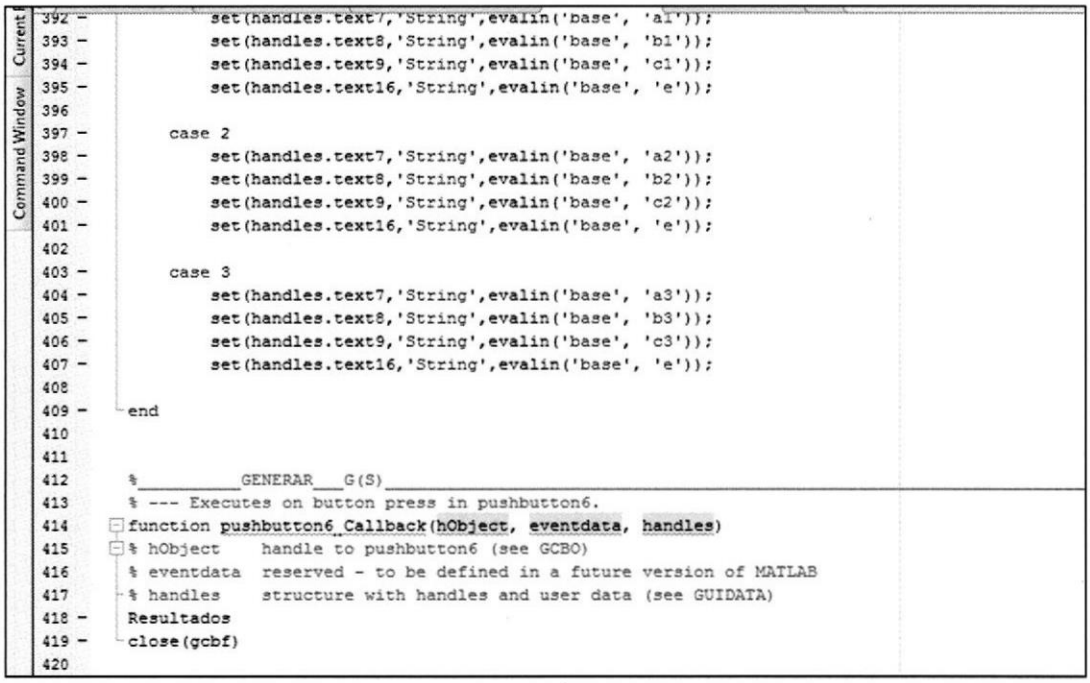
Figura 37: Código Fuente “tercero.m” (PARTE 3).

```
295 -           e=str2double(get(handles.edit6,'String'));
296
297 -           assignin('base','a1',a1);
298 -           assignin('base','b1',b1);
299 -           assignin('base','c1',c1);
300 -           assignin('base','e',e);
301
302 -           set(handles.text7,'String',evalin('base', 'a1'));
303 -           set(handles.text8,'String',evalin('base', 'b1'));
304 -           set(handles.text9,'String',evalin('base', 'c1'));
305 -           set(handles.text16,'String',evalin('base', 'e'));
306
307
308 -       case 2
309 -           global a2 b2 c2
310
311 -           a2=str2double(get(handles.brazo1,'String'));
312 -           b2=str2double(get(handles.brazo2,'String'));
313 -           c2=str2double(get(handles.brazo3,'String'));
314 -           e=str2double(get(handles.edit6,'String'));
315
316 -           assignin('base','a2',a2);
317 -           assignin('base','b2',b2);
318 -           assignin('base','c2',c2);
319 -           assignin('base','e',e);
320
321 -           set(handles.text7,'String',evalin('base', 'a2'));
322 -           set(handles.text8,'String',evalin('base', 'b2'));
323 -           set(handles.text9,'String',evalin('base', 'c2'));
324 -           set(handles.text16,'String',evalin('base', 'e'));
325
326 -       case 3
327 -           global a3 b3 c3
328
329 -           a3=str2double(get(handles.brazo1,'String'));
330 -           b3=str2double(get(handles.brazo2,'String'));
331 -           c3=str2double(get(handles.brazo3,'String'));
332 -           e=str2double(get(handles.edit6,'String'));
333
334 -           assignin('base','a3',a3);
335 -           assignin('base','b3',b3);
336 -           assignin('base','c3',c3);
337 -           assignin('base','e',e);
338
339 -           set(handles.text7,'String',evalin('base', 'a3'));
340 -           set(handles.text8,'String',evalin('base', 'b3'));
341 -           set(handles.text9,'String',evalin('base', 'c3'));
342 -           set(handles.text16,'String',evalin('base', 'e'));
343
344 -   end
```

Figura 38: Código Fuente "tercero.m" (PARTE 4).

```
346
347       %____________________PREDETERMINADOS_____________________________________
348       % --- Executes on button press in pushbutton5.
349       function pushbutton5_Callback(hObject, eventdata, handles)
350       % hObject    handle to pushbutton5 (see GCBO)
351       % eventdata  reserved - to be defined in a future version of MATLAB
352       % handles    structure with handles and user data (see GUIDATA)
353 -     global a1 b1 c1 a2 b2 c2 a3 b3 c3 e densidad  W L H R Msm asm bsm csm
354 -     a1=0.15;
355 -     b1=0.087;
356 -     c1=0.028;
357 -     a2=0.189;
358 -     b2=0.116;
359 -     c2=0.033;
360 -     a3=0.16;
361 -     b3=0.10;
362 -     c3=0.034;
363 -     e=0.0033;
364 -     densidad=150;
365 -     W=0.105;
366 -     H=0.07;
367 -     L=0.085;
368 -     R=0.0075;
369 -     Msm=0.043;
370 -     asm=0.0198;
371 -     bsm=0.0398;
372 -     csm=0.0369;
373
374
375 -     assignin('base','a1',a1);
376 -     assignin('base','b1',b1);
377 -     assignin('base','c1',c1);
378 -     assignin('base','a2',a2);
379 -     assignin('base','b2',b2);
380 -     assignin('base','c2',c2);
381 -     assignin('base','a3',a3);
382 -     assignin('base','b3',b3);
383 -     assignin('base','c3',c3);
384 -     assignin('base','densidad',densidad);
385 -     assignin('base','e',e);
386 -     set(handles.text12,'String',evalin('base', 'densidad'));
387
388 -     prede=get(handles.brazos,'Value');
389 -     switch prede
390
391 -         case 1
392 -             set(handles.text7,'String',evalin('base', 'a1'));
393 -             set(handles.text8,'String',evalin('base', 'b1'));
```

Figura 39: Código Fuente “tercero.m” (PARTE 5).



```
392 -            set(handles.text7,'String',evalin('base', 'a1'));
393 -            set(handles.text8,'String',evalin('base', 'b1'));
394 -            set(handles.text9,'String',evalin('base', 'c1'));
395 -            set(handles.text16,'String',evalin('base', 'e'));
396
397 -        case 2
398 -            set(handles.text7,'String',evalin('base', 'a2'));
399 -            set(handles.text8,'String',evalin('base', 'b2'));
400 -            set(handles.text9,'String',evalin('base', 'c2'));
401 -            set(handles.text16,'String',evalin('base', 'e'));
402
403 -        case 3
404 -            set(handles.text7,'String',evalin('base', 'a3'));
405 -            set(handles.text8,'String',evalin('base', 'b3'));
406 -            set(handles.text9,'String',evalin('base', 'c3'));
407 -            set(handles.text16,'String',evalin('base', 'e'));
408
409 -    end
410
411
412      %_________GENERAR___G(S)_____________________________________________
413      % --- Executes on button press in pushbutton6.
414    function pushbutton6_Callback(hObject, eventdata, handles)
415    % hObject    handle to pushbutton6 (see GCBO)
416      % eventdata  reserved - to be defined in a future version of MATLAB
417      % handles    structure with handles and user data (see GUIDATA)
418 -    Resultados
419 -    close(gcbf)
420
```

Figura 40: Código Fuente “tercero.m” (PARTE 6).

ANEXO G

Código fuente del archivo "resultado.m"

Desde la Figura 41 hasta la Figura 49 se presenta el código fuente del archivo "resultado.m" con sus comentarios correspondientes.

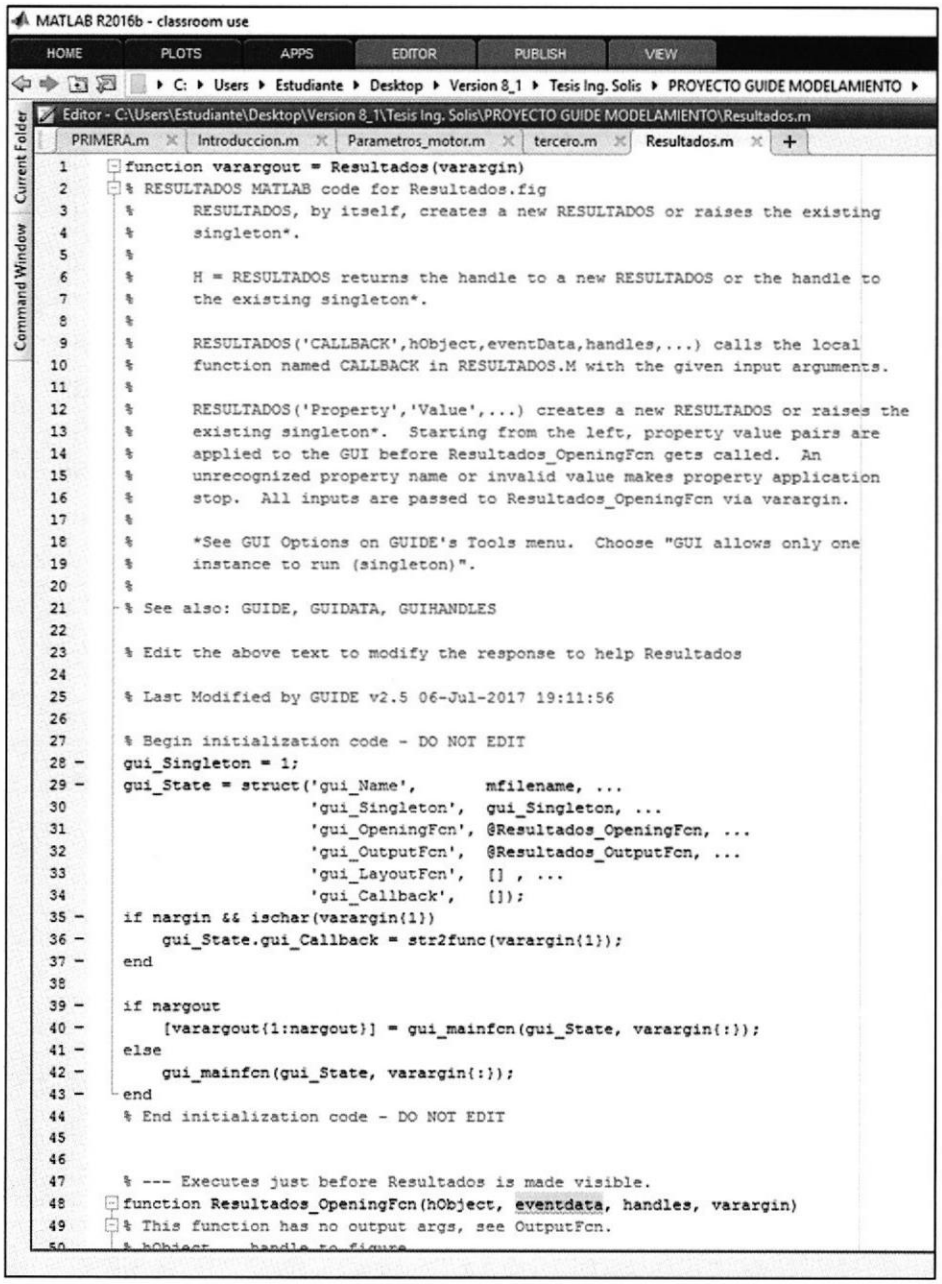


Figura 41: Código Fuente "resultado.m" (PARTE 1).

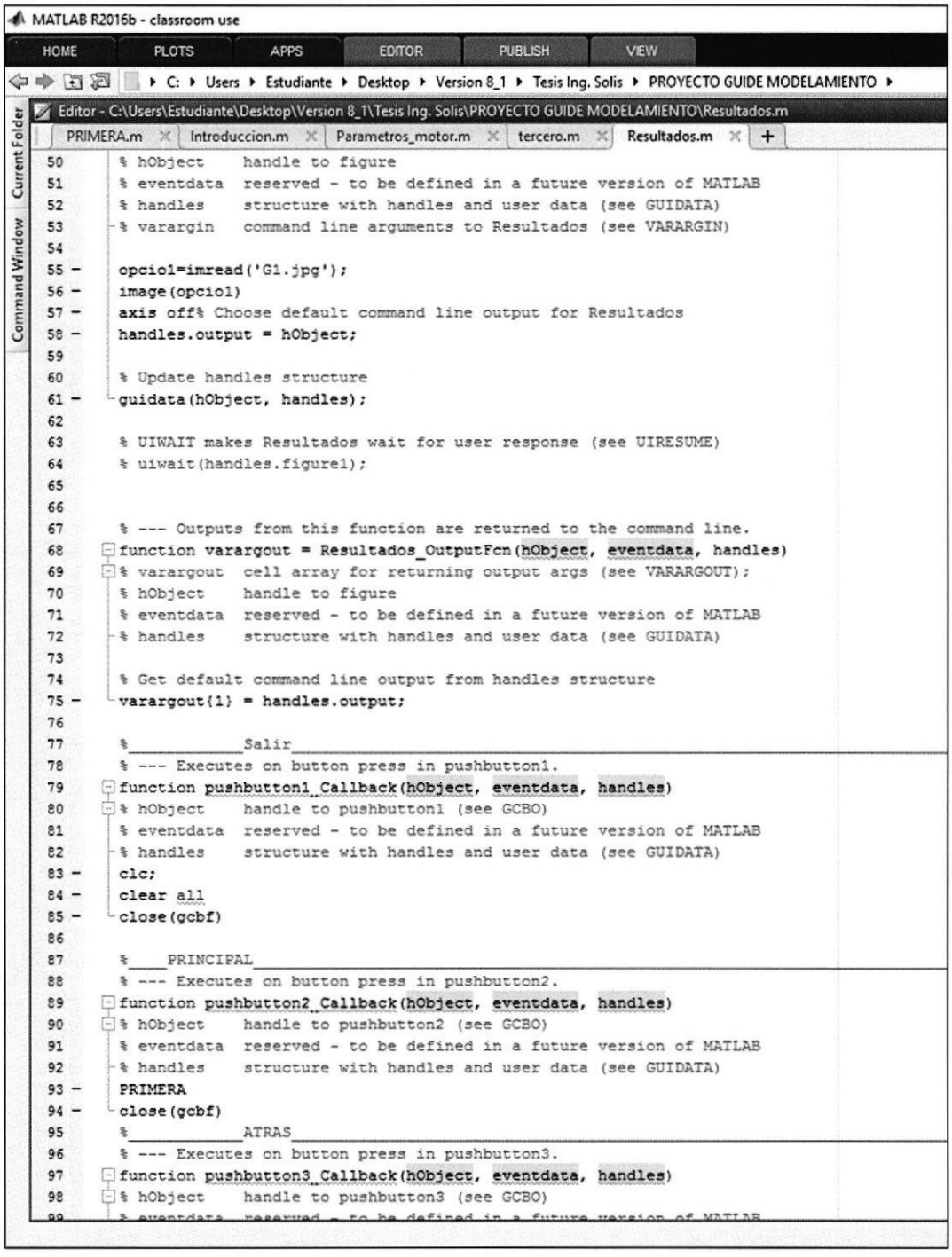


Figura 42: Código Fuente “resultado.m” (PARTE 2).

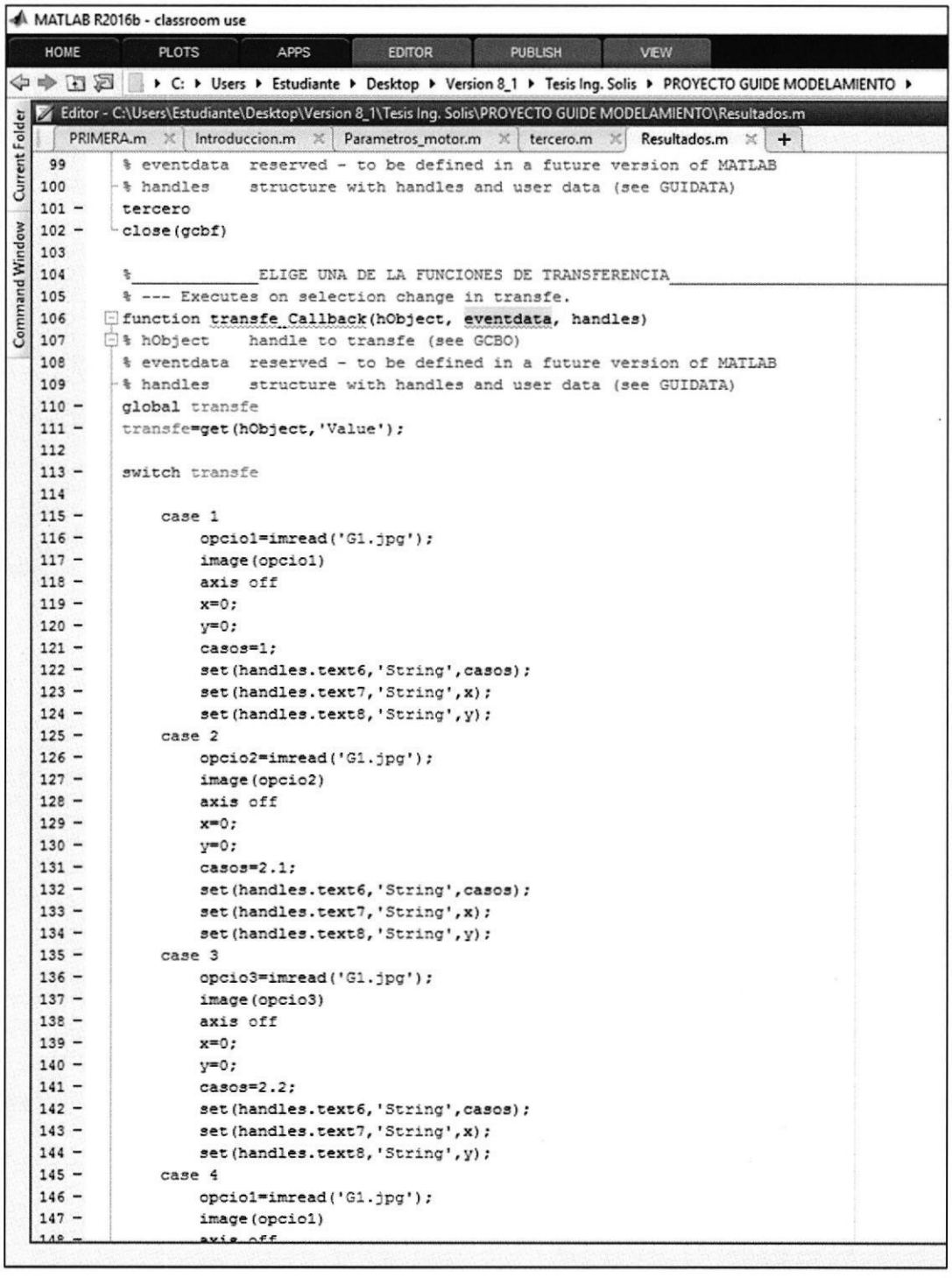


Figura 43: Código Fuente "resultado.m" (PARTE 3).

Figura 44: Código Fuente "resultado.m" (PARTE 4).

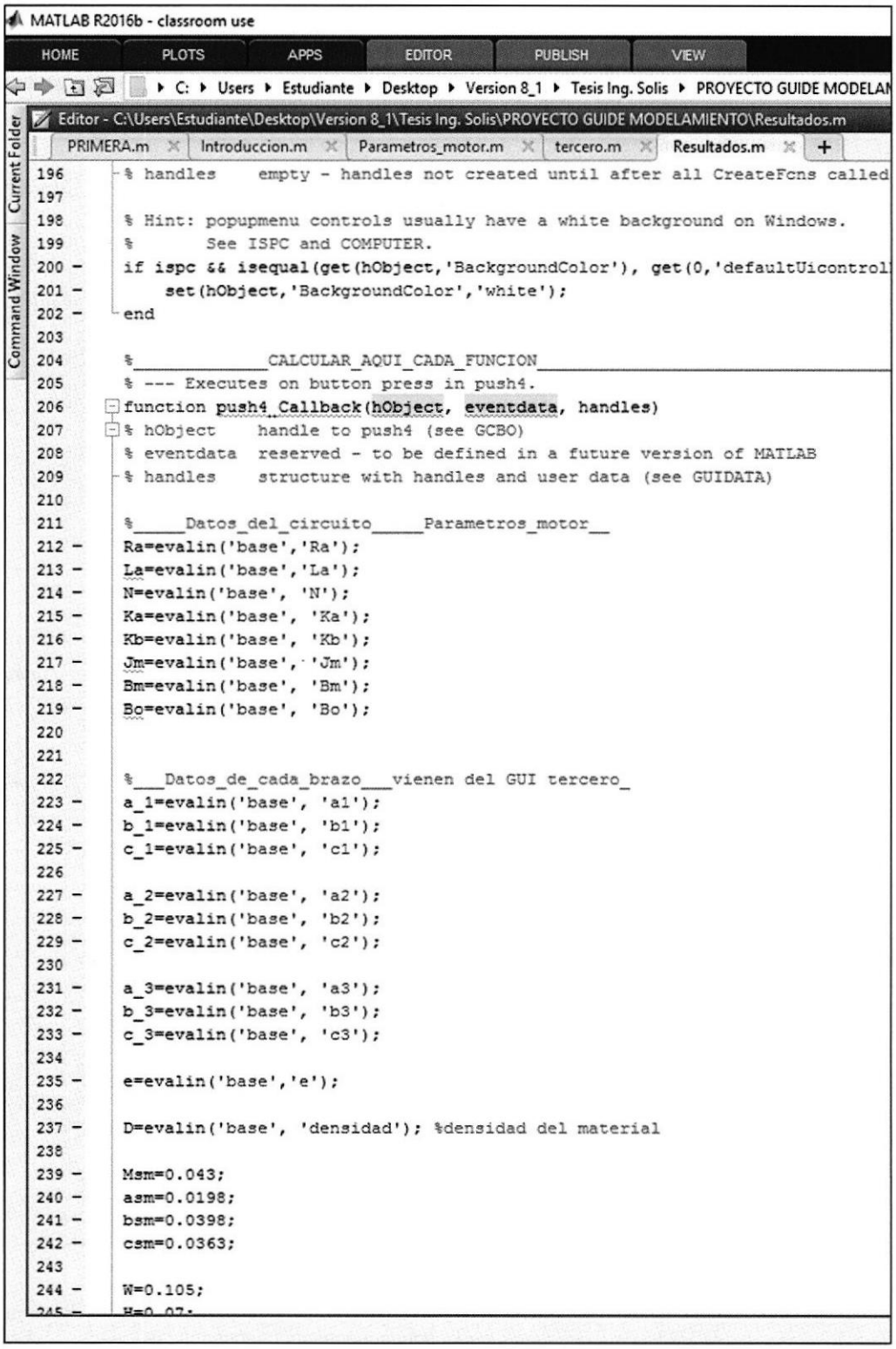


Figura 45: Código Fuente “resultado.m” (PARTE 5).

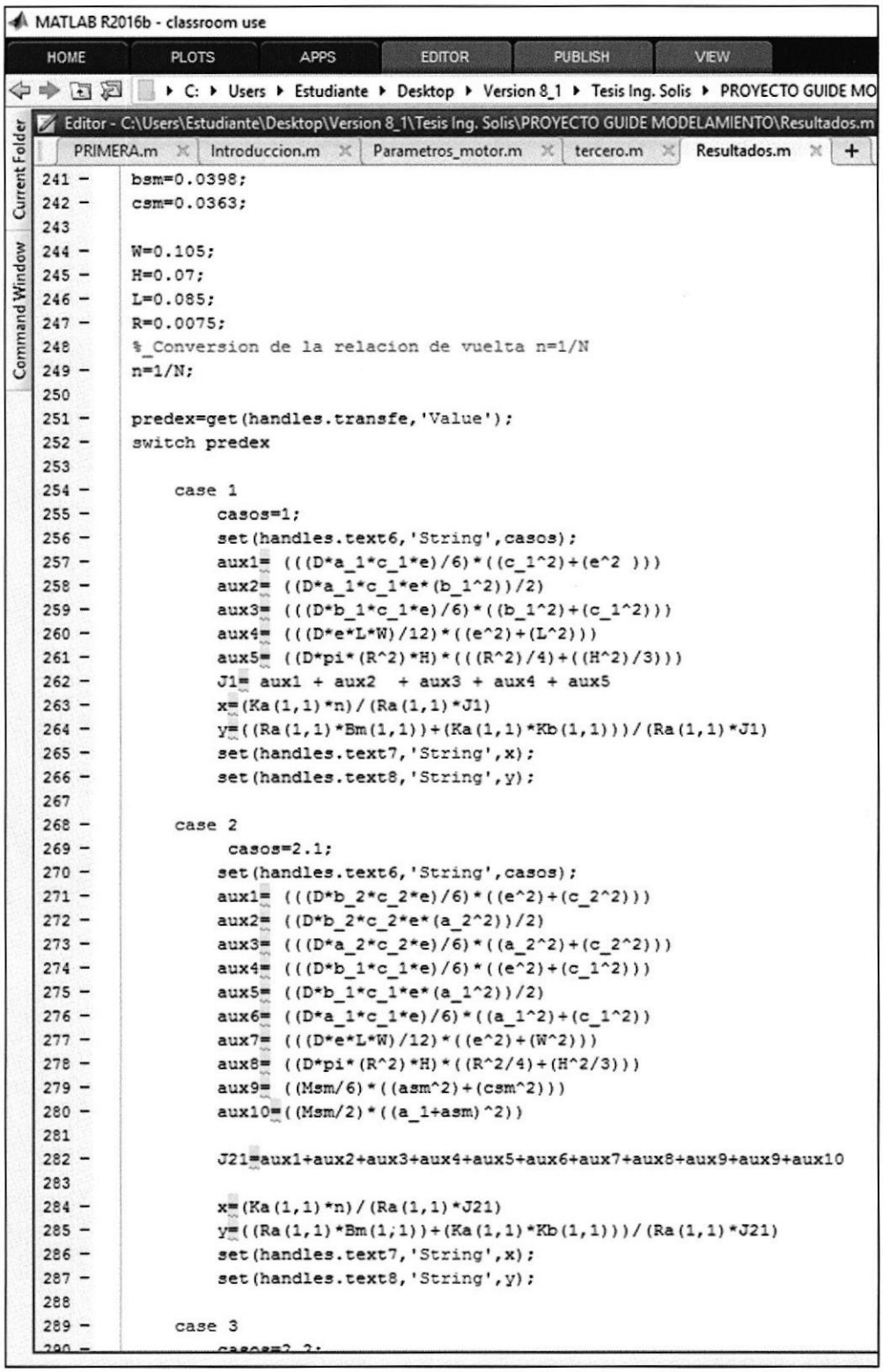


Figura 46: Código Fuente "resultado.m" (PARTE 6).

Figura 47: Código Fuente “resultado.m” (PARTE 7).

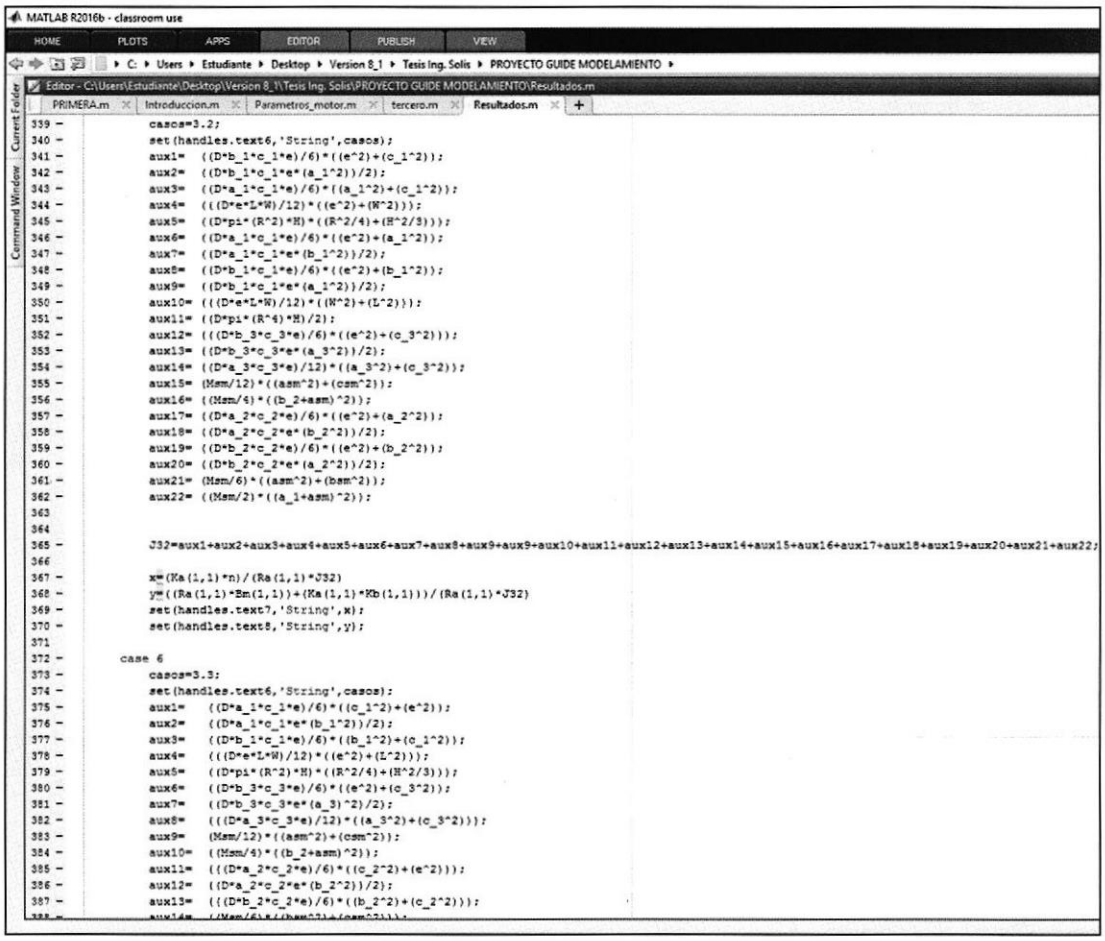


Figura 48: Código Fuente "resultado.m" (PARTE 8).

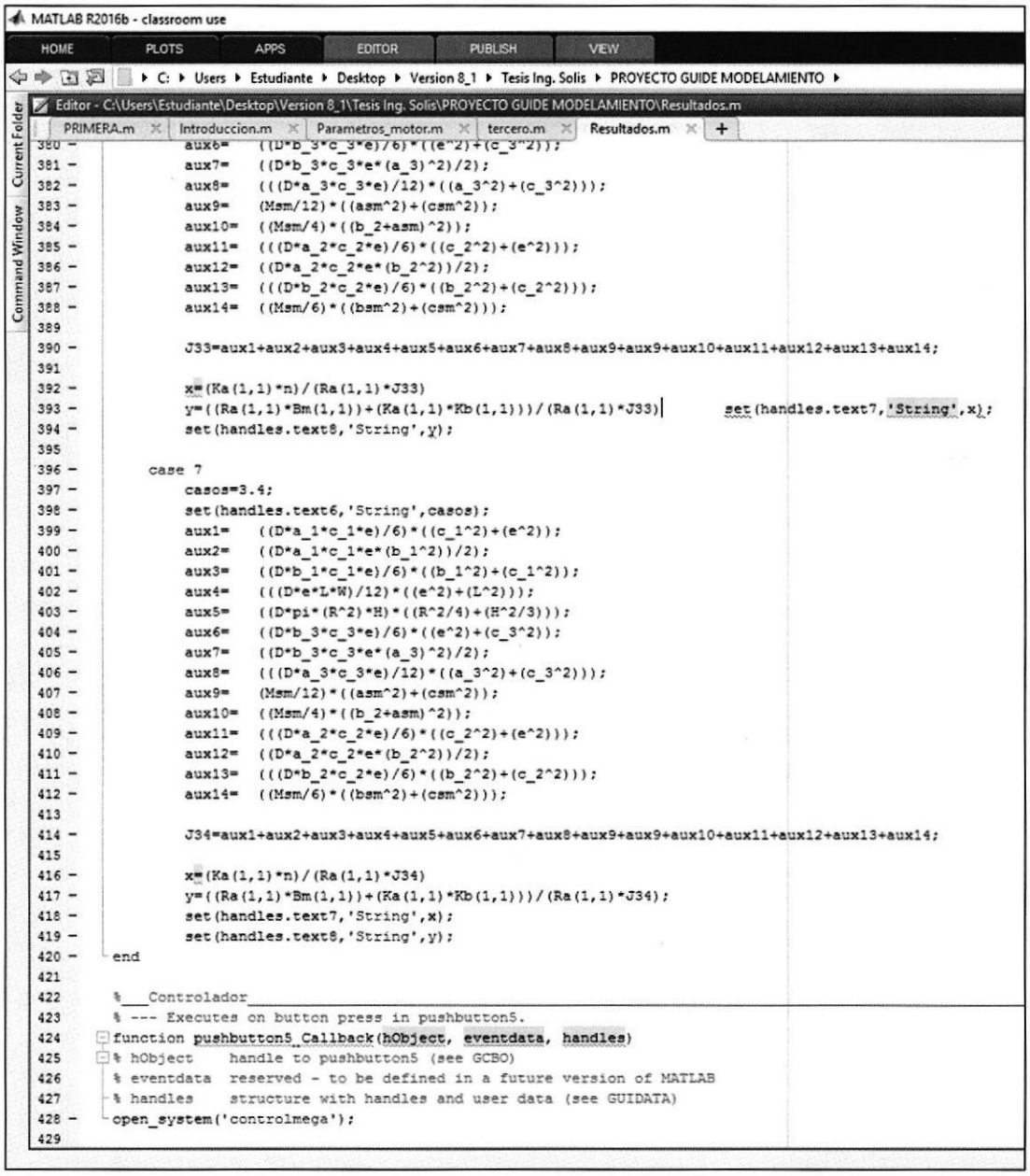


Figura 49: Código Fuente "resultado.m" (PARTE 9).