Universidad de Pamplona
Departamento de Mecánica, Mecatrónica e Industrial
Ingeniería Mecatrónica

Desarrollo de un péndulo de Furuta como herramienta didáctica para la aplicación de diferentes métodos de control regulatorio y control inteligente en la Universidad de Pamplona

Juan Manuel Calderón Agudelo

Director trabajo de grado
Cristhian Iván Riaño Jaimes

Trabajo de Grado

Pregrado Ingeniería Mecatrónica

2021

Contenido

Lista de figuras

Abstract

During the reading of this degree work will go through different chapters which present specific topics for a particular area, but all with the same objective to show a system of inverted rotational pendulum or pendulum of Furuta with various control methods, but all with the same goal in common, balance the pendulum of the system in inverted equilibrium position.

Mathematical modeling methods for this system, materials for the construction of an inverted rotational pendulum and the application of a controller to it are also presented. The electronics accompanying this system and a 3D animation with what would be an imitation of the inverted rotational pendulum for display. At the end the graphs show the result that the controllers express in the simulations made in simulink.

It is important to mention the importance of control strategies within a system when testing new methods or techniques, the inverted rotational pendulum system is a module for academic purposes for study and analysis with complex dynamics to control, so it is used to analyze control systems.

Resumen

Durante la lectura de este trabajo de grado se pasaran por diferentes capítulos los cuales presentan temas en concreto para un área en particular, pero todas con un mismo objetivo mostrar un sistema de péndulo rotacional invertido o péndulo de Furuta con varios métodos de control, pero todos con la misma meta en común, balancear el péndulo del sistema en posición de equilibrio invertida.

También se presentan métodos de modelado matemático para este sistema, materiales para la construcción de un péndulo rotacional invertido y la aplicación de un controlador al mismo. La electrónica que acompaña a este sistema y una animación en 3D con lo que sería una imitación del péndulo rotacional invertido para su exhibición. Al final las gráficas muestran el resultado que expresan los controladores en las simulaciones realizadas en simulink.

Es importante mencionar la importancia que poseen las estrategias de control dentro de un sistema al momento de testear nuevos métodos o técnicas, el sistema del péndulo rotacional invertido se trata de un módulo con fines académicos para estudio y análisis con una dinámica compleja de controlar, por esto es utilizado para analizar sistemas de control.

1 Introducción

1.1 Planteamiento del Problema

Materias como teoría de control, control industrial y control inteligente hacen parte fundamental de la formación integral del alumnado de la carrera de ingeniería mecatrónica en la Universidad de Pamplona. A lo largo de la carrera, se vienen desarrollando una serie de actividades, a manera de dar al estudiante las herramientas adecuadas para desarrollar su conocimiento de manera óptima. Aunque es de conocer que no todos los estudiantes están en las condiciones socioeconómicas para desenvolverse plenamente en todas estas actividades, realizadas en algunas materias de la carrera, ya que se requiere de la adquisición de materiales y de componentes

electrónicos los cuales, dependiendo del nivel de complejidad de dicha actividad van a ser más costosos y dan como resultado un gran impacto en la economía de cada uno de los estudiantes, haciendo que estos tengan la polémica decisión entre desempeñarse adecuadamente en la materia o mantenerse económicamente a flote.

Con esta propuesta se buscó brindar a la academia una herramienta que permite profundizar tanto de manera teórica, como de manera práctica en las materias relacionadas con el control de procesos. Permitiendo estar en contacto con un sistema de control que es una herramienta muy completa a modo de práctica de laboratorio, que da al estudiante la posibilidad de poder ejecutar con esta herramienta sus diseños de controladores, tanto difusos, clásicos y RNA

1.2 Propósito del Estudio

La idea principal de este estudio fue realizar una herramienta y/o sistema didáctico con el fin de aplicar diferentes estrategias de control, cualesquiera que pudieran ser implementadas (control regulatorio y/o inteligente). Se utilizó como planta principal el péndulo rotacional invertido o también llamado péndulo de Furuta, sobre dicha planta o sistema se desarrollaron, implementaron y probaron algunas estrategias de control las cuales son descritas a lo largo de este documento. La idea principal fue desarrollar un sistema a ser controlado, partiendo de una idea, referencias, artículos científicos y libros sobre el tema, luego se establecieron unos bocetos para posteriormente pasar al diseño que para este caso se establece un diseño en 3D por medio del software Solidworks del año 2015.

1.3 Objetivos de la Tesis

1.4 Objetivo General

Desarrollar un péndulo rotacional invertido como herramienta didáctica para la aplicación de diferentes métodos de control regulatorio y control inteligente.

1.5 Objetivos Específicos

- Comprender y formular un modelo matemático que represente el comportamiento dinámico del sistema.
- Desarrollar diferentes técnicas de control que cubran áreas de control regulatorio y control inteligente.
- Desarrollar un prototipo virtual CAD del módulo que permita implementar y analizar físicamente cada una de las técnicas de control.
- Construir, ensamblar y poner a prueba el prototipo.
- Abalizar los resultados y consolidar material instructivo para la operación y manejo del módulo.

1.6 Estructura del Documento

En el **Capítulo 2 Marco Teórico y Estado del Arte** se aborda de manera monográfica algunos casos en los cuales se expone de forma breve la aplicación de distintos sistemas de control hacia un prototipo de péndulo rotacional invertido. Se exponen algunos métodos interesantes de control y se mencionan algunos métodos que han sido importantes a lo largo del desarrollo de los diferentes sistemas que han sido implementados en la academia.

En el **Capítulo 3 Modelado Matemático del Péndulo de Furuta** se establecen los parámetros y la matemática implícita en la dinámica del péndulo rotacional invertido así como las respectivas ecuaciones que la describen y sus respectivas explicaciones. Aquí se definen cada una de las ecuaciones o fórmulas matemáticas que describen la dinámica, comportamiento y reacciones del péndulo rotacional invertido ante ciertas entradas así como las diferentes ecuaciones utilizadas en momentos específicos.

En el **Capítulo 4 Diseño de los Controladores** se presentan los controladores implementados a lo largo del desarrollo del documento. El controlador Swing Up es una parte importante para este sistema de péndulo rotacional invertido ya que es el que lleva el péndulo a la posición de equilibrio deseada,tareas que controladores PID y fuzzy no serían capaces de obtener por sí solos. Se explica una forma de obtener un trabajo aceptable por parte de un controlador PID, para ello se explica como se trabaja con sistema de alta linealidad y el desempeño del controlador sobre esta planta. Se explica la forma de diseñar un controlador difuso de tipo Mamdani, como se compone internamente y la forma en la que se relacionan los valores de salida y entrada al sistema de control difuso. En la sección final se muestra una forma sencilla de trabajar con redes neuronales y como se puede replicar el comportamiento del controlador Swing Up para este sistema de péndulo rotacional invertido en específico.

En el **Capítulo 5 Simulación** En esta sección se presentan los controladores implementados en el modelo matemático simulado por medio de sus ecuaciones dinámicas en Simulink. También se muestra el proceso de representación de un modelo CAD para una mejor representación y herramienta visual para observar el comportamiento del sistema a lo largo del tiempo simulado.

En el **Capítulo 6 Desarrollo y Construcción** se muestra paso a paso la elaboración de cada una de las partes que componen al sistema de péndulo rotacional invertido. Es importante ver cada uno de los pequeños sistemas o componentes que hacen parte de este otro gran sistema pendular. Componentes electrónicos como mecánicos conviven y se relacionan para dar al espectador una experiencia de control desafiante y diferentes a las acostumbradas.

En el **Capítulo 7 Resultados Obtenidos** En esta sección las gráficas y resultados de las simulaciones de muestran para dar a entender lo sucedido a lo largo de los capítulos anteriores. Aquí se comparan las gráficas y se realizan lecturas de las mismas para entender qué fue de lo sucedido con respecto a los controladores implementados. También se describen a modo de manual cómo se implementan bloques con Scripts para el desarrollo o implementación de librerías de Arduino dentro de Simulink, también se describe cómo realizar la ejecución de una placa de Arduino Mega con un driver TTL-USB diferente al ATMega 16u2.

2 Marco Teórico y Estado del Arte

2.1 Introducción

En esta sección se empieza a entablar una conversación con el lector de acuerdo a las diferentes etapas por las cuales transcurren las investigaciones que suceden cuando se habla del péndulo rotacional invertido. Para poder entrar en contexto con respecto al tema se mencionan algunas definiciones que serán de gran importancia a lo largo del documento. Se explica de donde surge la idea de un péndulo rotacional invertido y como este se establece a sí mismo como un reto o como una planta que reta a ser controlada.

2.2 Estado del Arte

2.2.1 Péndulo de Furuta

El péndulo de furuta, o péndulo rotacional invertido, es un sistema experimental comúnmente utilizado con fines educativos en la teoría de control moderna. Debido a la gravedad y el acoplamiento de la fuerza de Coriolis y la fuerza centrípeta, es un ejemplo obvio de un sistema poco impulsado, subactuado y extremadamente no lineal. Avelar (2013)

Aquellos sistemas mecánicos que así mismos se describen como subactuados, infraaccionados son sistemas mecánicos con menos entrada de control que grados de libertad. Pertenecen a la clasificación de sistemas no lineales. Los sistemas no lineales no tienen un método universal para resolver todo tipo de sistemas no lineales. Por tanto, existen sistemas con insuficientes sistemas operativos que permiten desarrollar y probar distintas técnicas o vías de control para poder ejecutar, controlar y estabilizar un sistema no lineal en un punto de operación. Mamani Churayra (2012)

En la Figura 1, se llega a observar un esquema del péndulo de furuta, el cual está compuesto por un péndulo de libre movimiento el cual no está sujeto a ningún actuador, pero si debe estar sujeto a un sensor el cual realimentará un sistema de control en todo momento.

El estudio de este sistema experimental, fue presentado en 1991 por Katsuhisa Furuta en el instituto de Tecnología de Tokio, llamado como "Péndulo TITech".

El sistema de Péndulo Rotacional Invertido o Péndulo de Furuta que fuese desarrollado por el Dr. K. Furuta, " *es un sistema infraaccionado o subactuado con solamente dos grados de libertad, ambos de los cuales se pueden rotar, llamados brazo y péndulo. El movimiento del brazo (primer grado de libertad) se realiza en un plano horizontal girando alrededor de un eje perpendicular al plano, y el péndulo se ubica en un extremo del brazo, y su eje de rotación es colineal entre ejes. El brazo y su movimiento se realizan en un plano normal a la superficie del brazo* ".Valera et al. (2002)

Posteriormente se han presentado diversos métodos de control y prototipos derivados del mismo como el llamado "Péndulo de Furuta doble" o "Péndulo de Furuta triple". Un péndulo

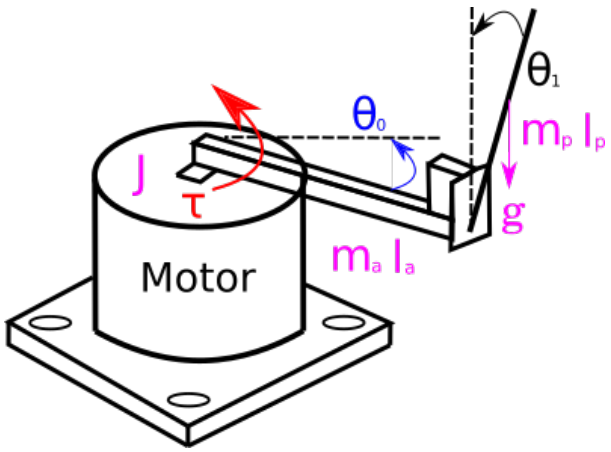

Figure 1: Representación gráfica del péndulo
Fuente: Adaptación Åström and Furuta (1996)

de furuta (de ahora en adelante se utilizarán las siglas PF para referirse al mismo), al ser un sistema subactuado se compone de unas sencillas partes, como son:

- El péndulo, que es en sí una barra de longitud considerable el cual, está conectado al brazo del PF.
- Un motor DC que, sujeto al brazo, permite la libre rotación del mismo y a su vez, produce un movimiento de balanceo en el péndulo con el fin de cambiar su estado natural de reposo a un estado controlado el cual consiste en el péndulo erguido en posición de equilibrio.

Como método de enseñanza en muchos centros de investigación, el uso de un péndulo invertido es más en el campo de la educación ya que representa en su mayor parte un problema de control clásico altamente no lineal. En el péndulo rotacional invertido se pueden diseñar y desarrollar problemas de control interesantes. Este se constituye de un sistema mecánico no lineal de acción insuficiente con una zona de alta inestabilidad en un punto de equilibrio deseado. Debido a las limitaciones del actuador, no se puede aplicar la tecnología de control desarrollada para un robot manipulador totalmente accionado.Gonzalez Fontanet et al. (2016)

El modelo de péndulo rotacional invertido no es posible linealizarse mediante realimentación de estado estática o dinámica debido a que se trata de un sistema altamente no lineal, lo que resulta difícil al momento de estabilizar cerca de la zona o punto de equilibrio deseado. Para lograr el cometido el péndulo rotacional invertido se han desarrollado, implementado y estudiado una variedad considerable de estratégias de control a conocer: Control basado en modelos, control neuronal, control difuso, algoritmos genéticos, control basado en energías, entre otros.Gonzalez Fontanet et al. (2016)

A continuación se mencionan algunas estrategias de control sobre el PF tomadas del artículo García et al. (2015). No se pretende nombrarlos todos ya que para eso está el artículo en sí. Se mencionarán algunos métodos destacando el más antiguo y uno de los más recientes.

2.2.2 Control Basado en Energías

En 1996, Åström, Karl Johan y Furuta, Katsuhisa desarrollaron un dispositivo de control que puede basarse en un simple péndulo giratorio invertido con impulso de energía. La definición de control está fundamentada en la cantidad de energía necesaria para llevar el péndulo del sistema desde una posición de reposo, es decir, con oscilaciones pocas o nulas, un péndulo naturalmente hacia abajo bajo efecto de la acción gravitacional a una posición erguida, posición vertical superior o levantado en zona de equilibrio. Åström and Furuta (1996)

Los autores Sukontanakarn and Parnichkun (2009) proponen un estudio en el que Control de PD basado en energía para lograr la elevación del péndulo, a través de un control LQR (lineal Regulador secundario) en tiempo real; completando el experimento y simulación.

2.2.3 Control por Realimentación de Estados

En 1991 se discute un nuevo dispositivo conocido como péndulo invertido rotacional o péndulo de Furuta,Furuta et al. (1991), en donde se propone un diseño para una estrategia de control para el balanceo *(también conocido como Swing-Up)* y posterior levantamiento por medio de una acción de control apagado - encendido *(ON-OFF)* mediante el uso de realimentación de estados usando información sobre el plano de fase *(conocido como control Bang-bang)*. La estabilización de manera robusta se logra a partir del uso de un controlador LQ *(Linear Quadratic)*; el cuál resulta ser un controlador lo bastante robusto inclusive para un péndulo doble, aunque sus resultados no fueron reportados.

Para el año 2014 un artículo presentado por Mandic et al. (2014). Como primera parte se realiza una caracterización la parte electromagnética para posterior obtener el modelo, también se estudia el modelo matemático y se considera la acción de control para el levantamiento y estabilización por medio de una realimentación de estados y un controlador PID. Considerando la mejora y estabilidad del péndulo a través de la retroalimentación parcial del estado de linealización y PID se realizan simulaciones y experimentos.

2.2.4 Control por Estructura Variable

Choi and Kim (2003) propusieron un diseño de control de modo deslizante con retroalimentación de salida, y un diseño que controla la estabilidad del péndulo de esta manera, que muestra un desempeño aceptable en la supresión de interferencias y alteraciones en el modelado; como el autor mencionó, una maniobra de control con observadores de modelado de errores, se compara el control LQ con la estabilidad, lo que demuestra que los dos primeros controles tienen mejor desempeño.

Se publicaron los resultados de un diseño de control adaptativo en el que además de las perturbaciones, también se consideran las incertidumbres a lo largo del tiempo. Lo primero que deben hacer estos autores es desacoplar la dinámica subactuada mediante la transformación de Olfati. Diseñe un controlador adaptativo y luego use el teorema de Lyapunov para realizar una

prueba de estabilidad de circuito cerrado. Se realizaron experimentos y simulaciones. Chen and Huang (2014).

2.2.5 Control Basado en Análisis Geométricos

En 2007 se restauró el diseño de control basado en análisis geométrico. El algoritmo de control del sistema completado por Zhang Juan, Chen Jie y Dong Lingxun se basa en poliedros y geometría poliédrica. Aquí se muestra la fuerte capacidad antiinterferente y de estabilidad. Juan et al. (2007)

Debido a estas limitaciones, en 2009 se propuso un curso de diseño de geometría basado en problemas. De acuerdo con la relación entre las coordenadas generalizadas y las limitaciones holo nómicas virtuales, se propone una curva homoclínica planificada previamente. Tiene un modelo de fricción de par para garantizar la estabilidad exponencial local. Dado que la fricción del par de torsión no se compensó adecuadamente, se encontró una diferencia entre el experimento y el movimiento requerido.La Hera et al. (2009)

2.2.6 Control Difuso

El control difuso se basa en la lógica difusa y puede describirse como un sistema de interpretación en el que el conjunto de límites entre elementos no está claramente definido (difuso). Estos sistemas difusos se basan en reglas descriptivas con operadores lógicos tradicionales (como SI-ENTONCES), que es un verdadero conocimiento heurístico que puede generar el desempeño requerido Yeh and Li (2004).

En 1996, se informó sobre el trabajo de Wang H.O. Se propone un método para estabilizar un péndulo invertido. Aquí se utiliza el modelo Takagi-Sugeno para mostrar el modelo no lineal de la fábrica, y se diseña un controlador difuso basado en este modelo basado en el concepto de compensación distribuida en paralelo, es decir, se llevan a cabo experimentos. Wang et al. (1996).

Se propone un método de control inteligente para el péndulo invertido. Este tipo de controlador difuso es un control inteligente basado en un modelo de lógica difusa, que no requiere modelos matemáticos precisos ni procesos de cálculo complejos, y puede manejar sistemas no lineales sin linealización. Los experimentos y simulaciones se llevan a cabo en MatLab-Simulink. Jain et al. (2013)

2.2.7 Control Inteligente

Este artículo presenta un método de modelado matemático de un sistema llamado péndulo de Furuta utilizando una función de energía, que toma en cuenta la dinámica no lineal del sistema físico y considera el acoplamiento existente entre equipos eléctricos y mecánicos. De manera global y establezca un área de operación (área "Swing Up"), y finalmente utilice una red

neuronal artificial (RNA) para simular el comportamiento de la función de energía. Escobar-Dávila et al. (2013)

2.2.8 Modelos de Péndulo de Furuta

La estructura básica por la cual se compone un péndulo invertido rotacional se puede describir como un sistema el cual sus grados de libertad se limitan a dos. Uno de estos se compone por un actuador o motor, normalmente de tipo DC y un pivote que compone al péndulo del sistema.

A continuación se muestran algunos modelos de péndulos rotacionales invertidos, en los cuales se basa el construido en este documento y que son los más representativos en su tipo.

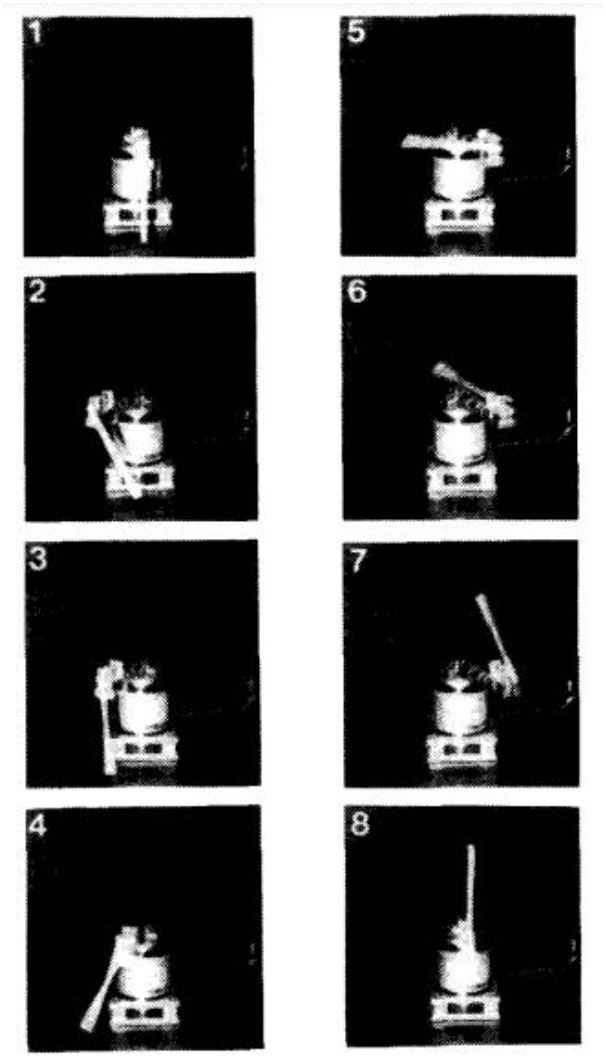

Figure 2: Movimiento Swing Up del péndulo
Fuente: Tomado de Furuta et al. (1991)

En 1991 Katsuhisa Furuta y Masaki Yamakita desarrollan un péndulo de furuta mostrado

en la Figura 2. A pesar de que no se logra visualizar muy bien el dispositivo, es bastante clara la intención detrás de este péndulo invertido rotacional. Además será de inspiración a nuevos diseños y prototipos del mismo.

Ahora damos un gran salto hasta el año 2002 en donde Valera et al. (2002), desarrolla un prototipo de péndulo invertido rotacional el cual se muestra en la Figura 3. El cual difiere un poco en su diseño en comparación al péndulo de Furuta et al. (1991)

Figure 3: Modelo de péndulo de furuta
Fuente: Tomado de Valera et al. (2002)

Una empresa llamada Quanser se dedica a fabricar plataformas de hardware de escritorio optimizadas para la academia, la investigación y la ingeniería en la educación. Entre estas plataformas se encuentran dos diseños se péndulo invertido rotacional los cuales se pueden apreciar en la Figura 4. Sus diseños son bastante prolijos, aunque estan compuestos por diferentes piezas las cuales pueden agregar más fricción entre sus juntas.

Figure 4: Péndulo rotacional invertido
Fuente: Tomado de Quanser (2021)

2.3 Marco Teórico

2.3.1 Sistemas de Control

Tanto en la ingeniería como en la vida, se necesitan establecer ciertas estrategias para el control de actividades o procesos en los cuales nos vemos involucrado directa o indirectamente. Los sistemas de control estudiados a lo largo de los años, se pueden clasificar en dos grandes tipos, de lazo abierto o de lazo cerrado. Estos últimos son lo que más llaman la atención debido a que utilizan sensores que permiten realizar una lectura de la salida del sistema permitiendo saber que pasa con la salida del sistema en tiempo real y por medio de esta, se realimenta con la entrada obteniendo un error el cual nos permite mediante un controlador poder corregirlo.

2.3.2 Control PID

El control proporcional integral derivativo, conocido comúnmente como control PID, es un controlador clásico de tipo regulatorio, que posee la característica de ser capaz de corregir las acciones con el propósito de aminorar las diferencias entre la referencia solicitada o a veces llamada SETPOINT y la salida del sistema, esto se conoce como error. Este tipo de control se ejerce en varios niveles como son:

- Proporcional, fijará el error en el instante de la acción.
- Integral, se usa la integral para corregir el error y reducirlo a cero (error estacionario).
- Derivativo, determinará la acción del tiempo, en que dicho el error se acciona.

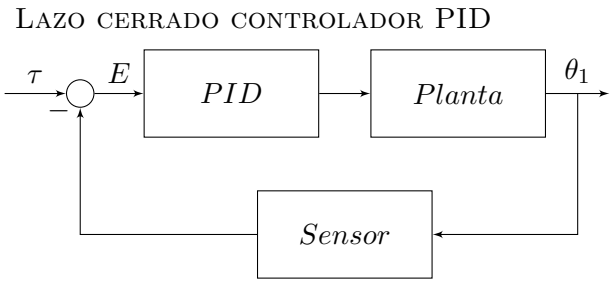

Figure 5: Representación diagrama de bloques controlador pid lazo cerrado
Fuente: Autoría propia

En el diagrama de bloques de la Figura 5 se observa el esquema general de manera muy simple, en la cual se implementa un controlador PID.

Si se puede determinar el modelo matemático del sistema, se pueden utilizar varios métodos de diseño para establecer las medidas requeridas del controlador, que realiza información detallada sobre los estados instantáneos y fijos del lazo de control cerrado. Sin embargo,se hace complicado adquirir un modelo matemático del sistema, no es posible obtener el método de diseño del controlador. En este caso, se elegirán métodos empíricos para ajustarlo. Ogata (2010)

La utilidad de los controles PID es que son casi universalmente aplicables a la mayoría de los sistemas de control. En particular, el control PID es más útil cuando se desconoce el modelo matemático del sistema o la planta, por lo tanto, no se puede utilizar el método de diseño analítico. En el campo de los sistemas de control de procesos, es un hecho bien conocido que se ha demostrado que los esquemas de control PID básicos y mejorados proporcionan un control satisfactorio,también es posible que no se logre proporcionar un control óptimo en muchas situaciones en específico.Ogata (2010)

El diagrama de bloques de la Figura 6 muestra de una manera más desglosada como está compuesto un controlador PID internamente. Posee una serie de ganancias, a saber proporcional K, integral T_i derivativa T_d estas ganancias son las que se deben ajustar al momento de sintonizar un controlador PID. Ya que el controlador PID es la suma de sus 3 partes proporcional, integral y derivativa, al eliminar la parte integral se optiene un controlador PD, o si se sustituyera la parte derivativa se obtiene un controlador PI, o también se dan los casos en donde no es necesario de un controlador PID completo, solo es necesaria la parte proporcional, se estaría hablando de un controlador P.

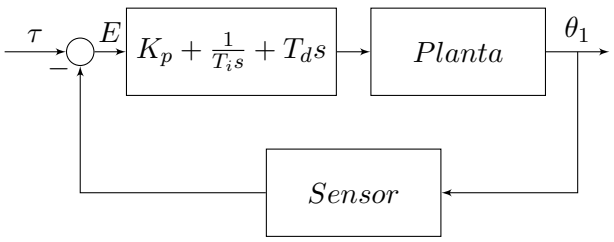

Figure 6: Diagrama control PID

2.3.3 Control Difuso o Fuzzy

El control difuso es sacado de la lógica difusa, cualquier libro sobre el tema lo explicaría de una mejor forma, por ello solo nos enfocamos en hablar sobre el control difuso como tal, si se hablan de tipos de control existen dos grandes grupos, el control convencional o también llamado clásico y el control moderno, en donde el primero se encuentran el controlador PID y compensadores de atraso o adelanto. El control difuso entra dentro de el control moderno ya que tiene diferentes características que lo destacan y hacen más llamativo al momento de aplicarlo sobre un sistema.

El control difuso viene de la implementación de la lógica difusa en un controlador, esto permite crear lo que en lógica difusa se conoce como reglas lingüísticas para una tarea o sistema en particular. Estas reglas lingüísticas emulan a una función, imagine una caja en donde llegan los valores de entrada y estos a su vez son tratados por estas reglas y enviandolas por la salida de dicha caja. Cruz (2011)

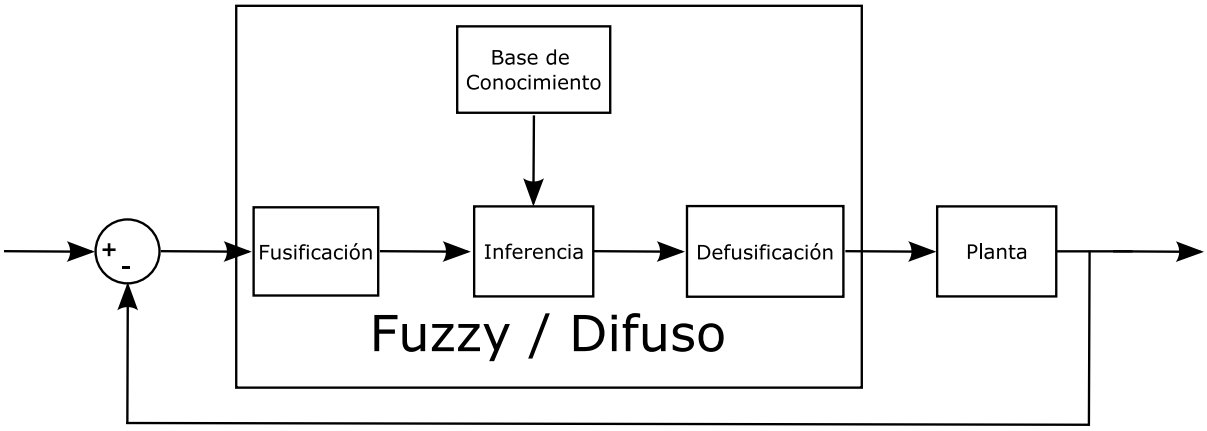

Figure 7: Lazo cerrado de control difuso/fuzzy
Fuente: Autoría propia

Según Cruz (2011), un controlador difuso se puede componer en cuatro piezas fundamentales: fusificación, base de conocimientos, inferencia y defusificación, según se detalla en el lazo de control cerrado de la Figura 7.

Para Kassir (2015), los controladores difusos poseen una serie de partes las cuales se pueden

definir en Base de conocimiento,fusificación, inferencia, defusificación.

La base de conocimiento está compuesta por una base de datos la cual es realizada a partir de la experiencia ya sea del operador o usuario que se encarga de darle sentido a las respuestas del sistema a partir de determinados estímulos. Se puede describir como un modelo Sí-Entonces.

Fusificación, fusificador, difusor son sinónimo de pasar de un estado no difuso a un estado difuso, el estado no difuso viene a ser los datos que entran al controlador difuso a ser tratados, el difusor recibe valores numéricos y los asocia a los conjuntos difusos del sistema previamente definidos.

La inferencia dentro del controlador difuso se encarga de asociar las reglas con los datos decibidos del fusificador.

Al igual que la parte fusificada, el defusificador hace todo lo contrario, pasando de los valores del controlador a valores numéricos entendibles para los sistemas fuera del controlador.

Al momento de realizar una defusificación se emplean métodos que se ajustan a las necesidades de cada aplicación. Una de las más utilizadas es la del centroide. Éste método del centroide consta en establecer una función que reuna las funciones de pertenencia que fueron tocadas dentro del proceso de inferencia para posteriormente sacar un promedio de sus valores, obteniendo lo que se conoce su valor de centroide.

Si se observa y se comparan la Figura 7, con un sistema de control de lazo cerrado con controlador PID, no se diferencian mucho en su distribución de bloques. Lo que interesa de estos dos sistema es como estan compuestos internamente y cómo se desempeñan en su ejecución.

Una de las diferencias entre este tipo de controlador difuso con otros controladores es que no necesita de un modelo matemático, no requiere de un modelo que describa la dinámica del sistema. Esto facilita demasiado el diseño del controlador para un sistema en particular, ya que al no requerirse de un análisis matemático o de describir por medio de ecuaciones la dinámica del mismo, se prescinde de esta práctica y se pasa netamente al análisis del funcionamiento de este controlador directamente sobre el sistema.

2.3.4 Redes Neuronales Artificiales (RNA)

Una de las muchas aplicaciones para las redes neuronales artificiales se definen para sistemas robóticos como son el control dinámico de trayectoria, controladores para robots en general, que son las que nos interesan. Según Cruz (2011), las redes neuronales artificiales son sistemas que realizan mapeos no lineales, su estructura principal se inspira en la estructura biológica de una neurona humana que consisten en procesadores lineales, estos por medio de una cantidad arbitraria o a veces no arbitraria llamada peso, establece cómo debe ser tratado este valor numérico.

Se llama red neuronal porque está inspirada en la estructura de la célula, la neurona animal o más especificamente la humana. La palabra artificial da a entender que se trata de un sistema que no hace parte o nace precisamente de la naturaleza propia.

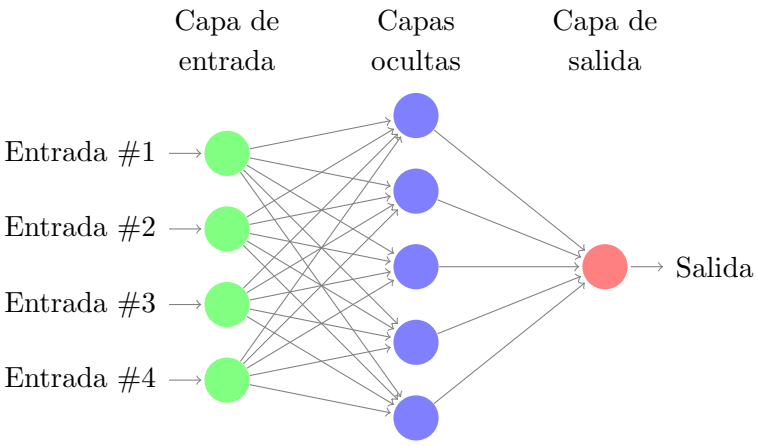

Figure 8: Estructura básica red neuronal artificial
Fuente: Autoría propia

Como se muestra en la Figura 8, la red neuronal artificial se compone de un conjunto de partes que procesan información y esta a su vez están conectadas entre sí, no necesariamente es el tipo de red que se usará más adelante pero se puede decir que es uno de los modelos que mejor representa a la conexión interna que posee una red neuronal artificial.

Los elementos básicos de una Red Neuroanl Artificial son:

- Neuronas o grupo de unidad básica de la red neuronal
- La conexión entre la neurona anterior y esta a su vez asociada a un valor o peso.
- Función de activación para cada neurona nombrada en el primer item.

La arquitectura de la red neuronal artificial describen la forma en la que se transforman los valores de las entradas en la salida deseada.

Por nombrar algunos tipos de topologías existentes según Cruz (2011), están las red Instar, red Outstar, red de Kohonen, redes recurrentes, red Hopfield, redes competitivas, red anfis, Learning Vector Quantization (LVQ), red de Hamming entre muchas otras. Las anteriores rede se usan más que todo para clasificar,predecir y reconocer patrones.

Las redes neuronales artificales son capaces de emular, aprender, clasificar simplemente conociendo la entrada y la salida de un sistema. Es decir lo trata como un sistema de caja negra y lo emula, lo imita. Inclusive si se le acopla un controlador que trabaje junto con la red neuronal artificial, son capaces de adaptarse y construir un sistema robusto y muy confiable.

Control supervisado: La red neuronal artificial es capaz de aprender un grupo de entradas y salidas optimas, y de esta manera es capaz de identificar cualquier sistema dinámico.

Control inverso directo: La red es capaz de aprender de forma inversa la dinámica del sistema, de como que aplicando estos datos es capaz de calcular lo que la red tiene que hacer en el sistema.

Retropropagación de utilidad: Es capaz de emular una función de transferencia u otra red neuronal por medio de este sistema de *backpropagation*.

Crítico adaptativo: Parecido al anterior pero sin la un sistema como modelo. Normalmente otra red se encarga de evaluar a esta y así obtener los valores correctos.

2.4 Creación de un Modelo en Simulink

A continuación se presenta brevemente una forma de introducir una ecuación diferencial en Simulink. También se explica como se ponen constantes dentro de una ejecución de diagramas de bloques también en Simulink. Esta explicación está basada en un articulo de Regalo Núñez (2016).

Partiendo de la Ecuación:

$$m\ddot{x} + c\dot{x} + kx = f(t) \tag{1}$$

Que representa un modelo matemático masa-resorte-amortiguador en donde se aplica una fuerza externa. En donde m es la masa que se mueve por acción del resorte, c es la constante de fricción del amortiguador o constante de viscosidad y k es la constante de elasticidad del resorte. En donde también $x(t)$ es la distancia con respecto al tiempo que se mueve la masa con respecto a su origen. En la Figura 9, se observa el diagrama de un sistema masa resorte amortiguador.

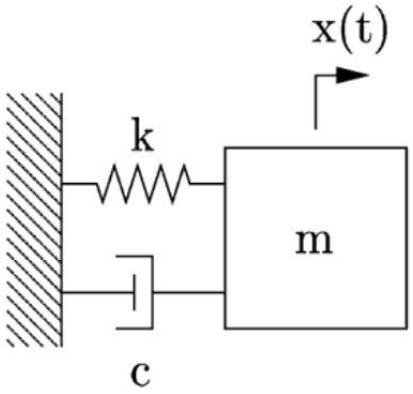

Figure 9: Modelo masa-resorte-amortiguador
Fuente: Sacado de Regalo Núñez (2016)

Antes de representarlo en Simulink se debe normalizar la ecuación, es decir, hacer que el coeficiente de la derivada de mayor orden sea uno. Ecuación 2.

$$\ddot{x} + \frac{c}{m}\dot{x} + \frac{k}{m}x = \frac{1}{m}f(t) \tag{2}$$

Luego de despeja la derivada de mayor orden. Ecuación 3.

$$\ddot{x} = \frac{1}{m}f(t) - \frac{c}{m}\dot{x} - \frac{k}{m}x \tag{3}$$

Como la ecuación diferencial es de orden dos, se necesitan dos bloques integradores en los cuales se introducen las condiciones iniciales $x_0 = x(0)$ y $x_0 = x(0)$. En la Figura 10 se puede ver el modelo en Simulink, en la Figura 11 se puede ver el contenido que hay dentro del subsistema creado, crear un subsistema es útil principalmente para ahorrar espacio y que se pueda ver más claramente el modelo que realizamos, para ello solo tenemos que seleccionar la parte del modelo que queremos meter dentro del subsistema y seleccionar Create Subsystem from Selection. Regalo Núñez (2016)

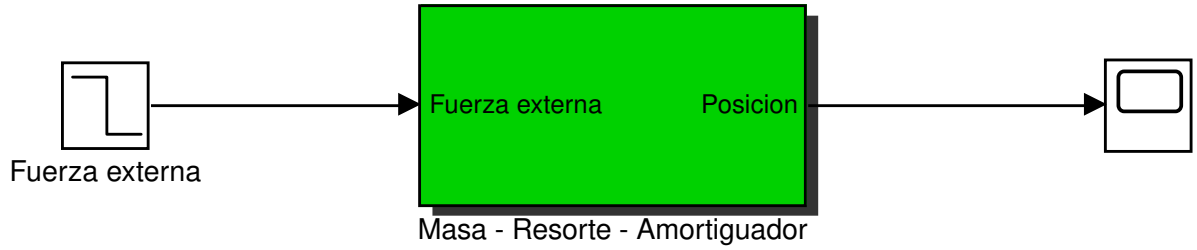

Figure 10: Modelo en Simulink
Fuente: Autoría propia

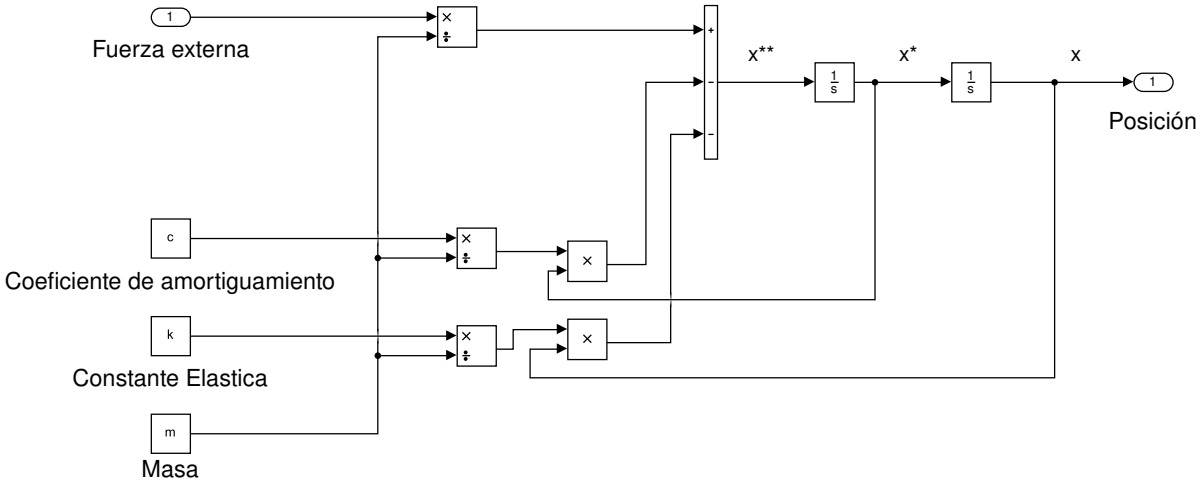

Figure 11: Contenido interno del subsistema
Fuente: Autoría propia

Se debe tener en cuenta que los valores de las constantes k,m y c se definen al comienzo de la simulación en la ventana de la Figura 12, esta ventana se activa al dar click derecho - Model Properties- Callbacks - InitFcn.

La respuesta a la salida del sistema se ve representada en un plano en función del tiempo como se puede apreciar en la Figura 13.

Es necesarios realizar ese ejercicio y entender un par de razones por las que simulink la toolbox de MatLab es fundamental a la hora de realizar procesos matemáticos y poder repre-

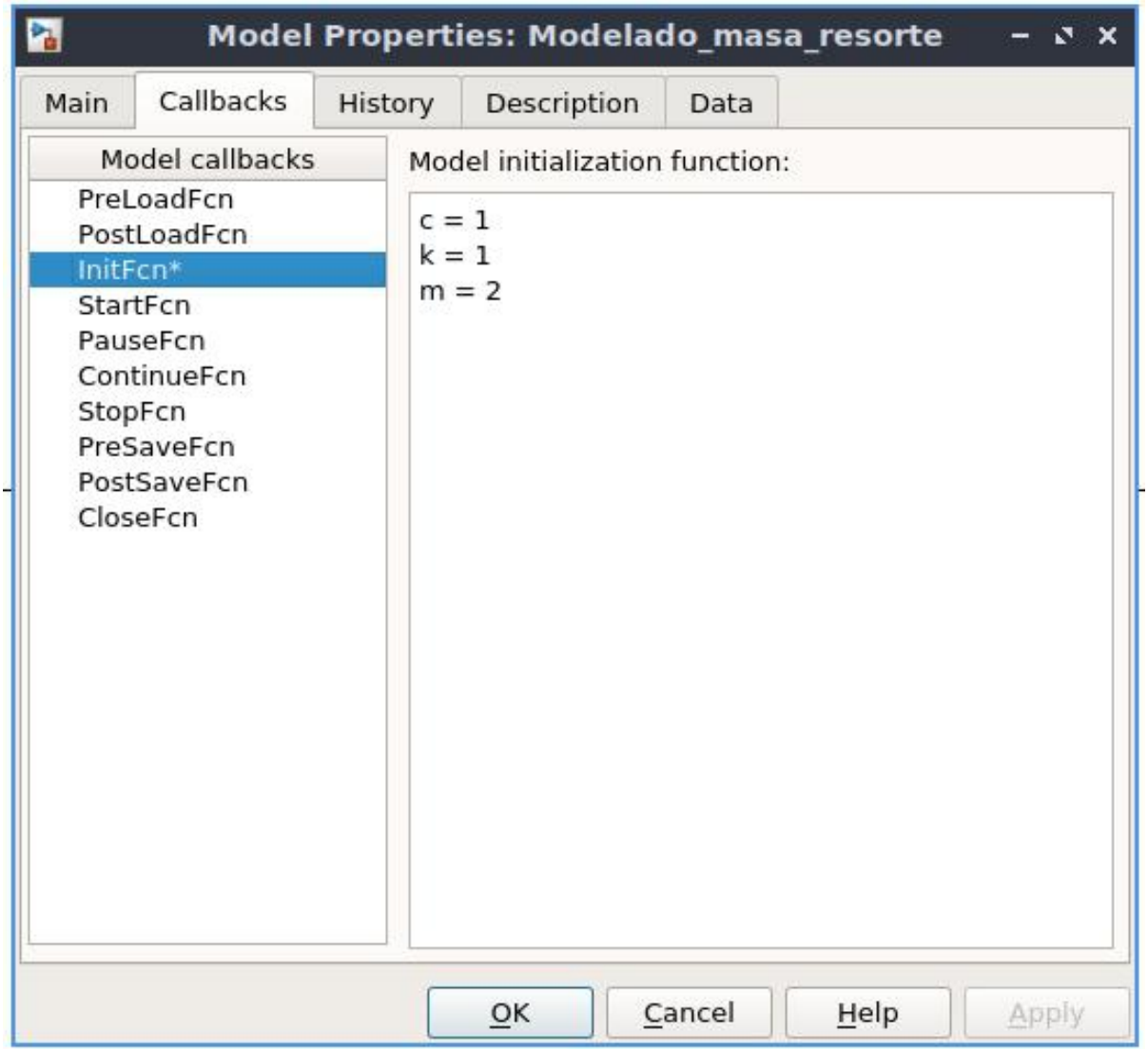

Figure 12: Ventana Model Properties
Fuente: Autoría propia

sentarlos por medio de bloques funcionales. Este es un lenguaje de programación de mucho más alto nivel lo cual nos permite enfocarnos en el problema a solucionar que en la programación que éste lleva implicita. Estos bloques que posee simulink dentro de su librería permite realizar funciones preconfiguradas y que son muy utilizadas tanto en la parte matemática como en la parte del estudio de diseño de controladores, sus aplicaciones y demás. Una vez creados nuestros sistemas a estudiar mediante simulaciones, descripciones de su dinámica y representaciones matemáticas de su comportamiento permiten el estudio de estos sistemas sin la necesidad de utilizar costosos equipos de laboratorio.

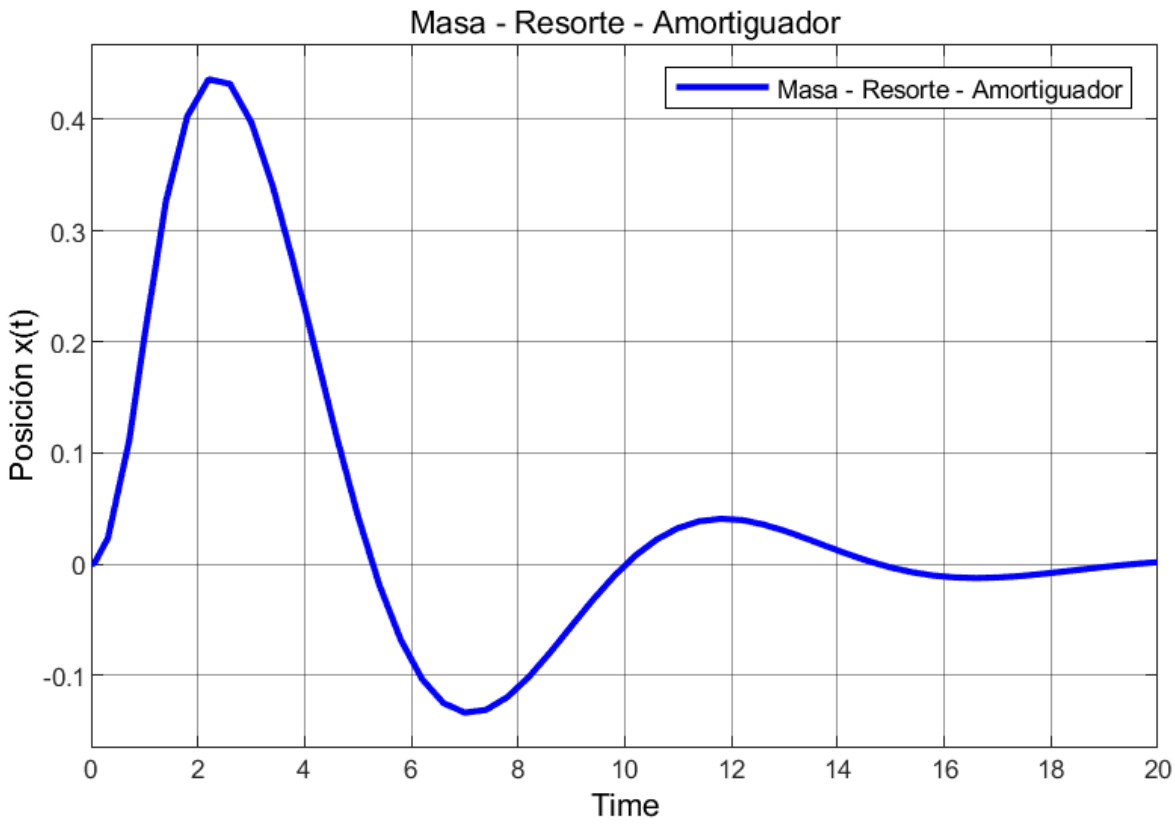

Figure 13: Respuesta a la salida del sistema
Fuente: Autoría propia

2.5 Síntesis

Se realizó un breve repaso por algunas de las estrategias de control que se han implementado a lo largo del tiempo sobre este sistema o planta de péndulo rotacional invertido. También se resumen algunos de los conceptos que se llegan a ver durante todo este documento a mencionar: Control PID, Control Difuso y Redes Neuronales. También se explicó de manera breve la forma de realizar modelos a partir de ecuaciones diferenciales de cualquier orden en simulink.

3 Modelado Matemático del Péndulo de Furuta

3.1 Introducción

A continuación se presenta de la manera más detallada posible un modelado matemático a cerca del péndulo rotacional invertido el cual será de vital importancia al momento de ejecutar cualquier tipo de estudio a cerca de éste.

El modelado matemático de cualquier sistema a controlar, ha de ser lo más importante a tener en cuenta; ya que es la base sobre la cual se van a realizar hipótesis, cálculos y demás pruebas de los controladores y el modelo matemático en sí. El siguiente modelado está basado en los trabajos de Gäfvert (2016) y Osorio Zúñiga et al. (2009)

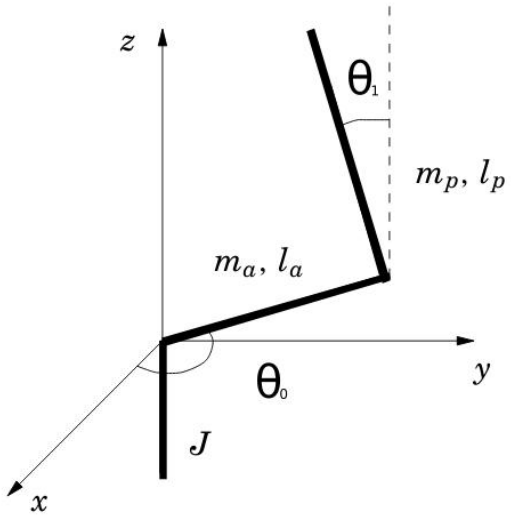

Figure 14: Diagrama del péndulo rotacional invertido con sus respectivas variables y parámetros. Fuente: Adaptado de Gäfvert (2016)

A continuación se en la Tabla 1 se detalla cada una de sus variables que acompaña al modelo físico del péndulo.

Símbolo	Variable
θ_0	Ángulo del brazo
θ_1	Ángulo del péndulo
τ	Torque aplicado al brazo
J	Inercia del brazo y motor
m_a	Masa del brazo
l_a	Longitud del brazo
m_p	Masa del péndulo
l_p	Longitud del péndulo
g	Gravedad
r_a	Posición radial del brazo
r_p	Posición radial del péndulo
I	Momento de inercia del péndulo
K	Energía Cinética
V	Energía Potencial
K_c	Energía Cinética motor
K_a	Energía Cinética brazo
K_p	Energía Cinética péndulo
V_c	Energía Potencial motor
V_a	Energía Potencial brazo
V_p	Energía Potencial péndulo
L	Sistema Lagrangiano o función Lagrangiana
V_m	Voltaje aplicado al motor DC
i	Sistema Lagrangiano o función Lagrangiana
R	Resistencia de armadura
L_a	Inductancia armadura
e_a	Fuerza contraelectromotriz
i_f	Constante corriente de campo
J_m	Momento inercia del motor
B	Coeficiente de fricción
T_m	Torque del motor
ω	Velocidad angular del motor

Table 1: Notación utilizada en este capítulo

3.2 Modelado a Partir de la Dinámica del Sistema

Las ecuaciones dinámicas de cualquier sistema mecánico se pueden obtener a partir de la mecánica clásica conocida (Newton). La desventaja de este formalismo es el uso de variables en forma de vectores. Por lo tanto, cuando las articulaciones aumentan o hay rotaciones, el análisis será particularmente complejo para este sistema en especial. En estos casos, es ventajoso utilizar ecuaciones de movimiento lagrangianas, que tienen forma escalar debido a consideraciones energéticas puras y, por tanto, facilitan el análisis de cualquier sistema mecánico.Duarte et al. (2016)

El péndulo de furuta, descrito en la Figura 14, se puede descomponer en sus dos partes más fundamentales como son el brazo y el péndulo. Estas a su vez se describen en dos ecuaciones diferenciales dinámicas, que parten de un modelado dinámico que surge a partir de las ecuaciones de movimiento de Lagrange como se puede ver en las ecuaciones (4).

A continuación se explica de manera concisa la obtención del modelo dinámico del péndulo de furuta.

3.2.1 Cinemática

La ubicación de un punto P acerca de el péndulo puede ser explicado por el vector posición Ecuación (4):

$$r(r_a, r_p) = (r_x(r_a, r_p), r_y(r_a, r_p), r_z(r_a, r_p))^T \tag{4}$$

En donde r_x, r_x y r_x se expresan como Ecuación (5)

$$\begin{aligned} r_x(r_a, r_p) &= r_a \cos\theta_0 - r_p \sin\theta_0 \sin\theta_1 \\ r_y(r_a, r_p) &= r_a \sin\theta_0 + r_p \sin\theta_1 \cos\theta_0 \\ r_z(r_a, r_p) &= r_p \cos\theta_1 \end{aligned} \tag{5}$$

La variable r_a es la posición radial del brazo, y r_p es la posición radial del péndulo. Las distancias radiales son medidas desde el centro de rotación de los cuerpos. Tomando derivadas respecto al tiempo de la Ecuación (4) se obtiene una expresión para la velocidad de un punto P sobre el péndulo Ecuación (6)

$$v(r_a, r_p) = (v_x(r_a, r_p), v_y(r_a, r_p), v_z(r_a, r_p))^T \tag{6}$$

Se derivan cada uno de los componentes del vector de la Ecuación (5) para así obtener los componentes de la Ecuación (7)

En donde v_x, y_x y z_x se describen en la Ecuación (7)

$$\begin{aligned} v_x(r_a, r_p) &= -r_a \sin\theta_0\dot{\theta}_0 - r_p \cos\theta_1 \sin\theta_0\dot{\theta}_1 - r_p \sin\theta_1 \cos\theta_0\dot{\theta}_0 \\ v_y(r_a, r_p) &= r_a \cos\theta_0\dot{\theta}_0 + r_p \cos\theta_1 \cos\theta_0\dot{\theta}_1 - r_p \sin\theta_1 \sin\theta_0\dot{\theta}_0 \\ v_z(r_a, r_p) &= -r_p \sin\theta_1\dot{\theta}_1 \end{aligned} \tag{7}$$

Esto se usa luego para expresar el cuadrado de la magnitud de la velocidad de P, es decir la Ecuación (8)

$$v^2(r_a, r_p) = (r_a^2 + r_p^2 \sin\theta_1)\dot{\theta}_1^2 + 2r_a r_p \cos\theta\dot{\theta}_0\dot{\theta}_1 + r_p^2\dot{\theta}_1^2 \tag{8}$$

3.2.2 Energía Cinética y Energía Potencial

Expresiones para la energía cinética y potencial son obtenidas en esta sección. La energía cinética K es obtenida a partir de la solución de la integral de la Ecuación (9)

$$K = \frac{1}{2}\int v^2 dm \tag{9}$$

La energía potencial V, se expresa de la forma. Ver Ecuación (10)

$$V = g\int r_z dm \tag{10}$$

La sumatoria de las energías potenciales y cinéticas de cada una de las partes que componen el sistema completo en la Ecuación (11) nos da el total de la energía completa

$$\begin{aligned} K &= K_c + K_a + K_p \\ V &= V_c + V_a + V_p \end{aligned} \tag{11}$$

En donde el subíndice c hace referencia al motor o pilar central. El subíndice a al brazo y el subíndice p al péndulo del que hace parte de todo el sistema, llámese péndulo rotacional invertido. A continuación se describen las ecuaciones que describen cada uno de los elementos de la sumatoria de las energías potencial y cinética. De la Ecuación (12) a la (17)

Pilar Central, Ecuación (12) y (13)

$$2K_c = J\dot{\theta}_0^2 \tag{12}$$

$$V_c = 0 \tag{13}$$

Brazo Horizontal Ecuación (14) y (15).

$$\begin{aligned} 2K_a &= \int_0^{la} v^2(s,0)\frac{m_a}{l_a}\,ds \\ K_a &= \frac{1}{3}m_a l_a^2 \dot{\theta}_0^2 \end{aligned} \tag{14}$$

$$V_a = 0 \tag{15}$$

Brazo Pendular Ecuación (16) y (17)

$$\begin{aligned} 2K_p &= \int_0^{lp} v^2(ra,s)\frac{m_p}{l_s}\,ds \\ &= m_p(l_a^2 + \frac{1}{3}l_p^2\sin^2\theta_1)\dot{\theta}_0^2 + m_a l_a l_p \cos\theta_1 \dot{\theta}_0\dot{\theta}_1 + \frac{1}{3}m_p l_p^2 \dot{\theta}_1^2 \end{aligned} \tag{16}$$

$$\begin{aligned} V_p &= g\int_0^{lp} r_z(la,s)\frac{m_p}{l_p}\,ds \\ &= \frac{1}{2}m_p g l_p \cos\theta_1 \end{aligned} \tag{17}$$

3.2.3 Ecuaciones Dinámicas a Partir de Lagrange

El modelo dinámico del péndulo de furuta, al ser un sistema de dos grados de libertad, está dado por las siguientes ecuaciones de movimiento de Lagrange. Ecuaciones (18)

$$\begin{aligned} \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_0} - \frac{\partial L}{\partial \theta_0} &= \tau \\ \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}_1} - \frac{\partial L}{\partial \theta_1} &= 0 \end{aligned} \tag{18}$$

$\dot{\theta}_0$ = Velocidad angular del brazo
$\dot{\theta}_1$ = Velocidad angular del péndulo

A continuación se muestra la ecuación del Lagrangiano Ecuación (19)

$$L = K - V \tag{19}$$

L = Sistema Lagrangiano o función Lagrangiana
K = Energía Cinética
V = Energía Potencial

Siendo τ el torque externo aplicado al brazo horizontal por el motor. Las derivadas parciales son Ecuaciones (20)

$$\begin{aligned}
&\frac{\partial L}{\partial \theta_0} = 0 \\
&\frac{\partial L}{\partial \dot{\theta}_0} = (J + (\frac{1}{3}m_a + m_p)l_a^2 + \frac{1}{3}m_p l_p^2 \sin^2\theta_1)\dot{\theta}_1 + \frac{1}{2}m_p l_a l_p \cos\theta_1 \dot{\theta}_1 \\
&\frac{\partial L}{\partial \theta_1} = \frac{1}{3}m_p l_p^2 \cos\theta_1 \sin\theta_1 \dot{\theta}_0^2 - \frac{1}{2}m_p l_a l_p \sin\theta_1 \dot{\theta}_0 \dot{\theta}_1 + \frac{1}{2}m_p g l_p \sin\theta_1 \\
&\frac{\partial L}{\partial \dot{\theta}_1} = \frac{1}{2}m_p l_a l_p \cos\theta_1 \dot{\theta}_0 + \frac{1}{3}m_p l_p^2 \dot{\theta}_1
\end{aligned} \tag{20}$$

Luego, a partir de la ecuación del Lagrangiano (19), que se asocia con las Ecuaciones 11, y luego de realizar las derivadas correspondientes en el sistema de Ecuaciones (18), la dinámica del péndulo de Furuta se encuentra expresan en las siguientes dos Ecuaciones (21)

$$\begin{aligned}
&\tau = (\alpha + \beta \sin^2\theta_0)\ddot{\theta}_1 + \gamma \cos\theta_1 \ddot{\theta}_1 + 2\beta \cos\theta_1 \sin\theta_1 \dot{\theta}_0 \dot{\theta}_1 - \gamma \sin\theta_1 \dot{\theta}_1^2, \\
&0 = \gamma \cos\theta_1 \ddot{\theta}_0 + \beta \ddot{\theta}_1 - \beta \cos\theta_1 \sin\theta_1 \dot{\theta}_0^2 - \sigma \sin\theta_1
\end{aligned} \tag{21}$$

En donde β, γ, σ y α son Ecuaciones (22) y (23)

$$\alpha = J + (\frac{1}{3}m_a + m_p)l_a^2, \qquad \beta = \frac{1}{3}m_p l_p^2, \tag{22}$$

$$\gamma = \frac{1}{2}m_p l_a l_p, \qquad \sigma = \frac{1}{2}m_p g l_p \tag{23}$$

3.3 Linealización del Punto de Equilibrio

A continuación se procede a linealizar las ecuaciones dinámicas del sistema partiendo de un punto de equilibrio.

Para realizar un análisis al comportamiento de un sistema de péndulo invertido rotacional se requiere de una linealización de sus ecuaciones diferenciales que describen su dinámica en torno a un punto de equilibrio, que para este caso es la posición vertical invertida del péndulo.

Lo ideal sería que a partir de dichas ecuaciones se realice la debida linealización sobre un punto de equilibrio de las ecuaciones que describen el comportamiento del péndulo invertido rotacional. Pero en este caso y para efectos prácticos se realiza la linealización de las ecuaciones que describen el comportamiento de un péndulo invertido sobre un riel o carro.

En la Figura 15, se realiza una analogía del diagrama del péndulo de furuta con el diagrama del péndulo invertido sobre riel o carro. Esto se realiza para poder linealizar el sistema en el punto de equilibrio de una manera más cómoda, dado que las ecuaciones diferenciales que representan al péndulo de furuta son algo más complejas al momento de querer linealizarlas sobre un punto de equilibrio.

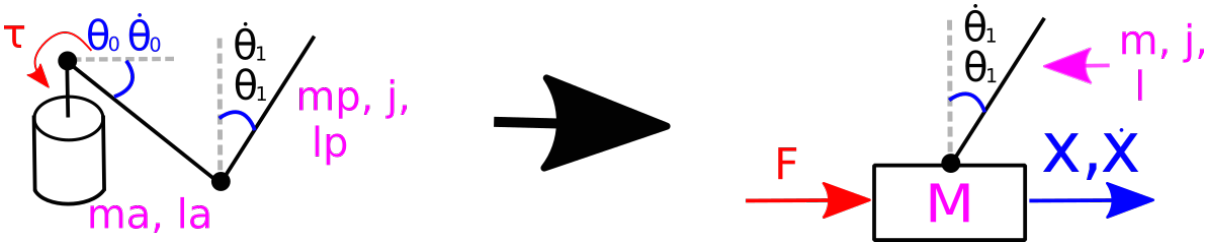

Figure 15: Conversión de parámetros de péndulo de furuta al péndulo invertido sobre carro.
Fuente: Autoría propia

Lo primero sería realizar un diagrama de cuerpo libre del péndulo invertido sobre un carro, donde posteriormente se deducen las ecuaciones de fuerzas que actúan sobre el sistema péndulo invertido sobre carro. Esto se realiza con el fin de conocer de donde sale la linealización del péndulo invertido sobre un carro y en últimas se realiza la comparación con respecto al péndulo de furuta y se utiliza esta ecuación linealizada para realizar los controladores usados sobre el péndulo de furuta.

La siguiente secuencia matemática para sacar las ecuaciones que describen el comportamiento del sistema péndulo-carro, se basa en la descrita en la página web de las universidades de Michigan, CMU y Detroit Mercy. (Bill Messner (2021))

Se realiza la sumatoria de las fuerzas que se muestran en la Figura 16, en la dirección horizontal y se obtiene la Ecuación (24)

$$M\ddot{x} + b\dot{x} + N = F \tag{24}$$

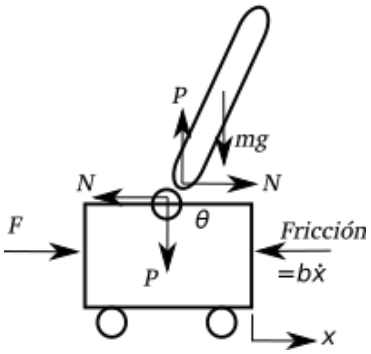

Figure 16: Diagrama de cuerpo libre péndulo - carro.
Fuente: Autoría propia

No se realiza sumatoria de fuerzas en el eje vertical del carro, ya que no se obtiene información relevante para las ecuaciones dinámicas.

La sumatoria de fuerzas del péndulo en dirección horizontal, se puede apreciar en la Ecuación (25)

$$N = M\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta \tag{25}$$

Si reemplaza la ecuación (25) en la ecuación (24) se obtiene una de las dos ecuaciones que describen el comportamiento de todo el sistema.

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F \tag{26}$$

Para obtener la segunda ecuación de movimiento de este sistema, sume las fuerzas perpendiculares al péndulo. Resolver el sistema a lo largo de este eje simplifica enormemente las matemáticas. Debería obtener la siguiente Ecuación (27).

$$P\sin\theta + N\cos\theta - mg\sin\theta = ml\ddot{\theta} + m\ddot{x}\cos\theta \tag{27}$$

Para reemplazar los términos P y N en la ecuación (27), sume los momentos alrededor del centroide del péndulo para obtener la Ecuación (28).

$$-Pl\sin\theta - Nl\cos\theta = I\ddot{\theta} \tag{28}$$

Combinando estas dos últimas expresiones, obtienes la segunda ecuación gobernante Ecuación (29).

$$(I + ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta \tag{29}$$

A partir de las ecuaciones obtenidas en (29) y (26) se proceden a linelizar dichas ecuaciones sobre la posición del péndulo invertida en equilibrio, más específicamente en $\theta = \pi$. Considerando que el sistema permanece dentro de una región no mayor a 20 grados de la posición del péndulo invertida en equilibrio. En donde ϕ es la desviación de la posición del péndulo en equilibrio, entonces $\theta = \pi + \phi$. Suponiendo que la desviación, es decir, (ϕ) desde la posición de equilibrio del péndulo es muy pequeña, se realizan las siguienten suposiciones en las Ecuaciones (30.)

$$\begin{aligned} \cos\theta = \cos(\pi + \phi) &\approx -1 \\ \sin\theta = \sin(\pi + \phi) &\approx -\phi \\ \dot{\theta}^2 = \dot{\phi}^2 &\approx 0 \end{aligned} \tag{30}$$

Después de sustituir las aproximaciones realizadas en (30), en las ecuaciones dinámicas no lineales, se llega a las dos ecuaciones de movimiento linealizadas. Se sustituye F por la notación u que normalmente se usa en control para denotar una entrada al sistema Ecuaciones (31) y (32).

$$(I + ml^2)\ddot{\theta} - mgl\theta = ml\ddot{x} \tag{31}$$

$$(M + m)\ddot{x} + b\dot{x} - ml\ddot{\theta} = u \tag{32}$$

3.4 Modelado Matemático del Motor de Corriente Continua

Es esta sección, se explica de forma breve un modelado de un motor de corriente continua el cual será de gran importancia al momento de realizar simulaciones y que estas sean lo más cercanas posible a un modelo real.

Según Giraldo et al. (2012) un motor DC, motor a corriente continua o simplemente motor CC, es un dispositivo mecánico - eléctrico es cuál al aplicarse un voltaje en corriente continua es capaz de proporcionar un movimiento rotatorio a su salida.

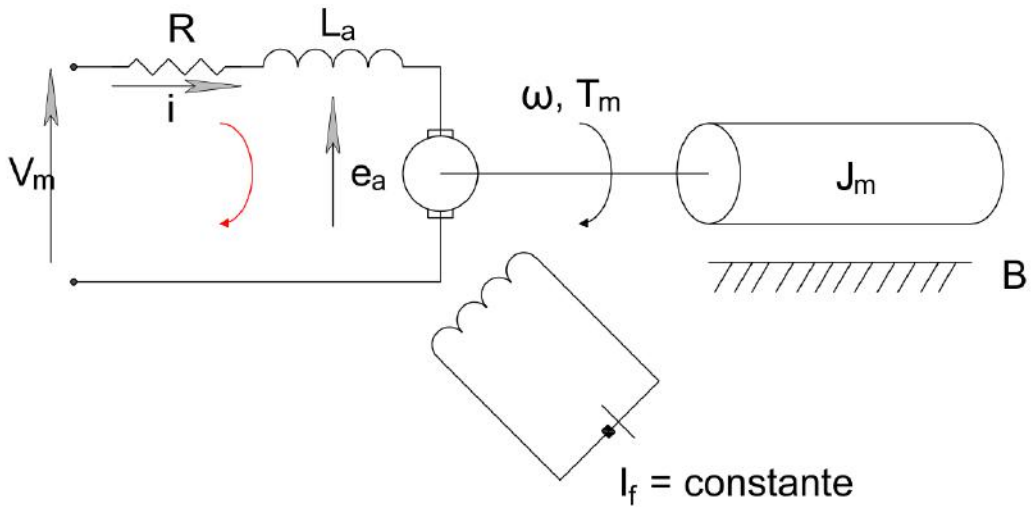

Figure 17: Representación electromecánica de un motor DC
Fuente: Autoría Propia

En la Figura 17 se puede observar un circuito el cual es equivalente a un motor DC, se expresa en función de su parte mecánica y de su parte eléctrica. El parámetro V_m es el voltaje aplicado a las terminales del motor eléctrico, éste mismo se toma como entrada del sistema al momento de realizar el modelado. La salida del mismo es el torque y la velocidad angular, T_m y ω respectivamente.

Para obtener la primera ecuación diferencial, se inicia por sacar un análisis de la malla en la sección eléctrica de la figura mostrada anteriormente Ecuación (33).

$$L_a \frac{di(t)}{d(t)} = v(t) - Ri(t) - E_a(t) \tag{33}$$

Dado que E_a es el voltaje que ha sido generado cuando el conductor del inducido o armadura es movido a través del flujo de excitación de la corriente de excitación i_fVer Orduñez Ruiz (2010).

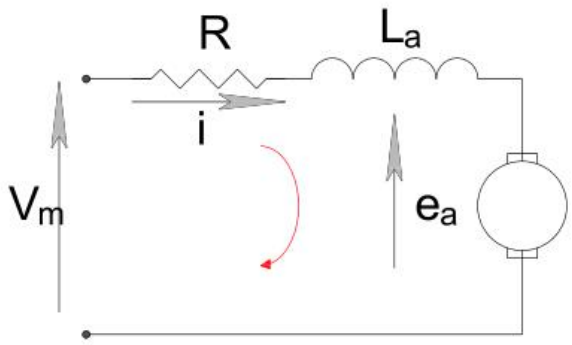

Figure 18: Calculando voltajes de malla motor DC
Fuente: Autoría Propia

La sección mecánica que se ilustra en la Figura 17 se presenta en la siguiente Ecuación 34, en la cual están representados los momentos, en donde el torque del motor T_m es igual a la suma del momento de inercia del motor J_m por la aceleración angular $\frac{d\omega(t)}{dt}$ más el coeficiente del motor B por la velocidad angular del motor $\omega(t)$ obteniendo de esta manera la segunda ecuación diferencial Ecuación (34).

$$T_m(t) = J\frac{d\omega(t)}{dt} + B\omega(t) \tag{34}$$

La tercer ecuación diferencial 35 es una estimación de la relación K_a de la fuerza contra-electromotriz y la velocidad angular a la salida del eje del motor $\omega(t)$. También se establece una relación K_m eléctrica y mecánica entre la corriente de armadura $i(t)$ y el torque $T_m(t)$ a la salida del motor Ecuación 36 .

$$E_a(t) = K_a\omega(t) \tag{35}$$

$$T_m(t) = K_m i(t) \tag{36}$$

3.5 Función de Transferencia

Una función de transferencia es una representación, un modelo matemático en el dominio de la frecuencia, también conocido como dominio de S en donde el cociente es igual a la salida en razón de su entrada, normalmente esta función parte de una ecuación diferencial linealizada.

3.5.1 Motor DC

Para este caso se debe tener en cuenta que el trabajar con una función de transferencia normalmente da a entender que se está trabajando con un sistema lineal o que ha sido linealizado. Por eso se realizó la linealización en el punto de equilibrio del péndulo.

Para el caso del motor DC, se pasan sus ecuaciones diferenciales al dominio de S o de la frecuencia y se procede a obtener la función de transferencia. La idea con esto es poder manejar ecuaciones diferenciales y obtener una función que me exprese la salida sobre la entrada para este sistema.

Según las ecuaciones (33), (34), (35) y (36) estas se deben pasar al dominio S antes de armar nuestra función de transferencia. Quedarían así Ecuaciones (37), (38), (39), (40):

$$LSi(s) = v(s) - Ri(s) - Ea(s) \tag{37}$$

$$T_m(s) = JS\omega(s) + B\omega(s) \tag{38}$$

$$E_a(s) = K_a\omega(s) \tag{39}$$

$$T_m(s) = K_m i(s) \tag{40}$$

Después de haber pasado las ecuaciones al dominio S se procede a armar nuestra función de transferencia, la idea es tener una función de transferencia en donde su entrada sea el voltaje aplicado al motor y su salida sea torque.

La función de transferencia quedaría de la siguiente manera, ver (41):

$$\frac{T_m(s)}{V_m(s)} = \frac{K_m(Js + B)}{LJs^2 + (RJ + LB)sRB + K_m K_a} \tag{41}$$

3.5.2 Péndulo Linealizado

La función de transferencia para un punto en específico del péndulo rotacional invertido permite establecer un control lineal sobre esa zona. La idea principal es una vez superada la zona swing up o de balanceo que el péndulo permanezca en posición de equilibrio invertido y arriba.

Al igual que con el motor DC, se procede a pasar las ecuaciones linealizadas al dominio S o al dominio de la frecuencia. De las ecuaciones 31, 32 se pasan al dominio S y se obtienen las siguientes Ecuaciones (42), (43)

$$(I + ml^2)\Theta(s)s^2 - mgl\Theta(s) = mlX(s)s^2 \tag{42}$$

$$(M + m)X(s)s^2 + bX(s)s - ml\Theta(s)s^2 = U(s) \tag{43}$$

Para este caso se debe establecer la entrada a la función de trasferencia como la fuerza que se aplica a este sistema y a la salida tenemos la posición del péndulo recordando que se está trabajando sobre una zona de equilibrio.

Se deja a $X(s)$ a un lado de la ecuación para poder reemplazar el valor en la ecuación (43)

$$X(s) = \left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\Theta(s) \tag{44}$$

Así se vería la ecuación (45)

$$(M + m)\left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\Theta(s)s^2 + b\left[\frac{I + ml^2}{ml} - \frac{g}{s^2}\right]\Theta(s)s - ml\Theta(s)s^2 = U(s) \tag{45}$$

La función de transferencia queda de la siguiente manera ver (46):

$$\frac{\Theta(s)}{U(s)} = \frac{\frac{ml}{q}s^2}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \tag{46}$$

En donde (47)

$$q = [(M + m)(I + ml^2) - (ml)^2] \tag{47}$$

3.6 Síntesis

Un modelado matemático nos permite realizar una aproximación de un sistema cualquiera y estudiarlo de la manera más detallada posible. La forma en la cual las ecuaciones nos permiten obtener la información necesaria capaz de describir de forma muy exacta cada uno de los movimientos de un sistema es de gran valor al momento de estudiar el mismo y realizar conclusiones al respecto.

No obstante se debe tener sumo cuidado al momento de tratar con sistemas sub-actuados ya que la manera en la que estos se deben modelar varían dependiendo de su aplicación. Se debe tener en cuenta que un controlador PID ha de funcionar para sistemas que se encuentren linealizados, así se evitan confusiones al momento de cualquier implementación del mismo.

4 Diseño de los Controladores

4.1 Introducción

Un controlador permite a la persona que lo implementa automatizar procesos que antes requerían de un operario para su ejecución. Un sistema de péndulo rotacional invertido requiere normalmente de dos controladores conmutados para su correcta ejecución. Uno ubicado en la zona de swing up o zona de balanceo y otro controlador en la zona de equilibrio o zona lineal. A continuación se presentan métodos por los cuales se plantean una serie de controladores que han de ser capaces de equilibrar el péndulo rotacional invertido.

4.2 Control Swing Up

Dentro del control de un péndulo rotacional invertido se debe tener en cuenta que se trata de un sistema altamente no lineal. Dentro del cual se tiene en cuenta que para lograr un control global de manera efectiva se deben establecer dos tipos de control para dos momentos en específico. El primero comienza con el péndulo en posición de reposo con el valor de su ángulo igual a cero grados. El otro momento es cuando el péndulo entra en zona de equilibrio, es decir, en un rango de 160 y 200 grados con respecto a su valor inicial. En este punto es donde el controlador PID empieza a trabajar, con un setpoint de 180 grados, manteniendo el péndulo en posición de equilibrio en un movimiento angular no mayor a 20 grados. En la Figura (19), se puede apreciar un diagrama de los momentos en donde cada uno de los controladores están activos.

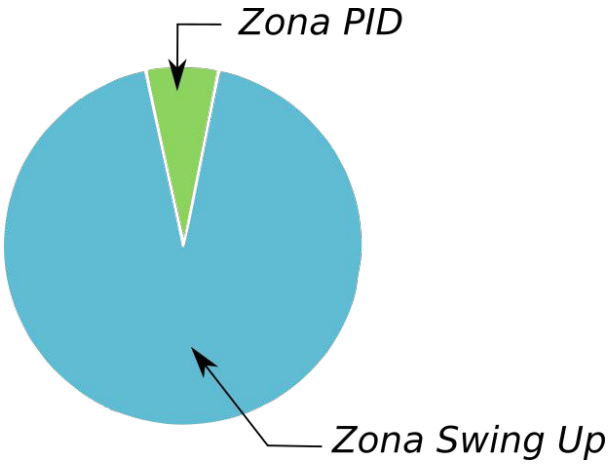

Figure 19: Swing Up & PID
Fuente: Autoría Propia

Balancear un péndulo invertido es un experimento clásico que se usa en laboratorios de control, ver Furuta et al. (1991). Puede lograrse mediante un control de tiempo mínimo. El péndulo se controla simplemente de tal manera que su energía es impulsada hacia un valor igual a la posición vertical en estado estacionario. Luego, el péndulo se acerca a la posición vertical donde se puede atrapar con una estrategia adecuada.

Para hallar la estrategia de control que va a balancear el péndulo para poder mantenerlo en posición de equilibrio de procede a realizar el siguiente procedimiento.

Considere un solo péndulo. Sea su masa [m] y el momento de inercia con respecto al punto de pivote sea [J]. Además, sea [I] la distancia desde el pivote hasta el centro de masa. El ángulo entre la vertical y el péndulo es [g], donde [g] es positivo en el sentido de las agujas del reloj. La aceleración de la gravedad es [g] y la aceleración del pivote es [u] . Las ecuaciones de movimiento del péndulo son Ecuación (48):

$$J_p\ddot{\theta} - mgl\sin\theta + mul\cos\theta = 0 \tag{48}$$

La idea básica es incrementar la energía del sistema de tal manera que finalmente tenga suficiente energía para llegar a una región cercana al punto de equilibrio inestable. Allí, se realiza un intercambio a una ley de control local que estabiliza el péndulo.

En la posición de equilibrio el péndulo tiene comportamientos diferentes a los que describen la Ecuación (48), por ejemplo su posición se establece como cero en ese punto, la ecuación que describe la energía del péndulo está dada por la Ecuación (49)

$$E = \frac{1}{2}J_p\dot{\theta}^2 - mgl(\cos\theta - 1) \tag{49}$$

En la ecuación (49) se puede observar que para llevar a la posición de equilibrio el péndulo, se deben hacer los valores de velocidad y de posición a cero. Giraldo et al. (2012)

Por medio de la teoría desarrollada por el físico y matemático de nacionalidad Rusa Aleksandr Lyapunov es posible encontrar una señal de control que sea capaz de balancear el péndulo hasta una posición donde pueda actuar adecuadamente el controlador PID o cualquier otra estrategia de control lineal. La siguientes ecuación es una aproximación a dicha estrategia Valera et al. (2002)

$$u = k(E - E_0)sign(\dot{\theta}\cos\theta) \tag{50}$$

No confundir $sign()$ con $sin()$ ya que la primera es una función de signo en donde el resultado que se obtiene de su salida es un signo ya sea positivo o negativo. El valor k es una variable de diseño la cual se puede cambiar dependiendo de la necesidad.

En la Figura 20, se expresa de manera gráfica el controlador Swing Up que será usado en las simulaciones. Su implementación es sencilla pero su aplicación es potente. Por medio de la adquisición de los datos como posición angular y velocidad angular se es capaz de balancer el péndulo de sistema hasta la zona deseada, o la zona PID. Se realiza un pequeño desfase al momento de ingresar la posición angular que nos ayuda a que el sistema se balancee en la dirección correcta.

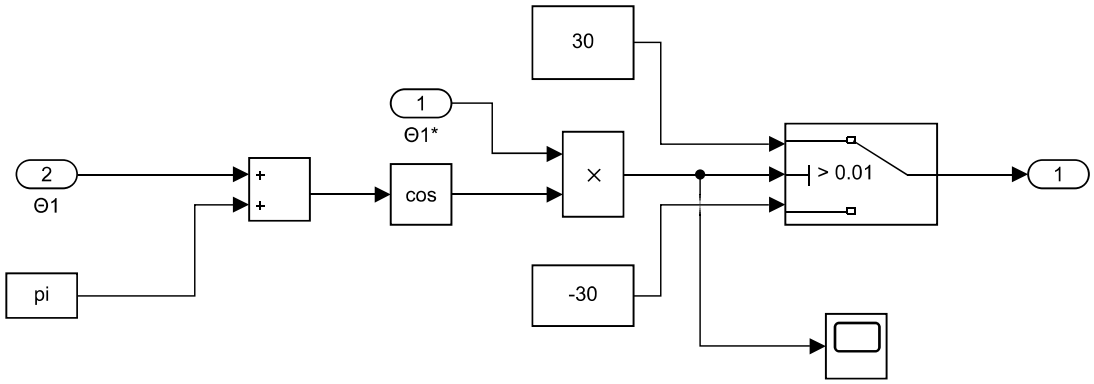

Figure 20: Controlador Swing Up en simulink
Fuente: Autoría propia

4.3 Control PID

Es necesario mostrar cómo se diseña un controlador PID de la manera más funcional posible. En esta sección se muestra que un controlador PID es capaz de estabilizar una función de transferencia del péndulo rotacional invertido en posición de equilibrio. Para ello se utiliza la función de transferencia de la zona de equilibrio del péndulo erguido y se le aplica un controlador PID en lazo cerrado, se muestran los resultados y se varían sus constantes Kp, Ki y Kd para posterior a ello ver su comportamiento. Una vez viendo lo que ocurre con el controlador se aplica la misma estrategia en la simulación junto con el controlador Swing Up.

Como se mencionó en el marco teórico los controladores PID son usados en sistema altamente lineales ya que su comportamiento a través del tiempo permite que su control sea el esperado por la persona que aplica o ejecuta este control.

Una función de transferencia se caracteriza por ser una representación lineal de un sistema en función de sus salida - entrada. A continuación se usa la función de transferencia obtenida en 46

La función de transferencia mostrada en (51), representa el punto de equilibrio del péndulo rotacional invertido para la aplicación del controlador PID

$$\frac{\Theta(s)}{U(s)} = \frac{\frac{ml}{q}s^2}{s^4 + \frac{b(I+ml^2)}{q}s^3 - \frac{(M+m)mgl}{q}s^2 - \frac{bmgl}{q}s} \tag{51}$$

Un controlador PID, una función de transferencia de un sistema lineal son sistemas SISO (Single Input - Single Output) o sistemas de entrada única y salida única.

Recordar que esta función de transferencia está basada en el sistema de péndulo invertido

por simplicidad en los cálculos, igualmente la masa del brazo ha de ser emulada como la masa del carro en movimiento y otros valores al momento de hacer la equivalencia con el péndulo rotacional invertido deben cambiarse dichos valores. Los valores han sido sacados de una medición directa de los componentes del péndulo rotacional invertido, de esto se habla en más detalle en Capítulo 6 Desarrollo y Construcción.

La dinámica para encontrar los valores adecuados para el controlador PID es empezar usando la unidad dentro de las constantes proporcional, integral y derivativa de nuestro controlador. Una vez hecho esto se empieza aumentando abruptamente la constante proporcional por ejemplo cien veces su valor y así obtener un sistema oscilante. Luego se aumenta 25 veces el valor de la constante derivativa y según el comportamiento del sistema se varía el valor de la constante integral.

A continuación se muestra la Tabla 2 con los datos usados en la función de transferencia para probar el controlador PID y ver si este responde ante perturbaciones en el sistema. Bill Messner (2021)

Constante	Valor	Descripción
M	$0.5Kg$	Masa del carro o de brazo en Kilogramos
m	$0.2Kg$	Masa del péndulo en Kilogramos
I	$0.006Kg.m^2$	Inercia del péndulo en Kilogramo-Metro cuadrado
b	$0.1N.m.s$	C. de fricción en Newton-Metro-Segundo
g	$9.82m/s^2$	Gravedad en metros por segundo cuadrado
l	$0.3m$	Longitud del péndulo en metros

Table 2: Constantes motor eléctrico

Según los valores de la tabla se empieza definiéndolos como variables para poder armar la función de transferencia en MatLab.

```
1
2 M = 0.089;
3 m = 0.056;
4 b = 0.1;
5 I = 0.00000847;
6 g = 9.81;
7 l = 0.15;
8
9
10 q = (M+m)*(I + m*l*l) - (m*l)^2;
11 s = tf('s');
12
13
14 P_pend = (m*l*s/q)/(s^3 + (b*(I + m*l^2))*s^2/q - ((M + m)*m*g*l)*s/q - b*m*g*l
       /q);
15
16
17 Kp = 1;
18 Ki = 1;
19 Kd = 1;
20 C = pid(Kp,Ki,Kd);
21 T = feedback(P_pend,C);
22
23
24 t=0:0.01:10;
25 impulse(T,t)
26
27 title({'Respuesta de la posicion del pendulo a un impulso perturbacion';'con
       Controlador PID: Kp = 1, Ki = 1, Kd = 1'});
28 ylabel("Amplitud");
29 grid on
```

Figure 21: Algoritmo PID

En la Figura 22 se observa que los valores de las constantes Kp, Ki y Kd no son lo suficientemente asertivos para lograr una estabilización dentro de este sistema. En otras palabras, la perturbación realizada sobre el sistema pendular es capaz de desestabilizarlo y no logra mantenerse en equilibrio.

A continuación se procede a ejecutar el mismo código mostrado anteriormente pero con la característica que se aumentará abruptamente el valor de la constante proporcional y evaluar su resultado.

Tal y como se muestra en la Figura 23, el aumento de la constante proporcional, es decir, Kp es capaz de estabilizar el sistema a los 0.2 segundos, pero aún así logra ser desestabilizado abruptamente dando lugar a unas oscilaciones no deseadas. Para ello se procede a aumentar la constante Kd del sistema, es decir la constante derivativa hasta un valor de 50, la idea principal es lograr que la perturbación que somete al sistema sea casi imperceptible ante la presencia del controlador PID.

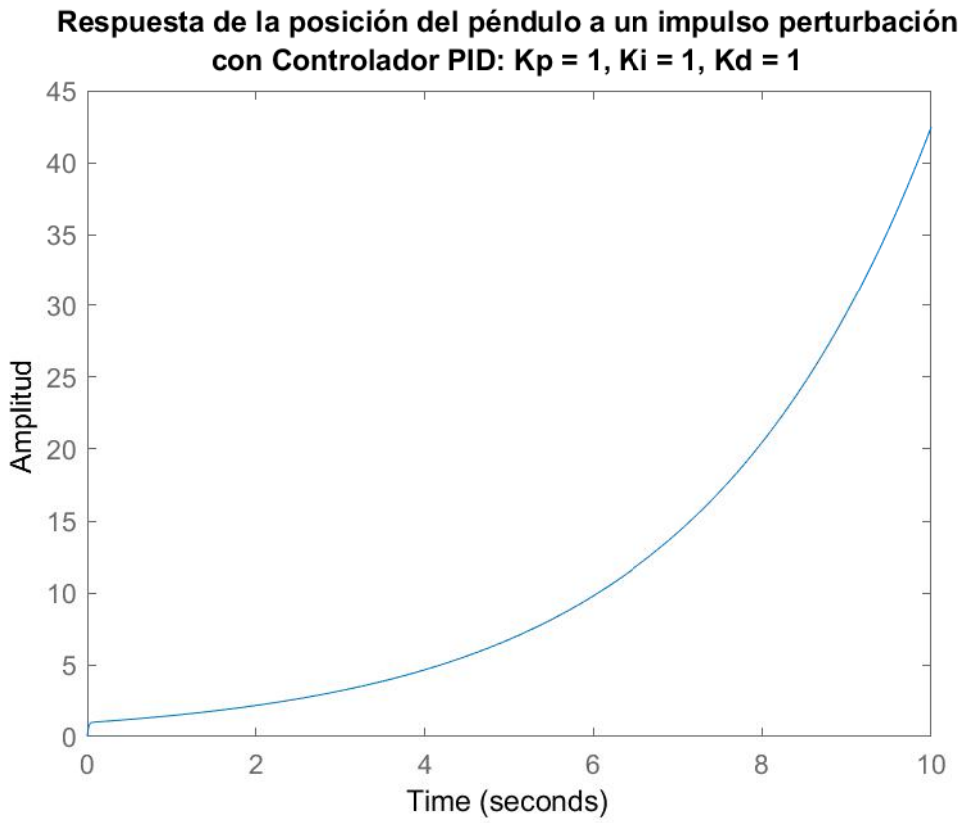

Figure 22: Respuesta de la posición del péndulo a una perturbación Kp = 1 Ki = 1 Kd = 1
Fuente: Autoría propia

La constante integral del controlador PID está pensada para actuar a largo plazo dentro de un sistema de control, es la última en actuar si se le ve de esa manera ya que su acción sobre el sistema es corregir estados estacionarios del sistema una vez se alcance.

En la Figura 24 se observa la corrección de las oscilaciones iniciales no deseadas del sistema. Esta corrección de las oscilaciones ocurre gracias al aumento de la constante derivativa Kd.

Una vez implementado el controlador PID para el sistema en posición de equilibrio del péndulo, éste nos da la garantía de funcionamiento para un sistema completo de péndulo rotacional invertido, estrictamente hablando para la posición de equilibrio de nuestro sistema. Para el balanceo y erguimiento del sistema de emplea otro método a saber el de Swing Up del inglés balanceo, aunque si se traduce palabra a palabra sería algo así como balanceo hacia arriba, algo que le da aún más sentido a la acción que cumple este controlador.

La perturbación sigue estando presente, pero ésta se hace algo insignificante frente al controlador PID, dando sentido a su ejercicio, su utilización o su trabajo.

Esta forma de implementación de un controlador PID es algo empírica, incluso se puede decir que se obtienen respuestas satisfactorias en poco tiempo y se ahorran análisis o cálculos

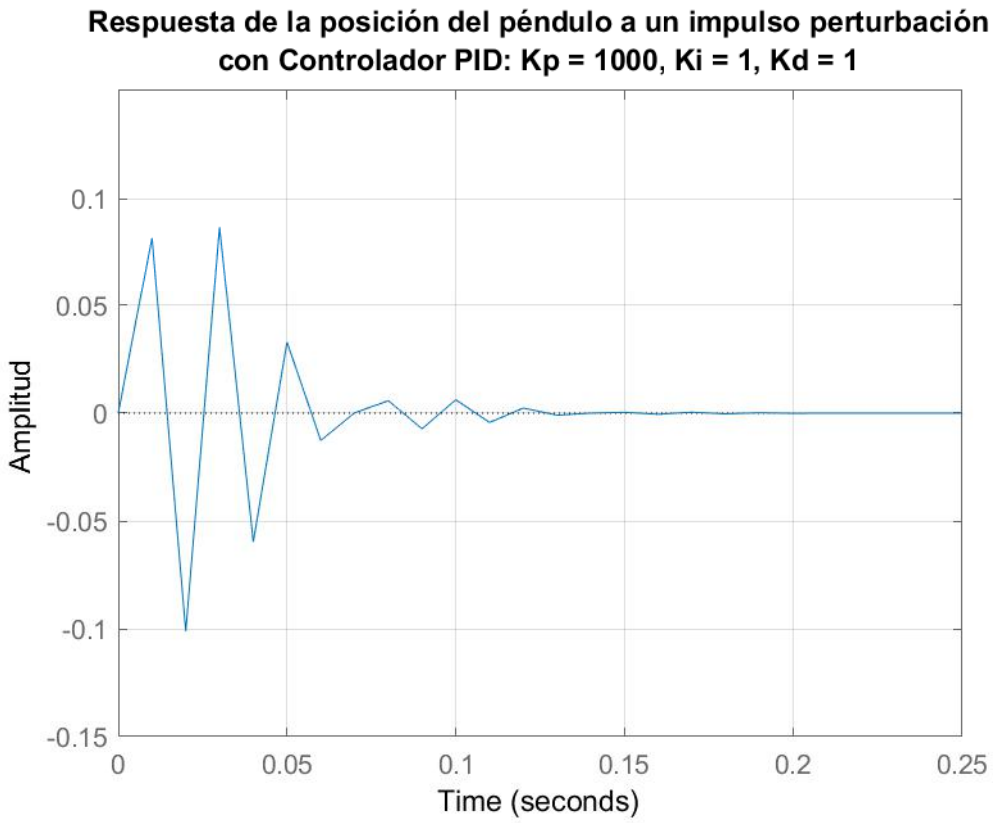

Figure 23: Respuesta de la posición del péndulo a una perturbación Kp = 1000 Ki = 1 Kd = 1
Fuente: Autoría propia

al momento de implementarse, es necesario entender que esto funciona para sistemas altamente lineales o para este caso sistemas controlados ya que no es el sistema completo en sí, más bien es una aproximación a una sección de este sistema pero que nos ayuda a entenderlo mejor. Hay más maneras o formas de llegar a analizar o implementar controladores PID en un sistema, ya sea por medio de análisis del lugar de las raíces, por medio de análisis en frecuencia o análisis en el espacio de estados, pero eso es tema para otro artículo. Es necesario entender el funcionamiento que hay detrás de un controlador tan importante como lo es un controlador PID, aunque si se compara con el sistema que se está estudiando a saber el péndulo rotacional invertido parece que queda pequeño nuestro controlador PID ante el sistema. No hay que despreciar el amplio y vasto potencial de un controlador PID.

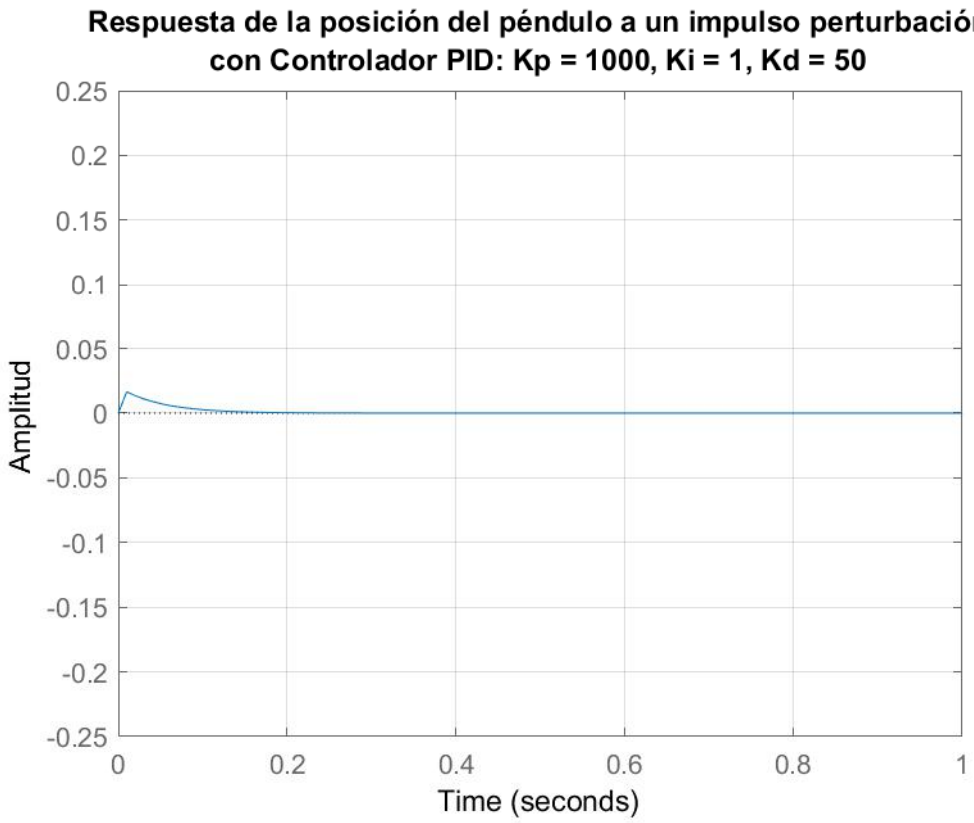

Figure 24: Respuesta de la posición del péndulo a una perturbación Kp = 1000 Ki = 1 Kd = 50

Fuente: Autoría propia

4.4 Control Difuso

Cambiando la dinámica tratada para el diseño de un controlador PID, se procede a diseñar el controlador difuso, esto nos indica que se puede configurar un controlador difuso basándonos solo en la experiencia que se tiene del sistema.

Inicialmente se define el tipo de controlador difuso a trabajar, para este caso se usa el controlador difuso de tipo Mamdami, según el trabajo de Buitrago et al. (2013) y Hernández et al. (2013) los controladores de este tipo se definen por poseer a la entrada de este un error, esta entrada nos permiten obtener datos del comportamiento del sistema y así poder lograr controlar de una manera óptima para la planta.

Se inicia con la ayuda de la toolbox de MatLab Fuzzy Logic Toolbox, se ejecuta con la palabra clave *fuzzyLogicDesigner* o simplemente la palabra *fuzzy* y se configura para nosotros tener dos entradas al controlador difuso, posterior a ello se asegura que se trata de un controlador difuso de tipo mamdani y que el sistema sea de una sola salida.

En la Figura 25, se establecen unos parámetros iniciales antes de entrar a la configuración

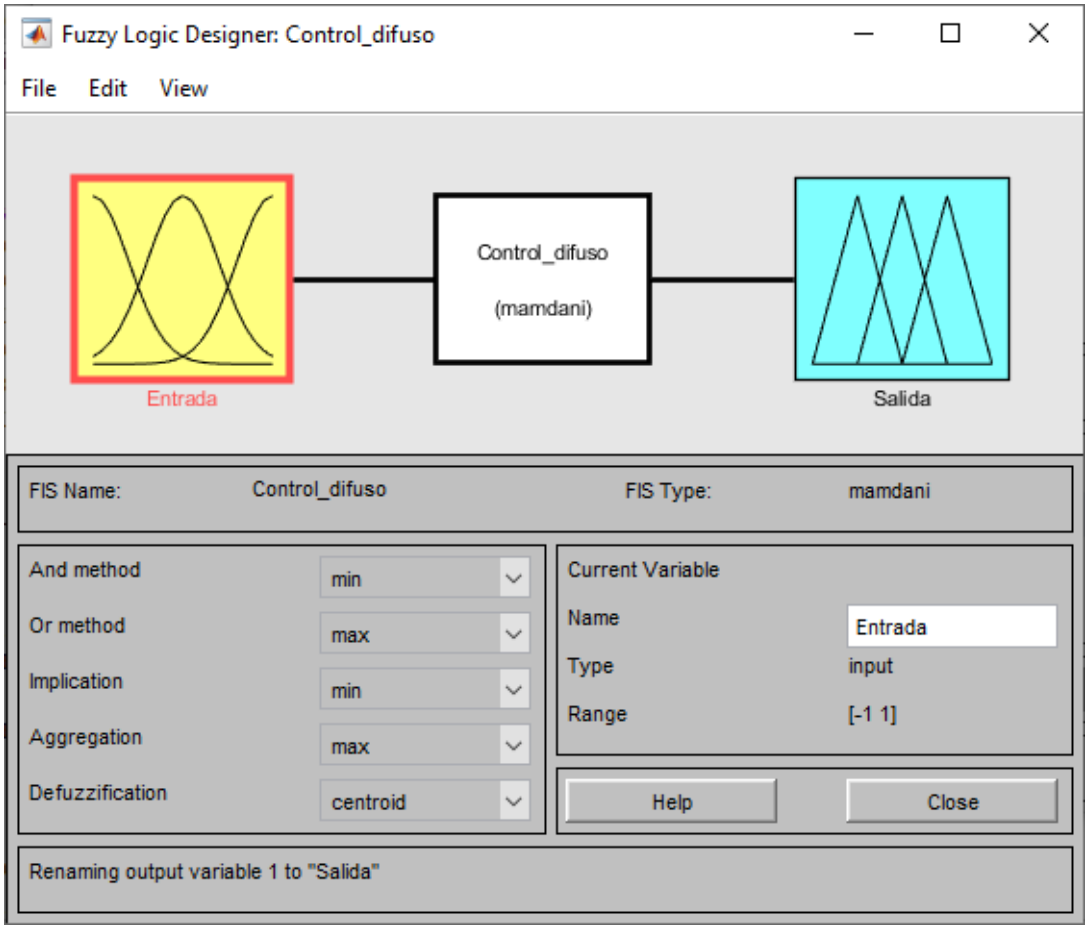

Figure 25: Control Fuzzy toolbox
Fuente: Autoría Propia

de las reglas y los métodos para diseñar el controlador, se establecen las dos entradas como se mencionó anteriormente, la salida única y la salida como de tipo centroide. Lo demás se deja tal cual como se encontró.

La persona que diseña este controlador difuso conoce por experiencia las variables y los valores de la entrada al sistema, por poner un ejemplo si se tratase de un sistema de control de temperatura el cual oscila entre 0 y 100 grados celcius y su voltaje de operación es de 0 a 24 voltios positivos, el sistema del controlador difuso ha de ser diseñado para este sistema de control de temperatura. En palabras simples a la entrada del controlador difuso se diseña para un rango de 0 a 100 grados celcius y a su salida para tratar valores entre 0 y 24 voltios positivos.

La Figura 26, muestra cómo se compone este sistema controlador difuso en su entrada, en el centro las funciones de membresía se componen de tres funciones triangulares, a los extremos se tienen dos funciones trapezoidales. El rango de este sistema difuso es importantes ya como se explicó anteriormente es el error que va a entrar al controlador difuso.

Por otro lado en la Figura 27, vemos como están compuestas la funciones de membresía

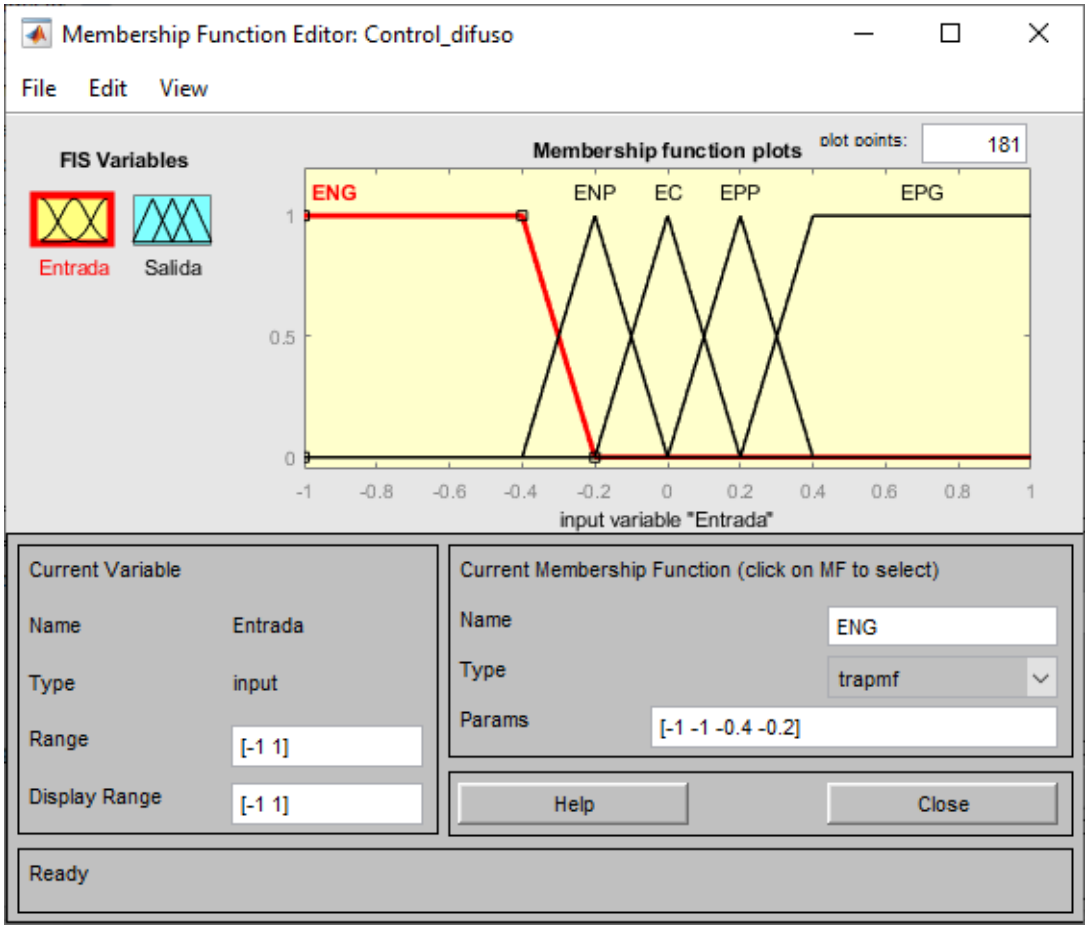

Figure 26: Funciones de membresía entrada
Fuente: Autoría Propia

para este sistema difuso las cuales no difieren mucho de las que están en la Figura 26, excepto por su rango el cual va de menos dos mil a dos mil con este valor aseguramos que el sistema actué de manera rápida ante los pequeños cambios que ocurren dentro del balanceo del péndulo erguido, es decir en posición de equilibrio.

Una vez establecidas las funciones de membresía para nuestro controlador difuso, se pasa a establecer, definir las reglas que componen a este sistema controlador difuso.

Las reglas del controlador difuso son el mecanismo de inferencia el cual permite asociar las funciones de membresía de entrada con las de salida. Es importante establecer la forma en la que estos dos grupos se comunican ya que por medio de esta relación es que el controlador difuso es capaz de cumplir su función de controlar valgase la redundancia en la oración.

Se resume la definición de las reglas en la Tabla 3, las cuales coinciden con la abreviación de las figuras anteriores. No obstante la reglas se definen de la siguiente manera:

Si el error es negativo grande, entonces la salida es negativa grande.

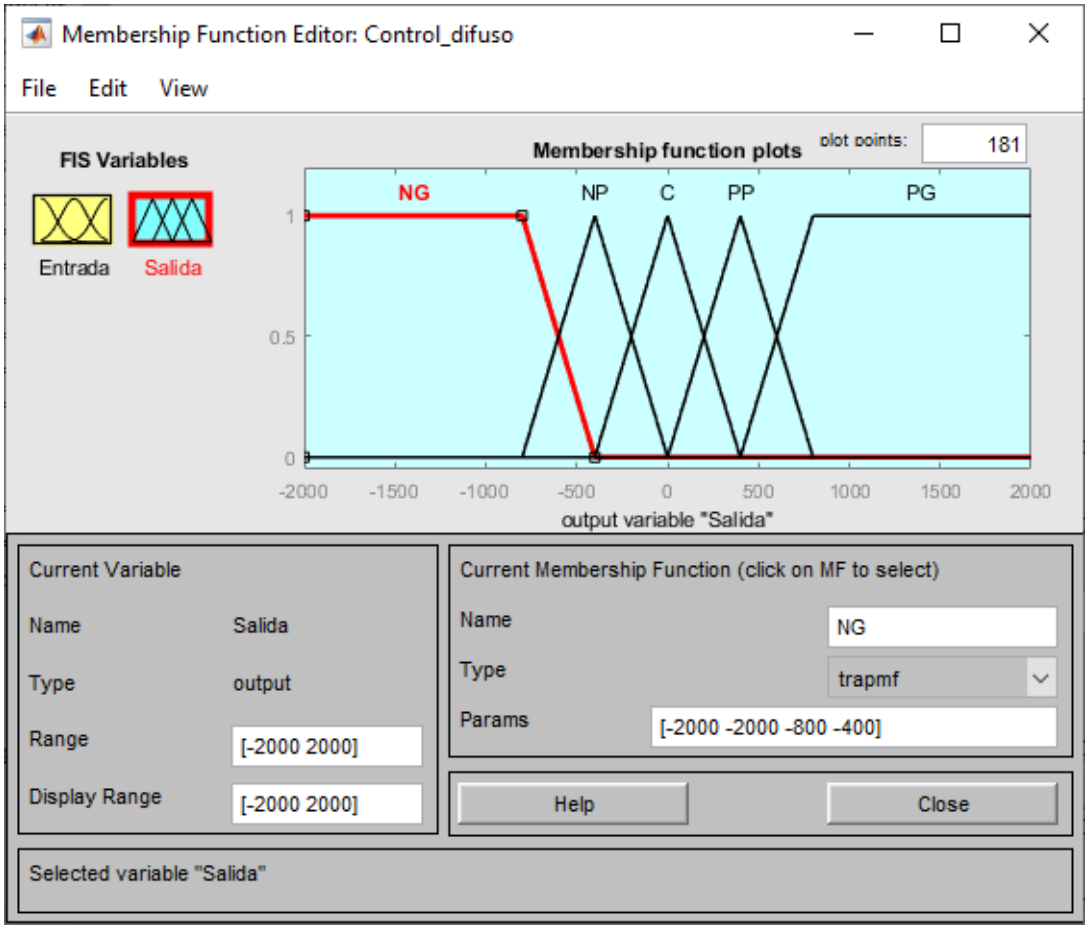

Figure 27: Funciones de membresía salida
Fuente: Autoría Propia

Si el error es negativo pequeño, entonces la salida es negativa pequeña.

Si el error es cero, entonces la salida es cero.

Si el error es positivo pequeño, entonces la salida es positiva pequeña.

Si el error es positivo grande, entonces la salida es positva grande.

Entrada	Salida
ENG	NG
ENP	NP
EC	C
EPP	PP
EPG	PG

Table 3: Tabla definición de las reglas de control

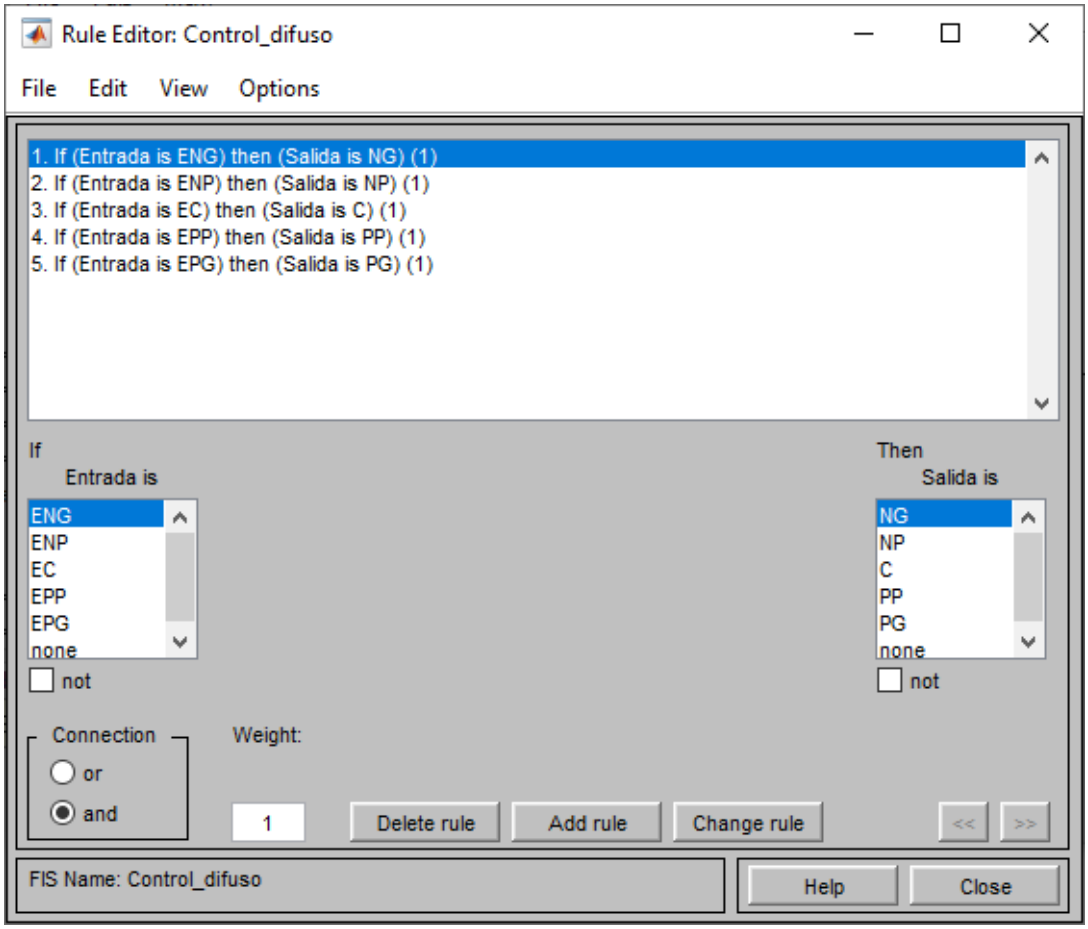

Figure 28: Reglas
Fuente: Autoría Propia

Como se observa en la Figura 28, la definición de las reglas nos dan a entender que existen cinco posibles para la entrada de que el error sea de alguna manera, la cual está relacionada por medio de una regla y ya que solamente se tiene una entrada con cinco posibles valores solo exiten cinco reglas o relaciones entre la entrada y la salida del sistema si hubiese otra entrada adicional esta multiplicaría el valor de las reglas para obtener las mayores combinaciones posibles. No es el caso así que se procede a mostrar la curva de control.

La curva de control mostrada en la Figura 29, como su nombre lo indica nos muestra la curva que se genera internamente dentro del controlador difuso y los valores correspondientes a los valores de entrada y salida que presencia el sistema difuso.

Se observa que en la Figura 29, en la parte superior de la ventana se alcanza a leer *Surface*, es decir superficie, esto es porque en ocasiones se necesitan dos entradas al sistema difuso y como resultado de las reglas establecidas entre las dos entradas y la salida de este sistema supuesto se genera una superficie de control, ya que para este sistema se define una única entrada y una única salida, nos da como resultado una gráfica en dos dimensiones y no en tres.

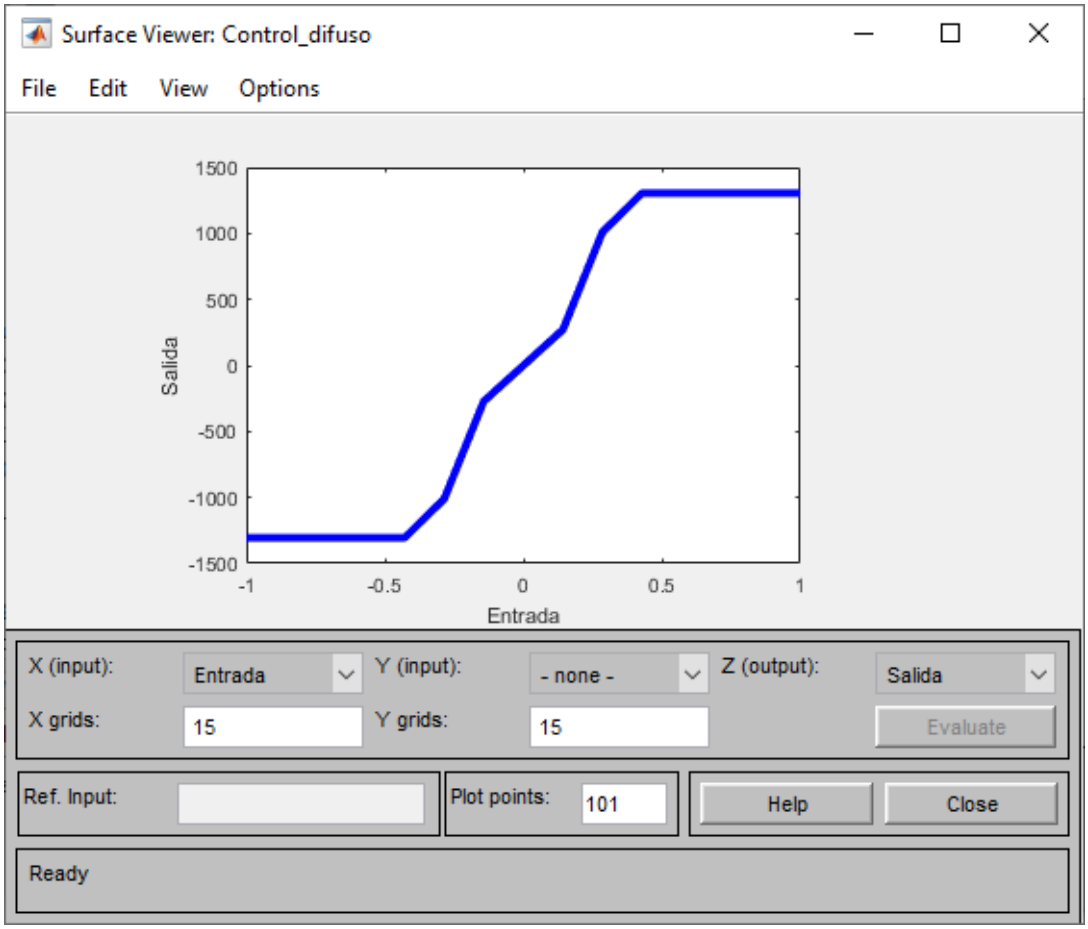

Figure 29: Curva de control
Fuente: Autoría Propia

Para la Figura 30, se observa como es que funciona el método de inferencia de Mamdani, ya que al ingresar un valor de entrada en esta ventana, se puede observar la salida que toma el sistema del controlador difuso y como se desplaza la barra vertical roja sobre las funciones de membresía.

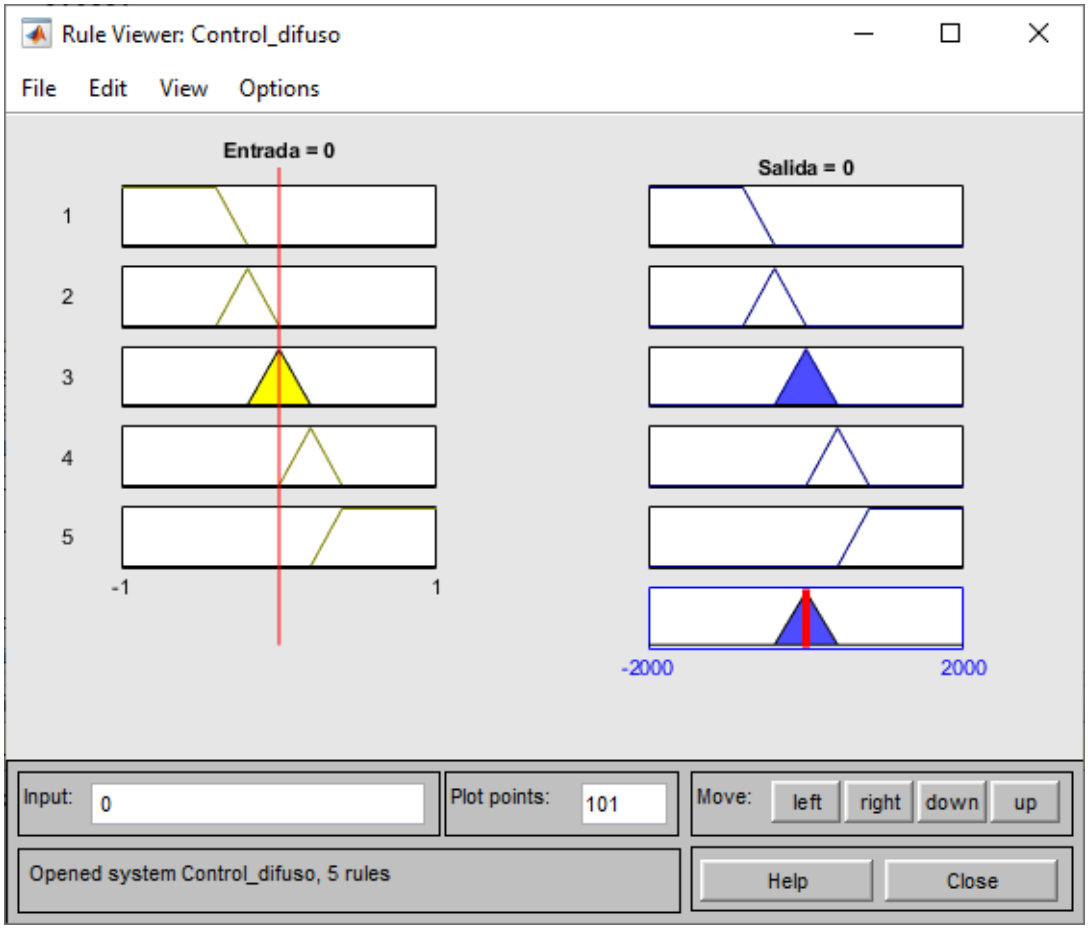

Figure 30: Reglas de control
Fuente: Autoría Propia

4.5 Control RNA

A continuación se presenta como se elabora a partir de un algoritmo corto y sencillo de entender una red neuronal capaz de emular la tarea del controlador Swing up para un péndulo rotacional invertido. Es importante y necesario entender que una red neuronal no es mágia, es un algoritmo de aproximaciones el cual permite imitar cualquier otro sistema con un modelo de caja negra si se sabe emplear.

En la Figura 31, se observa el algoritmo usado para crear la red neuronal usada como controlador, este consta de dos entradas a la red, una salida y cinco capas ocultas. Al final del algoritmo este genera un bloque que se usa para simulink el cual será de gran utilidad en el siguiente capítulo.

La función clave dentro del algoritmo de la Figura 31, es *newff* que consta de una función para la creación de una red neuroral artificial de tipo retropropagación de retroalimentación.

La arquitectura para esta red se resume en la Figura 32 se resumen en una capa de red

```
 % Programa Red neuroral Swing Up

% Entrada a la red neuronal

 entrada = [entrada_1'; entrada_2];

% Salida o target de la RNA

salida = salida_1';


Red=newff(minmax(entrada),[1 5 1],{'logsig' 'logsig' 'purelin'});
Red.trainparam.epochs=100000;

Red.trainparam.goal=1e-10;

Red=train(Red, entrada, salida);

YY = sim(Red, entrada);

gensim(Red);


```

Figure 31: Algoritmo Red Neuronal Artificial

neuronal con una función de activación de tipo sigmoidal, unas cinco capas ocultas con funciones de activación también de tipo sigmoide y una capa a la salida con una función de activación de tipo lineal. Cada una de las capas se establecen de forma arbitraria, normalmente no existen reglas para establecer el número de capas al momento de crear una red neuronal artificial.

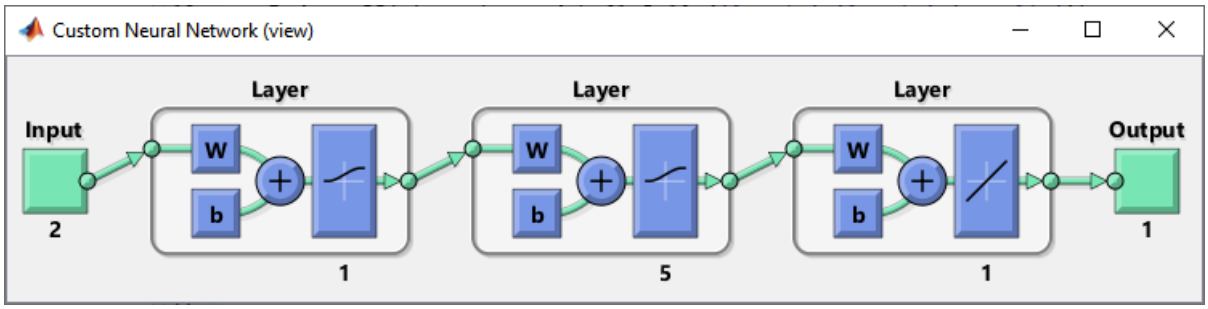

Figure 32: Arquitectura Red Neuronal Artificial
Fuente: Autoría Propia

Sabiendo que existen diferentes formas de conformar o crear una red neuronal artificial, se usan las configuraciones por defecto que traen las funciones de MatLab para la creación de este tipo de aplicaciones. En la Figura 33, se observan características de las configuraciones que traen estos algoritmos por defecto.

Inicialmente se muestra la arquitectura de la red, que es la misma de la Figura 32, luego en el apartado de *Algorithms*, se observan parámetros como *Training* o entrenamiento de la red neuronal artificial, en donde nos muestra qué tipo de configuración está utilizando el algoritmo, que para este caso es el Levenberg - Marquadt que no es más que un algoritmo de mínimos cuadrados amortiguados. Estos parámetros se pueden cambiar al momento de escribir nuestro algoritmo según sea la necesidad de su uso o qué tipo de configuración vaya a usar el usuario.

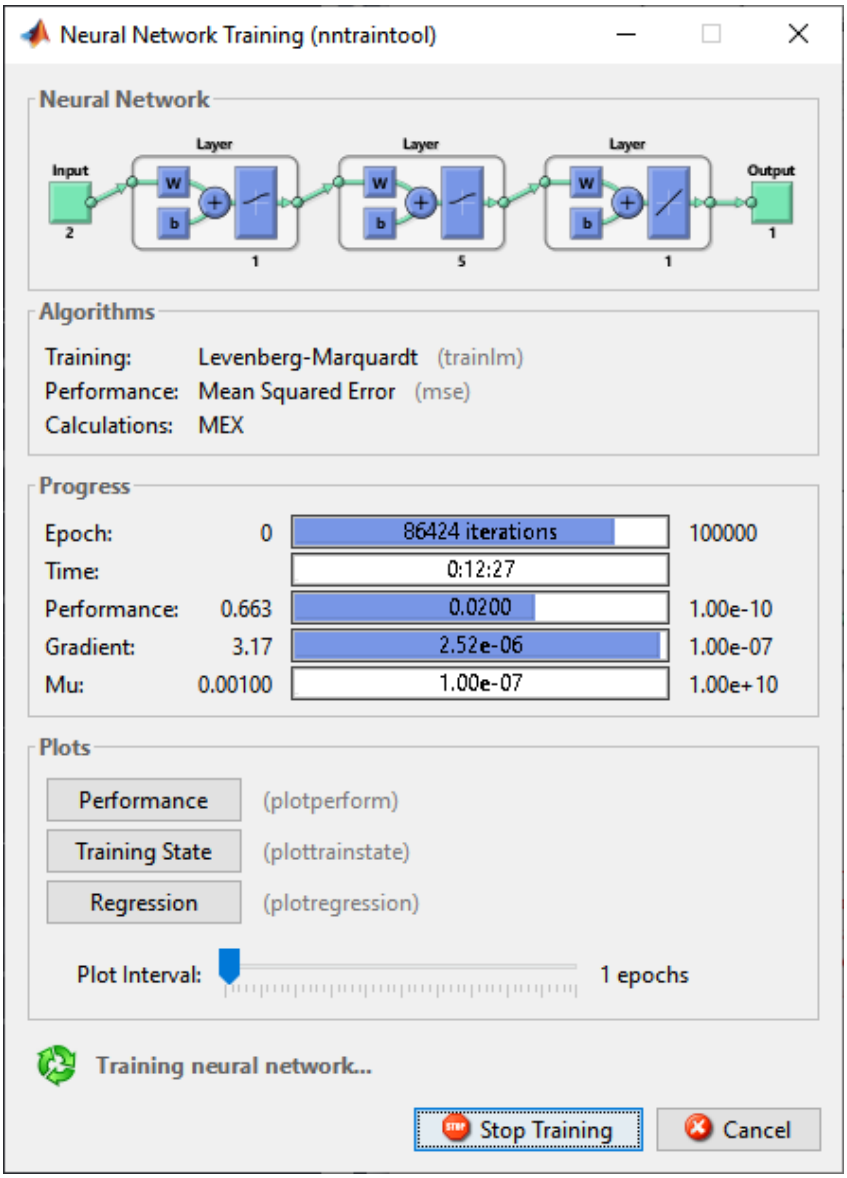

Figure 33: Configuración Red Neuronal Artificial
Fuente: Autoría Propia

En el apartado de *Progress* o progreso se muestran los avances que ocurren dentro del entrenamiento de la red neuronal artificial, el tiempo que transcurre entrenando a la red para llegue a los valores deseados de gradiente o hasta que cumpla las épocas que tiene como límite

para realizar el entrenamiento.

La sección de *Plots* o gráficas muestran a través del tiempo el desempeño que tuvo el entrenamiento de la red neuronal artificial, este se puede configurar para que muestre el número de épocas y así poder visualizar unas gráficas no tan saturadas. Es necesario recordar que no se debe esperar a que se completen la cantidad de épocas que tiene el sistema como límite para que la red neuronal quede bien entrenada valga la redundancia.

4.6 Síntesis

La elaboración o el diseño de estos sistemas de control difieren al momento de establecerlos para cualquier tipo de sistema y claro esta, también difieren entre sí, Un controlador Swing Up como se ha llamado en esta sección está necesariamente diseñado para esta aplicación en específico, ya que si se sale de los parámetros de su diseño, este controlador no tendría ninguna utilidad. Esto nos lleva a establecer que la utilidad de un controlador está limitada a su uso adecuado y dentro de unos parámetros o límites establecidos.

Del controlador PID, Fuzzy que son ampliamente usados para diferentes tipos de aplicaciones se pueden escribir libros enteros de su utilidad dentro de lo establecido por ellos mismos. Un controlador PID bien ejecutado resulta en un sistema muy robusto para su uso. Al igual que un controlador difuso bien definido, nos permite confiar en su ejecución dentro de un sistema a controlar.

Con respecto a la Red Neuronal Artificial, aunque no son nada nuevo ya que su creación data de los años 60s, es importante estudiarlas a profundidad y realizar una ejecución correcta de la misma.

5 Simulación

5.1 Introducción

La plataforma en la cual se desarrollará el respectivo modelo matemático del péndulo de furuta es una toolbox de MatLab de nombre Simulink. Es muy común usar esta toolbox para simular el comportamiento de sistemas dinámicos, sistemas de control entre diferentes tipos de sistemas.Puede simular sistemas lineales y no lineales, modelos en tiempo continuo y tiempo discreto y sistemas híbridos de todos los anteriores.

A continuación se presenta de manera gráfica el uso que se dió a las ecuaciones matemáticas establecidas en las secciones anteriores, los resultados de estas simulaciones se presentan en el capítulo de resultados obtenidos.

5.2 Desarrollo de la Simulación

5.2.1 Sistema Péndulo Rotacional Invertido

A partir de unas ecuaciones se puede obtener un modelo matemático que sea capaz de emular o asemejarse al comportamiento del sistema propiamente descrito con dichas ecuaciones, dependiendo de la complejidad del mismo se obtiene un sistema difícil de leer a simple vista tratándose de diagramas de bloques.

La idea con el sistema mostrado en la Figura 34, es representar de la mejor forma posible el comportamiento dinámico del péndulo rotacional invertido, para ellos se usan operaciones básicas de adición, sustracción, multiplicación así como fuciones trigonométricas todo lo anterior interconectado para arrojar las ecuaciones descritas en Ecuación (52).

Teniendo en cuenta las ecuaciones generadas en (22) y (23), se elaboran los respectivos bloques a partir de estas. Lo primero es definir cada una de las constantes que hacen parte del sistema a modelar.

Posterior a la definición de las constantes se definen cada una de las ecuaciones diferenciales obtenidas.

Según las ecuaciones definidas en (21) y del ejemplo explicado en la sección anterior. Las ecuaciones normalizadas quedarían de la siguiente manera.

$$\begin{aligned}\ddot{\theta}_1 &= \left[\tau - \gamma\cos\theta_1\ddot{\theta}_1 - 2\beta\cos\theta_1\sin\theta_1\dot{\theta}_0\dot{\theta}_1 + \gamma\sin\theta_1\dot{\theta}_1^2\right]\left[\frac{1}{(\alpha+\beta\sin^2\theta_0)}\right], \\ \ddot{\theta}_0 &= \left[\beta\cos\theta_1\sin\theta_1\dot{\theta}_0^2 + \sigma\sin\theta_1 - \beta\ddot{\theta}_1\right]\left[\frac{1}{\gamma\cos\theta_1}\right]\end{aligned} \tag{52}$$

Partiendo de las ecuaciones descritas en (21), se crean las respectivas conexiones por bloques de funciones, las cuales se dividen en subsistemas, cada uno correspondiente a una ecuación de (22).

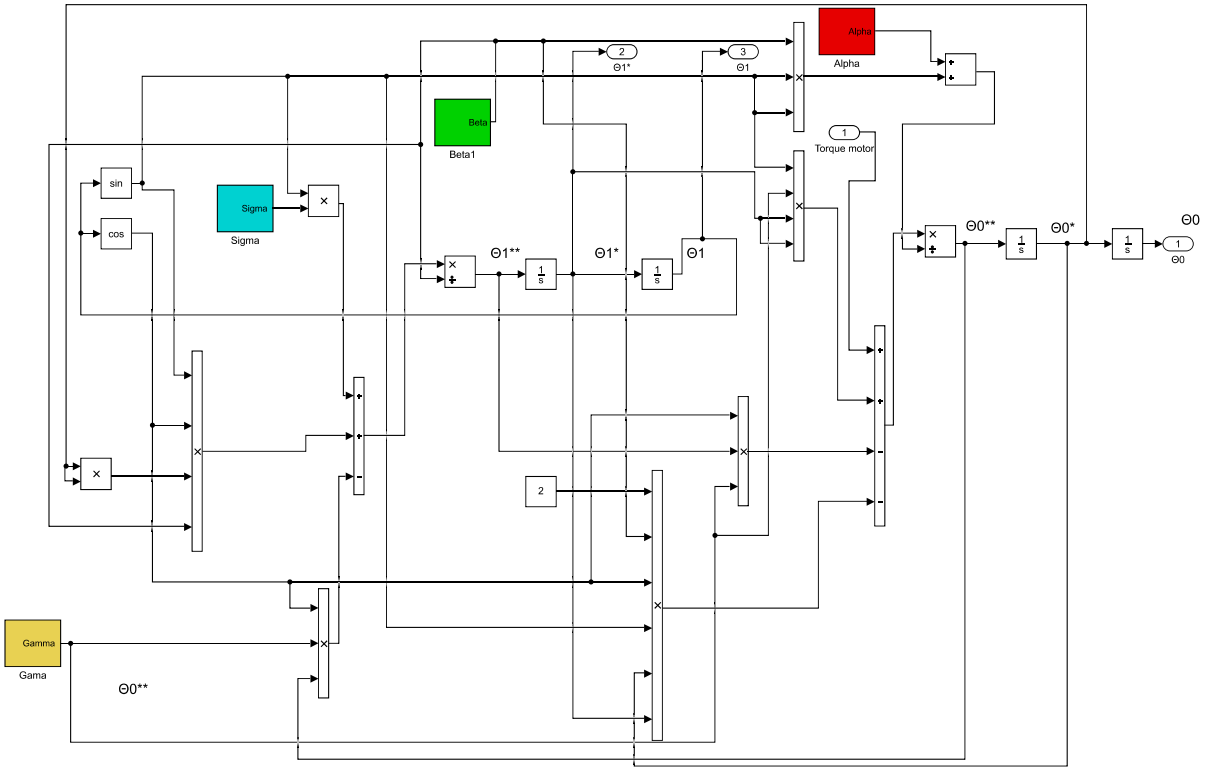

Figure 34: Diagrama completo de la ecuaciones dinámicas del sistema
Fuente: Autoría Propia

En la Figura 35 se muestra una ventana en la cual se rellenan estos datos de manera manual. Inicialmente se está en el espacio de trabajo de Simulink, se presiona click derecho sobre cualquier espacio en blanco y se selecciona en la parte inferior la opción *Model properties*, luego se hace click sobre *InitFcn* y en este apartado se colocan las constantes que se quieren que se inicialicen al principio de la simulación, sin la necesidad de ejecutarlas en el *workspace* de MatLab.

En las Figuras de la 36 a la 39, se observan como están compuestos internamente cada uno de los subsistemas correspondientes a cada una de sus variables asignadas.

Estos parámetros se obtienen de mediciones reales obtenidas de la medición directa del prototipo presentado en el Capítulo 6. La idea en un principio fue realizar emulaciones y simulaciones en Simulink que fueran fieles al modelo real, pero al no tener información suficiente obtenida del datasheet y de los sistemas implementados se opta por realizar aproximaciones.

Las constantes mostradas en la Figura 35 se definen como *mp* masa pendular o masa del péndulo, *ma* masa del brazo conectado al motor, *lp* longitud del péndulo, *la* longitud del brazo,

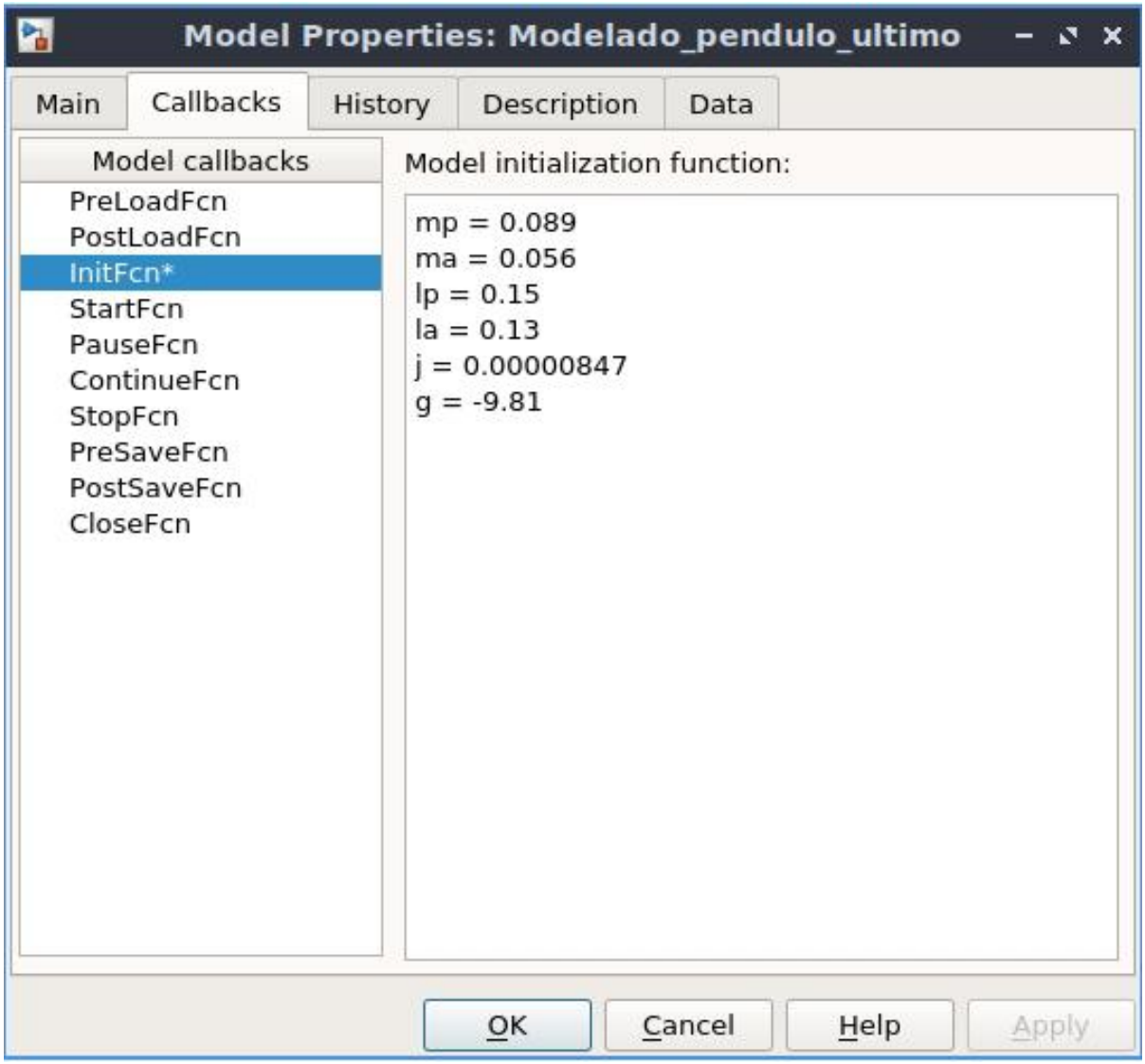

Figure 35: Constantes definidas al inicio de la simulación
Fuente: Autoría Propia

j constante inercial del sistema, g constante gravitacional terrestre.

Es importante ver que de estos sencillos valores se establecen para la simulación de un sistema sacado de un modelo real y funcional.

Lo importante al separar los bloques Alpha, Beta, Sigma y Gamma, es que inicialmente fueron parte de la ecuación que que describe su comportamiento dinámico y permite realizar dicho estudio del sistema. Más adelante se mostrará en una imagen la forma final de estos sistemas y de estos bloques funcionales.

La idea principal al momento de realizar conexiones entre estos bloques de funciones, a parte de representar la dinámica del sistema de péndulo rotacional invertido, fue la de realizar

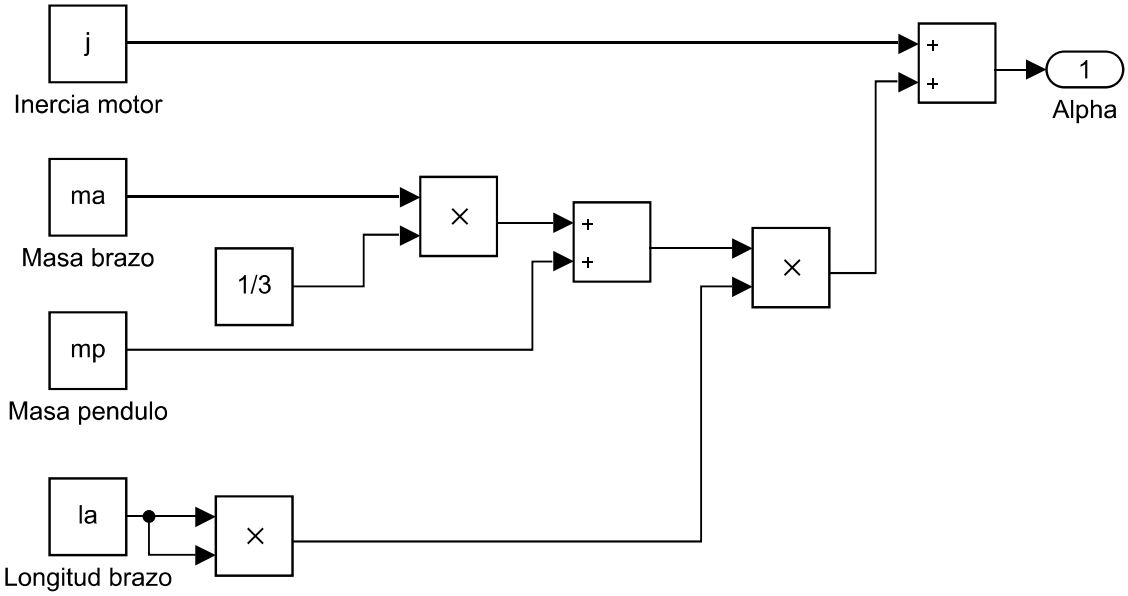

Figure 36: Diagrama interno del subsistema Alpha
Fuente: Autoría Propia

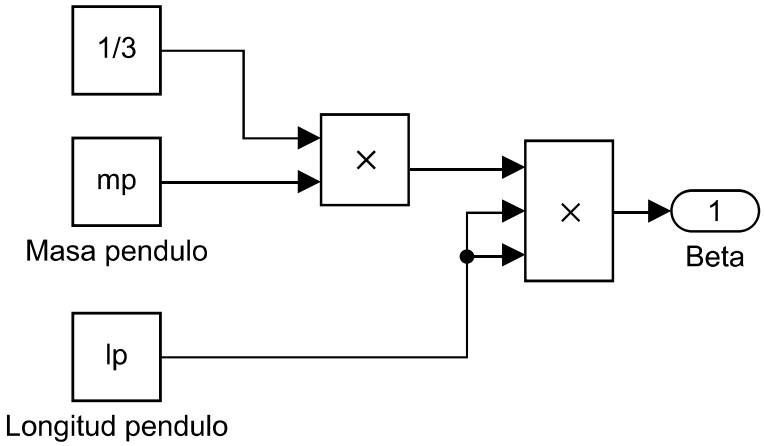

Figure 37: Diagrama interno del subsistema Beta
Fuente: Autoría Propia

una demostración visual, una simulación para mostrar el comportamiento de los controladores dentro de un sistema controlado y que fuera del agrado al momento de realizar una demostración

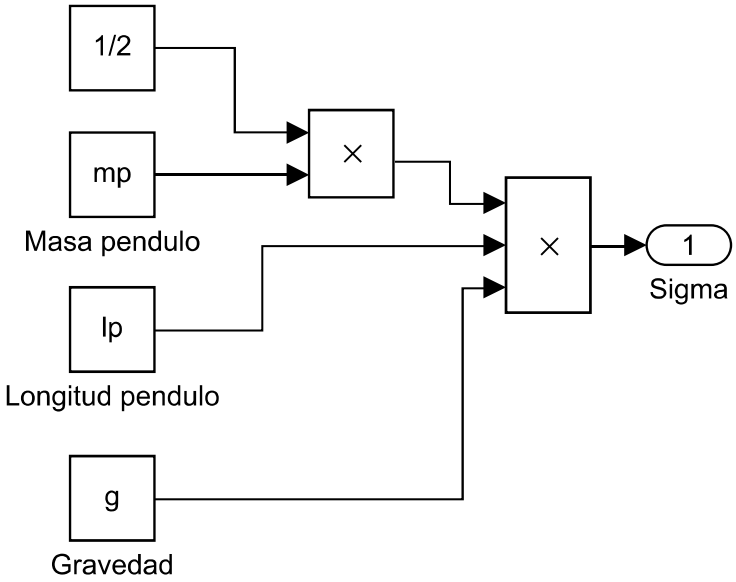

Figure 38: Diagrama interno del subsistema Sigma
Fuente: Autoría Propia

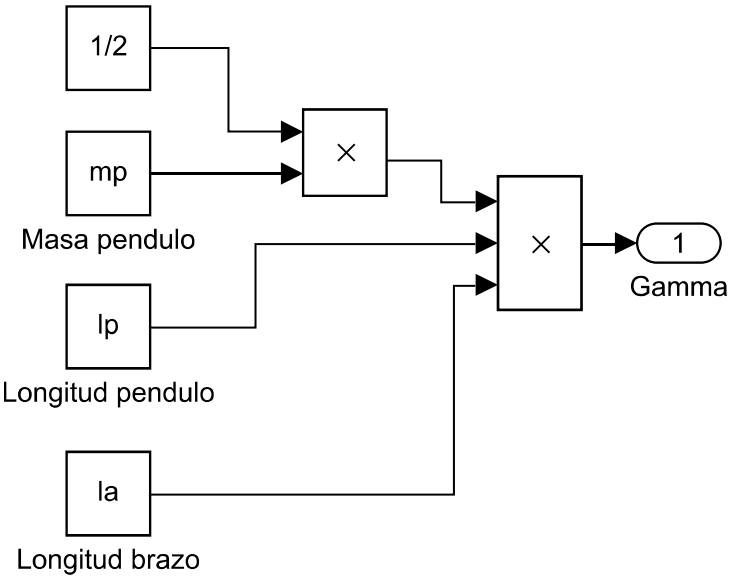

Figure 39: Diagrama interno del subsistema Gamma
Fuente: Autoría Propia

visual. Si se piensa por un momento los valores de alpha, beta, sigma y gamma no son más

que operaciones de adición y multiplicación entre las constantes inicialmente establecidas en la Figura 35.

5.2.2 Sistema Motor DC

Igualmente para la simulación del modelo del motor DC se utilizan los siguientes parámetros sacados de Bill Messner (2021). Estos parámetros como son resistencia e inductancia de armadura del motor, inercia y coeficiente de fricción son un poco de complicados de obtener de algún datasheet de un motor comercial, normalmente encontramos otros parámetros como son rpm, torque, voltaje de trabajo y corriente nominal, por ello se opta por usar unos parámetros de otro trabajo académico.

En la Tabla 2 que se observa a continuación se ven las constantes eléctricas y mecánicas de un motor DC para la simulación del mismo. Con este modelado se pretende que la variable de entrada del sistema de péndulo rotacional invertido sea el voltaje del motor DC, más no su torque. La variable de salida del sistema sigue siendo el ángulo de rotación del péndulo rotacional invertido.

Constante	Valor	Descripción
R	4Ω	Resistencia de armadura en Ohmnios
L_a	$2.75 * 10^{-6} H$	Inductancia armadura en Henrios
J_m	$3.2284 * 10^{-6} Kg.m^2$	M. inercia en Kilogramo-Metro cuadrado
B	$3.5077 * 10^{-6} N.m.s$	C. de fricción en Newton-Metro-Segundo
K_a	$0.0274 V/rad/sec$	Relación Voltaje-Velocidad Angular
K_m	$0.0274 N.m/A$	Constante Torque Motor

Table 4: Constantes motor eléctrico

A continuación se muestran los bloques internos que representan matemáticamente a un motor DC.

En la Figura 40 se puede observar un bloque que representa matemáticamente las ecuaciones diferenciales que describen el comportamiento del motor DC modelado para la simulación

A partir de aquí la idea principal es tener como entrada el voltaje del motor de la base del péndulo rotacional invertido. Y a la salida la posición angular del péndulo. Esta es básicamente la composición de todo el sistema si éste se tratase como un sistema de caja negra, entrada voltaje del motor y salida la posición angular del péndulo, aunque se sabe que el sistema a controlar es más complejo que eso.

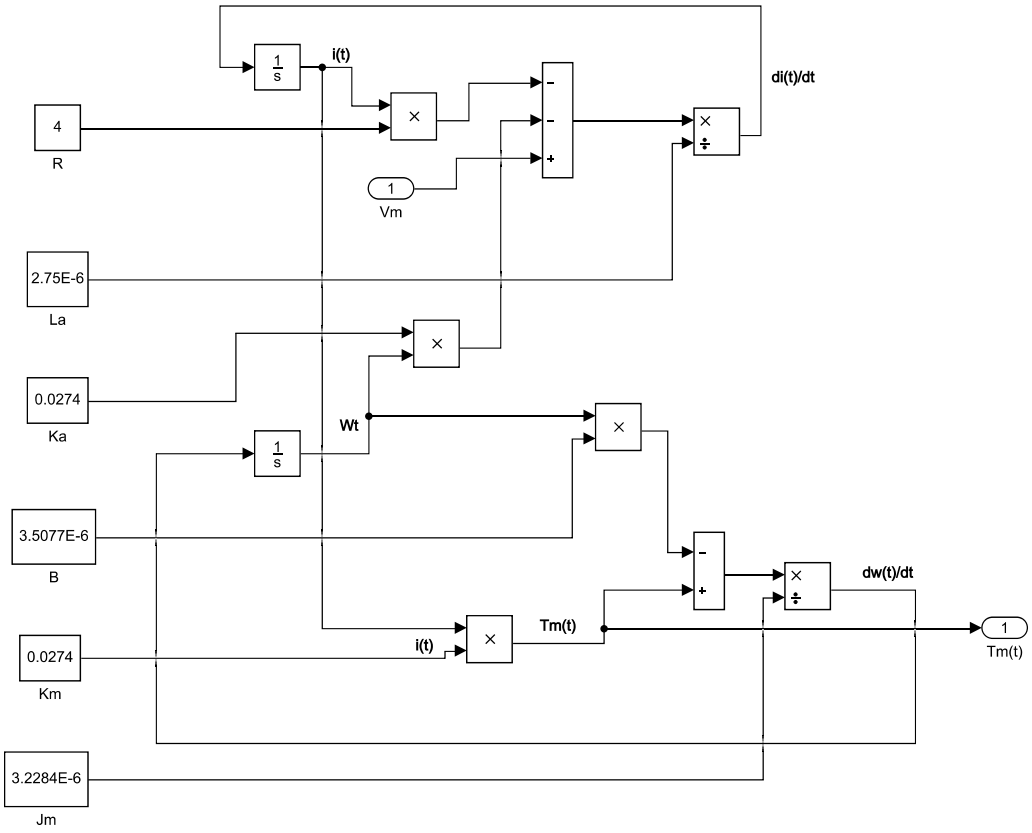

Figure 40: Bloque motor DC
Fuente: Autoría Propia

5.3 Creación de una Representación CAD del Péndulo de Furuta con Simulink 3D Animation

Matlab cuenta con un toolbox que permite diseñar modelos en 3D que se pueden colocar en Simulink y moverse en función de las señales que le llegan, para esto usamos el bloque V Sink situado en el grupo de Simulink 3D animation situado en la librería de Simulink.

Una vez ubicado el bloque V Sink y puesto en el area de trabajo de Simulink, se le da doble click para acceder a su configuración. Una vez allí le damos en el boton New. Nos aparecerá una ventana como la que se muestra en la Figura 41. Esta explicación está basada en el articulo de Regalo Núñez (2016).

Una vez allí lo siguiente es crear a partir de figuras geométricas simples como cilindros, esferas cada una de las partes que componen a nuestro péndulo de Furuta.

Al momento de construir cada una de las partes que componen al sistema se debe tener en cuenta aquellas que son solidarias o pertenecen a otra parte esta se debe añadir como children

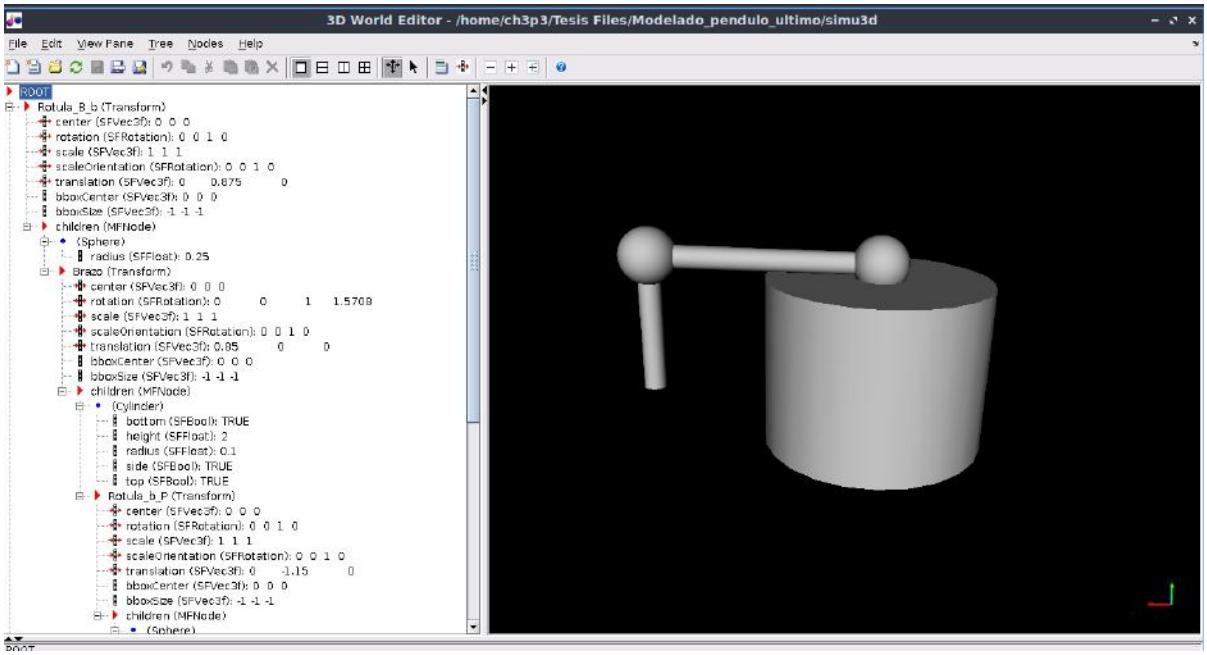

Figure 41: 3D World Editor
Fuente: Autoría Propia

del anterior en un diagrama de árbol que se va creando al costado izquierdo de la interfaz de la Figura 41.

En la Figura 41, se termina de enseñar cada uno de los componentes de el diagrama de árbol. Puede verse algo extenso pero en realidad no se usarán todas las componentes de dicho diagrama, solo algunas serán usadas para la simulación.

En la parte lateral izquierda se encuentran todos los parámetros de ubicación, tamaños y características que hacen posible el modelo 3D de la Figura 41 de la derecha.

Todo este entramado de datos, números y coordenadas empiezan por una base, la cual es un elemento fijo de todo el sistema, es decir no tiene movimiento al momento de la simulación. Esto nos da a entender que puede ser usado como punto de partida para todo el sistema, los demás elementos como las rótulas, el brazo y el péndulo son los que nos van a dar la representación del movimiento de todo el sistema.

Si se piensa detenidamente lo único que se mueven son las rótulas del sistema, el brazo y el péndulo están ligadas a estas, pensemos en las rótulas como los rodamientos del motor y del encoder que hacen parte de todo este conjunto de péndulo rotacional invertido.

En la Figura 42, se observan los parámetros y las características que definen a nuestro modelo 3D a utilizar dentro de la simulación realizada en Simulink y sus ecuaciones matemáticas y dinámicas descritas.

Al momento de ejecutar este sistema o modelo 3D dentro del entorno de Simulink se deben escoger los parámtremos controlador por variables externas a este modelo 3D, para este caso

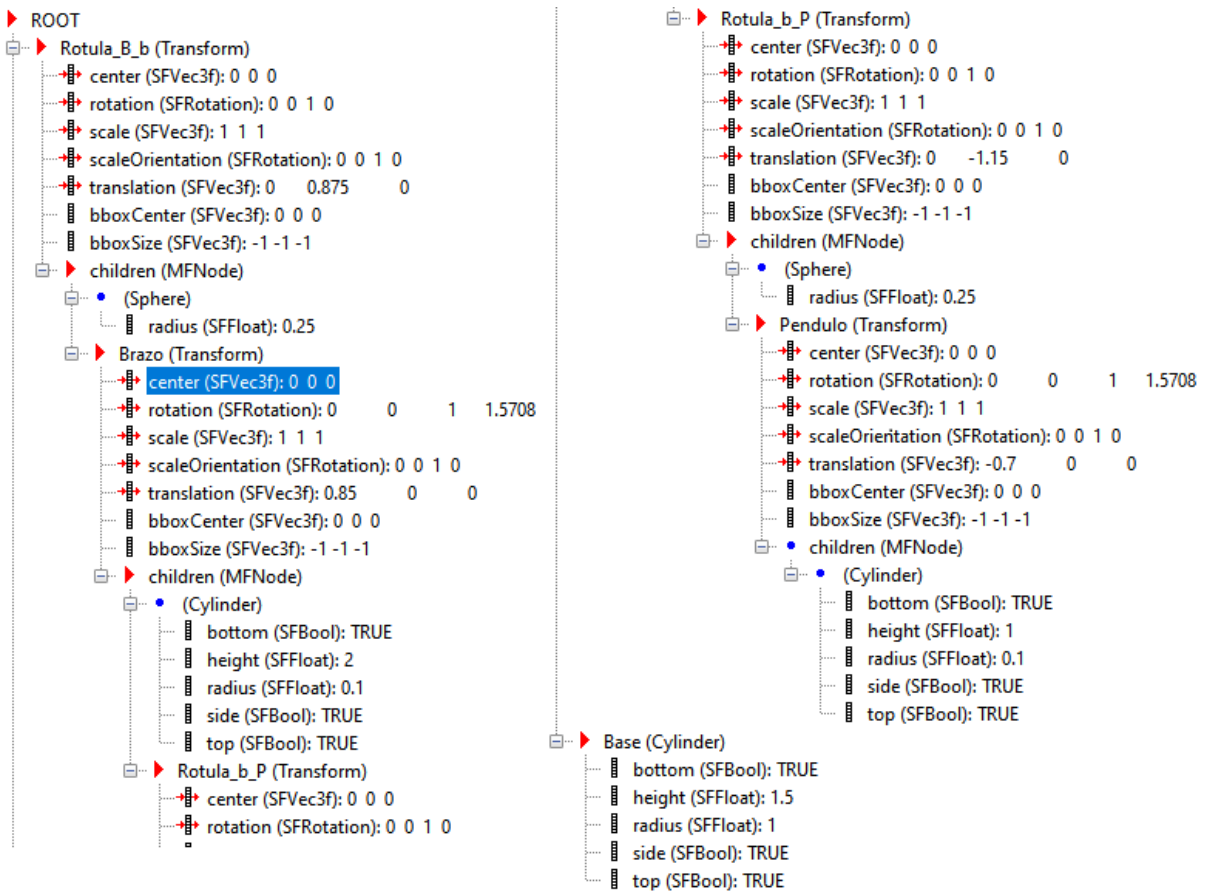

Figure 42: Parámetros y caracteristicas 3D
Fuente: Autoría Propia

serían las ecuaciones y sus valores dentro de las mismas como son torque del motor y desplazamiento angular del péndulo, por esto mismo y como se ve en la Figura 43, se deben seleccionar los valores adecuados para que el sistema se comporte acorde a lo establecido o deseado.

Dentro de esta misma Figura 43, se observan en toda la parte lateral derecha casillas de confirmación sin marcar y una que otra marcada, esto mismo es lo que comentaba anteriormente, se debe seleccionar el parámetro correcto a ser emulado por el sistema en el entorno de simulink, por ejemplo si se desea exclusivamente el movimiento del motor esta casilla correspondería a la rótula llamada *Rotula B b* que hace referencia a la Rótula Base del brazo.

Otros parámetros como el tiempo de muestreo nos permite para este caso ajustar la velocidad a la cual se muestra nuestra simulación dentro de Simulink aunque no siempre sea de esta manera.

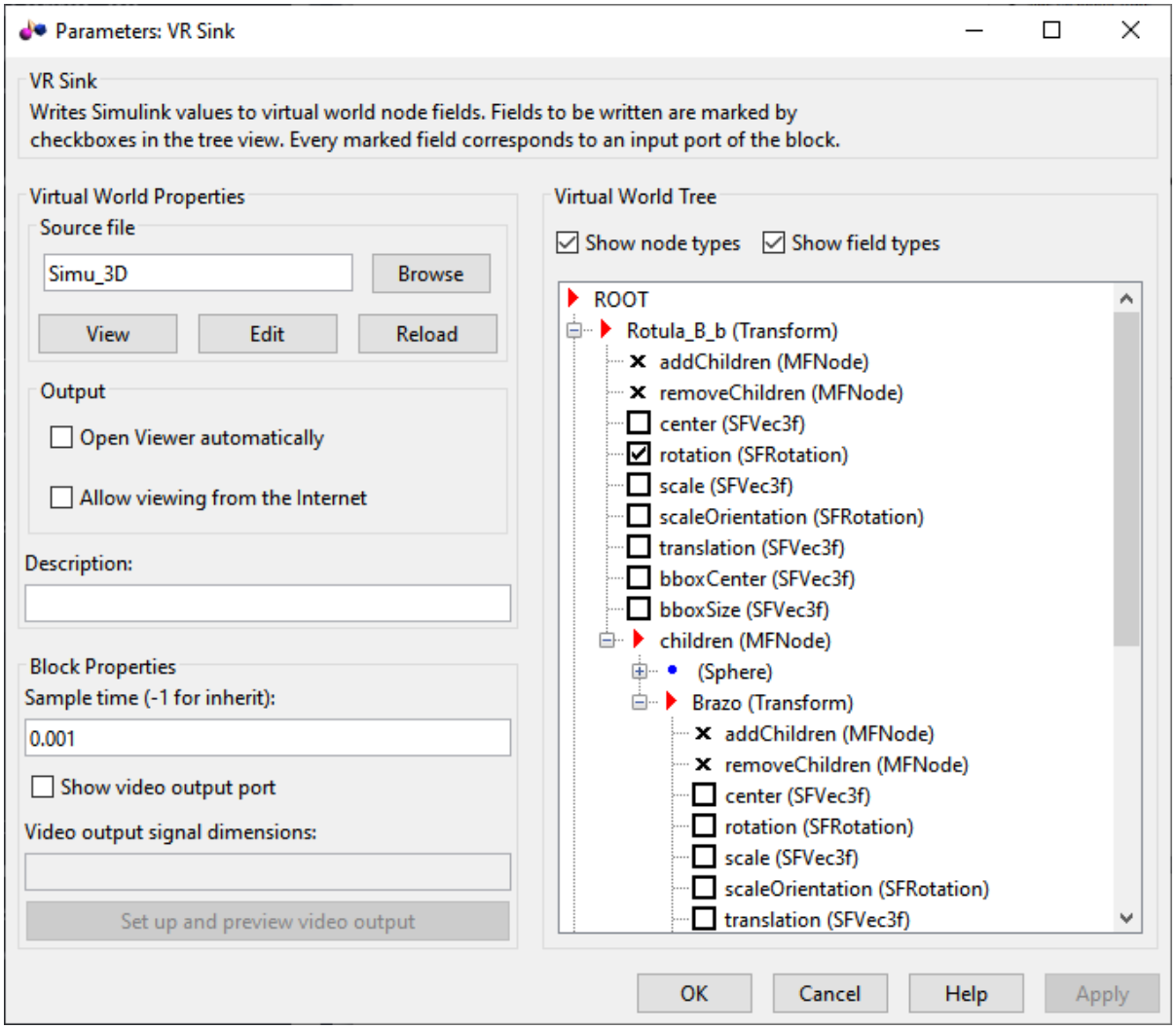

Figure 43: Parámetros VR Sink
Fuente: Autoría Propia

5.4 Síntesis

Se realiza una descripción del método usado en la elaboración de una herramienta gráfica que provee MatLab y con ayuda de Simulink para su ejecución. Esta herramienta nos permite enseñarnos de manera visual la forma de moverse de un sistema en particular, dandonos una mejor perspectiva que si solo se analizaran gráficas con respecto al tiempo.

6 Desarrollo y Construcción

6.1 Introducción

Existen infinitas formas en las cuales se puede llegar a construir un péndulo invertido rotacional. A continuación se presenta la experiencia y las formas las cuales llevaron a la construcción de un prototipo de péndulo rotacional invertido. También se mostrarán tanto los materiales como los componentes que hicieron parte de este sistema.

6.2 Péndulo Rotacional Invertido

Su composición más simple consta de un motor en corriente continua y un sensor para la posición angular del péndulo. Lo anterior es para la parte mecánica y solo se precisa de un actuador y un sensor. A grandes rasgos la parte electrónica consta de un microcontrolador, un puente H o driver para el motor DC y una fuente que alimente todo lo anterior. El resto de sus partes constan de piezas impresas en 3D que acoplan todo en su lugar, tornillos, tuercas, una base de madera, un rodamiento y una varilla hueca de aluminio para el péndulo.

La idea principal que lleva la construcción de un péndulo rotacional invertido, es crear un sistema el cuál sea totalmente replicable y económicamente accesible para cualquier persona que desee construir uno similar o con algunas variaciones.

6.2.1 Partes Móviles

Existen solamente dos partes móviles en este sistema de péndulo invertido rotacional, que son el motor conectado a la base y el encoder que está conectado directamente al péndulo y estos giran sobre su propio eje, es decir, el motor anteriormente mencionado.

El encoder utilizado se puede apreciar en la Figura 44, es uno de tipo rotativo el cual posee una resolución de 360 pulsos por cada giro de 360º. Se escogió este tipo de sensor ya que posee una muy buena resolución al momento de tomar datos y no se limita en cuanto al número de giros en comparación a otro tipo de sensores usados para medir ángulo como lo son potenciometros lineales.

Algunas características del encoder son:

- Voltaje de trabajo 5-24 V DC
- Tamaño del Eje 6 x 13 mm
- Tamaño del encoder: 38 x 35.5 mm
- Salida: circuito de impulsos ortogonales rectangulares de salida AB 2fase, la salida es NPN

- Velocidad mecánica máxima: 5000 rpm
- Frecuencia de respuesta: 0-20 kHz

Figure 44: Encoder óptico rotativo incremental

Figure 45: Motor DC 24V

El actuador utilizado se puede apreciar en la Figura 45, es un motor de 24 V DC de la marca japonesa de motores NISSEI DENKI el cual proporciona las condiciones ideales para poner en

funcionamiento el péndulo de furuta.

Lamentablemente no se consigue información detallada a cerca de las características técnicas de este dispositivo, salvo por algunas características básicas que proporciona el vendedor al momento de comprar dicho motor.

Estas son algunas características del motor:

- Voltaje nominal 22.5 V DC
- Corriente nominal 100 mA
- Velocidad nominal 2600 rpm
- Tamaño del Eje 2.5 x 9.4 mm
- Tamaño del motor: 50.2 x 30.7 mm

6.2.2 Partes Electrónicas

La Figura 46 nos muestra el microcontrolador de la familia AVR el cual es indispensable para la conexión con simulink y la ejecución del código que controla el péndulo de furuta.

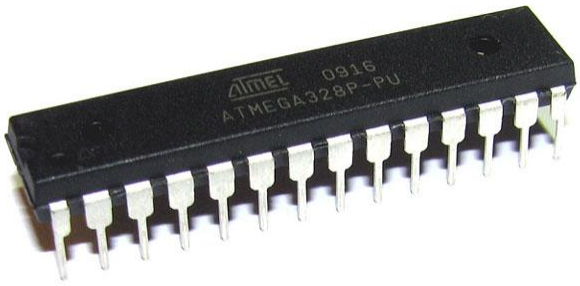

Figure 46: Microcontrolador Atmega 328p

Estas son algunas características técnicas del controlador:

- Voltaje de operación 1.8-5.5 v
- Flash 32 Kbytes
- SRAM 2 Kbytes
- Cantidad Pines 28
- Frecuencia máxima de operación 20 MHz
- Pines máximos de E/S 23

- Interrupciones externas 2
- Eeprom 1Kbytes
- Canales PWM 6

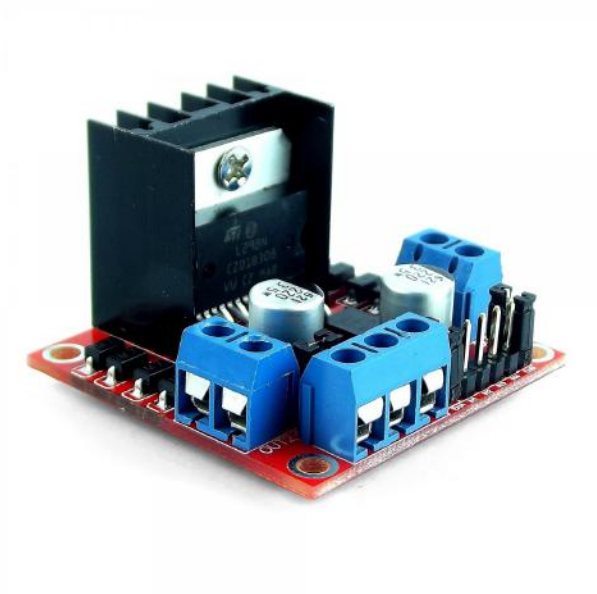

Figure 47: Modulo driver L298N

La parte de potencia que se encarga de controlar el motor es un módulo que incluye el driver L298N, Figura 47. El cuál se conforma de dos puente H capaces de permitir el control de dos motores simultáneamente. El puente H es un componente compuesto por 4 transistores, lo que nos permite invertir el sentido de la corriente y de esta forma revertir el sentido de rotación de la corriente en el motor.

El rango de voltaje de trabajo de este módulo es de 3 V a 35 V, y la intensidad de corriente es de hasta 2 A. Al suministrar energía al módulo, se debe considerar que los componentes electrónicos del módulo consumen alrededor de 3V de voltaje, por lo que el voltaje recibido por el motor es 3V menor que el voltaje cuando el módulo está en funcionamiento.

Estas son algunas características del módulo:

- Interfaz de potencia 7V-46V
- Corriente máxima 2A por canal
- Voltaje de control 5V
- Corriente de control 36mA
- Potencia de salida 25W

- Temperatura de operación -20°C hasta +135°C

Otros componentes electrónicos utilizados para este proyecto consisten en una fuente DC de 24V, un par de resistencias de 4,7 kΩ, las cuales se usan como resistencias pull - up al momento de conectar las fases de nuestro encoder óptico.

6.3 Diseño de una Shield para Arduino Uno

La palabra shield si se traduce literalmente del inglés hace referencia a escudo. En el argot electrónico una shield hace alusión a una tarjeta, circuito electrónico que se acopla o extiende a otro circuito normalmente asociado a arduino. Ya que su objetivo es darle una extensión en su funcionamiento. Es una extensión de su hardware el cual nos facilita conexiones o nos permite realizar tareas en específico que sin dicha shield sería más difícil su funcionamiento.

Para la creación de dicho circuito se utilizó el software libre Kicad el cual es un paquete para la automatización del diseño electrónico. Gracias a su facilidad de manejo y su multitud de herramientas es el programa de diseño electrónico ideal para el desarrollo de este tipo de circuitos.

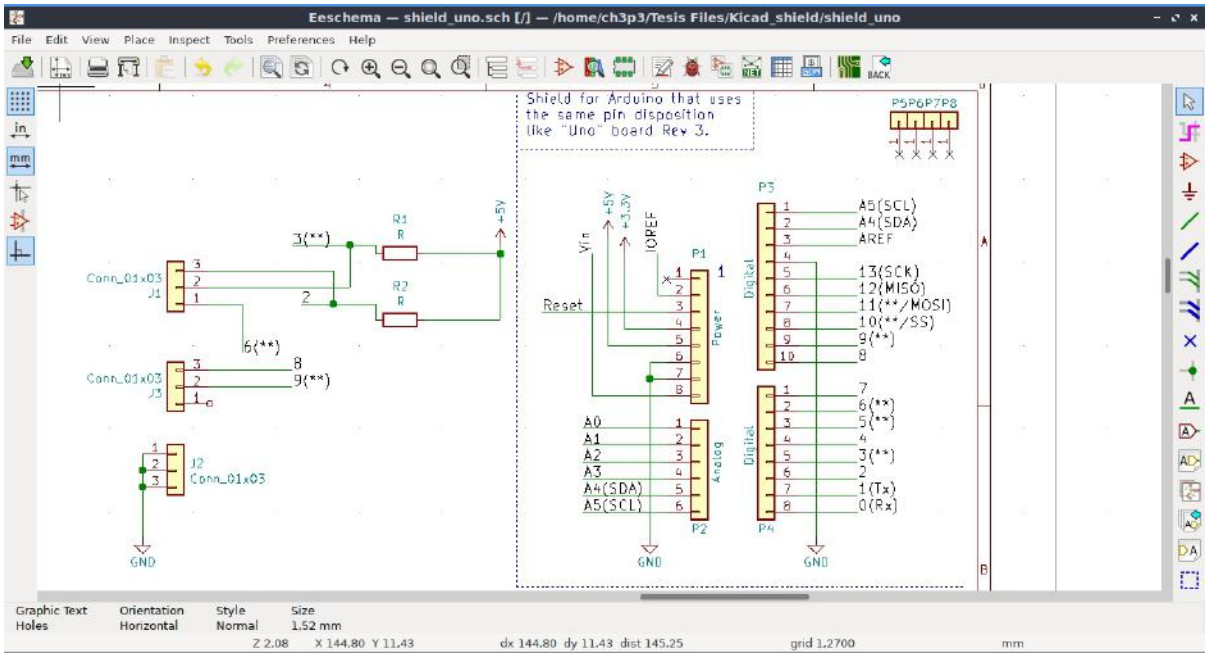

Figure 48: Esquemático para shield arduino uno

Aunque se trata de un diseño simple, hace más cómodas las conexiones realizadas entre la fuente, el módulo L298N y el arduino uno.

En la Figura 49 se puede ver el diseño de la PCB que será plasmada en una baquela por el método de transferencia por medio de la aplicación de calor. Posterior a esto se sumerje la

baquela en una solución de cloruro férrico más agua, que da lugar a un circuito electrónico listo para ser perforado y soldado.

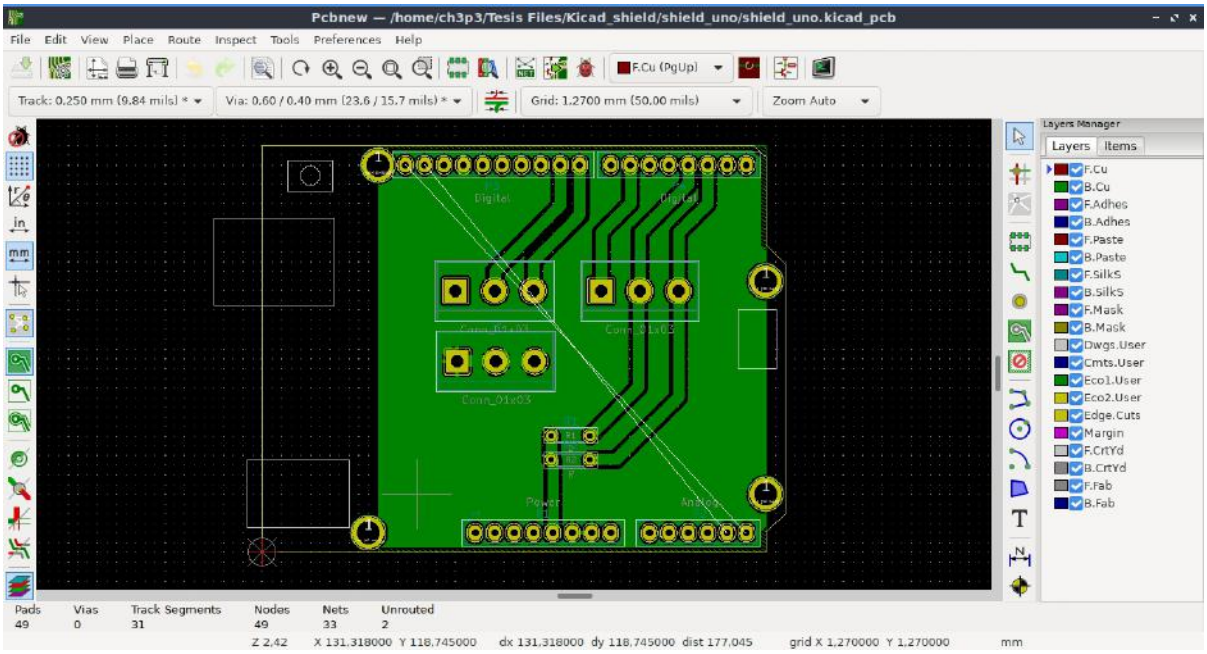

Figure 49: PCB para shield arduino uno

6.4 Diseño de una Estructura 3D en Solidworks

Para el diseño de la estructura se pensó en trabajar en materiales que fueran lo suficientemente asequibles para cualquier estudiante o persona que quisiera replicar dicho artefacto, siendo el material PLA el más idóneo para este tipo de labor. Con ayuda de una impresora 3D y el respectivo modelo CAD a imprimir se puede realizar esta tarea con total comodidad, dejando la tarea de replicar las partes que componen al péndulo de furuta a una impresora 3D.

No obstante la impresora 3D no puede hacer todo el trabajo por sí sola, se necesita de un archivo en formato STL el cual será el modelo o la pieza a imprimir.

La base que sostiene al brazo y péndulo que componen a toda la planta se puede apreciar en la Figura 50, se escogió este diseño ya que da un soporte sobre la estructura móvil que permite su libre desplazamiento sobre su eje.

Teniendo en cuenta los mecanismos vistos al principio del documento se toman como ejemplo para la construcción de nuestro mecanismo de péndulo rotacional invertido. La idea principal fue construir un mecanismo pequeño con respecto a los vistos en internet y a los de las referencias tomadas al comienzo de este archivo y sacar un modelo no tan ostentoso, que fuera de fácil transporte y que cumpla con los requisitos que tiene un péndulo al momento de la ejecución de su ejercicio.

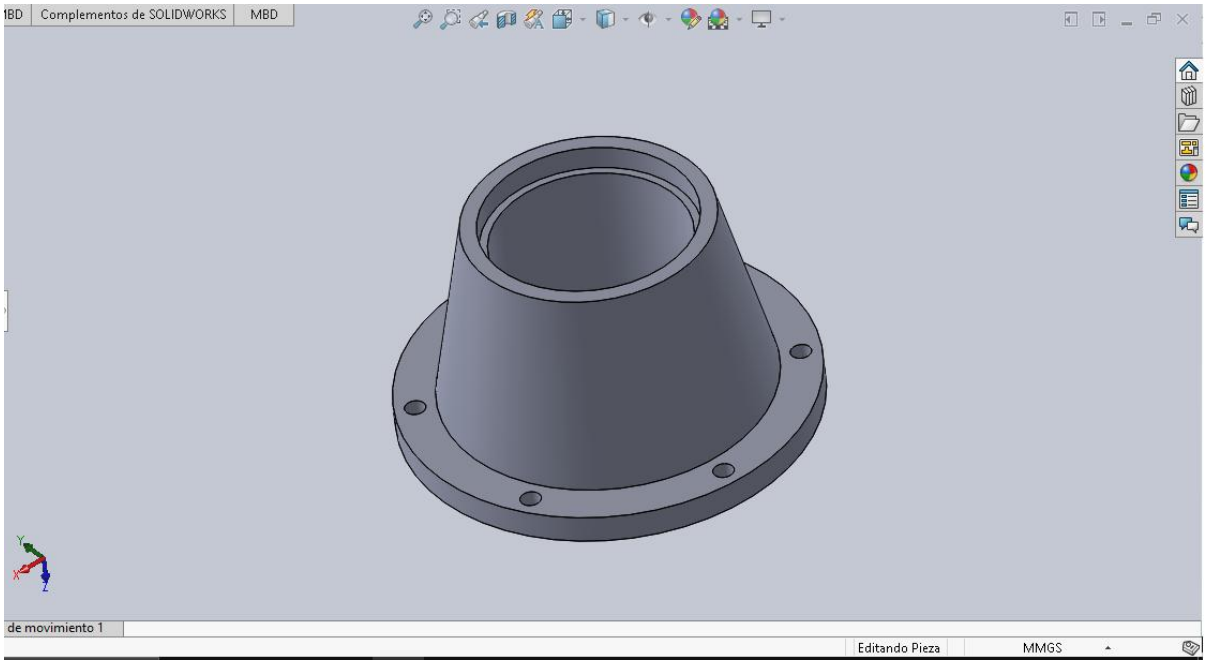

Figure 50: Base cónica

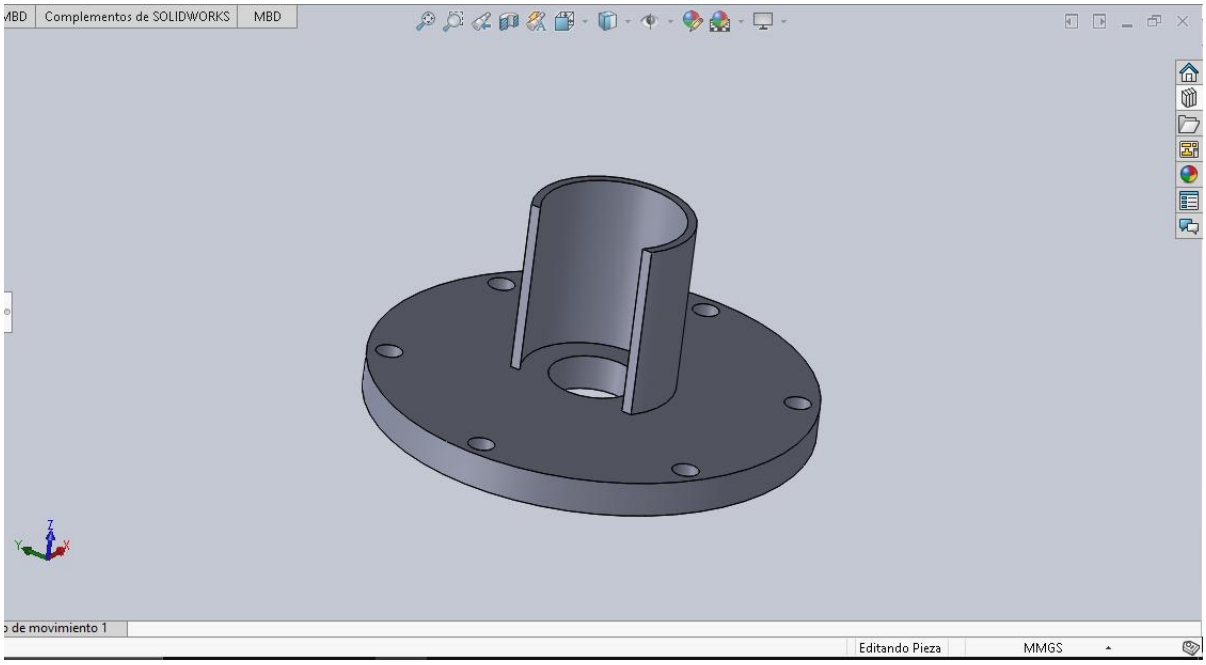

Figure 51: Base soporte motor dc

En la Figura 50, se observa lo que será la base que compone al péndulo en principio. Posee una serie de agujeros que servirán para ajustarlo a una superficie de más estabilidad. En la parte superior se instala una arandela de 52 mm de diametro exterior y 40 mm de diámetro

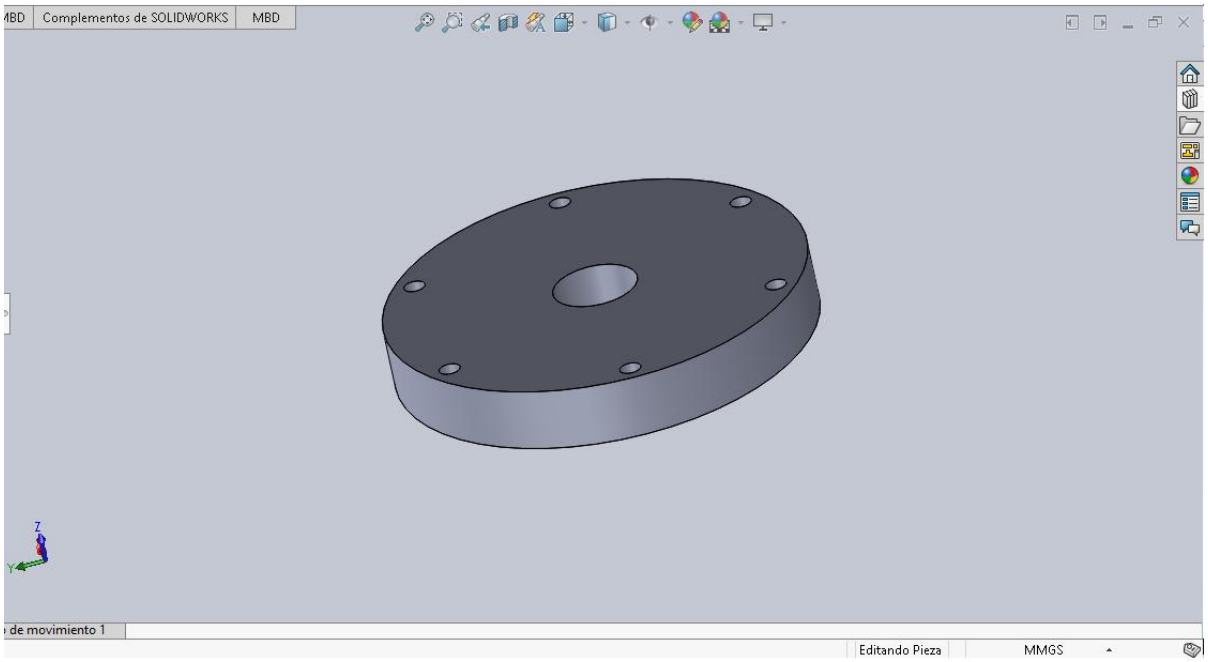

Figure 52: Extensor altura base cónica

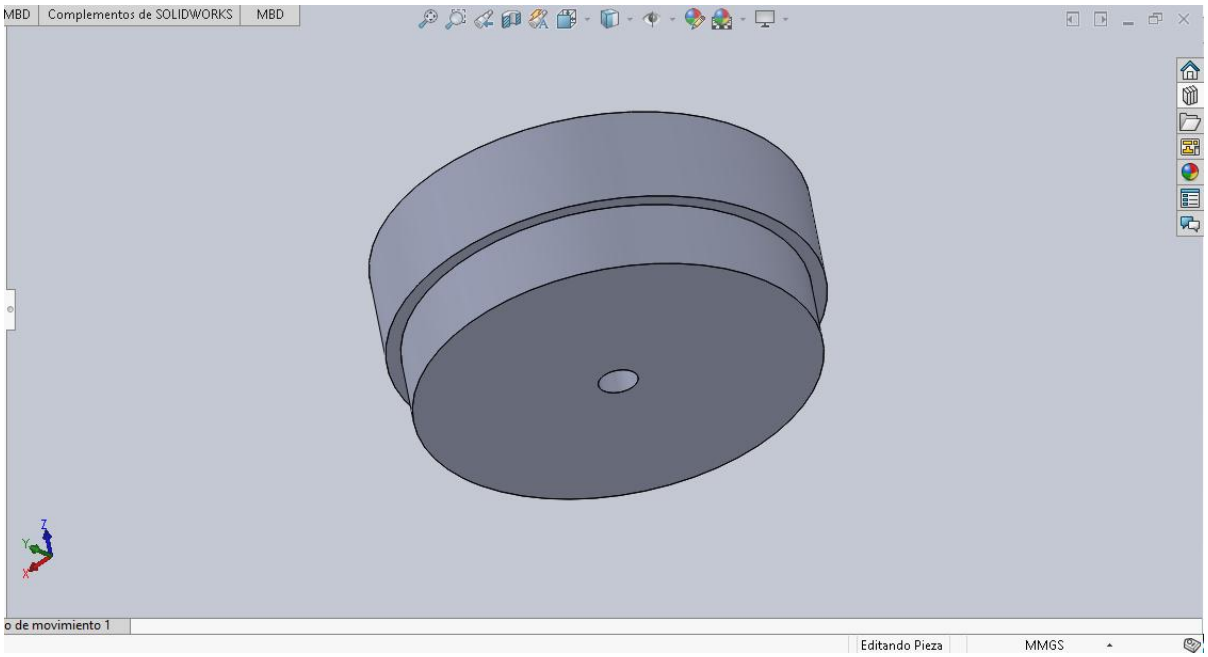

Figure 53: Unión rodamiento encoder

interno. Esto con el objetivo de acoplar el movimiento del encoder y en péndulo con el motor, de manera que el juego o balanceo hacia los lados sea casi cero, dando estabilidad y libre rotación al mecanismo.

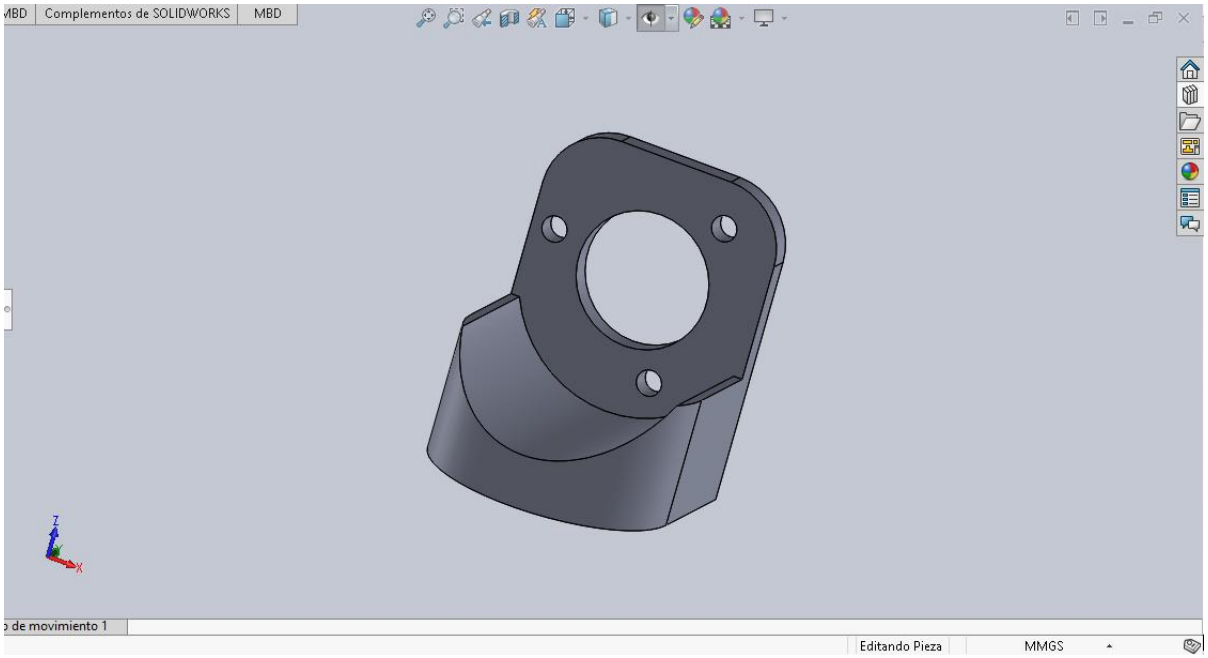

Figure 54: Soporte encoder

Con respecto a la Figura 51, este soporte para el motor dc permite mantener fijamente sin ningún tipo de movimiento no deseado y reducir las vibraciones del motor lo máximo posible, la idea es que a través de este sistema se pueda pasar el motor y que este sobre salga por la parte de arriba de la base cónica, permitiendo conectar la unión rodamiendo - encoder de la Figura 53.

La Figura 52, muestra un extensor que se diseñó como una solución rápida debido a un error en los cálculos de la base cónica al momento de instalarla junto al motor y su soporte.

El soporte del encoder mostrado en la Figura 54, se diseño para ser lo más ajustado posible al encoder, permitiendo ajustarlo de manera segura por medio de unos agujeros para unos tornillos de tipo M3 que ya trae instalado nuestro encoder. Luego este soporte de encoder se ajusta a la unión del rodamiento encoder y motor para dar a toda esta pequeña estructura movimiento libre de giro y permitir seguir con el estudio de este sistema.

6.5 Diseño de Caja Base

Es importante tener en cuenta que un contenedor para los componentes electrónicos es indispensable y útil para mantener el orden y realizar un diseño más agradable a la vista.

Un sitio web de bastante utilidad es *es.makercase.com* donde puedes diseñar una caja en cuestión de segundos escogiendo simplemente los parámtros de la misma y se obtiene un archivo .dwg listo para cortar en laser.

Para comodidad se escoge una lámina de madera aglomerada, también conocida como lámina mdf, la cual es económica y fácil de trabajar, esta es de un grosor o espesor de 3mm. El modelo quedaría como la Figura 55.

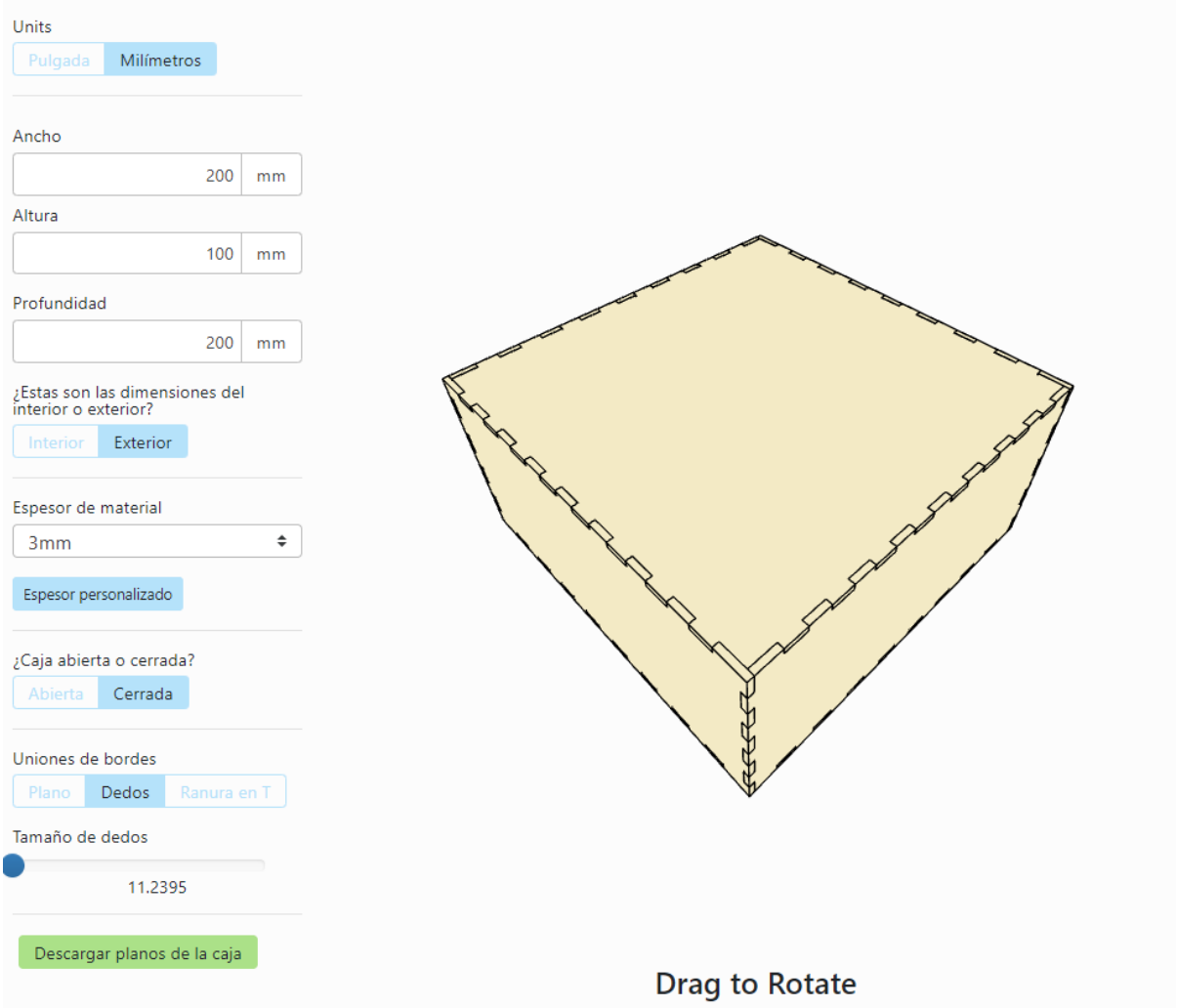

Figure 55: Base caja
Fuente: Captura tomada de es.makercase.com

Las dimensiones del esta caja para el sistema en cuestión son de 150 mm x 150 mm x

80 mm. Se escogen estas medidas arbitrariamente para dar un mejor soporte al sistema en cuestión. Dado que la caja no se encontrará vacía, junto con el uso de una superficie que sea lo suficientemente áspera para evitar deslizamiento innecesarios se espera que como mínimo exista un movimiento mínimo y alta fricción para con todo el sistema.

Se desechó la idea de anclar al suelo todo el sistema ya que arruinaría la característica que se desea en este sistema que es la portabilidad y fácil traslado.

6.6 Síntesis

Los tipos de diseño realizados para un sistema subactuado como lo es el péndulo rotacional invertido pueden ser variadas, es cierto que su mecanismo o estructura de acción es una sola, pero las maneras en las que se diseña teniendo en cuenta los materiales y la ergonomía del mismo varían demasiado. Se pudo haber realizado uno de los muchos modelos existentes en la internet sobre este mecanismo, pero no se compara con la satisfacción de crear algo sacado de los planos o borrador creados por la misma persona que realiza el ensamble. Es necesario hablar de mejores al mecanismo, pero ningún diseño está exento a evolucionar, por medio de un buen patrocinio y del estudio correcto de todas las partes se pueden llegar a crear mecanismos increíbles que ayuden con el avance de la ciencia.

7 Resultados Obtenidos

7.1 Introducción

En esta sección empezamos hablando del modelo físico real obtenido, mostrando imágenes del modelo creado en SolidWorks, las gráficas obtenidas de la simulaciones hechas en simulink en capítulos anteriores y el resultado de la aplicación de los controladores sobre el sistema simulado en cuestión. También se realizan unas breves descripciones las cuales nos sirven para saber un poco más sobre el proceso que se llevó a cabo durante toda la elaboración del trabajo de grado.

7.2 Resultados Desarrollo y Construcción

Partiendo de los modelos mostrados en el capítulo anterior y siguiente el orden de las cosas se muestra el resultado de la elaboración, del desarrollo de este sistema a partir de materiales de madera aglomerada o mdf para la caja y PLA para el resto de la estructura, en las Figuras 56 y 57 se muestra el resultado sacado de la inspiración y devoción hasta este instrumento, esta planta creada con el único propósito de ser controlada y servir como herramienta educativa para la academia y las personas que se esfuerzan cada día por estudiar materias de control moderno.

Figure 56: Resultado péndulo rotacional invertido
Fuente: Autoría propia

En un principio no se contaba con acceso a una impresora 3d, la cual fue de gran utilidad al momento de querer elaborar un sistema para el control como lo es el péndulo rotacional invertido. Esto se solucionó con una impresora 3d económica pero altamente funcional para la creación de piezas en material PLA.

Figure 57: Resultado péndulo rotacional invertido
Fuente: Autoría propia

Inicialmente se pensó en añadir unos potenciómetros junto con sus respectivas perillas para la manipulación de variables de control, como lo son las constantes kp, ki y kd de un sistema de control PID, pero esto no resultó en una gran idea ya que se toma una tarea sencilla y se vuelve en algo poco práctico que no ayuda en mucho al sistema en cuestión.

7.3 Resultados Simulaciones y Controladores

Como se observa en la Figura 58, la implementación del controlador PID dentro del sistema simulado es completa, se poseen dos controladores coordinados para su correcta ejecución y simulación con el modelo 3D está bien configurada para su correcta ejecución el sistema se ejecuta en cincuenta 50 segundos, pero su estabilización se realiza a los cinco 5 segundos. Es así debido a que el control Swing Up tarda en encontrar la zona PID a los cuatro segundos y medio, 4.5 segundos.

El esquema completo se usó también con la misma distribución para el controlador difuso y RNA. El *setpoint* o posición deseada es la misma, si se pone el valor correcto en radianes π *radianes* es posible que surjan errores con la compilación del diagrama de simulink debido a la naturaleza de este número irracional, por esta misma razón se decide poner este valor a simplemente el mostrado 58.

El bloque de switcheo que se muestra en el diagrama de bloques de simulink de la Figura 58, se puede observar que está configurado en un valor de dos punto nueve 2.9 este valor es la posición angular con respecto a la posición de reposo del péndulo, en esta posición de switcheo

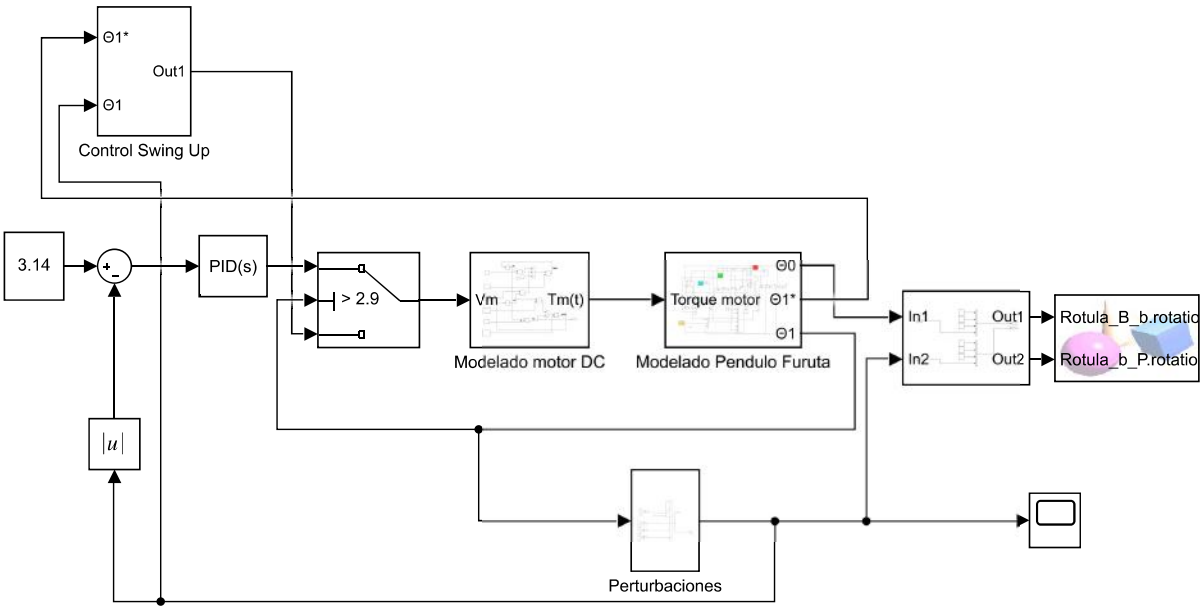

Figure 58: Esquema completo con controlador PID
Fuente: Autoría propia

el péndulo pasa de ser controlador por un sistema de control de tipo swing up a un sistema de controlador PID. Esto debido a la alta no linealidad que se mencionó anteriormente que posee nuestro sistema de péndulo rotacional invertido.

También se puede observar el modelo matemático dinámico tanto para el motor dc como para el péndulo de furuta. Aunque estén separados eso no implica que su comportamiento sea distinto al esperado. Después del bloque del péndulo y antes del bloque de la animación hay un bloque con dos entradas y dos salidas, este bloque se encarga de conectar correctamente las entradas a la animación que representarán la posición de los dos eslabones del sistema. Sin este bloque no se está claro de cuál será el movimiento de qué eslabón, por tanto es importante la existencia de este bloque para la ejecución de la animación.

Al momento de simular o poner en marcha este sistema en simulink se puede obtener una animación como la mostrada en la Figura 59.

Como se logra observar en la Figura 59, el sistema logra su cometido de lograr la estabilización. Se ponen unos fotogramas de la animación pero no se comparan con ver la simulación ejecutándose en tiempo real con sus respectivos tiempo de muestreo adecuados para una simulación fluida que permita observar los detalles que nos dan los controladores al momento de su ejecución.

Las perturbaciones implementadas en el sistema son simplemente pulsos creados con el objetivo de tratar de desestabilizar el sistema, cosa que no se logra, alcanza a desestabilizarse pero no lo suficiente para hacer que el péndulo pierda su centro.

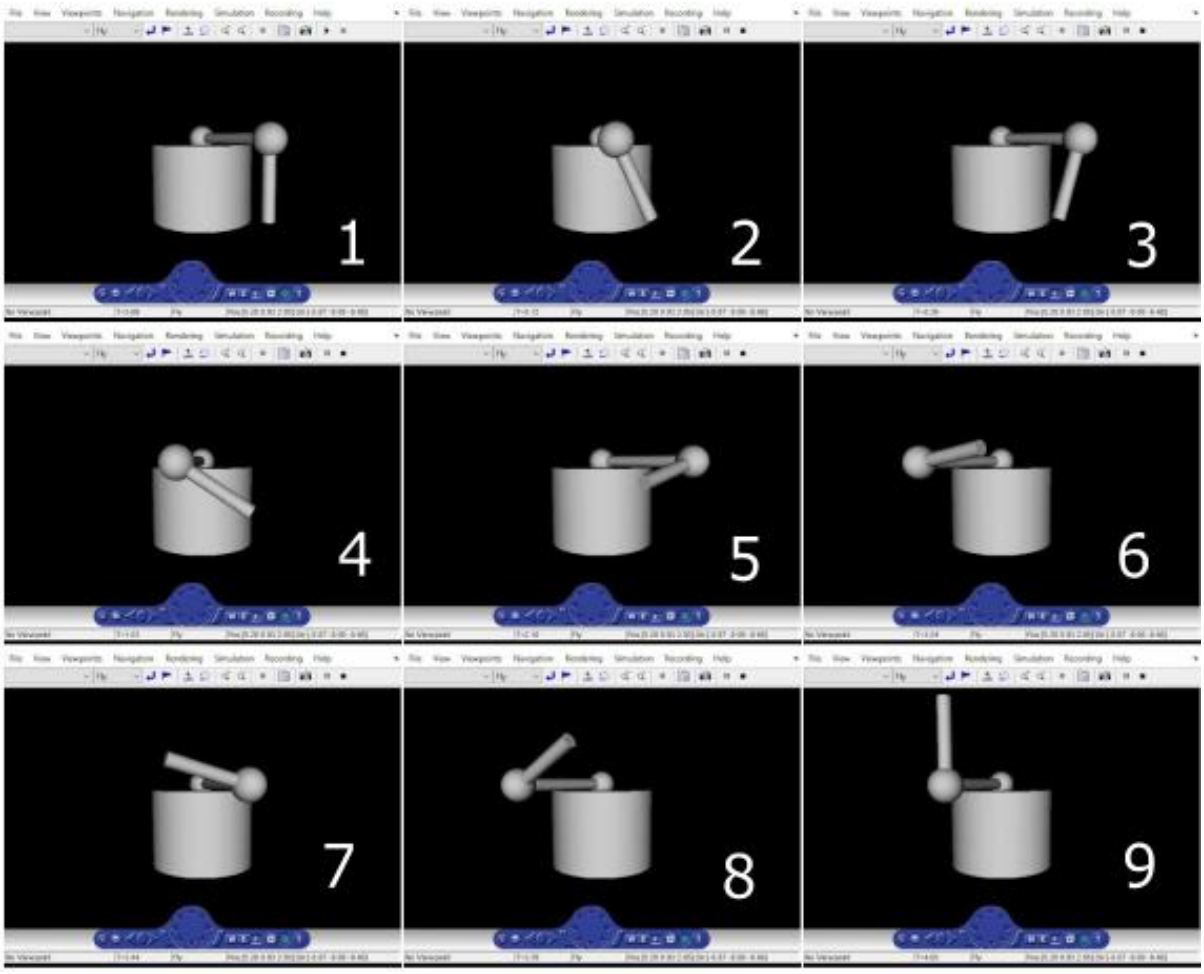

Figure 59: Fotogramas sacados de la animación
Fuente: Autoría propia

En la Figura 60, se muestra la respuesta a la salida del sistema, es decir, la posición angular del péndulo con respecto al tiempo, a simple vista el sistema reacciona como se espera, si se realiza un acercamiento como el de la Figura 61.

En la Figura 60, se muestra la respuesta a la salida del sistema, es decir, la posición angular del péndulo con respecto al tiempo, a simple vista el sistema reacciona como se espera, si se realiza un acercamiento como el de la Figura 61. Es posible observar con detalle las perturbaciones que ocurren en los tiempo aproximados de siete 7, nueve 9, doce 12 y quince 15 segundos. Son perturbaciones que aunque se ven pequeñas si representan cierta inestabilidad al sistema, inclusive se alcanzan a percibir una leve vibración si se observa la animación en tiempo real. Lo importante no es la perturbación en sí, lo que más llama la atención y concentra nuestro interés es la respuesta del controlador ante dichas perturbaciones.

El esquema presentado en la Figura 62, es bastante similar al presentado en 58, salvo por el controlador implementado, un controlador de tipo difuso el cual fue explicado en capítulos

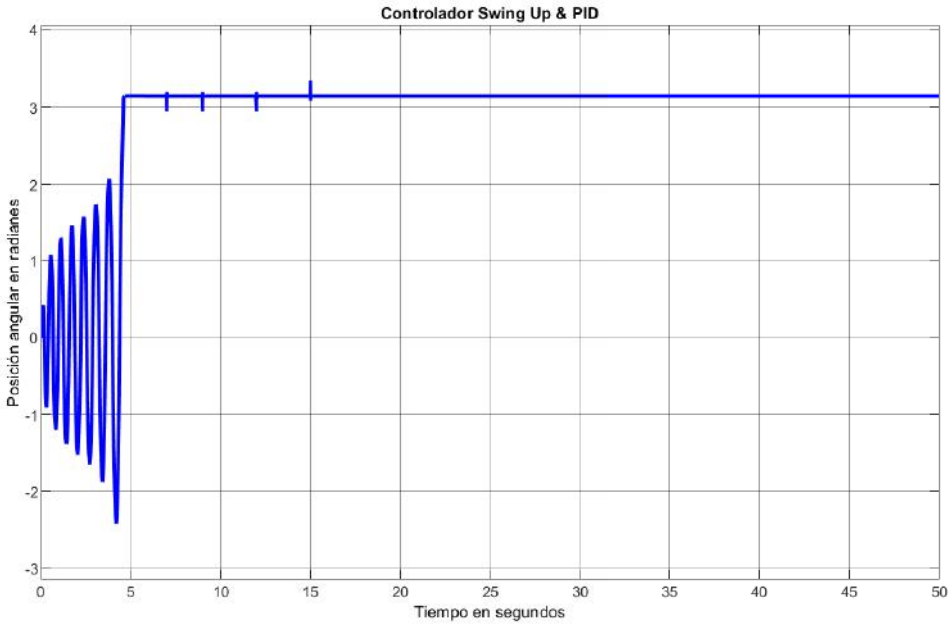

Figure 60: Posición del Péndulo vs Tiempo control PID
Fuente: Autoría propia

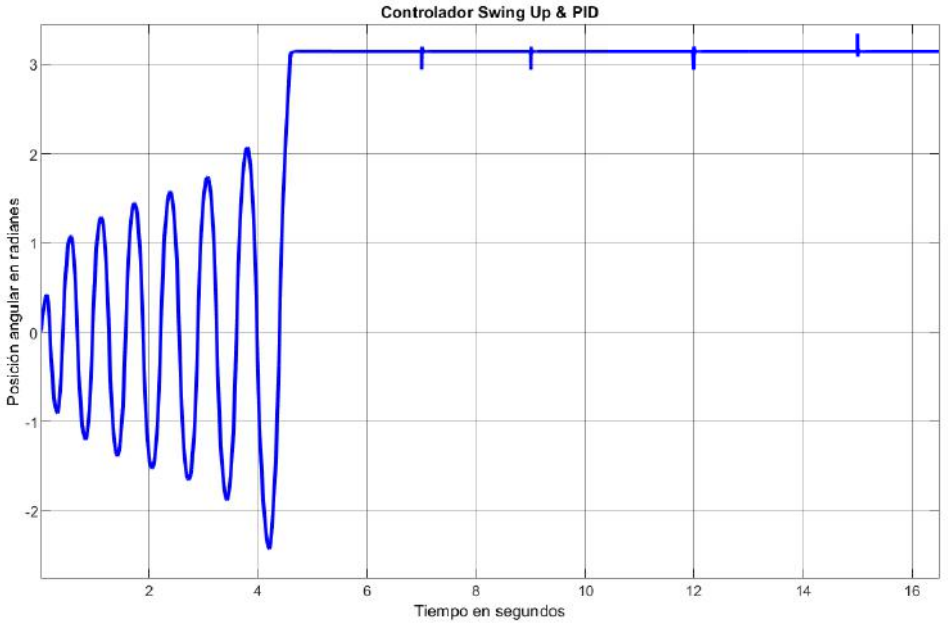

Figure 61: Posición del Péndulo vs Tiempo ampliado control PID
Fuente: Autoría propia

anteriores. El control difuso es el mismo, por ende la gráfica muestra un comportamiento similar a la mostrada del controlador PID hasta el tiempo exacto del switcheo.

Al momento del cambio al controlador difuso, su respuesta es algo sobreamortiguada y con algo de perturbación antes de la respuesta de estado transitoria, de todas formas es un control que cumple las expectativas que son las de mantener el péndulo en posición de equilibrio.

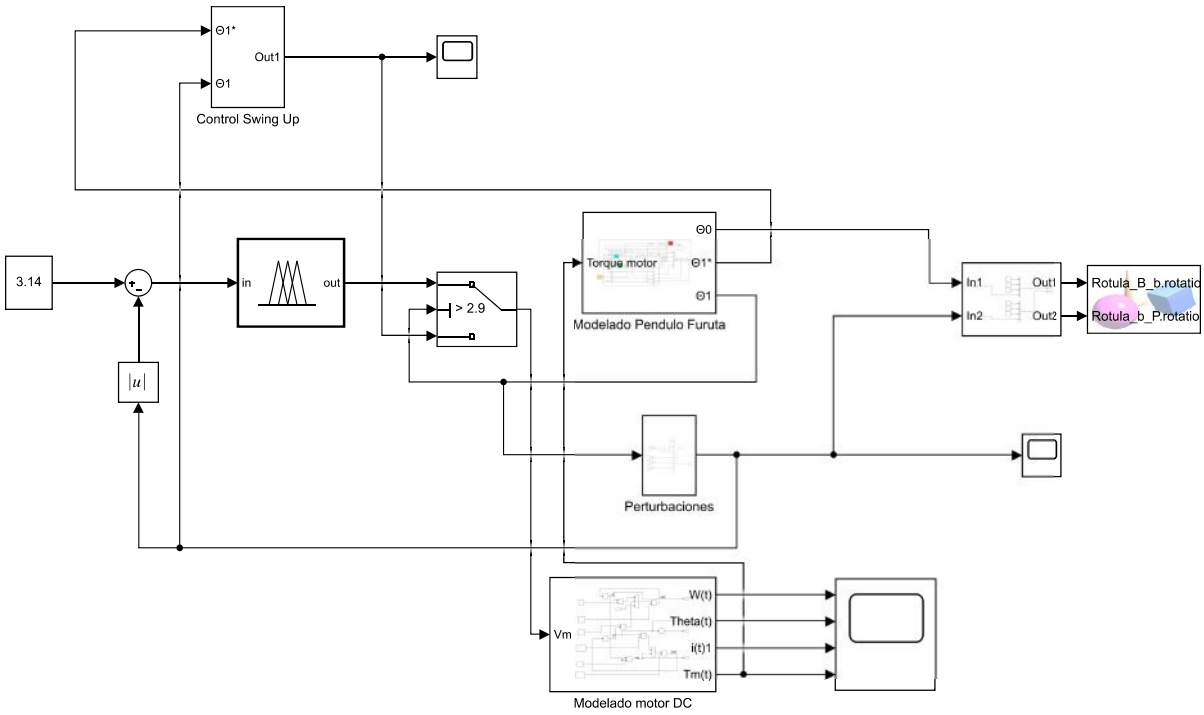

Figure 62: Esquema completo con controlador Fuzzy
Fuente: Autoría propia

El comportamiento del sistema en las animaciones es bastante similar al expresado en la Figura 59, por lo mismo se usa la misma imagen de referencia para dirigirse a lo que sucede dentro de este bloque de animación, salvo por las perturbaciones claro está, igualmente esto se puede observar con claridad en las gráficas.

Como se puede observar en la Figura 63, la gráfica es bastante similar a las del controlador PID salvo por las perturbaciones del sistema que resultan en una perturbación se podría decir nula del sistema, en ese sentido el controlador difuso se comporta de manera óptima ante estas pequeñas perturbaciones.

Aunque existan similitudes entre estos dos sistemas de control, entre estos dos controladores lo importante es cómo se comporta cada uno en este ambiente controlado, aplicado sobre un sistema real su comportamiento es bastante aproximado y es lo que se busca, pero va a variar incluso en un veinte por cierto 20 % solo por dar un valor.

Por otro lado y pasando al controlador implementado con redes neuronales artificiales RNA, la dinámica para este sistema es diferente, ya que se buscó imitar el sistema del controlador Swing Up, para ello se entrenó una red neuronal artificial con ayuda de las herramientas que provee MatLab para estos sistemas, aunque resulten obsoletas según sea la versión de MatLab,

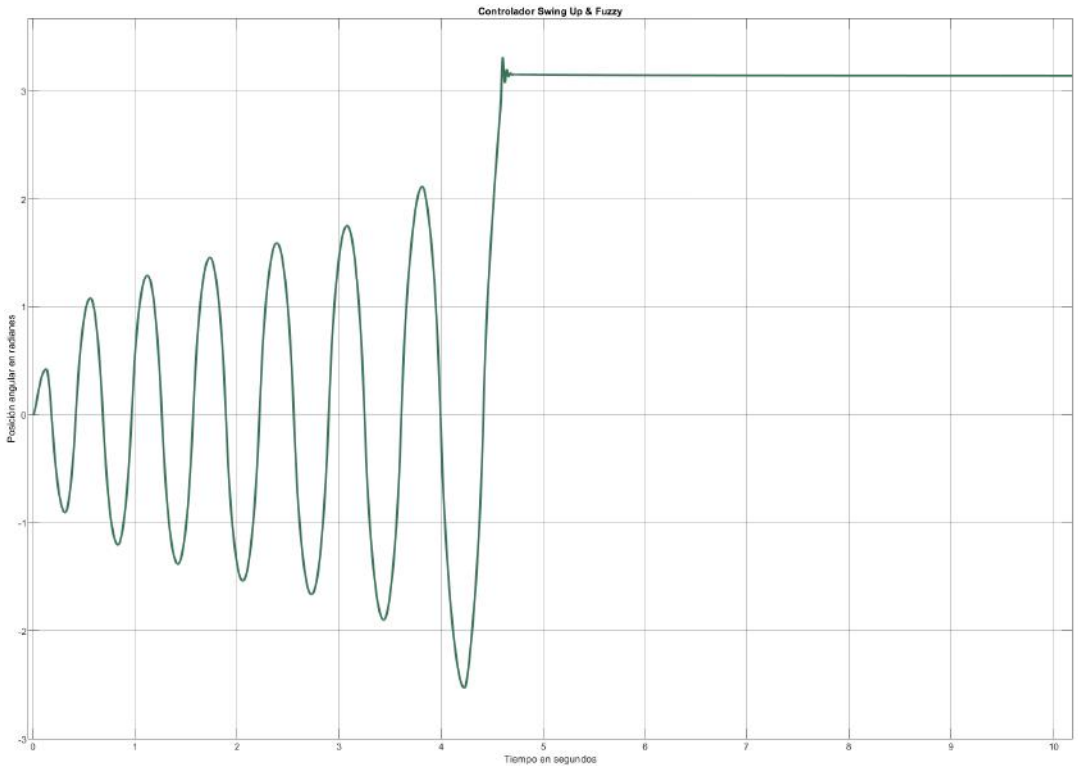

Figure 63: Posición del Péndulo vs Tiempo Control fuzzy
Fuente: Autoría propia

son lo suficientemente versátiles, útiles y de fácil manejo para la aplicación de este ejercicio.

A continuación, en la Figura 64 se presenta el diagrama de bloques realizado en simulink y con la ayuda de la toolbox que ofrece MatLab para trabajar con redes neuronales artificiales se crea este bloque grande de color azul específicamente para simulink, este adopta la estructura de la red neuronal creada en capítulos anteriores y al ser una red neuronal artificial entrenada previamente, es capaz de funcionar sin estimulaciones externas.

Se observa en la Figura 65, el comportamiento del controlador de redes neuronales artificiales junto con el controlador PID. A simple vista ambos no difiere demasiado, las diferencias entre esta respuesta y la del propio controlador Swing Up son parecidas, recordemos que este controlador de redes neuronales artificiales tiene el mismo objetivo, la misma tarea asignada que el controlador Swing Up, salvo que su estructura es bastante diferente, uno está compuesto por ecuaciones, funciones matemáticas y el otro se compone enteramente de redes neuronales y está entrenado con los valores que nos arroja el controlador Swing Up.

La respuesta de este sistema, plasmada en la Figura 65, presenta una diferencia con la respuesta expuesta en 60 y en 61, ambas son con un controlador de tipo Swing Up para el levantamiento del péndulo del sistema, y al compararlas con este último su tiempo de respuesta es menor en los primeros casos con respecto a este último ya que inicialmente el controlador Swing Up consigue llevar a la zona de equilibrio o cerca en menos de cinco 5 segundos, pero

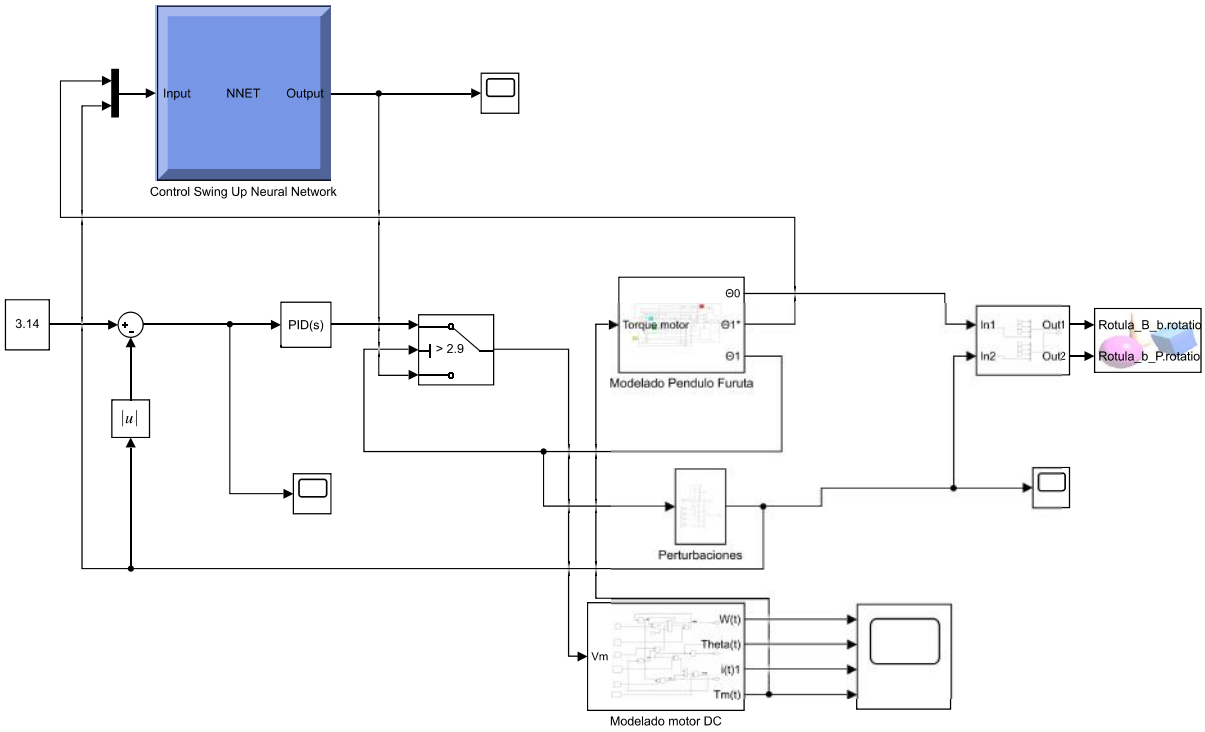

Figure 64: Esquema controlador RNA
Fuente: Autoría propia

usando las redes neuronales artificiales su tiempo se duplica aproximadamente hasta un valor de ocho segundos y medio 8.5, esto no quiere decir que el controlador no cumpla su objetivo, solo que es un poco más lento que el anterior.

Es importante mostrar como estas redes neuronales son capaces de imitar sistemas de control o en este caso modelos dinámicos o ecuaciones matemáticas, por ello es importante saber de su existencia y mostrar el potencial que poseen para este tipo de casos. Aún así el controlador con redes neuronales artificiales muestra ser muy útil para esta tarea en particular, imitando con precisión y ejecutándose limpiamente dentro del sistema. La animación para este caso sigue siendo demasiado parecida a la mostrada en la Figura 59, sobra decir que es más gratificante poder mostrar animaciones que fotogramas de ellas.

No todo el mérito se lo puede llevar la red neuronal artificial, ya que es el programador el que hace posible esta tarea. Aunque si es cierto que no se puede medir con exactitud hasta el momento no se sabe cuál es el número correcto de capas ocultas, de entrada o de salida que son las precisas para una tarea en particular.

El comportamiento del controlador PID en este último sistema es similar al presentado al principio de esta subsección, ya que al estar switcheado el sistema los valores a tomar están aislados unos de otros.

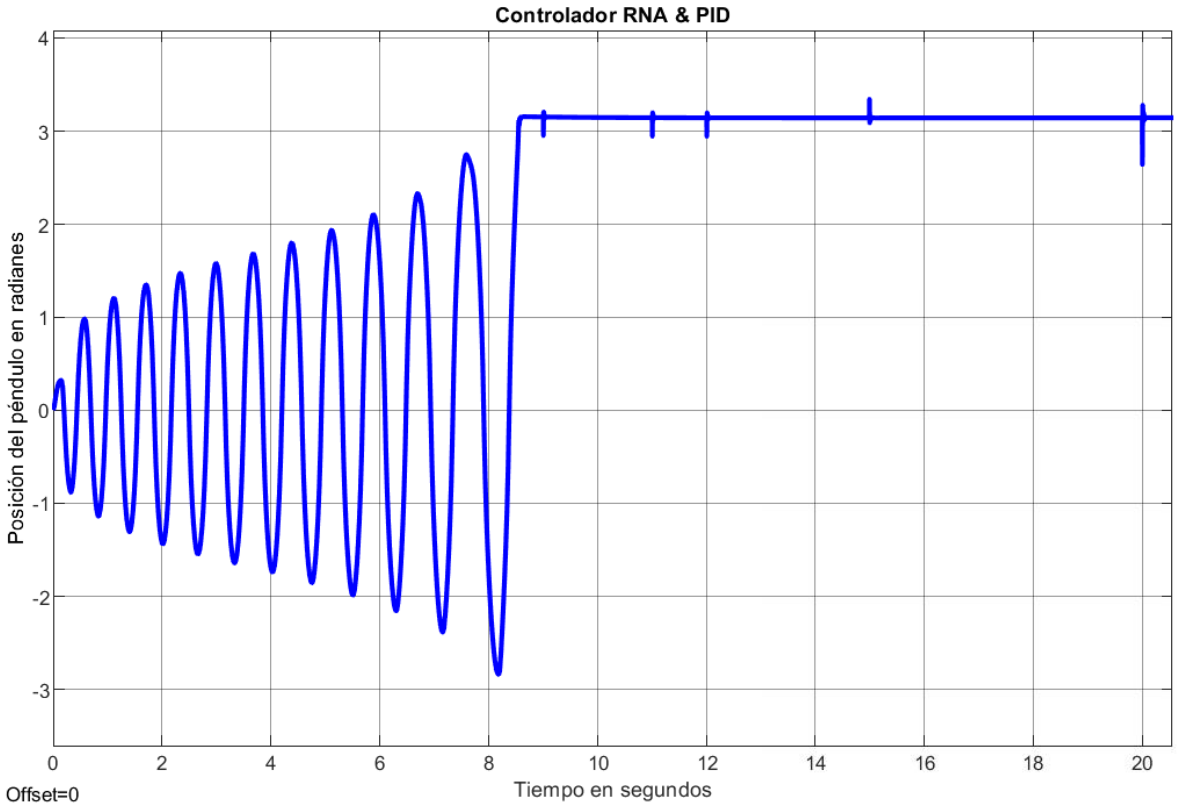

Figure 65: Posición del Péndulo vs Tiempo ampliado control RNA
Fuente: Autoría propia

7.4 Resultados del sistema físico real

En esta sección se evidencia parte de las pruebas realizadas para el sistema físico real. Inicialmente se requiere de un sistema de bloques funcionales construidos en Simulink los cuales junto con el bloque de función *S-Function Builder* y los bloques de funciones propios de la librearía de Arduino para Simulink se obtiene un esquema completo de bloques funcionales capaces de poner en marcha el sistema de péndulo rotacional invertido.

El sistema mostrado en la Figura 66, tiene algunos bloques que sirven como topes al momento en el cual el péndulo entre en zonas no deseadas o sea totalmente erróneo en su comportamiento, esto evita que el sistema llegue a dañarse debido a comportamientos no deseados y es más fácil manipularlo de esta manera.

Los bloques propios de la librería de Arduino para Simulink establecen modos de trabajo para las salidas o entradas del sistema embebido, en la Figura 66, se observan estos bloques para los pines 8,9 y 6 para la placa Arduino los cuales se encargarán de enviar las ordenes al motor, control de velocidad por pwm en el pin 6 y sentido de giro por cambio de *HIGH* a *LOW* en los pines 8 y 9. Los datos del encoder son enviados a los pines 2 y 3 los cuales admiten interrupciones externas al sistema embebido.

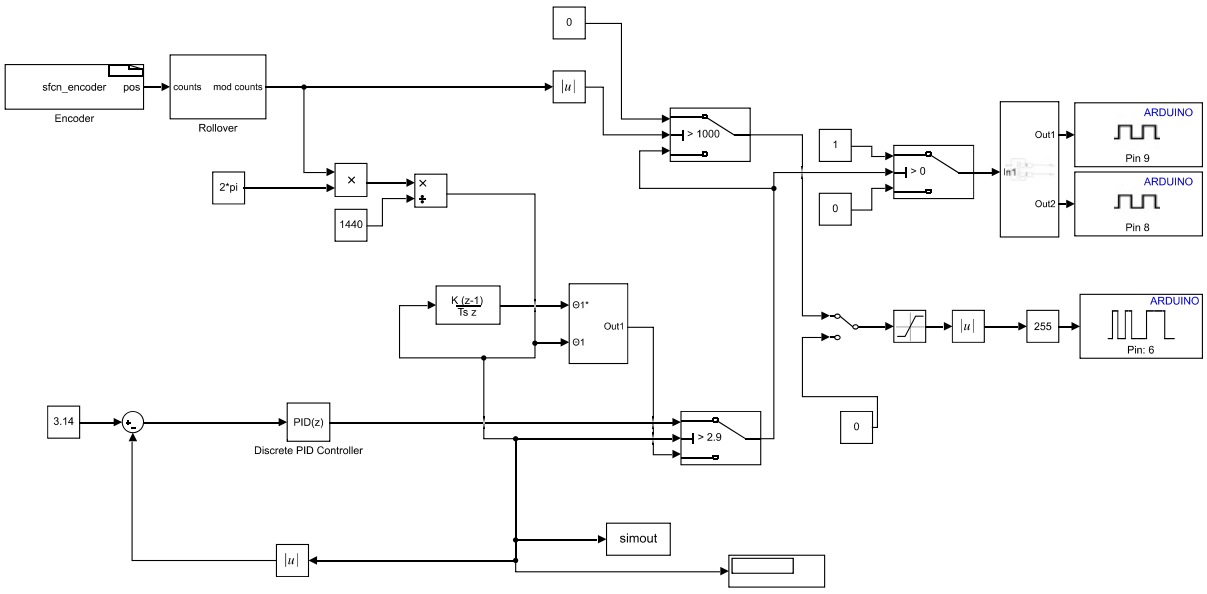

Figure 66: Simulink diagrama de bloques conexión con Arduino
Fuente: Autoría propia

Al tratarse de un sistema digital no se puede usar el bloque de función PID continuo es necesario usar el de tipo discreto. Para el sistema de Swing Up se usan casi los mismos sistemas de bloques de la Figura 20 excepto por unas variaciones en el mismo, se cambian los valores de treinta 30 y menos treinta -30 por los valores de cero punto siete 0.7 y menos cero punto siete -0.7. Estos valores deben ajustarse me manera experimental y encontrar el que mejor se ajuste al sistema, por lo general esto depende del valor de la fuente ya que entre menor sea su valor al momento de alimentar el módulo PWM de control de giro del motor, mayor a de ser este valor de cada uno de los límites en el controlador Swing Up.

La diferencia con respecto a los sistemas simulados se puede ver en la Figura 67, el cual es más rápida su respuesta si se habla del sistema de balanceo Swing Up. Con respecto al controlador PID la respuesta de este es aceptable pero no del todo satisfactoria ya que el sistema está expuesto a perturbaciones externas o que son generadas por vibraciones presentes en los componentes que hacen parte del módulo de péndulo rotacional invertido e influyen constantemente en la posición del péndulo en zona de equilibrio. Aún así el péndulo se mantiene en esta posición erguida con un poco de ayuda manual. Se debe tener en cuenta que al no tener un control de la posición del motor con respecto a su propio eje, este tiene a irse hacia un lado o hacia otro depende del valor que tenga el encoder que mide su posición angular.

El estudiante al momento de poner en marcha este sistema debe asegurarse de tener el péndulo en posición de reposo apuntando hacia abajo para comenzar su posición inicial en cero grados, si no se tiene en cuenta esto al momento de iniciar el programa, el sistema tomará cualquiera que sea la posición del péndulo como su posición inicial de cero grados o cero radianes. También tenga en cuenta que el cable conectado al encoder necesita de cierta longitud para

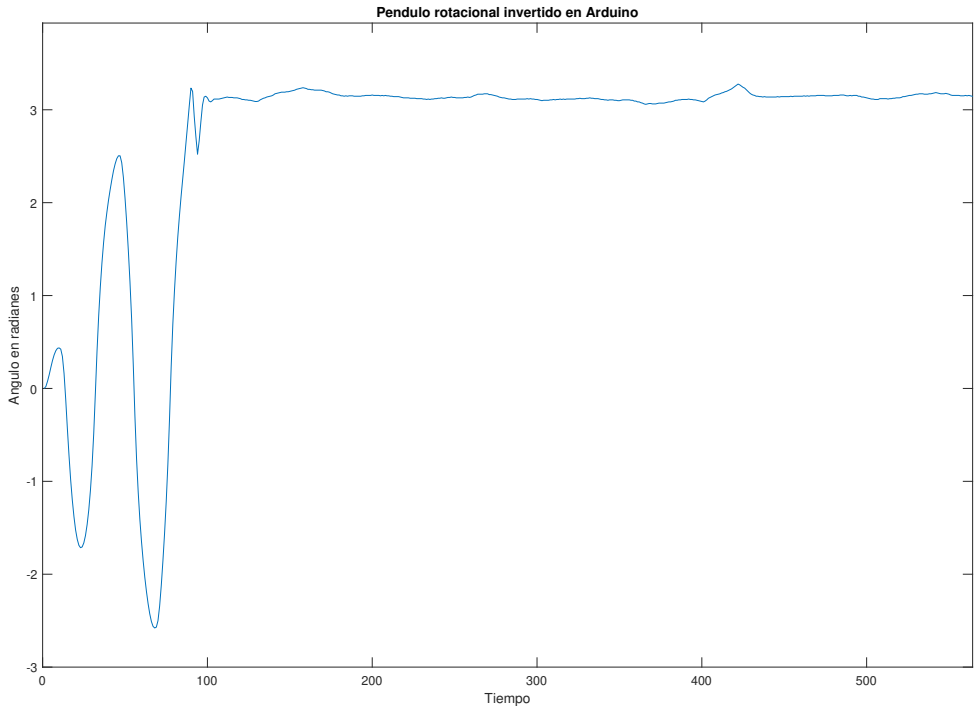

Figure 67: Respuesta a la salida del sistema conexión con Arduino péndulo rotacional invertido
Fuente: Autoría propia

que el péndulo se mueva de manera correcta durante su ejecución, si se llega a enredar puede ocasionar tensiones que afecten el desempeño del sistema.

7.5 Usar librerías de Arduino dentro de Simulink

Es muy frecuente encontrar aplicaciones, ejercicios, tutoriales en los cuales se usen librerias de terceros para hacer más ameno el realizar programas y desarrollar código usando las herramientas que ofrece arduino.

El inconveniente surge cuando se requiere utilizar estas librerías dentro de un entorno diferente a lo que es arduino. Dentro de Simulink se pasa de la programación utilizada en Arduino (un tipo lenguaje C++) a un entorno de programación visual como lo es Simulink.

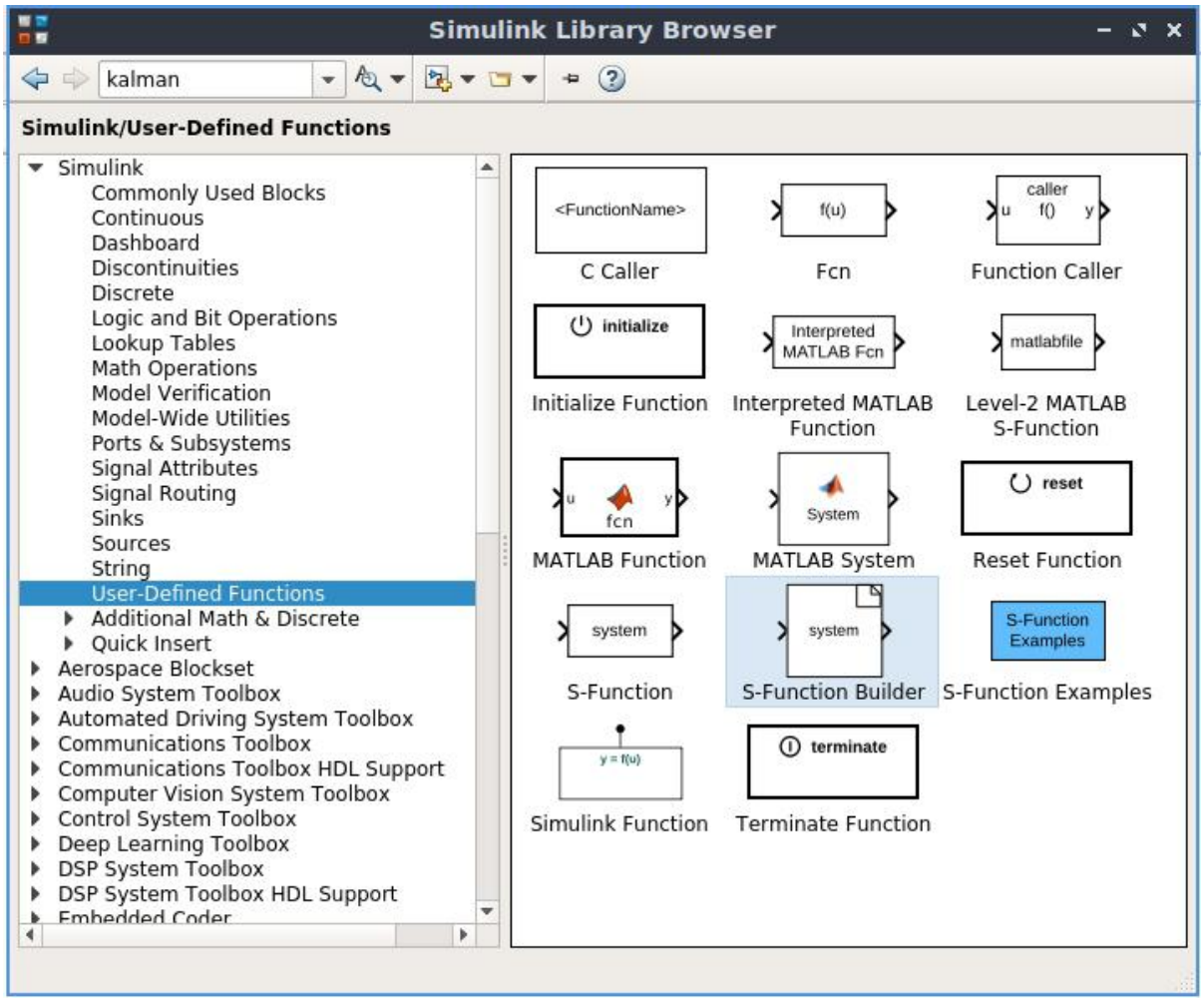

Figure 68: Buscador de Librerias de Simulink
Fuente: Autoría Propia

Lo ideal es saber como trabajamos con estas librerías dentro de MatLab y más especificamente dentro de Simulink.

A continuación se muestra lo más detalladamente posible cómo poder implementar nuestras propias librerías de Arduino al momento de querer ejecutar tareras con éstas.

Lo primero a tener en cuenta es saber cómo empezar a gestionar la librería a implementar. En otras palabras lo que se pretende realizar con esta acción es utilizar un código de Arduino dentro de un bloque de función de Simulink y así poder utilizar más recursos de la tarjeta y no limitarse por los bloques de salidas y entradas que tenemos en Simulink.

En la Figura 68 se puede observar el buscador de librerías de Simulink. Dentro de éste, se selecciona el bloque que tiene por nombre "S-Function Builder" el cual nos permite crear funciones para luego poder implementarlas en el espacio de trabajo de Simulink.

Una vez añadido el bloque funcional al espacio de trabajo se procede a configurarlo. Al momento de hacer doble click sobre este bloque, se nos abrirá la ventana mostrada en la Figura 69.

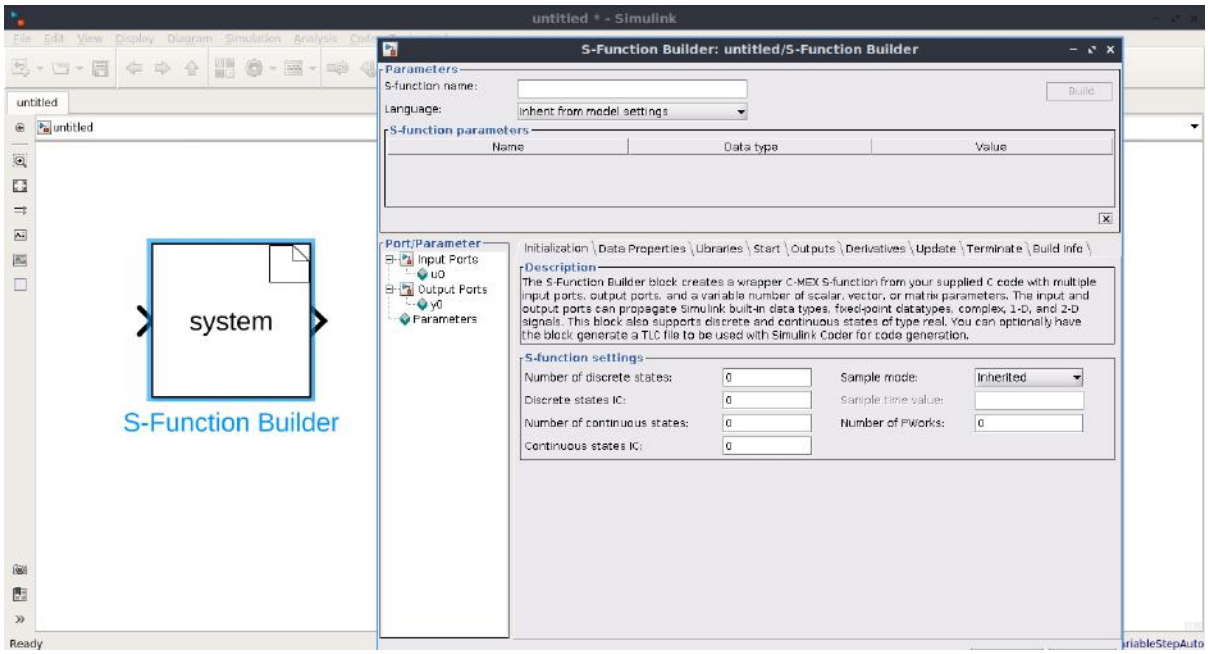

Figure 69: Interfaz del bloque S-Function Builder
Fuente: Autoría Propia

En el panel ubicado en la parte izquierda el cual tiene como título *Port/Parameter*, se pueden apreciar los *Input Ports*, que son las entradas al bloque así como los *Output Ports*, que vendrían a ser las salidas al bloque funcional. Los parámetros son para este caso los pines que se usaran en la tarjeta embebida, los cuales no interactúan directamente dentro de la lógica en el espacio de trabajo de simulink.

El primer de los parámetros a configurar se encuentra dentro de la pestaña *Initialization*, allí se explica de manera breve en su descripción los tipos de datos que se permiten integran dentro de la programación en C y su posible configuración como entrada, salidas o parámetros. Para este caso solo se establecerá en 1 el número de estados discretos *(Number of discrete states)*. El resto de los parámetros se pueden dejar tal cual. La opción Sample Mode puede ser configurada aquí o en la opción que ofrece Simulink en Tools - Run and Target - Options - Solver.

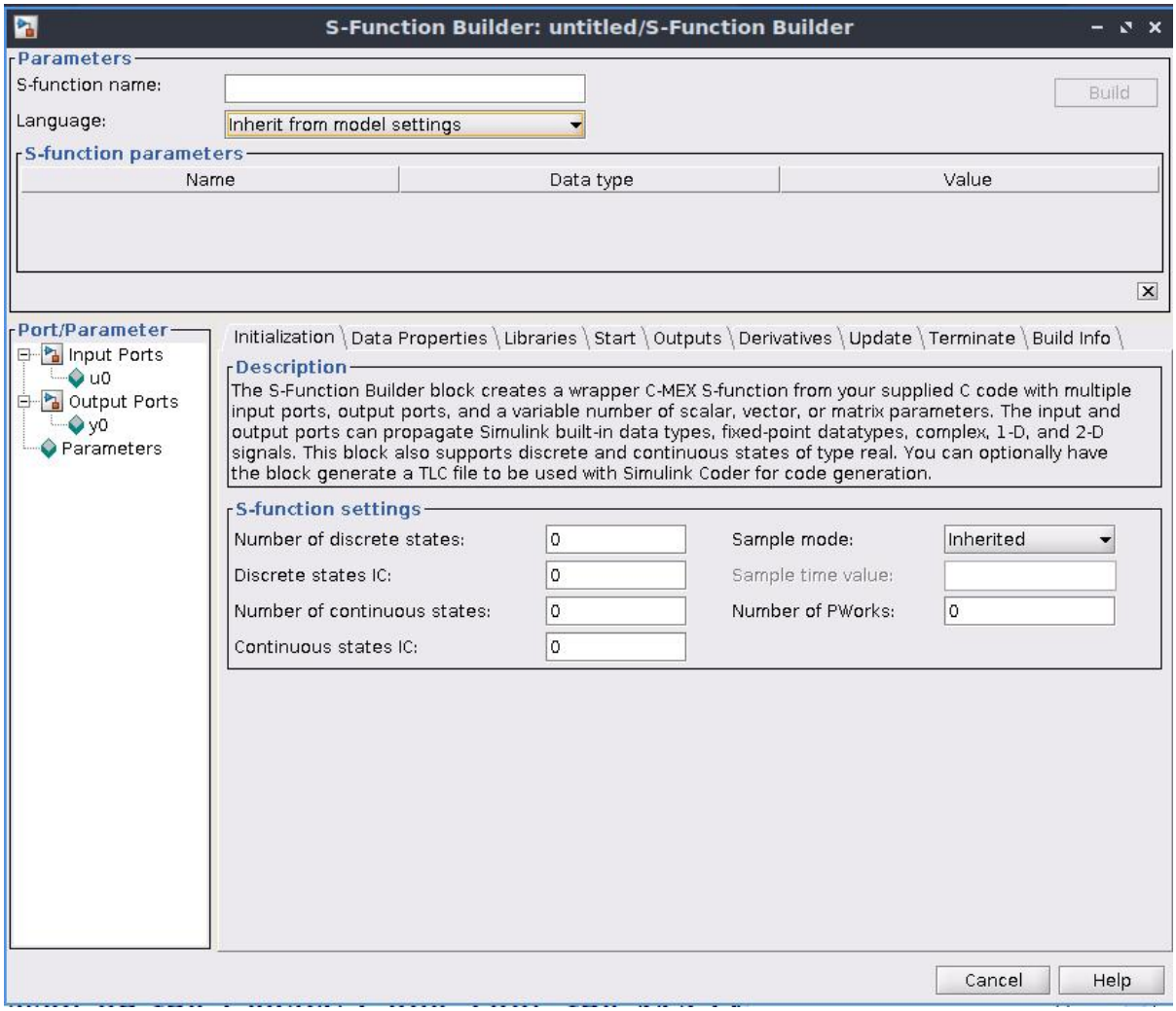

Figure 70: Configuración de la pestaña Initialization
Fuente: Autoría Propia

En la Figura 70, se puede apreciar con mejor detalle cada uno de los aspectos a configurar mencionados anteriormente.

Para la siguiente pestaña la cual tiene como nombre *Data Properties*, se elimina la variable alojada en Input Ports, ya que no se necesita un puerto de entrada para nuestro bloque, en vez de eso se añaden 3 variables en el apartado de Parámetros y se nombran tal cual se observa en la Figura 71. Es importante mencionar que los pines elegidos para las variables pinA y pinB no son elegidos al azar, si no que corresponden a los pines que nuestra placa Arduino tiene disponible para las interrupciones.

Es importante saber cuales son las interrupciones que posee nuestro sistema o tarjeta embebida ya que estas interrupciones son las que permiten que todos los datos que se envian desde el encoder sean leídos por el arduino.

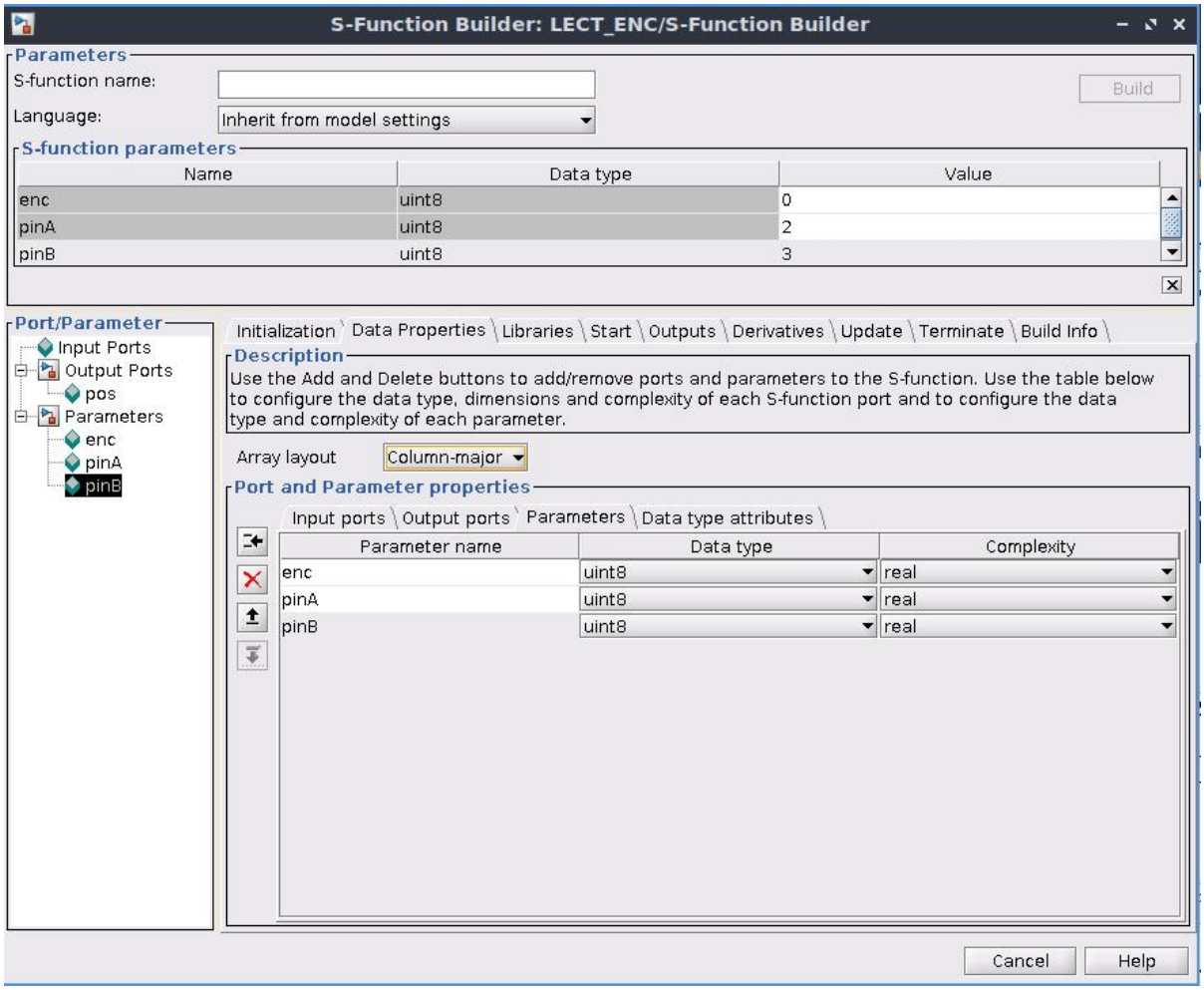

Figure 71: Configuración de la pestaña Data Properties
Fuente: Autoría Propia

Dentro de la pestaña Libraries que es la más fundamental, ya que nos permite ingresar el código en lenguaje C que usamos regularmente en el IDE de arduino. Allí implementaremos el siguiente código:

```
# ifndef MATLAB_MEX_FILE
# include <Arduino.h>

typedef struct { int pinA; int pinB; int pos; int del;} Encoder;
volatile Encoder Enc[3] = {{0,0,0,0}, {0,0,0,0}, {0,0,0,0}};

/* auxiliary function to handle encoder attachment           */
int getIntNum(int pin) {
/* returns the interrupt number for a given interrupt pin
   see http://arduino.cc/it/Reference/AttachInterrupt        */
switch(pin) {
  case 2:
```

```
    return 0;
  case 3:
    return 1;
  case 21:
    return 2;
  case 20:
    return 3;
  case 19:
    return 4;
  case 18:
    return 5;
  default:
    return -1;
  }}
/* auxiliary debouncing function                                */
void debounce(int del) {
  for (int k=0;k<del;k++) {
    /* can't use delay in the ISR so need to waste some time
       perfoming operations, this uses roughly 0.1ms on uno  */
    k = k +0.0 +0.0 -0.0 +3.0 -3.0;
  }}
/* Interrupt Service Routine: change on pin A for Encoder 0  */
void irsPinAEn0(){

  /* read pin B right away                                      */
  int drB = digitalRead(Enc[0].pinB);

  /* possibly wait before reading pin A, then read it           */
  debounce(Enc[0].del);
  int drA = digitalRead(Enc[0].pinA);

  /* this updates the counter                                   */
  if (drA == HIGH) {   /* low->high on A? */

    if (drB == LOW) {  /* check pin B */
    Enc[0].pos++;  /* going clockwise: increment         */
    } else {
    Enc[0].pos--;  /* going counterclockwise: decrement  */
    }

  } else {                      /* must be high to low on A */

    if (drB == HIGH) { /* check pin B */
    Enc[0].pos++;  /* going clockwise: increment         */
    } else {
    Enc[0].pos--;  /* going counterclockwise: decrement  */
    }

  } /* end counter update                                       */

} /* end ISR pin A Encoder 0                                    */
/* Interrupt Service Routine: change on pin B for Encoder 0  */
```

```
void isrPinBEn0(){

  /* read pin A right away                                    */
  int drA = digitalRead(Enc[0].pinA);

  /* possibly wait before reading pin B, then read it         */
  debounce(Enc[0].del);
  int drB = digitalRead(Enc[0].pinB);

  /* this updates the counter                                 */
  if (drB == HIGH) {   /* low->high on B? */

    if (drA == HIGH) { /* check pin A */
    Enc[0].pos++;  /* going clockwise: increment         */
    } else {
    Enc[0].pos--;  /* going counterclockwise: decrement  */
    }

  } else {                      /* must be high to low on B */

    if (drA == LOW) {  /* check pin A */
    Enc[0].pos++;  /* going clockwise: increment         */
    } else {
    Enc[0].pos--;  /* going counterclockwise: decrement  */
    }
  } /* end counter update */
} /* end ISR pin B Encoder 0  */

/* Interrupt Service Routine: change on pin A for Encoder 1  */
void irsPinAEn1(){

  /* read pin B right away                                    */
  int drB = digitalRead(Enc[1].pinB);

  /* possibly wait before reading pin A, then read it         */
  debounce(Enc[1].del);
  int drA = digitalRead(Enc[1].pinA);

  /* this updates the counter                                 */
  if (drA == HIGH) {   /* low->high on A? */

    if (drB == LOW) {  /* check pin B */
    Enc[1].pos++;  /* going clockwise: increment         */
    } else {
    Enc[1].pos--;  /* going counterclockwise: decrement  */
    }

  } else { /* must be high to low on A                         */
      if (drB == HIGH) { /* check pin B */
    Enc[1].pos++;  /* going clockwise: increment         */
    } else {
    Enc[1].pos--;  /* going counterclockwise: decrement  */
```

```
117     }
118
119   } /* end counter update                                        */
120
121 } /* end ISR pin A Encoder 1                                     */
122
123
124 /* Interrupt Service Routine: change on pin B for Encoder 1  */
125 void isrPinBEn1(){
126
127   /* read pin A right away                                      */
128   int drA = digitalRead(Enc[1].pinA);
129
130   /* possibly wait before reading pin B, then read it        */
131   debounce(Enc[1].del);
132   int drB = digitalRead(Enc[1].pinB);
133
134   /* this updates the counter                                   */
135   if (drB == HIGH) {   /* low->high on B? */
136
137     if (drA == HIGH) { /* check pin A */
138     Enc[1].pos++;  /* going clockwise: increment         */
139     } else {
140     Enc[1].pos--;  /* going counterclockwise: decrement  */
141     }
142
143   } else { /* must be high to low on B                         */
144
145     if (drA == LOW) {  /* check pin A */
146     Enc[1].pos++;  /* going clockwise: increment         */
147     } else {
148     Enc[1].pos--;  /* going counterclockwise: decrement  */
149     }
150
151   } /* end counter update                                       */
152
153 } /* end ISR pin B Encoder 1                                    */
154
155
156 /* Interrupt Service Routine: change on pin A for Encoder 2  */
157 void irsPinAEn2(){
158
159   /* read pin B right away                                      */
160   int drB = digitalRead(Enc[2].pinB);
161
162   /* possibly wait before reading pin A, then read it        */
163   debounce(Enc[2].del);
164   int drA = digitalRead(Enc[2].pinA);
165
166   /* this updates the counter                                   */
167   if (drA == HIGH) {   /* low->high on A? */
168
```

```
169     if (drB == LOW) {  /* check pin B */
170     Enc[2].pos++;  /* going clockwise: increment         */
171     } else {
172     Enc[2].pos--;  /* going counterclockwise: decrement  */
173     }
174
175   } else { /* must be high to low on A                      */
176
177     if (drB == HIGH) { /* check pin B */
178     Enc[2].pos++;  /* going clockwise: increment         */
179     } else {
180     Enc[2].pos--;  /* going counterclockwise: decrement  */
181     }
182
183   } /* end counter update                                   */
184
185 } /* end ISR pin A Encoder 2                               */
186
187
188 /* Interrupt Service Routine: change on pin B for Encoder 2  */
189 void isrPinBEn2(){
190
191   /* read pin A right away                                  */
192   int drA = digitalRead(Enc[2].pinA);
193
194   /* possibly wait before reading pin B, then read it       */
195   debounce(Enc[2].del);
196   int drB = digitalRead(Enc[2].pinB);
197
198   /* this updates the counter                               */
199   if (drB == HIGH) {   /* low->high on B? */
200
201     if (drA == HIGH) { /* check pin A */
202     Enc[2].pos++;  /* going clockwise: increment         */
203     } else {
204     Enc[2].pos--;  /* going counterclockwise: decrement  */
205     }
206
207   } else { /* must be high to low on B                      */
208
209     if (drA == LOW) {  /* check pin A */
210     Enc[2].pos++;  /* going clockwise: increment         */
211     } else {
212     Enc[2].pos--;  /* going counterclockwise: decrement  */
213     }
214
215   } /* end counter update                                   */
216
217 } /* end ISR pin B Encoder 2                               */
218
219 # endif
```

El código en sí no es lo importante sino la lógica detrás del mismo es lo que verdaderamente es fundamental. Luego de colocar el código de la librería para lectura de encoders de tipo incremental dentro de la ventana *Include files and external function declarations*, Figura 72

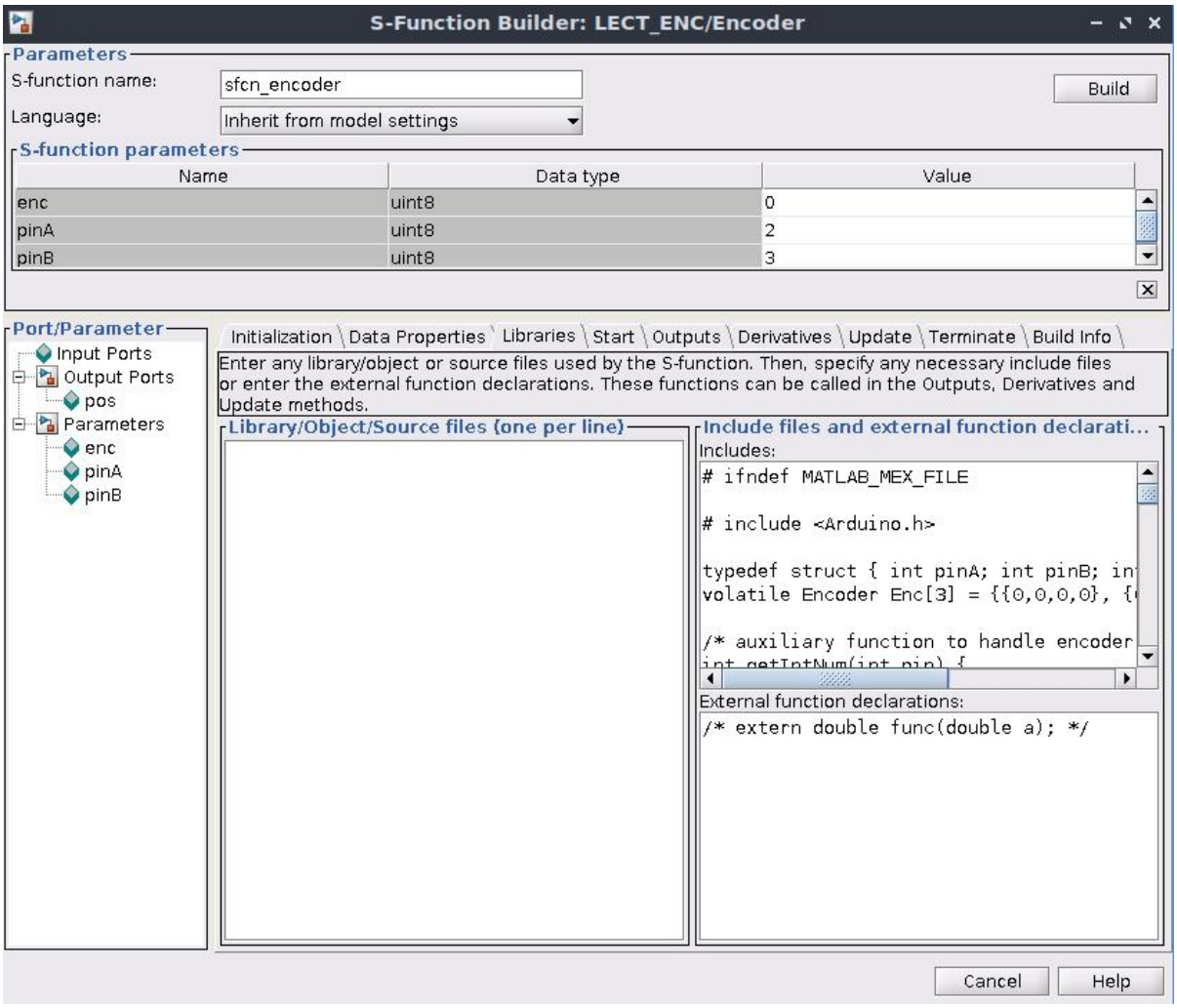

Figure 72: Configuración de la pestaña Libraries
Fuente: Autoría Propia

Pasamos a la pestaña *Outputs* en donde se ingresan unas pocas líneas de código. Las pestañas *Start, Derivates, Terminate* se dejan así tal cual se encontraron.

```
/* wait until after initialization is done */
if (xD[0]==1) {
    /* don't do anything for mex file generation */
    # ifndef MATLAB_MEX_FILE

        /* get encoder position and set is as output */
        pos[0]=Enc[enc[0]].pos;

    # endif
```

```
10      }
```

Como se puede apreciar en la Figura 73, en esta pestaña se configura los estados discretos y continuos de nuestro algoritmo.

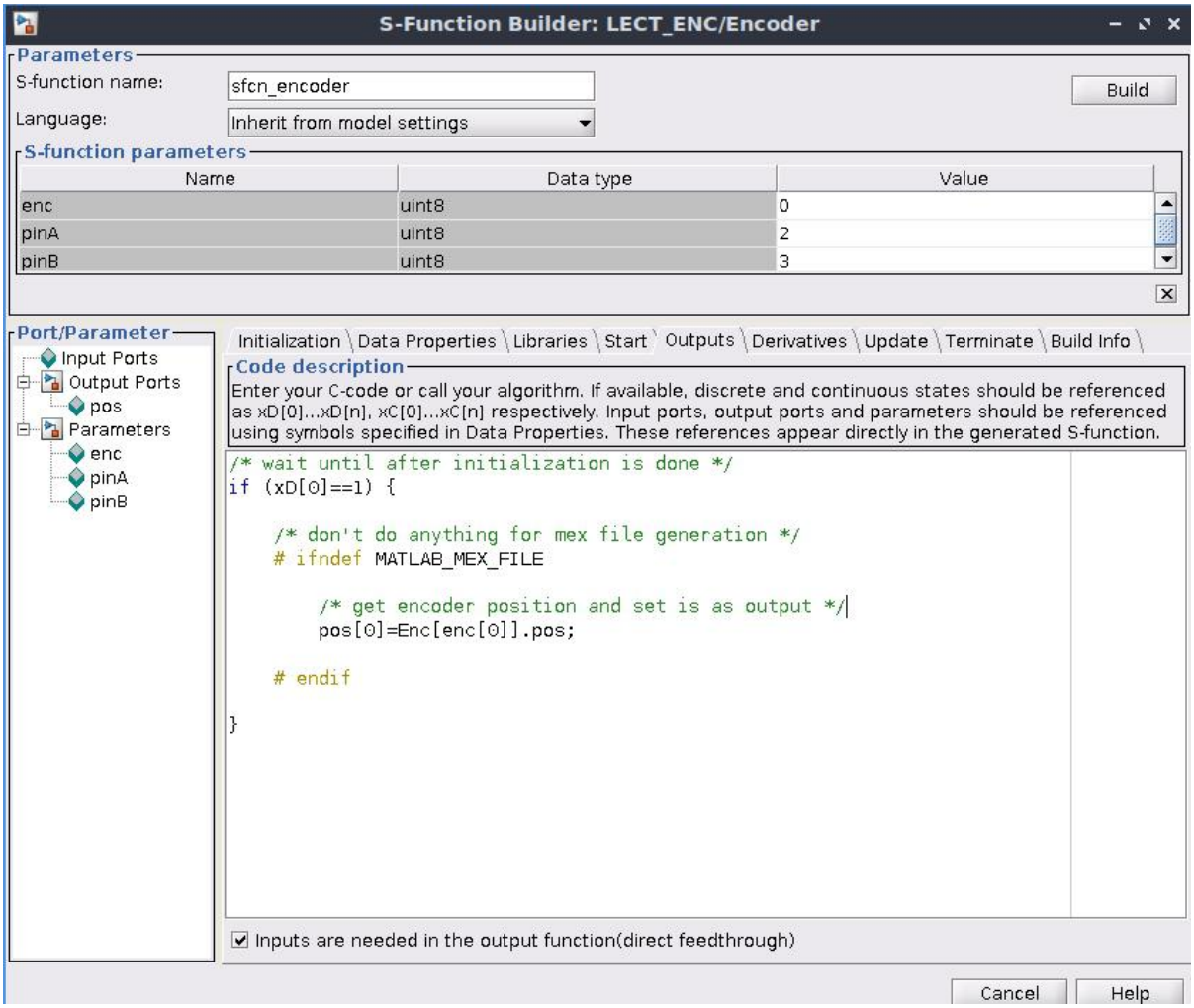

Figure 73: Configuración de la pestaña Outputs
Fuente: Autoría Propia

Esta sección es opcional y se utiliza para actualizar los estados discretos. En la Figura 74, se puede observar la pestaña *Update*, en la cual se ingresa el siguiente códgo mostrado a continuación. Solo se llama si la función S tiene uno o más estados discretos. Los estados de la función S son de tipo doble y deben ser referenciados como XD [0] ... XD [N]. Los puertos de entrada, los puertos de salida y los parámetros deben ser referenciados usando símbolos especificados en las propiedades de datos. Estas referencias aparecen directamente en la función S generada.

```
1 if (xD[0]!=1) {
2
```

```
    /* don't do anything for MEX-file generation */
    # ifndef MATLAB_MEX_FILE

        /* enc[0] is the encoder number and it can be 0,1 or 2   */
        /* if other encoder blocks are present in the model      */
        /* up to a maximum of 3, they need to refer to a         */
        /* different encoder number                              */

        /* store pinA and pinB in global encoder structure Enc   */
        /* they will be needed later by the interrupt routine    */
        /* that will not be able to access s-function parameters */

        Enc[enc[0]].pinA=pinA[0];      /* set pin A              */
        Enc[enc[0]].pinB=pinB[0];      /* set pin B              */

        /* set encoder pins as inputs                            */
        pinMode(Enc[enc[0]].pinA, INPUT);
        pinMode(Enc[enc[0]].pinB, INPUT);

        /* turn on pullup resistors                              */
        digitalWrite(Enc[enc[0]].pinA, HIGH);
        digitalWrite(Enc[enc[0]].pinB, HIGH);

        /* attach interrupts                                     */
        switch(enc[0]) {
          case 0:
            attachInterrupt(getIntNum(Enc[0].pinA), irsPinAEn0, CHANGE);
            attachInterrupt(getIntNum(Enc[0].pinB), isrPinBEn0, CHANGE);
            break;
          case 1:
            attachInterrupt(getIntNum(Enc[1].pinA), irsPinAEn1, CHANGE);
            attachInterrupt(getIntNum(Enc[1].pinB), isrPinBEn1, CHANGE);
            break;
          case 2:
            attachInterrupt(getIntNum(Enc[2].pinA), irsPinAEn2, CHANGE);
            attachInterrupt(getIntNum(Enc[2].pinB), isrPinBEn2, CHANGE);
            break;
          }

        # endif

    /* initialization done */
    xD[0]=1;
}
```

Por último debemos dar sobre el botón *Build* para así poder dar por finalizada la creación de nuestro bloque funcional y utilizarlo en el espacio de trabajo de Simulink.

En la pestaña *Build Info* se pueden apreciar las resultados obtenidos o dicho de otra forma un diagnostico del proceso de compilado con todos los posibles errores que llegasen a ocurrir durante la compilación del mismo. después de realizada la compilación del bloque funcional

creado a nuestra conveniencia.

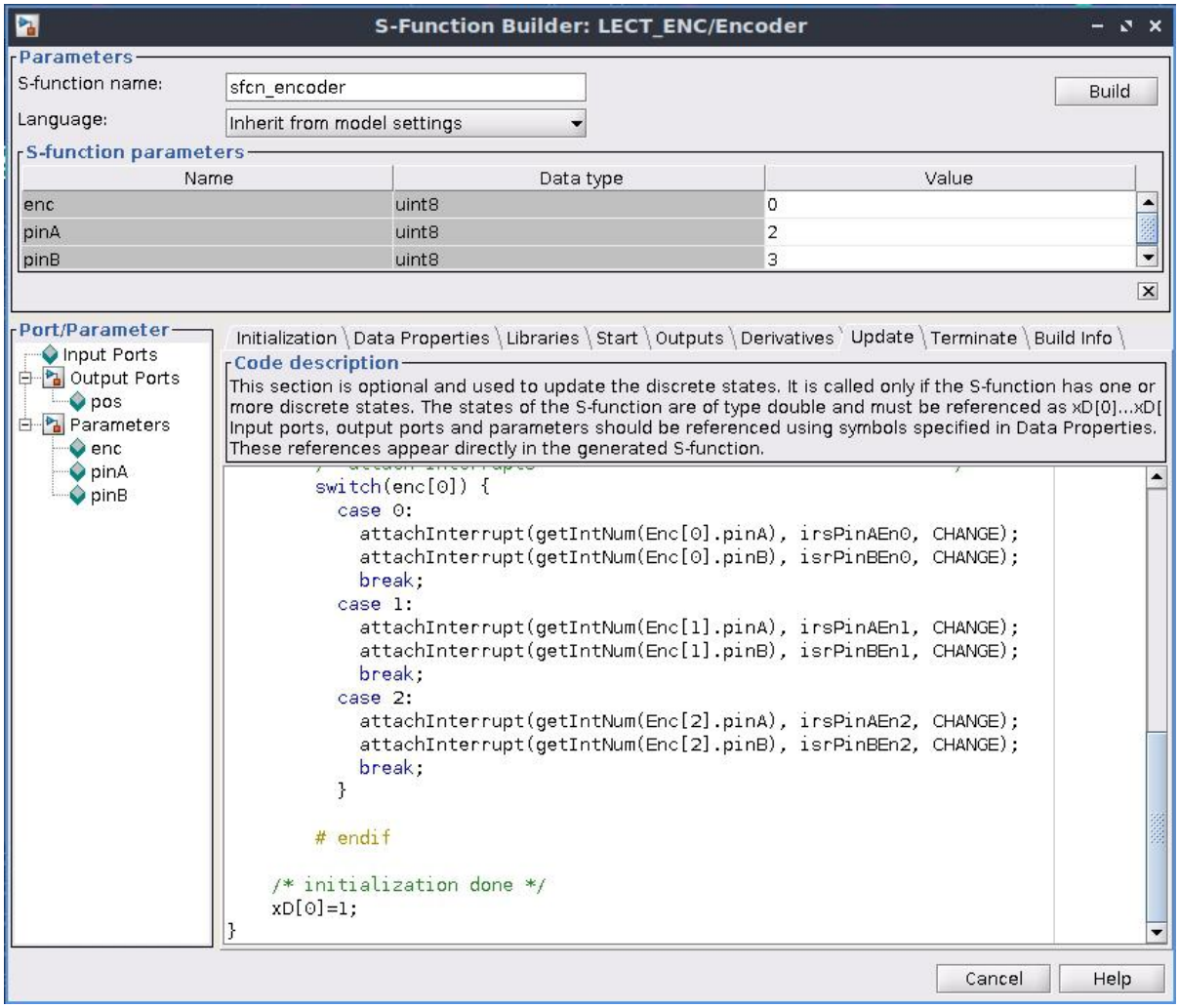

Figure 74: Configuración de la pestaña Update
Fuente: Autoría Propia

7.6 Conexión entre Arduino - MatLab con un driver TTL - Usb diferente al ATMEGA16u2

Es normal que al tratar de subir un programa desde Simulink o MatLab a nuestro sistema embebido Arduino este presente problemas al momento de realizar la conexión entre estos. Por ello a continuación se explicarán un par de ajustes en caso de que la conexión con nuestro MatLab-Arduino falle y más especificamente si estamos usando una tarjeta Arduino con un chip TTL diferente al ATMEGA16u2. Cabe mencionar que la versión de MatLab con la que se probó este método de configuración fue la R2018b 64-bit.

Lo primero es iniciar por configurar nuestro entorno en Simulink para poder establecer la conexión serial entre el entorno y la tarjeta Arduino. Para este ejemplo se usará la placa Arduino Uno, pero sirve para otros tipos de Arduino.

Normalmente las tarjetas que usan un chip, un integrado diferente al ATmega 16u2, no son capaces de realizar una comunicación serial a más de cierta cantidad de baudios, por lo tanto es importante que para poder utilizar dicha placa se configure esta opción.

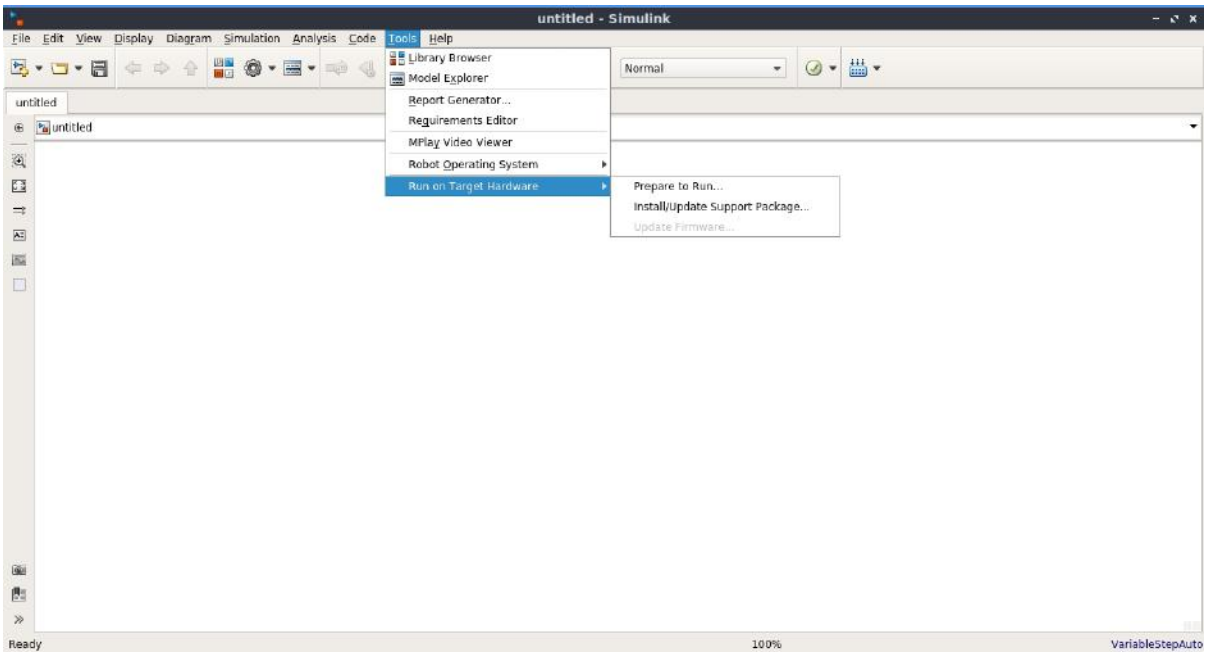

Figure 75: Estableciendo las configuraciones para la conexión en serie con Arduino
Fuente: Autoría Propia

Lo primero es ingresar al apartado *Tools/Run on Target Hardware/Prepare to Run*, una vez allí se mostrará una ventana como se muestra en la Figura 76, en donde se escogerá la tarjeta a utilizar.

Una vez realizado lo anterior, se pasa al apartado *solver*, tal como se muestra en la Figura 77, allí también se configuran opciones como el tiempo de inicio y parada de la simulación, el tipo de solver a a implementar, ya sea de paso fijo (*Fixed-step*) o de paso variable (*Variable-*

step), también podemos configurar el *Sample time*, o dicho de otra forma el tiempo de muestreo para los cálculos que se realizan en el sistema a implementar. Este tiempo es importante ya que al hacerse más cercano a cero, mejor respuesta tendrá nuestro sistema, tarjeta embebida dentro de la simulación. Si el programa llegase a ser muy complejo y nos arrojase error al realizar la conexión arduino - simulink, es posible solucionar este inconveniente aumentando este tiempo de sampleo.

Para poder realizar la conexión de un arduino con un chip o driver diferentes al ATMEGA16u2, lo más sencillo sería antes de empezar cualquier proyecto, conseguir un arduino que cumpla estas características y así evitar cambiar o configurar partes del código de funcionamiento de nuestro dispositivo. Pero dado el caso de no tener cómo cambiar el tipo de conexión serial, se puede a acceder a configurar los baudios a los que se trabajan en dicha conexión.

Por ejemplo, si vamos a la configuración de los baudios con los que se conecta nuestra tarjeta debemos configurarlos correctamente. Con el siguiente código podemos verificar cuál es la velocidad actual en baudios a la cual se conecta nuestro dispositivo:

```
codertarget.arduinobase.registry.setBaudRate (gcs)
```

Al ingresar el comando anterior en el *Command Window* de MatLab nos tiene que arrojar un valor en baudios (*Baud rate is 921600*) como por ejemplo 921600 que es un valor por defecto configurado en MatLab. El inconveniente está en que los drivers diferentes al ATMEGA16u2 no soportan esa velocidad de comunicación, para ello se debe cambiar este valor por uno de menor tamaño, hasta encontrar el que se nos pueda ajustar a nuestras opciones.

Para este ejemplo voy a poner un valor de 115200, aunque es importante probar con qué valor nuestro arduino es capaz de trabajar.

A continuación se muestra el código para cambiar el valor por defecto de baudios.

```
codertarget.arduinobase.registry.setBaudRate (gcs,115200)
```

Es importante conocer este tipo de configuraciones para no quedarnos varados de ninguna manera al momento de ejecutar nuestros proyectos. También ser concientes que al momento de realizar y probar proyectos con un sistema embebido como arduino este posee unas serias limitaciones si hablamos en términos de velocidad de procesamiento de datos y comunicación. Por tanto es fundamental conocer otros sistemas, realizar sus respectivas comparaciones y juicios con respecto a cuál es el más idóneo a trabajar, así como cuál será su rendimiento junto a la plataforma de MatLab-Simulink, aunque es de reconocer que el trabajo con MatLab-Simulink está enfocado hacia otras áreas, se puede llegar a trabajar cómodamente usando sus herramientas.

La correcta ejecución de las librería dentro de un sistema, da como resultado una correcta interpretación y análisis para un sistema dado. Esta librería nos provee de estas ventajas ante un mecanismo complejo y su correcta implementación es crucial al momento de realizar pruebas a estos sistemas. No es tan importante el saber programar, es más importante aún saber leer e

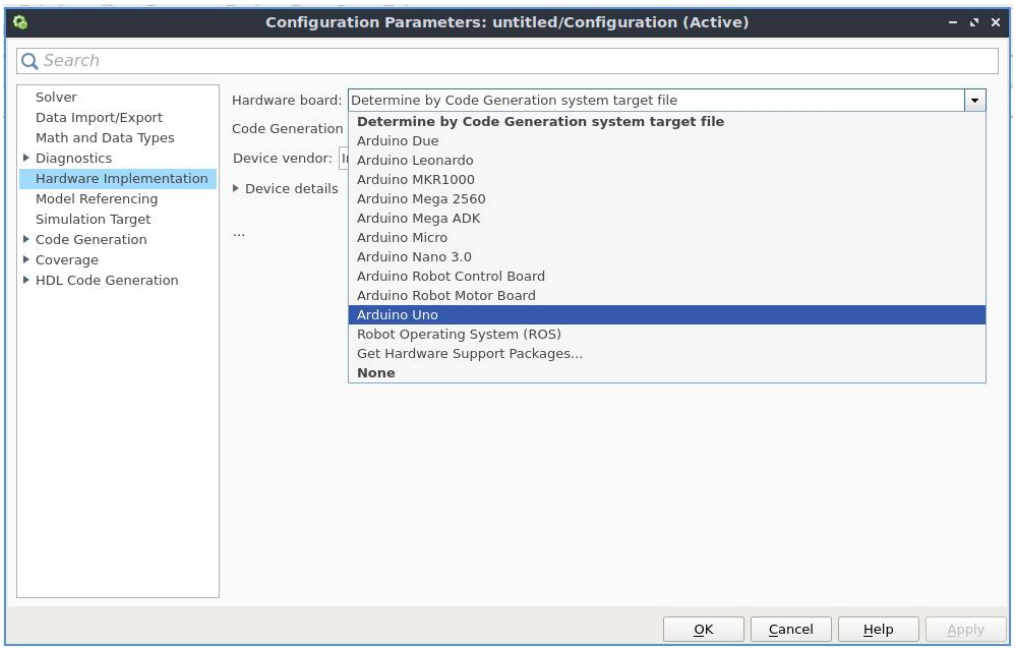

Figure 76: Escogiendo tarjeta para la conexión
Fuente: Autoría Propia

interpretar los programas ya realizados, debido al tiempo que pasamos leyendo código de otras personas.

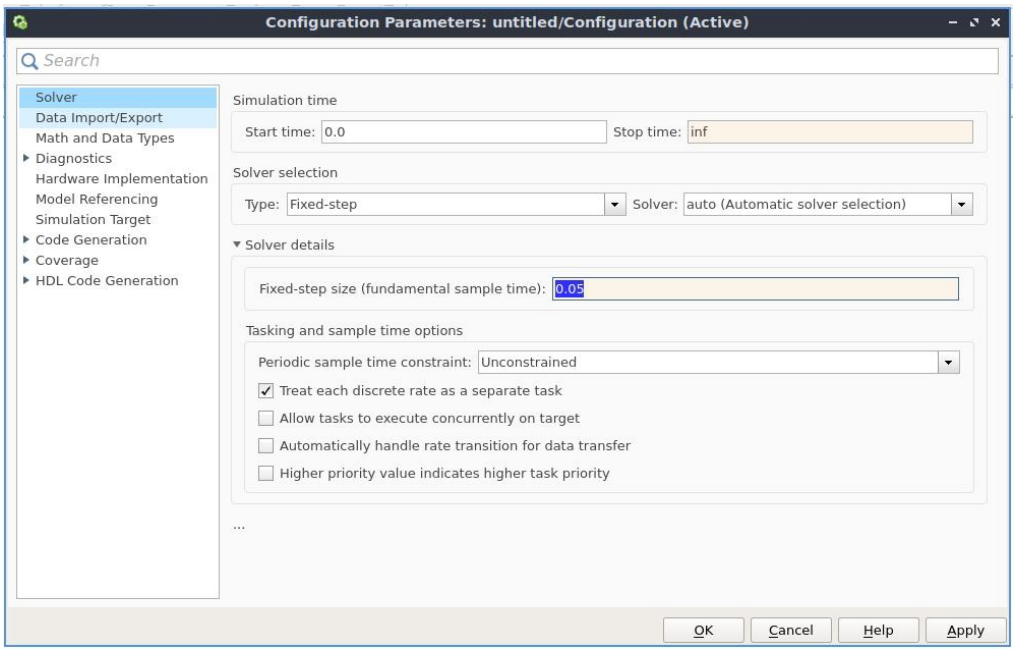

Figure 77: Configuración *solver*
Fuente: Autoría Propia

7.7 Síntesis

Se presentaron los resultados, como son, fotos del sistema físico real, resultados de las simulaciones realizadas en simulink de los diferentes sistemas controlados. El controlador PID presenta una respuesta adecuada en las simulaciones y reacciona muy bien ante perturbaciones creadas a partir de una serie de impulsos. El controlador difuso es el que mejor responde a estos impulsos y su respuesta es más limpia que el del controlador PID. Para la red neuronal artificial sobra decir que es excelente para los problemas de clasificación y en este caso más específico para la emulación o imitación de un controlador para el balanceo en un péndulo rotacional invertido.

8 Conclusiones

Un controlador cuyo propósito es el de disminuir el error presente en la lectura de la variable de la planta que precisa ser controlada con respecto a un valor deseado. En el caso de los controladores utilizados para el desarrollo y construcción de un sistema de péndulo rotacional invertido, existen dos propósitos diferentes dos tareas a cumplir, se puede decir que existen dos tipos de controladores en este trabajo de grado, pero que si se juntan los de un tipo con los del otro se obtiene una meta en común, controlar de manera global un péndulo rotacional invertido o péndulo de Furuta.

Como objetivos para este proyecto se cumplió con la comprensión y la formulación de un modelo matemático el cual representa de manera coherente el comportamiento de la dinámica del péndulo rotacional invertido. También se desarrollaron diferentes técnicas de control las cuales cubren las áreas de control regulatorio y control inteligente, a saber, control PID, control difuso, control por balanceo o Swing Up y RNA. Se desarrolló de manera exitosa un prototipo virtual CAD que permitió dar una visión más precisa a cerca del comportamiento dinámico del sistema. También se construyo, se ensambló y se puso a prueba el prototipo de péndulo rotacional invertido. Se abalizaron los resultados y se consolidó material instructivo para el manejo del módulo.

Como objetivos para este proyecto se cumplió con el desarrollo de un péndulo rotacional invertido y se aplicaron los diferentes tipos de control sobre el mismo. Se estableció el modelo matemático del péndulo rotacional invertido y también para el sistema de motor dc, se desarrolló el sistema virtual CAD se simuló con la ayuda de las ecuaciones dinámicas de los sistemas y se construyó un prototipo para la ejecución de pruebas.

El controlador Swing Up encargado de balancear el péndulo hasta una zona en la que los controladores lineales de salida y entrada única tuvieran razón de ser, al igual que su homólogo el controlador de redes neuronales artificiales cumplen su objetivo a cabalidad. Se encontraron controladores que tienen la capacidad de actuar frente a sistemas altamente lineales y un controlador que no precisan de un sistema altamente lineal a controlar como lo es el controlador Swing Up. Este controlador precisa de una calibración al momento de ajustar los parámetros de giro del motor lo que le da la fuerza al mismo para realizar un correcto balanceo del péndulo.

Se observó la capacidad que poseen los sistemas de redes neuronales artificiales para llevar a cabo tareas y ejercicios de emulación o imitación de mecanismos de control, en este caso para un controlador de tipo Swing Up para un péndulo rotacional invertido.

Con respecto a los controladores SISO (Single Input Single Output), es decir, entrada única y salida única como son el controlador PID y controlador fuzzy, cumplen como se esperaba la tarea a cumplir, aunque se mostró más eficaz el controlador difuso con respecto a las perturbaciones generadas en estado estacionario. Estos controladores funcionan mejor dentro de una rango entre más o menos diez (10) grados o cero punto dieciocho (0.18) radianes para el péndulo en posición de equilibrio inestable. El método de sintonización para un controlador de tipo

PID se puede realizar de manera estricta mediante métodos detallados de sintonización como el método de ajuste de Cohen Coon, método de sintonización de Ziegler - Nichols en lazo abierto o cerrado, método de Smith entre otros, algunos de estos sistemas de sintonización son para sistemas altamente lineales en lazo abierto lo que no sería correcto de emplear en nuestro péndulo rotacional invertido al tratarse de un sistema altamente no lineal, para ello se recurrió a métodos experimentales con ayuda de las simulaciones.

Con respecto a la sintonización del controlador difuso comúnmente no se habla de una sintonización per se, más bien se hace referencia a esta sencillamente hablando de un ajuste en las reglas del sistema después que sea analizada su salida dentro del sistema de lazo cerrado.

Con respecto a la construcción del péndulo rotacional invertido se logró el diseño, construcción y ensamble del mismo con relativamente pocos materiales, se espera inspirar a más estudiantes a la construcción de prototipos mecatrónicos que sean un reto al oficio mismo como lo es la ingeniería mecatrónica. Las simulaciones realizadas al mecanismo con ayuda del microcontrolador Arduino requieren de una respuesta más rápida por parte del microcontrolador se recomienda usar un sistema embebido con mayor capacidad de computo para esta tarea a cabalidad, sin embargo el uso del sistema embebido de Arduino Uno es suficiente para realizar experimentos sobre el módulo, se realizaron diferentes pruebas con sistemas embebidos diferentes al Arduino Uno, como Arduino Mega, Modulo Stm32f103c aunque el problema con este último fue la falta de compatibilidad con el entorno de trabajo de Simulink.

Con respecto a las perturbaciones se realizaron las mismas tanto para el controlador PID como para el controlador difuso, cabe mencionar que el control difuso es el más adecuado al tratar con estas perturbaciones al tratarlas de manera rápida y casi imperceptibles dentro del análisis a la salida de este sistema. Al aplicar perturbaciones a la sección de control de balanceo del péndulo el resultado siempre es una aumento del tiempo del péndulo en llegar a la zona de alta linealidad.

Las grandes diferencias que existen entre los controladores usados a lo largo de este documento pueden ser divididas en dos, los controladores para zonas de alta linealidad y los controladores para zonas de alta no linealidad, su manera de diseñarlos son indiferentes al objetivo común del sistema completo en sí, el control global de un péndulo rotacional invertido no precisa de un solo controlador, precisa la forma correcta de afrontar el problema y dar solución al mismo.

Como futura mejora para el sistema de péndulo rotacional invertido, es necesario añadir un sensor de posición angular para la base o más especifico para el motor, se sugiere el uso de un controlador por variables de estados el cuál permite una manipulación de la posición de equilibrio del péndulo y la posición angular del motor en la base del sistema. También sería indispensable añadir dentro del módulo un slip ring que impida el enredo del cable del encoder del propio módulo, se intentó la creación de uno para el sistema no con mucho éxito.

References

Åström, K. J. and Furuta, K. (1996). Swinging up a pendulum by energy control. *IFAC proceedings volumes*, 29(1):1919–1924.

Avelar, C. A. A. (2013). Control del pendulo de furuta usando la tecnica de linealizacion por retroalimentacion.

Bill Messner, D. T. (2021). Control tutorials for matlab and simulink. Accedido en 1-04-2021 a url http://ctms.engin.umich.edu/.

Buitrago, D. C., Valverde, N. A., and Guarnizo, J. G. (2013). Diseño e implementación de un controlador difuso para un péndulo invertido.

Chen, Y.-F. and Huang, A.-C. (2014). Adaptive control of rotary inverted pendulum system with time-varying uncertainties. *Nonlinear Dynamics*, 76(1):95–102.

Choi, J. J. and Kim, J. S. (2003). Robust control for rotational inverted pendulums using output feedback sliding mode controller and disturbance observer. *KSME international journal*, 17(10):1466–1474.

Cruz, P. P. (2011). *Inteligencia artificial con aplicaciones a la ingeniería*. Alfaomega.

Duarte, J., Montero, B., Ospina-Henao, P., and González, E. (2016). Mecánica lagrangiana para el modelado dinámico y simulación mediante simulink-matlab de un péndulo de furuta. In *Congreso Internacional de Ciencias Básicas e Ingeniería, Villavicencio, Colombia*.

Escobar-Dávila, L. F., Montoya-Giraldo, O. D., and Giraldo-Buitrago, D. (2013). Control global del péndulo de furuta empleando redes neuronales artificiales y realimentación de variables de estado. *TecnoLógicas*, pages 71–94.

Furuta, K., Yamakita, M., and Kobayashi, S. (1991). Swing up control of inverted pendulum. In *IECON*, volume 91, pages 2193–2198.

Gäfvert, M. (2016). *Modelling the furuta pendulum*. Department of Automatic Control, Lund Institute of Technology (LTH).

García, G. T., Molina, J. T. B., and Ramírez, J. M. (2015). Control de un péndulo de furuta. una revisión del estado del arte. *Scientia et technica*, 20(4):322–330.

Giraldo, O. D. M., Hernández, J. G. V., and Buitrago, D. G. (2012). Control global del péndulo rotacional invertido empleando modelos de energía. *Scientia et technica*, 1(52):16–25.

Gonzalez Fontanet, J. G., Lusson Cervantes, A., and Bausa Ortiz, I. (2016). Alternativas de control para un péndulo de furuta. *Revista Iberoamericana de Automática e Informática Industrial*, 13(4):410–420.

Hernández, J. G. V., Giraldo, O. D. M., and Buitrago, D. G. (2013). Lógica difusa aplicada al control local del péndulo invertido con rueda de reacción. *Scientia et technica*, 18(4):623–632.

Jain, A., Tayal, D., and Sehgal, N. (2013). Control of non-linear inverted pendulum using fuzzy logic controller. *International Journal of Computer Applications*, 69(27).

Juan, Z., Jie, C., and Lingxun, D. (2007). Research on control of rotary inverted pendulum via polytope techniques. In *2007 IEEE International Conference on Control and Automation*, pages 2885–2889. IEEE.

Kassir, E. E. (2015). Sistemas de control difuso. *Eslava Zuluaga, AF*.

La Hera, P. X., Freidovich, L. B., Shiriaev, A. S., and Mettin, U. (2009). New approach for swinging up the furuta pendulum: Theory and experiments. *Mechatronics*, 19(8):1240–1250.

Mamani Churayra, E. A. (2012). Diseño y control en tiempo real del sistema subactuado pendubot.

Mandic, P., Lazarevic, M., Stojanovic, S., and Ristanovic, M. (2014). Real time control of rotary inverted pendulum. *Annals of the Faculty of Engineering Hunedoara*, 12(2):211.

Ogata, K. (2010). *Ingeniería de Control Moderna (5 ta Ed.)*. Madrid: Pearson Educación.

Orduñez Ruiz, M. (2010). *Simulación dinámica de Máquinas de corriente directa*. PhD thesis, Universidad Central" Marta Abreu" de Las Villas.

Osorio Zúñiga, C. A. et al. (2009). Diseño, construcción y control de un péndulo invertido rotacional utilizando técnicas lineales y no lineales/design, construction and control of a rotational inverted pendulum using linear and nonlinear techniques. *Departamento de Ingeniería Eléctrica y Electrónica*.

Quanser (2021). Rotary inverted pendulum. Accedido en 30-03-2021 a url https://www.quanser.com/products/rotary-inverted-pendulum/.

Regalo Núñez, C. (2016). Control y simulación del péndulo de furuta.

Sukontanakarn, V. and Parnichkun, M. (2009). Real-time optimal control for rotary inverted pendulum. *American journal of applied sciences*, 6(6):1106.

Valera, A., Vallés, M., and Cardo, M. (2002). Desarrollo y control de un péndulo de furuta. *Dpto. Ingeniería de Sistemas y Automática. Universidad Politécnica de Valencia. Camino de Vera*, 14:46022.

Wang, H. O., Tanaka, K., and Griffin, M. F. (1996). An approach to fuzzy control of nonlinear systems: Stability and design issues. *IEEE transactions on fuzzy systems*, 4(1):14–23.

Yeh, Z.-M. and Li, K.-H. (2004). A systematic approach for designing multistage fuzzy control systems. *Fuzzy Sets and Systems*, 143(2):251–273.

Anexos

Guía de laboratorio para el desarrollo de un controlador PID sobre un péndulo rotacional invertido

Introducción

A continuación se muestra una guía para el desarrollo de un controlador PID sobre un péndulo rotacional invertido el cuál sirve para la implementación de controladores de todo tipo y ampliar la base de conocimientos que posee el usuario de esta guía.

Material y equipo

- Modulo péndulo rotacional invertido
- Multímetro para la medición de voltaje
- Computador con una versión reciente de MatLab instalada

Objetivos

- Familizarizar al usuario con el módulo de péndulo rotacional invertido.
- Realizar experimentos en los que se incluyan al módulo como planta y varios controladores
- Comparar los resultados obtenidos de los controladores implementados

Descripción de equipo y materiales

Péndulo rotacional invertido

El péndulo de furuta o péndulo rotacional invertido, el cual está compuesto por un péndulo de libre movimiento el cual no esta sujeto a ningún actuador, pero si debe estar sujeto a un sensor el cual realimentará un sistema de control en todo momento.

Multímetro para la medición de voltaje

Este dispositivo nos permite la medición directa de fuentes de voltaje tanto en corriente directa como en corriente alterna. Sabiendo lo anterior se pueden evitar accidentes al momento de manipular el módulo verificando que el voltaje alterno sea el correcto al momento de conectarlo a la energía alterna.

Computador con MatLab y Simulink instalados

Con ayuda de la interfaz de programación de simulink se establecen las pruebas para la correcta manipulación.

Actividad 1

Antes de realizar cualquier actividad con el péndulo rotacional invertido el estudiante debe estar familiarizado o debe consultar los siguientes ítems: Manipulación de pines PWM, digitales, análogos con Arduino - Simulink, controlr de motor dc con puente H, controlador, planta, lazo de control abierto y cerrado, sistema embebido, sistema subactuado, linealidad, sistema altamente no lineal, controlador PID, sintonización de controlador PID, conexión entre Simulink y sistema embebido, control swing up, creación de lazos de control en Simulink, identificación de sistemas dinámicos.

Actividad 2

Una vez estudiado con conceptos de la Actividad 1, proceda a realizar una identificación del péndulo rotacional invertido y sus partes a simple vista.

- Tipo de sistema, lineal o no lineal.
- Controladores a usar para controlar el sistema y hacer que pase de posición de reposo a posición de equilibrio invertida.
- Señale el actuador, sensor, sistema embebido, grados de libertad, si es un sistema subactuado o no.

Actividad 3

En esta actividad el estudiante deberá establecer un entorno de adquisición de datos para hallar una función de transferencia que modele el péndulo en posición de equilibrio invertido y compare los resultados de la función de transferencia obtenida de manera matemática para la zona de alta linealidad del péndulo rotacional invertido. Una vez se tengan los datos establezca una relación entre estos y con ayuda del comando *ident* o *systemIdentification* para analizar estos datos.

Posterior a lo anterior crear un sistema de bloques de funciones en simulink para cada una de las funciones de transferencia obtenidas anteriormente y comparar cada uno de sus resultados.

Actividad 4

Cree un sistema en simulink para la estabilización del péndulo del módulo con ayuda del sistema embebido Arduino, un controlador PID y su correspondiente librería. Para la creación de lo anterior debe tener en cuenta de realizar una adquisición de los datos del sensor del módulo, es decir, el encoder y manipular correctamente el actuador que viene con el sistema.

- El pin 2 y 3 del Arduino Uno, corresponden a los pines que admiten interrupcciones externas.
- Los pines 8 9 del arduino van a manipular el sentido de giro del motor DC y el pin 6 la señal PWM del mismo.
- El módulo debe conectarse a 110V AC para su correcto funcionamiento.