

UNIVERSIDAD NACIONAL DE TRUJILLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA MECATRÓNICA



IDENTIFICACIÓN NO PARAMÉTRICA DE UN MOTOR DC POR MEDIO DE UN MODELO DE MACHINE LEARNING PARA SINTONIZAR UN CONTROLADOR PID DE VELOCIDAD APLICANDO ALGORITMOS GENÉTICOS

**TESIS
PARA OPTAR EL TÍTULO PROFESIONAL DE
INGENIERO MECATRÓNICO**

AUTOR:

Br. Pereyra González, Irvin Jair

ASESOR:

Mg. Ing. León Lescano, Edward Javier

Trujillo, Perú

2018

AGRADECIMIENTO

A Dios, por ser el amigo que jamás falla y fuente de la fuerza para seguir siempre adelante.

Al Mg. Javier León Lescano, mi asesor, por el fundamental apoyo brindado a lo largo del desarrollo del presente trabajo.

Al Ing. Jairo Rodriguez Ochoa y al Ing. Luis Gonzales Dávila, docentes de la escuela de Ingeniería Mecatrónica, por su apoyo valioso y sincero.

DEDICATORIA

A mis padres, Gladys y Víctor, cuyo sacrificio diario por amor a su familia, los hace merecerse lo mejor del mundo.

A mis abuelos, Gladis, Nélida, José y Víctor, por su amor inmenso, apoyo incondicional, y enseñanzas que han marcado enormemente mi vida.

A mis hermanas, Lorena y Alina, por ser mis acompañantes más importantes y que vuelven más divertido este caótico viaje que es la vida.

RESUMEN

El presente trabajo se centra en la identificación no paramétrica de un motor dc a través de un modelo de machine learning, y cómo este modelo puede ser incluido en un proceso de sintonización basado en algoritmos genéticos para hallar un controlador PID de velocidad; cuyo desempeño fue comparado respecto al de los controladores PID obtenidos mediante el primer método de Ziegler-Nichols y la aplicación PID Tuner de Matlab.

Para la identificación del motor dc por medio de un modelo de machine learning, se utilizó una señal de excitación tipo escalón con una amplitud cambiante cada 4.798s, y la estructura de modelo seleccionada fue el modelo de regresión lineal Elastic Net; mientras que, para la identificación por medio de una función de transferencia, se utilizó una señal de excitación tipo escalón con amplitud constante. El modelo de Elastic Net, mostró un ajuste de más del 99% sobre un conjunto de prueba; mientras que la función de transferencia mostró un ajuste de más de un 95%.

La sintonización de los controladores PID se realizaron por medio de tres estrategias: La primera utilizó algoritmos genéticos y el criterio de la integral del tiempo multiplicada por el valor absoluto del error; la segunda. el primer método de Ziegler-Nichols; y la tercera, la aplicación PID Tuner de Matlab. La función de transferencia fue utilizada en los dos últimos métodos, mientras que el modelo de Elastic Net, en el primero.

Se evaluaron los controladores obtenidos sobre el sistema de control, encontrándose que el controlador obtenido por el método de Ziegler-Nichols provocaba una respuesta oscilante que no llegaba a estabilizarse. Para el caso de los otros métodos, ambos controladores mostraron ser relativamente adecuados; sin embargo, el controlador obtenido por medio de algoritmos genéticos y el modelo de Elastic Net mostró menor oscilación antes de estabilizarse y un menor tiempo de establecimiento.

Se utilizó una placa Arduino UNO R3 para implementar tanto el muestreo de datos como el controlador digital.

ABSTRACT

The present work focuses on a dc motor's nonparametric identification by means of a machine learning model, and how this model can be included in a genetic algorithms based PID tuning process to find a speed PID controller; whose performance was compared with that of the PID controllers obtained by the first Ziegler-Nichols method and Matlab's PID Tuner application.

For the dc motor's identification by means of a machine learning model, a step-type excitation signal with a changing amplitude every 4,798s was used, and the selected model structure was the Elastic Net linear regression model; while, for the identification by means of a transfer function, a step-type excitation signal with constant amplitude was used. The Elastic Net model showed an adjustment of more than 99% over a test set; while the transfer function showed an adjustment of more than 95%.

The PID controllers tuning was carried out by means of three strategies: The first one used genetic algorithms and the integral of time multiplied by the error's absolute value criterion; the second, the first Ziegler-Nichols method; and the third, Matlab's PID Tuner application. The transfer function was used in the last two methods, while the Elastic Net model was used in the first.

The controllers obtained were evaluated on the control system, and it was found that the controller obtained by the Ziegler-Nichols method caused an oscillating response that did not stabilize. For the other methods, both controllers showed to be relatively adequate; however, the controller obtained by means of genetic algorithms and the Elastic Net model showed less oscillation before stabilizing and a shorter establishment time.

An Arduino UNO R3 board was used to implement both the data sampling and the digital controller.

CONTENIDO

AGRADECIMIENTO	ii
DEDICATORIA.....	iii
RESUMEN	iv
ABSTRACT	v
1 CAPÍTULO I: INTRODUCCIÓN.....	1
1.1 Realidad problemática.....	1
1.2 Formulación del problema	4
1.3 Justificación del estudio.....	4
1.3.1 Relevancia Tecnológica.....	4
1.3.2 Relevancia Institucional.....	4
1.3.3 Relevancia Social	4
1.3.4 Relevancia Económica	4
1.3.5 Relevancia Ambiental	4
1.4 Antecedentes.....	5
1.5 Hipótesis	6
1.6 Objetivos	6
1.6.1 General.....	6
1.6.2 Específicos.....	6
1.7 Marco teórico	7
1.7.1 Sistema de control digital	7
1.7.2 Controlador automático	8
1.7.3 Controlador PID discreto.....	10
1.7.4 Filtro paso bajo.....	11
1.7.5 Frecuencia de muestreo.....	12
1.7.6 Modulación por ancho de pulso	15
1.7.7 valor eficaz.....	16

1.7.8	Análisis de la respuesta temporal:	17
1.7.9	Señales de excitación	18
1.7.10	Machine learning	19
1.7.11	Elastic Net	22
1.7.12	Criterio de sintonía de controladores PID	24
1.7.13	Algoritmos genéticos	26
1.7.14	Método de Ziegler – Nichols	29
1.7.15	Arduino	30
1.7.16	The jupyter notebook	32
1.7.17	MatLab	33
1.7.18	Identificación de sistema	36
1.8	Marco conceptual	38
1.8.1	Señal de entrada típica	38
1.8.2	Proceso de muestreo	38
1.8.3	Tiempo de establecimiento	38
1.8.4	Encoder	38
2	CAPÍTULO II: MARCO METODOLÓGICO	39
2.1	Variables	39
2.1.1	Variables independientes	39
2.1.2	Variables dependientes	39
2.2	Metodología	42
2.2.1	Tipo de estudio	42
2.2.2	Diseño	42
2.3	Población y muestra	42
2.3.1	Población	42
2.3.2	Muestra	42
2.4	Método de investigación	42

2.5 Técnicas e instrumentos de recolección de datos	43
2.5.1 Técnicas.....	43
2.5.2 Instrumentos	43
2.5.3 Métodos de análisis de datos.....	43
3 CAPÍTULO III: DESARROLLO Y RESULTADOS	44
3.1 Etapa de adquisición y procesamiento de datos.....	44
3.1.1 Implementación del sistema de control de velocidad	45
3.1.2 Elección de las señales de entrada típica y muestreo de datos.....	45
3.1.3 Identificación de sistema del motor dc y obtención de modelos	52
3.1.4 Decodificación de un cromosoma para obtener parámetros PID, definición del desempeño del controlador digital y simulación de su accionar usando el modelo de machine learning como planta de control	61
3.1.5 Sintonización, en base a algoritmos genéticos, de un controlador PID digital haciendo uso de la simulación realizada	66
3.1.6 Sintonización de un controlador PID digital en base al método de Ziegler-Nichols y otro usando el PID Tuner de MatLab	68
3.2 Etapa de control.....	71
3.2.1 Prueba del desempeño de los controladores en el sistema de control de velocidad	73
4 CAPÍTULO IV: DISCUSIÓN	79
5 CAPÍTULO V: CONCLUSIONES.....	83
6 CAPÍTULO VI: SUGERENCIAS	85
7 CAPÍTULO VII: BIBLIOGRAFÍA	87
8 ANEXOS.....	93

INDICE DE FIGURAS

Figura 1.1: Diagrama de bloques de un sistema de control digital que muestra las señales en forma binaria o gráfica (Botterón 2016).....	7
Figura 1.2: Diagrama de bloques de un sistema de control industrial	8
Figura 1.3: Variación en la ganancia proporcional	14
Figura 1.4: Respuesta del sistema frente a una entrada tipo escalón	15
Figura 1.5: Una señal de onda cuadrada de amplitud acotada	16
Figura 1.6: Señal tipo pulso con periodo T	17
Figura 1.7: Input excitation types in system identification	18
Figura 1.8: Parámetros de diseño de una señal tipo escalón	19
Figura 1.9: Definiendo los componentes de un algoritmo de aprendizaje	19
Figura 1.10: Población, cromosoma y gen.....	27
Figura 1.11: Punto de crossover	28
Figura 1.12: Intercambio de genes entre los padres (Crossover)	28
Figura 1.13: Nueva descendencia	28
Figura 1.14: Antes y después de una mutación	28
Figura 1.15: Respuesta a un escalón unitario de una planta	30
Figura 1.16: Curva de respuesta en forma de S	30
Figura 1.17: Primer método de Z-N	30
Figura 1.18: Arduino IDE 1.8.5 on macOS 10.13.....	31
Figura 1.19: Arduino Uno R3	31
Figura 1.20: Ventana de trabajo de Matlab	33
Figura 1.21: Simulink.	34
Figura 1.22: Herramienta de identificación de sistemas de Matlab.	34
Figura 1.23: PID Tuner.	36
Figura 2.1: Diagrama de flujo del diseño experimental.....	42
Figura 3.1: Esquema de la disposición elementos de software y hardware.....	45
Figura 3.2: Disposición de los elementos de hardware del sistema de control.....	45
Figura 3.3: Ejemplo de una señal de excitación para el sistema (color azul) y su respuesta (color naranja).....	46
Figura 3.4: Señal de entrada para el sistema (color azul) y su velocidad (color naranja) en una ventana de tiempo de 70s.	47
Figura 3.5: Procesos ligados al microprocesador del uC.....	48
Figura 3.6: Procesos ligados al timer del uC.....	49
Figura 3.7: Almacenamiento de datos en memoria.	50
Figura 3.8: Creación de la hoja de cálculo.....	51
Figura 3.9: Vista de los datos guardados en la hoja de cálculo.....	52
Figura 3.10: Parámetros de entrada X y salida Y para entrenar y validar el modelo de machine learning.....	53
Figura 3.11: Vista del conjunto de entrada generado.	54
Figura 3.12: Vista del conjunto de salida generado.....	54
Figura 3.13: Esquema de entrenamiento y validación del modelo.....	55

Figura 3.14: Porcentaje de ajuste sobre el conjunto de prueba.	55
Figura 3.15: Entrada X para usar el modelo y la salida Y que genera.	56
Figura 3.16: Esquema de uso del modelo.	56
Figura 3.17: Diagrama de flujo del proceso de generación de los conjuntos de entrada.	57
Figura 3.18: Diagrama de flujo del proceso de entrenamiento y validación del modelo.	58
Figura 3.19: Vectores en el espacio de trabajo de MatLab.	59
Figura 3.20: Uso de los vectores en la System Identification App.	59
Figura 3.21: Elección de la estructura de modelo.	60
Figura 3.22: Función de transferencia obtenida y exportada en la variable P2DZ.	61
Figura 3.23: Palabras digitales obtenidas a partir de un cromosoma.	63
Figura 3.24: Simulación del desempeño del controlador para un setpoint de 250 rad/s.	64
Figura 3.25: Decodificación del cromosoma y simulación y desempeño del controlador.	64
Figura 3.26: Decodificación del cromosoma y simulación y desempeño del controlador. (continuación).	65
Figura 3.27: Sintonización del controlador mediante algoritmos genéticos.	66
Figura 3.28: Diagramas de bloque en Simulink.	68
Figura 3.29: Simulación de la respuesta de la función de transferencia en lazo abierto frente a una entrada tipo escalón.	68
Figura 3.30: Acercamiento al punto de inflexión para determinar los parámetros T y L requeridos por el primer método de Ziegler-Nichols.	69
Figura 3.31: Parámetros PID obtenidos por el primer método de Ziegler-Nichols.	69
Figura 3.32: Simulación de la respuesta de la función de transferencia en lazo cerrado frente a una entrada tipo escalón.	70
Figura 3.33: Sintonización por medio del PID Tuner App.	70
Figura 3.34: Simulación de la respuesta de la función de transferencia en lazo cerrado frente a una entrada tipo escalón.	71
Figura 3.35: Esquema de la disposición de elementos de software y hardware.	71
Figura 3.36: Procesos ligados al microprocesador del uC.	72
Figura 3.37: Procesos ligados al timer del uC.	73
Figura 3.38: Serial Plotter Arduino. Respuesta del sistema frente a una entrada tipo escalón.	74
Figura 3.39: Serial Plotter Arduino. Respuesta del sistema frente a una entrada tipo escalón con amplitud variable.	74
Figura 3.40: Serial Plotter Arduino. Respuesta del sistema frente a una entrada tipo escalón.	75
Figura 3.41: Serial Plotter Arduino. Respuesta del sistema frente a una entrada tipo escalón con amplitud variable.	75
Figura 3.42: Serial Plotter Arduino. Respuesta del sistema frente a una entrada tipo escalón.	76

Figura 3.43: Serial Plotter Arduino. Respuesta del sistema frente a una entrada tipo escalón con amplitud variable	76
Figura 3.44: Análisis conjunto	77
Figura 3.45: Análisis conjunto (Zoom).....	78

INDICE DE TABLAS

Tabla 2.1: Variables	39
Tabla 3.1: Elementos de software	44
Tabla 3.2: Elementos de hardware	44
Tabla 3.3: Población obtenida después de 40 generaciones	67
Tabla 3.4: Análisis conjunto	78

1 CAPÍTULO I: INTRODUCCIÓN

1.1 Realidad problemática

El control Proporcional-Integral-Derivativo (PID) es el algoritmo de control más comúnmente utilizado en la industria actual (Kocur, Kozak y Dvorscak 2014) y ha sido el método predominante desde el inicio de los sistemas de automatización (Vilanova y Visioli 2012 p. 460). El impacto ha sido tal que más del 90% de todos los circuitos de control son PID. Y se utiliza para una amplia gama de problemas: control de procesos, accionamientos de motores, memorias magnéticas y ópticas, industria automotriz, control de vuelo, instrumentación, etcétera. Por lo que es bastante razonable predecir que el control PID continuará usándose en el futuro. La realimentación ha tenido una influencia revolucionaria en prácticamente todas las áreas donde se ha utilizado y continuará haciéndolo (Åström y Hägglund 2001).

La razón principal de esta popularidad es la facilidad de ajuste y puesta en servicio de controladores PID SISO (Vilanova y Visioli 2012 p. 255). Considerando además que la mayoría de sistemas utilizados son SISO. El PID proporciona un algoritmo óptimo para perturbaciones no medidas y dinámicas desconocidas, el caso común en la manufactura (Vilanova y Visioli 2012 p. 460). A pesar de su popularidad, el foco principal de la investigación en esta área se concentra principalmente en el desarrollo de metodologías de ajuste de los tres parámetros de control PID (Vilanova y Visioli 2012 p. 320). El ajuste mencionado siempre es un tema de gran interés práctico y, a pesar del pequeño número de parámetros ajustables, sigue siendo tema de investigación (Vilanova y Visioli 2012 p. 283).

Aún la mayoría de documentos sobre control tratan el método de Ziegler-Nichols como punto de referencia para sintonizar controladores PID. Esta es una situación muy insatisfactoria porque las reglas de Ziegler-Nichols son conocidas por dar resultados muy pobres en muchos casos (Vilanova y Visioli 2012 p. 416). Es muy fácil demostrar que cualquier controlador con un ajuste razonable superará a un PID con el ajuste por Ziegler-Nichols. Incluso muchas estrategias propuestas pueden eliminarse fácilmente si se comparan con un PID bien ajustado (Åström and Hägglund 2001).

En las últimas dos décadas, los procesos modernos a gran escala han experimentado grandes desafíos y se han vuelto cada vez más complicados en todos los sectores industriales posibles (Yin et al. 2015). El rápido crecimiento de la complejidad de plantas

de procesos modernos, tanto en términos de flujo de materiales como de intercambio de energía, ha incrementado sustancialmente el número de circuitos de control en lazo cerrado para mantener las condiciones de producción y calidad deseadas del producto (Vilanova y Visioli 2012 p. 255), Como resultado, los enfoques tradicionales basados en modelos, que requieren conocer previamente el proceso o modelo físico obtenido a partir de principios primarios, se han vuelto poco prácticos, especialmente para la industria a gran escala (Yin et al. 2015).

Frente a esto, se puede optar por métodos inteligentes para expandir las capacidades de un controlador, y un PID mejorado incrementará la eficiencia (reducir materias primas, servicios públicos y desechos), flexibilidad, operabilidad, mantenimiento, rentabilidad y seguridad, que determina el cumplimiento y la competitividad. El uso innovador de la capacidad de desarrollo y expansión del PID es la clave para la fabricación sostenible (Vilanova y Visioli 2012 p. 460).

En plantas de control con una fuerte no linealidad, se han propuesto métodos de ajuste de parámetros PID que utilizan redes neuronales y algoritmos genéticos, aunque con ciertas limitaciones. Actualmente hay pocos esquemas de ajuste de parámetros PID de uso práctico para sistemas no lineales (Funes et al. 2015). Respecto de estos nuevos métodos, se muestran controladores sintonizados por medio de algoritmos evolutivos que superan de forma amplia los métodos clásicos (Marin Cano, Hernandez Rivero y Jimenez Builes 2018; Colorado Arellano et al. 2018). Así también, se implementaron algoritmos bio-inspirados obteniendo mejores resultados en comparación con los métodos tradicionales. Destacando su efectividad en plantas con retardos y de orden superior (López Mora 2014). En una realidad más cercana, en la escuela de Ingeniería Mecatrónica de la Universidad Nacional de Trujillo, Garcia Pereda (2018) aplicó un algoritmo de colonia de hormigas y en otro trabajo, Balcázar Llanos (2017), un algoritmo genético, para sintonizar controladores PID obteniéndose en ambos casos resultados sobresalientes frente a métodos tradicionales. En todos los casos mencionados anteriormente se necesitó realizar una identificación de la función de transferencia de la planta antes del proceso de sintonización. Sin embargo; la principal limitación de las funciones de transferencia es que solo pueden usarse para diseñar controladores de sistemas lineales o de sistemas no lineales que permiten construir sus aproximaciones lineales alrededor de un punto de equilibrio de interés (Åström y Murray 2006 p. 243).

Por otro lado, debido al crecimiento significativo del grado de automatización en plantas modernas, se generan una gran cantidad de datos de proceso (Yin et al. 2015), que pueden ser aprovechados por los métodos desarrollados en el campo de machine learning y las capacidades computacionales actuales. Por ejemplo, Burgos Molina y Chacón Pacheco (2014) presentan el modelamiento de fenómenos dinámicos en el tiempo a través de una máquina de soporte vectorial. Así también, Bautista Thompson, Guzmán Ramírez y Figueroa Nazuno (2004) utilizan una máquina de soporte vectorial para abordar un problema de predicción de series de tiempo de múltiples puntos con resultados positivos. Así, en otro estudio se describen las importantes ventajas teóricas y prácticas de la regresión de soporte vectorial en la identificación de sistemas de caja negra. La simplicidad de su implementación y su buen rendimiento respecto a la predicción de series temporales, hace de este método una alternativa atractiva frente a las técnicas estándar de identificación de sistemas (Gretton et al. 2002). Drezet y Harrison (1998) mencionan que las propiedades inherentes de las máquinas de soporte vectorial cumplen la mayoría de los requisitos para la identificación de sistemas lineales y no lineales de propósito general. Además, Schölkopf y Smola (2001) mencionan que la estimación regularizada se ha utilizado ampliamente en aplicaciones relacionadas a la estadística y el aprendizaje automático. Pero en los últimos años, se ha explotado en la identificación y predicción de sistemas no lineales. Así, Kahl et al. (2015) realizan la identificación de un modelo de presión de impulso dinámico para usarse en simulaciones, por medio de la técnica de regularización Lasso. De la misma forma, Antonio H. Ribeiro y Luis A. Aguirre (2018) proponen un algoritmo que usa la técnica de regularización Lasso para estimar modelos NARMAX, y fue probado también sobre dos sistemas, uno lineal y el otro no lineal.

Por lo tanto, se presenta un escenario idóneo en el que los procesos de sintonización de controladores usando algoritmos genéticos podrían utilizar modelos de machine learning, como estimadores estadísticos o máquinas de soporte vectorial, que ofrezcan muchas más prestaciones y capacidades al momento de identificar una planta de control.

1.2 Formulación del problema

¿Cómo identificar un motor dc a través de una mejor estructura de modelo que la función de transferencia, para usarse como planta de control durante la sintonización basada en algoritmos genéticos de un controlador PID de velocidad?

1.3 Justificación del estudio

1.3.1 Relevancia Tecnológica

Las capacidades computacionales de la tecnología actual y el avance de métodos de optimización e identificación de sistemas, presentan un escenario ideal para abordar problemáticas entorno al control de procesos, como es la sintonización de controladores PID, desde nuevos enfoques que integren técnicas que se aplicaban comúnmente de forma separada y en otros contextos.

1.3.2 Relevancia Institucional

Resolver problemas de control e identificación de sistemas, son campos en desarrollo, por lo que dar soluciones desde nuevos enfoques como los mencionados en este trabajo, permitirá impulsar la producción científica en la Escuela de Ingeniería Mecatrónica.

1.3.3 Relevancia Social

La integración de técnicas modernas, como las mencionadas en este trabajo, pueden ser un aporte para dar inicio a la exploración de nuevas soluciones a los problemas que los estudiantes o cualquier persona interesada en los sistemas de control, se enfrentan a menudo.

1.3.4 Relevancia Económica

Todas las herramientas utilizadas en este trabajo son de uso y distribución libre, por lo que este trabajo permite la apertura a la colaboración y customización que las herramientas como el software privativo no admiten.

1.3.5 Relevancia Ambiental

El desarrollo de mejores controladores implica optimizar los procesos de control, lo cual, a su vez, está ligado a un ahorro de recursos en las acciones industriales, científicas o didácticas, contribuyendo a la preservación del ambiente.

1.4 Antecedentes

En relación con métodos de simulación y optimización, se presenta el uso parcial de un modelo de caja negra, demostrando ser una herramienta eficiente para el control, incluso, de algunos sistemas no lineales. Sin embargo, se señala que para el uso exitoso de esta herramienta se requiere al menos una comprensión cualitativa del sistema en consideración (Strmčnik et al. 1987).

Wu (1999) propone el uso de una red neuronal para tomar el lugar de la planta de control en el procedimiento de ajuste de su controlador PID usando algoritmos genéticos. De esta manera, fuera de línea, las ganancias PID se determinan y posteriormente se aplican a la planta para el control en línea. Luego, manteniendo la planta en línea, se realiza un nuevo ajuste de los parámetros PID aplicando algoritmos genéticos. Mostrando resultados alentadores, pero señalando la necesidad de posteriores esfuerzos en el desarrollo de estos conceptos.

Fliess, Join y Sira-Ramirez (2006) muestran un enfoque de control predictivo basado en modelos de caja negra, indicando su potencial uso en plantas industriales donde es difícil lograr una identificación paramétrica y que aún se debe seguir investigando para hacer controladores universales que sean más precisos.

Saad, Jamaluddin y Darus (2012) presentan la implementación de un sintonizador de controlador PID usando técnicas de algoritmos genéticos y evolución diferencial, cuyo desempeño es evaluado usando los criterios de la integral del error absoluto y el error cuadrático medio. Además, comparan el desempeño del controlador PID sintonizado, frente al obtenido mediante el método de Ziegler-Nichols, obteniendo resultados positivos.

En otro trabajo, se presenta un método para diseñar controladores PID usando redes neuronales modulares LMN y algoritmos evolutivos, aunque indicando la limitación que supone el cálculo computacional de la computadora utilizada (Hametner et al. 2013).

Hissem, Doumbia y Keddar (2018) presentan un nuevo método de sistema de caja negra para aproximar sistemas de alto orden y diseñar controladores, sin necesidad del uso de funciones de transferencia, y el diseño se basa únicamente en elementos ponderados que caracterizan la respuesta de salida de la planta. Desarrollaron y aplicaron un controlador

PI al sistema real. Se investigó el rendimiento de este nuevo enfoque mediante el criterio de la integral del error cuadrático (ISE) y el método de aproximación de Pade, mostrando resultados positivos.

1.5 Hipótesis

Será posible la identificación no paramétrica de un motor dc por medio de un modelo de machine learning, para usarse como planta de control en la sintonización basada en algoritmos genéticos de un controlador PID digital de velocidad.

1.6 Objetivos

1.6.1 General

Identificar de forma no paramétrica un motor dc por medio de un modelo de machine learning, para ser la planta de control en un proceso de sintonización, basada en algoritmos genéticos, de un controlador PID digital de velocidad.

1.6.2 Específicos

- Implementar el sistema de control de velocidad.
- Elegir las señales de entrada típica para el sistema de control y muestreo de datos.
- Realizar la identificación de sistema del motor dc para obtener dos modelos: una función de transferencia y un modelo de machine learning.
- Decodificación de un cromosoma para obtener parámetros PID, definición del desempeño del controlador digital y simulación de su accionar usando el modelo de machine learning como planta de control
- Sintonizar, en base a algoritmos genéticos, un controlador PID digital haciendo uso de la simulación realizada.
- Sintonizar un controlador PID digital en base al método de Ziegler-Nichols y otro usando el PID Tuner de MatLab.
- Probar el desempeño de los controladores en el sistema de control de velocidad.

1.7 Marco teórico

1.7.1 Sistema de control digital

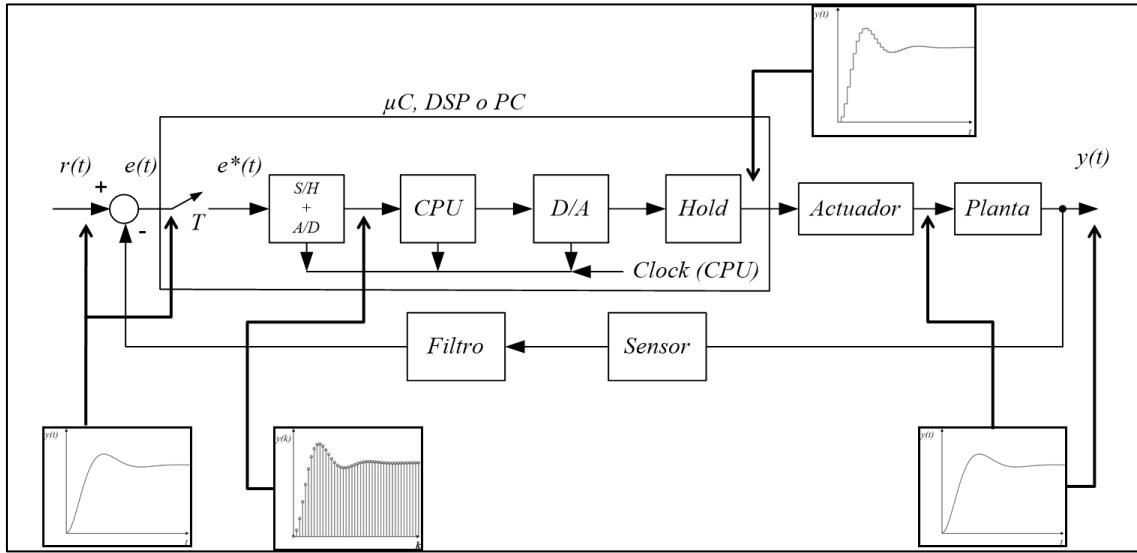


Figura 1.1: Diagrama de bloques de un sistema de control digital que muestra las señales en forma binaria o gráfica (Botterón 2016).

La salida de la planta es una señal en tiempo continuo. La señal de error se convierte a forma digital mediante el circuito de muestreo y retención y el convertidor analógico-digital. La conversión se hace en el tiempo de muestreo. La computadora digital procesa las secuencias de números por medio de un algoritmo y produce nuevas secuencias de números. En cada instante de muestreo se debe convertir un número codificado (en general un número binario que consiste en ocho o más dígitos binarios) en una señal física de control, la cual normalmente es una señal en tiempo continuo o analógica. El convertidor digital-analógico y el circuito de retención convierten la secuencia de números en código numérico a una señal continua por secciones. El reloj en tiempo real de la computadora sincroniza los eventos. La salida del circuito de retención, una señal en tiempo continuo, se alimenta a la planta, ya sea de manera directa o a través de un actuador, para controlar su dinámica.

La operación que transforma las señales en tiempo continuo en datos en tiempo discreto se denomina muestreo o discretización. La operación inversa, que transforma datos en tiempo discreto en una señal en tiempo continuo, se conoce como retención de datos; ésta realiza la reconstrucción de la señal en tiempo continuo a partir de la secuencia de datos en tiempo discreto. Esto por lo regular se logra al utilizar alguna de las muchas técnicas

de extrapolación. En la mayoría de los casos esto se realiza manteniendo constante la señal entre los instantes de muestreo sucesivos.

El circuito de muestreo y retención (S/H, del inglés Sample-and-Hold) y el convertidor analógico-digital (A/D) convierten la señal en tiempo continuo en una secuencia de palabras binarias codificadas numéricamente. Dicho proceso de conversión A/D se conoce como codificación. La combinación del circuito S/H y el convertidor analógico-digital se puede visualizar como un interruptor que cierra instantáneamente en cada intervalo de tiempo T y genera una secuencia de números en código numérico. La computadora digital procesa dichos números en código numérico y genera una secuencia deseada de números en código numérico. El proceso de conversión digital-analógico (D/A) se denomina decodificación (Ogata 1995).

1.7.2 Controlador automático

Un controlador automático compara el valor real de la salida de una planta con la entrada de referencia (el valor deseado), determina la desviación y produce una señal de control que reduce la desviación a cero o a un valor pequeño. La manera en la cual el controlador automático produce la señal de control se denomina acción de control. La Figura 1.2 es un diagrama de bloques de un sistema de control industrial que consiste en un controlador automático, un actuador, una planta y un sensor (elemento de medición).

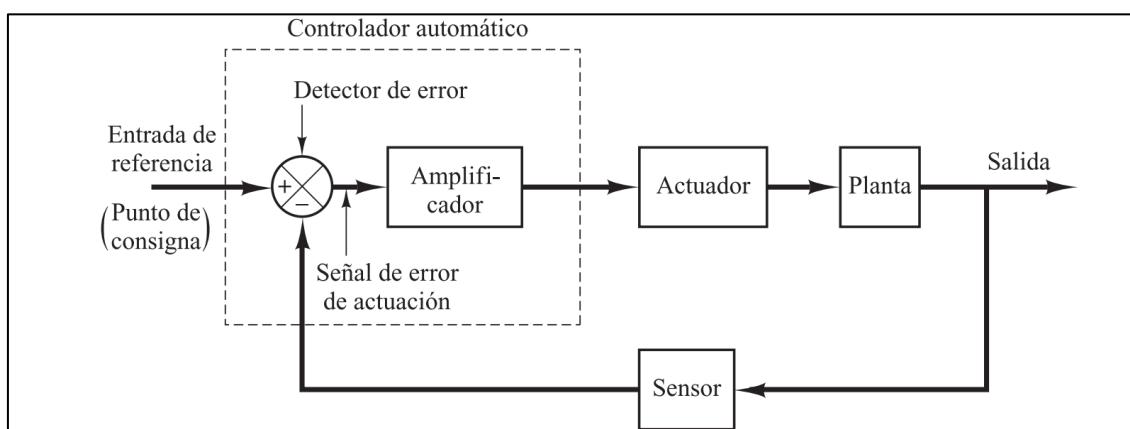


Figura 1.2: Diagrama de bloques de un sistema de control industrial, formado por un controlador automático, un actuador, una planta y un sensor (Ogata 2010).

El controlador detecta la señal de error que, por lo general, está en un nivel de potencia muy bajo, y la amplifica a un nivel lo suficientemente alto. La salida de un controlador automático alimenta a un actuador, como un motor o una válvula neumática, un motor

hidráulico o un motor eléctrico (El actuador es un dispositivo de potencia que produce la entrada para la planta de acuerdo con la señal de control, a fin de que la señal de salida se aproxime a la señal de entrada de referencia). El sensor, o elemento de medición, es un dispositivo que convierte la variable de salida en otra variable manejable, como un desplazamiento, una presión o un voltaje, que pueda usarse para comparar la salida con la señal de entrada de referencia. Este elemento está en la trayectoria de realimentación del sistema en lazo cerrado. El punto de ajuste del controlador debe convertirse en una entrada de referencia con las mismas unidades que la señal de realimentación del sensor o del elemento de medición.

Acción de control proporcional

Para un controlador con acción de control proporcional, la relación entre la salida del controlador $u(t)$ y la señal de error $e(t)$ es:

$$u(t) = k_p e(t)$$

Donde k_p se considera la ganancia proporcional. Cualquiera que sea el mecanismo real y la forma de la potencia de operación, el controlador proporcional es, en esencia, un amplificador con una ganancia ajustable.

Acción de control integral

En un controlador con acción de control integral, el valor de la salida del controlador $u(t)$ se cambia a una razón proporcional a la señal de error $e(t)$. Es decir,

$$u(t) = k_i \int_0^t de(t)dt$$

donde k_i es una constante ajustable.

Acción de control proporcional-integral

La acción de control de un controlador proporcional-integral (PI) mejora la estabilidad relativa y el error en estado estable al mismo tiempo, pero el tiempo de levantamiento se incrementa. Se define mediante

$$u(t) = k_p e(t) + k_i \int_0^t de(t)dt$$

Acción de control proporcional – derivativa:

La acción de control de un controlador proporcional-derivativa (PD) es en esencia anticipativa, mejora el amortiguamiento y tendrá efecto en el error de estado estable sólo si el error varía con respecto al tiempo. Se define mediante:

$$u(t) = k_p e(t) + k_d \frac{de(t)}{dt}$$

donde k_d es el tiempo derivativo.

Acción de control proporcional-integral-derivativa

La combinación de la acción de control proporcional, la acción de control integral y la acción de control derivativa se denomina acción de control proporcional-integral-derivativa. Esta acción combinada tiene las ventajas de las acciones de control PI y PD.

La ecuación de un controlador con esta acción combinada está dada por:

$$u(t) = k_p e(t) + k_i \int_0^t de(t)dt + k_d \frac{de(t)}{dt}$$

donde K_p es la ganancia proporcional, k_i es el tiempo integral y k_d es el tiempo derivativo. (Ogata 2010)

1.7.3 Controlador PID discreto

El controlador PID en el dominio en tiempo continuo se describe como:

$$G_c(s) = k_p + k_d s + \frac{k_i}{s}$$

El componente proporcional k_p se implementa en forma digital mediante una ganancia constante k_p . Ya que un computadora digital o procesador tiene una longitud de palabra finita, la constante k_p no puede realizarse con resolución infinita.

La derivada con respecto al tiempo de una función $f(t)$ en $t = kT$ se puede aproximar mediante la regla de diferencia hacia atrás, empleando los valores de $f(t)$ medidos en $t = kT$ y $t = (k - 1)T$; esto es:

$$\frac{df(t)}{dt} \Big|_{t=T} = \frac{1}{T} (f(kT) - f[(k-1)T])$$

La función de transferencia z del diferenciador digital es:

$$G_D(z) = k_d \frac{z - 1}{Tz}$$

Donde k_d es la constante proporcional de la derivada del controlador. Y en general, la selección del periodo de muestreo es muy importante, el valor de T debe ser lo suficientemente pequeño, para que la aproximación digital sea exacta.

La regla de integración trapezoidal, aproxima el área bajo la función $f(t)$ mediante una serie de trapezoides. Sea la integral de $f(t)$ evaluado en $t = kT$ designado como $u(kT)$. Entonces:

$$u(kT) = u[(k-1)T] + \frac{T}{2}(f(kT) - f[(k-1)T])$$

La función de transferencia z del integrador digital usando la regla trapezoidal es:

$$G_I(z) = k_I \frac{U(z)}{f(z)} = k_I \frac{Tz}{z-1}$$

Donde k_I es la constante proporcional de la derivada del controlador.

Al combinar la operación proporcional, integral (Método trapezoidal) y derivativa descritas anteriormente, el controlador digital PID es modelado mediante la siguiente función de transferencia:

$$G_I(z) = \frac{(k_p + Tk_i/2 + k_d/T)z^2 + (Tk_i/2 - k_p - 2k_d/T)z + k_d/T}{z(z-1)}$$

Una vez que la función de transferencia de un controlador digital se determina, el controlador puede implementarse mediante un procesador digital o computadora. El operador z^{-1} se interpreta como un tiempo de retardo de T segundos. En la práctica, el tiempo de retardo se implementa mediante el almacenamiento de una variable en alguna localidad de memoria en la computadora y entonces se saca después de que han pasado T segundos. (Kuo y Aranda Pérez 1996)

1.7.4 Filtro paso bajo

La forma general en el dominio de Laplace de un filtro paso bajo de primer orden es:

$$F(s) = \frac{\omega_c}{s + \omega_c} = \frac{1}{\frac{s}{\omega_c} + 1}$$

$$\omega_c = 2\pi f_c$$

Donde f_c es la frecuencia de corte en Hz y ω_c es la frecuencia de corte en rad .

Para implementar el filtro en un microcontrolador, se debe trabajar con la convolución del bloque del filtro, con un bloque de muestreo y retención de orden cero para evitar señales atrasadas:

$$H_{0(S)}F_{(S)} = \left(\frac{1 - e^{-T_s S}}{S}\right) \left(\frac{\omega_c}{S + \omega_c}\right)$$

Para poder utilizar el filtro en el programa, es necesario cambiar del dominio de Laplace al dominio Zeta:

$$H_{0(Z)}F_{(Z)} = \frac{Z^{-1}(1 - e^{-\omega_c T_s})}{1 - Z^{-1}e^{-\omega_c T_s}} = \frac{Y_{(Z)}}{U_{(Z)}}$$

Escribiendo la ecuación de la señal en tiempo discreto queda:

$$y_{(k)} = u_{(k-1)}(1 - e^{-\omega_c T_s}) + y_{(k-1)}e^{-\omega_c T_s}$$

Donde T_s es el tiempo de muestreo en segundos.

1.7.5 Frecuencia de muestreo

La frecuencia de muestreo es el número de muestras por unidad de tiempo que se toman de una señal en tiempo continuo para producir una secuencia de valores en tiempo discreto. Se expresa en Hercios (Hz).

La frecuencia de muestreo debe cumplir con el teorema de Nyquist – Shannon, que demuestra que la reconstrucción exacta de una señal periódica continua en banda base a partir de sus muestras, es matemáticamente posible si la señal está limitada en banda y la tasa de muestreo es superior al doble de su ancho de banda.

Si la frecuencia más alta contenida en una señal analógica $x_a(t)$ es $F_{max} = B$ y la señal se muestrea a $F_s > 2F_{max} = B$, entonces $x_a(t)$ se puede recuperar totalmente a partir de sus muestras mediante la siguiente función de interpolación:

$$g(t) = \frac{\sin 2\pi Bt}{2\pi Bt}$$

Así, $x_a(t)$ se puede expresar como:

$$x_a(t) = \sum_{n=-\infty}^{\infty} x_a\left(\frac{n}{F_s}\right) g\left(t - \frac{n}{F_s}\right)$$

Donde $x_a\left(\frac{n}{F_s}\right) = x_a(nT) = x(n)$ son las muestras de $x_a(t)$

La frecuencia $2B$ es llamada la razón de muestreo de Nyquist. La mitad de su valor, B , es llamada algunas veces la frecuencia de Nyquist. Si el criterio no es satisfecho, existirán frecuencias cuyo muestreo coincide con otras (el llamado aliasing).

No aporta nada incrementar la tasa de muestreo una vez que ésta cumple el criterio de Nyquist. En la práctica y dado que no existen los filtros analógicos pasa-bajo ideales, se debe dejar un margen entre la frecuencia máxima que se desea registrar y la frecuencia de Nyquist (frecuencia crítica) que resulta de la tasa de muestreo elegida (por ejemplo, para CD-Audio las frecuencias de los componentes a registrar y reproducir oscilan entre 20Hz y 20 kHz y la frecuencia crítica que se elige es de 22,05 kHz; puesto que la razón de muestreo de Nyquist debe ser mayor a $2 \times 20\text{kHz} = 40\text{ kHz}$, y se emplea la tasa de 441000 muestras por segundo, es decir 44.1 kHz; un margen del 10% aproximadamente para esta aplicación). Pero este margen es una necesidad que resulta de las limitaciones físicas de un filtro de reconstrucción (o filtro antialiasing) real, y no una consideración que contemple (o deba contemplar) el teorema (Lavry 2004).

Por otro lado, el periodo de muestreo $T_s = \frac{1}{F_s}$ y por lo tanto, la frecuencia de muestreo; debe tomar ciertas consideraciones adicionales para la implementación de un controlador PID digital:

La respuesta en lazo cerrado de un sistema controlado por un PID digital va a depender de este período de muestreo. Si este tiempo es demasiado alto, la estabilidad del sistema será menor y el sistema puede llegar a hacerse inestable y no ser controlable. Un método para estimar el período de muestreo consiste en calcular el período de oscilación del sistema en lazo cerrado con una ganancia que provoque oscilaciones. Se tomará el período de muestreo como la décima parte del tiempo o período de oscilación.

En el ejemplo que aparece a continuación se ha aumentado la ganancia proporcional hasta que se mantengan las oscilaciones en la respuesta al escalón. El período de oscilación es entonces de 5.6 segundos y por lo tanto el período de muestreo debe ser menor de 0.56 segundos.

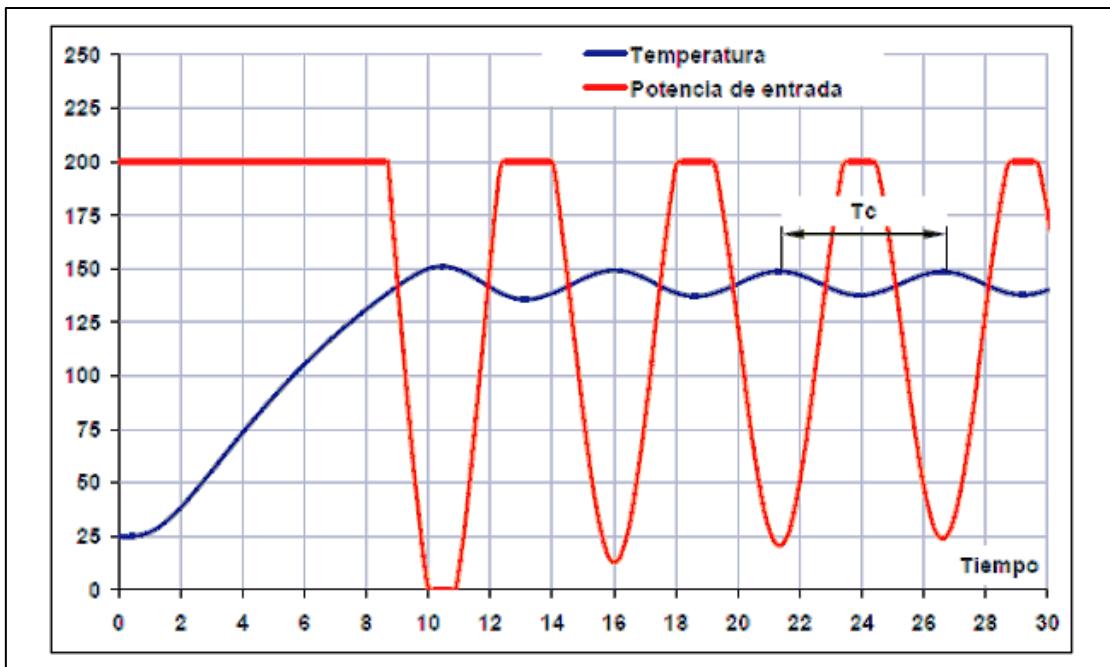


Figura 1.3: Variación en la ganancia proporcional (Pardo 2018).

$$T_c = 26.8 - 21.2 = 5.6 \text{ segundos}$$

$$T < \frac{T_c}{10} = 0.56 \text{ segundos (Período de muestreo)}$$

Si el sistema es sobre amortiguado y no presenta oscilaciones, el criterio para escoger el tiempo de muestreo partirá de la respuesta al escalón. Como regla general se acepta que T debe ser 10 veces menor que el tiempo de subida del sistema ante un escalón en lazo abierto. Este tiempo de subida se puede calcular como el tiempo que tarda el sistema en subir desde un 10% hasta un 90% del valor final. Por ejemplo, un sistema térmico que exhiba la respuesta al escalón que se muestra a continuación:

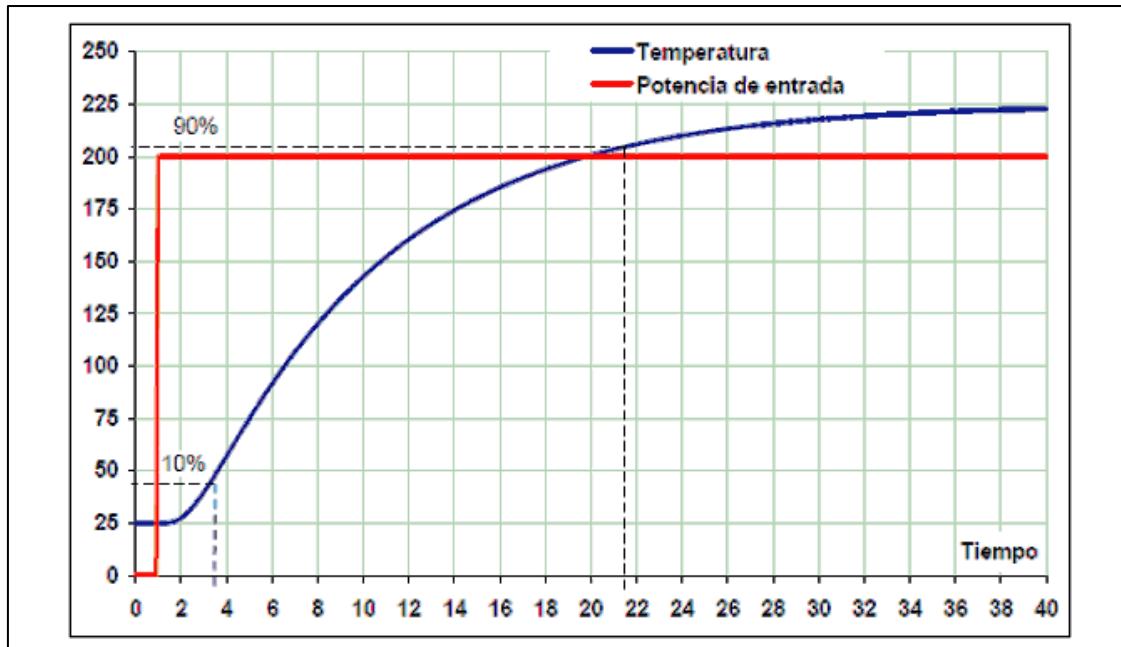


Figura 1.4: Respuesta del sistema frente a una entrada tipo escalón (Pardo 2018).

Este sistema tarda en subir desde el 10% hasta el 90% del valor final $21.5 - 3.5 = 18$ segundos. Por lo tanto, para este sistema de ejemplo el tiempo de muestreo del controlador PID debe ser como máximo una décima parte de los 18 segundos:

$$T < \frac{\text{Tiempo de respuesta}}{10}$$

$$T < \frac{18}{10} = 1.8 \text{ segundos}$$

En los dos casos se ha utilizado la misma planta para calcular el tiempo de muestreo. Como puede verse los resultados son muy diferentes. Con el segundo método el tiempo de muestreo es tres veces mayor que con el primero. Por lo tanto, el tiempo de muestreo depende también de la respuesta que se vaya a conseguir y del tipo de sistema. Siempre que se pueda utilizar el primer método, será preferible puesto que calcula tiempos menores y por lo tanto más seguros. (Pardo 2018)

1.7.6 Modulación por ancho de pulso

La modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de pulse width modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal o una cuadrada, por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o

para controlar la cantidad de energía que se envía a una carga. El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período (Greer 2014).

Expresado matemáticamente:

$$D = \frac{\tau}{T}$$

D es el ciclo de trabajo

τ es el tiempo en que la función es positiva (ancho del pulso)

T es el período de la función

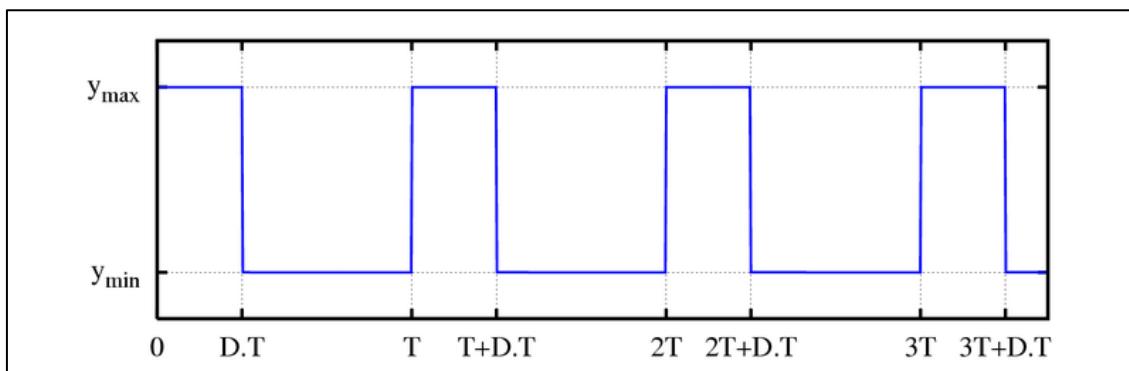


Figura 1.5: Una señal de onda cuadrada de amplitud acotada ($y_{\min}; y_{\max}$) mostrando el ciclo de trabajo D (Buttay 2006).

1.7.7 valor eficaz

Se denomina valor eficaz al valor cuadrático medio de una magnitud eléctrica. El concepto de valor eficaz se utiliza especialmente para estudiar las formas de onda periódicas, a pesar de ser aplicable a todas las formas de onda, constantes o no. En ocasiones se denomina con el extranjerismo RMS (del inglés, root mean square).

El valor eficaz de la tensión es:

$$V_{ef} = \sqrt{\frac{1}{T} \int_{t_0}^{t_0+T} v^2(t) dt}$$

El significado físico del valor eficaz es designar el valor de una corriente rigurosamente constante que al circular sobre una determinada resistencia óhmica produciría los mismos efectos caloríficos que dicha corriente variable. De este modo, se establece un paralelismo

entre cualquier tipo de corriente variable y la corriente continua que simplifica los cálculos con esta última (Alcalde 2014).

Para un tren de pulsos (Una señal PWM típica):

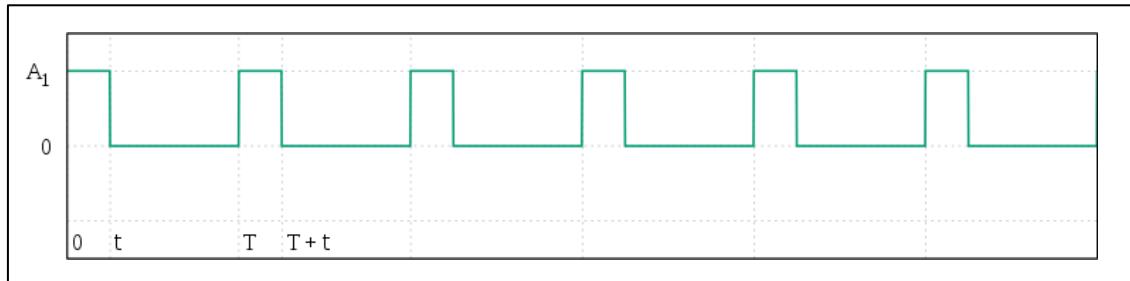


Figura 1.6: Señal tipo pulso con periodo T (MrLejinad 2016).

Cuya fórmula es:

$$y = \begin{cases} A_1 & \text{frac}(ft) < D \\ 0 & \text{frac}(ft) > D \end{cases}$$

Su valor eficaz es:

$$A_1\sqrt{D}$$

Donde D , está expresada en "por unidad" del periodo T . ($D = \frac{t}{T}$).

1.7.8 Análisis de la respuesta temporal:

El análisis de respuesta temporal estudia la respuesta (De un sistema), es decir, el resultado como una función del tiempo.

La respuesta de tiempo total $c(t)$ de un sistema de control consiste en respuesta transitoria (respuesta dinámica $c_t(t)$) y respuesta en estado estable $c_{ss}(t)$.

$$c(t) = c_t(t) + c_{ss}(t)$$

Donde:

$c(t)$: respuesta de tiempo total

$c_t(t)$: respuesta transitoria

$c_{ss}(t)$: respuesta de estado estacionario.

Antes de proceder con el análisis de la respuesta temporal de un sistema de control, es necesario probar la estabilidad del sistema a través de pruebas indirectas. La naturaleza de la respuesta transitoria de un sistema es dependiente del sistema en sí mismo y no del

tipo de entrada. Por lo tanto, se debe analizar la respuesta transitoria con alguna señal de excitación. La señal tipo escalón es la más usada para este propósito. (Bhattacharya 2009)

1.7.9 Señales de excitación

Como perturbaciones se pueden utilizar muchos tipos diferentes de señales, pero las de tipo escalón, PRBS (pseudo random binary sequence) y multisenos son las preferidas.

Además, las señales de excitación pueden dividirse ampliamente en tres categorías principales, llamadas señales de propósito general, señales de prueba optimizadas y señales dedicadas avanzadas, como se ilustra en la siguiente figura.

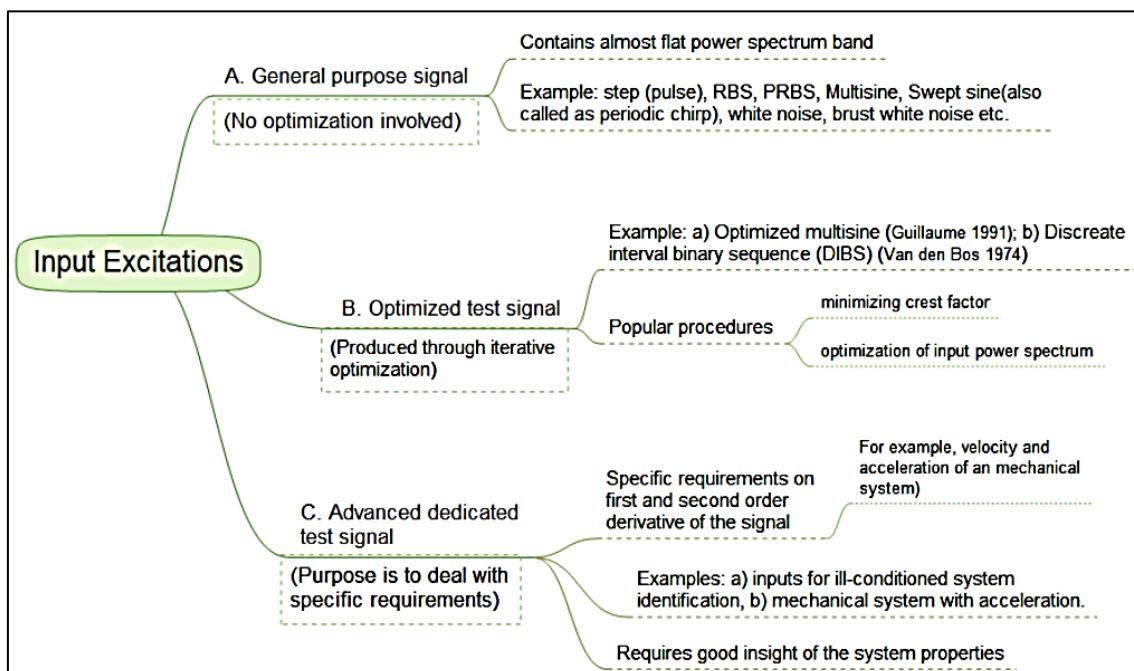


Figura 1.7: Input excitation types in system identification (Ramkrishna 2016).

Señal escalón:

Una señal tipo escalón está definida por:

$$u(t) = \begin{cases} 0 & t < 0 \\ u_0 & t \geq 0 \end{cases}$$

Donde u_0 es la amplitud de la señal. Debido a su simplicidad es usada extensivamente en la práctica. A menudo, las señales tipo escalón se usan con fines de pre identificación antes de aplicar señales avanzadas para captar la información básica del sistema, como la ganancia estática, las constantes de tiempo, las características de salida, etc. Típicamente, esta información del sistema es considerablemente útil para el diseño de experimentos avanzados (Ramkrishna 2016).

Los parámetros de diseño de una señal tipo escalón son la amplitud (A) de la señal y la duración (ΔT) del escalón.

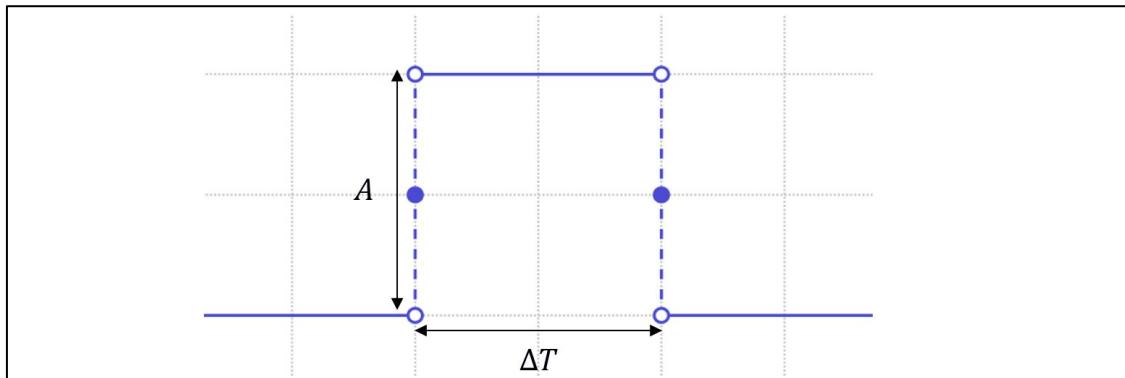


Figura 1.8: Parámetros de diseño de una señal tipo escalón (Ramkrishna 2016).

1.7.10 Machine learning

La idea de Machine Learning es que habrá algún algoritmo de aprendizaje que ayudará a una máquina a aprender a partir de unos datos.

El profesor Mitchell lo definió de la siguiente manera: "Se dice que un programa de computadora aprende de la experiencia E con respecto a algunas clases de tareas T y una medida de rendimiento P. Si su desempeño en las tareas T, medido por P, mejora con la experiencia E." Estos tres parámetros T, P y E son los componentes principales de cualquier algoritmo de aprendizaje.

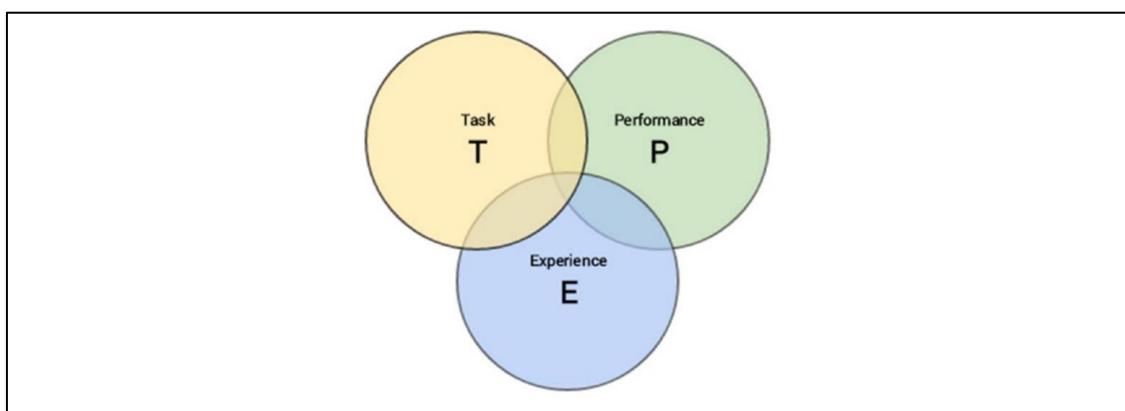


Figura 1.9: Definiendo los componentes de un algoritmo de aprendizaje (Sarkar y BaliSharma 2018).

Se puede simplificar la definición como sigue. El machine learning es un campo que consiste de algoritmos de aprendizaje que:

- ✓ Mejoran su desempeño P
- ✓ Al ejecutar alguna tarea T
- ✓ Durante el tiempo sometido a una experiencia E

Definición de la tarea, T:

Desde el punto de vista del problema, la tarea, T, es básicamente el problema del mundo real que se debe resolver, que puede ser desde encontrar el mejor marketing o combinación de productos hasta predecir fallas en la infraestructura.

En el mundo del Machine learning, es mejor si puede definir la tarea de la forma más concreta posible, de modo que hable sobre cuál es el problema exacto que planea resolver y cómo puede definir o formular el problema en una tarea específica de Machine learning. Volviendo a las tareas típicas que podrían clasificarse como tareas de Machine learning, la siguiente lista describe algunas tareas populares:

- Clasificación o categorización
- Regresión
- Detección de anomalías
- Anotación estructurada
- Traducción
- Agrupación o agrupamiento
- Transcripciones

Definiendo la Experiencia, E

El proceso de consumir un conjunto de datos que consiste en muestras de datos o puntos de datos tales que un algoritmo o modelo de aprendizaje aprende patrones inherentes se define como la experiencia, E que se gana con el algoritmo de aprendizaje. Cualquier experiencia que obtenga el algoritmo proviene de muestras de datos o puntos de datos y esto puede ocurrir en cualquier punto del tiempo. Puede alimentar las muestras de datos de una sola vez utilizando datos históricos o incluso proporcionar nuevas muestras de datos cada vez que se adquieren. Por lo tanto, la idea de que un modelo o algoritmo gane experiencia generalmente ocurre como un proceso iterativo, también conocido como entrenamiento del modelo. Cuando una máquina realmente aprende, se basa en datos que se le suministran de vez en cuando, lo que le permite adquirir experiencia y conocimiento

sobre la tarea a resolver, tales como que puede usar esta experiencia, E, para predecir o resolver la misma tarea, T, en el futuro para puntos de datos no vistos anteriormente.

El aprendizaje automático se trata de aprender algunas propiedades de un conjunto de datos y aplicarlos a nuevos datos. Esta es la razón por la cual una práctica común en el machine learning para evaluar un algoritmo es dividir los datos disponibles en dos conjuntos, uno que llamamos **conjunto de entrenamiento** en el que aprendemos propiedades de datos y otro que llamamos el **conjunto de pruebas** en el que probamos estas propiedades.

Definiendo el rendimiento, P

El rendimiento, P, suele ser una medida cuantitativa o métrica que se utiliza para ver qué tan bien el algoritmo o modelo está realizando la tarea, T, con experiencia, E.

Las medidas de rendimiento típicas incluyen exactitud, precisión, recuperación, puntaje F1, sensibilidad, especificidad, tasa de error, tasa de clasificación errónea y muchas más.

Las medidas de rendimiento generalmente se evalúan en muestras de datos de entrenamiento (usadas por el algoritmo para ganar experiencia, E) así como en muestras de datos que no se han visto o aprendido anteriormente, que generalmente se conocen como muestras de datos de prueba y validación. La idea detrás de esto es generalizar el algoritmo para que no se vuelva demasiado parcial solo en los puntos de datos de entrenamiento y tenga un buen rendimiento en el futuro en puntos de datos más nuevos. Se hablará más sobre capacitación, validación y datos de prueba cuando hablemos sobre la construcción y validación de modelos. (Sarkar y BaliSharma 2018)

En general, un problema de aprendizaje considera un conjunto de n muestras de datos y luego trata de predecir las propiedades de datos desconocidos. Si cada muestra es más que un número único y, por ejemplo, una entrada multidimensional (también conocida como datos multivariantes), se dice que tiene varios atributos o características (features). Podemos separar los problemas de aprendizaje de acuerdo a los tipos de algoritmos que, en función de su salida, se utilizan como estrategia de resolución:

Aprendizaje supervisado, en el cual los datos vienen con atributos adicionales que queremos predecir. Este problema puede ser:

- **claseficación**: las muestras pertenecen a dos o más clases y queremos aprender de los datos ya etiquetados cómo predecir la clase de los datos sin etiqueta.

- **regresión:** si el resultado deseado consiste en una o más variables continuas, entonces la tarea se llama regresión. Un ejemplo de un problema de regresión sería la predicción de la longitud de un salmón en función de su edad y peso. En estadística, el análisis de la regresión es un proceso estadístico para estimar las relaciones entre variables. Incluye muchas técnicas para el modelado y análisis de diversas variables, cuando la atención se centra en la relación entre una variable dependiente y una o más variables independientes (o predictoras). Más específicamente, el análisis de regresión ayuda a entender cómo el valor de la variable dependiente varía al cambiar el valor de una de las variables independientes, manteniendo el valor de las otras variables independientes fijas.

Aprendizaje no supervisado, en el cual los datos de entrenamiento consisten en un conjunto de vectores de entrada x sin ningún valor objetivo correspondiente. El objetivo en tales problemas puede ser descubrir grupos de ejemplos similares dentro de los datos, donde se llama agrupamiento, o determinar la distribución de datos dentro del espacio de entrada, conocida como estimación de densidad, o proyectar los datos desde una dimensión alta. espacio hasta dos o tres dimensiones para visualización.

El proceso de aprendizaje tiene como resultado (o salida) un modelo para resolver una tarea dada. Un tipo es el modelo de regresión lineal llamado modelo lineal generalizado en el que se espera que el valor objetivo sea una combinación lineal de las variables de entrada. En noción matemática, si \hat{y} es el valor predicho:

$$\hat{y}(\omega, x) = \omega_0 + \omega_1 x_1 + \cdots + \omega_p x_p$$

A través del módulo, designamos el vector $\omega = (\omega_0, \dots, \omega_p)$ como coeficientes y ω_0 como el término independiente (Pedregosa et al. 2011).

1.7.11 Elastic Net

Elastic net es un método de regresión regularizada que puede usarse como un modelo lineal generalizado (modelo de regresión lineal) en ciertas bibliotecas de lenguajes de programación como python. Es recomendable usar Elastic net cuando se tiene varias variables altamente correlacionadas. Elastic net es un híbrido de regresión Ridge y regularización Lasso. Al igual que Lasso, Elastic net puede generar modelos reducidos mediante la generación de coeficientes de valor cero. Estudios empíricos han sugerido

que Elastic net puede superar a Lasso en datos con predictores altamente correlacionados. Lasso proporciona la regularización de Elastic net cuando establece el valor de Alpha a un número estrictamente entre 0 y 1.

Detalles de Lasso y Elastic net

Lasso es una técnica de regularización para realizar regresiones lineales. Lasso incluye un término de penalización que restringe el tamaño de los coeficientes estimados. Por lo tanto, se parece a la regresión Ridge. Lasso es un estimador que genera estimaciones de coeficientes que están sesgados para ser pequeños. Sin embargo, un estimador Lasso puede tener un error cuadrático medio menor que un estimador ordinario de mínimos cuadrados cuando se lo aplica a datos nuevos.

Usar Lasso para:

- ✓ Reducir el número de predictores en un modelo de regresión.
- ✓ Identificar predictores importantes.
- ✓ Seleccionar entre los predictores redundantes.
- ✓ Producir estimaciones de contracción con errores predictivos potencialmente más bajos que los mínimos cuadrados ordinarios.

A diferencia de la regresión Ridge, a medida que aumenta el término de penalización, Lasso establece más coeficientes a cero. Esto significa que el estimador Lasso es un modelo más pequeño, con menos predictores. Como tal, Lasso es una alternativa a la regresión por pasos y otras técnicas de selección de modelo y reducción de dimensiones.

Definición de Lasso

Para un valor dado de λ , un parámetro no negativo, Lasso soluciona el problema:

$$\min_{\beta_0 \beta} \left(\frac{1}{2N} \sum_{i=1}^N (y_i - \beta_0 - x_i^T \beta)^2 + \lambda \sum_{j=1}^p |\beta_j| \right)$$

Donde:

N es el número de observaciones.

y_i es la respuesta en la observación i .

x_i es data, un vector de valores p en la observación i .

λ es un parámetro de regularización positivo que corresponde a un valor de Lambda.

Los parámetros β_0 y β son un escalar y un vector- p -dimensional respectivamente.

A medida que λ aumenta, disminuye el número de componentes β distintos de cero.

Lasso involucra la norma L1 de β , en contraste con Elastic Net (Tibshirani 1996).

Siendo:

$$L_1 = \lambda \sum_{j=1}^p |\beta_j|$$

Definición de Elastic Net

Para un α estrictamente entre 0 y 1, y un λ no negativo, Elastic net resuelve el problema:

$$\min_{\beta_0 \beta} \left(\frac{1}{2N} \sum_{i=1}^N (y_i - \beta_0 - x_i^T \beta)^2 + \lambda P_\alpha(\beta) \right)$$

Donde

$$P_\alpha(\beta) = \frac{(1-\alpha)}{2} L_2 + \alpha L_1 = \frac{(1-\alpha)}{2} \|\beta\|_2^2 + \alpha \|\beta\|_1 = \sum_{J=1}^P \left(\frac{(1-\alpha)}{2} \beta_J^2 + \alpha |\beta_J| \right)$$

Elastic net es lo mismo que Lasso cuando $\alpha = 1$. A medida que α se contrae hacia 0, Elastic net se aproxima a la regresión Ridge. Para otros valores de α , el término de penalización $P_\alpha(\beta)$ interpola entre la norma L_1 de β y la norma L_2 cuadrada de β (Zou y Hastie 2005).

A finales de 2014, se demostró que Elastic Net puede reducirse a una máquina de soporte vectorial lineal (Zhou et al. 2015). Una reducción similar se probó previamente para LASSO en 2014 (Jaggi 2014). Los autores demostraron que para cada instancia de Elastic Net, un problema de clasificación binaria artificial puede construirse de tal manera que la solución de hiperplano de una máquina de soporte vectorial lineal (SVM) es idéntica a la solución β . La reducción permite de inmediato el uso de solucionadores SVM altamente optimizados para problemas de Elastic Net. La reducción es una transformación simple de los datos originales y las constantes de regularización.

1.7.12 Criterio de sintonía de controladores PID

La intención de un sistema de control realimentado es minimizar el error en la salida, en función del tiempo transcurrido, después de haberse producido una alteración en el sistema. Ambos, la magnitud del error y el tiempo durante el cual existe contribuyen a la definición del control óptimo. Parece lógico proponer un parámetro por medio del cual se caracterice el tiempo de respuesta del sistema por la relación:

$$\phi = \int_0^{\infty} F[e(t), t] dt$$

Donde F es una función de ambos, error y tiempo.

Integrando F con respecto al tiempo se obtiene un número característico del tiempo de respuesta del sistema. El criterio de comportamiento ϕ será cero solamente si el error es cero durante todo el tiempo, lo cual es imposible cuando se han introducido perturbaciones. Por tanto ϕ no desaparecerá, pero cuanto más pequeño sea su valor, mejor será el comportamiento del sistema.

Con el fin de completar la definición matemática del control óptimo, se debe proponer una función $F[e(t), t]$ de forma que ϕ tenga las propiedades primero de ser cero solamente si el error es exactamente cero, y segundo de poseer un mínimo valor consistente con las nociones intuitivas de control óptimo. Puesto que la decisión final de lo que es mejor se basa en juicios o incluso en gustos personales, hay diferentes funciones F , utilizadas para definir el óptimo.

La integral del cuadrado del error (ISE) es un criterio de comportamiento propuesto comúnmente para los sistemas de control.

$$ISE = \int_0^{\infty} e^2(t) dt$$

El ISE (Integral Squared Error), es relativamente insensible a pequeños errores, pero los grandes errores contribuyen al fuertemente al valor de la integral. Consecuentemente, utilizando el ISE como criterio de comportamiento dará como resultado una respuesta con pequeños sobrepasamientos pero largos tiempos de estabilización, puesto que los pequeños errores a lo largo del tiempo contribuyen muy poco a la integral.

La integral del valor absoluto del error (Integral Absolute Error) ha sido propuesto con frecuencia como criterio de comportamiento:

$$IAE = \int_0^{\infty} |e(t)| dt$$

Este criterio es más sensible a pequeños errores, pero menos sensible que el ISE a grandes errores.

Grahan y Lathrop (The synthesis of optimum transient response: criteria and standard forms, Transaction IEEE, nov. 1953), introducen la integral del tiempo multiplicada por el valor absoluto del error (ITAE) como criterio de comportamiento:

$$ITAE = \int_0^{\infty} t|e(t)|dt$$

ITAE (Integral Time Absolute Error), es insensible a los errores iniciales, y a veces inevitables, pero penalizan fuertemente los errores que permanecen a lo largo del tiempo. La respuesta óptima definida por ITAE, consecuentemente, mostrará tiempos cortos de respuesta total y mayores sobre pasamientos que los otros criterios (Acedo Sánchez 2006).

1.7.13 Algoritmos genéticos

Un algoritmo genético es una búsqueda heurística que está inspirada en la teoría de la evolución natural de Charles Darwin. Este algoritmo refleja el proceso de selección natural donde los individuos más aptos son seleccionados para la reproducción con el fin de producir descendencia de la próxima generación.

Noción de selección natural

El proceso de selección natural comienza con la selección de individuos más aptos de una población. Producen descendientes que heredan las características de los padres y se agregarán a la generación siguiente. Si los padres tienen una mejor condición física (fitness), sus hijos serán mejores que los padres y tendrán más posibilidades de sobrevivir. Este proceso continúa iterando y, al final, se encontrará una generación con las personas más aptas. Esta noción se puede aplicar a un problema de búsqueda. Consideramos un conjunto de soluciones para un problema y seleccionamos el conjunto de las mejores. Cinco fases se consideran en un algoritmo genético.

- Población inicial
- Función de la aptitud
- Selección
- Crossover
- Mutación

Población inicial

El proceso comienza con un conjunto de individuos que se llama Población. Cada individuo es una solución al problema que desea resolver. Un individuo (o cromosoma) se caracteriza por un conjunto de parámetros (variables) conocidos como Genes. Los genes se unen en una cuerda para formar un cromosoma. En un algoritmo genético, el conjunto de genes de un individuo se representa utilizando una cuerda, en términos de un alfabeto. Por lo general, se utilizan valores binarios (cadena de unos y ceros). Decimos que codificamos los genes en un cromosoma.

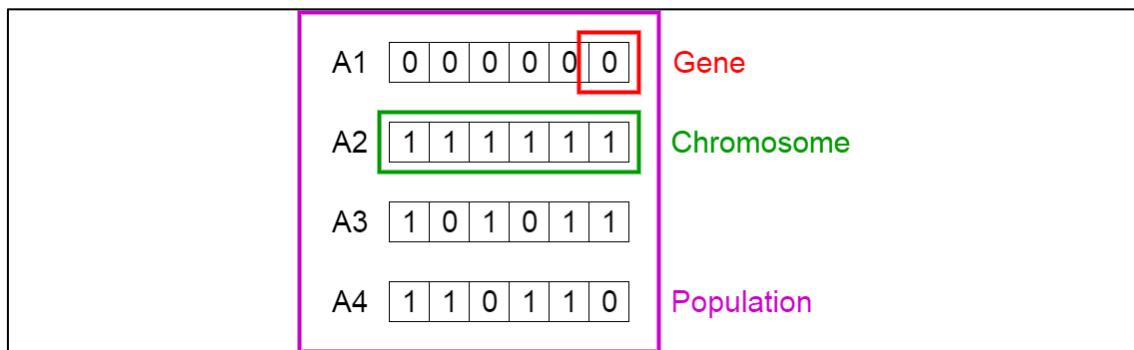


Figura 1.10: Población, cromosoma y gen (Mallawaarachchi 2018).

Función de aptitud

La función de aptitud otorga un puntaje (fitness) a cada individuo, determina qué tan bueno es (capacidad de un individuo para competir con otros individuos). La probabilidad de que un individuo sea seleccionado para la reproducción se basa en su puntaje.

Selección

La idea de la fase de selección es seleccionar a los individuos más aptos y dejarles pasar sus genes a la próxima generación. Se seleccionan dos pares de individuos (padres) en función de sus fitness. Las personas con un mayor fitness tienen más posibilidades de ser seleccionadas para la reproducción.

Crossover

Crossover es la fase más importante en un algoritmo genético. Para cada par de padres que se aparearán, se elige al azar un punto de cruce dentro de los genes. Por ejemplo, considere a tres como el punto de cruce como se muestra a continuación.

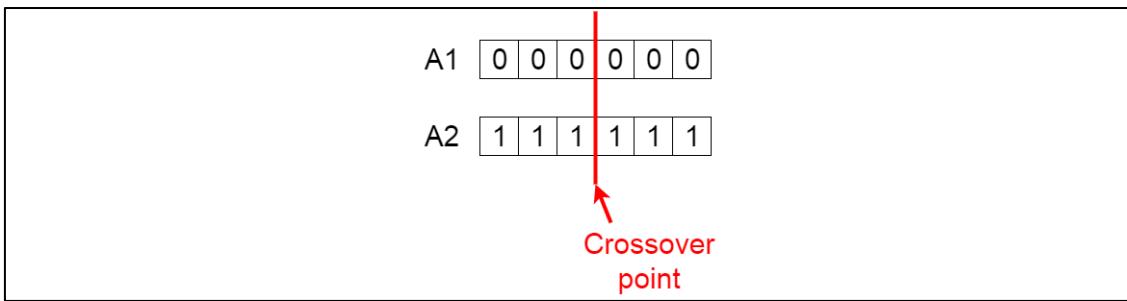


Figura 1.11: Punto de crossover (Mallawaarachchi 2018).

Crossover

Los descendientes se crean intercambiando los genes de los padres entre ellos hasta que se alcanza el punto de cruce.

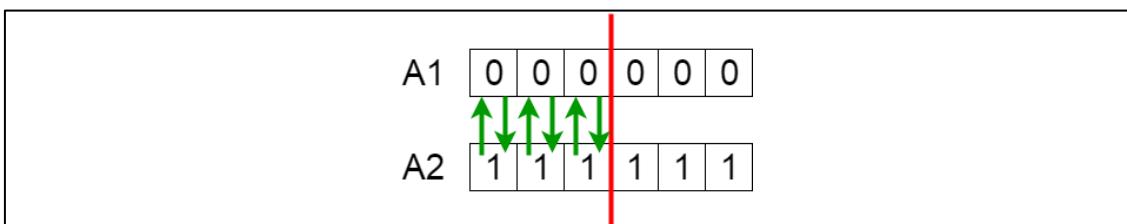


Figura 1.12: Intercambio de genes entre los padres (Crossover) (Mallawaarachchi 2018).

Los nuevos descendientes se agregan a la población.

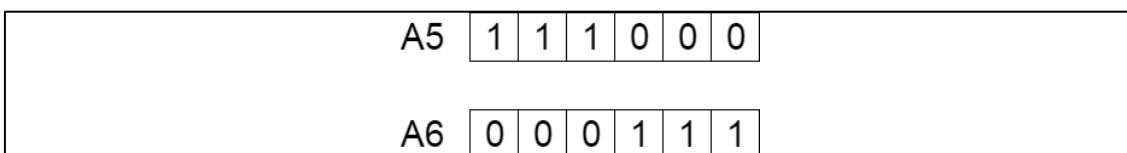


Figura 1.13: Nueva descendencia (Mallawaarachchi 2018).

Mutación

En ciertos descendientes recién formados, algunos de sus genes pueden ser sometidos a una mutación con una baja probabilidad aleatoria. Esto implica que algunos de los bits en la cadena de bits pueden cambiar.

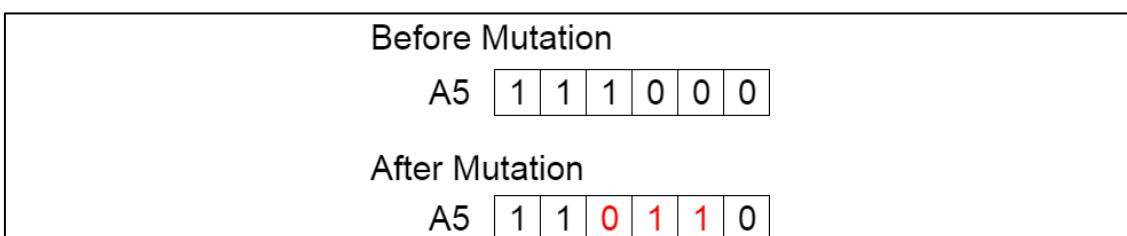


Figura 1.14: Antes y después de una mutación (Mallawaarachchi 2018).

La mutación ocurre para mantener la diversidad dentro de la población y prevenir la convergencia prematura.

Terminación

El algoritmo finaliza si la población ha convergido (no produce descendencia que sea significativamente diferente de la generación anterior) o si se han llevado a cabo un número finito de iteraciones. Luego se dice que el algoritmo genético ha proporcionado una serie de soluciones a nuestro problema.

Comentarios

La población tiene un tamaño fijo. A medida que se forman las nuevas generaciones, mueren las personas con el menor nivel de fitness, lo que proporciona espacio para nuevos descendientes.

La secuencia de fases se repite para producir individuos en cada nueva generación que son mejores que la generación anterior (Mallawaarachchi 2018).

1.7.14 Método de Ziegler – Nichols

Ziegler y Nichols propusieron reglas para determinar los valores de la ganancia proporcional K_p , del tiempo integral T_i y del tiempo derivativo T_d , basándose en las características de respuesta transitoria de una planta dada.

➤ Primer método de sintonización

En el primer método, la respuesta de la planta a una entrada escalón unitario se obtiene de manera experimental, tal como se muestra en la Figura 1.15. Si la planta no contiene integradores ni polos dominantes complejos conjugados, la curva de respuesta escalón unitario puede tener forma de S, como se observa en la Figura 1.16. Este método se puede aplicar si la respuesta muestra una curva con forma de S. Tales curvas de respuesta escalón se pueden generar experimentalmente o a partir de una simulación dinámica de la planta. La curva con forma de S se caracteriza por dos parámetros: el tiempo de retardo L y la constante de tiempo T . El tiempo de retardo y la constante de tiempo se determinan dibujando una recta tangente en el punto de inflexión de la curva con forma de S y determinando las intersecciones de esta tangente con el eje del tiempo y con la línea $c(t) = K$, tal como se muestra en la Figura 1.16 (Ogata 2010).

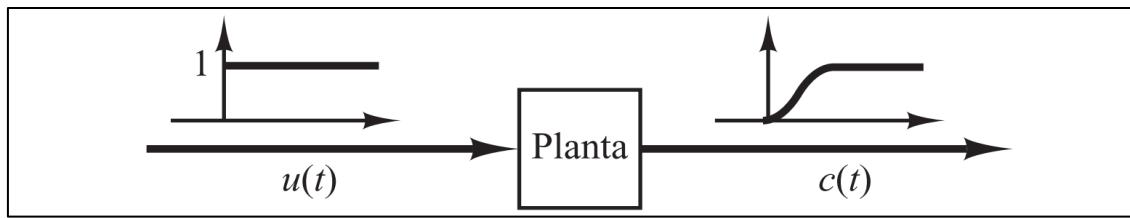


Figura 1.15: Respuesta a un escalón unitario de una planta (Ogata 2010).

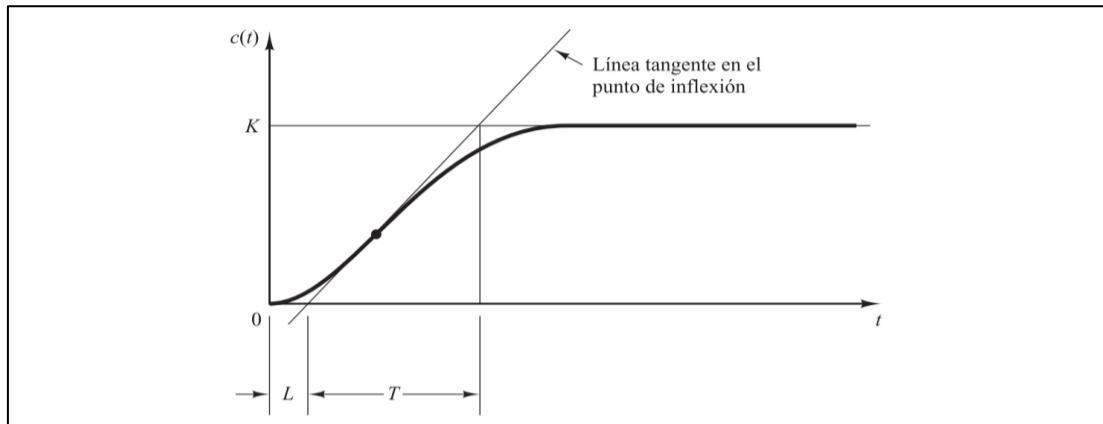


Figura 1.16: Curva de respuesta en forma de S (Ogata 2010).

Tipo de controlador	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9 \frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2 \frac{T}{L}$	$2L$	$0.5L$

Figura 1.17: Primer método de Z-N basada en la respuesta a un escalón de la planta - (Ogata 2010).

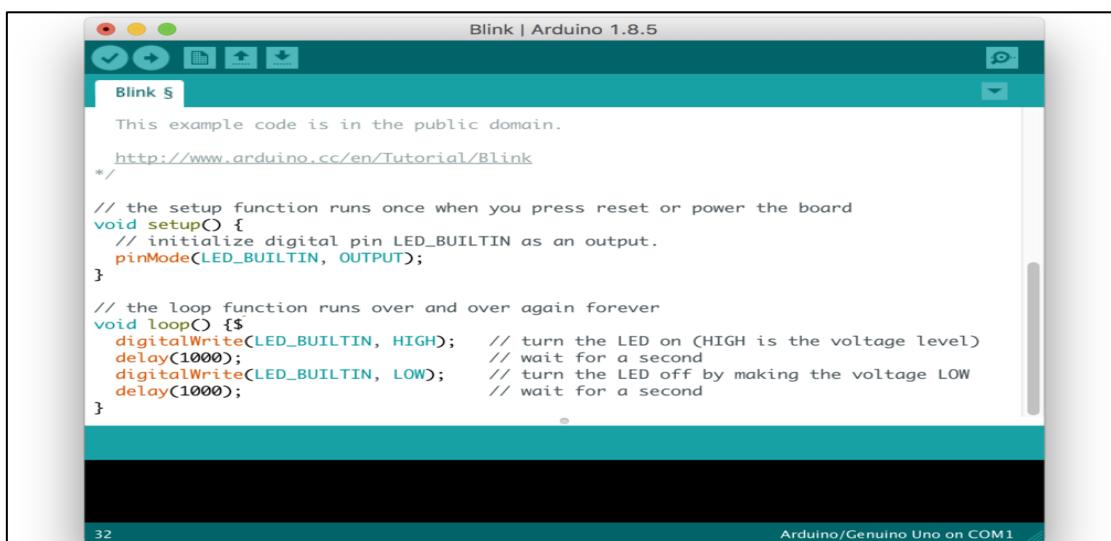
1.7.15 Arduino

Arduino es en realidad tres cosas:

Una **placa hardware libre** que incorpora un microcontrolador reprogramable y una serie de pines-hembra (los cuales están unidos internamente a las patillas de E/S del microcontrolador) que permiten conectar allí de forma muy sencilla y cómoda diferentes sensores y actuadores. Un **software** (más en concreto, un “entorno de desarrollo”) **gratis, libre y multiplataforma** que se debe instalar en un ordenador y que permite escribir, verificar y guardar (“cargar”) en la memoria del microcontrolador de la placa Arduino el

conjunto de instrucciones que deseamos que este empiece a ejecutar. Un **lenguaje de programación libre** que permite especificar las instrucciones exactas que se quiere programar en el microcontrolador de la placa. Estos comandos se escriben mediante el entorno de desarrollo Arduino (Torrente Artero 2013 p. 63-65).

La conexión USB de la placa Arduino, además de servir como alimentación eléctrica, sobre todo es un medio para poder transmitir datos entre el computador y la placa, y viceversa. Este tráfico de información que se realiza entre ambos aparatos se logra gracias al uso del protocolo USB, un protocolo de tipo serie que tanto el computador como la placa Arduino son capaces de entender y manejar. No obstante, el protocolo USB internamente es demasiado complejo para que el microcontrolador ATmega328P pueda comprenderlo por sí mismo sin ayuda, ya que él tan solo puede comunicarse con el exterior mediante protocolos mucho más sencillos técnicamente como son el I2C o el SPI y pocos más. Por tanto, es necesario que la placa disponga de un elemento “traductor” que facilite al ATmega328P (concretamente, al receptor/transmisor serie de tipo TTL-UART que lleva incorporado) la manipulación de la información transferida por USB sin que este tenga que conocer los entresijos de dicho protocolo (Torrente Artero 2013 p. 88).



```

Blink | Arduino 1.8.5

Blink §

This example code is in the public domain.

http://www.arduino.cc/en/Tutorial/Blink

/*
 * the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);     // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}

```

32

Arduino/Genuino Uno on COM1

Figura 1.18: Arduino IDE 1.8.5 on macOS 10.13 (Cedar101 2016).



Figura 1.19: Arduino Uno R3 (SparkFun Electronics 2013).

1.7.16 The jupyter notebook

Jupyter les permite a los usuarios combinar código en vivo, texto enriquecido con rebajas y látex, imágenes, tramas y más en un solo documento. Como el sucesor del notebook IPython, Jupyter fue renombrado ya que la plataforma comenzó a admitir otros kernels de software. Debido a que comenzó con soporte para Julia, Python y R, se renombró como JuPyteR, aunque ahora la plataforma es compatible con Scala, Haskell y Ruby, entre muchos otros (Cook 2017 p. 57).

Notebook document

Notebook document (o "documents") son documentos producidos por la aplicación Jupyter Notebook, que contiene tanto código de computadora (por ejemplo, python) como elementos de texto enriquecido (párrafos, ecuaciones, figuras, enlaces, etc.). Los documentos del cuaderno son documentos legibles por humanos que contienen la descripción del análisis y los resultados (figuras, tablas, etc.) así como documentos ejecutables que se pueden ejecutar para realizar análisis de datos.

Aplicación de Jupyter Notebook

La aplicación Jupyter Notebook es una aplicación cliente-servidor que permite editar y ejecutar documentos portátiles a través de un navegador web. La aplicación Jupyter Notebook se puede ejecutar en un escritorio local que no requiere acceso a Internet (como se describe en este documento) o puede instalarse en un servidor remoto y acceder a través de Internet.

Además de mostrar/editar/ejecutar Notebook documents, la aplicación de Jupyter Notebook tiene un " Dashboard " (Notebook Dashboard), un "panel de control" que muestra los archivos locales y permite abrir Notebook documents o cerrar sus núcleos.

Kernel

Un kernel de notebook es un "motor de computación" que ejecuta el código contenido en un documento de Notebook. El núcleo ipython, por ejemplo, ejecuta código python. Existen núcleos para muchos otros lenguajes (kernels oficiales).

Cuando abre un Notebook document, el kernel asociado se inicia automáticamente. Cuando se ejecuta el Notebook (celda por celda o con el menú Celda -> Ejecutar todo), el kernel realiza el cálculo y produce los resultados. Dependiendo del tipo de cálculos, el

kernel puede consumir CPU y RAM significativas. Tenga en cuenta que la memoria RAM no se libera hasta que el kernel se apaga.

Notebook Dashboard

El Notebook Dashboard es el componente que se muestra primero al iniciar la aplicación Jupyter Notebook. Notebook Dashboard se usa principalmente para abrir Notebook documents y para administrar los kernels en ejecución (visualizar y cerrar).

El panel de instrumentos del Notebook tiene otras características similares a un administrador de archivos, a saber, navegación de carpetas y cambio de nombre / eliminación de archivos (Ingargiola y colaboradores 2015).

1.7.17 MatLab

MATLAB combina un entorno de escritorio perfeccionado para el análisis iterativo y los procesos de diseño con un lenguaje de programación que expresa las matemáticas de matrices y arrays directamente.

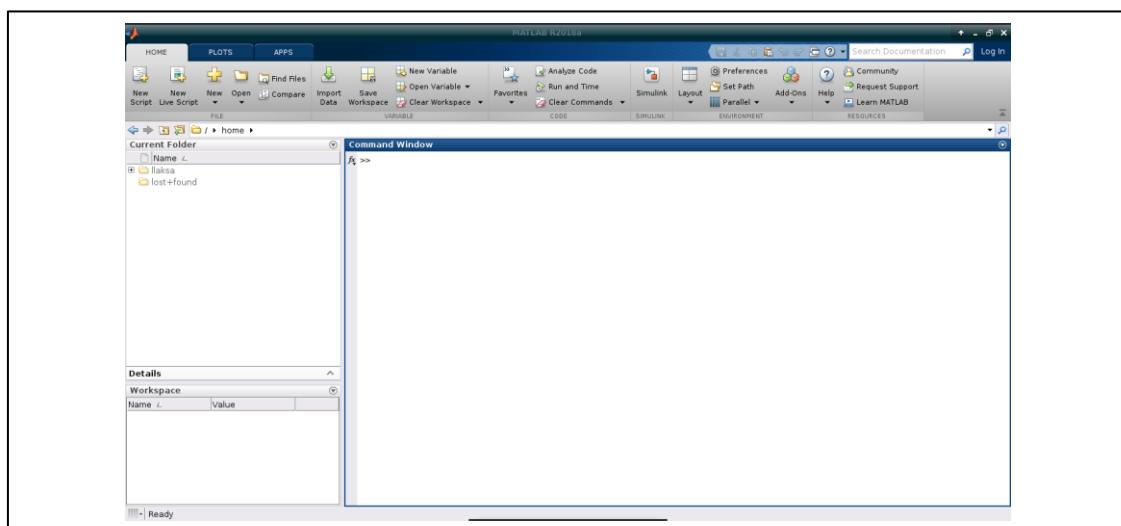


Figura 1.20: Ventana de trabajo de Matlab.

Simulink

Es un entorno de diagramas de bloques para simulación multidominio y diseño basado en modelos. Soporta diseño de nivel de sistema, simulación, generación automática de código y continua prueba y verificación de sistemas embebidos. Simulink provee un editor gráfico, bibliotecas de bloques personalizables y solucionadores de problemas para modelar y simular sistemas dinámicos.

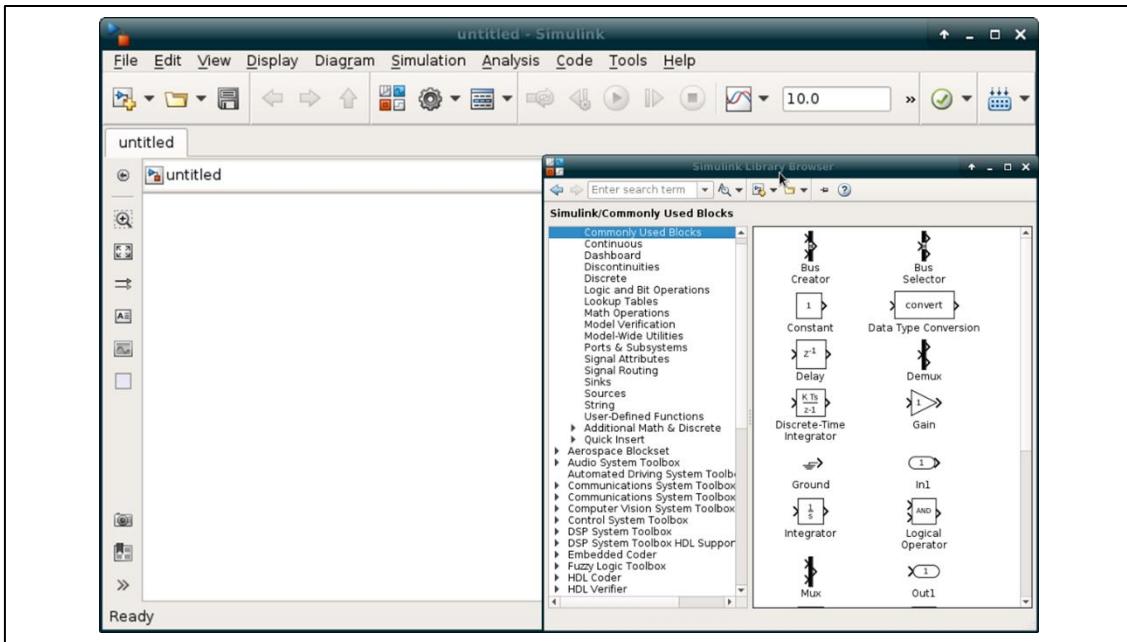


Figura 1.21: Simulink.

System Identification Toolbox

Proporciona a MATLAB funciones, bloques de Simulink y una aplicación para la construcción de modelos matemáticos de sistemas dinámicos a partir de la medición de datos de entrada-salida. Permite crear y utilizar modelos de sistemas dinámicos no fácilmente modelados a partir de principios primarios o especificaciones. Puede utilizar datos de entrada-salida de dominio del tiempo y dominio de la frecuencia para identificar las funciones de transferencia de tiempo continuo y tiempo discreto, modelos de procesos y modelos de espacio de estado. También proporciona algoritmos para estimación de parámetros en línea.

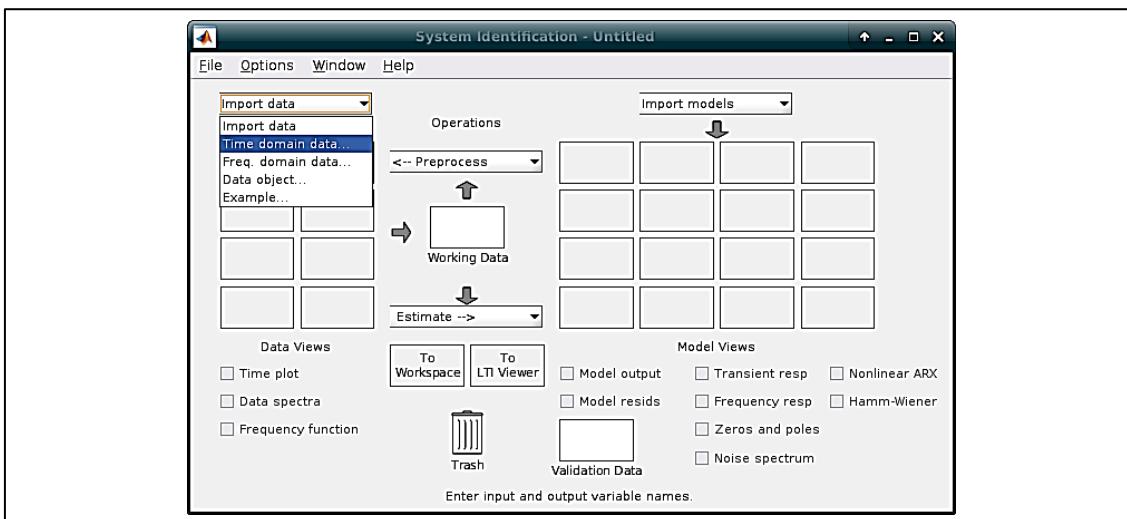


Figura 1.22: Herramienta de identificación de sistemas de Matlab.

System Identification Toolbox calcula los parámetros del modelo minimizando el error entre la salida del modelo y la respuesta medida. Este error, llamado función de pérdida o función de costo, es una función positiva de los errores de predicción. En general, esta función es una suma ponderada de cuadrados de los errores.

La fiabilidad del modelo obtenido se caracteriza por algunos parámetros como:

- **FitPercent:** Error cuadrático medio normalizado (NRMSE) expresado como porcentaje, definido como:

$$FitPercent = 100 \left(1 - \frac{\|y_{measured} - y_{model}\|}{\|y_{measured} - \bar{y}_{measured}\|} \right)$$

Dónde:

$y_{measured}$ son los datos de salida medidos.

$\bar{y}_{measured}$ es la media.

y_{model} es la respuesta simulada o predicha del modelo.

$\|\cdot\|$ Indica la norma-2D de un vector.

FitPercent varía entre $-\infty$ (mal fit) y 100 (fit perfecto). Si el valor es igual a zero, entonces el modelo no es mejor ajustando los datos medidos que una línea recta igual a la media de los datos.

- **MSE:** Error cuadrático Medio, medido como:

$$MSE = \frac{1}{N} \sum_{t=1}^N e^T(t)e(t)$$

Dónde:

$e(t)$ es la señal cuya norma se minimiza para la estimación.

N es el número de muestras de datos en el conjunto de datos de estimación.

PID Tuner

La aplicación del PID Tuner sintoniza automáticamente las ganancias de un controlador PID para una planta SISO para lograr un equilibrio entre rendimiento y robustez. Puede especificar el tipo de controlador, como PI, PID con filtro derivativa o PID de dos grados-de-libertad (2-DOF) controladores. Parcelas de análisis le permiten examinar el rendimiento de controlador de dominios de tiempo y frecuencia.

Puede usar PID Tuner con una planta representada por un modelo numérico de LTI como una función de transferencia o espacio de estado (The Mathworks Inc. Team 2018).

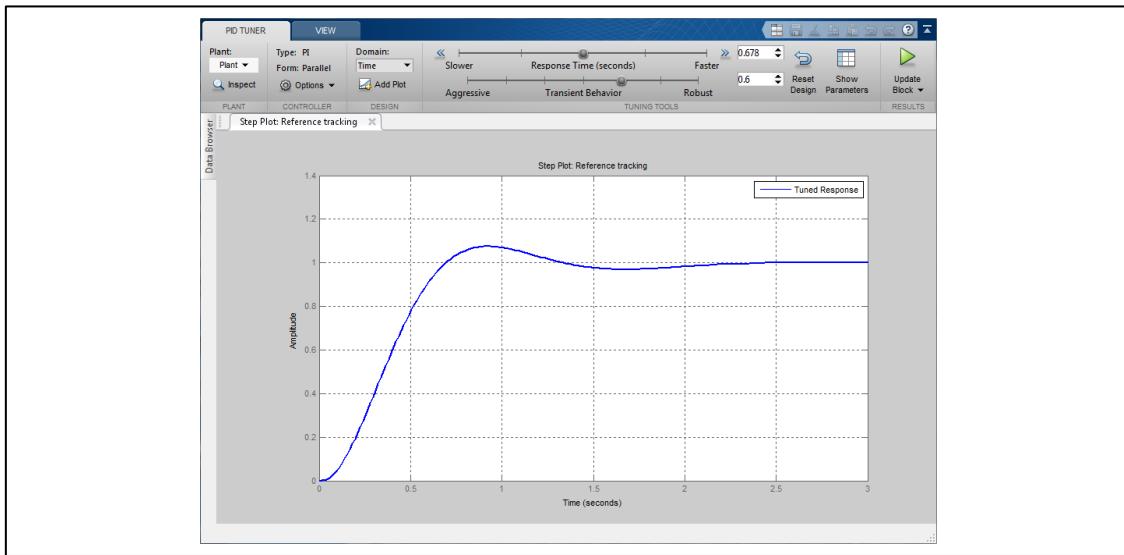


Figura 1.23: PID Tuner.

1.7.18 Identificación de sistema

La identificación del sistema es una metodología para construir modelos matemáticos de sistemas dinámicos que utilizan mediciones de las señales de entrada y salida del sistema. El proceso de identificación de sistema requiere:

- **Medir las señales de entrada - salida del sistema en el dominio de tiempo o frecuencia**

La identificación del sistema utiliza las señales de entrada y salida que mide desde un sistema para estimar los valores de los parámetros ajustables en una estructura de modelo determinada. Obtener un buen modelo de un sistema depende de qué tan bien sus datos medidos reflejen el comportamiento del sistema. Se puede construir modelos usando señales de entrada-salida en el dominio del tiempo, datos de respuesta de frecuencia, señales de series de tiempo y espectros de series de tiempo.

Los datos del dominio del tiempo constan de las variables de entrada y salida del sistema registradas en un intervalo de muestreo uniforme durante un período de tiempo.

Los datos de dominio de frecuencia representan mediciones de las variables de entrada y salida del sistema registradas en el dominio de la frecuencia. Las señales de dominio de frecuencia son transformadas de Fourier de las correspondientes señales de dominio de tiempo. Los datos de dominio de frecuencia también pueden representar la respuesta de

frecuencia del sistema, representada por el conjunto de valores de respuesta complejos en un rango de frecuencia determinado. La respuesta de frecuencia describe las salidas a las entradas sinusoidales.

➤ **Seleccionar una estructura de modelo**

Una estructura de modelo es una relación matemática entre las variables de entrada y salida que contiene parámetros desconocidos. Los ejemplos de estructuras de modelo son las funciones de transferencia con polos y ceros ajustables, ecuaciones de espacio de estado con matrices de sistema desconocidas y funciones parametrizadas no lineales. Se puede usar uno de los siguientes enfoques para elegir la estructura del modelo:

- Modelado de caja negra: Genera el más simple posible modelo que puede reproducir los datos medidos.
- Modelado de caja gris: Genera una estructura específica para el modelo, que puede haber sido derivado de principios primarios, pero no conoce los valores numéricos de sus parámetros. Luego se puede representar la estructura del modelo como un conjunto de ecuaciones o sistema de espacio de estados y estimar los valores de sus parámetros a partir de los datos.

➤ **Aplicar un método de estimación, para estimar el valor de los parámetros ajustables en la estructura de modelo candidata**

Se puede configurar el método de estimación por:

- Configuración del criterio de minimización para enfocar la estimación en un rango de frecuencia deseado. O configurar la simulación o la predicción como los criterios y elegir como objetivo a las necesidades de la aplicación prevista para el modelo.
- Especificación de opciones de optimización para algoritmos de estimación iterativa.

➤ **Evaluar el modelo estimado**

En última instancia, se debe evaluar la calidad del modelo en función de si el modelo atiende adecuadamente las necesidades de la aplicación.

1.8 Marco conceptual

1.8.1 Señal de entrada típica

Las señales de prueba que se usan regularmente son funciones escalón, rampa, parábola, impulso, etc. Con estas señales de prueba, es posible realizar con facilidad análisis matemáticos y experimentales de sistemas de control, ya que las señales son funciones del tiempo muy simples. La forma de la entrada a la que el sistema estará sujeto con mayor frecuencia en una operación normal determina cuál de las señales de entrada típicas se debe usar para analizar las características del sistema.

1.8.2 Proceso de muestreo

El muestreo de señales en tiempo continuo reemplaza la señal en tiempo continuo por una secuencia de valores en puntos discretos de tiempo. Y es seguido por un proceso de cuantificación (Ogata 2010).

1.8.3 Tiempo de establecimiento

El tiempo de asentamiento o establecimiento es el tiempo que se requiere para que la curva de respuesta alcance un rango alrededor del valor final del tamaño especificado por el porcentaje absoluto del valor final (por lo general, de 2 o 5%). El tiempo de asentamiento se relaciona con la mayor constante de tiempo del sistema de control. Los objetivos del diseño del sistema en cuestión determinan qué criterio de error en porcentaje utilizar (Ogata 2010).

1.8.4 Encoder

Un encoder óptico se compone de un par de dispositivos optoelectrónicos, uno de los cuales constituye una fuente de luz y el otro es el receptor. Entre la fuente y el receptor se coloca un disco ranurado, acoplado mecánicamente al eje, compuesto por ranuras transparentes. Cuando el disco gira se produce una señal alterna entre el emisor y el receptor (Corona Ramírez, Abarca Jiménez y Mares Carreño 2014).

2 CAPÍTULO II: MARCO METODOLÓGICO

2.1 Variables

2.1.1 Variables independientes

- Amplitud de señal de entrada.
- Duración de señal de entrada.
- Frecuencia de muestreo.
- Número de muestras.

2.1.2 Variables dependientes

- Velocidad del motor.
- Ganancia proporcional.
- Ganancia integral.
- Ganancia derivativa.

Tabla 2.1: Variables.

Variable	Definición conceptual	Definición operacional	Indicador	Escala de medición
Amplitud de señal de entrada.	Medida de la variación máxima del voltaje que varía periódicamente en el tiempo.	Valor cuadrático medio de una señal PWM de excitación del sistema.	V	0 – 12
Duración de señal de entrada.	Tiempo que dura una señal de entrada.	Tiempo en que se mantiene la amplitud de una señal de entrada.	s	> 0

Frecuencia de muestreo	Número de muestras por unidad de tiempo que se toman de una señal continua para producir una señal discreta.	Número de muestras por segundo de los valores de manipulación del duty cycle de la señal PWM y los valores de velocidad motor.	Hz	> 0
Número de muestras	Cantidad de porciones extraídas de un conjunto por métodos que permiten considerarla como representativa de él.	Cantidad de datos obtenidos por el proceso de muestreo digital.	n°	> 0
Velocidad del motor	Magnitud física que expresa el ángulo de scrito en la unidad d etiempo por el radio de un cuerpo que gira en torno de un eje.	Magnitud física que expresa la cantidad de radianes por segundo recorridos por el eje del motor.	rad/s	> 0
Ganancia proporcional	Valor del controlador que puede reducir, pero no eliminar, el error en estado estacionario.	Salida del controlador es proporcional a la señal de error.	n°	> 0

Ganancia integral	Valor del controlador que elimina errores estacionarios.	Salida del controlador que es la integral de la señal de error.	nº	> 0
Ganancia derivativa	Valor del controlador que anticipa el efecto de la acción proporcional para estabilizar la variable controlada después de cualquier perturbación.	Salida del controlador que es la derivada de la señal de error.	nº	> 0

2.2 Metodología

2.2.1 Tipo de estudio

Exploratorio

2.2.2 Diseño

Experimental.



Figura 2.1: Diagrama de flujo del diseño experimental.

2.3 Población y muestra

2.3.1 Población

Motores dc.

2.3.2 Muestra

Motor dc de 12V de voltaje de operación nominal.

2.4 Método de investigación

Cuantitativo.

2.5 Técnicas e instrumentos de recolección de datos

2.5.1 Técnicas

- Revisión de bibliografía especializada para explorar señales de excitación sobre diversos sistemas de control.
- Estudio del comportamiento del sistema de control frente a diversos estímulos.

2.5.2 Instrumentos

- Jupyter Notebook
- Plataforma Arduino
- MatLab

2.5.3 Métodos de análisis de datos

Pre procesamiento de la señal de salida de la planta de control: aplicación de filtrado digital sobre los datos provenientes del sensor para su posterior procesamiento.

Contrastación del desempeño de los controladores obtenidos por distintos métodos.

3 CAPÍTULO III: DESARROLLO Y RESULTADOS

En la Tabla 3.1 y la Tabla 3.2 se listan todos los elementos utilizados:

ELEMENTO	DESCRIPCIÓN
Arduino IDE	Entorno de programación para lograr que la placa Arduino funcione como tarjeta de adquisición de datos y como elemento controlador.
Jupyter Notebook	Aplicación en donde se realizará el proceso de identificación de sistema y el proceso de sintonización usando algoritmos genéticos.
MatLab	Programa que permitirá realizar procesos de sintonización usando el PID Tuner y el método de Ziegler-Nichols.

Tabla 3.1: Elementos de software.

ELEMENTO	ESPECIFICACIONES	DESCRIPCIÓN
Computadora	Procesador 4 núcleos mínimo	Almacena y procesa los datos.
Placa Arduino Uno R3	Microcontrolador: ATmega328P Voltaje de Operación: 5V Voltaje de alimentación: 6-20V Corriente máxima entrada/salida: 40mA	Tarjeta de adquisición de datos y elemento controlador
<u>Driver Mosfet IRF520</u>	Voltaje de salida: 0-24 V DC Voltaje de control: 5V TTL Corriente máxima: 9 A Corriente nominal: 6 A	Amplifica la salida del controlador y actúa sobre la planta de control
Motor dc	Operación nominal: 6 a 12 VDC	Planta de control
Encoder	Voltaje de Operación: 5V Acople mecánico de un disco de 4 pulsos por vuelta	Elemento sensor
Fuente de voltaje	Voltajes de salida: 5 y 12 VDC	Alimentación del sistema de control

Tabla 3.2: Elementos de hardware.

3.1 Etapa de adquisición y procesamiento de datos

En esta etapa, la placa Arduino será programada para funcionar como tarjeta de adquisición de datos. La placa se comunicará por medio del puerto serial a la laptop. En la laptop se realizará el procesamiento de datos a través de Jupyter Notebook. Notar que, obviando la laptop, la disposición de elementos corresponde a un sistema de control en lazo abierto.

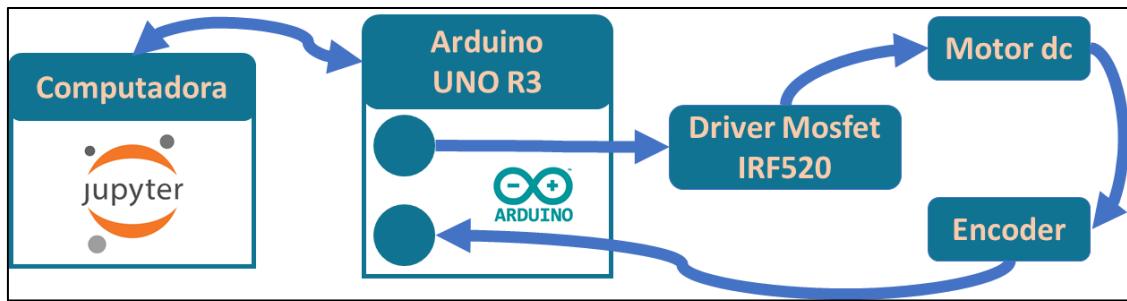


Figura 3.1: Esquema de la disposición de los elementos de software y hardware.

3.1.1 Implementación del sistema de control de velocidad

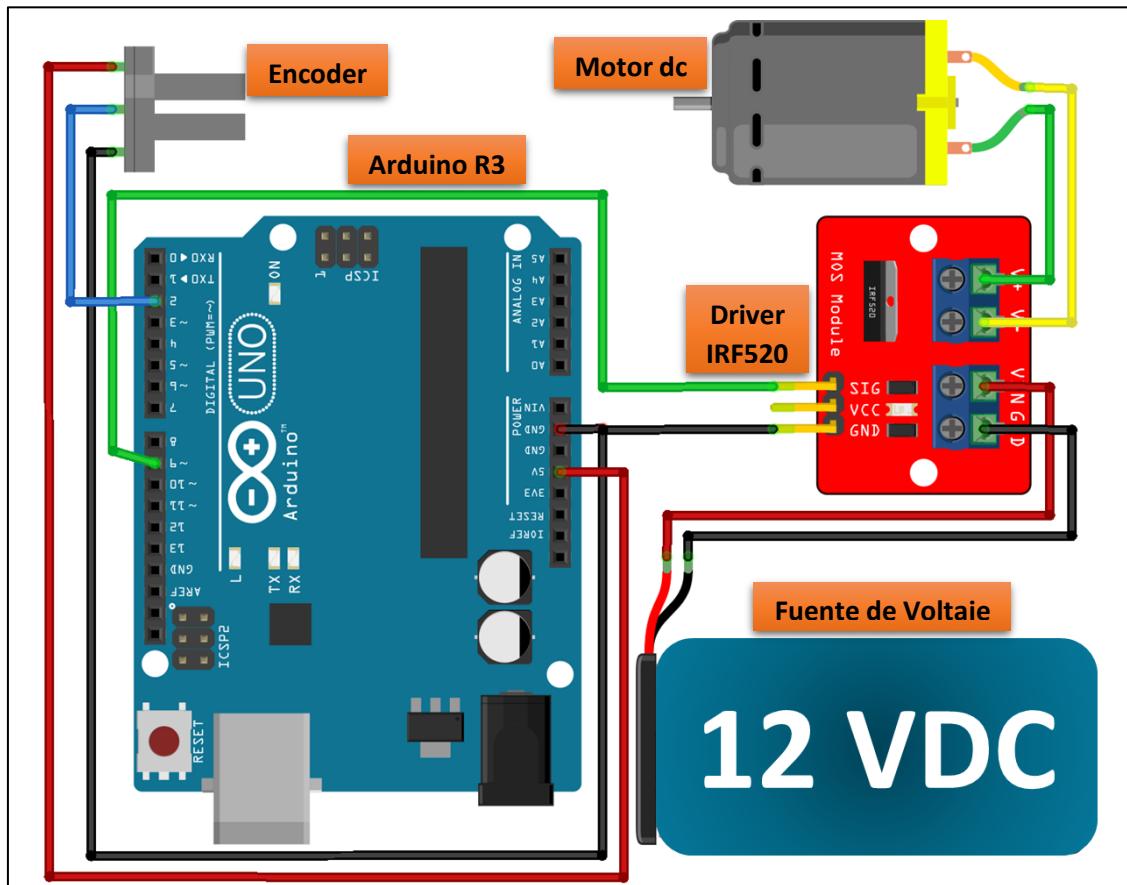


Figura 3.2: Disposición de los elementos de hardware del sistema de control.

3.1.2 Elección de las señales de entrada típica y muestreo de datos

El sistema estará sujeto a perturbaciones repentinas. Por lo tanto, una señal de entrada de tipo escalón será apropiada para obtener la mayor información de la planta de control.

Por otro lado, el motor dc admite un valor máximo de voltaje de 12 voltios, que es cuando alcanza su máxima velocidad. Para manipular la velocidad (rad/s) y simular una señal de entrada tipo escalón con amplitud variable, lo que se varió fue el valor cuadrático medio

del voltaje del motor, que puede ser manipulado por una señal PWM; de esta forma, una señal PWM con 0% de duty cycle (correspondiente al valor 0 programado en el IDE de Arduino para un pin PWM) permite obtener un valor de voltaje promedio de 0, y una señal PWM con 100% de duty cycle (correspondiente al valor 255 programado en el IDE de Arduino para un pin PWM) permite obtener el valor de voltaje cuadrático medio de 12 V.

Se observó que una señal PWM (señal de entrada) que cambia su duty cycle de 0% a 100%, necesitó un tiempo de 4.798s para dejar que la velocidad del motor (señal de salida) se estabilice. Y se probó que cualquier otro cambio en la señal PWM, necesitaba un tiempo de establecimiento menor. Notar que el tiempo de 4.798s corresponde a 147 ciclos del timer del controlador, el cual se programó a 32.64ms por ciclo.

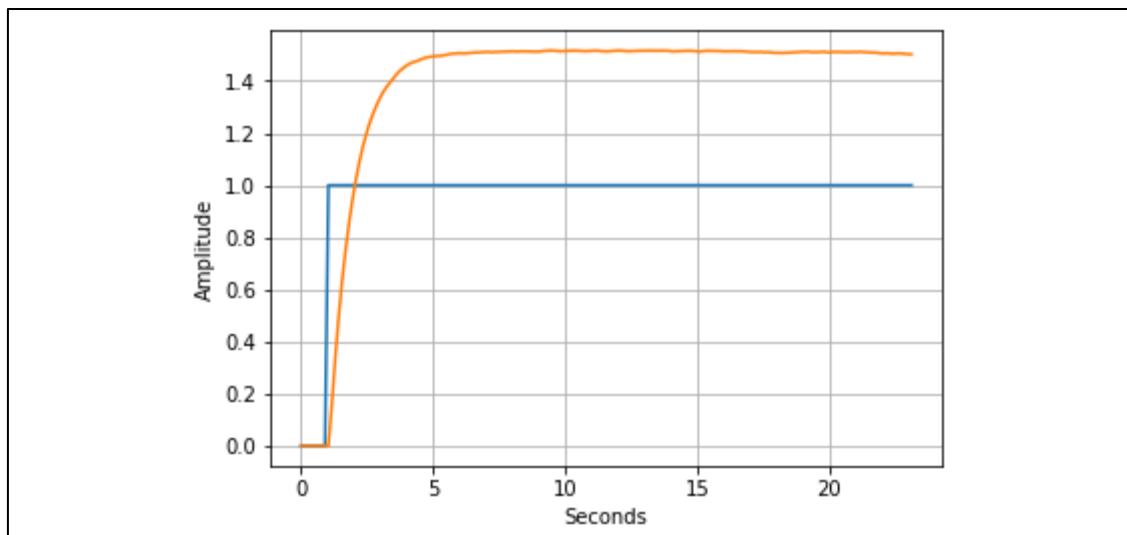


Figura 3.3: Ejemplo de una señal de excitación para el sistema (color azul) y su respuesta (color naranja).

Si bien sería suficiente una señal como la de la Figura 3.3 para el muestreo digital y la identificación del motor dc a través de una función de transferencia (relacionada al dominio de la frecuencia); para obtener un modelo de machine learning apropiado (relacionado al dominio del tiempo) será necesario asignar los siguientes parámetros para la señal de entrada:

- ✓ **Tipo:**
Escalón

✓ **Amplitud:**

0 – 12 V (voltaje RMS gobernado por valores entre 0 y 255 programados en el IDE de Arduino para controlar el duty cycle de la señal PWM de entrada entre 0% y 100%)

✓ **Duración:**

4.798 s

Notar entonces que la señal de entrada mostrada en la Figura 3.4 cambió aleatoriamente de amplitud cada 4.798s y permitió la identificación del sistema que resultó en la obtención del modelo de machine learning.

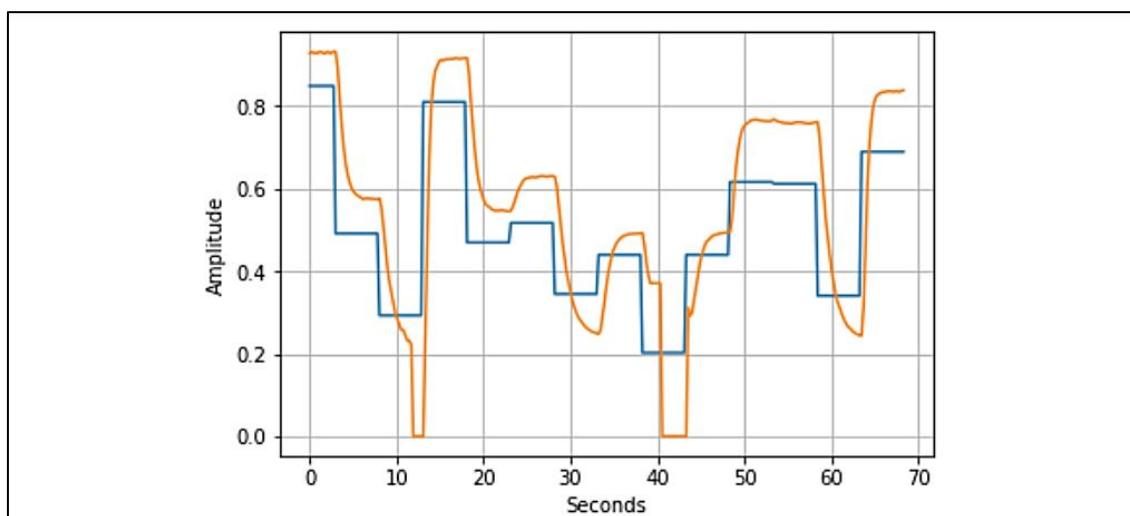


Figura 3.4: Señal de entrada para el sistema (color azul) y su velocidad (color naranja) en una ventana de tiempo de 70s.

Para el proceso de muestreo digital, se definió un tiempo de muestreo de 0.22848s. Además, debido al ruido en la salida del sistema, fue necesario usar un filtro digital pasa bajo de primer orden con una frecuencia de corte $f_c = 0.1\text{Hz}$ y el tiempo de muestreo antes definido $T_s = 228.48\text{ ms}$:

$$y_{(k)} = 0.1337 * u_{(k-1)} + 0.8663 * y_{(k-1)}$$

Notar que el tiempo de 0.22848s corresponde a 7 ciclos del timer del controlador programado en 32.64ms.

Se tomaron 20000 muestras de la señal mostrada en la Figura 3.4 que fueron guardadas en la hoja de cálculo “motor. xlsx” para la identificación de sistema por medio de un modelo de machine learning y se tomaron 20000 muestras de la señal mostrada en la

Figura 3.3 que fueron guardadas en la hoja de cálculo “motor-stepentry.xlsx” para la identificación de sistema por medio de una función de transferencia. A continuación, se muestran los diagramas de flujo del proceso de muestreo digital:

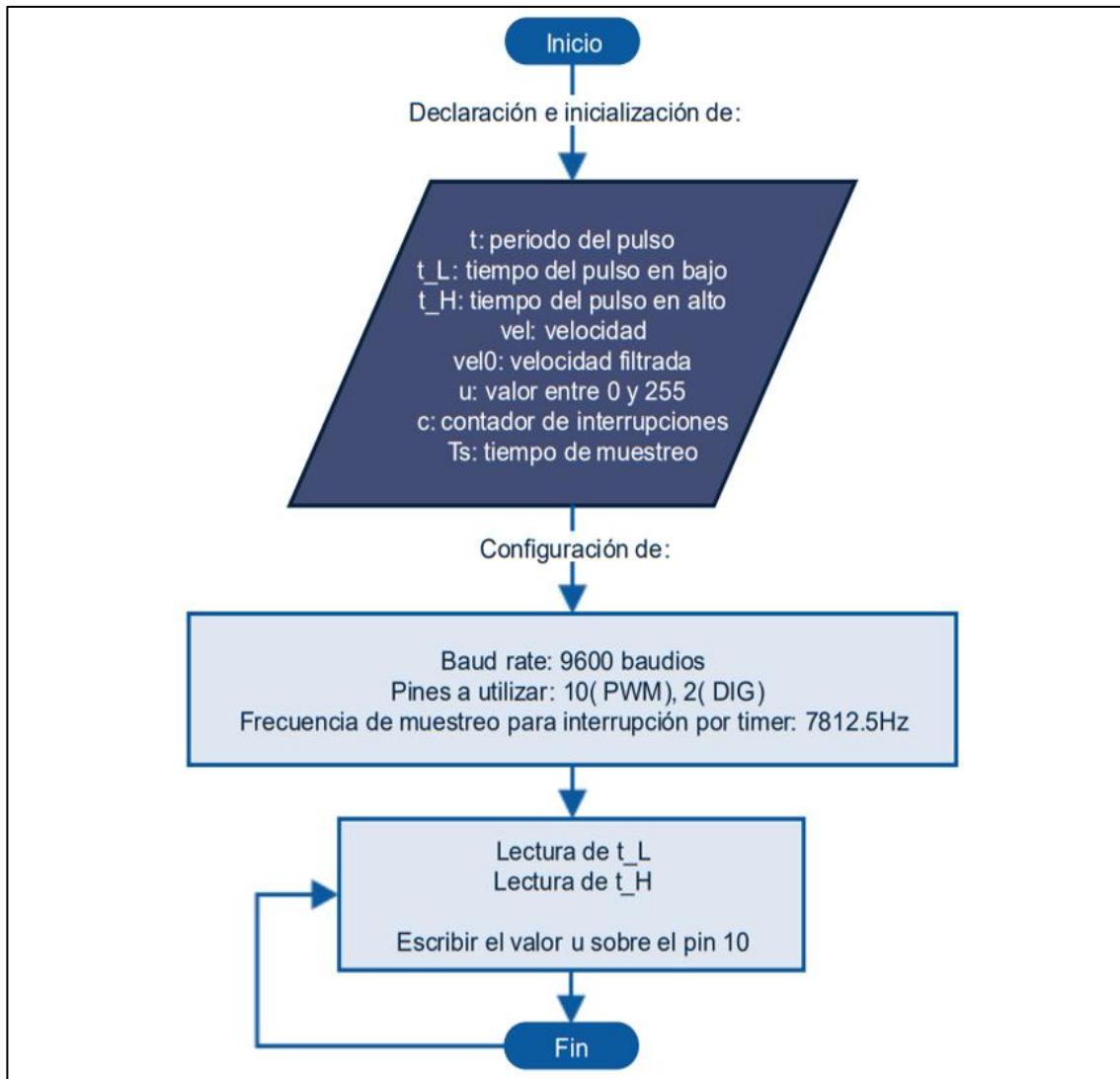


Figura 3.5: Procesos ligados al microprocesador del uC.

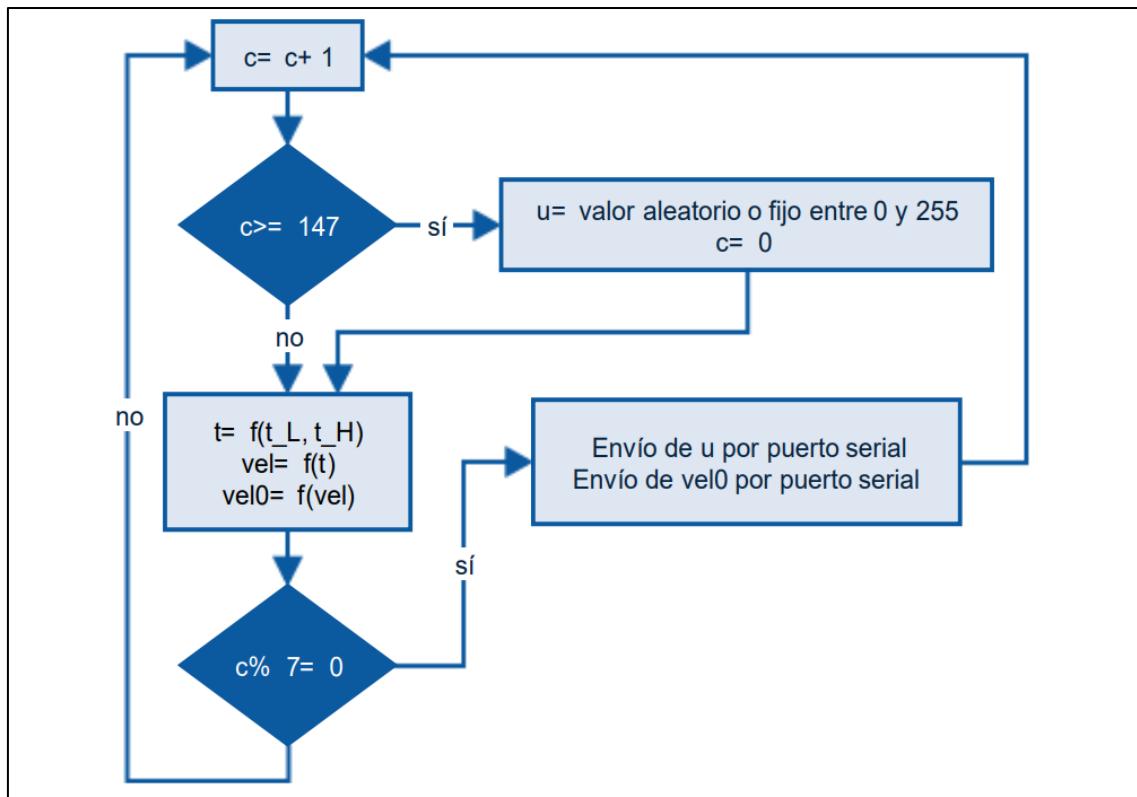


Figura 3.6: Procesos ligados al timer del uC.

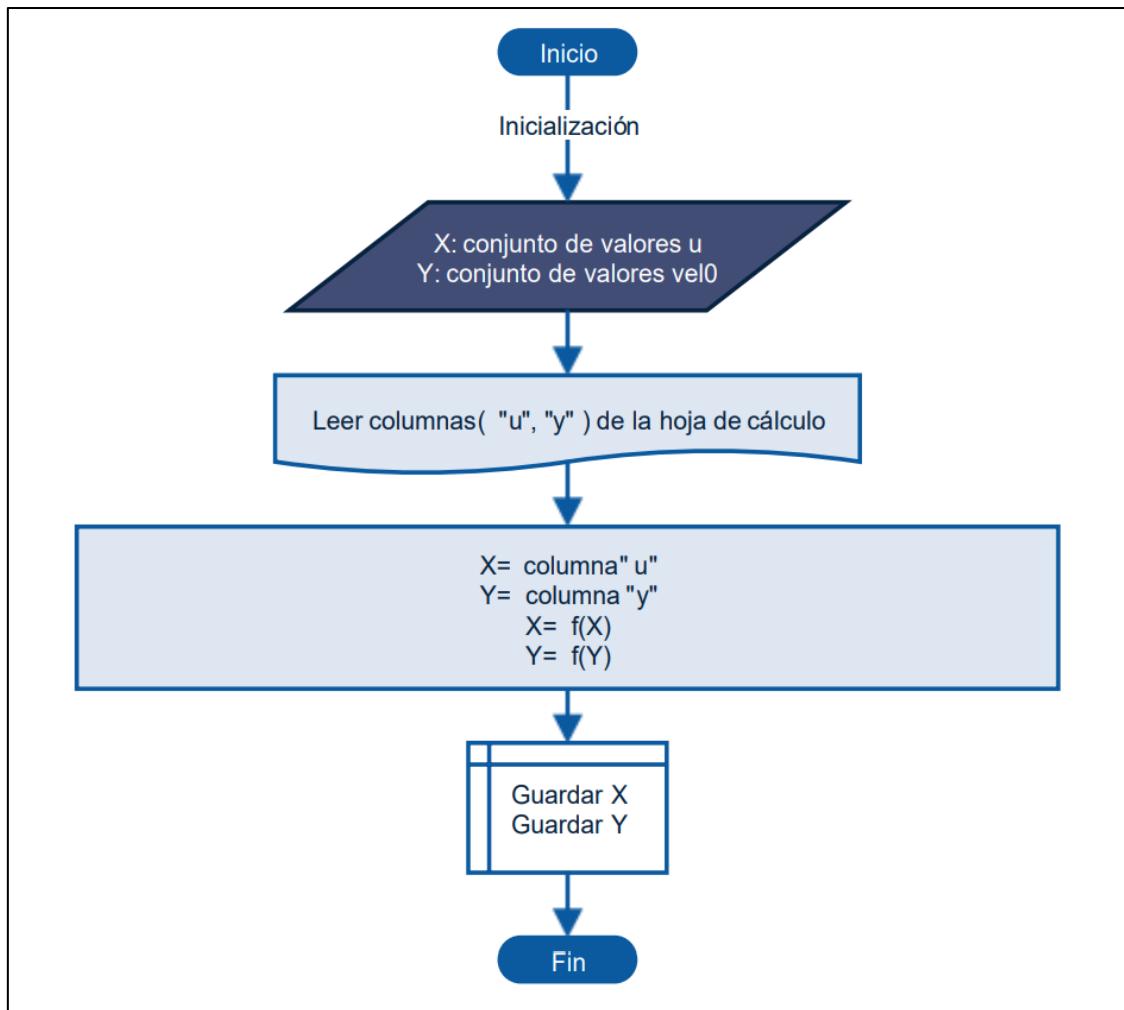


Figura 3.7: Almacenamiento de datos en memoria.

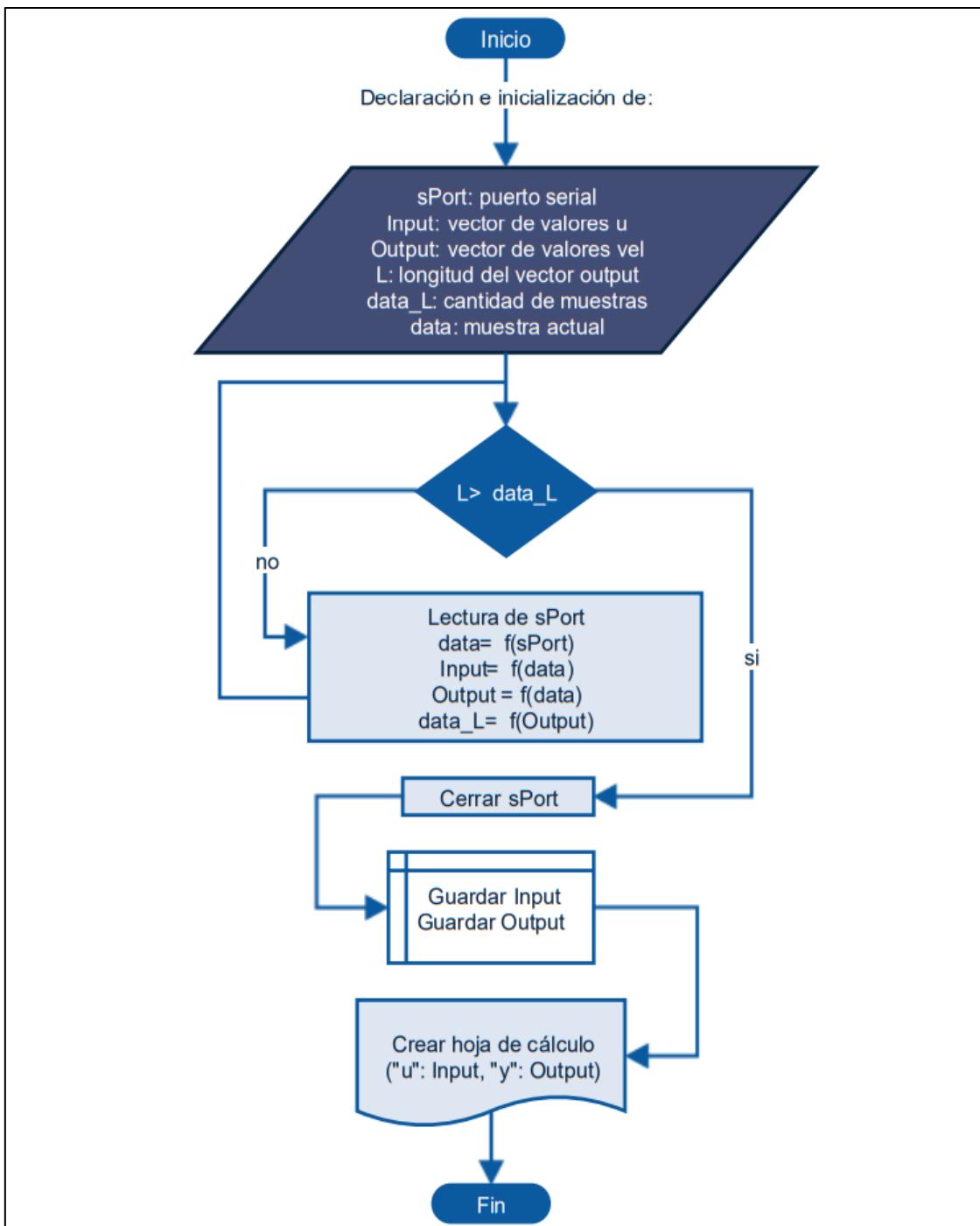


Figura 3.8: Creación de la hoja de cálculo.

El código correspondiente a los diagramas mostrados anteriormente se muestra en la sección de anexos correspondiente.

3.1.3 Identificación de sistema del motor dc y obtención de modelos

3.1.3.1 Modelo 1: Modelo de machine learning:

Se importó la hoja de cálculo “motor.xlsx”.

	u	y
0	200	0.00
1	200	0.00
2	200	0.00
3	200	0.00
4	200	0.00
5	200	0.00
6	200	8.80
7	200	38.14
8	200	72.02
9	200	103.73
10	200	132.93
25	200	296.59
26	200	300.25
27	200	302.42
28	200	303.53
29	200	305.75
...
19970	39	0.00
19971	39	0.00
19972	39	0.00
19973	39	0.00
19974	39	0.00
19975	39	0.00
19989	39	0.00
19990	39	0.00
19991	39	0.00
19992	39	0.00
19993	39	0.00
19994	144	0.00
19995	144	0.00
19996	144	0.00
19997	144	0.00
19998	144	12.40
19999	144	34.54

20000 rows x 2 columns

Figura 3.9: Vista de los datos guardados en la hoja de cálculo.

En donde la columna “u” representa el vector de valores entre 0 y 255 que gobiernan el voltaje RMS que recibe el motor y la columna “y”, el vector de velocidades del sistema (rad/s).

Para la identificación del motor dc por medio de un modelo de machine learning a partir de los datos obtenidos en la hoja de cálculo, se determinaron los parámetros de los conjuntos de entrada y salida para el entrenamiento y validación del modelo en base a cinco muestras (correspondientes a una ventana de tiempo de $1.1424s = 5 \times 0.22848s$). Quedando:

➤ Entrenamiento y validación del modelo de machine learning

Parámetros del conjunto de entrada:

- x_1 : Un valor de la columna “u”
- x_2 : Valor de la columna “u” anterior a x_1
- x_3 : Valor de la columna “y” relacionado a x_2

- x_4 : Valor de la columna “u” anterior a x_2
- x_5 : Valor de la columna “y” relacionado a x_4
- x_6 : Valor de la columna “u” anterior a x_4
- x_7 : Valor de la columna “y” relacionado a x_6
- x_8 : Valor de la columna “u” anterior a x_6
- x_9 : Valor de la columna “y” relacionado a x_8

Parámetros del conjunto de salida:

- y_1 : Valor de la columna “y” relacionado a x_1

Entonces los parámetros quedan de la siguiente manera:

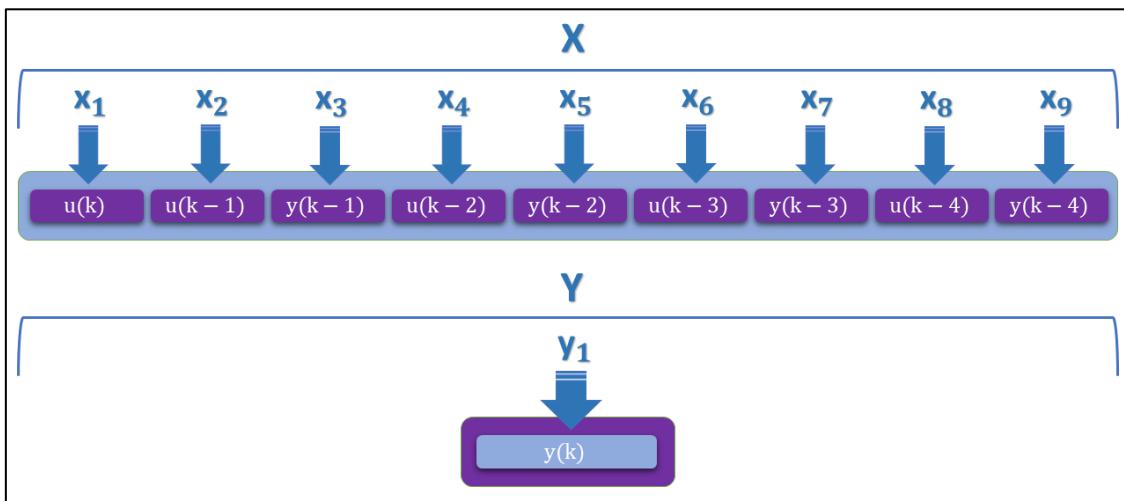


Figura 3.10: Parámetros de entrada X y salida Y para entrenar y validar el modelo de machine learning.

	u	u1	y1	u2	y2	u3	y3	u4	y4
0	200	200.0	0.00	200.0	0.00	200.0	0.00	200.0	0.00
1	200	200.0	0.00	200.0	0.00	200.0	0.00	200.0	0.00
2	200	200.0	0.00	200.0	0.00	200.0	0.00	200.0	8.80
3	200	200.0	0.00	200.0	0.00	200.0	8.80	200.0	38.14
4	200	200.0	0.00	200.0	8.80	200.0	38.14	200.0	72.02
5	200	200.0	8.80	200.0	38.14	200.0	72.02	200.0	103.73
19990	39	39.0	0.00	39.0	0.00	39.0	0.00	144.0	0.00
19991	39	39.0	0.00	39.0	0.00	144.0	0.00	144.0	0.00
19992	39	39.0	0.00	144.0	0.00	144.0	0.00	144.0	0.00
19993	39	144.0	0.00	144.0	0.00	144.0	0.00	144.0	0.00
19994	144	144.0	0.00	144.0	0.00	144.0	0.00	144.0	12.40
19995	144	144.0	0.00	144.0	0.00	144.0	12.40	144.0	34.54

19996 rows × 9 columns

Figura 3.11: Vista del conjunto de entrada generado.

0	0.00	25	296.59	19986	0.00
1	0.00	26	300.25	19987	0.00
2	0.00	27	302.42	19988	0.00
3	0.00	28	303.53	19989	0.00
4	0.00	29	305.75	19990	0.00
5	0.00		...	19991	0.00
6	8.80	19966	0.00	19992	0.00
7	38.14	19967	0.00	19993	0.00
8	72.02	19968	0.00	19994	0.00
9	103.73	19969	0.00	19995	0.00
10	132.93	19970	0.00		Name: y, Length: 19996, dtype: float64

Figura 3.12: Vista del conjunto de salida generado.

Como se puede observar, el número de datos mostrados en la Figura 3.12 es menor en 4 datos que la Figura 3.9, esto es debido a que los parámetros elegidos según la Figura 3.10, tienen un impacto en la construcción del conjunto de entrada, quedando menos datos debido a la eliminación de datos indeterminados. Ya que los conjuntos de entrada y salida deben tener la misma cantidad de datos, dicho impacto también debe reflejarse en el conjunto de salida, que entonces será igual a los valores de la columna “y” pero sin los últimos cuatro valores.

El modelo de machine learning utilizado fue el modelo de regresión lineal “Elastic Net” de la biblioteca “Scikit Learn” de python.

Se usó el 60% de los datos para entrenar el modelo y el 40% de los datos para validarlo, según el esquema de la Figura 3.13:

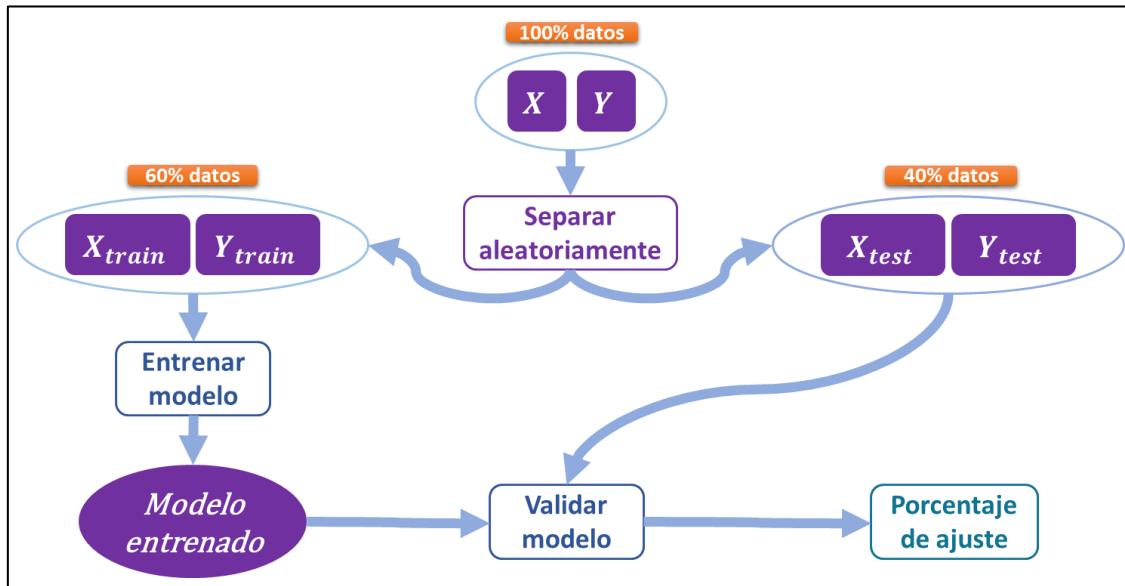


Figura 3.13: Esquema de entrenamiento y validación del modelo.

Se obtuvo un ajuste de más del 99% sobre el conjunto de prueba, según se puede observar en la Figura 3.14:

```
model.score(x_test,y_test)
0.9982657593973125
```

Figura 3.14: Porcentaje de ajuste sobre el conjunto de prueba.

➤ **Uso del modelo:**

La Figura 3.15 muestra el formato de la entrada para usar el modelo de machine learning y la salida que genera.

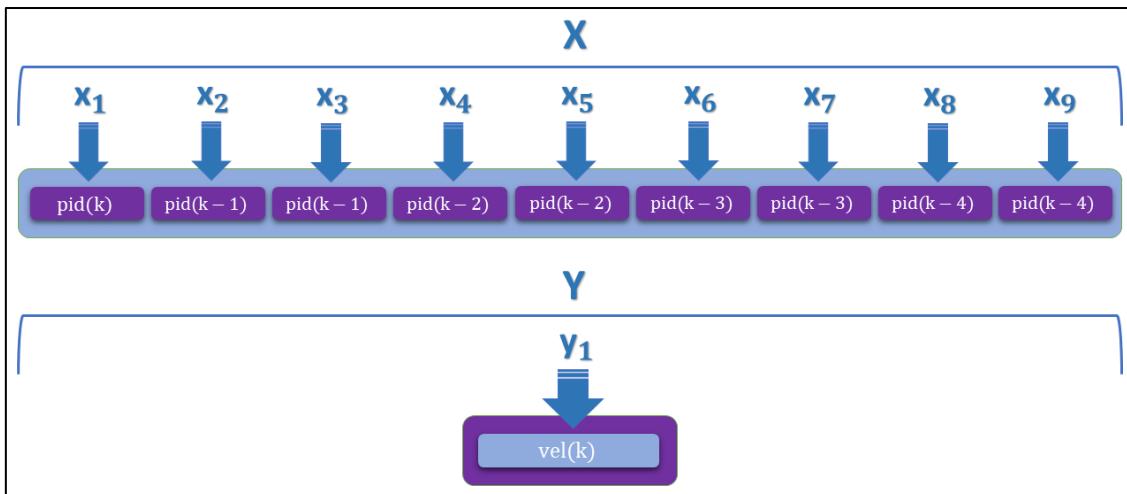


Figura 3.15: Entrada X para usar el modelo y la salida Y que genera.

Donde pid se refiere al valor de salida del controlador y vel, al valor de la velocidad. Entonces el modelo de machine learning puede usarse según la Figura 3.16.

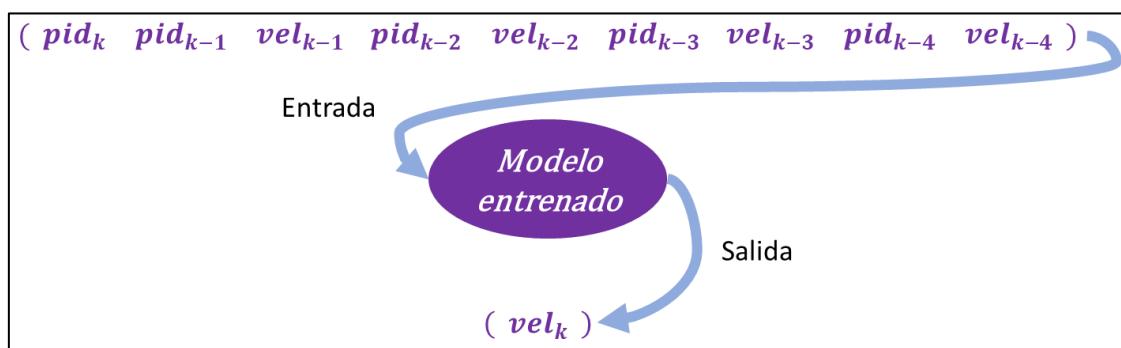


Figura 3.16: Esquema de uso del modelo.

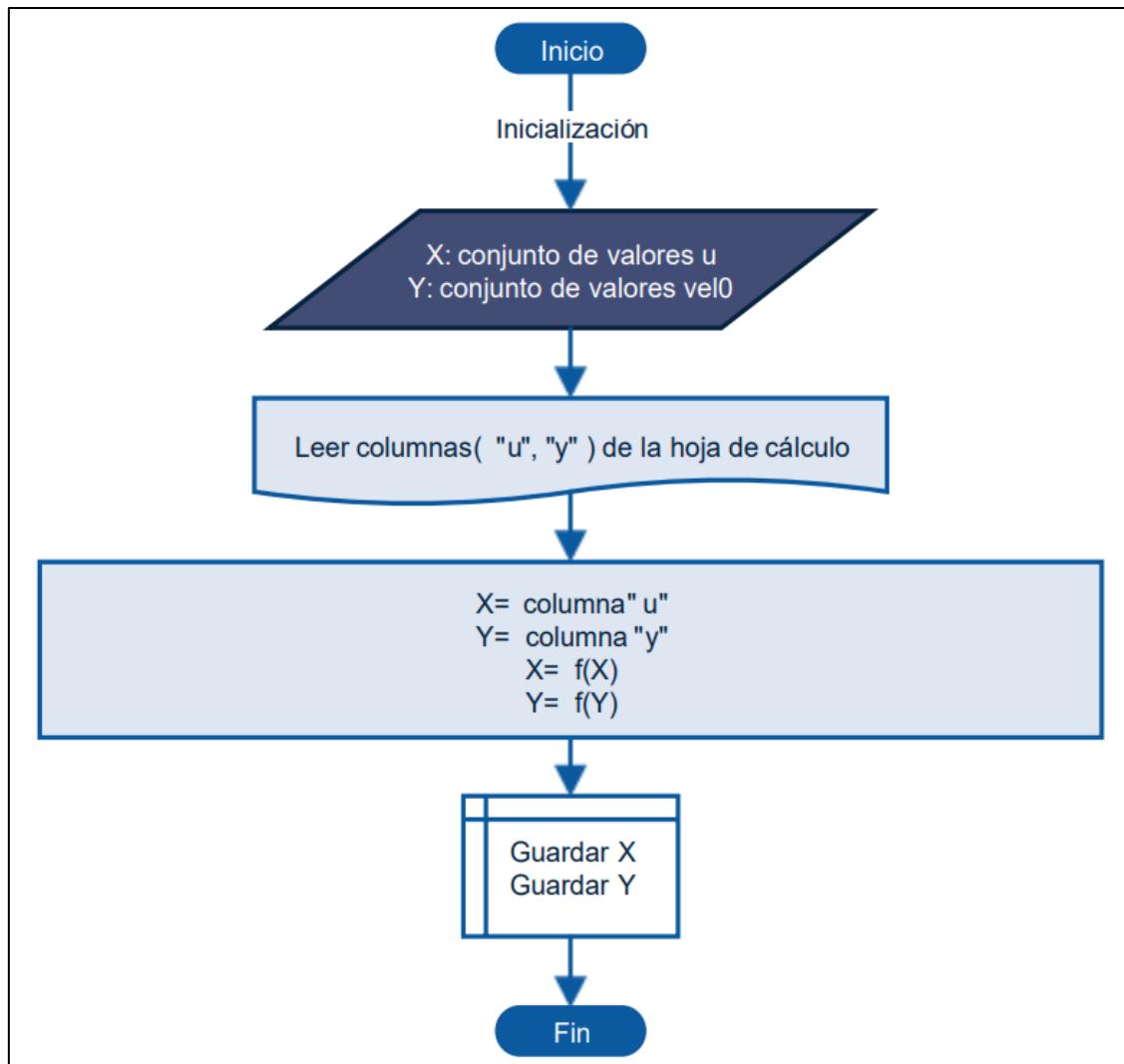


Figura 3.17: Diagrama de flujo del proceso de generación de los conjuntos de entrada.

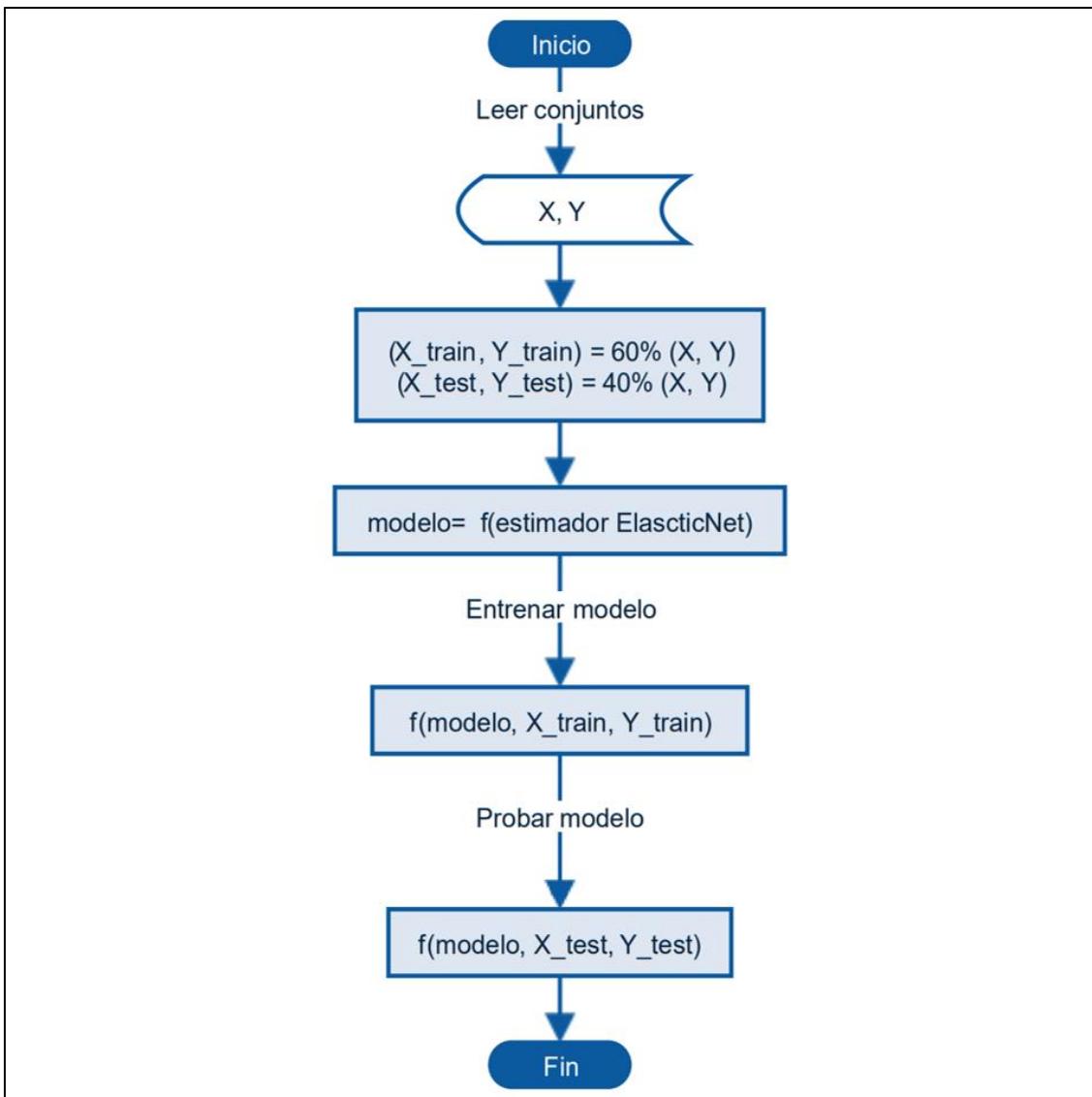


Figura 3.18: Diagrama de flujo del proceso de entrenamiento y validación del modelo.

El código correspondiente a los diagramas mostrados anteriormente se muestra en la sección de anexos correspondiente.

Los valores muestreados (datos en la hoja de cálculo) correspondientes a la máxima y mínima velocidad (rad/s) del motor definirán las restricciones del modelo entrenado para evitar valores de salida ajenos a la planta de control real:

- **Velocidad máxima:** 388 rad/s
- **Velocidad mínima:** 0 rad/s

3.1.3.2 Modelo 2: Función de transferencia:

En este caso, se importó la hoja de cálculo “motor-stepentry.xlsx”. A continuación, se muestra el proceso de identificación para obtener una función de transferencia.

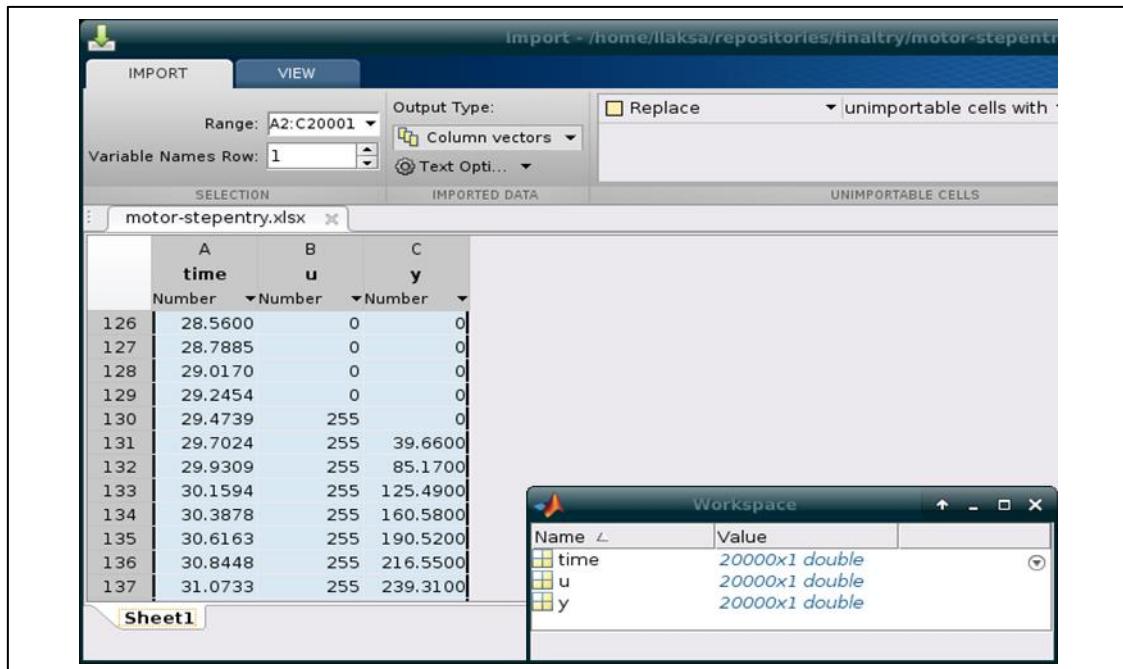


Figura 3.19: Vectores en el espacio de trabajo de MatLab.

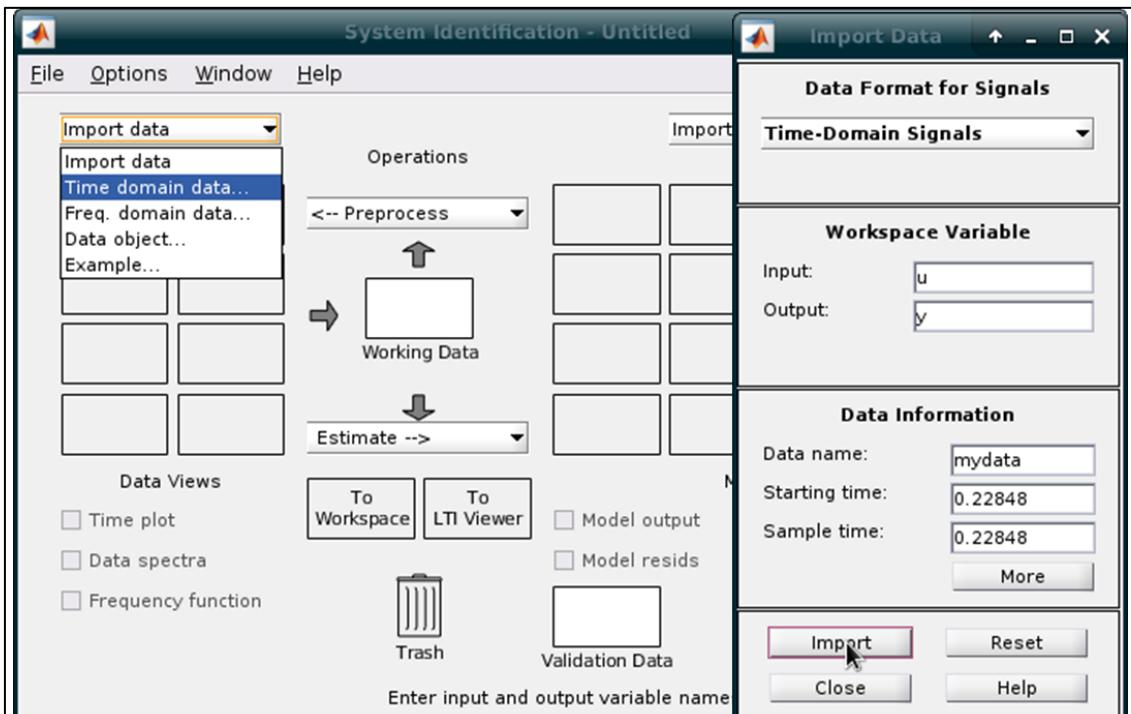


Figura 3.20: Uso de los vectores en la System Identification App.

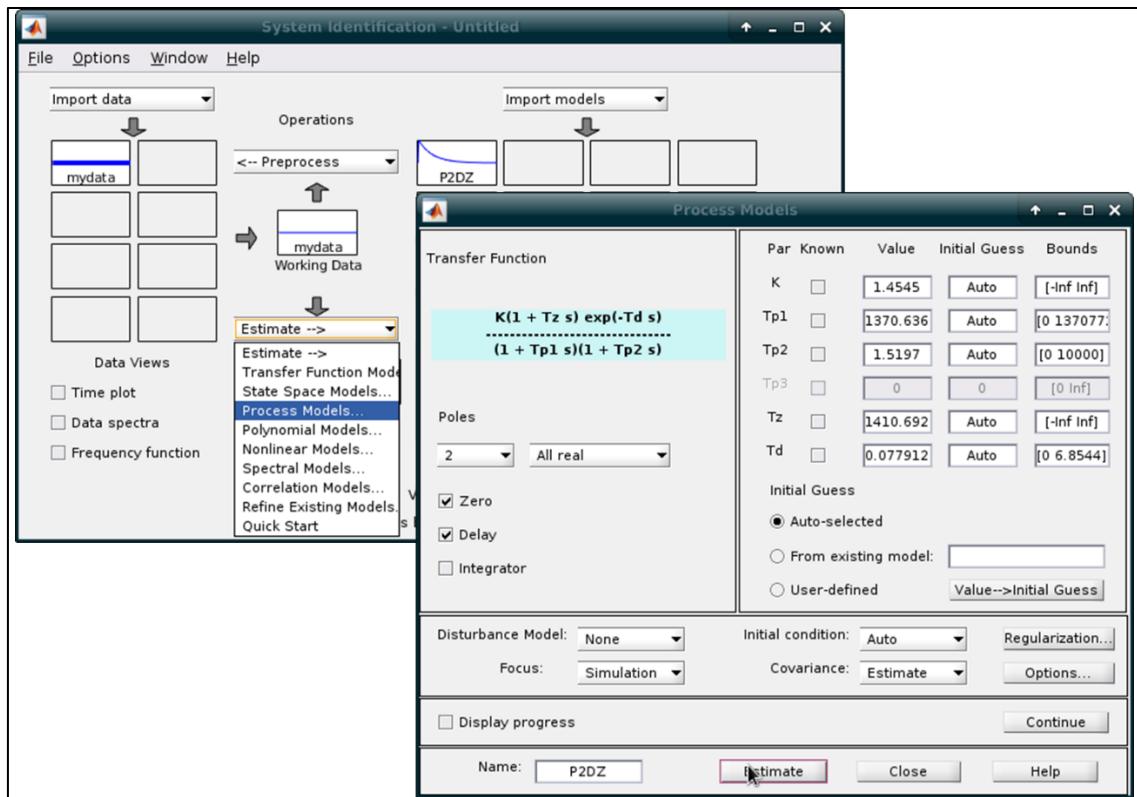
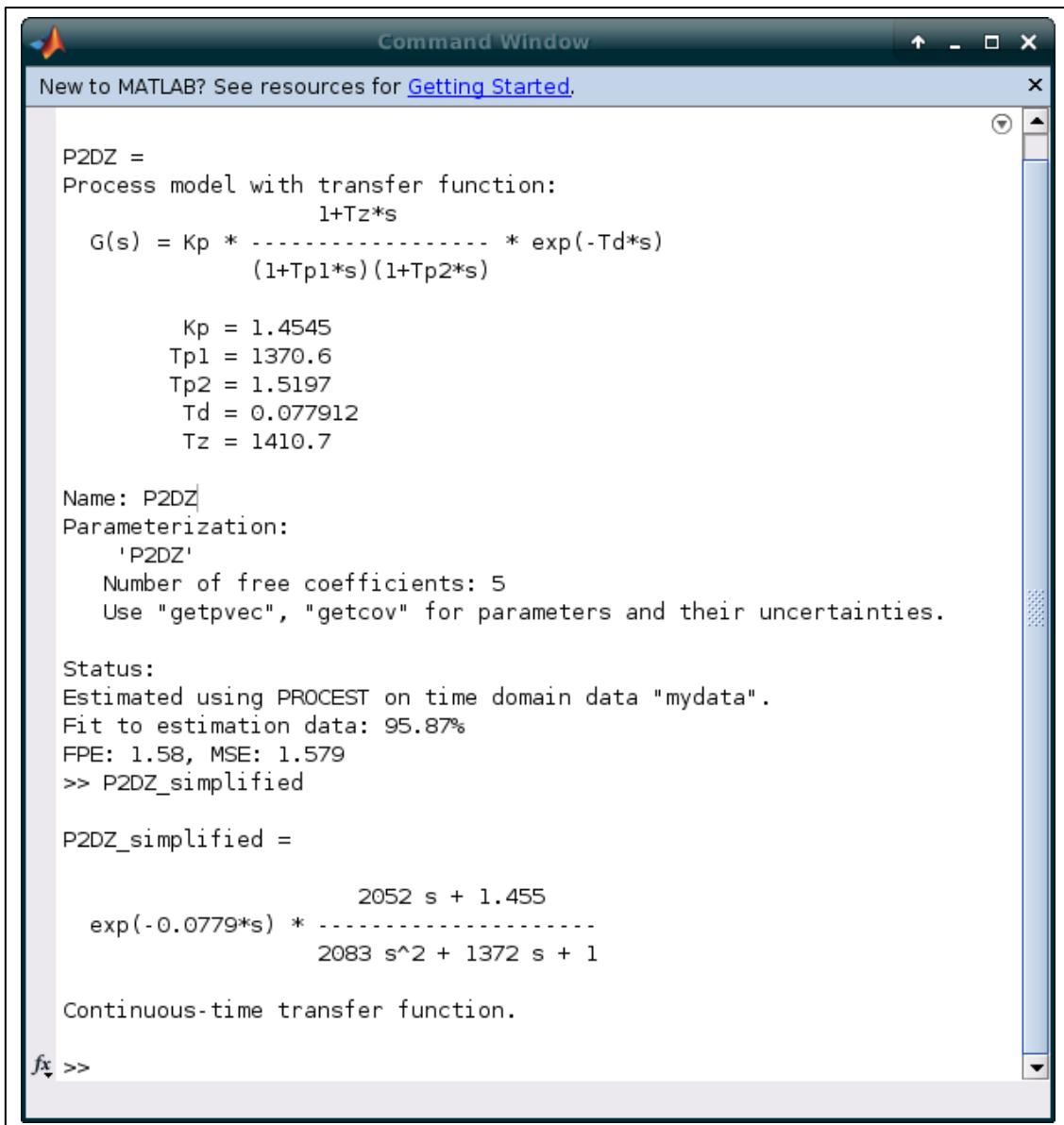


Figura 3.21: Elección de la estructura de modelo.



The screenshot shows the MATLAB Command Window with the title 'Command Window'. The window displays the following text:

```

P2DZ =
Process model with transfer function:
  1+Tz*s
G(s) = Kp * ----- * exp(-Td*s)
        (1+Tp1*s)(1+Tp2*s)

  Kp = 1.4545
  Tp1 = 1370.6
  Tp2 = 1.5197
  Td = 0.077912
  Tz = 1410.7

Name: P2DZ
Parameterization:
  'P2DZ'
  Number of free coefficients: 5
  Use "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using PROCEST on time domain data "mydata".
Fit to estimation data: 95.87%
FPE: 1.58, MSE: 1.579
>> P2DZ_simplified

P2DZ_simplified =

  2052 s + 1.455
exp(-0.0779*s) * -----
  2083 s^2 + 1372 s + 1

Continuous-time transfer function.

fx >>

```

Figura 3.22: Función de transferencia obtenida y exportada en la variable P2DZ.

3.1.4 Decodificación de un cromosoma para obtener parámetros PID, definición del desempeño del controlador digital y simulación de su accionar usando el modelo de machine learning como planta de control

Se determinó que los cromosomas (individuos) usados por los algoritmos genéticos serían cadenas binarias. A continuación, se muestra el cromosoma del que derivó el controlador PID con mejor performance:

110101110101100000110000100011000

La decodificación de un cromosoma y la obtención de los parámetros PID, se dio por medio de:

$$Kp = Kp_{min} + bin2dec(Kp^*) \times \frac{Kp_{max} - Kp_{min}}{2^{Lp} - 1}$$

$$Ki = Ki_{min} + bin2dec(Ki^*) \times \frac{Ki_{max} - Ki_{min}}{2^{Li} - 1}$$

$$Kd = Kd_{min} + bin2dec(Kd^*) \times \frac{Kd_{max} - Kd_{min}}{2^{Ld} - 1}$$

Donde:

$Kp_{min}, Ki_{min}, Kd_{min}, Kp_{max}, Ki_{max}$ y Kd_{max} son los valores mínimos y máximos que determinan los rangos de búsqueda de los parámetros PID; Kp^*, Ki^* y Kd^* son las palabras digitales asignadas a los parámetros PID y obtenidas a partir de un cromosoma; Lp, Li y Ld son las resoluciones (cantidad de bits) de las palabras digitales; y $bin2dec$ es una función que transforma un número binario a decimal.

Por ejemplo, para el cromosoma mostrado, se definieron previamente:

$$Lp = 11 \text{ bits}$$

$$Li = 11 \text{ bits}$$

$$Ld = 11 \text{ bits}$$

$$[Kp_{min}; Kp_{max}] = [0; 2]$$

$$[Ki_{min}; Ki_{max}] = [0; 2]$$

$$[Kd_{min}; Kd_{max}] = [0; 2]$$

Las resoluciones digitales de 11 bits y los rangos de búsqueda de [0; 2] determinaron una precisión de:

$$\frac{2 - 0}{2^{11} - 1} = 0.00196$$

Además, las palabras digitales Kp^*, Ki^* y Kd^* se obtuvieron del cromosoma según la Figura 3.23.

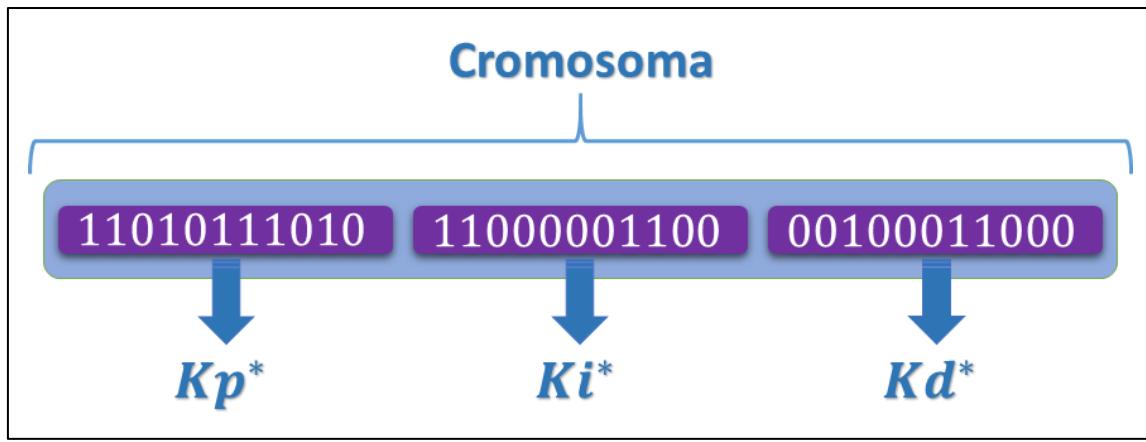


Figura 3.23: Palabras digitales obtenidas a partir de un cromosoma.

Así, se obtuvieron los siguientes parámetros PID:

$$kp = 0 + bin2dec(11010111010_2) \times \frac{2 - 0}{2^{11} - 1} = 0 + 1722 \times \frac{2 - 0}{2^{11} - 1} = 1.6824$$

$$ki = 0 + bin2dec(11000001100_2) \times \frac{2 - 0}{2^{11} - 1} = 0 + 1548 \times \frac{2 - 0}{2^{11} - 1} = 1.5124$$

$$kd = 0 + bin2dec(00100011000_2) \times \frac{2 - 0}{2^{11} - 1} = 0 + 280 \times \frac{2 - 0}{2^{11} - 1} = 0.2735$$

Por otro lado, para la simulación de la acción del controlador, se implementó un bucle que ejecuta mil veces la acción de un controlador PID digital sobre el modelo de machine learning. Puesto que el tiempo de muestreo elegido fue de 0.22848s (7 ciclos del timer del microcontrolador), las mil iteraciones evalúan el desempeño del controlador durante $1000 \times 0.22848s = 228.48s$.

Por último, se definió la evaluación de desempeño en base a un acumulador del tiempo multiplicado por el valor absoluto del error (derivado de la integral del tiempo multiplicado por el valor absoluto del error):

$$\sum_{k=1}^{1000} (k)(|e_k|)$$

Donde k es el número de iteración y e_k es el error en la iteración k .

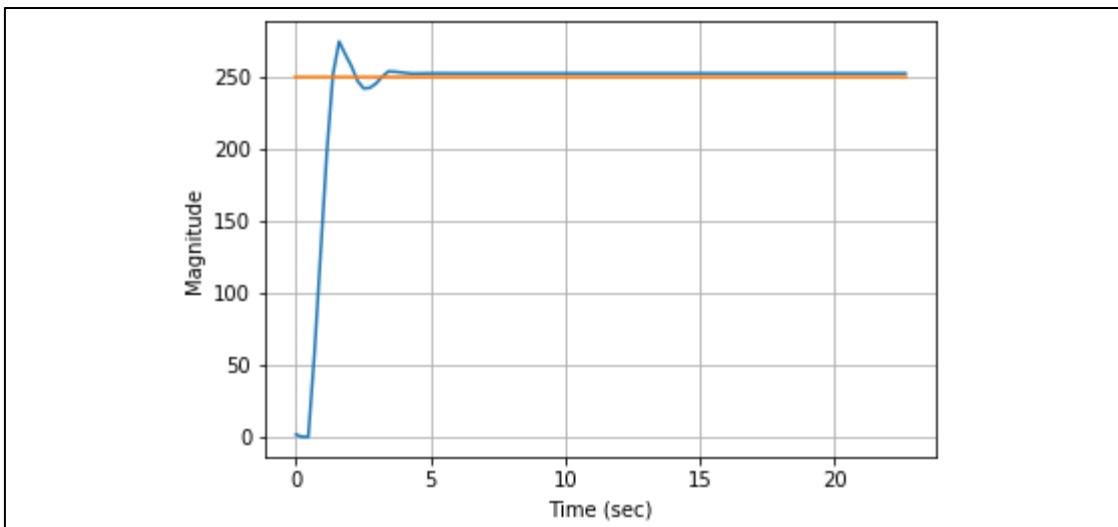


Figura 3.24: Simulación del desempeño del controlador para un setpoint de 250 rad/s.

A continuación, se muestra el diagrama de flujo de los procesos descritos anteriormente:

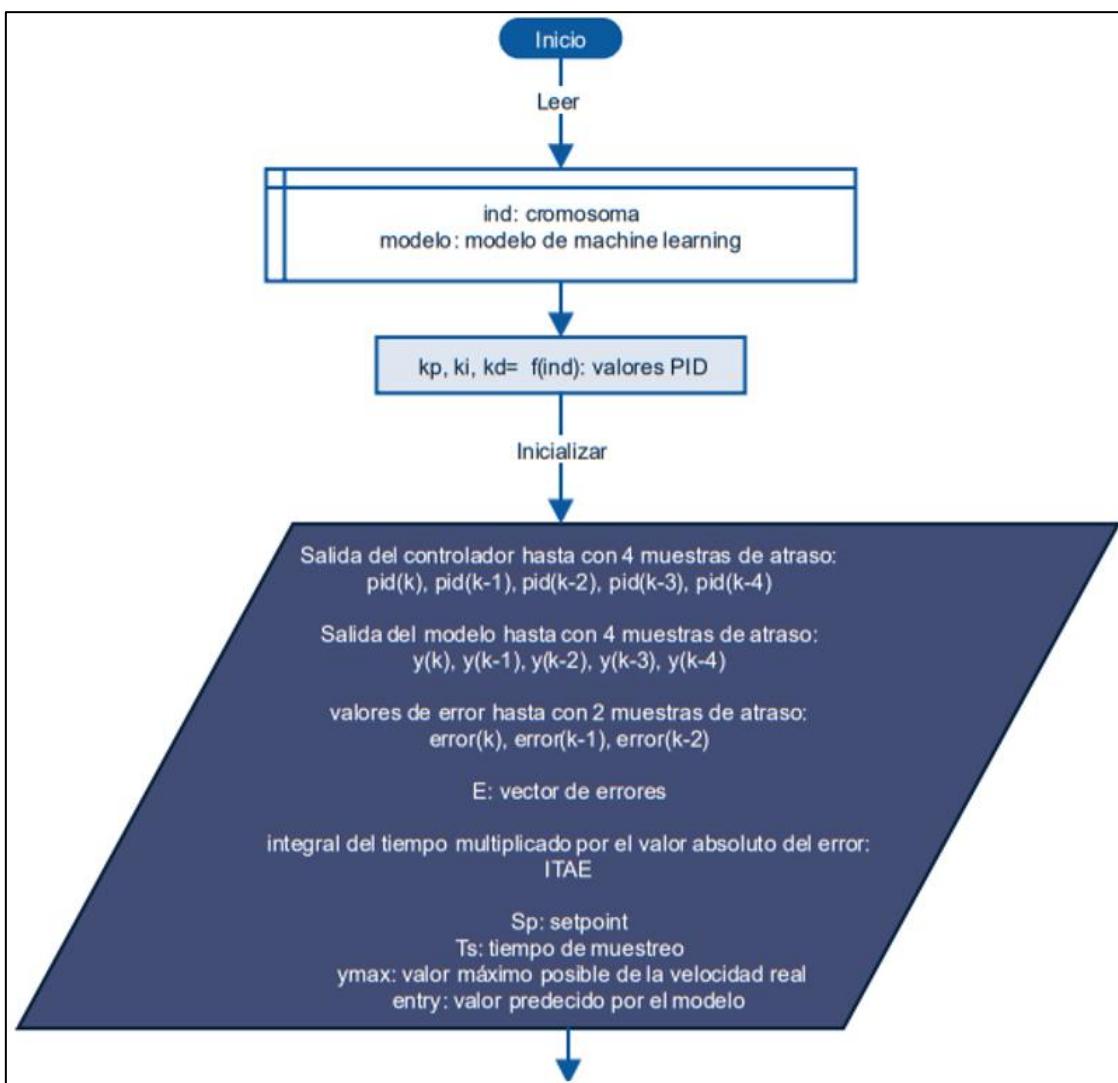


Figura 3.25: Decodificación del cromosoma y simulación y desempeño del controlador.

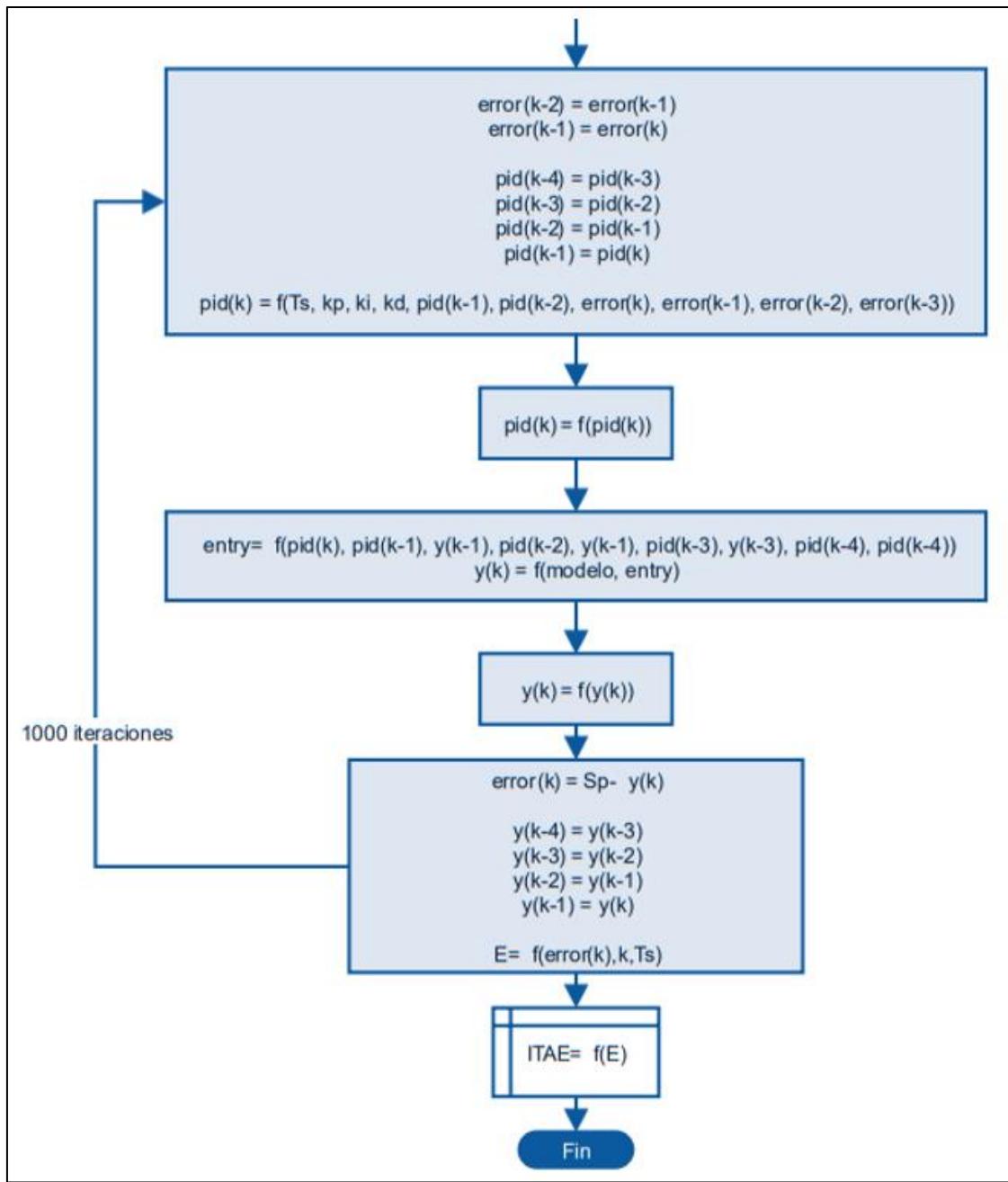


Figura 3.26: Decodificación del cromosoma y simulación y desempeño del controlador. (continuación).

El código correspondiente a los diagramas mostrados anteriormente se muestra en la sección de anexos correspondiente.

3.1.5 Sintonización, en base a algoritmos genéticos, de un controlador PID digital haciendo uso de la simulación realizada

Se definieron los parámetros propios del algoritmo genético: recombinación en 0.7, mutación en 0.1, tamaño de la población en 40 individuos y el número de generaciones en 40.

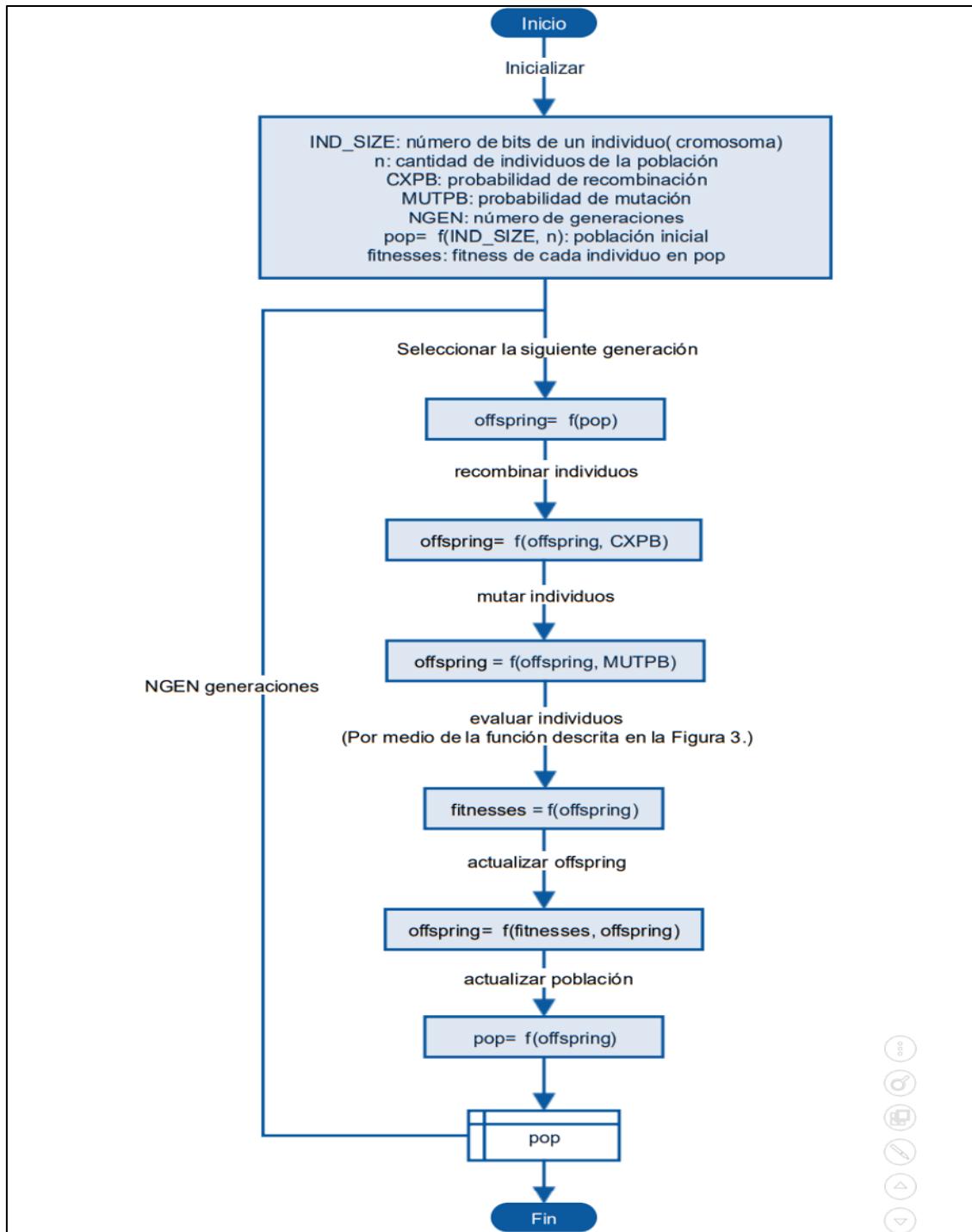


Figura 3.27: Sintonización del controlador mediante algoritmos genéticos.

El código correspondiente a los diagramas mostrados anteriormente se muestra en la sección de anexos correspondiente.

ITAE	Kp	Ki	Kd
1813.59779	1.20234604	2	0
2150.44882	1.10459433	1.69501466	0.10948192
1825.63799	1.16715543	1.93743891	0
2007.01817	1.12805474	1.93548387	0.06256109
1868.11738	1.16324536	1.71065494	0
1842.01106	1.06940371	1.87487781	0.00195503
1842.01106	1.1945259	1.87487781	0.00195503
1826.012	1.0342131	1.93548387	0
1827.1332	1.23753666	1.92961877	0
1813.97616	1.10068426	1.99804497	0
1850.34669	1.22580645	1.99804497	0.01564027
1833.09226	1.10068426	1.89833822	0
130470.017	1.14565005	1.92766373	0.50244379
1827.02957	1.20625611	1.98240469	0.00391007
1825.63799	1.10459433	1.93743891	0
1826.012	1.13196481	1.93548387	0
1853.10013	1.19061584	2	1.7595E+15
1826.7596	1.17106549	1.9315738	0
1831.97759	1.06940371	1.90420332	0
1831.97759	1.06940371	1.90420332	0
1815.11045	1.23753666	1.99217986	0
1842.01106	1.06940371	1.87487781	0.00195503
1855.29102	1.20625611	1.86705767	0.00782014
1815.11045	1.23753666	1.99217986	0
1910.9572	1.07722385	1.49951124	0.01564027
1910.9572	1.07722385	1.49951124	0.01564027
1815.11045	1.23753666	1.99217986	0
1851.06904	1.1085044	1.99217986	0.01564027
1855.12398	1.23362659	1.96089932	0.01564027
1855.12398	1.1085044	1.96089932	0.01564027
1864.61028	1.19843597	1.74975562	0.00195503
1853.10013	1.19843597	2	1.7595E+15
1816.62094	1.21798631	1.98435973	0
1826.012	1.23753666	1.93548387	0
1815.11045	1.10068426	1.99217986	0
2637.46943	1.20625611	0.60606061	0
1803.97616	1.68246214	1.51245725	0.27357108
1803.97616	1.68246214	1.51245725	0.27357108
1805.11045	1.58866634	1.43234001	0.00781632
1809.02957	1.20234604	0.99902248	0.12512219

Tabla 3.3: Población obtenida después de 40 generaciones.

3.1.6 Sintonización de un controlador PID digital en base al método de Ziegler-Nichols y otro usando el PID Tuner de MatLab

3.1.6.1 Sintonización basada en el primer método de Ziegler-Nichols

Se implementó la función de transferencia en un lazo abierto y un lazo de control realimentado para evaluar su respuesta.

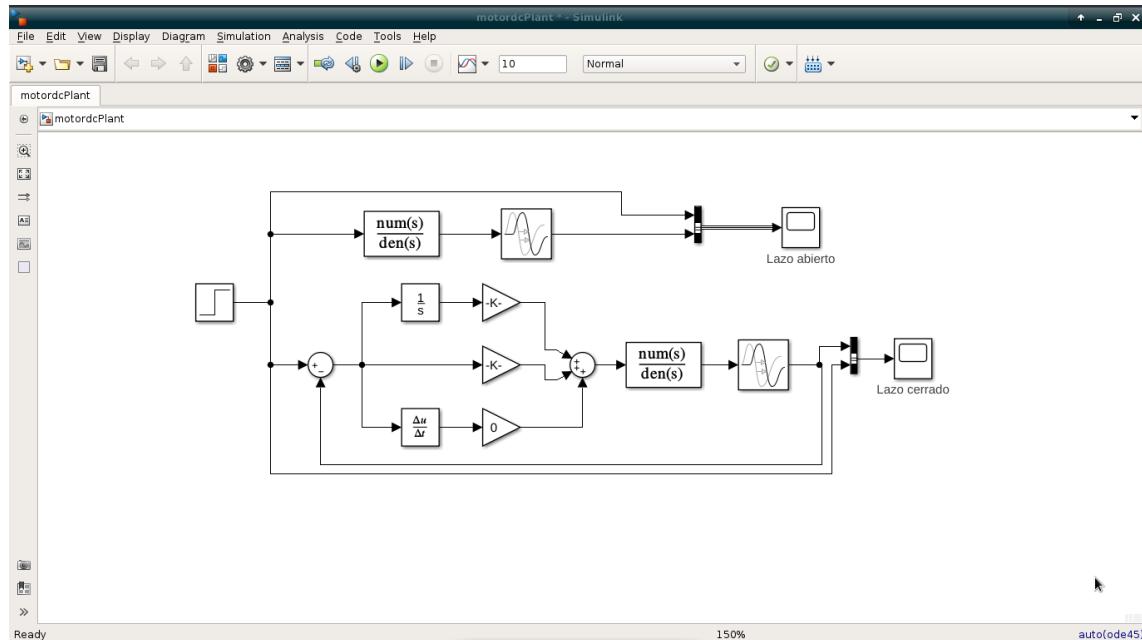


Figura 3.28: Diagramas de bloque en Simulink.

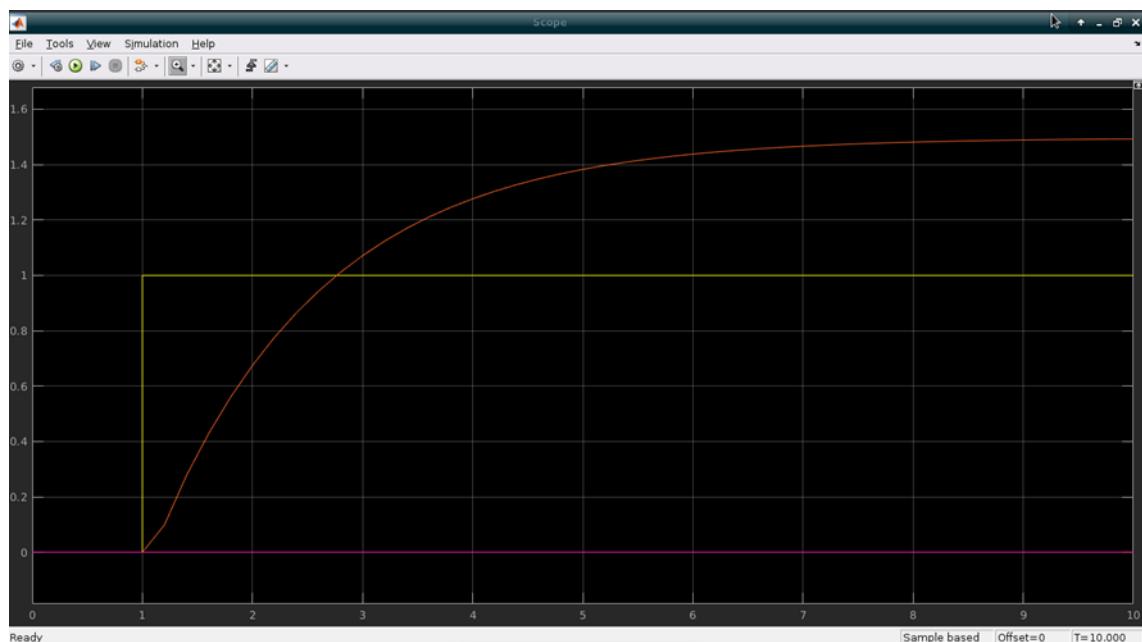


Figura 3.29: Simulación de la respuesta de la función de transferencia en lazo abierto frente a una entrada tipo escalón.

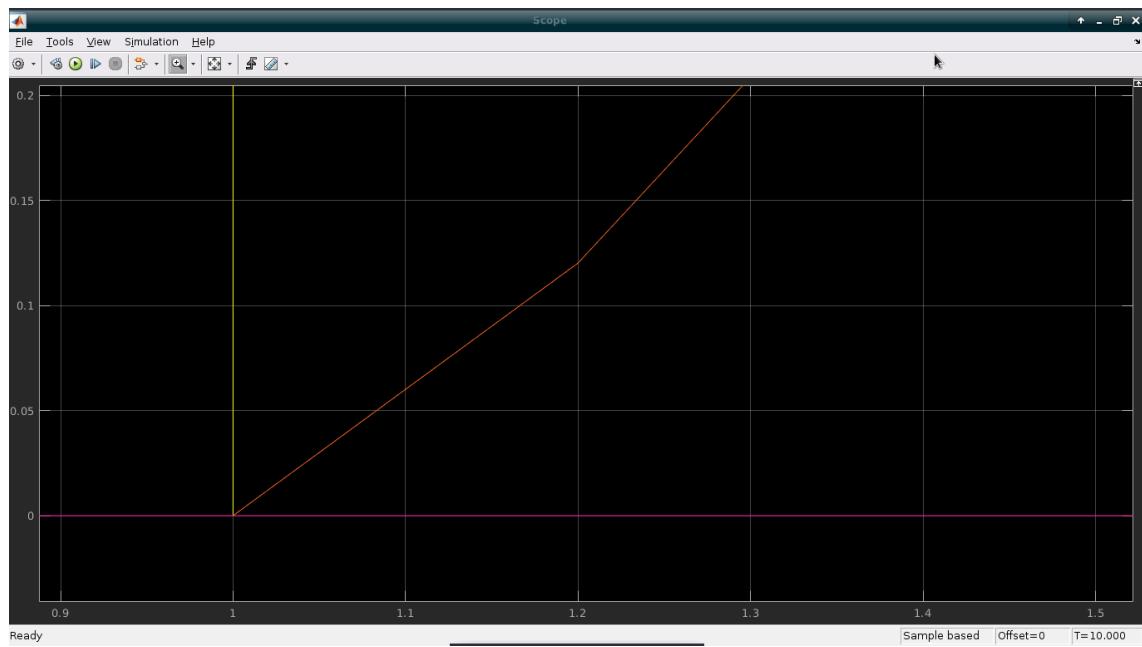


Figura 3.30: Acercamiento al punto de inflexión para determinar los parámetros T y L requeridos por el primer método de Ziegler-Nichols.

Se determinaron los siguientes parámetros T y L :

$$L = 0.1167$$

$$T = 0.4234:$$

```

>> L = 0.1167;
>> T = 0.4234;
>> ZN_kp = 1.2*T/L;
>> ZN_ki = ZN_kp/(2*L);
>> ZN_kd = ZN_kp*(0.5*L);
>> pid = [ZN_kp ZN_ki ZN_kd]

pid =
    4.3537    18.6535    0.2540

```

Figura 3.31: Parámetros PID obtenidos por el primer método de Ziegler-Nichols.

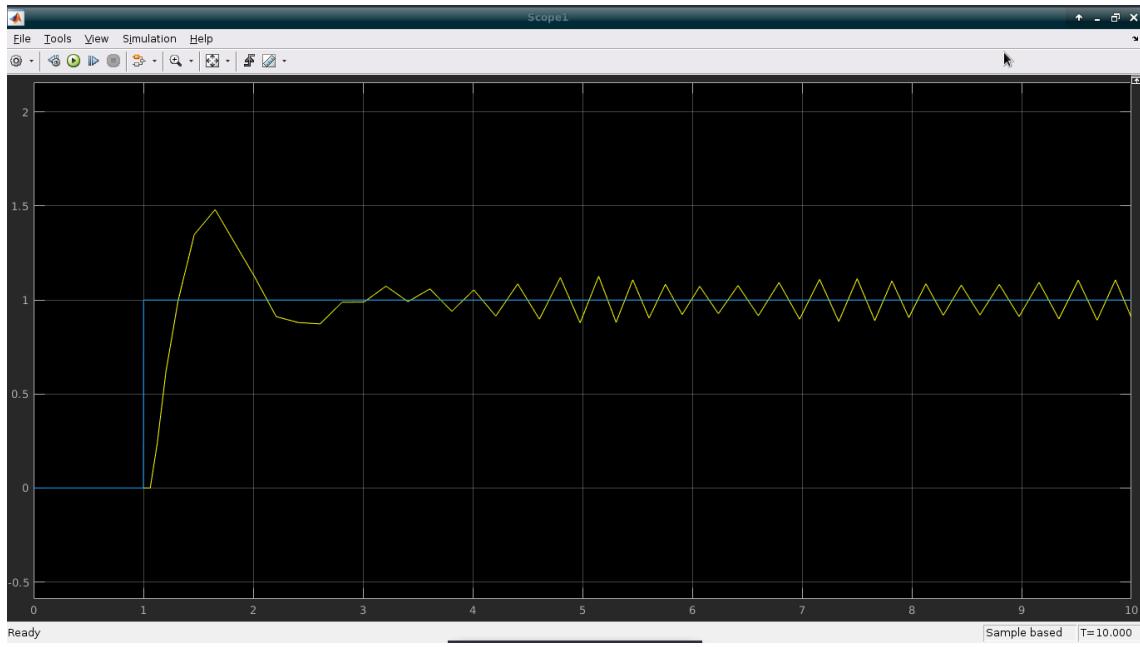


Figura 3.32: Simulación de la respuesta de la función de transferencia en lazo cerrado frente a una entrada tipo escalón.

3.1.6.2 Sintonización basada en el PID Tuner de MatLab

Se importó la planta P2DZ al PID Tuner App de MatLab.

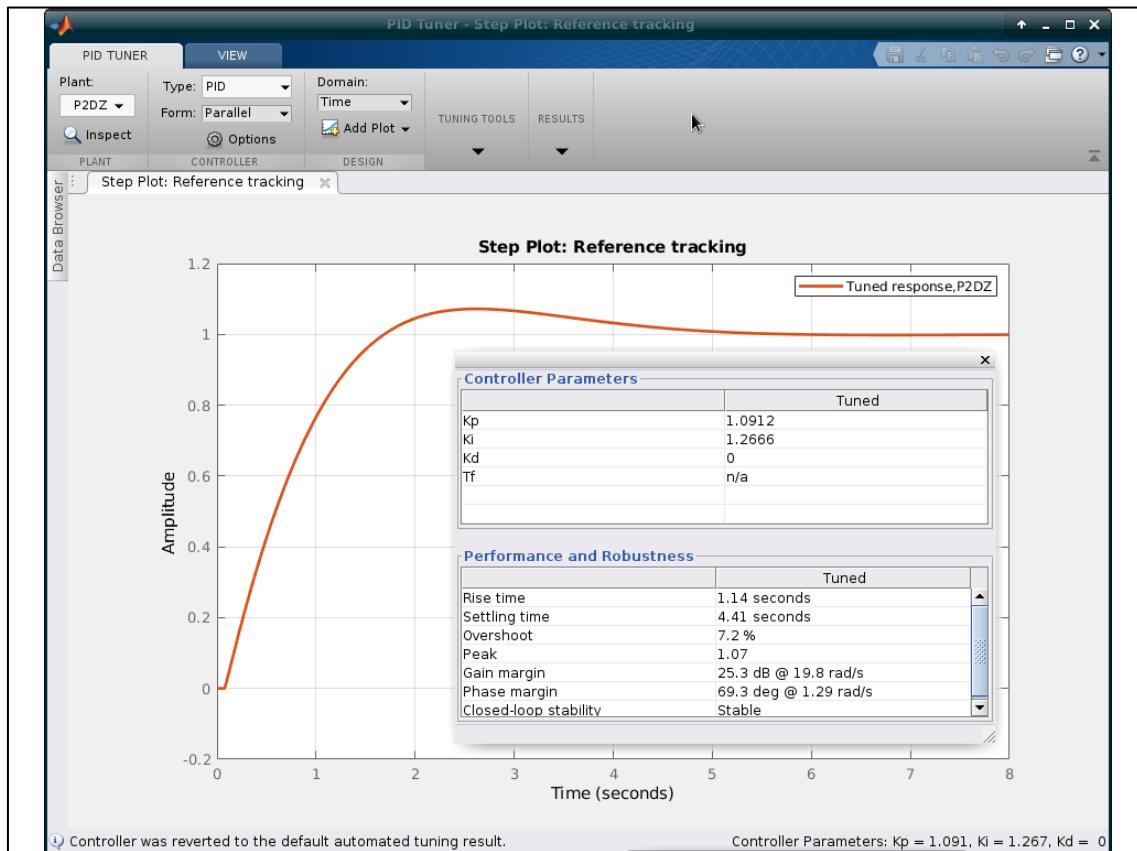


Figura 3.33: Sintonización por medio del PID Tuner App.

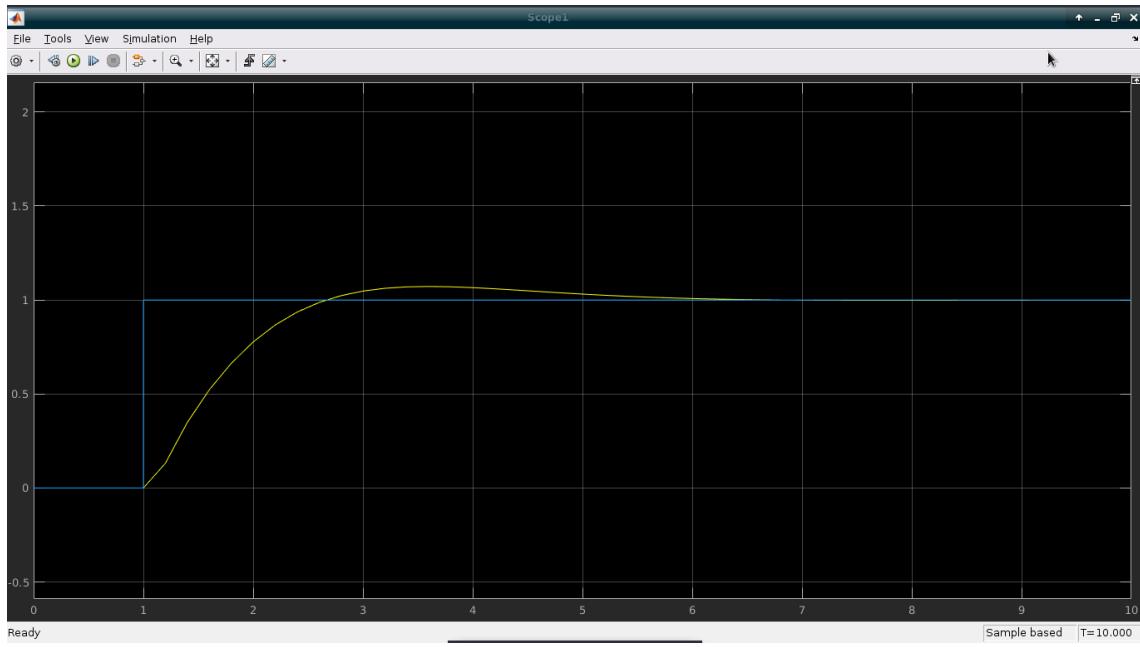


Figura 3.34: Simulación de la respuesta de la función de transferencia en lazo cerrado frente a una entrada tipo escalón.

3.2 Etapa de control

En esta etapa, la placa Arduino funcionará como elemento controlador, ya con sus parámetros PID determinados.

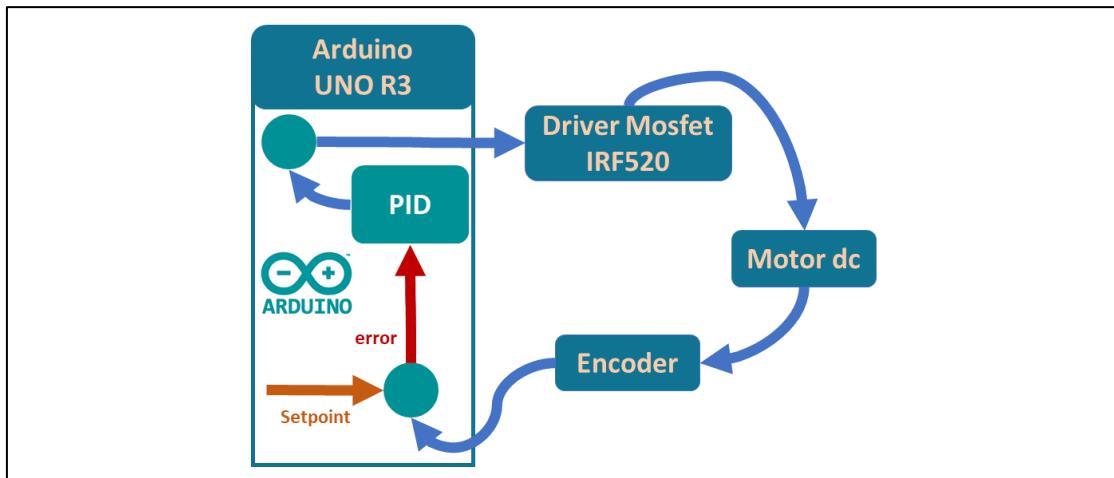


Figura 3.35: Esquema de la disposición de elementos de software y hardware.

A continuación, se muestran los diagramas de flujo de la implementación del controlador PID digital en la placa Arduino UNO R3.

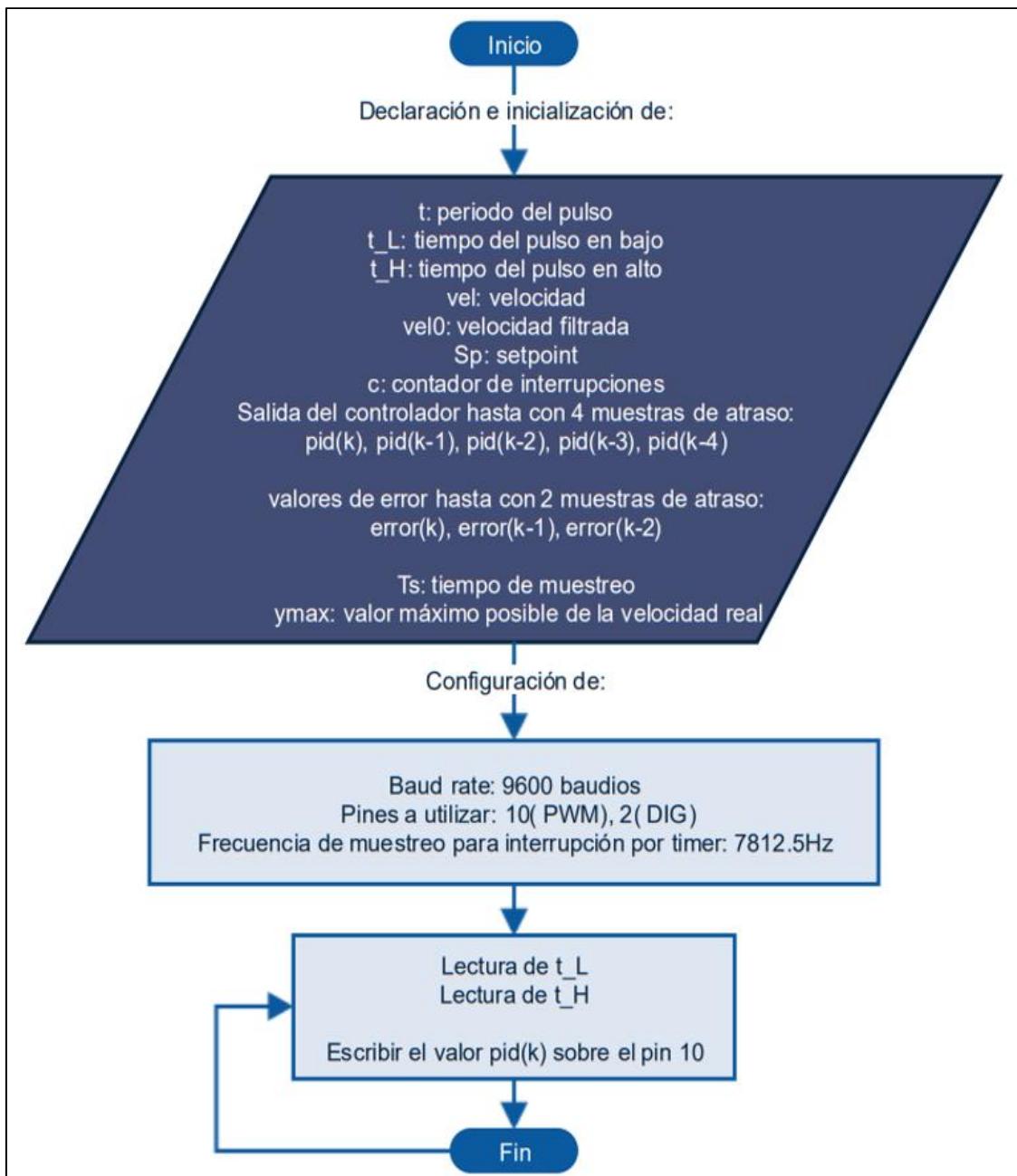


Figura 3.36: Procesos ligados al microprocesador del uC.

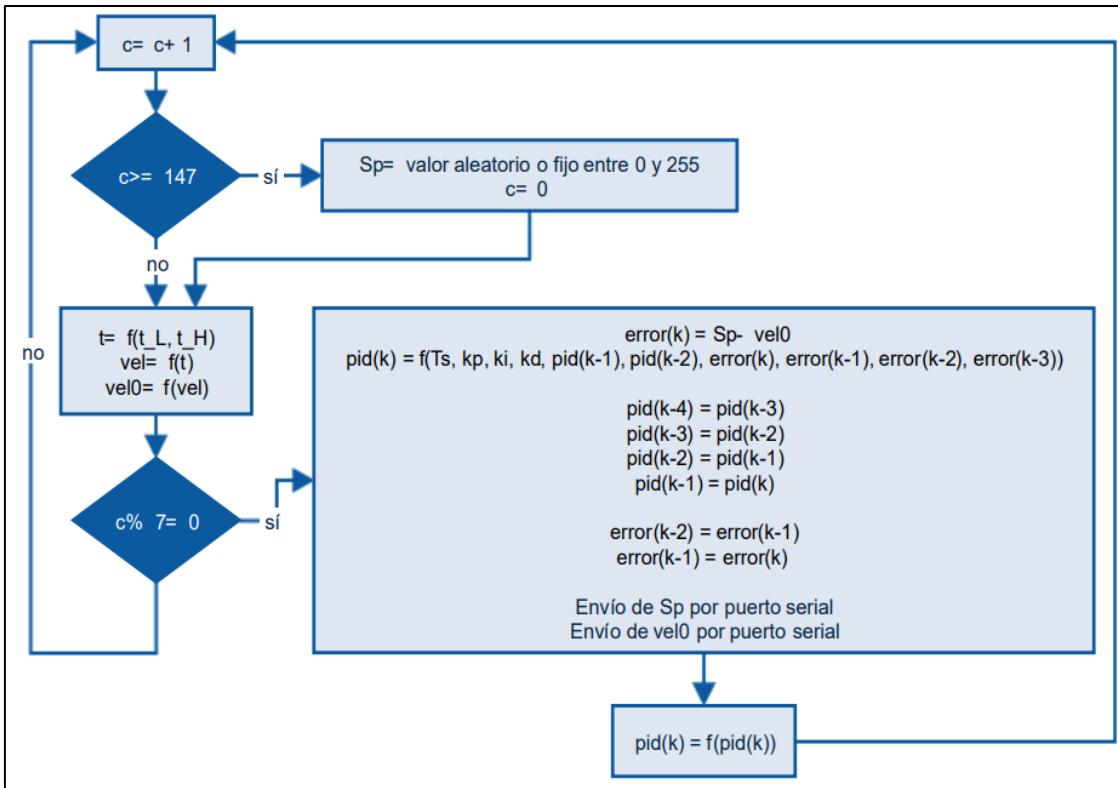


Figura 3.37: Procesos ligados al timer del uC.

El código correspondiente a los diagramas mostrados anteriormente se muestra en la sección de anexos correspondiente.

3.2.1 Prueba del desempeño de los controladores en el sistema de control de velocidad

Se probaron distintas entradas al sistema de control y se observaron las siguientes respuestas de la planta de control real.

3.2.1.1 Controlador obtenido por medio de algoritmos genéticos y machine learning

Como se observa en la Figura 3.38, la respuesta del sistema (velocidad del motor) se estabiliza antes de las cien iteraciones del controlador PID. Se escogió controlador con el menor valor ITAE de la tabla 3.3 ($K_p=1.6824$, $K_i=1.5124$, $K_d=0.2735$).

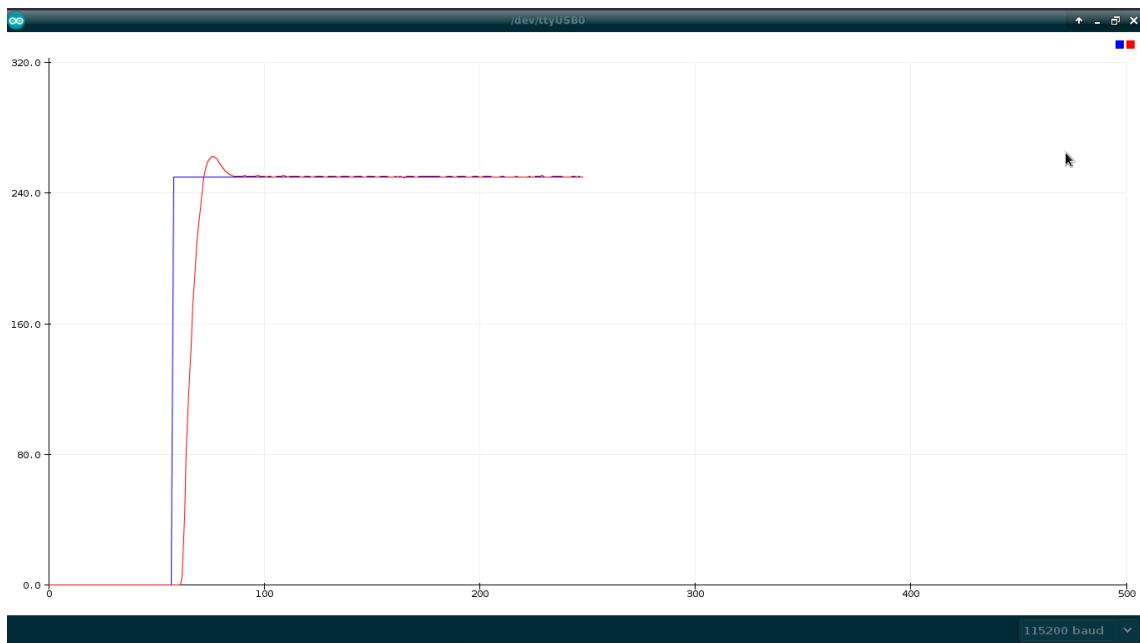


Figura 3.38: Serial Plotter Arduino. Respuesta del sistema frente a una entrada tipo escalón.

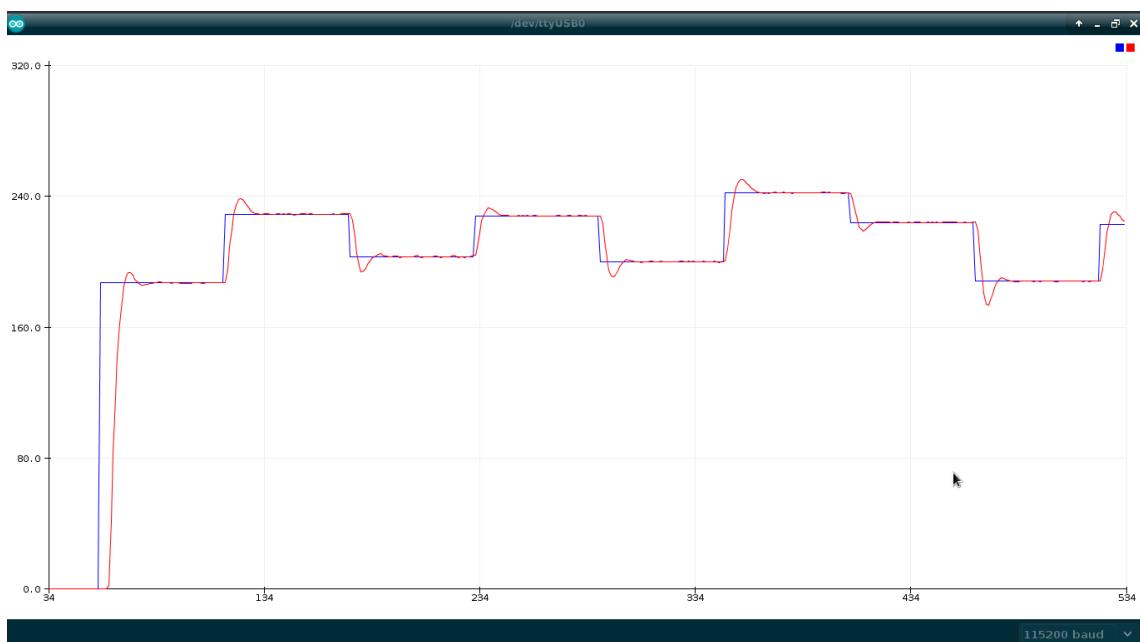


Figura 3.39: Serial Plotter Arduino. Respuesta del sistema frente a una entrada tipo escalón con amplitud variable.

3.2.1.2 Controlador obtenido por medio del PID Tuner App

Como se observa en la Figura 3.40, la respuesta del sistema (velocidad del motor) se estabiliza después de las cien iteraciones del controlador PID.

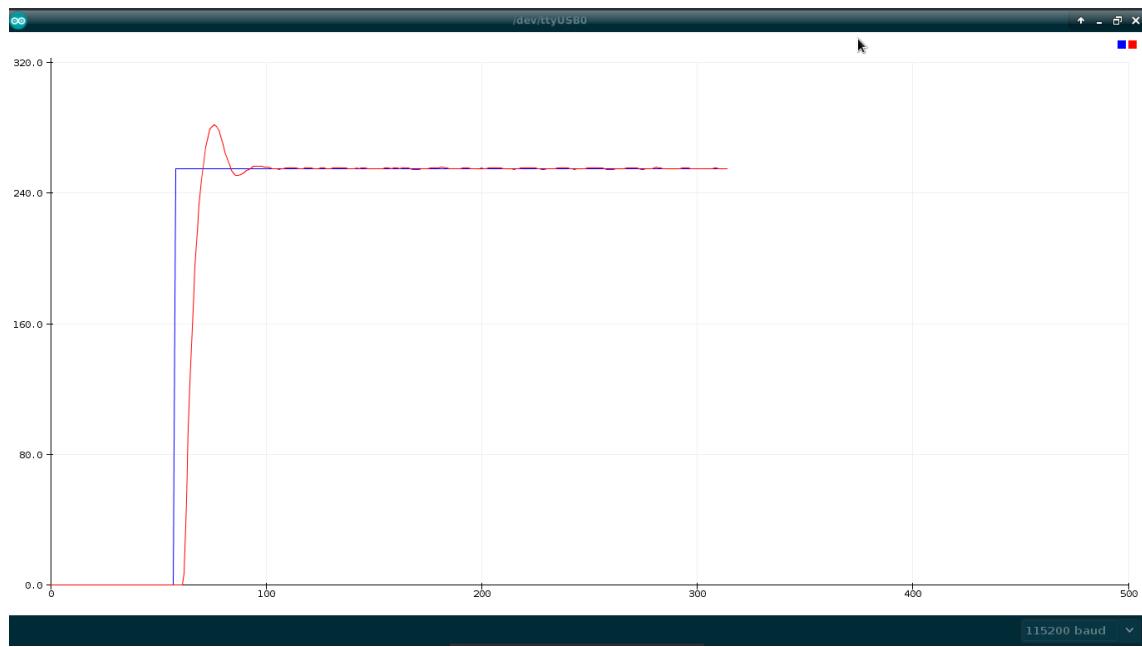


Figura 3.40: Serial Plotter Arduino. Respuesta del sistema frente a una entrada tipo escalón.

Como se observa en la Figura 3.41, la respuesta del sistema (velocidad del motor) presenta mayores oscilaciones que el controlador PID mostrado anteriormente.

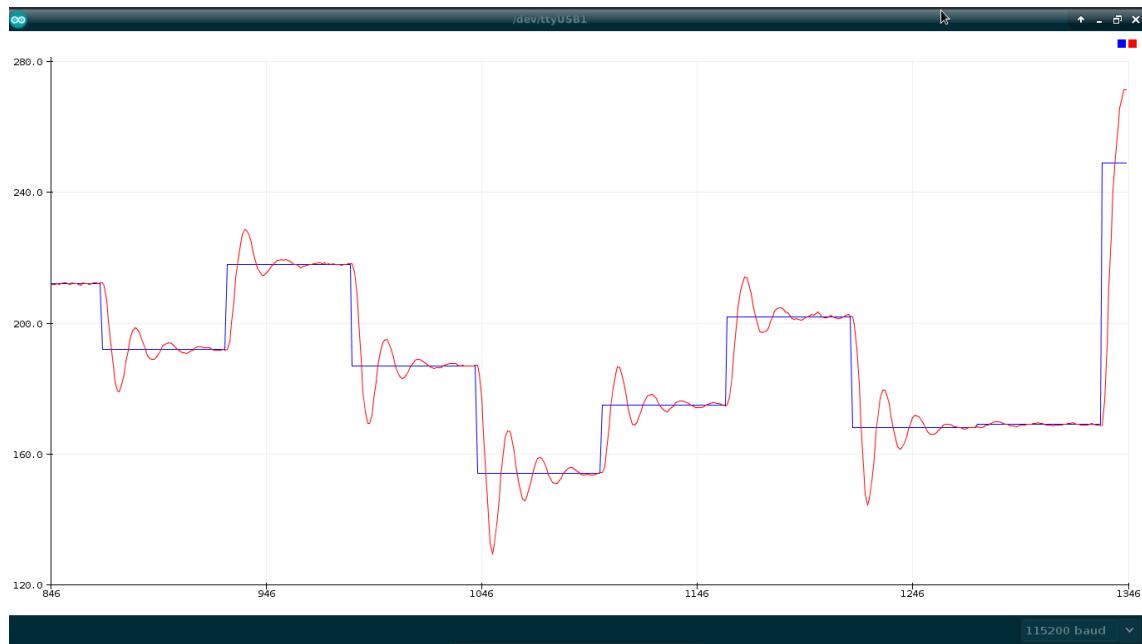


Figura 3.41: Serial Plotter Arduino. Respuesta del sistema frente a una entrada tipo escalón con amplitud variable.

3.2.1.3 Controlador obtenido por medio del primer método de Ziegler-Nichols

Como se observa en la Figura 3.42, la respuesta del sistema (velocidad del motor) no se estabiliza y muestra oscilaciones.

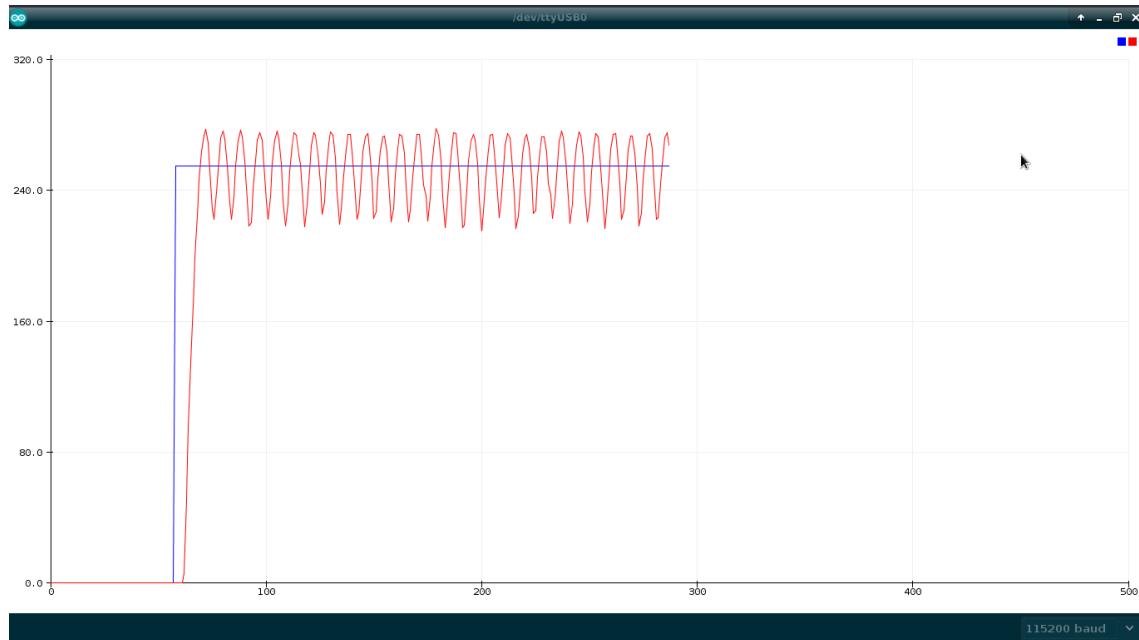


Figura 3.42: Serial Plotter Arduino. Respuesta del sistema frente a una entrada tipo escalón.

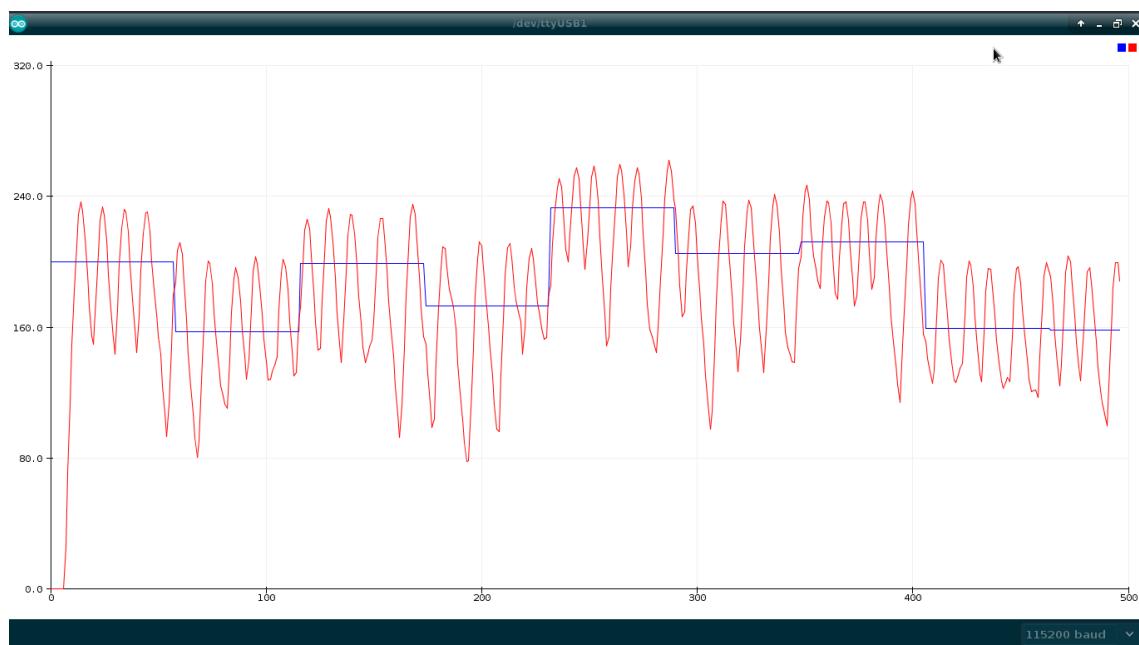


Figura 3.43: Serial Plotter Arduino. Respuesta del sistema frente a una entrada tipo escalón con amplitud variable.

3.2.1.4 Comparación de controladores

Se realizó un análisis conjunto del desempeño de los controladores sobre el sistema de control. Adicionalmente, en dicho análisis fueron considerados el segundo y el tercer individuo con menor factor ITAE de la tabla 3.3. Obteniéndose:

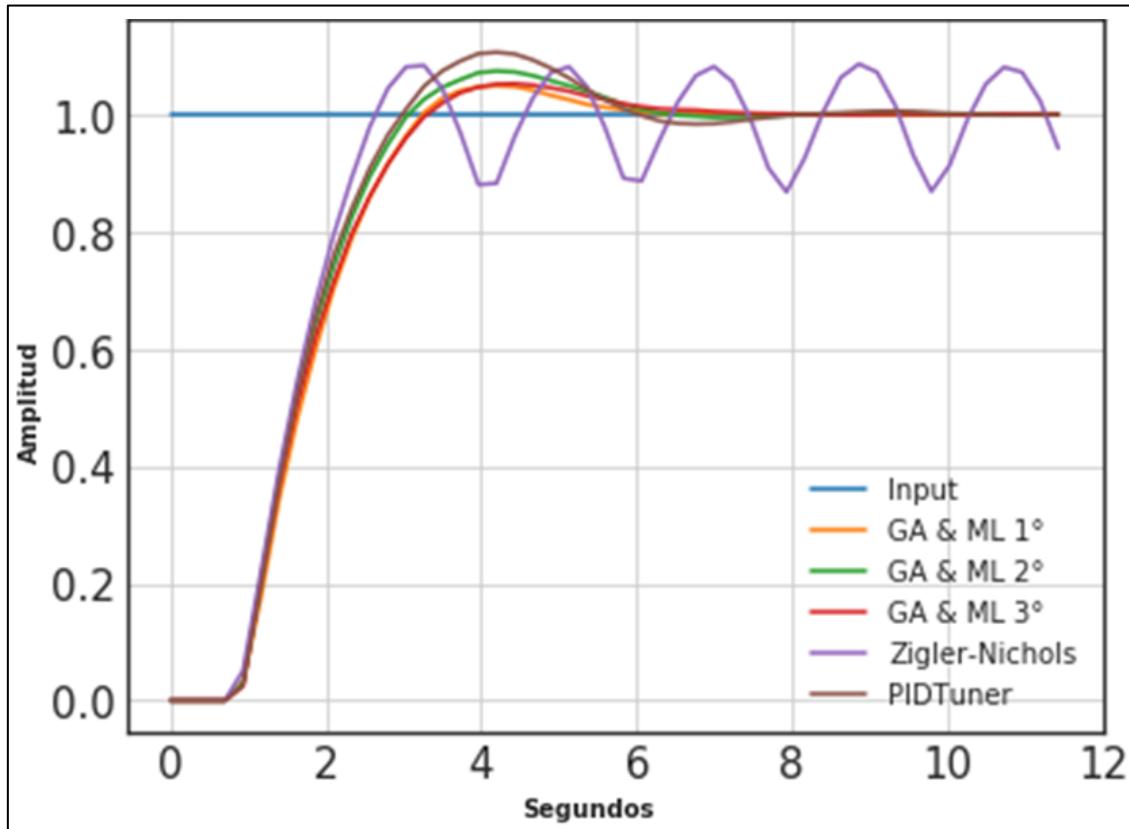


Figura 3.44: Análisis conjunto.

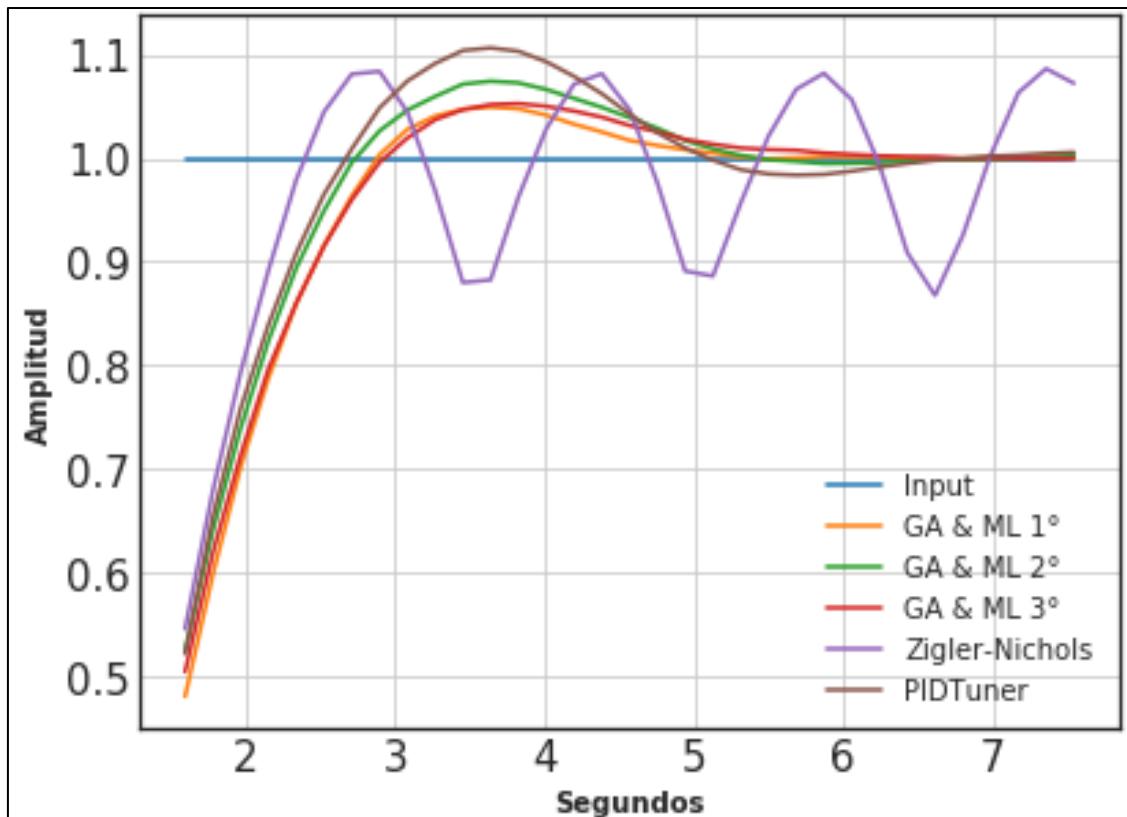


Figura 3.45: Análisis conjunto (Zoom).

	td	tr	M_p	tp	ts
<i>Genetic A & ML 1</i>	1.82784	3.19872	1.04904	4.11264	5.25504
<i>Genetic A & ML 2</i>	1.59936	3.19872	1.07403922	4.11264	5.712
<i>Genetic A & ML 3</i>	1.59936	3.4272	1.05252	4.34112	5.712
<i>Ziegler-Nichols</i>	-	-	1.09560784	-	-
<i>PIDTuner</i>	1.59936	2.97024	1.10615686	4.11264	5.712

Tabla 3.4: Características de la respuesta transitoria frente a un escalón unitario.

Donde:

td: tiempo en que la salida alcanza el 50% del valor final.

tr: tiempo en que la salida alcanza el 100% del valor final.

M_p: valor del pico máximo de la salida.

T_p: tiempo en el que ocurre el pico máximo.

ts: tiempo de establecimiento (error de $\pm 2\%$).

4 CAPÍTULO IV: DISCUSIÓN

4.1 El sistema de control de velocidad implementado

- Debido al disco acoplado mecánicamente, el encoder envió 4 pulsos por vuelta. Aunque dicha cantidad resultó suficiente para implementar un sistema de control de velocidad, el mismo disco serviría para controlar la posición angular del eje, pero solo en desplazamientos de 45° .
- Debido al microcontrolador ATmega328 (utilizado por la placa Arduino UNO R3), los tiempos de interrupción que se pueden implementar en el reloj (timer) ya están definidos. Por lo tanto, los posibles tiempos de muestreo, ya sea para fines de adquisición de datos o control, solo podrán ser múltiplos del tiempo de interrupción seleccionado. Por ejemplo, en este trabajo, el tiempo de interrupción seleccionado para el reloj del microcontrolador fue de 32.64 ms.
- La no linealidad en el comportamiento del motor dc a bajas velocidades, no permite realizar un buen control a velocidades menores que 70 rad/s, evidenciando la necesidad de cambiar o mejorar la estrategia de control.
- El sistema estuvo sometido al ruido ocasionado por algunos elementos del sistema de control como la fuente de alimentación y el motor, esto puede afectar el trabajo del controlador PID, sobre todo respecto de la ganancia integral debido a la acumulación de los valores del error que el ruido genera.

4.2 Las señales de entrada

- El sistema de control estudiado no estuvo sujeto a entradas que cambien de forma gradual en un determinado tiempo, ni tampoco a entradas de choque; solo estuvo sometido a cambios repentinos en la entrada. En caso contrario, la señal de entrada no sólo debería ser de tipo escalón, puesto que no se obtendría un conjunto de datos que refleje de forma exitosa el comportamiento del sistema.
- Una señal de entrada como la mostrada en la Figura 3.3 es suficiente para obtener una función de transferencia, pero no para obtener un modelo de machine learning, puesto que en este último caso el modelo corresponde a un tiempo discreto y necesita aprender el comportamiento del sistema en el estado no estacionario de la mayor cantidad de posibles entradas.

- Ya que los datos obtenidos en el proceso de muestreo digital corresponden a un tiempo de muestreo determinado, el modelo de machine learning que se entrenó con los datos mencionados, se podría entender prácticamente como un modelo de la planta de control en tiempo discreto que corresponderá a un igual tiempo de muestreo.

4.3 La identificación del motor dc

- En el caso de identificación del sistema por medio de un modelo de machine learning, los parámetros para conformar los conjuntos de entrada y prueba fueron seleccionados partiendo de la consideración de que las plantas de control de velocidad de motores dc, como el utilizado en este trabajo, son a menudo de segundo o tercer orden en el dominio discreto al aplicar métodos tradicionales de identificación de sistemas. Esto permitió inducir que el modelamiento de un sistema depende de estados actuales y atrasados, en este caso, posiblemente hasta en tres muestras. Para entrenar el modelo de machine learning, se halló una mejor performance usando desde estados actuales hasta atrasados en cuatro muestras.
- Hay un rango de valores entre los que se puede escoger el tiempo de muestreo para una planta de control; sin embargo, mientras el tiempo de muestreo sea mayor, la cantidad de parámetros requeridos en la identificación de un sistema por medio de un modelo de machine learning será menor.
- El tipo de modelo de machine learning “Elastic Net” fue elegido debido a que los parámetros del conjunto de entrada están correlacionados unos con otros; una característica claramente presente en un sistema dinámico.
- En la identificación por medio de una función de transferencia, se obtuvo un modelo confiable puesto que mostró un alto grado de ajuste; sin embargo, el comportamiento no lineal de un sistema dinámico no puede ser captado por esta estructura de modelo.

4.4 Simulación de la acción de control

- La simulación de la ejecución del controlador PID digital se implementó correspondiendo al tiempo de muestreo de los datos con el que fue entrenado el

modelo de machine learning. Este tiempo de muestreo tiene restricciones adicionales que debe cumplir, definidas en la referencia teórica 1.75.

- El resultado de la evaluación de desempeño del controlador dado por la integral del tiempo multiplicado por el valor absoluto del error, es un criterio puramente cuantitativo con el cual no se puede caracterizar al funcionamiento del controlador en base a parámetros cualitativos muy ligados al proceso o a los elementos físicos del sistema de control.

4.5 Sintonización del controlador PID por medio de algoritmos genéticos y el modelo de machine learning

- Las probabilidades de mutación y recombinación se definieron para que el algoritmo genético permita que la gran mayoría de los individuos de una generación sea una composición mejorada de la generación anterior; pero dejando abierta la posibilidad de obtener individuos mejorados radicalmente debido a variaciones aleatorias en sus características. Los valores escogidos para el tamaño de la población y el número de generaciones permiten al algoritmo genético recorrer por todo el rango definido de forma eficiente.
- La cantidad de bits de cada ganancia se eligió para tener un amplio número de cromosomas (individuos) de búsqueda para el algoritmo genético dentro del rango dado.
- Como se observa en la tabla 3.3, hay individuos con un alto factor ITAE. A pesar de que la población corresponde a la última generación, esta diversidad de individuos es posible debido al valor elegido de la probabilidad de mutación.

4.6 Sintonización de un controlador PID por medio del método de Ziegler-Nichols y otro a través del PID Tuner App

- La función de transferencia sometida a una entrada tipo escalón, mostró una respuesta adecuada para sintonizar el sistema usando el primer método de Ziegler-Nichols. Sin embargo; el cálculo correcto de los parámetros que requiere este método por medio de una simulación, estará limitado por las capacidades del software utilizado.

- El uso del PID Tuner App permite obtener un controlador equilibrado entre robustez y velocidad de respuesta; sin embargo, estos parámetros pueden ser manipulados por el usuario de acuerdo a sus necesidades y las limitaciones del funcionamiento propio del sistema dinámico que se quiere controlar. En el caso del controlador obtenido por este método, variar la robustez o velocidad de respuesta no mejoraron la respuesta de la planta controlada.
- La simulación del sistema de control para ambos controladores mostró que el controlador obtenido por Ziegler-Nichols originaba una respuesta osculatoria de la planta de control, mientras que el controlador obtenido por el PID Tuner App mostró un comportamiento adecuado.

4.7 Desempeño de los controladores

- La respuesta de la planta de control real frente a entradas tipo escalón con amplitud constante, mostró que: en el caso del controlador obtenido por el método de Ziegler-Nichols, fue oscilante y no se stabilizó; en el caso del controlador obtenido por el PID Tuner App, se stabilizó mostrando una pequeña oscilación previa; y en el caso del controlador obtenido mediante el uso de algoritmos genéticos y machine learning, también se stabilizó, pero mostrando menor oscilación previa y un tiempo de establecimiento menor.
- La respuesta de la planta de control real frente a entradas tipo escalón con amplitud variable, mostró que: en el caso del controlador obtenido por el método de Ziegler-Nichols, fue oscilante y no se stabilizó; en el caso del controlador obtenido por el PID Tuner App, se estabilizaba mostrando pequeñas oscilaciones previas; y en el caso del controlador obtenido mediante el uso de algoritmos genéticos y machine learning, también se estabilizaba, pero mostrando menores oscilaciones previas y tiempos de establecimiento menores.
- Los parámetros característicos de la respuesta transitoria fueron más positivos para el controlador hallado por algoritmos genéticos y machine learning. Mostrando un menor pico, menor tiempo de pico, menor tiempo de establecimiento, menor tiempo para alcanzar el 50% y el 100% del valor en estado estable

5 CAPÍTULO V: CONCLUSIONES

- 5.1** El disco de 4 ranuras acoplado mecánicamente al encoder, los demás elementos de hardware y, por otro lado, la programación de un tiempo de muestreo de 0.22848s tanto para el controlador digital como para el filtro digital pasa bajos de frecuencia de corte 0.1 Hz; fueron adecuados para lograr implementar el sistema de control de velocidad.
- 5.2** La elección de una señal tipo escalón, con una duración de 4.798s antes del siguiente cambio en su amplitud a algún valor entre 0 y 255 para manipular la señal PWM de excitación de la planta, permitió obtener un conjunto de datos que realmente reflejó en gran magnitud el comportamiento transitorio del sistema frente a diversas excitaciones, y entrenar el modelo de machine learning.
- 5.3** La elección de una señal tipo escalón, con una amplitud de 255 para manipular la señal PWM de excitación de la planta, permitió obtener un conjunto de datos adecuado para el proceso de identificación por medio de una función de transferencia.
- 5.4** El modelo de machine learning basado en el método de regresión lineal Elastic Net y entrenado con datos actuales y atrasados hasta en cuatro muestras, logró un ajuste de más del 99% sobre el conjunto de prueba. Mientras que la función de transferencia basada en un cero, dos polos reales y un retardo, logró un ajuste de más del 95%.
- 5.5** Bajo el supuesto de que el modelo de machine learning obtenido, era prácticamente una abstracción del comportamiento de la planta en tiempo discreto correspondiendo a un tiempo de muestreo de 0.22848, y teniendo en cuenta sus restricciones de velocidad máxima de 388 rad/s y mínima de 0 rad/s; se logró implementar con éxito un controlador PID digital con el mismo tiempo de muestreo para evaluar su desempeño en base al criterio cuantitativo que proporciona la integral del tiempo multiplicado por el valor absoluto del error.

- 5.6** El tiempo de muestreo máximo recomendado para la implementación del controlador, según la referencia teórica, es la décima parte del tiempo requerido para la estabilización de la señal de salida del sistema: 4.798 s (147 ciclos del timer). Es decir, 0.4798 s. Por lo tanto, un tiempo de muestreo de 0.22848 es un tiempo adecuado.
- 5.7** Los parámetros escogidos para el algoritmo genético permitieron hallar con éxito una tendencia rápida hacia la solución. Además, al escoger una cantidad de 11 bits para cada ganancia y sus rangos de búsqueda de 0 a 2, se proporcionó 2048 valores de búsqueda para el algoritmo genético, lo que significa una precisión de 0.00097.
- 5.8** El controlador hallado por algoritmos genéticos y machine learning mostró una mejor performance y mejores resultados en el análisis conjunto, obteniéndose tiempos de 1.82784s y de 3.19878s para alcanzar el 50% y el 100%, respectivamente, de la señal en estado estable. Así también un pico máximo de 1.04904%, correspondiente a un tiempo de pico de 4.11264s y un tiempo de establecimiento de 5.25504.

6 CAPÍTULO VI: SUGERENCIAS

- 6.1** La planta de control utilizada en este trabajo cumple con características idóneas para implementar un sistema de control de velocidad angular, pero si se busca implementar un sistema de control de posición angular entonces se debería acoplar un disco con más aberturas, según la precisión que se busque.
- 6.2** Si se busca controlar el motor dc a bajas velocidades, un controlador PID que mantenga sus ganancias constantes no será suficiente debido a que el motor presenta un comportamiento no lineal a velocidades menores que 70 rad/s. Para solucionar esto, podría optarse por implementar estrategias de control adicionales como el uso de lógica difusa para cambiar las ganancias del controlador según determinadas regiones de trabajo del motor dc.
- 6.3** Es fundamental elegir una señal o las señales de entrada adecuadas para garantizar el éxito de la identificación del sistema dinámico estudiado a través de un modelo de machine learning. Esto se logra tomando en cuenta todas las perturbaciones a las que podría estar sometida la planta de control y estudiando su comportamiento frente a ellas; sin importar si la planta de control es un sistema lineal simple de una entrada y una salida o es un sistema complejo no lineal de múltiples entradas y salidas.
- 6.4** Los parámetros seleccionados para conformar el conjunto de entradas para entrenar el modelo de machine learning dependen directamente del tiempo de establecimiento de la planta y la cantidad de variables de entrada del sistema de control. Es así que, si un sistema tiene un tiempo de establecimiento relativamente prolongado, será necesario aumentar la cantidad de estados anteriores (número de atrasos en las muestras) para cada una de las variables de entradas y salidas consideradas en la selección de parámetros.
- 6.5** Si bien se puede entrenar un modelo de machine learning y obtener un ajuste cercano al 100% sobre un conjunto de prueba, esto no es garantía suficiente para que el modelo sea confiable. Por diversos motivos, como la mala elección de parámetros, se podrían presentar problemas de sobre ajuste. Por lo tanto, siempre es necesario evaluar el modelo a través de diversas métricas o visualizaciones.

- 6.6** No se usaron métodos analíticos para seleccionar los rangos de búsqueda de los algoritmos genéticos para los parámetros PID. A pesar de que los métodos analíticos pueden ser muy eficientes en sistemas simples, mientras la complejidad de un sistema va aumentando, su análisis también. Es aquí donde se muestra la ventaja principal de incorporar técnicas de machine learning y algoritmos genéticos en problemas de control, puesto que se pueden implementar pruebas para estimar la performance del controlador y dar así con valores adecuados para sintonizarlo con éxito. Así también, se podría usar los parámetros PID obtenidos por otras técnicas de sintonización como valores de entrada. Y el usuario podría establecer categorías de búsqueda para las ganancias. Además, para aprovechar mejor las capacidades computacionales de la tecnología actual, se podría evaluar cada controlador en menos iteraciones e iniciar realizando un control proporcional y continuar con el resto de ganancias. Esto sería la base para intentar automatizar el proceso de sintonización.
- 6.7** Los individuos con menores factores ITAE de la Tabla 3.3 no presentan grandes diferencias en este valor, por lo que podría ser adecuado probar más de un individuo para hallar un controlador, y hasta posiblemente tomar en cuenta parámetros adicionales que proporcionen información cualitativa del desempeño del controlador; para elegir alguno que se ajuste a necesidades particulares,

7 CAPÍTULO VII: BIBLIOGRAFÍA

ACEDO SÁNCHEZ, J. 2006. Instrumentación y control avanzado de procesos. Madrid: Díaz de Santos. ISBN 9788479787547.

ALCALDE SAN MIGUEL, PABLO (2014). Electrotecnia: Instalaciones eléctricas y automáticas. 6^a ed. Madrid: Paraninfo. ISBN 8428398771, 9788428398770.

ÅSTRÖM, K. y HÄGGLUND, T. 2001. The future of PID control. Control Engineering Practice. Vol. 9, no. 11, pp. 1163-1175. DOI 10.1016/s0967-0661(01)00062-4. Elsevier BV.

ÅSTRÖM, K. y MURRAY, R. 2006. Feedback Systems: An Introduction for Scientists and Engineers. 2da ed v4. Åström and Richard M. Murray.

BALCÁZAR LLANOS, J. 2017. Uso De Algoritmos Genéticos Para La Sintonización De Un Controlador Pid Aplicado Al Control De Nivel Utilizando El Entrenador De Control De Procesos “DI 2314” Del Instituto Superior Tecnológico Publico Nueva Esperanza. [en línea]. Tesis de titulación. [Consulta: 1 septiembre 2018]. Disponible en: <http://dspace.unitru.edu.pe/handle/UNITRU/9482>.

BAUTISTA THOMPSON, E., GUZMÁN RAMÍREZ, E. y FIGUEROA NAZUNO, J. 2004. Predicción de Múltiples Puntos de Series de Tiempo Utilizando Support Vector Machines. Computación y Sistemas [en línea]. Vol. 7, no. 3, pp. 148-155. [Consulta: 1 septiembre 2018]. ISSN 1405-5546. Disponible en: http://www.scielo.org.mx/scielo.php?script=sci_abstract&pid=S1405-55462004000100002&lng=es&nrm=iso. Scielo

BHATTACHARYA, S. 2009. Control systems engineering. Delhi: Dorling Kindersley. ISBN 8131718204, 9788131718209.

BOTTERÓN, F., 2018. Sistema de Control Digital. image. 2018.

BURGOS MOLINA, C. y CHACÓN PACHECO, M. 2014. ENTRENAMIENTO MUESTRAL DE MODELOS DINÁMICOS CON SVM. [en línea]. [Consulta: 1 septiembre 2018]. Disponible en: <http://www.jcc2014.ucm.cl/jornadas/EVENTOS/ECC%202014/ECC-2.pdf>

- BUTTAY, C. 2006. Upload.wikimedia.org [en línea]. [Consulta: 1 septiembre 2018]. Disponible en: https://upload.wikimedia.org/wikipedia/commons/9/9e/Duty_cycle_general.png.
- CEDAR101 2016. Upload.wikimedia.org [en línea]. [Consulta: 1 septiembre 2018]. Disponible en: https://upload.wikimedia.org/wikipedia/commons/thumb/a/a1/Arduino_IDE_-_Blink.png/1024px-Arduino_IDE_-_Blink.png.
- COOK, J. 2017. Docker for data science. Santa Monica: Apress. ISBN 978-1-4842-3011-4.
- CORONA RAMÍREZ, L., ABARCA JIMÉNEZ, G. and MARES CARREÑO, J. 2014. Sensores y actuadores. Distrito Federal: Larousse - Grupo Editorial Patria. ISBN 978-607-438-936-4.
- DREZET, P. y HARRISON, R. 1998. Support vector machines for system identification. UKACC International Conference on Control (CONTROL '98). pp. 688-692. DOI 10.1049/cp:19980312. IEE
- FLIESS, M., JOIN, C. y SIRA-RAMIREZ, H. 2006. Complex Continuous Nonlinear Systems: Their Black Box Identification And Their Control. 14th IFAC Symposium on System Identification (SYSID 2006) [en línea]. [Consulta: 26 septiembre 2018]. HAL Id: inria-00000824. Disponible en: <https://hal.inria.fr/inria-00000824>.
- FUNES, E., ALLOUCHE, Y., BELTRÁN, G. and JIMÉNEZ, A. 2015. A Review: Artificial Neural Networks as Tool for Control Food Industry Process. Journal of Sensor Technology. Vol. 05, no. 01, pp. 28-43. DOI 10.4236/jst.2015.51004. Scientific Research Publishing, Inc.
- GARCIA PEREDA, J. 2018. Empleo del algoritmo de colonia de hormigas para la sintonización de un controlador pid aplicado a un circuito rc. [en línea]. Tesis de titulación. Trujillo: Universidad Nacional de Trujillo [Consulta: 1 septiembre 2018]. Disponible en: <http://dspace.unitru.edu.pe/handle/UNITRU/898>.
- GREER, J. 2014. Electronica de Potencia Practica 6. Jessica Greer [en línea]. [Consulta: 1 septiembre 2018]. Disponible en: <https://jesicagreer.wordpress.com/electronica-de-potencia-practica-6/>

- GRETTON, A., DOUCET, A., HERBRICH, R., RAYNER, P. y SCHOLKOPF, B. 2002. Support vector regression for black-box system identification. Proceedings of the 11th IEEE Signal Processing Workshop on Statistical Signal Processing (Cat. No.01TH8563). pp. 341-344. DOI 10.1109/ssp.2001.955292. IEEE.
- HAMETNER, C., MAYR, C. y KOZEK, M. 2013. PID controller design for nonlinear systems represented by discrete-time local model networks. International Journal of Control. Vol. 86, no. 9, pp. 1453–1466. DOI 10.1080/00207179.2012.759663. Informa UK Limited
- HISSEM, S., DOUMBIA, M. y KEDDAR, M. 2018. Novel Controller Design Based on Black Box Systems Approach. 2018 Annual American Control Conference (ACC). pp. 6427-6432. DOI 10.23919/acc.2018.8431035. IEEE
- INGARGIOLA, A. y colaboradores 2018. 1. What is the Jupyter Notebook? — Jupyter/IPython Notebook Quick Start Guide 0.1 documentation. Jupyter-notebook-beginner-guide.readthedocs.io [en línea]. [Consulta: 17 septiembre 2018]. Disponible en: https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html.
- JAGGI, M. 2014. An Equivalence between the Lasso and Support Vector Machines. [en línea]. no. 2, pp. 1-19. [Consulta: 1 septiembre 2018]. ID arXiv:1303.1152v2. Disponible en: <https://arxiv.org/abs/1303.1152v2>. arXiv.org
- KAHL, M., KROLL, A., KÄSTNER, R. and SOFSKY, M. 2015. Application of model selection methods for the identification of a dynamic boost pressure model. IFAC-PapersOnLine. Vol. 48, no. 28, pp. 829-834. DOI 10.1016/j.ifacol.2015.12.232. Elsevier BV
- KOCUR, M., KOZAK, S. and DVORSCAK, B. 2014. Design and Implementation of FPGA - digital based PID controller. Proceedings of the 2014 15th International Carpathian Control Conference (ICCC). pp. 1-4. DOI 10.1109/carpthiancc.2014.6843603. IEEE.
- KUO, B. y ARANDA PÉREZ, G. 1996. Sistemas de control automático. 7^a ed. México [etc.]: Prentice Hall Hispanoamericana. ISBN 9688807230, 978-9688807231.
- LAVRY, DAN 2012. Sampling Theory For Digital Audio. Lavry Engineering, Inc. [en línea]. [Consulta: 1 septiembre 2018]. Disponible en: <http://lavryengineering.com/pdfs/lavry-sampling-theory.pdf>

LÓPEZ MORA, C. 2014. Evaluación de desempeño de dos técnicas de optimización bio-inspiradas: Algoritmos Genéticos y Enjambre de Partículas. Revista Tekhnê [en línea]. Vol. 11, no. 1, pp. 49-58. [Consulta: 1 septiembre 2018]. ISSN 1692-8407. Disponible en: <http://revistas.udistrital.edu.co/ojs/index.php/tekhne/article/view/8951>. Tekhnê.

MALLAWAARACHCHI, V. 2018. Introduction to Genetic Algorithms — Including Example Code. Towards Data Science [en línea]. [Consulta: 17 septiembre 2018]. Disponible en: <https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3>

MARIN CANO, A., HERNANDEZ RIVERO, J. y JIMENEZ BUILES, J. 2018. Tuning Multivariable Optimal PID Controller for a Continuous Stirred Tank Reactor Using an Evolutionary Algorithm. IEEE Latin America Transactions. Vol. 16, no. 2, pp. 422-427. DOI 10.1109/tla.2018.8327395. Institute of Electrical and Electronics Engineers (IEEE).

MRLEJINAD 2016. Upload.wikimedia.org [en línea]. [Consulta: 1 septiembre 2018]. Disponible en: https://upload.wikimedia.org/wikipedia/commons/thumb/c/c9/Wave_square_labels.svg/800px-Wave_square_labels.svg.png.

OGATA, K. 1995. Discrete time control systems. 2da ed. London, NJ: Prentice-Hall International. ISBN 0-13-034281-5.

OGATA, K. 2010. Modern control engineering. 5th ed. Upper Saddle River, NJ: Prentice-Hall/Pearson. ISBN 0136156738, 9780136156734.

PARDO, C. (2018). PID digital - Picuino. [online] Sites.google.com. Disponible en: https://sites.google.com/site/picuino/digital_pid#TOC-Per-odo-de-muestreo [fecha de consulta: 1 de junio del 2018].

PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M. and DUCHESNAY, E. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. Vol. 12, pp. 2825--2830.

RAMKRISHNA, GHOSH, 2016, Input designs for identification of Ill-conditioned multivariable systems. Finland: Åbo Akademi University. ISBN 978-952-12-3434:7.

RIBEIRO, A. y AGUIRRE, L. 2018. Lasso Regularization Paths for NARMAX Models via Coordinate Descent. 2018 American Control Conference [en línea]. Vol. 2, pp. 1-6. [Consulta: 8 de junio del 2018]. ID arXiv: 1710.00598v2. Disponible en: <https://arxiv.org/abs/1710.00598v2>. arXiv.org

SAAD, M., JAMALUDDIN, H. y DARUS, I. 2012. PID Controller Tuning Using Evolutionary Algorithms. WSEAS TRANSACTIONS on SYSTEMS and CONTROL. Vol. 7, no. 4, pp. 139-149. DOI E-ISSN: 2224-2856.

SARKAR, DIPANJAN, BALI, RAGHAV y SHARMA, TUSHAR, 2018, Practical machine learning with Python. Bangalore, Karnataka, India: Raghav Bali. ISBN 1484232070, 9781484232071.

SCHÖLKOPF, B. y SMOLA, A. J. (2001). Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA. ISBN 9780262194754.

SPARKFUN ELECTRONICS 2013. Cdn.sparkfun.com [en línea]. [Consulta: 1 septiembre 2018]. Disponible en: <https://cdn.sparkfun.com/assets/parts/6/8/1/6/11224-04.jpg>.

STRMČNIK, S., ŠEGA, M., PETROVČIČ, J. y TRAMTE, P. 1987. Black-box Modelling in Control System Design - A Case Study. IFAC Proceedings Volumes. Vol. 20, no. 12, pp. 327-332. DOI 10.1016/s1474-6670(17)55651-7. Elsevier BV.

THE MATHWORKS INC. (2018). Documentación. [online] mathworks.com. Disponible en: https://la.mathworks.com/help/?s_tid=hp_ff_1_doc [fecha de consulta: 1 de junio del 2018].

TIBSHIRANI, R. 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society, Series B, Vol 58, No. 1, pp. 267–288.

TORRENTE ARTERO, O. 2013. Arduino. México, D.F.: Alfaomega. ISBN 978-607-707-648-3.

VILANOVA, R. y VISIOLI, A. 2012. PID Control in the Third Millennium. Dordrecht: Springer. ISBN 9781447124252, 1447124251.

WU, CHIA-JU, 1999, Genetic Tuning of PID Controllers Using a Neural Network Model: A Seesaw Example. *Journal of Intelligent and Robotic Systems*. 1999. Vol. 25, no. 1, pp. 43-59. DOI 10.1023/a:1008077610571. Springer Nature.

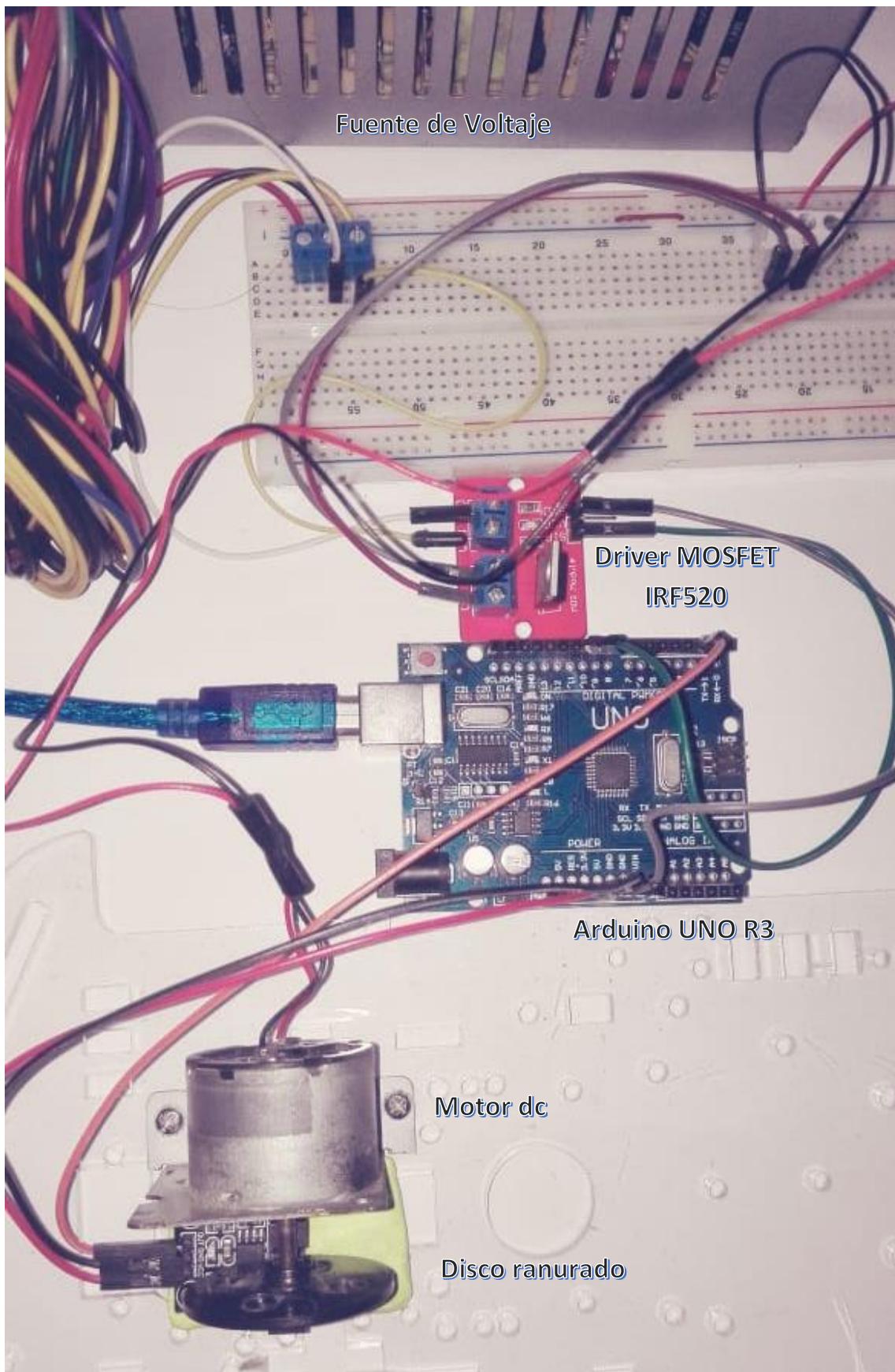
YIN, S., LI, X., GAO, H. and KAYNAK, O. 2015. Data-Based Techniques Focused on Modern Industry: An Overview. *IEEE Transactions on Industrial Electronics*. Vol. 62, no. 1, pp. 657-667. DOI 10.1109/tie.2014.2308133. Institute of Electrical and Electronics Engineers (IEEE).

ZHOU, Q., CHEN, W., SONG, S., GARDNER, J., WEINBERGER, K. and CHEN, Y. 2015. A Reduction of the Elastic Net to Support Vector Machines with an Application to GPU Computing. *AAAI Conference on Artificial Intelligence Twenty-Ninth AAAI Conference on Artificial Intelligence PRESENTATIONS* [en línea]. pp. 3210-3216. [Consulta: 1 septiembre 2018]. Disponible en: <https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9856/10002>.

ZOU, H. y HASTIE, T. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*. Vol. 67, no. 2, pp. 301–320.

8 ANEXOS

8.1 Disposición de los elementos de hardware



8.2 Código en Python de configuraciones previas

```
# Instalando dependencias
import sys
!{sys.executable} -m pip install pyserial times numpy pandas
xlsxwriter matplotlib sklearn deap

# Llamando dependencias
import serial
import random
import numpy as np
import pandas as pd
import xlsxwriter
import matplotlib.pyplot as plt
from time import time
from sklearn.model_selection import train_test_split
from sklearn.linear_model import ElasticNet
from deap import base, creator
from deap import tools
```

8.3 Código en Python para almacenar los datos en memoria y crear la hoja de cálculo “motor.xlsx”

```
ser = serial.Serial('/dev/ttyUSB0', 115200) # Configuración del puerto serial

# Vector para valores entre 0 y 255
inputData = []
# Vector para valores de velocidad (rad/s)
outputData = []

# Número de muestras requeridas
data_lenght = 20000
# Variable para almacenar la longitud de outputData
L = 0

# Tirar datos basura en el serial buffer
ser.flushInput()

while L < data_lenght:
    ser.reset_input_buffer()

    # Leer valor actual del puerto serial
    data = ser.readline().decode("utf-8").split(' ')

    # Actualizar inputData y outputData con valores muestreados
    try:
        inputData.append(float(data[0]))
        outputData.append(float(data[1]))
    except: pass

    # Actualizar variable L
    L = len(outputData)

# Cerrar puerto serial
ser.close()
```

```
# Creación de la hoja de cálculo a partir de inputData y
outputData
workbook = xlsxwriter.Workbook('motor.xlsx')
worksheet = workbook.add_worksheet()

worksheet.write('A1', 'time')
worksheet.write('B1', 'u')
worksheet.write('C1', 'y')

k = 1
Ts = 0.22848

for entry, output in zip(inputData, outputData):
    worksheet.write(k, 0, Ts * k)
    worksheet.write(k, 1, entry)
    worksheet.write(k, 2, output)
    k += 1

workbook.close()
```

8.4 Código en Python para generar los conjuntos de entrada X e Y

```
x = pd.read_excel('motor.xlsx')
x = x.drop('time', axis=1)
L = len(x)

# Columnas u(k-1) y y(k-1)
q1 = pd.DataFrame()
q1['u1'] = pd.Series(x['u'], index=x.index)
q1['y1'] = pd.Series(x['y'], index=x.index)
q1 = q1.shift(periods=-1, freq=None, axis=0)

x['u1'] = pd.Series(q1['u1'], index=x.index)
x['y1'] = pd.Series(q1['y1'], index=x.index)

# Columnas u(k-2) y y(k-2)
q2 = pd.DataFrame()
q2['u2'] = pd.Series(x['u'], index=x.index)
q2['y2'] = pd.Series(x['y'], index=x.index)
q2 = q2.shift(periods=-2, freq=None, axis=0)

x['u2'] = pd.Series(q2['u2'], index=x.index)
x['y2'] = pd.Series(q2['y2'], index=x.index)

# Columnas u(k-3) y y(k-3)
q3 = pd.DataFrame()
q3['u3'] = pd.Series(x['u'], index=x.index)
q3['y3'] = pd.Series(x['y'], index=x.index)
q3 = q3.shift(periods=-3, freq=None, axis=0)

x['u3'] = pd.Series(q3['u3'], index=x.index)
x['y3'] = pd.Series(q3['y3'], index=x.index)

# Columnas u(k-4) y y(k-4)
q4 = pd.DataFrame()
q4['u4'] = pd.Series(x['u'], index=x.index)
q4['y4'] = pd.Series(x['y'], index=x.index)
```

```
q4 = q4.shift(periods=-4, freq=None, axis=0)

x['u4'] = pd.Series(q4['u4'], index=x.index)
x['y4'] = pd.Series(q4['y4'], index=x.index)

x = x.drop(x.index[L-4:L])

y = pd.DataFrame()
y['y'] = x['y'] # Conjunto Y listo

x = x.drop('y', axis=1) # Conjunto x listo
```

8.5 Código en Python para entrenar y probar el modelo de machine learning

```
# Separando los conjuntos de entrada: 60% para entrenamiento y
# 40% para test
x_train, x_test, y_train, y_test =
train_test_split(x,y,test_size=0.4)

# Instanciando el modelo
model = ElasticNet()

# Entrenamiento del modelo
model.fit(x_train, y_train)

# Porcentaje de ajuste del modelo sobre el conjunto de test
model.score(x_test,y_test)

# Inspección visual como una de las medidas para prevenir
# sobre ajuste
predicted = model.predict(x_test)
%matplotlib inline
plt.hist([predicted, y_test['y']])
```

8.6 Código en Python para sintonizar el controlador mediante algoritmos genéticos

```
# Configuración para buscar controladores que minimizan el error
creator.create("ControllerFitness", base.Fitness, weights=(-
1.0,))

# Creación de la clase para crear individuos (cromosomas)
creator.create("Controller", list,
fitness=creator.ControllerFitness)

# _ki bits + _kp bits + _kd bits
IND_SIZE = 30

# Configuración de los individuos como palabras binarias
toolbox = base.Toolbox()
toolbox.register("bit_attribute", random.randint, 0, 1)
toolbox.register("controller", tools.initRepeat,
creator.Controller, toolbox.bit_attribute, n=IND_SIZE)
toolbox.register("population", tools.initRepeat, list,
toolbox.controller)
```

8.7 Código en Python para evaluar la performance del controlador decodificado

```
def evaluate(individuo):
    # ===== Decodificación de individuo =====
    controller = ''.join(map(str, individuo))

    # Palabra digitales
    _kp = int(controller[0:11], 2)
    _ki = int(controller[11:22], 2)
    _kd = int(controller[22:33], 2)

    # Valores decodificados
    kp = 0 + _kp * ( (2 - 0) / (pow(2, 11) - 1) )
    ki = 0 + _ki * ( (2 - 0) / (pow(2, 11) - 1) )
    kd = 0 + _kd * ( (2 - 0) / (pow(2, 11) - 1) )
    # =====

    setPoint = 200
    ymax = 390
    Ts = 0.22848

    pid1 = 0
    pid0 = 0
    pid2 = 0
    pid3 = 0
    pid4 = 0

    error0 = 0
    error1 = 0
    error2 = 0

    y = 0
    y1 = 0
    y2 = 0
    y3 = 0
    y4 = 0
```

```

E = [] # Vector de errores

for k in np.arange(1000):
    q1 = kp + 0.5 * Ts * ki + (kd / Ts)
    q2 = Ts * ki * 0.5 - kp - (2 * kd / Ts)
    q3 = kd / Ts

    error2 = error1
    error1 = error0

    pid4 = pid3
    pid3 = pid2
    pid2 = pid1
    pid1 = pid0

    pid0 = pid1 + q1 * error0 + q2 * error1 + q3 * error2

    if (pid0 > 255):
        pid0 = 255
    elif (pid0 < 0):
        pid0 = 0
    else:
        pid0 = pid0

    # Generación de una entrada "u"
    u = {'pwm': [pid0], 'pwm1': [pid1], 'y1': [y1], 'pwm2': [pid2],
          'y2': [y2], 'pwm3': [pid3], 'y3': [y3], 'pwm4': [pid4],
          'y4': [y4]}

    # Predicción del modelo para la entrada "u"
    _y = model.predict(pd.DataFrame(data=u))
    y = _y[0]

    # Implementación de límites de la planta real
    if (y > ymax):
        y = ymax
    elif (y < 0):

```

```

y = 0

error0 = setPoint - y

y4 = y3
y3 = y2
y2 = y1
y1 = y

# Tiempo multiplicado por el valor absoluto del error
itae = error0*k*Ts
E.append(itae)

# ITAE
total_error = sum(list(map(abs, E)))
return (total_error)

# Mostrar parámetros PID y el factor itae
print([kp, ki, kd, total_error])

```

8.8 Código en Python del proceso de sintonización

```
# Configuración de las funciones de crossover, mutación,
selección y evaluación de individuos
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutUniformInt, low=0, up=1,
indpb=0.1)
toolbox.register("select", tools.selTournament, tournsize=3)
toolbox.register("evaluate", evaluate) # "evaluate": Misma
función que decodifica y evalua el desempeño de un controlador

def main():
    # Inicializar población
    pop = toolbox.population(n=50)

    # Factores de crossover, mutación y número de generaciones
    CXPB, MUTPB, NGEN = 0.7, 0.5, 50

    # Evaluar la población
    fitnesses = list(map(toolbox.evaluate, pop))
    for ind, fit in zip(pop, fitnesses):
        ind.fitness.values = fit

    minFit = (1,)

    for g in range(NGEN):
        # Seleccionando la siguiente generación
        offspring = toolbox.select(pop, len(pop))
        # Clonando los individuos seleccionados
        offspring = list(map(toolbox.clone, offspring))

        # Aplicando crossover y mutación
        for child1, child2 in zip(offspring[::2],
offspring[1::2]):
            if random.random() < CXPB:
                toolbox.mate(child1, child2)
                del child1.fitness.values
```

```

        del child2.fitness.values

    for mutant in offspring:
        if random.random() < MUTPB:
            toolbox.mutate(mutant)
        del mutant.fitness.values

    # Evaluación de desempeño de los individuos con un
    fitness inválido mediante
    invalid_ind = [ind for ind in offspring if not
    ind.fitness.valid]
    fitnesses = list(map(toolbox.evaluate, invalid_ind))
    minFit = min(fitnesses)
    for ind, fit in zip(invalid_ind, fitnesses):
        ind.fitness.values = fit

    # The population is entirely replaced by the offspring
    pop[:] = offspring

    return pop

best_controllers_pop = main()

```

8.9 Código en Python para simular un controlador

```
def evaluateOne(p, i, d, setpoint):  
    kp = p  
    ki = i  
    kd = d  
  
    # ======  
    pid1 = 0  
    pid0 = 0  
    pid2 = 0  
    pid3 = 0  
    pid4 = 0  
  
    error0 = 0  
    error1 = 0  
    error2 = 0  
  
    Y = 0  
    y1 = 0  
    y2 = 0  
    y3 = 0  
    y4 = 0  
  
    E = []  
    Y = []  
  
    Sp = setpoint  
    Ts = 0.22848  
  
    for k in np.arange(1000):  
        q1 = kp + 0.5 * Ts * ki + (kd / Ts)  
        q2 = Ts * ki * 0.5 - kp - (2 * kd / Ts)  
        q3 = kd / Ts  
  
        error2 = error1  
        error1 = error0
```

```

pid4 = pid3
pid3 = pid2
pid2 = pid1
pid1 = pid0

pid0 = pid1 + q1 * error0 + q2 * error1 + q3 * error2

if (pid0 > 255):
    pid0 = 255
elif (pid0 < 0):
    pid0 = 0

d = {'pwm': [pid0], 'pwm1': [pid1], 'y1': [y1], 'pwm2':
[pid2], 'y2': [y2], 'pwm3': [pid3], 'y3': [y3], 'pwm4': [pid4],
'y4': [y4]}

_y = model.predict(pd.DataFrame(data=d))

y = _y[0]

if (y > 374.5):
    y = 374.5
elif (y < 0):
    y = 0

error0 = Sp - y

y4 = y3
y3 = y2
y2 = y1
y1 = y
Y.append(y)
E.append(error0)

total_error = sum(list(map(abs, E)))

return [Y, E, total_error]

```

8.10 Código del muestreo digital para Arduino

```
volatile unsigned int c = 0;
unsigned long t1, t2, t;

int PwmPin = 10;
int encoderDIGPin = 2;

float f, vel, vel0, vel1;
float pi = 3.14159265359;
int u=0, n=0;

void setup() {
    Serial.begin(115200);
    pinMode(PwmPin, OUTPUT);
    pinMode(encoderDIGPin, INPUT);

    SREG = (SREG & 0b01111111);
    TIMSK2 = TIMSK2 | 0b00000001;
    TCCR2B = 0b00000111;
    SREG = (SREG & 0b01111111) | 0b10000000;
}

void loop() {
    t1 = pulseIn(encoderDIGPin, HIGH);
    t2 = pulseIn(encoderDIGPin, LOW);

    analogWrite(PwmPin, u);
}

ISR(TIMER2_OVF_vect){
    c++;
    if(c > 147) {
        u = random(0, 255);
        c=0;
    }

    t = t1 + t2;
    if (t == 0) {
        f = 500000;
    } else {
        f = 500000/t;
    }

    if (f > 1000) {
        f = 0;
    }
    f = f * 17 / 8;
}
```

```
vel = pi * 0.5 * f;
vel0 = 0.0203 * vel + 0.9797 * vel1;
vel1 = vel0;

if(c % 7 == 0) {
    Serial.print(u);
    Serial.print(" ");
    Serial.print(vel0);
    Serial.println(" ");
}
}
```

8.11 Código del control digital para Arduino

```
volatile unsigned int c = 0;
unsigned long t1, t2, t;

int PwmPin = 10;
int encoderDIGPin = 2;

float f, vel, vel0, vel2;
float pi = 3.14159265359;
int u=0, n=0;

float pid0 = 0;
float pid1 = 0;
float pid2 = 0;
float error0 = 0;
float error1 = 0;
float error2 = 0;
float y1 = 0;
float y2 = 0;

float kp = 1.1378299120234605;
float ki = 1.0557184750733137;
float kd = 0.015640273704789834;
float Ts = 0.22848;

float q1 = kp + 0.5 * Ts * ki + (kd / Ts);
float q2 = Ts * ki * 0.5 - kp - (2 * kd / Ts);
float q3 = kd / Ts;

void setup() {
    Serial.begin(115200);
    pinMode(PwmPin, OUTPUT);
    pinMode(encoderDIGPin, INPUT);

    SREG = (SREG & 0b01111111);
    TIMSK2 = TIMSK2|0b00000001;
    TCCR2B = 0b00000111;
    SREG = (SREG & 0b01111111) | 0b10000000;
}

void loop() {
    t1 = pulseIn(encoderDIGPin, HIGH);
    t2 = pulseIn(encoderDIGPin, LOW);

    analogWrite(PwmPin, pid0);
}
```

```

ISR (TIMER2_OVF_vect){ // función que se Llama cada 32.64ms
    c++;
    if(c > 147) {
        Sp = random(150, 255);
        c=0;
    }

    t = t1 + t2;
    if (t == 0) {
        f = 500000;
    } else {
        f = 500000/t;
    }

    if (f > 1000) {
        f = 0;
    }
    f = f * 17 / 8;

    vel = pi * 0.5 * f;
    vel0 = 0.0203 * f + 0.9797 * vel1;
    vel1 = vel0;

    if(c % 7 == 0) {
        error2 = error1;
        error1 = error0;

        pid2 = pid1;
        pid1 = pid0;

        error0 = Sp - f;

        pid0 = pid1 + q1 * error0 + q2 * error1 + q3 * error2;

        if (pid0 > 255)
        {
            pid0 = 255;
        }
        else if (pid0 < 0)
        {
            pid0 = 0;
        }

        Serial.print(Sp);
        Serial.print(" ");
        Serial.print(vel0);
        Serial.println(" ");
    }
}

```

8.12 123 primero datos de la hoja de cálculo “motor.xlsx”

N	time	u	y	N	time	u	y	N	time	u	y
1	0.22848	200	0	42	9.59616	200	316.83	83	18.96384	200	320.88
2	0.45696	200	0	43	9.82464	200	318.37	84	19.19232	200	321.48
3	0.68544	200	0	44	10.05312	200	318.67	85	19.4208	200	320.8
4	0.91392	200	0	45	10.2816	200	317.64	86	19.64928	200	320.11
5	1.1424	200	0	46	10.51008	200	318.05	87	19.87776	200	320.88
6	1.37088	200	0	47	10.73856	200	319.61	88	20.10624	200	321.36
7	1.59936	200	8.8	48	10.96704	200	319.37	89	20.33472	200	320.31
8	1.82784	200	38.14	49	11.19552	200	318.27	90	20.5632	200	319.89
9	2.05632	200	72.02	50	11.424	200	318.85	91	20.79168	200	320.82
10	2.2848	200	103.73	51	11.65248	200	320.26	92	21.02016	200	320.9
11	2.51328	200	132.93	52	11.88096	200	320.14	93	21.24864	200	319.93
12	2.74176	200	158.75	53	12.10944	200	319.17	94	21.47712	200	319.79
13	2.97024	200	179.66	54	12.33792	200	319.75	95	21.7056	200	320.76
14	3.19872	200	197.13	55	12.5664	200	320.74	96	21.93408	200	320.78
15	3.4272	200	212.3	56	12.79488	200	319.87	97	22.16256	200	319.47
16	3.65568	200	226.62	57	13.02336	200	319.6	98	22.39104	200	319.64
17	3.88416	200	239.9	58	13.25184	200	320.87	99	22.61952	200	320.98
18	4.11264	200	250.62	59	13.48032	200	320.81	100	22.848	200	320.56
19	4.34112	200	258.87	60	13.7088	200	319.76	101	23.07648	200	319.3
20	4.5696	200	265.44	61	13.93728	200	320.21	102	23.30496	200	319.76
21	4.79808	200	272.74	62	14.16576	200	321.5	103	23.53344	200	321.06
22	5.02656	200	280.11	63	14.39424	200	321.07	104	23.76192	200	320.63
23	5.25504	200	285.34	64	14.62272	200	320.4	105	23.9904	200	319.57
24	5.48352	200	288.74	65	14.8512	200	321.01	106	24.21888	200	320.08
25	5.712	200	292.35	66	15.07968	200	321.57	107	24.44736	200	321.01
26	5.94048	200	296.59	67	15.30816	200	320.43	108	24.67584	200	320.26
27	6.16896	200	300.25	68	15.53664	200	320.17	109	24.90432	200	319.89
28	6.39744	200	302.42	69	15.76512	200	321.53	110	25.1328	200	320.83
29	6.62592	200	303.53	70	15.9936	200	321.4	111	25.36128	200	320.95
30	6.8544	200	305.75	71	16.22208	200	320.22	112	25.58976	200	320.01
31	7.08288	200	309.14	72	16.45056	200	320.81	113	25.81824	200	319.99
32	7.31136	200	310.19	73	16.67904	200	321.81	114	26.04672	200	321.22
33	7.53984	200	310.19	74	16.90752	200	321.18	115	26.2752	200	321.02
34	7.76832	200	311.7	75	17.136	200	320.6	116	26.50368	200	320.02
35	7.9968	200	314.03	76	17.36448	200	321.58	117	26.73216	200	320.25
36	8.22528	200	314.81	77	17.59296	200	321.84	118	26.96064	200	321.37
37	8.45376	200	314.2	78	17.82144	200	320.48	119	27.18912	200	320.72
38	8.68224	200	314.96	79	18.04992	200	320.58	120	27.4176	200	319.77
39	8.91072	200	316.85	80	18.2784	200	321.8	121	27.64608	200	320.52
40	9.1392	200	317.16	81	18.50688	200	321.34	122	27.87456	200	321.16
41	9.36768	200	316.47	82	18.73536	200	320.37	123	28.10304	200	320.49

8.13 123 primeros datos de la hoja de cálculo “motor-stepentry.xlsx”

N	time	u	y	N	time	u	y	N	time	u	y
1	0.22848	0	0	42	9.59616	0	0	83	18.73536	0	0
2	0.45696	0	0	43	9.82464	0	0	84	18.96384	0	0
3	0.68544	0	0	44	10.05312	0	0	85	19.19232	0	0
4	0.91392	0	0	45	10.2816	0	0	86	19.4208	0	0
5	1.1424	0	0	46	10.51008	0	0	87	19.64928	0	0
6	1.37088	0	0	47	10.73856	0	0	88	19.87776	0	0
7	1.59936	0	0	48	10.96704	0	0	89	20.10624	0	0
8	1.82784	0	0	49	11.19552	0	0	90	20.33472	0	0
9	2.05632	0	0	50	11.424	0	0	91	20.5632	0	0
10	2.2848	0	0	51	11.65248	0	0	92	20.79168	0	0
11	2.51328	0	0	52	11.88096	0	0	93	21.02016	0	0
12	2.74176	0	0	53	12.10944	0	0	94	21.24864	0	0
13	2.97024	0	0	54	12.33792	0	0	95	21.47712	0	0
14	3.19872	0	0	55	12.5664	0	0	96	21.7056	0	0
15	3.4272	0	0	56	12.79488	0	0	97	21.93408	0	0
16	3.65568	0	0	57	13.02336	0	0	98	22.16256	0	0
17	3.88416	0	0	58	13.25184	0	0	99	22.39104	0	0
18	4.11264	0	0	59	13.48032	0	0	100	22.61952	0	0
19	4.34112	0	0	60	13.7088	0	0	101	22.848	0	0
20	4.5696	0	0	61	13.93728	0	0	102	23.07648	0	0
21	4.79808	0	0	62	14.16576	0	0	103	23.30496	0	0
22	5.02656	0	0	63	14.39424	0	0	104	23.53344	0	0
23	5.25504	0	0	64	14.62272	0	0	105	23.76192	0	0
24	5.48352	0	0	65	14.8512	0	0	106	23.9904	0	0
25	5.712	0	0	66	15.07968	0	0	107	24.21888	0	0
26	5.94048	0	0	67	15.30816	0	0	108	24.44736	0	0
27	6.16896	0	0	68	15.53664	0	0	109	24.67584	0	0
28	6.39744	0	0	69	15.76512	0	0	110	24.90432	0	0
29	6.62592	0	0	70	15.9936	0	0	111	25.1328	0	0
30	6.8544	0	0	71	16.22208	0	0	112	25.36128	0	0
31	7.08288	0	0	72	16.45056	0	0	113	25.58976	0	0
32	7.31136	0	0	73	16.67904	0	0	114	25.81824	0	0
33	7.53984	0	0	74	16.90752	0	0	115	26.04672	0	0
34	7.76832	0	0	75	17.136	0	0	116	26.2752	0	0
35	7.9968	0	0	76	17.36448	0	0	117	26.50368	0	0
36	8.22528	0	0	77	17.59296	0	0	118	26.73216	0	0
37	8.45376	0	0	78	17.82144	0	0	119	26.96064	0	0
38	8.68224	0	0	79	18.04992	0	0	120	27.18912	0	0
39	8.91072	0	0	80	18.2784	0	0	121	27.4176	0	0
40	9.1392	0	0	81	18.50688	0	0	122	27.64608	0	0
41	9.36768	0	0	82	18.73536	0	0	123	27.87456	0	0