UNIVERSIDAD DE CUENCA

Facultad de Ingeniería

Carrera de Electrónica y Telecomunicaciones

Evaluación del desempeño de dos algoritmos de control en tiempo real usando Arduino y FPGA. Caso de estudio: Péndulo de Furuta

Trabajo de titulación previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones

Autor:

Ismael Mateo Alvarez Serrano

CI: 010575177-0

mateoalvarezserrano@gmail.com

Director:

Ing. Darwin Fabián Astudillo Salinas, PhD

CI: 010390703-6

Cuenca-Ecuador

18-agosto-2021

Resumen

El presente trabajo consiste en el diseño y la implementación de dos algoritmos de control para un péndulo de Furuta. La implementación de estos algoritmos de control se lo realiza en dos plataformas embebidas basadas en el microcontrolador Arduino y en Field Programmable Gate Array (FPGA).

En la primera fase del desarrollo del trabajo se realiza una revisión teórica de las técnicas de control aplicadas al péndulo (Proporcional Integral Derivativo (PID) y retroalimentación de estados). En la segunda fase, se diseña e implementa el controlador PID. La calibración de sus parámetros se lo realiza a través de la técnica heurística de prueba-error.

En la tercera fase se realiza el controlador de retroalimentación de estados. Esta fase empieza con el modelamiento matemático del sistema. Luego se valida el modelo con la ayuda de Matlab. Una vez validado y parametrizado el modelo, se linealiza y se obtiene el vector de retroalimentación con la fórmula de Ackerman.

Estos dos controladores son implementados en las plataformas embebidas Arduino y FPGA. Los datos de los experimentos realizados para estos controladores son adquiridos por medio de una tarjeta de adquisición de datos de National Instruments. Para la interpretación de estos datos, se utiliza Matlab, en donde se digitaliza y se calculan los índices de error promedio, energía promedio y energía total.

Al evaluar los resultados de los experimentos realizados, se obtiene que los dos algoritmos de control cumplen con el objetivo de estabilizar al péndulo. Al comparar los resultados, tanto para el controlador PID como para el controlador de retroalimentación de estados, sus índices indican que la plataforma embebida que mejor controla es la FPGA. Al momento de decidir entre qué algoritmo utilizar para controlar la planta, se debe tener en cuenta la cantidad de información técnica disponible de la misma.

Palabras clave : Arduino. FPGA. PID. Ackerman. Controlador. Matlab. Simulink.

Abstract

This paper is about the design and implementation of two control algorithms for a Furuta pendulum. The implementation of these control algorithms is being done on two embedded platforms based on the microcontroller Arduino and FPGA.

In the first development phase of the work, a theoretical review of the control techniques applied to the pendulum (PID and states feedback) is done. In the second phase, the PID controller is designed and implemented. The calibration of its parameters is done through the trial-error heuristic technique.

In the third phase, the controller of the states feedback is performed. This phase begins with the mathematical modeling of the system. Then, the model is validated with Matlab's help. Once the model has been validated and parameterized, it is linearized, and the feedback vector is obtained with the Ackerman formula.

Both controllers are implemented in the Arduino and FPGA embedded platforms. Data from the experiments that were done for these controllers are acquired by a National Instruments acquisition card. Matlab is used to interpret these data, where are digitized and the average error rates, average energy and total energy are calculated.

After evaluating the results of the experiments, it is found that both control algorithms fulfill the objective of stabilizing the pendulum. When comparing the results for both the PID controller and the status feedback controller, their indices indicate that the best controlling embedded platform is the FPGA. To decide between which algorithm should be used to control the plant, the amount of technical information available must be considered first.

Keywords : Arduino. FPGA. PID. Ackerman. Controller. Matlab. Simulink.

Índice general

Índice de figuras

Índice de tablas

Cláusula de Propiedad Intelectual

Yo, Ismael Mateo Alvarez Serrano, autor de la tesis "Evaluación del desempeño de dos algoritmos de control en tiempo real usando Arduino y FPGA. Caso de estudio: Péndulo de Furuta", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 18 de agosto de 2021

Ismael Mateo Alvarez Serrano
010575177-0

Cláusula de licencia y autorización para publicación en el Repositorio Institucional

Yo, Ismael Mateo Alvarez Serrano en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de titulación "Evaluación del desempeño de dos algoritmos de control en tiempo real usando Arduino y FPGA. Caso de estudio: Péndulo de Furuta", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos. Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 18 de agosto de 2021

Ismael Mateo Alvarez Serrano
010575177-0

Certifico

Que el presente proyecto de tesis: Evaluación del desempeño de dos algoritmos de control en tiempo real usando Arduino y FPGA. Caso de estudio: Péndulo de Furuta, fue dirigido y revisado por mi persona.

Ing. Darwin Fabián Astudillo Salinas, PhD
Director

Certifico

Que el presente proyecto de tesis: Evaluación del desempeño de dos algoritmos de control en tiempo real usando Arduino y FPGA. Caso de estudio: Péndulo de Furuta, fue dirigido y revisado por mi persona.

Ing. Luis Ismael Minchala Ávila, PhD
Co-director

Dedicatoria

Dedico esta tesis a mi padre Geovanny, a mi madre María Augusta y a mi hermana Andrea, quienes siempre han sabido estar ahí para darme un consejo, una palabra de aliento, un regaño o lo que se necesite en esos momentos. De igual manera, dedico esta tesis a mis abuelos, que siempre me han demostrado su cariño y apoyo. A mis amigos Loko, Damián y David que siempre han estado en los buenos y malos momentos de mi carrera universitaria. Y por último dedico esta tesis a toda mi familia, amigos y amigas que siempre han estado pendientes de mi.

Ismael Mateo Alvarez Serrano

Agradecimientos

Primero agradezco a Dios y a la vida, por permitirme llegar hasta aquí con salud y ganas de seguir cumpliendo y planteándome nuevos objetivos a nivel personal y profesional. Segundo agradezco a mi familia por haberme brindado y seguir brindando el apoyo necesario para cumplir los diferentes objetivos planteados en mi vida. Tercero agradezco a Eddy Zúñiga por el apoyo brindado durante la carrera universitaria y sobretodo para el desarrollo de esta tesis. Como olvidarme de agradecer a mi gran amigo y vicepresidente Emilio Oyervide quien hizo que el año al frente de la aso escuela se haga mucho mas fácil y atractivo. Agradezco a la Facultad de Ingeniería de la Universidad de Cuenca que a través de sus docentes supieron transmitirme sus conocimientos y me prepararon para enfrentar esta nueva etapa de mi vida que esta por comenzar. Por último, agradezco a mis directores de tesis por el apoyo brindado para el desarrollo de esta tesis y de esta manera poder cumplir mi sueño y de mis familiares.

Ismael Mateo Alvarez Serrano

Abreviaciones y Acrónimos

APL Arduino Programming Language. 5

CFM Configurable Flash Memory. 6
CLB Configurable logic blocks. 5

FPGA Field Programmable Gate Array. 2–6, 25, 33, 34, 45–48

IAE Integral del Error Absoluto. 19
IDE Integrated Development Environment. 5
IOB In Out Block. 6

LQR Regulador Cuadrático Lineal. 19
LTI Lineal Time Invariant. 13, 16

MIMO Multiple Input Multiple Output. 12

PID Proporcional Integral Derivativo. 2, 3, 7, 9–11, 19, 20, 23–25, 34, 40, 48, 49
PWM Pulse-Width Modulation. 5, 23, 25, 33

USB Universal Serial Bus. 5

Capítulo 1

Introducción

Este capítulo presenta la identificación del problema, alcance y los objetivos del presente proyecto.

1.1. Identificación del problema

Los péndulos invertidos fueron inventados en los años setenta como un objeto de estudio, y en la actualidad se siguen manteniendo para el estudio de sistemas de control no-lineal. Esto gracias a que sus modelos matemáticos presentan algunas similitudes con sistemas más complejos como, por ejemplo, generadores conectados en sincronía, aviones de combate, entre otros [1, 2]. Estos péndulos tienen diferentes modelos desde su creación, siendo estos: Lagranje, oscilador armónico amortiguado, teoría de perturbaciones, péndulo empotrado y péndulo de Furuta [2, 3].

La fase de control en los péndulos invertidos es una etapa crítica, tomando en cuenta que mediante el control manual o automático se busca la estabilidad del sistema [4]. La complejidad de hallar la estabilidad se verá afectada directamente por los parámetros físicos del péndulo, mientras más grados de libertad o mayor peso tenga el sistema, su complejidad aumentará. Para lograr la estabilización automática del sistema se usan dispositivos electrónicos, tales como: sensores, *encoders*, motores y tarjetas de procesamiento [3, 5].

El péndulo de Furuta es un modelo creado por el Dr. K. Furuta, que consiste en sistema subactuado de dos grados de libertad, ambos rotacionales, llamados brazo y péndulo. El movimiento del brazo se realiza en un plano horizontal girando alrededor de un eje perpendicular al plano, mientras que el péndulo se encuentra colocado en un extremo del brazo, su eje de giro es colineal al eje axial del brazo y su movimiento se realiza perpendicular al de este último [5, 6]. La Fig. 1.1 muestra un modelo esquemático del péndulo de Furuta.

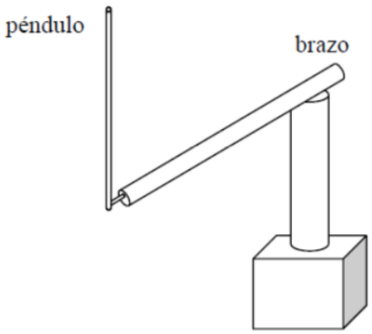


Figura 1.1: Modelo esquemático del péndulo de Furuta. [6]

1.2. Alcance

Se plantea la evaluación del desempeño de algoritmos de control en tiempo real para un péndulo de Furuta. Se parte de la hipótesis que al usar una tarjeta FPGA es posible mejorar el rendimiento en el procesamiento del algoritmo de control PID, en comparación con el desarrollo en una plataforma microcontrolada (Arduino).

Determinar el algoritmo de control, en tiempo real, más eficiente entre las opciones PID y retroalimentación de estados, para un péndulo de Furuta, usando las dos tarjetas de procesamiento (Arduino y FPGA), a través de la comparación del desempeño de los algoritmos de control de las dos plataformas. El algoritmo de control se elige a partir de estudios sistematizados, en simulación e implementación, de las alternativas de control moderno y clásico previamente mencionadas.

1.3. Objetivos

1.3.1. Objetivo general

Diseñar e implementar dos algoritmos de control (PID y retroalimentación de estados) aplicados a un péndulo de Furuta usando plataformas embebidas basadas en el microcontrolador Arduino y en FPGA.

1.3.2. Objetivos específicos

El presente trabajo tiene los siguientes objetivos específicos:

- Implementar el algoritmo de control PID en Arduino y FPGA.
- Implementar el algoritmo de control de realimentación de estados en Arduino y FPGA.
- Comparar los índices de desempeño de energía y error para los dos algoritmos de control.

Capítulo 2

Marco teórico y estado del arte

En esta sección se presentan los conceptos necesarios para poder entender de una forma correcta el desarrollo de esta investigación. En la Sección 2.1, se habla sobre el péndulo de Furuta y los microcontroladores a utilizar para su respectiva implementación (Arduino y FPGA). En la Sección 2.2, se detallan los sistemas de control a utilizar (PID y espacio de estados), y en la Sección 2.3, se indican algunos trabajos relacionados de suma importancia.

2.1. Péndulo de Furuta

El péndulo de Furuta es una planta o sistema muy utilizado en el ámbito educativo, con el fin de representar un sistema sub-actuado de dos grados de libertad, donde se aplican técnicas de control lineal y no lineal para cumplir con la finalidad de estabilizar el péndulo. Se puede comparar este sistema con el balanceo de una escoba en la palma de la mano teniendo como objetivo encontrar su punto de equilibrio y así evitar que la escoba caiga [7].

El primer prototipo del péndulo de Furuta, denominado péndulo TITech, fue desarrollado en 1992 por el Doctor Katsuhisa Furuta en el Instituto de Tecnología de Tokio [7]. El objetivo principal es suplir las limitaciones físicas que se presentan en el péndulo invertido tradicional. Esto lo logra eliminando la limitación de la posición del péndulo permitiéndole tener una trayectoria circular de 360º y girar en ambas direcciones; lo cual libera restricciones y desencadena dinámicas no lineales de un orden superior dificultando el desarrollo del modelo matemático [8].

Los investigadores de diferentes ramas de la ciencia hacen del péndulo de Furuta uno de los sistemas más implementados dentro de la gama de los péndulos invertidos para el estudio de diferentes aplicaciones, entre estas: estabilización de aviones, estabilización de la cabina de un barco, control antisísmico de edificios, y construcción de herramientas mecánicas con un gran número de grados de libertad [9, 10]. Para la estabilización de estos péndulos se usa la teoría de control, en donde se tiene diferentes técnicas tales como: controladores PID, control por realimentación de variables de estado, control Fuzzy más realimentación de variables de estado, control Fuzzy más control predictivo, entre otras [10]. Estas técnicas pueden ser implementadas en tiempo real o en software de simulaciones. En cuestión de este estudio se hace la implementación en tiempo real de dos técnicas de control (controladores PID y control por realimentación de variables de estado) en dos tarjetas de

procesamiento (Arduino y FPGA).

La estructura del péndulo de Furuta consta de dos cuerpos inerciales conectados; un pilar central con momento de inercia J, rígidamente conectado a un brazo horizontal con longitud l_a y masa m_a homogéneamente distribuida en línea, y el péndulo con longitud l_p y masa m_p homogéneamente en línea; donde el ángulo del péndulo θ ha sido definido como cero cuando se encuentra erguido, mientras que el ángulo del brazo ϕ se define como positivo cuando el brazo gira en sentido anti horario y negativo cuando gira en sentido horario [10] (Ver Fig. 1.1).

2.1.1. Arduino

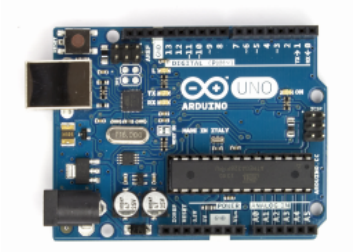


Figura 2.1: Arduino UNO R3. [11]

Arduino nace en el año 2005, donde un grupo de estudiantes del Instituto de Diseño Interactivo de Ivrea ven la necesidad de contar con un dispositivo que funcione en cualquier sistema operativo y sea de bajo coste; bajo este criterio crean una placa de software y hardware con una circuitería interna determinada para uso interno de la escuela, pero el mismo año el Instituto se ve obligado a cerrar sus puertas y se ven en la necesidad de poner en conocimiento y liberar la placa a la comunidad, y permitir que de esta manera puedan participar de la evolución de este proyecto [12].

Desde su entonces se lleva varios desarrollos sobre la placa principal hasta que en el año 2010 hacen el lanzamiento de la primera placa comercialmente conocida como Arduino Uno. Hasta la actualidad esta placa ha sido revisada y mejorada tres veces, lo cual permite tener hoy en día en el mercado la placa Arduino UNO R3 (Ver Fig. 2.1). Sin embargo, se han realizado otros desarrollos paralelamente por parte del equipo de Arduino dando una variedad de placas oficiales de Arduino como [11, 12]:

- Arduino Mega 2560.
- Arduino Mega ADK.
- Arduino Ethernet.
- Arduino Fio.
- Arduino Pro.
- Arduino Lilypad.
- Arduino Nano.
- Arduino Mini.
- Arduino Pro Mini.
- Arduino Leonardo.
- Arduino Micro.
- Arduino Due.
- Arduino Ethernet Shield.

- Arduino Wireless SD Shield.
- Arduino Wireless Proto Shield.
- Arduino WiFi Shield.
- Arduino Motor Shield.
- Arduino Proto Shield.

Todas estas tarjetas microcontroladas de Arduino tienen diferencias, pero la idea principal de su funcionamiento y bajo coste se mantiene. Estas diferencias se ven en las capacidades de almacenamiento, modelos de los microcontroladores (aun siendo de la misma familia ATMEL), velocidades de su reloj, entre otras [12].

El equipo de Arduino ha basado su lenguaje de programación en el lenguaje Wiring llegando a crear su propio lenguaje de programación, siendo este el Arduino Programming Language (APL). El principal objetivo de este lenguaje es disminuir significativamente la dificultad de programar los diferentes algoritmos. El Integrated Development Environment (IDE) de Arduino permite la creación del *sketch* del Hardware que al ser compilada automáticamente el IDE convierte el *sketch* en Código C o C++, posterior a esto lo convierte en código máquina y lo sube al microcontrolador. Esta transferencia se lo realiza por un puerto serial o por el puerto Universal Serial Bus (USB) de la computadora [11].

En este proyecto se usa la placa Arduino UNO R3. En la Tabla 2.1 se presenta un conjunto de especificaciones de esta tarjeta microcontrolada [11, 12].

Tabla 2.1: Características tarjeta Arduino UNO R3. [12]

Microcontrolador	ATMEGA328P
Memoria Flash	32KB
Memoria SRAM	2KB
Memoria EEPROM	1KB
Protocolos de Comunicación	I2C/TWI, SPI
Alimentación	5 Vdc
E/O digitales	14
E/O analógicas	6
Salidas Pulse-Width Modulation (PWM)	6 Pines(3,5,6,9,10,11)

2.1.2. FPGA

Una placa FPGA es un dispositivo configurable que ofrece un arreglo de compuertas lógicas para ser interconectadas de la manera deseada. La estructura de un FPGA se basa en 4 partes principales (ver Fig. 2.2), siendo estas [13]:

- **Configurable logic blocks (CLB):** Conforman parte del hardware que por medio del software se pueden configurar para una tarea específica. Conformado de dos grupos:
  - **Gates:** Son unidades básicas de la electrónica digital que cumplen operaciones booleanas para desarrollar circuitos digitales, pero al interconectarse entre ellas pueden llegar a desarrollar circuitos digitales complejos.

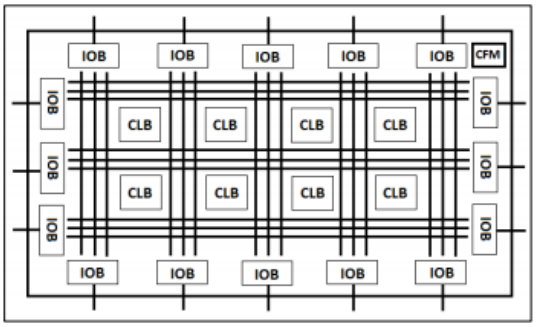


Figura 2.2: Bloques FPGA. [13]

- **Register:** Son unidades de memoria para almacenar información. Estas unidades utilizan un método de round robin; la información antigua es sobre-escrita con la información nueva cuando ya no hay espacio en la memoria.

- **In Out Block (IOB):** Terminales del FPGA que se encargan de enviar o recibir información de o hacia el exterior.
- **Configurable Flash Memory (CFM):** Almacena la configuración de cada bloque del FPGA para cuando se apague y vuelva a iniciar, pueda saber qué configuración tiene.
- **Los wires:** Rutas por las cuales transporta la información.

La placa FPGA, como todo dispositivo electrónico, tiene ventajas y desventajas [13, 14], pero de una manera generalizada se tiene lo siguiente:

- Ventajas.
  - Paralelismo.
  - Velocidad.
  - Múltiples y reconfigurables pines I/O.
- Desventajas.
  - Costo elevado.
  - Consumo de energía.
  - Complejidad.
  - Incompatibilidad.
  - Lenguaje de programación complicado.

En el mercado existen dos tipos: los FPGA comerciales y los FPGA libres. Estos difieren en que el costo y las prestaciones. Los FPGA comerciales tienen mejores prestaciones que los FPGA libres, pero involucran una mayor complejidad al momento de utilizarlos. Dentro de la categoría de los FPGA libres se encuentra la tarjeta Alhambra II (Ver Fig. 2.3), misma que se usa para el desarrollo de esta investigación [14].

La FPGA libre Alhambra II fue diseñada por Eladio Delgado con la idea de que sea fácil de usar y programar por parte de los usuarios y de esta manera desarrolló una placa similar a la de Arduino, permitiendo de esta manera conectar elementos electrónicos externos de una manera fácil [13, 14].

Las características del FPGA Libre Alahambra II se presentan en la Tabla 2.2.

Figura 2.3: FPGA Alahmbra II. [13]

Tabla 2.2: Características tarjeta FPGA Alhambra II. [13]

CHIP	ICE40HX4K-TQ144
Memoria Flash	32 MB
Oscilador	12 MHz
Conectores I/O (3,3V hasta 5V)	20
LED´s de Usuario	8
Pulsador reset	Si
Interruptor ON/OFF	Si
Pulsadores de usuario	2
Pines I/O	206

2.2. Sistemas de control

En el siglo XVIII, el regulador de velocidad centrifuga de James Watt es considerado como el primer trabajo significativo de control automático y a partir de ello se dan aportes significativos para el desarrollo de la teoría de control; uno de ellos se da en 1932 cuando Nyquist plantea un procedimiento para determinar la estabilidad de un sistema a lazo cerrado a partir de la respuesta en lazo abierto. El próximo gran avance se da en los años cuarenta cuando los diagramas de Bode hacen posible el diseño de sistemas de control lineales en lazo cerrado que cumplen los requisitos de estabilidad. Durante la década de los cuarenta también se empieza el uso de los controladores y muchas industrias hacen uso de un controlador PID industrial con el fin de controlar diferentes variables como la temperatura, velocidad, presión, entre otras. La sintonización de los controladores PID se los realizaba en base a las reglas de sintonía de Ziegler-Nichols, presentadas a inicios de los años cuarenta. A inicios de los cincuenta se presenta el método del lugar de las raíces que en conjunto con el método de la respuesta en frecuencia, crean la base de la teoría de control clásica [15].

Con el avance de la tecnología, la teoría de control clásica queda obsoleta debido a que esta cumple para sistemas con una entrada y una salida, pero las plantas modernas empiezan a presentar múltiples entradas y múltiples salidas derivando en sistemas lineales o no lineales. Esto significa que su grado de complejidad a nivel de ecuaciones aumente, y gracias a la disponibilidad de las computadoras digitales, en 1960 se de inicio a la teoría de control moderno. Esta teoría se basa en el análisis en el dominio del tiempo y la síntesis a partir de variables de estado, permitiendo de esta manera manejar la complejidad creciente en las plantas modernas y los requisitos cada vez más estrictos [15, 16].

La teoría de control moderno define a un sistema de control como “Cualquier interconexión de componentes que satisfacen una función deseada”; donde su objetivo principal es controlar las salidas mediante las entradas a través de diferentes sistemas de control (controladores) [16, 17]. La parte del sistema a ser controlada es conocida como planta o proceso. El sistema de control puede ser de dos tipos [15]:

(a) Lazo Abierto (b) Lazo Cerrado

Figura 2.4: Tipos de Sistemas de Control. [17]

- Sistema de control en lazo abierto.
  - Son sistemas en los cuales la salida no tiene efecto sobre la acción de control (Ver Fig. 2.4a).
  - Entre sus principales ventajas se tiene:
    - Construcción simple.
    - Facilidad de mantenimiento.
    - Menor costo de implementación.
  - Como desventajas se tiene:
    - No es inmune a las perturbaciones.
    - Se necesita hacer re-calibraciones periódicas.
- Sistema de control en lazo cerrado.
  - Son sistemas en los cuales se mantiene una relación entre la salida y la entrada de referencia, comparándolas y usando la diferencia como acción de control. Estos sistemas también son conocidos como sistemas de control realimentados (Ver Fig. 2.4b).
  - Como ventajas se tiene:
    - Relativamente insensible a perturbaciones.
    - Se puede usar elementos poco precisos y baratos.
  - Como desventajas se tiene:
    - Mayor coste de implementación.
    - Más difícil de obtener el punto de estabilidad.

Un sistema de control para poder ser analizado y diseñado, tiene que ser representado a través de su función de transferencia, la misma que no es más que la relación de la salida $y(t)$ respecto a su entrada $r(t)$ en el dominio de Laplace [17] (Ver Eq. (2.1)).

$$T(s) = \frac{Y(s)}{R(s)}\Big|_{\text{con condiciones iniciales} = 0} \tag{2.1}$$

Donde la función de transferencia del polinomio denominador, es el polinomio característico del sistema y es el que contiene las raíces características del sistema [17]. Las raíces del sistema determinan si el sistema es estable o no es estable de acuerdo al concepto de estabilidad, el cual indica que un sistema es estable si todas sus raíces características están a la izquierda del eje imaginario del plano complejo. En la Fig. 2.5 se puede observar un conjunto de ejemplos de sistemas estables y no estables [17].

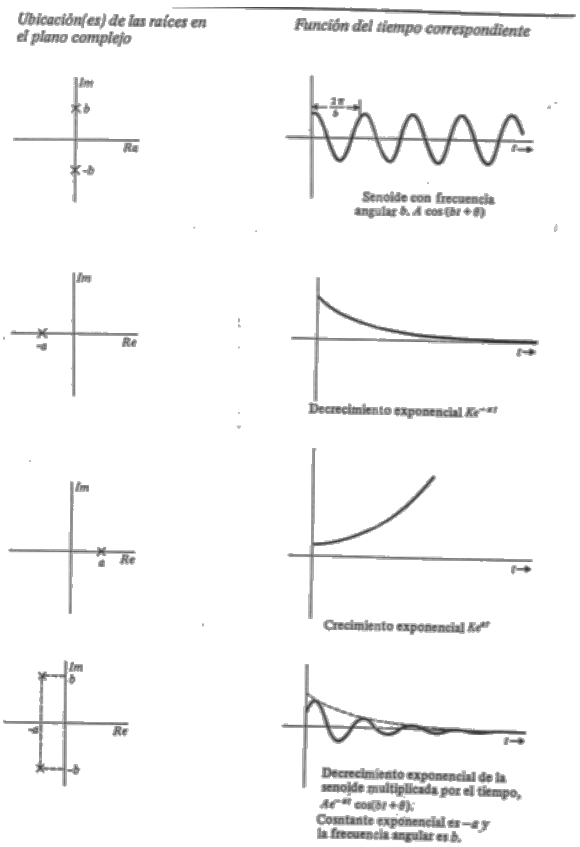


Figura 2.5: Sistemas estables y no estables. [17]

2.2.1. Controlador proporcional-integral-derivativo

El controlador PID es la técnica de control más usada desde su aparición con los diseños de limitadores de velocidad. Esta técnica se basa en un lazo de retroalimentación que permite comparar la salida con el valor deseado a obtener y en función a este error, hacer las correcciones necesarias para obtener la salida deseada. Este algoritmo de control PID integra la acción proporcional, la acción derivativa y la acción integral en un solo sistema [18, 19].

Este controlador se adaptó de una manera eficaz en los diferentes sistemas, debido a que si no se conocen los valores exactos de la planta, por medio de las técnicas de calibración de los controladores PID se puede llegar a tener un control adecuado a las necesidades del sistema [20].

La acción proporcional: Genera una señal de control proporcional a la señal de error (ver Eq. (2.2)).

$$u_p(t) = K_p * e(t) \tag{2.2}$$

Donde $u_p(t)$ es la señal de control de la acción proporcional, K_p es la ganancia proporcional o de lazo y $e(t)$ es la señal de error. La función de transferencia de la acción proporcional está definida en la Eq. (2.3).

$$u_p(s) = K_p \tag{2.3}$$

K_p viene a ser conocida como la ganancia proporcional o la ganancia del lazo de retroalimentación y se puede concluir que esta constante es directamente proporcional a la señal de control.

Una de las desventajas de esta acción es que generalmente cae en un error de estado estacionario diferente de cero, y una forma de evitar esto es que la constante K_p sea elevada y de esta manera el error de estado estacionario disminuirá considerablemente pero no se llegará a eliminar el error completamente [18, 19].

La acción integral: Genera una señal de control proporcional a la integral de la señal de error (ver Eq. (2.4)).

$$u_i(t) = \frac{K_p}{T_i} * \int_0^t e(t)\,dt \tag{2.4}$$

Donde $u_i(t)$ es la señal de control de la acción integral, K_p es la ganancia proporcional o de lazo, T_i es el tiempo integral (ajusta la acción integral) y $e(t)$ es la señal de error. La función de transferencia de la acción integral está definida en la Eq. (2.5).

$$u_i(s) = \frac{K_p}{T_i s} \tag{2.5}$$

Esta acción está enfocada en hacer que la salida concuerde con la referencia en estado estacionario, pero al aplicar esta acción se puede ocasionar inestabilidades en el sistema debido a que implica introducir un polo en el origen a la función de transferencia del sistema a lazo abierto, empeorando de esta manera la respuesta transitoria del sistema. Sin embargo, con la ayuda de una acción proporcional se puede reducir el riesgo de inestabilidad del sistema [18, 19, 21].

La acción derivativa: Genera una señal de control proporcional a la derivada de la señal de error (ver Eq. (2.6)).

$$u_d(t) = K_p * T_d * \frac{d}{dt}e(t) \tag{2.6}$$

Donde $u_d(t)$ es la señal de control de la acción derivativa, K_p es la ganancia proporcional o de lazo, T_d es el tiempo derivativo (ajusta la acción derivativa) y $e(t)$ es la señal de error. La función de transferencia de la acción derivativa está definida en la Eq. (2.7).

$$u_d(s) = K_p * T_d * s \tag{2.7}$$

Esta acción está enfocada en mejorar la estabilidad del sistema en lazo cerrado mediante el conocimiento de las características dinámicas. Gracias a la derivada se puede anticipar a una señal de error excesiva y realizar los ajustes necesarios para no permitir el crecimiento excesivo del error que podría significar la pérdida de estabilidad del sistema. Este tipo de acción siempre debe ir acompañada de algún otro tipo de acción como una proporcional o una integral [19, 21, 22].

Una vez explicados los parámetros, la señal de control de un controlador PID está definida por la Eq. (2.8).

$$u_{PID}(t) = u_p(t) + u_i(t) + u_d(t) \tag{2.8}$$

Remplazando las Eqs. (2.2), (2.4), (2.6) en la Eq. (2.8), se tiene la ecuación del controlador PID (ver Eq. (2.9)).

$$u_{PID}(t) = K_p * e(t) + \frac{K_p}{T_i} * \int_0^t e(t)\,dt + K_p * T_d * \frac{d}{dt}e(t) \tag{2.9}$$

Y la función de transferencia del controlador está definida por la Eq. (2.10).

$$u_{PID}(s) = Kp + \frac{K_p}{T_i s} + K_p * T_d * s \tag{2.10}$$

La representación de bloques de un controlador PID se puede observar en la Fig. 2.6. [19]

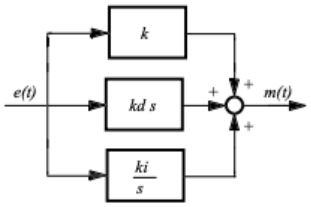


Figura 2.6: Representación en bloque de un controlador PID. [19]

Los valores de las constantes de ganancia proporcional, derivativa e integral tienen que ser determinados para que cumplan el comportamiento deseado de la planta. El proceso de encontrar estas ganancias se lo conoce como la sintonización del controlador PID y para ello existen varias técnicas como [23]:

- Ziegler y Nichols.
- Cohen y Coon.
- López et al.
- Kaya y Sheib.
- Sung et al.

En [20], se presenta un método heurístico de sintonización de los controladores PID. Este método está desarrollado en tres pasos, siendo estos:

1. **Paso 1.-** Acción Proporcional.
   - Tiempo integral (T_i), a su máximo valor.
   - Tiempo derivativo (T_d), a su mínimo valor.
   - Se empieza con una ganancia baja y se aumenta hasta obtener las características de respuestas deseadas.
2. **Paso 2.-** Acción Integral.
   - Reducir el T_i hasta anular el error en estado estacionario, aunque la oscilación sea excesiva.
   - Disminuir ligeramente la ganancia.
   - Repetir hasta obtener las características de respuesta deseadas.
3. **Paso 3.-** Acción Derivativa.
   - Mantener la ganancia y el tiempo integral obtenidos anteriormente.
   - Aumentar el T_d hasta obtener características similares, pero con la respuesta más rápida.
   - Aumentar ligeramente la ganancia si fuera necesario.

2.2.2. Espacio de estados

Debido a que el método de evaluación a través de la función de transferencia presenta ciertas limitaciones, la teoría de control moderno establece un mecanismo diferente de análisis basándose en la teoría del análisis de espacio de estados. De esta manera se puede controlar un grupo más grande de sistemas que no podían ser

evaluados a través de la teoría de control tradicional. Los principales beneficios de hacer un estudio en el espacio de estados son que permite tener sistemas con múltiples entradas y múltiples salidas (Multiple Input Multiple Output (MIMO)), sistemas variantes en el tiempo, sistemas no lineales, adicional a todo lo que con la teoría de control tradicional se podía analizar [24].

El método del análisis en el espacio de estados "Se basa en la descripción del sistema en términos de n ecuaciones en diferencias o diferenciales de primer orden, que pueden combinarse en una ecuación matricial en diferencias o diferencial de primer orden" [25].

Para conocer un poco más de los componentes de un espacio de estados se introduce los siguientes conceptos [24]:

- **Estado:** Es la mínima información de la planta en un instante dado $t = to$, a partir del cual, junto con el conocimiento de la señal de entrada para instantes $t > to$, se puede determinar el comportamiento y evolución de la planta para cualquier instante de tiempo $t > to$.
- **Variables de estado:** Es el conjunto de estados en el instante $t = to$. Estas variables de estado no necesariamente tienen que ser cantidades medibles u observables.
- **Vector de estado:** Es el conjunto de variables de estado que permiten determinar el estado $x(t)$ para cualquier tiempo $t > to$.
- **Espacio de estados:** Espacio n-dimensional donde los estados toman valores y pueden representarse.

Los sistemas de tiempo discreto y de tiempo continuo son modelados sin ninguna variación. La representación de un sistema discreto variante en el tiempo está definido por la Eq. (2.11) y su salida por la Eq. (2.12) [24].

$$\mathbf{x}(k+1) = \mathbf{f}[\mathbf{x}(k), \mathbf{u}(k), k] \tag{2.11}$$

$$\mathbf{y}(k) = \mathbf{g}[\mathbf{x}(k), \mathbf{u}(k), k] \tag{2.12}$$

Mientras que la representación de un sistema continuo variante en el tiempo está definido por la Eq. (2.13) y su salida por la Eq. (2.14) [25].

$$\mathbf{x}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t] \tag{2.13}$$

$$\mathbf{y}(t) = \mathbf{g}[\mathbf{x}(t), \mathbf{u}(t), t] \tag{2.14}$$

Simplificando la Eq. (2.11) y Eq. (2.12) para sistemas lineales variantes en el tiempo, se tienen las Eqs. (2.15), (2.16) [24].

$$\mathbf{x}(k+1) = \mathbf{G}(k)\mathbf{x}(k) + \mathbf{H}(k)\mathbf{u}(k) \tag{2.15}$$

$$\mathbf{y}(k) = \mathbf{C}(k)\mathbf{x}(k) + \mathbf{D}(k)\mathbf{u}(k) \tag{2.16}$$

Donde,

x(k) = vector n	(vector de estado)
y(k) = vector m	(vector de salida)
u(k) = vector r	(vector de entrada)
G(k) = matriz n x n	(matriz de estado)
H(k) = matriz n x r	(matriz de entrada)
C(k) = matriz m x n	(matriz de salida)
D(k) = matriz m x r	(matriz de transmisión directa)

Y para un sistema Lineal Time Invariant (LTI), se tienen las Eqs. (2.17), (2.18). [24]:

$$\mathbf{x}(k+1) = \mathbf{G}\mathbf{x}(k) + \mathbf{H}\mathbf{u}(k) \tag{2.17}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) \tag{2.18}$$

La Fig. 2.7 muestra su representación en diagrama de bloques.

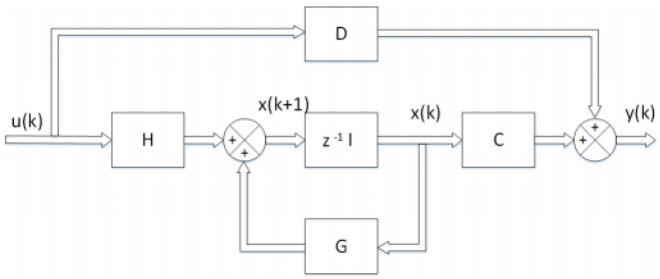


Figura 2.7: Representación en bloque de un Sistema de control LTI. [24]

Una planta no tiene una única representación en el espacio de estados, pues esta puede tomar diferentes representaciones, pero lo que no puede variar es la cantidad de variables de estado. Para la obtención de estas representaciones en el espacio de estados se parte de la consideración de un modelo de sistema definido por su función de transferencia (ver Eq. (2.19)) [25] [26].

$$\frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + ... + b_n z^{-n}}{1 + a_1 z^{-1} + ... + a_n z^{-n}} \tag{2.19}$$

Se divide Eq. (2.19) para z^n y se obtiene la Eq. (2.20):

$$\frac{Y(z)}{U(z)} = \frac{b_0 z^n + b_1 z^{n-1} + ... + b_n}{z^n + a_1 z^{n-1} + ... + a_n} \tag{2.20}$$

De acuerdo con [25], existen varias representaciones y estas son obtenidas a partir de la Eq. (2.20).

- **Forma canónica controlable:** Las ecuaciones de estado y de salida, están dadas por las Eqs. (2.21) y

(2.22), respectivamente.

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1)) \\ \vdots \\ x_{n-1}(k+1) \\ x_n(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & \cdots & -a_1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_{n-1}(k) \\ x_n(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u(k) \tag{2.21}$$

$$y(k) = \begin{bmatrix} b_n - a_n b_0 \vdots b_{n-1} - a_{n-1} b_0 \vdots \cdots \vdots b_1 - a_1 b_0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix} + b_o u(k) \tag{2.22}$$

- **Forma canónica observable:** Las ecuaciones de estados y de salida, están dadas por las Eqs. (2.23) y (2.24), respectivamente.

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ \vdots \\ x_{n-1}(k+1) \\ x_n(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & -a_n \\ 1 & 0 & \cdots & 0 & 0 & -a_{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & -a_2 \\ 0 & 0 & \cdots & 0 & 1 & -a_1 \end{bmatrix} = \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_{n-1}(k) \\ x_n(k) \end{bmatrix} + \begin{bmatrix} b_n - a_n b_0 \\ b_{n-1} - a_{n-1} b_0 \\ \vdots \\ b_2 - a_2 b_0 \\ b_1 - a_1 b_0 \end{bmatrix} u(k) \tag{2.23}$$

$$y(k) = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_{n-1}x(k) \\ x_n(k) \end{bmatrix} + b_0 u(k) \tag{2.24}$$

- **Forma canónica diagonal:** Si los polos de la función de transferencia son todos distintos, su representación está dada por las Eqs. (2.25) y (2.26).

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ \vdots \\ x_n(k+1) \end{bmatrix} = \begin{bmatrix} p_1 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_n \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} u(k) \tag{2.25}$$

$$y(k) = \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix} + b_0 u(k) \tag{2.26}$$

2.2.2.1. Controlabilidad

"El sistema de control en tiempo discreto dado por la Eq. (2.27) se dice es de estado completamente controlable, o simplemente de estado controlable, si existe una señal de control constante por intervalos $u(kT)$

definida a lo largo de un número finito de períodos de muestreo de forma que, al partir de cualquier estado inicial, el estado $x(kT)$ pueda ser transferido al estado deseado x_j en n períodos de muestreo como máximo" [25]. Es decir que el sistema sea posible de transferirse desde un estado inicial arbitrario a cualquier estado deseado en un tiempo finito; o será completamente controlable, si cada variable de estado se puede controlar en un tiempo finito por una señal de control que no esté anidada a ningún tipo de restricción [24].

$$X((k+1)T) = Gx(kT) + Hu(kT) \tag{2.27}$$

Solucionando la Eq. (2.27) se obtiene la Eq. (2.28) y posteriormente la Eq. (2.29).

$$x(nT) = G^n x(0) + \sum_{j=0}^{n-1} G^{n-j-1} Hu(jT) \tag{2.28}$$

$$x(nT) = G^n x(0) + G^{n-1} Hu(0) + G^{n-2} Hu(T) + \cdots + Hu((n-1)T) \tag{2.29}$$

De donde se obtiene la Eq. (2.30):

$$\mathbf{x}(nT) - \mathbf{G}^n \mathbf{x}(0) = \left[\mathbf{H}\vdots\mathbf{GH}\vdots\cdots\vdots\mathbf{G}^{n-1}\mathbf{H}\right] \begin{bmatrix} u((n-1)T) \\ u((n-2)T) \\ \vdots \\ u(0) \end{bmatrix} \tag{2.30}$$

Debido a que $\mathbf{H}$ es un vector columna se puede determinar que el conjunto de matrices $\mathbf{H}\vdots\mathbf{GH}\vdots\cdots\vdots\mathbf{G}^{n-1}\mathbf{H}$ también es un vector columna y su rango es n (Ver Eq. (2.31)) [24]:

$$rango([\mathbf{H}\vdots\mathbf{GH}\vdots\cdots\vdots\mathbf{G}^{n-1}\mathbf{H}]) = n \tag{2.31}$$

Por lo tanto la matriz de controlabilidad se puede ver en Eq. (2.32).

$$M = [\mathbf{H}\vdots\mathbf{GH}\vdots\cdots\vdots\mathbf{G}^{n-1}\mathbf{H}] \tag{2.32}$$

2.2.2.2. Observabilidad

"El sistema se dice ser completamente observable si cualquier estado inicial $\mathbf{x}(0)$ puede determinarse a partir de la observación de $\mathbf{y}(kT)$ sobre un número finito de períodos de muestreo. El sistema, por lo tanto, es completamente observable, si cualquier transición del estado de manera eventual afecta a todos los elementos del vector de salida" [25]. Este aspecto es de suma importancia para poder determinar cuantos estados pueden ser observados o medidos y cuantos no, a su vez y en función a esto se puede determinar la necesidad de usar estimadores u observadores para los estados que no pueden ser medidos [24] [26].

Para la obtención de la condición de observabilidad se parte de las Eqs. (2.33) y (2.34).

$$\mathbf{x}(kT) = \mathbf{G}^k \mathbf{x}(0) \tag{2.33}$$

$$\mathbf{y}(kT) = \mathbf{C}\mathbf{G}^k \mathbf{x}(0) \tag{2.34}$$

Aplicando el concepto de observabilidad se tiene la Eq. (2.35).

$$\begin{aligned} \mathbf{y}(0) &= \mathbf{C}\mathbf{x}(0) \\ \mathbf{y}(T) &= \mathbf{C}\mathbf{G}\mathbf{x}(0) \\ &\vdots \\ \mathbf{y}((n-1)T) &= \mathbf{C}\mathbf{G}^{n-1}\mathbf{x}(0) \end{aligned} \tag{2.35}$$

Para de esta manera obtener un conjunto de n ecuaciones linealmente independientes, siendo este conjunto conocido como la matriz de observabilidad $\mathbf{N}$ y esta definida por la Eq. (2.36) [24].

$$\mathbf{N} = \begin{bmatrix} \mathbf{C} \\ \mathbf{C}\mathbf{G} \\ \vdots \\ \mathbf{C}\mathbf{G}^{n-1} \end{bmatrix} \tag{2.36}$$

2.2.3. Controlador por retroalimentación de estados

Este es un método que se basa en la técnica de ubicación de polos. Los polos se pueden ubicar en cualquier localización con la única condición de que el sistema sea completamente observable, y a partir de eso se puede hacer la localización de los polos con una retroalimentación de los estados a través de una matriz de ganancias $\mathbf{K}$ [24, 26].

Se considera el sistema de la Eq. (2.37) con representación a lazo abierto que se puede observar en la Fig. 2.7.

$$\mathbf{x}(k+1) = \mathbf{G}\mathbf{x}(kT) + \mathbf{H}u(kT) \tag{2.37}$$

Donde:

$\mathbf{x}(k)$	vector de estado
$u(k)$	señal de control
$\mathbf{G}$	matriz de estado
$\mathbf{H}$	matriz de salida

En este caso al tratarse de un sistema LTI la matriz D tiene un valor de 0 y por lo tanto la representación del sistema se puede observar en la Fig. 2.8.

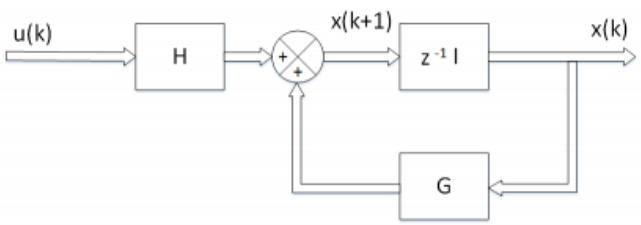


Figura 2.8: Representación en espacios de estados de un sistema LTI en lazo abierto. [24]

La ley de control está dada por la Eq. (2.38):

$$u(k) = -\mathbf{K}\mathbf{x}(k) \tag{2.38}$$

Donde **K** es la matriz de retroalimentación con una dimensión de $1xn$. Esta ley de control permite que el sistema sea convertido a un sistema de control a lazo cerrado. Su representación en bloques se puede observar en la Fig. 2.9.

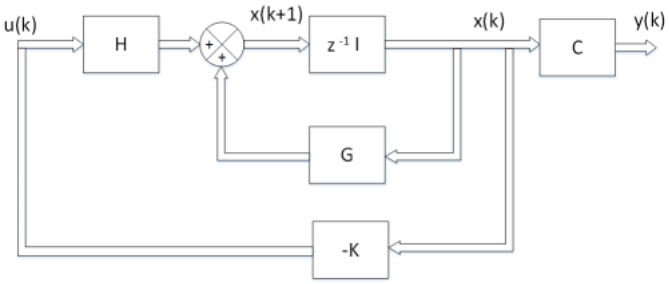


Figura 2.9: Representación en espacios de estados de un sistema LTI en lazo cerrado. [24]

Para la obtención de su ecuación de estado se remplaza la Eq. (2.38) en la Eq. (2.37) y se obtiene la Eq. (2.39).

$$\mathbf{x}(k+1) = \mathbf{(G\text{-}HK)x}(k) \tag{2.39}$$

La matriz de ganancia **K** se elige de tal manera que los valores característicos de $\mathbf{G} - \mathbf{HK}$ sean los polos en lazo cerrado escogidos $\mu_1, \mu_2, \mu_3, \cdots, \mu_n$. Un método para encontrar la matriz de ganancia es la aplicación de la Fórmula de Ackerman [24] [25].

2.2.4. Fórmula de Ackerman

Esta fórmula permite realizar el cálculo de la matriz de ganancia **K** y para el cálculo se parte de la consideración de un sistema completamente controlable el cual está definido por la Eq. (2.17), su realimentación está dada por la Eq. (2.38), y sus polos deseados en lazo cerrado están ubicados en [24]:

$$\begin{aligned} z_1 &= \mu_1 \\ z_2 &= \mu_2 \\ \vdots &\ \vdots \ \ \vdots \\ z_n &= \mu_n \end{aligned}$$

Una recomendación para la ubicación de los polos es hacer uso de los polinomios normalizados de Bessel. La ecuación característica está dada por la Eq. (2.40) [24].

$$\begin{aligned} |z\mathbf{I} - \mathbf{G} + \mathbf{HK}| &= (z-\mu_1)(z-\mu_2)\cdots(z-\mu_n) \\ |z\mathbf{I} - \mathbf{G} + \mathbf{HK}| &= z^n + \alpha_1 z^{n-1} + \alpha_2 z^{n-2} + \cdots + \alpha_{n-1} z + \alpha_n = 0 \end{aligned} \tag{2.40}$$

Se determina lo siguiente (ver Eq. (2.41)):

$$\hat{\mathbf{G}} = \mathbf{G} - \mathbf{HK} \tag{2.41}$$

Donde $\hat{\mathbf{G}}$ satisface su propia ecuación característica de acuerdo al teorema de Cayley-Hamilton [24], por lo tanto se obtiene la Eq. (2.42).

$$\hat{\mathbf{G}}^n + \alpha_1\hat{\mathbf{G}}^{n-1} + \alpha_2\hat{\mathbf{G}}^{n-2} + \cdots + \alpha_{n-1}\hat{\mathbf{G}} + \alpha_n\mathbf{I} = \phi(\hat{\mathbf{G}}) = 0 \tag{2.42}$$

Siendo este el criterio para la deducción de la fórmula de Ackerman. Se tiene la Eq. (2.43) [24].

$$\begin{array}{ccc} \mathbf{I} & = & \mathbf{I} \\ \hat{\mathbf{G}} & = & \mathbf{G} - \mathbf{HK} \\ \hat{\mathbf{G}}^2 & = & (\mathbf{G} - \mathbf{HK})^2 = \mathbf{G}^2 - \mathbf{GHK} - \mathbf{HK}\hat{\mathbf{G}} \\ \hat{\mathbf{G}}^3 & = & (\mathbf{G} - \mathbf{HK})^3 = \mathbf{G}^3 - \mathbf{G}^2\mathbf{HK} - \mathbf{GHK}\hat{\mathbf{G}} - \mathbf{HK}\hat{\mathbf{G}}^2 \\ \vdots & \vdots & \vdots \\ \hat{\mathbf{G}}^n & = & (\mathbf{G} - \mathbf{HK})^n = \mathbf{G}^n - \mathbf{G}^{n-1}\mathbf{HK} - \cdots - \mathbf{HK}\hat{\mathbf{G}}^{n-1} \end{array} \tag{2.43}$$

Se multiplica la Eq. (2.43) por $\alpha_n, \alpha_{n-1}, \cdots, \alpha_0$ donde $\alpha_0 = 1$ y se obtiene la Eq. (2.44) [24].

$$\begin{aligned} \alpha_n\mathbf{I} + \alpha_{n-1}\hat{\mathbf{G}} + \alpha_{n-2}\hat{\mathbf{G}}^2 + \cdots + \hat{\mathbf{G}}^n = \alpha_n\mathbf{I} + \alpha_{n-1}\mathbf{G} \\ + \alpha_{n-2}\mathbf{G}^2 + \cdots + \mathbf{G}^n - \alpha_{n-1}\mathbf{HK} - \alpha_{n-2}\mathbf{GHK} \\ - \alpha_{n-2}\mathbf{HK}\hat{\mathbf{G}} - \cdots - \mathbf{G}^{n-1}\mathbf{HK} - \cdots - \mathbf{HK}\hat{\mathbf{G}}^{n-1} \end{aligned} \tag{2.44}$$

Reescribiendo la Eq. (2.44), se obtiene la Eq. (2.45).

$$\begin{aligned} (\hat{\mathbf{G}}) &= \phi(\mathbf{G}) - \alpha_{n-1}\mathbf{HK} - \alpha_{n-2}\mathbf{GHK} - \alpha_{n-2}\mathbf{HK}\hat{\mathbf{G}} - \cdots - \mathbf{HK}\hat{\mathbf{G}}^{n-1} - \mathbf{G}^{n-1}\mathbf{HK} \\ &= \phi(\mathbf{G}) - \left[\mathbf{H}\vdots\mathbf{GH}\vdots\cdots\vdots\mathbf{G}^{n-1}\mathbf{H}\right]\begin{bmatrix} \alpha_{n-1}\mathbf{K} + \alpha_{n-2}\mathbf{K}\hat{\mathbf{G}} + \cdots + \mathbf{K}\hat{\mathbf{G}}^{n-1} \\ \alpha_{n-2}\mathbf{K} + \alpha_{n-3}\mathbf{K}\hat{\mathbf{G}} + \cdots + \mathbf{K}\hat{\mathbf{G}}^{n-2} \\ \vdots \\ \mathbf{K} \end{bmatrix} \end{aligned} \tag{2.45}$$

Donde se considera la Eq. (2.46).

$$\phi(\hat{\mathbf{G}}) = 0 \tag{2.46}$$

Por lo tanto, se obtiene la Eq. (2.47):

$$\phi(\mathbf{G}) = \left[\mathbf{H}\vdots\mathbf{GH}\vdots\cdots\vdots\mathbf{G}^{n-1}\mathbf{H}\right]\begin{bmatrix} \alpha_{n-1}\mathbf{K} + \alpha_{n-2}\mathbf{K}\hat{\mathbf{G}} + \cdots + \mathbf{K}\hat{\mathbf{G}}^{n-1} \\ \alpha_{n-2}\mathbf{K} + \alpha_{n-3}\mathbf{K}\hat{\mathbf{G}} + \cdots + \mathbf{K}\hat{\mathbf{G}}^{n-2} \\ \vdots \\ \mathbf{K} \end{bmatrix} \tag{2.47}$$

Al ser un sistema completamente controlable, la Eq. (2.47) se puede escribir mediante la Eq. (2.48) [24]:

$$\begin{bmatrix} \alpha_{n-1}\mathbf{K} + \alpha_{n-2}\mathbf{K}\hat{\mathbf{G}} + \cdots + \mathbf{K}\hat{\mathbf{G}}^{n-1} \\ \alpha_{n-2}\mathbf{K} + \alpha_{n-3}\mathbf{K}\hat{\mathbf{G}} + \cdots + \mathbf{K}\hat{\mathbf{G}}^{n-2} \\ \vdots \\ \mathbf{K} \end{bmatrix} = \left[\mathbf{H}\vdots\mathbf{GH}\vdots\cdots\vdots\mathbf{G}^{n-1}\mathbf{H}\right]^{-1}\phi(\mathbf{G}) \tag{2.48}$$

Se multiplica ambos miembros de la Eq. (2.48) por $\begin{bmatrix}0 & 0 & \cdots & 0 & 1\end{bmatrix}$ y se tiene la Eq. (2.49), siendo esta la fórmula de Ackerman [24].

$$\mathbf{K} = \begin{bmatrix}0 & 0 & \cdots & 0 & 1\end{bmatrix}\begin{bmatrix}\mathbf{H} \vdots \mathbf{GH} \vdots \cdots \vdots \mathbf{G}^{n-1}\mathbf{H}\end{bmatrix}^{-1} \phi(\mathbf{G}) \tag{2.49}$$

2.3. Trabajos Relacionados

Gafvert [27] en su investigación demuestra las ecuaciones de movimiento de un péndulo de Furuta partiendo de las ecuaciones de Euler-Lagrange. Al demostrar estas ecuaciones nota la no linealidad del sistema y detalla los puntos de equilibrio con los que se linealiza y encuentra un control de retroalimentación de estado lineal. Esta investigación es la base para el desarrollo de diferentes investigaciones o implementaciones con péndulos de Furuta.

La investigación de Cervantes et al. [9] presenta una metodología para implementar un péndulo de Furuta en físico. Esta metodología comienza por el diseño del mismo con la ayuda del software SolidWorks. Como segundo punto obtienen el modelo matemático con la finalidad de obtener las ecuaciones de movimiento y simularlas en MATLAB, una vez que tienen simuladas estas ecuaciones realizan el desarrollo de un control PID y la calibración de las ganancias K_p, K_i y K_d. Una vez que tienen las constantes calibradas pasan a la implementación del controlador en una placa física y para lo cual usan una tarjeta de control basada en el PIC18F4550. Por último, ejecutan el sistema y como respuesta tienen que el péndulo se estabiliza, pero solo para pequeñas perturbaciones externas.

Además, existe el estudio comparativo del desempeño de tres tipos de controladores para el péndulo invertido de Furuta [28], en donde el modelo matemático del péndulo es tomado de [27] y es implementado en la herramienta de programación SIMULINK. El autor desarrolla tres tipos de controladores: PID, Regulador Cuadrático Lineal (LQR), y Fuzzy con la finalidad de adquirir los datos de respuesta de cada controlador y hacer un comparativo a través del Integral del Error Absoluto (IAE). Los resultados de la implementación y comparación de los controladores determinan que el controlador LQR presenta mejor respuesta en el péndulo, seguido por el controlador Fuzzy y por último el controlador PID.

La investigación de Regalo [29] basa las ecuaciones de movimiento de su péndulo en las ecuaciones dadas en [27]. A partir de esto, utiliza SIMULINK para simular las ecuaciones y modelar el péndulo usando Vrealm. Implementa dos técnicas de control: controlador PID y controlador LQR. En el trabajo se implementa la suma de dos controladores PID, el uno en función al ángulo del brazo horizontal (ϕ) y el otro en función al ángulo del péndulo (θ). Los resultados muestran que los dos controladores responden de una buena forma en la simulación y casi no existe diferencia entre ellos; mientras que, al momento de implementar estos controladores sobre el péndulo físico el que mejor rendimiento presenta es el controlador LQR.

El trabajo realizado por Ismael Minchala y Rubén Marbán en [30], parte del modelo matemático propuesto en [27] y aplican tres técnicas de control diferentes, siendo estas: controlador de retroalimentación de estados con LQR, controlador de modos deslizantes y controlador de modos deslizantes adaptativos. El controlador de retroalimentación de estados con LQR estabiliza el sistema pero tiene un estado inestable (ϕ). Por otra parte, el controlador de modos deslizantes estabiliza el sistema con todos sus estados estables. Y por último, el controlador de modos deslizantes adaptativos estabiliza el sistema y mejora la estabilización de cada estado.

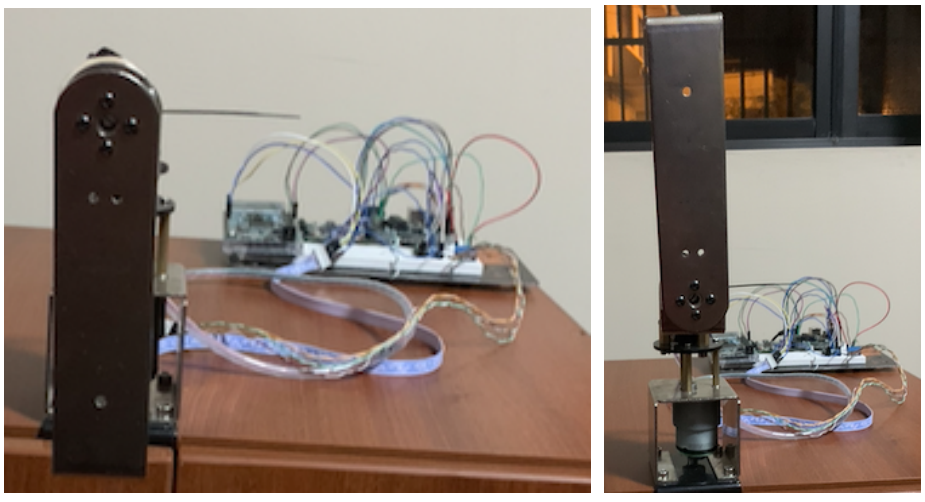
CAPÍTULO 3

Diseño e implementación de controladores

En esta sección se presenta una descripción detallada sobre la identificación de la planta (Sección 3.1), y sobre el diseño e implementación del controlador PID (Sección 3.2) y del controlador de retroalimentación de estados (Sección 3.3) utilizados.

3.1. Identificación de la planta

En la Sección 2.1 se describen los componentes de un péndulo de Furuta de manera general. En la Fig. 3.1 se presenta el sistema real del péndulo de Furuta a usarse para el desarrollo de la tesis.



(a) Estado natural (b) Estado Erguido

Figura 3.1: Péndulo de Furuta.

Sus componentes son:

Motor EMG30: Es un motor de 12 Vdc con relación de reducción 20:1 y equipado con un encoder de cuadratura. (Ver Fig. 3.2). Especificaciones:

- Corriente nominal: 530mA.
- Velocidad sin carga: 216.
- Corriente sin carga: 150mA.
- Corriente de parada: 2.5A.
- Rendimiento nominal: 4.22W.
- Codificadores por vuelta del eje de salida: 260.

Figura 3.2: Motor EMG30

Encoder WDD35D4-5K: Ver Fig. 3.3

- Diámetro: 36.5 mm.
- Diámetro eje: 6 mm.
- Resistencia: 5K.
- Tolerancia de resistencia ±15 %
- Linealidad independiente ±0, 5 %, ±0, 1 %
- Ángulo eléctrico teórico: 345º ±2º
- Resolución: ilimitada
- Potencia 2W.
- Coeficiente de temperatura de resistencia (ppm/grado): < ±400
- Ángulo Mecánico: 360º

Figura 3.3: Encoder WDD35D4-5K

Brazo horizontal de aluminio: Ver Fig. 3.4.

- Medidas: 150x35x2 mm.
- Masa: 227 gr.

Figura 3.4: Brazo Horizontal

Péndulo de aluminio: Ver Fig. 3.5.

- Medidas: 150x35x2 mm.
- Masa: 152 gr.

Figura 3.5: Péndulo

El momento de inercia J del motor se calcula con la Eq. (3.1) presentado en [31]; mientras que el momento de inercia J_p del péndulo se calcula con la Eq. (3.2) presentado en [32].

$$J = \frac{Mr^2}{2} \tag{3.1}$$

$$J_p = \frac{Ma^2}{3} \tag{3.2}$$

Los valores de las constantes del péndulo se presentan en la Tabla 3.1.

Tabla 3.1: Constantes péndulo.

J	$1{,}96 * 10^{-5} Kgm^2$
l_a	$0{,}15m$
m_a	$0{,}163Kg$
J_p	$6{,}24 * 10^{-5} Kgm^2$
l_p	$0{,}15m$
m_p	$0{,}152Kg$
M	$0{,}023Kg$

3.2. Controlador PID

Como primera etapa se implementa el controlador PID con el péndulo en su estado natural. El objetivo es familiarizarse con la planta. El desarrollo de este algoritmo está basado en las ecuaciones presentadas en la Sección 2.2.1, la posición final que se busca está mostrada en la Fig. 3.1a y su diagrama de flujo se observa en la Fig. 3.6.

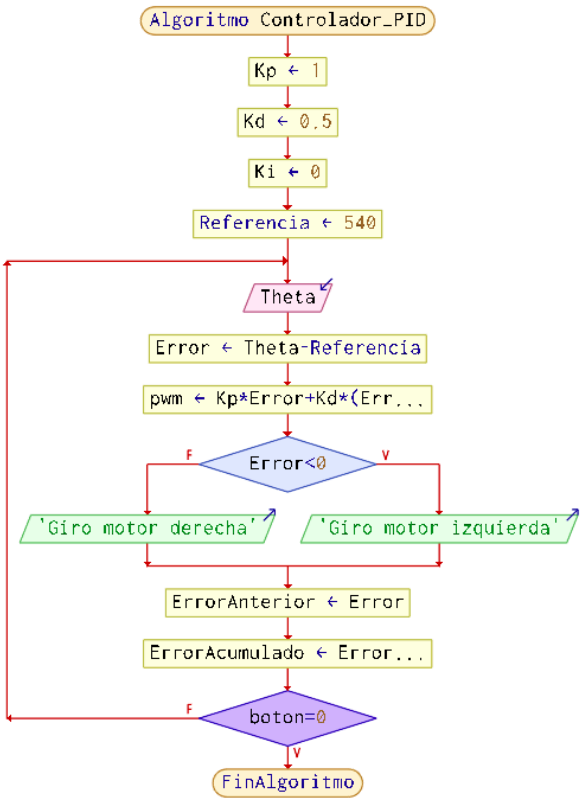


Figura 3.6: Diagrama de flujo del controlador PID para el péndulo en estado natural.

Una vez que el controlador empieza a funcionar, se aplica una pequeña perturbación al péndulo y el controlador debe llevar al péndulo a la posición inicial lo más rápido posible. Esto se puede conseguir haciendo que el brazo del péndulo gire en posición opuesta a la del péndulo, es decir, cuando el péndulo se encuentra inclinado hacia la izquierda de su referencia, el brazo del péndulo gira hacia la derecha, y cuando el péndulo se encuentra a la derecha de su referencia, el brazo del péndulo gira hacia la izquierda. De esta manera se logra que el péndulo regrese nuevamente a su referencia o estado inicial.

En el diagrama de flujo, lo primero que se realiza es la calibración de los parámetros del controlador (K_p, K_i, K_d), mismos que son calibrados de manera heurística como se explica en la Sección 2.2.1. Estos parámetros se presentan en la Tabla 3.2. Después de hacer la calibración de los parámetros K_p, K_i, K_d se configura la referencia o cero del sistema que en esta ocasión es 540. Una vez definidas las constantes anteriores, el algoritmo lee la posición del péndulo (θ), calcula el error respecto a la referencia, y por último, envía la salida PWM del controlador, que es nombrada como "pwm", a los motores.

El signo del error determina el sentido de giro del motor. Si el error es negativo el motor gira a la izquierda y si es positivo gira a la derecha. Además, se guardan los valores de la posición del péndulo (θ) y del error, en

variables auxiliares y se repite el proceso a partir de la lectura de la posición del péndulo (θ).

Tabla 3.2: Parámetros del controlador PID para péndulo en estado natural.

	K_p	K_i	K_d
1	1	0	0
2	10	0	0
3	0.1	0	0
4	0.8	0	0
5	0.1	0	0.8
6	0.3	0	0.8

En una segunda etapa, se implementó el controlador PID para el péndulo en estado erguido y su posición final se muestra en la Fig. 3.1b. La diferencia de control del péndulo en estado erguido respecto al péndulo en estado natural se encuentra en el movimiento del motor de acuerdo al error. En este caso el brazo del péndulo se mueve al mismo lado de la inclinación del péndulo. Es decir, si el péndulo está inclinado a la derecha de la referencia, el brazo del péndulo gira hacia la derecha, y si el péndulo está inclinado hacia la izquierda de la referencia, el brazo del péndulo gira a la izquierda. De esta manera, el péndulo busca regresar a la referencia o estado inicial. Su diagrama de flujo se presenta en la Fig. 3.7.

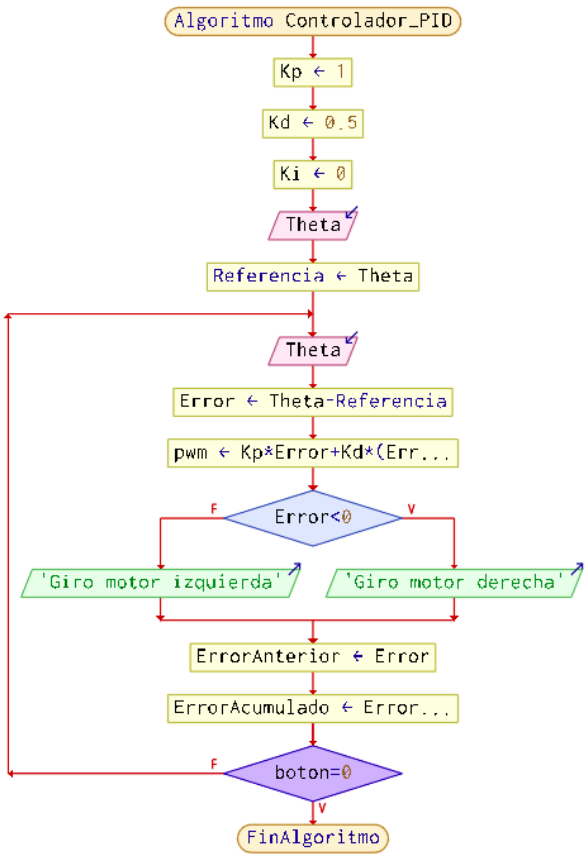


Figura 3.7: Diagrama de flujo del controlador PID para el péndulo en estado erguido.

En el diagrama de flujo de la Fig. 3.7, el primer proceso es la calibración de los parámetros del controlador (K_p, K_i, K_d). Después de hacer la calibración de los parámetros, se configura la referencia o cero del sistema, que

en esta ocasión es el valor de la primera lectura de la posición del péndulo (θ). Una vez definidas las constantes anteriores, el algoritmo realiza la lectura de la posición del péndulo (θ), calcula el error respecto a la referencia; y por último, envía la salida PWM del controlador, que es nombrada como "pwm", a los motores.

El sentido de giro lo determina al signo del error. Si el error es negativo el motor gira a la derecha y si es positivo gira a la izquierda. Por último, se guardan los valores de la posición del péndulo (θ) y el error en variables auxiliares. El proceso se repite a partir de la lectura de la posición del péndulo (θ).

El algoritmo de control para el péndulo en estado erguido, se implementa en las tarjetas Arduino y FPGA. Los valores de las diferentes calibraciones se presentan en las Tablas 3.3 y 3.4, respectivamente. Para el caso de FPGA, la adquisición de datos del péndulo se realiza con un Arduino y por medio del puerto serial se envían los datos al FPGA. En este se realiza el procesamiento del control y envía las acciones al motor.

Tabla 3.3: Parámetros del controlador PID en Arduino para el péndulo en estado erguido.

	K_p	K_i	K_d
1	4,9	0	10
2	4,7	0	10
3	4,7	0	10.9
4	4,7	0	10.7
5	4,7	0	10.5
6	4,7	0	10.3

Tabla 3.4: Parámetros del controlador PID en el FPGA para el péndulo en estado erguido.

	K_p	K_i	K_d
1	4,7	0	10,7
2	4,7	0	10,9
3	4,6	0	10,9
4	4,6	0	11
5	4,6	0	10
6	4,5	0	9

Como se observa en las Tablas 3.2, 3.3, y 3.4; en ninguno de los controladores PID se utiliza el componente integral (K_i). Esta decisión se debe a que al ingresar este componente al controlador se introduce un polo en el origen a la función de transferencia del sistema a lazo abierto, empeorando de esta manera la respuesta transitoria del sistema (Sección 2.2.1). De esta manera, los controladores para el péndulo en estado natural y en estado erguido, cuentan sólo con una componente derivativa (K_d) y una componente proporcional (K_p).

3.3. Controlador de retroalimentación de estados

Este controlador usa la técnica de ubicación de polos (Sección 2.2.3). El controlador requiere obtener las ecuaciones dinámicas que caracterizan al sistema, linealizar las ecuaciones, representarlas en su espacio de estados, y obtener el vector de retroalimentación.

3.3.1. Modelo matemático

Este modelo se basa en la teoría de Euler-Lagrange [27], en el cual un punto P sobre el péndulo está dado por su vector de posición mostrado en la Eq. (3.3).

$$r(r_a, r_p) = (r_x(r_a, r_p), r_y(r_a, r_p), r_z(r_a, r_p))^T; \tag{3.3}$$

La equivalencia de cada coordenada se muestra en la Eq. (3.4).

$$\begin{aligned} r_x(r_a, r_p) &= r_a \cos\phi - r_p \sin\phi \sin\theta \\ r_y(r_a, r_p) &= r_a \sin\phi + r_p \cos\phi \sin\theta \\ r_z(r_a, r_p) &= r_p \cos\theta \end{aligned} \tag{3.4}$$

donde, el subíndice a se refiere al brazo horizontal y el subíndice p se refiere al brazo del péndulo, como se observa en la Fig. 3.8. Las distancias radiales son medidas desde el centro de rotación de cada cuerpo, por lo tanto r_a y r_p son las distancias radiales del brazo horizontal y del brazo del péndulo, respectivamente.

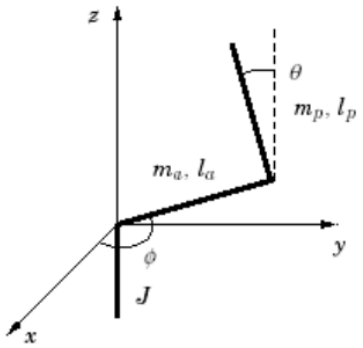


Figura 3.8: Parámetros del péndulo de Furuta.

Con el vector de posición definido, se obtiene el vector de velocidad derivando la Eq. (3.3) respecto al tiempo. El resultado se muestra en la Eq. (3.5).

$$v(r_a, r_p) = (v_x(r_a, r_p), v_y(r_a, r_p), v_z(r_a, r_p))^T; \tag{3.5}$$

La equivalencia de la velocidad de cada coordenada está mostrada en la Eq. (3.6).

$$\begin{aligned} v_x(r_a, r_p) &= -r_a \sin\phi\dot{\phi} - r_p \cos\theta \sin\phi\dot{\theta} - r_p \sin\theta \cos\phi\dot{\phi} \\ v_y(r_a, r_p) &= -r_a \cos\phi\dot{\phi} + r_p \cos\theta \cos\phi\dot{\theta} - r_p \sin\theta \sin\phi\dot{\phi} \\ v_z(r_a, r_p) &= -r_p \sin\theta\dot{\theta} \end{aligned} \tag{3.6}$$

Para obtener las ecuaciones de movimiento se utiliza el Lagrangiano de la Eq. (3.7), el mismo que define sus ecuaciones de movimiento con la Eq. (3.8).

$$L = T - V \tag{3.7}$$

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\phi}}\right) - \frac{\partial L}{\partial \phi} &= \tau_\phi \\ \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} &= 0 \end{aligned} \tag{3.8}$$

Por lo tanto, para obtener las ecuaciones de movimiento, se calculan las energías cinéticas (T) y potenciales

(V) del sistema, mismas que son obtenidas con las Eq. (3.9) y (3.10), respectivamente.

$$T = \frac{1}{2}\int v^2 dm \tag{3.9}$$

$$V = g\int r_z \tag{3.10}$$

Para facilitar el cálculo, el sistema fue dividido en cuatro cuerpos. A continuación, se indica cada cuerpo con sus respectivas fórmulas para calcular la energía cinética y la energía potencial.

- Pilar central.
  - $2T_c = J\dot{\phi}^2$
  - $V_c = 0$
- Brazo horizontal.
  - $2T_a = \int_0^{l_a} v^2(s,0)m_a/l_a ds = \frac{1}{3}m_a l_a^2\dot{\phi}^2$
  - $V_a = 0$
- Brazo del péndulo.
  - $2T_p = \int_0^{l_p} v^2(r_a,s)m_p/l_p ds = m_p(l_a^2 + \frac{1}{3}l_p^2\sin^2\theta)\dot{\phi}^2 + m_p l_a l_p\cos\theta\dot{\phi}\dot{\theta} + \frac{1}{3}m_p l_p^2\dot{\theta}^2$
  - $V_p = g\int_0^{l_p} r_z(l_a,s)m_p/l_p ds = \frac{1}{2}m_p g l_p\cos\theta$
- Masa balanceada.
  - $2T_m = M(l_a^2 + l_p^2\sin^2\theta)\dot{\phi}^2 + 2Ml_a l_p\cos\theta\dot{\phi}\dot{\theta} + Ml_p^2\dot{\theta}^2$
  - $V_m = Mgl_p\cos\theta$

Por lo tanto, se tienen las ecuaciones mostradas en la Eq. (3.11).

$$\begin{aligned} T &= T_c + T_a + T_p + T_m \\ V &= V_c + V_a + V_p + V_m \end{aligned} \tag{3.11}$$

Aplicando las derivadas parciales de la Eq. (3.8), se obtiene el desarrollo mostrado en la Eq. (3.12).

$$\frac{\partial L}{\partial\phi} = 0$$

$$\begin{aligned} \frac{\partial L}{\partial\dot{\phi}} &= (J + (M + \tfrac{1}{3}m_a + m_p)l_a^2 + (M + \tfrac{1}{3}m_p)l_p^2\sin^2\theta)\dot{\phi} + (M + \tfrac{1}{2}m_p)l_a l_p\cos\theta\dot{\theta} \\ \frac{\partial L}{\partial\theta} &= (M + \tfrac{1}{3}m_p)l_p^2\cos\theta\sin\theta\dot{\phi}^2 - (M + \tfrac{1}{2}m_p)l_a l_p\sin\theta\dot{\phi}\dot{\theta} + (M + \tfrac{1}{2}m_p)gl_p\sin\theta \end{aligned} \tag{3.12}$$

$$\frac{\partial L}{\partial\dot{\theta}} = (M + \tfrac{1}{2}m_p)l_a l_p\cos\theta\dot{\phi} + (M + \tfrac{1}{3}m_p)l_p^2\dot{\theta}$$

Para facilidad de cálculo se concatena en las siguientes variables:

$$\alpha = J + (M + \tfrac{1}{3}m_a + m_p)l_a^2 \qquad \beta = (M + \tfrac{1}{3}m_p)l_p^2$$

$$\gamma = (M + \tfrac{1}{2}m_p)l_a l_p \qquad \delta = (M + \tfrac{1}{2}m_p)gl_p$$

Remplazando la Eq. (3.12) en la Eq. (3.8), se obtiene la Eq. (3.13).

$$(\alpha + \beta \sin^2\theta)\ddot{\phi} + \gamma \cos\theta\ddot{\theta} + 2\beta \cos\theta \sin\theta\dot{\phi}\dot{\theta} - \gamma \sin\theta\dot{\theta}^2 = \tau_\phi$$
$$\gamma \cos\theta\ddot{\phi} + \beta\ddot{\theta} - \beta \cos\theta \sin\theta\dot{\phi}^2 - \delta \sin\theta = 0 \tag{3.13}$$

Las ecuaciones de movimiento (Eq. (3.13)) pueden ser representadas en forma matricial. Compuestas por la matriz de inercias, la matriz de fuerzas centrípetas y Coriolis, y la matriz de fuerzas gravitacionales, como se observa en la Eq. (3.14).

$$\begin{bmatrix} \alpha + \beta \sin^2\theta & \gamma \cos\theta \\ \gamma \cos\theta & \beta \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} 2\beta \cos\theta \sin\theta\dot{\phi}\dot{\theta} - \gamma \sin\theta\dot{\theta}^2 \\ -\beta \cos\theta \sin\theta\dot{\phi}^2 \end{bmatrix} + \begin{bmatrix} 0 \\ -\delta \sin\theta \end{bmatrix} = \begin{bmatrix} \tau_\phi \\ 0 \end{bmatrix} \tag{3.14}$$

3.3.2. Validación y ajuste del modelo matemático

Para validar el modelo se utiliza la herramienta de simulación Simulink. Las Eqs. (3.13) se implementan en lazo abierto para validar el modelo matemático; remplazando $\tau_\phi = 0$ en la Eq. (3.13) se tiene la Eq. (3.15). Su implementación se observa en la Fig. 3.9. Los valores que se utilizan para la simulación se muestran en la Tabla 3.1.

$$(\alpha + \beta \sin^2\theta)\ddot{\phi} + \gamma \cos\theta\ddot{\theta} + 2\beta \cos\theta \sin\theta\dot{\phi}\dot{\theta} - \gamma \sin\theta\dot{\theta}^2 = 0$$
$$\gamma \cos\theta\ddot{\phi} + \beta\ddot{\theta} - \beta \cos\theta \sin\theta\dot{\phi}^2 - \delta \sin\theta = 0 \tag{3.15}$$

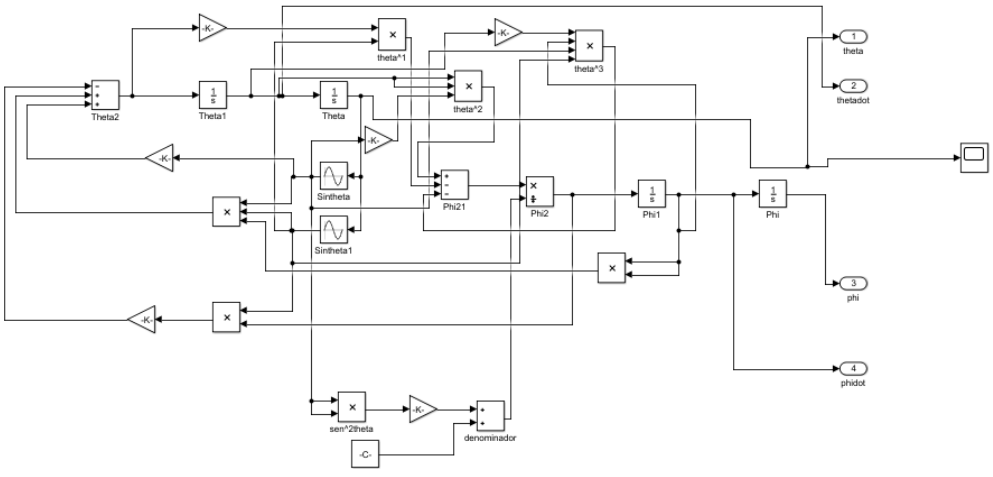


Figura 3.9: Modelo matemático del péndulo de Furuta en Simulink

Este modelo considera el cero del sistema cuando el péndulo se encuentra en su posición erguida. En este caso, para la toma de datos de la simulación, se indica que el péndulo no comienza en la posición de 0°, sino a 45°. De esta manera, el péndulo tendría que caer y estabilizarse alrededor de los 180°. Esto se observa en la Fig. 3.10.

Una vez obtenidos los resultados de la simulación del modelo matemático. El modelo se valida realizando una adquisición de datos de la planta física bajo los mismos parámetros de simulación. la Fig. 3.11 muestra los

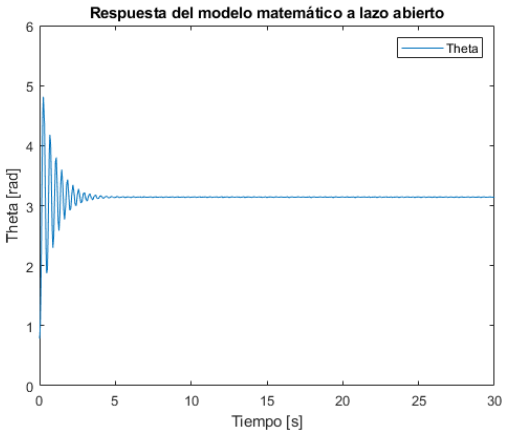


Figura 3.10: Simulación del modelo matemático del péndulo de Furuta en Simulink.

datos adquiridos.

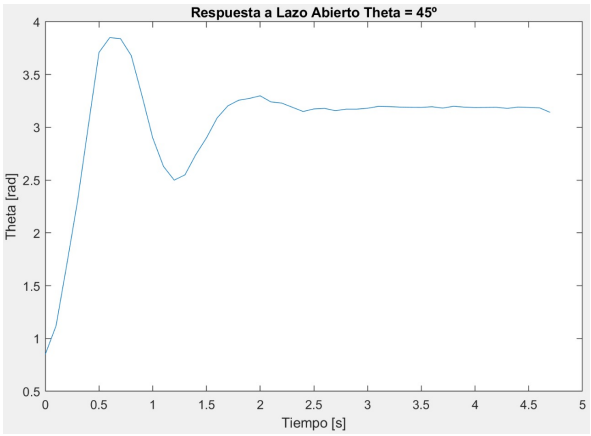
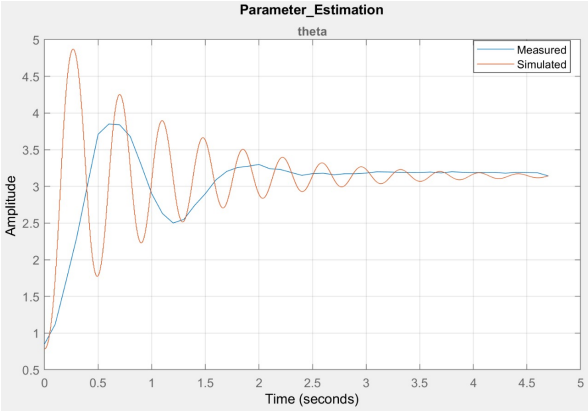


Figura 3.11: Respuesta del péndulo en lazo abierto a 45°

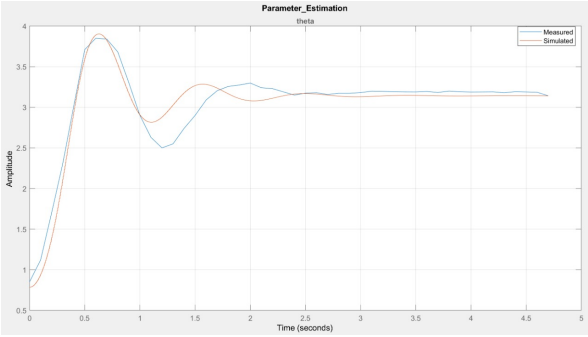
Con los datos simulados y los datos reales, se utiliza la herramienta *Parameter Estimation* de Matlab. Esta herramienta ayuda a realizar un ajuste más fino de los parámetros del péndulo automáticamente, haciendo que los datos simulados se acerquen mucho más a los datos reales. Esto se realiza debido a que los momentos de inercia de la planta no están dados en su detalle técnico y se obtuvieron mediante fórmulas matemáticas, las mismas que desprecian ciertos aspectos físicos del sistema. De esta manera, este estimador de parámetros permite mejorar estas aproximaciones y acercar más el modelo matemático al modelo real.

En la Fig. 3.12a se observan las respuestas del sistema real y del sistema simulado sin usar la herramienta *Parameter Estimation* y en la Fig. 3.12b se observan las respuestas de los sistemas una vez estimados los parámetros con la herramienta *Parameter Estimation*.

Al usar el estimador de parámetros, el sistema simulado responde de manera similar al sistema real (ver Fig. 3.12); esto lo realiza variando el valor de las inercias del sistema (J, J_p). Los nuevos parámetros de sistema se presentan en la Tabla 3.5.



(a) Sin estimación de parámetros



(b) Con estimación de parámetros

Figura 3.12: Validación del modelo matemático

3.3.3. Linealización del modelo

Debido a que se trabaja con un sistema que tiene dinámica Lagrangiana, su modelo es descrito bajo la representación de la Eq. (3.16).

$$M(\mathbf{q})\ddot{\mathbf{q}} + V(\mathbf{q}, \dot{\mathbf{q}}) + D\dot{\mathbf{q}} + G(\mathbf{q}) = \mathbf{Q} \tag{3.16}$$

Tabla 3.5: Constantes del péndulo con estimación de parámetros.

J	$1{,}96 * 10^{-5} Kgm^2$
l_a	$0{,}15m$
m_a	$0{,}163Kg$
J_p	$6{,}24 * 10^{-5} Kgm^2$
l_p	$0{,}15m$
m_p	$0{,}152Kg$
M	$0{,}023Kg$

Donde:

$\ddot{\mathbf{q}}$	=	Vector de aceleración
$\dot{\mathbf{q}}$	=	Vector de velocidad
$\mathbf{q}$	=	Vector de posición
$M(\mathbf{q})$	=	Matriz de inercias
$V(\mathbf{q}, \dot{\mathbf{q}})$	=	Matriz de fuerzas centrifugas y de Coriolis
D	=	Matriz de amortiguamiento
$G(\mathbf{q})$	=	Matriz de fuerzas gravitacionales y potenciales
Q	=	Matriz de fuerzas generalizadas

Para la aplicación del péndulo de Furuta, la matriz de amortiguamiento es cero. El modelo matemático representado por la Eq. (3.16) se muestra en la Eq. (3.14). Para la linealización del modelo se deben encontrar los puntos de equilibrio del sistema. Estos se obtienen mediante la Eq. (3.17).

$$G(\mathbf{q}_{eq}) = 0 \tag{3.17}$$

Aplicando la Eq. (3.17) se obtiene la Eq. (3.18), y resolviendo la Eq. (3.18) se obtiene la Eq. (3.19), siendo estos los puntos de equilibrio del sistema.

$$G(\mathbf{q}_{eq}) = \begin{bmatrix} 0 \\ -\delta \sin\theta \end{bmatrix} = 0 \tag{3.18}$$

$$\mathbf{q}_{eq} = \begin{bmatrix} \phi_{eq} \\ \theta_{eq} \end{bmatrix} = \begin{bmatrix} \phi \\ n\pi \end{bmatrix}; \quad \begin{matrix} \phi \in \mathbb{R} \\ n \in \mathbb{Z} \end{matrix} \tag{3.19}$$

Donde en este caso se utiliza $n = 0$ para linealizar, esto indica que se lo hace alrededor del cero, siendo que este punto presenta un equilibrio inestable del sistema. Para llegar a la matriz linealizada se toma en consideración las siguientes condiciones:

$$|\dot{\mathbf{q}}| < \epsilon_1 \to 0 \qquad |\mathbf{q}| < \epsilon_2 \to 0$$

$$\therefore$$

$$\theta \text{ es pequeña}$$

$$\sin\theta = \theta \qquad \cos\theta = 1 \qquad \dot{\theta}^2 \to 0$$

De esta manera aplicando las condiciones anteriores, se llega a la matriz linealizada del modelo que se presenta en la Eq. (3.20) y a las ecuaciones linealizadas del modelo en la Eq. (3.21).

$$\begin{bmatrix} \alpha & \gamma \\ \gamma & \beta \end{bmatrix} \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ -\delta\theta \end{bmatrix} = \begin{bmatrix} \tau_\phi \\ 0 \end{bmatrix} \tag{3.20}$$

$$\begin{aligned} \alpha\ddot{\phi} + \gamma\ddot{\theta} &= \tau_\phi \\ \gamma\ddot{\phi} + \beta\ddot{\theta} - \delta\theta &= 0 \end{aligned} \tag{3.21}$$

3.3.4. Espacio de estados y Ackerman

Una forma más sencilla de representar el modelo matemático del sistema es hacerlo a través de su espacio de estados. Para lo cual se declaran variables auxiliares en la Eq. (3.22).

$$\begin{aligned} x_1 &= \theta \\ x_2 &= \dot{\theta} \\ x_3 &= \phi \\ x_4 &= \dot{\phi} \end{aligned} \tag{3.22}$$

La Eq. (3.23) muestra la derivada de las variables auxiliares.

$$\begin{aligned} \dot{x_1} &= \dot{\theta} = x_2 \\ \dot{x_2} &= \frac{2\beta\gamma\dot{\phi}\dot{\theta}\cos^2\theta\sin\theta - \gamma^2\dot{\theta}^2\cos\theta\sin\theta}{\alpha\beta-\gamma^2+(\beta^2+\gamma^2)\sin^2\theta} + \frac{\delta(\alpha+\beta\sin^2\theta)\sin\theta - \gamma u\cos\theta}{\alpha\beta-\gamma^2+(\beta^2+\gamma^2)\sin^2\theta} + \frac{\beta\dot{\phi}^2(\alpha+\beta\sin^2\theta)\cos\theta\sin\theta}{\alpha\beta-\gamma^2+(\beta^2+\gamma^2)\sin^2\theta} \\ \dot{x_3} &= \dot{\phi} = x_4 \\ \dot{x_4} &= \frac{\beta u}{\alpha\beta-\gamma^2+(\beta^2+\gamma^2)\sin^2\theta} - \frac{\beta\gamma\dot{\phi}^2\cos^2\theta\sin\theta}{\alpha\beta-\gamma^2+(\beta^2+\gamma^2)\sin^2\theta} - \frac{2\beta^2\dot{\phi}\dot{\theta}\cos\theta\sin\theta}{\alpha\beta-\gamma^2+(\beta^2+\gamma^2)\sin^2\theta} + \frac{\beta\gamma\dot{\theta}^2\sin\theta - \gamma\delta\cos\theta\sin\theta}{\alpha\beta-\gamma^2+(\beta^2+\gamma^2)\sin^2\theta} \end{aligned} \tag{3.23}$$

Como siguiente paso se linealiza las Eqs. (3.23) de acuerdo a las condiciones que se presentan en la Sección 3.3.3. Las ecuaciones linealizadas se muestran de forma matricial en la Eq. (3.24).

$$\dot{\mathbf{x}} = \mathbf{A}x + \mathbf{B}u = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{\beta\delta}{\alpha\beta-\gamma^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{-\gamma\delta}{\alpha\beta-\gamma^2} & 0 & 0 & 0 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 0 \\ \frac{-\gamma g}{\alpha\beta-\gamma^2} \\ 0 \\ \frac{\alpha g}{\alpha\beta-\gamma^2} \end{bmatrix}\mathbf{u} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 49{,}7274 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -0{,}7845 & 0 & 0 & 0 \end{bmatrix}\mathbf{x} + \begin{bmatrix} 0 \\ -49{,}5963 \\ 0 \\ 293{,}4741 \end{bmatrix}\mathbf{u} \tag{3.24}$$

Ahora se aplica la ley de control de la Eq. (2.38). El vector $\mathbf{K}$ de la Eq. (2.38) se lo encuentra con la fórmula de Ackerman que se explica en la Sección 2.2.4, con un tiempo de muestreo de 5 ms. Los polos que se utilizan en la fórmula de Ackerman son los polos que da el polinomio de Bessel de cuarto orden (Ver Eq. (3.25)).

$$\begin{aligned} p_1 &= 4{,}0156 + j5{,}0723 \\ p_2 &= 4{,}0156 - j5{,}0723 \\ p_3 &= 5{,}5281 + j1{,}6553 \\ p_4 &= 5{,}5281 - j1{,}6553 \end{aligned} \tag{3.25}$$

Para obtener el vector de retroalimentación se utiliza el comando *acker()* de Matlab. Este comando pide como entradas las matrices del espacio de estados del sistema (Eq. (3.24)), la ubicación de los polos (Eq. (3.25)) y el tiempo de muestreo. Su salida se muestra en la Eq. (3.26).

$$\mathbf{K} = \begin{bmatrix} -4{,}860 & -0{,}679 & -0{,}0953 & -0{,}050 \end{bmatrix} \tag{3.26}$$

Una vez que se tiene el vector de retroalimentación de Ackerman se procesa el algoritmo de control. El diagrama de flujo se observa en la Fig. 3.13.

La Fig. 3.13 muestra el diagrama de flujo de todo el proceso. Antes del proceso de control, se realiza la

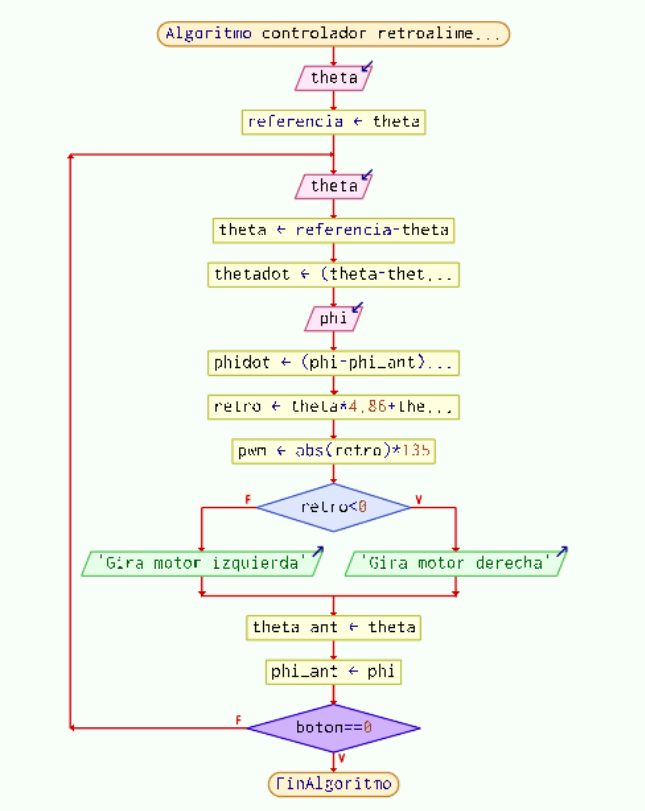


Figura 3.13: Diagrama de flujo del controlador de retroalimentación de estados para péndulo en estado erguido

calibración del sensor del péndulo (θ) para indicar que la primera lectura de la posición del péndulo es el cero del sistema. Una vez calibrado el sensor inicia el proceso de control. Este proceso lee las variables θ y ϕ, calcula las velocidades instantáneas $\dot{\theta}$, $\dot{\phi}$ y arma el vector de datos $[\theta\ \dot{\theta}\ \phi\ \dot{\phi}\ ]'$. Después obtiene el valor de la retroalimentación, para lo cual multiplica el vector de datos con el vector de retroalimentación de la Eq. (3.26).

Además, calcula el valor del PWM para lo cual toma el valor absoluto de la retroalimentación y lo multiplica por una constante, para este modelo es 135. Ahora se determina el sentido de giro de motor con un valor dado de PWM, el cual es dependiente del signo de la retroalimentación. Es decir, si el valor de la retroalimentación es positivo, el motor gira hacia la izquierda, caso contrario, gira hacia la derecha. Por último, se guardan los valores de θ y ϕ en variables auxiliares y se repite el proceso de control hasta que el botón de activación cambie de estado o el péndulo se sature y caiga.

El algoritmo de control de retroalimentación de estados se lo implementa en las tarjetas electrónicas Arduino y FPGA. En el caso de Arduino la adquisición de datos, el procesamiento de datos, y la orden de control se lo realiza en la misma tarjeta. Mientras que, en el caso del FPGA, la adquisición de datos se lo realiza con la ayuda del Arduino, el cual envía los datos por el puerto serie al FPGA; este último se encarga de procesar y enviar la orden de control al motor.

Capítulo 4

Resultados

En esta sección se presenta una descripción detallada de los resultados obtenidos para el controlador PID en Arduino (Sección 4.1), del controlador PID en FPGA (Sección 4.2), del controlador de retroalimentación de estados en Arduino y FPGA (Sección 4.3). La adquisición de datos de cada controlador se lo hace con una tarjeta *NI-DAQmx* de National Instruments. Esta tarjeta digitaliza los datos de manera continua con un tiempo de muestreo de 5 ms usando Labview.

En la implementación de los controladores PID tanto en Arduino como en FPGA la calibración de los parámetros del algoritmo se lo realiza de una manera heurística con el método de prueba error (Sección 2.2.1). Donde la variación del parámetro K_p afecta directamente al error del sistema en régimen permanente y la variación de K_d afecta a la respuesta transitoria del sistema.

En la implementación de los controladores de retroalimentación de estados en Arduino y en FPGA, al ser controladores diseñados a partir de su modelo matemático y simulados en entornos virtuales, no necesita realizar pruebas para calibrar el vector de retroalimentación a la planta real, como es el caso del controlador PID, que se lo hace bajo el método heurístico de prueba error para la calibración de sus parámetros.

Después de realizar la adquisición de datos de cada experimento de los controladores (PID, retroalimentación de estados), se calcula los índices de error promedio (Eq. (4.1)), energía promedio (Eq. (4.2)) y energía total (Eq. (4.3)).

$$IEP = \frac{\sum_1^n |Theta - referencia|}{n} \tag{4.1}$$

$$ITP = \frac{\sum_1^n \frac{1}{2} * m * v^2}{n} \tag{4.2}$$

$$ITT = \sum_1^n \frac{1}{2} * m * v^2 \tag{4.3}$$

4.1. Controlador PID en Arduino

En el primer experimento (Fig. 4.1) se observa que el péndulo arranca estabilizado, pero ante una pequeña perturbación, el péndulo empieza a oscilar alrededor de la referencia sin llegar a estabilizarse. Durante este proceso su índice de error promedio es 4,07°, el índice de energía promedio es 0,6852 J y su índice de energía total es 701 J.

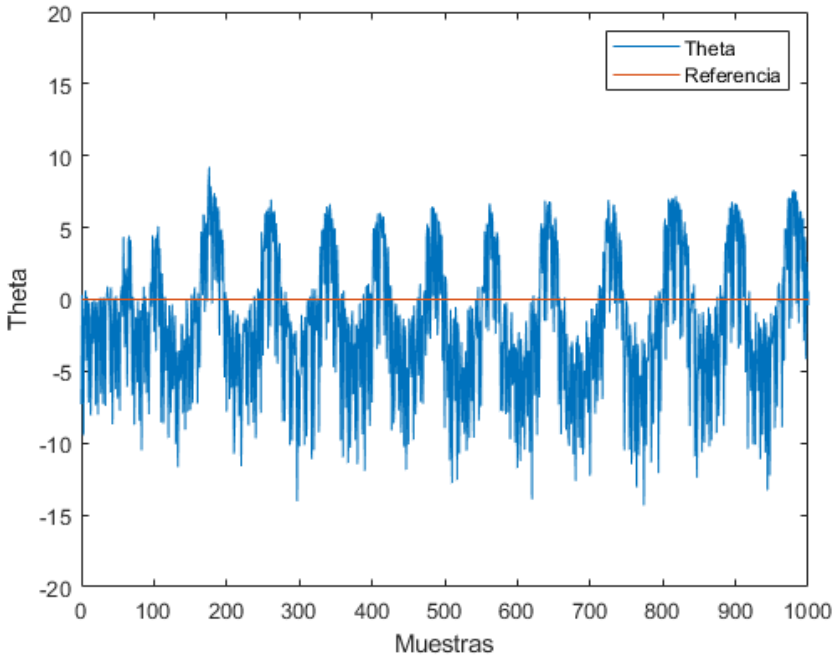


Figura 4.1: Respuesta del primer experimento de PID en Arduino ($K_p = 4,9$, $K_i = 0$, $K_d = 10$).

En el segundo experimento, para disminuir el error del sistema en régimen permanente se disminuye su ganancia proporcional K_p. La Fig. 4.2 presenta el resultado; se observa que el péndulo mantiene sus oscilaciones alrededor de la referencia sin llegar a estabilizarse. Sin embargo, su índice de error promedio disminuye a 3,99°. Durante esta prueba, la energía promedio en el movimiento del péndulo es 0,62 J y su energía total es 614 J.

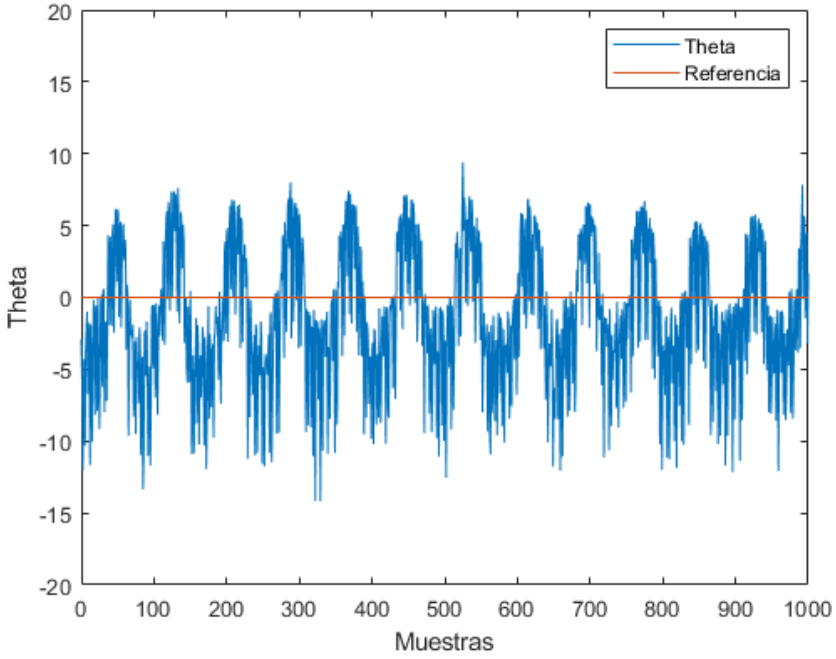


Figura 4.2: Respuesta del segundo experimento de PID en Arduino ($K_p = 4,7$, $K_i = 0$, $K_d = 10$).

La Fig. 4.3 muestra el resultado del tercer experimento. En este se aumenta la respuesta transitoria al sistema a través de la constante K_d. Los ajustes se realizan para aumentar la velocidad de respuesta del sistema. El péndulo se mantiene por un momento estabilizado, pero al recibir una perturbación, el péndulo oscila alrededor de la referencia sin llegar a estabilizar su posición. Sus índices de error y de energía disminuyen en relación a los del controlador con parámetros $K_p = 4{,}7$, $K_i = 0$, $K_d = 10$ (Ver Tabla 4.1).

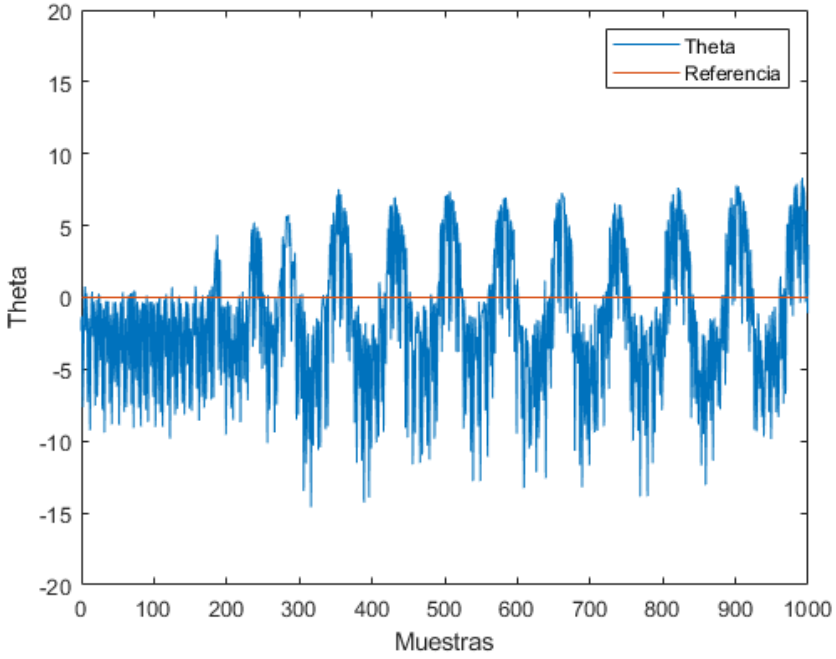


Figura 4.3: Respuesta del tercer experimento de PID en Arduino ($K_p = 4,7$, $K_i = 0$, $K_d = 10,9$).

En el tercer experimento (Fig. 4.3), al aumentar el valor de la constante K_d, el péndulo tiene un periodo de estabilización, pero aun persiste la inestabilidad del sistema. En el cuarto experimento se disminuye K_d y se observa que el sistema vuelve a oscilar alrededor de la referencia sin estabilizarlo (Ver Fig. 4.4). Sin embargo, su índice de error promedio disminuye, siendo este 2,67°. El índice de energía promedio y total también disminuyen a 0,37 J y 388 J, respectivamente.

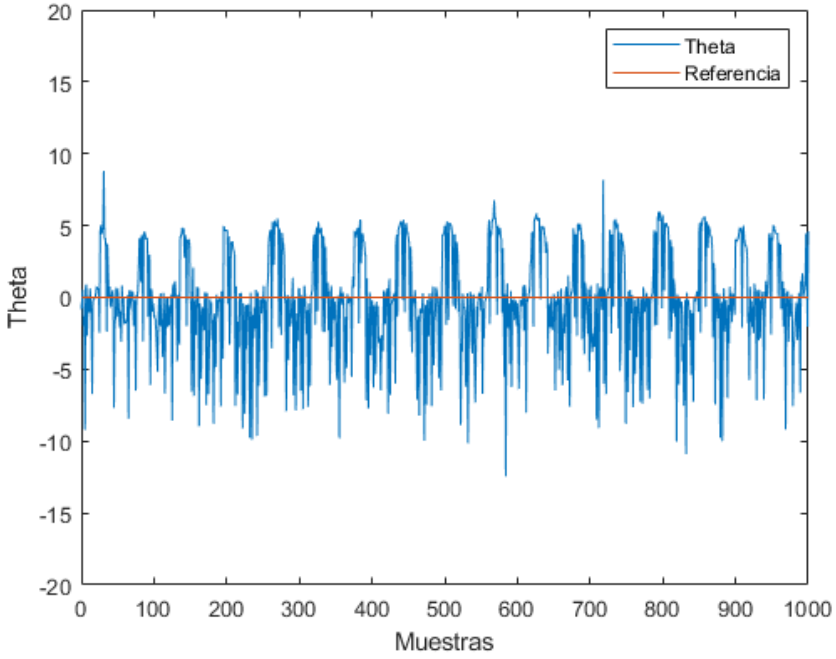


Figura 4.4: Respuesta del cuarto experimento de PID en Arduino ($K_p = 4,7$, $K_i = 0$, $K_d = 10,3$).

Después de disminuir la respuesta transitoria en el cuarto experimento y ver que el sistema regresa a oscilar alrededor de la referencia sin mostrar periodos de estabilización, en el quinto experimento se aumenta levemente la constante K_d. La respuesta de este experimento se muestra en la Fig. 4.5, en donde el sistema sigue oscilando alrededor de la referencia, pero presenta pequeñas etapas de estabilización. El índice de error promedio aumenta a 3,92° y los índices de energía promedio y energía total aumentan a 0,41 J y 425 J, respectivamente.

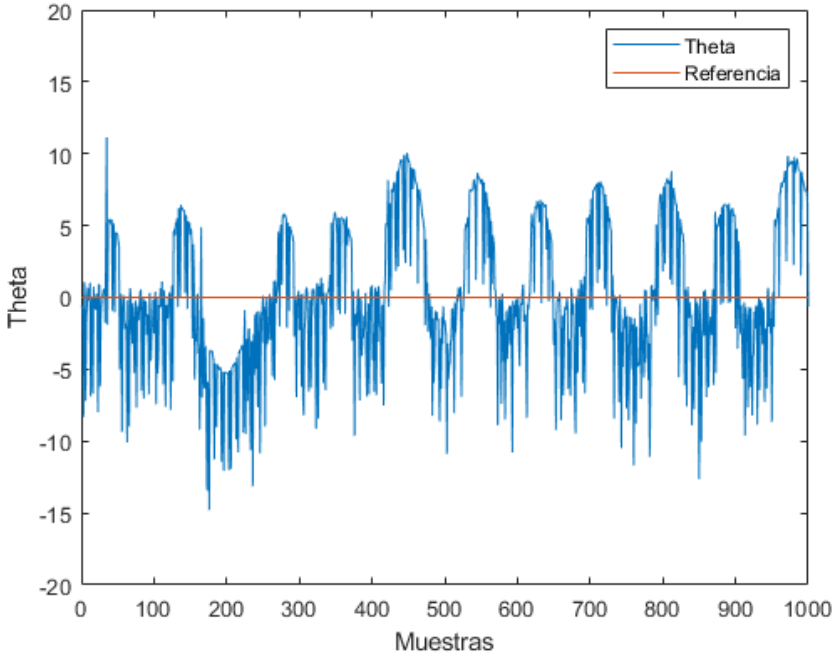


Figura 4.5: Respuesta del quinto experimento de PID en Arduino ($K_p = 4,7$, $K_i = 0$, $K_d = 10{,}5$).

Al aumentar un poco la constante derivativa K_d se ganan unos reducidos periodos de estabilidad (Ver Fig. 4.5). Por lo cual, en el sexto experimento nuevamente se aumenta la constante derivativa, y su respuesta se observa en la Fig. 4.6, en donde el péndulo llega a estabilizarse. Sin embargo, presenta ligeras perturbaciones a las cuales el sistema de control responde eficientemente volviendo a estabilizar el péndulo. En este experimento disminuye el índice de error promedio, energía promedio y energía total siendo estos 1,83°, 0,40 J y 416 J, respectivamente.

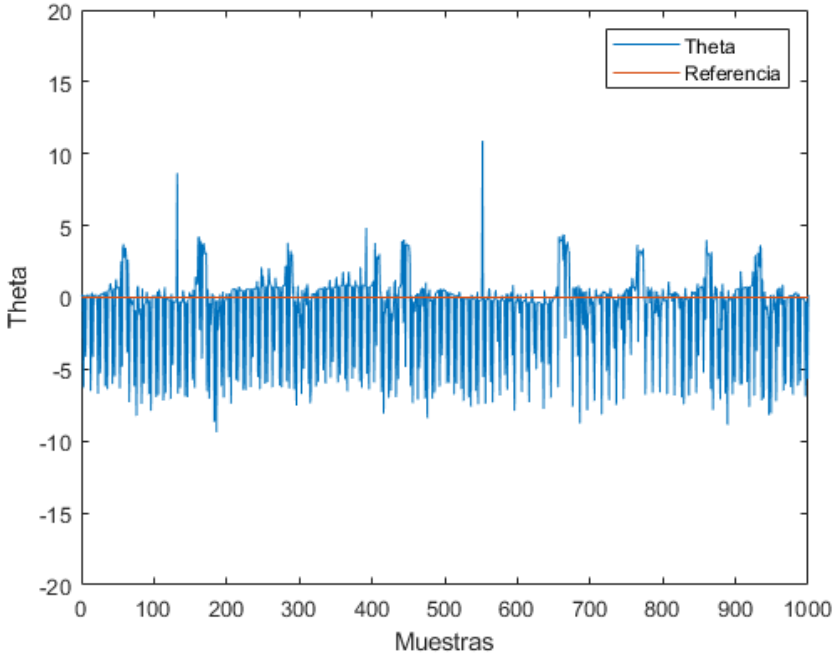


Figura 4.6: Respuesta del sexto experimento de PID en Arduino ($K_p = 4,7$, $K_i = 0$, $K_d = 10,7$).

En la Tabla 4.1 se observa que el controlador PID que mejor respuesta tiene de acuerdo a su índice de error promedio, es el tercer controlador con constantes $K_p = 4{,}7$, $K_i = 0$ y $K_d = 10{,}7$. Sin embargo, de acuerdo a los índices de energía, el de mejor respuesta es el controlador con constantes $K_p = 4{,}7$, $K_i = 0$ y $K_d = 10{,}3$. En esta planta lo que favorece a su estabilidad es que el índice de error promedio sea lo más cercano a cero. Por lo tanto, el controlador que mejor respuesta tiene es el controlador del experimento 6, con constantes $K_p = 4{,}7$, $K_i = 0$ y $K_d = 10{,}7$.

Tabla 4.1: Índices de error y energía para controlador PID en Arduino

Experimento	Kp	Ki	Kd	Índice Error Promedio [°]	Índice Energía Promedio [J]	Índice Energía Total [J]	Resultado
1	4,9	0	10	4,07	0,68	701	Fig. 4.1
2	4,7	0	10	3,99	0,62	641	Fig. 4.2
3	4,7	0	10,9	3,94	0,60	620	Fig. 4.3
4	4,7	0	10,3	2,67	0,37	388	Fig. 4.4
5	4,7	0	10,5	3,92	0,41	425	Fig. 4.5
6	4,7	0	10,7	1,83	0,40	416	Fig. 4.6

4.2. Controlador PID en FPGA

Una vez obtenidos los resultados del controlador PID en Arduino (Sección 4.1), como primer experimento para este controlador se utilizan los valores de las constantes K_p, K_i, K_d del sexto experimento del controlador PID en Arduino. Los resultados de este experimento se muestran en la Fig. 4.7. Donde se observa que el péndulo no se estabiliza, oscila mucho tiempo alrededor de los 20° y al final termina oscilando alrededor de la referencia.

En este experimento, su índice de error promedio es 24,16° siendo este un error elevado y no aceptable para este tipo de aplicación.

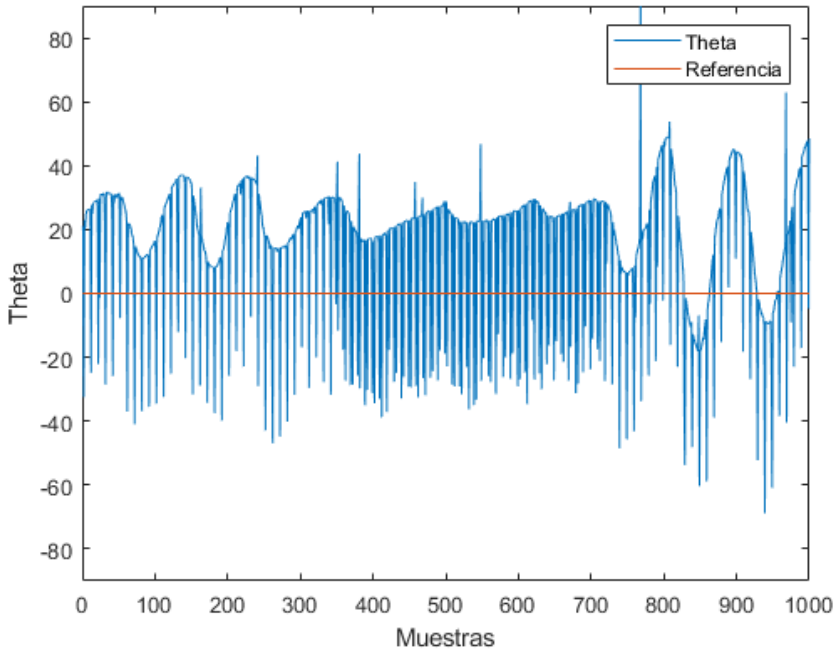


Figura 4.7: Respuesta del primer experimento de PID en FPGA ($K_p = 4,7$, $K_i = 0$, $K_d = 10,7$).

A partir de las constantes del primer experimento, se realiza un segundo experimento disminuyendo la constante K_d con el objetivo de disminuir las oscilaciones del péndulo. En la Fig. 4.8 se presenta el resultado del experimento y se observa como el algoritmo de control consigue el objetivo, pero el péndulo sigue oscilando alrededor de los 20°. Su índice de error promedio es 22,08°, siendo aún inaceptable este error.

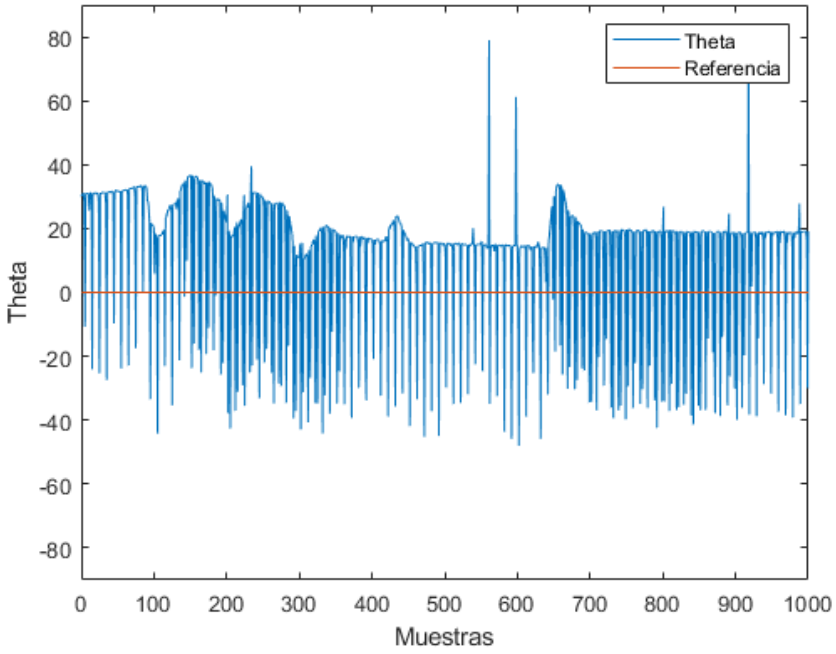


Figura 4.8: Respuesta del segundo experimento de PID en FPGA ($K_p = 4,7$, $K_i = 0$, $K_d = 10,9$).

En el segundo experimento (Fig. 4.8) la referencia en donde oscila el péndulo está desfasada alrededor de los 20°. Como tercer experimento se modifica la constante proporcional para que el péndulo oscile alrededor de la referencia. En la Fig. 4.9 se muestra la respuesta del tercer experimento. El algoritmo con esta modificación consigue que el péndulo oscile alrededor de la referencia, pero aún presenta oscilaciones fuertes. El índice de error promedio aumenta a 14,63°.

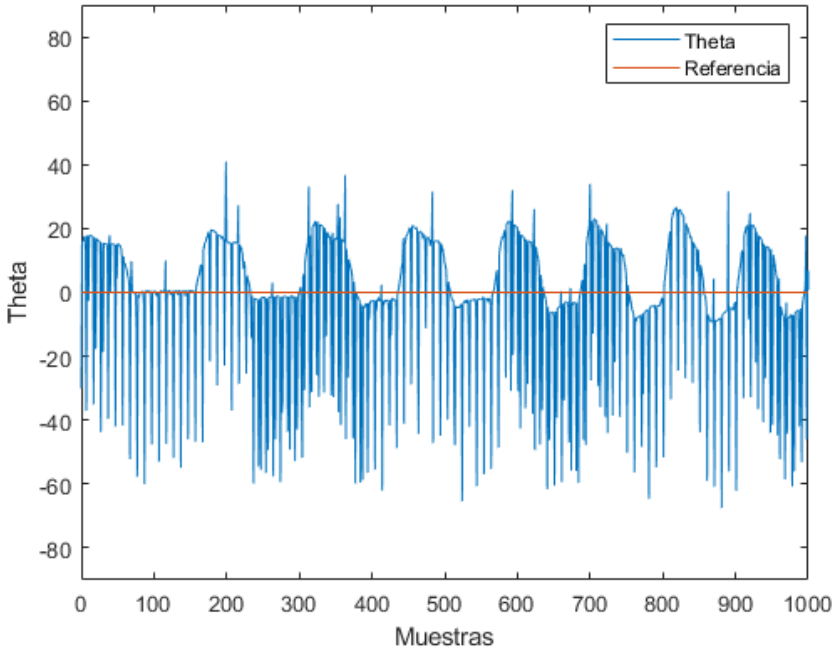


Figura 4.9: Respuesta del tercer experimento de PID en FPGA ($K_p = 4,6$, $K_i = 0$, $K_d = 10,9$).

En el cuarto experimento se aumenta el valor de la constante derivativa K_d. La respuesta de este experimento se observa en la Fig. 4.10. El algoritmo de control hace menos estable al sistema y aumenta su índice de error promedio a 15,74°.

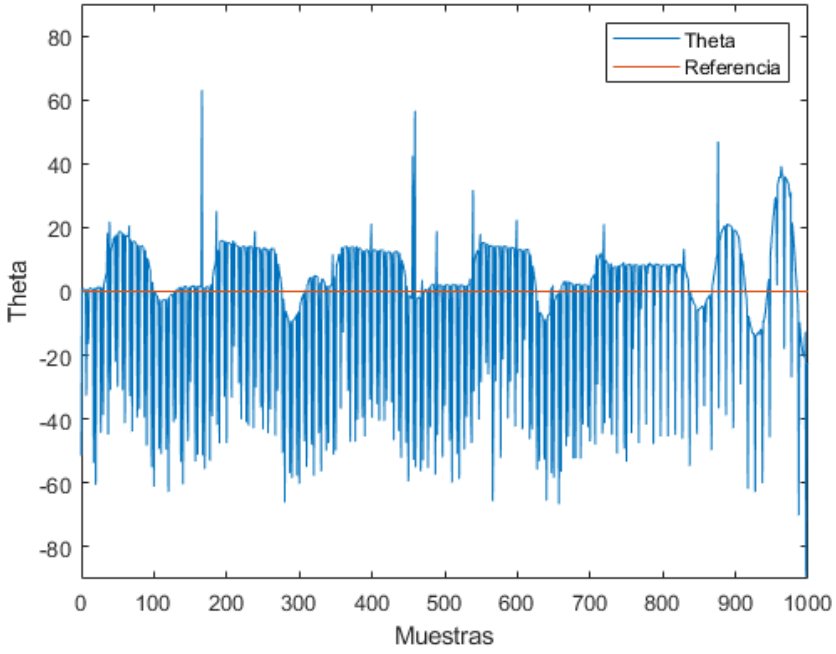


Figura 4.10: Respuesta del cuarto experimento de PID en FPGA ($K_p = 4,6$, $K_i = 0$, $K_d = 11$).

En el quinto experimento se disminuye drásticamente el valor de la constante derivativa. La Fig. 4.11 presenta la respuesta del quinto experimento, donde se observa que el algoritmo de control disminuye las oscilaciones alrededor de la referencia, pero aún presenta un valor elevado del índice de error promedio (Ver Tabla 4.2).

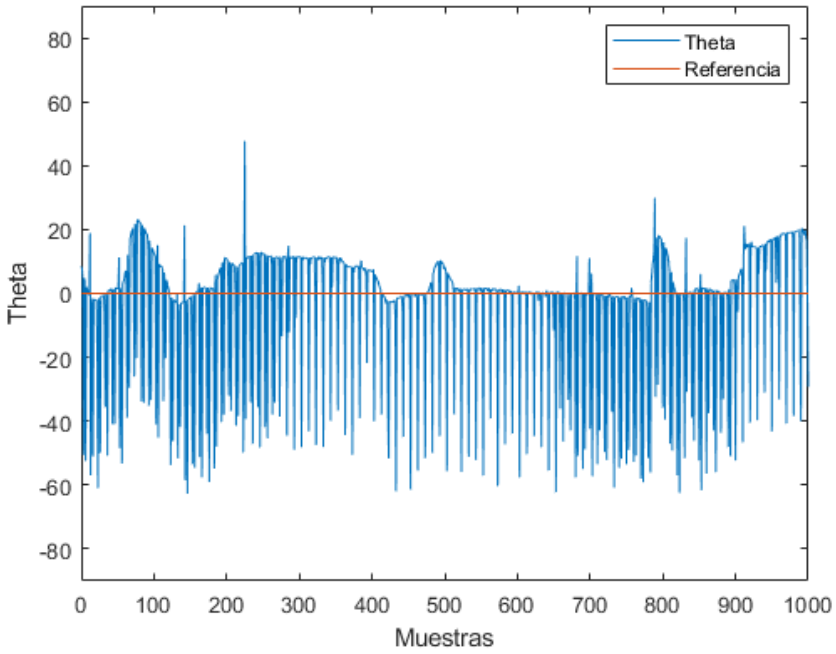


Figura 4.11: Respuesta del quinto experimento de PID en FPGA ($K_p = 4,6$, $K_i = 0$, $K_d = 10$).

En el sexto experimento se disminuye en una unidad el valor de la constante derivativa y para contrarrestar los efectos de este cambio, también se disminuye el valor de la constante proporcional en 0,1 unidades. Su respuesta (Fig. 4.12) muestra que el péndulo se estabiliza alrededor de la referencia. Su índice de error promedio, su índice de energía promedio y su índice de energía total disminuyen notablemente respecto a los índices de los experimentos anteriores (Ver Tabla 4.2).

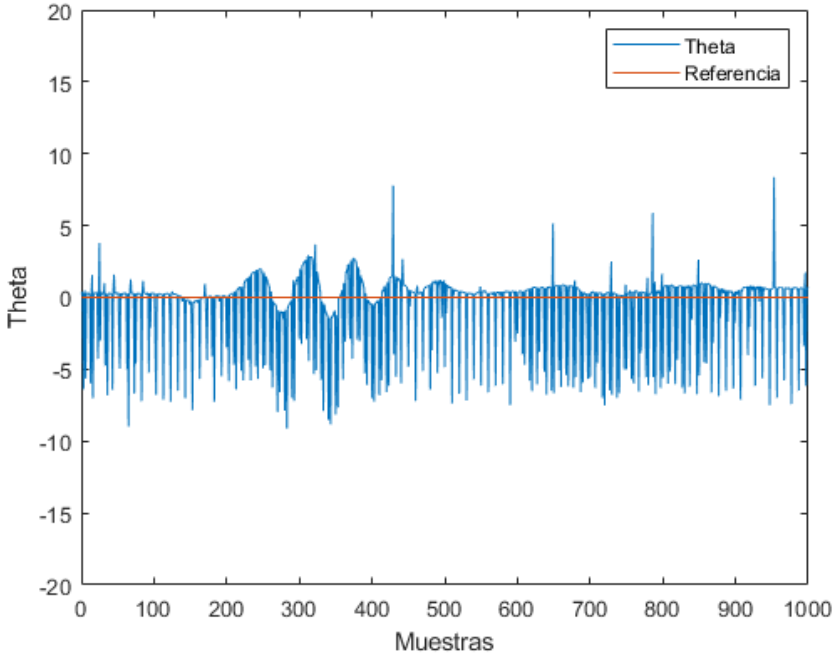


Figura 4.12: Respuesta del sexto experimento de PID en FPGA ($K_p = 4,5$, $K_i = 0$, $K_d = 9$).

En la Tabla 4.2 se muestra que los primeros cinco experimentos presentan sus índices de energía y error elevados y en sus figuras (Ver Fig. 4.7, 4.8, 4.9, 4.10, 4.11) se observa que el péndulo en ninguna de estos experimentos logra estabilizarse teniendo oscilaciones respecto a la referencia altas y alejadas. Sin embargo, el sexto experimento maneja índices de error y energía bajos en comparación a los anteriores, y esto se debe a que sus oscilaciones son pocas y pequeñas alrededor de su referencia.

Tabla 4.2: Índices de error y energía para controlador PID en FPGA

Experimento	Kp	Ki	Kd	Índice Error Promedio [°]	Índice Energía Promedio [J]	Índice Energía Total [J]	Resultado
1	4,7	0	10,7	24,16	17,72	18k	Fig. 4.7
2	4,7	0	10,9	22,08	19,30	19k	Fig. 4.8
3	4,6	0	10,9	14,63	17,69	18k	Fig. 4.9
4	4,6	0	11	15,74	22,45	23k	Fig. 4.10
5	4,6	0	10	12,23	18,72	19k	Fig. 4.11
6	4,5	0	9	1,54	0,32	333	Fig. 4.12

4.3. Controlador de retroalimentación de estados

En el primer experimento se implementa el controlador de retroalimentación de estados en la tarjeta Arduino y su respuesta se observa en la Fig. 4.13. Como segundo experimento se implementa el mismo controlador en la tarjeta FPGA y su respuesta se observa en la Fig. 4.14.

Tanto en el primer experimento (Fig. 4.13) como en el segundo experimento (Fig. 4.14), se observa que el péndulo arranca estabilizado y ante pequeñas perturbaciones el sistema de control responde rápidamente y

estabiliza el péndulo. Los índices de error promedio, energía promedio y energía total se presentan en la Tabla 4.3.

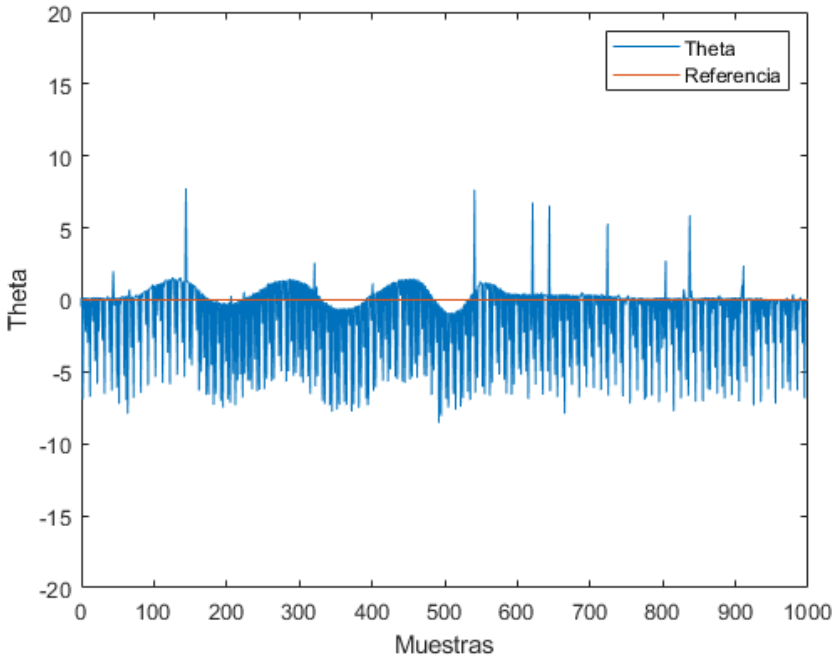


Figura 4.13: Respuesta del péndulo con controlador de retroalimentación de estados en Arduino.

En el caso del sistema de control implementado en la tarjeta FPGA se observa que el péndulo se estabiliza en un punto muy cercano a su referencia, mientras que en el caso del sistema de control implementado en Arduino la estabilidad del péndulo se encuentra justo en su punto de referencia.

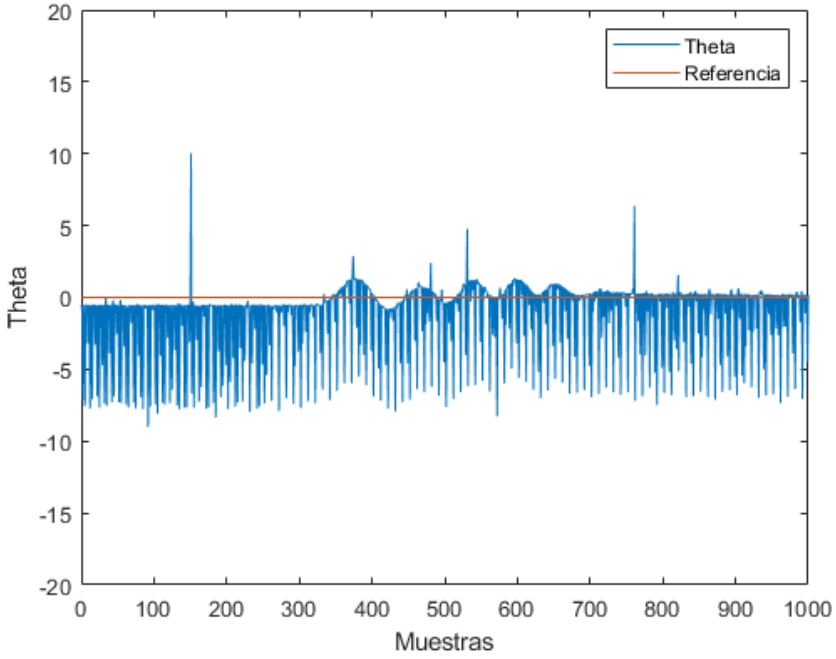


Figura 4.14: Respuesta del péndulo con controlador de retroalimentación de estados en FPGA.

En la Tabla 4.3 se observan los índices de error promedio, energía promedio y energía total para un controlador de retroalimentación de estados implementado en diferentes tarjetas electrónicas. Se observa que el controlador implementado en la FPGA tiene la mejor respuesta al evaluar los diferentes índices.

Tabla 4.3: Índices de error y energía para controlador de retroalimentación de estados.

Experimento	**Índice Error Promedio [°]**	**Índice Energía Promedio [J]**	**Índice Energía Total [J]**	**Resultado**
Arduino	1.76	0.40	414	Fig. 4.13
FPGA	1.57	0.35	358	Fig. 4.14

Capítulo 5

Conclusiones y recomendaciones

5.1. Conclusiones

En este trabajo se ha implementado un controlador PID y un controlador de retroalimentación de estados en las tarjetas electrónicas Arduino y FPGA, para controlar un péndulo de Furuta. El sistema de control en Arduino se realiza de forma secuencial, y el sistema de control en FPGA se realiza de forma paralela.

Primero, se realizó la implementación del algoritmo de control PID en Arduino y en FPGA. Los parámetros K_p, K_i y K_d de los controladores son calibrados usando un proceso heurístico de prueba y error. Observando y comparando los resultados de las Tablas 4.1 y 4.2, se determina que los controladores PID que mejor responden, tanto en Arduino como en FPGA, son el sexto experimento de cada sección.

Por otra parte, se implementa un controlador de retroalimentación de estados en Arduino y en FPGA. El vector de retroalimentación se encuentra con la fórmula de Ackerman, a partir de su modelo matemático. Observando y comparando los resultados de la Tabla 4.3, los dos algoritmos de control implementados responden de manera correcta y estabilizan al péndulo.

De acuerdo a los índices visualizados en las Tablas 4.1 y 4.2, el controlador PID que mejor responde, es el sexto experimento implementado en FPGA. Con respecto al controlador de retroalimentación de estados, en relación a sus índices mostrados en la Tabla 4.3, el controlador que mejor responde, nuevamente es el implementado en FPGA.

En la estabilización del péndulo, los dos controladores presentan resultados similares de acuerdo a sus índices presentados en las Tablas 4.1, 4.2 y 4.3. Es por esto que para decidir cuál de los dos algoritmos implementar en el control del péndulo de Furuta, se basa en la cantidad de información técnica que se tenga de la planta. Si se tiene una planta de la cual no se conocen sus detalles técnicos o se tiene poca información, la técnica de control que mejor se acopla es el controlador PID. Caso contrario, si se conocen los detalles técnicos de la planta, el controlador que mejor se acopla es el de retroalimentación de estados.

5.2. Recomendaciones

En la implementación de los controladores en la tarjeta Arduino, activar el puerto serial para visualizar los resultados en el computador implica aumentar el tiempo de muestreo y por lo tanto el control se vuelve más complejo, ya que la respuesta debe de ser lo más rápida posible.

Para diseñar el controlador de retroalimentación de estados, se recomienda comparar la respuesta simulada con la respuesta del sistema real y hacer que la simulación sea lo más parecida a la respuesta real. En caso de tener una respuesta diferente entre lo simulado y lo real, se recomienda utilizar el *estimador de parámetros* de Matlab.

5.3. Trabajos futuros

Este trabajo implementa una primera versión del controlador en espacio de estados y aún se observa que un estado queda inestable. Por este motivo, lo ideal como una segunda versión es diseñar un controlador de modos deslizantes en espacio de estados, de esta forma se asegura un control estable. Con respecto al controlador PID, se propone implementar una segunda versión del controlador con un sistema de control en cascada, es decir, un controlador PID adicional para regular la velocidad del motor.

Bibliografía

[1] "Una aplicación del péndulo de Furuta," 2018.

[2] S. Mori, H. Nishihara, y K. Furuta, "Control of unstable mechanical system Control of pendulum," *International Journal of Control*, vol. 23, num. 5, pp. 673–692, may 1976. [En línea]. Disponible: https://www.tandfonline.com/doi/full/10.1080/00207177608922192

[3] C. Olivero y N. Jimenez Serrata, "Modelos de péndulo invertido," 07 2016.

[4] J. Åkesson, "Safe manual control of unstable systems," 2000.

[5] M. W. Spong y L. Praly, "Control of underactuated mechanical systems using switching and saturation," in *Control Using Logic-Based Switching*, A. Stephen Morse, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 162–172.

[6] A. Valera, M. Vallés, y M. Cardo, "Diseño y control de un péndulo de Furuta," 2017.

[7] Octavio Augusto García Alarcón, "Diseño, construcción y control de un péndulo de Furuta," Tesis Maestria, Instituto, Mexico, 2013. [En línea]. Disponible: http://tesis.ipn.mx/handle/123456789/20260

[8] L. F. Escobar-Dávila, O. D. Montoya-Giraldo, y D. Giraldo-Buitrago, "Control Global del Péndulo de Furuta Empleando Redes Neuronales Artificiales y Realimentación de Variables de Estado," *TecnoLógicas*, pp. 71 – 94, 06 2013. [En línea]. Disponible: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0123-77992013000100005&nrm=iso

[9] J. Medina Cervantes, J. E. Gallardo Sánchez, R. Villafuerte Díaz, y E. Mejía Sánchez, "Desarrollo de un péndulo de furuta," *Revista Electrónica sobre Tecnología, Educación y Sociedad*, vol. 4, num. 7, ene. 2017. [En línea]. Disponible: https://www.ctes.org.mx/index.php/ctes/article/view/619

[10] J. G. G. Fontanet, A. L. Cervantes, y I. B. Ortiz, "Alternativas de control para un Péndulo de Furuta," *Revista Iberoamericana de Automática e Informática Industrial RIAI*, vol. 13, num. 4, pp. 410–420, oct 2016. [En línea]. Disponible: https://polipapers.upv.es/index.php/RIAI/article/view/9255

[11] S. Casco, "Raspberry pi, arduino y beaglebone black comparación y aplicaciones," *vol*, vol. 1, pp. 4–8, 2014.

[12] O. T. Artero, *ARDUINO. Curso práctico de formación*. RC Libros, 2013, google-Books-ID: 6cZhDmf7suQC.

[13] J. M. López Monterroso, "Fpga libre alhambra ii como herramienta de aprendizaje para la electrónica digital," Ph.D. dissertation, Universidad de San Carlos de Guatemala, 2020.

[14] A. Balean, "Aplicación de la placa fpga icezum alhambra en prácticas de laboratorio de automatización de sistemas," *Universitat Jaume I*, 2018.

[15] K. Ogata, *Ingeniería de control moderna*. Pearson Educación, 2003.

[16] F. Gonzalez-Longatt, "Introducción a la teoría de control," 01 2008.

[17] G. H. Hostetter, C. J. Savant, y R. Stefani, *Sistemas de control*. Nueva editorial Interamericana S.A. de C.V., 1984.

[18] M. A. Moreno, "apuntes de control pid," *La Paz Enero*, vol. 8, pp. 6–7, 2001.

[19] S. Castro, *Teoría de control. Diseño electrónico*, ser. Politext: Tecnología eléctrica y electrónica. Edicions de la UPC, S.L., 1998. [En línea]. Disponible: https://books.google.com.ec/books?id=Jro3rHU_urMC

[20] F. M. García, "El controlador pid," *Online] Disponible en: http://es. slideshare. net/MnicaMoreno/el-controlador-pid*, 2007.

[21] V. Mazzone, "Controladores pid," *Quilmes: Universidad Nacional de Quilmes*, 2002.

[22] M. A. P. L. Medina, M. Saba, M. de Guevara Durán, y M. Silva, "Controladores pid y controladores difusos," *Revista de ingeniería industrial*, vol. 5, num. 1, 2011.

[23] V. M. A. Ruíz, "Métodos de sintonización de controladores pid que operan como reguladores," *Revista Ingeniería*, vol. 12, num. 1-2, pp. 21–36, 2002.

[24] O. F. Guiñansaca Soria, "Diseño y construcción de una interfaz háptica de un grado de libertad con interfaz hombre-máquina."

[25] K. Ogata, *Sistemas de control en tiempo discreto*. Pearson educación, 1996.

[26] D. R. Ramírez y C. B. Alba, "Apuntes de ingeniería de control," *2005*, 2005.

[27] M. Gäfvert, "Modelling the furuta pendulum," *Department of Automatic Control, Lund Institute of Technology (LTH)*, 1998.

[28] A. E. Salazar Andrade, "Estudio comparativo del desempeño de tres tipos de controladores para el péndulo invertido furuta." B.S. thesis, 2018.

[29] C. Regalo Núñez, "Control y simulación del péndulo de furuta," 2016.

[30] Ismael Minchala Avila y Rubén Marbán Romero, "Nonlinear Control of the Furuta Pendulum."

[31] M. Ing, A. Oscar, y O. Bellon Hernandez, "Descripción teórica de un procedimiento para determinar los parámetros de un motor de corriente continua theoretical description of a method for determining the parameters of a direct current motor," 03 2014.

[32] H. D. Young, R. A. Freedman *y otros*, *Sears-Zemansky física universitaria*, 2009.