**ABSTRACCIÓN DE UN SISTEMA MOTOR DC – ENCODER EN UN MODELO DE MACHINE LEARNING PARA SINTONIZAR UN CONTROLADOR PID DE VELOCIDAD APLICANDO ALGORITMOS GENÉTICOS**

# **DEDICATORIA**

# **AGRADECIMIENTO**

# **RESUMEN**

# **ABSTRACT**

# **INTRODUCCIÓN**

## **Realidad problemática**

Es conocida la magnitud de la importancia del control de procesos en ambientes industriales, investigativos y didácticos. Tan es así que, en uno de los casos más difundidos de controladores, como es el controlador Proporcional Integral Derivativo debido a su alto desempeño y facilidad de implementación, se han desarrollado diversos métodos analíticos y experimentales para lograr hallar parámetros que garanticen un comportamiento adecuado del sistema de control. Sin embargo, con el desarrollo de nuevas tecnologías se han abierto múltiples posibilidades para expandir este campo de estudio y abordar la sintonización de controladores desde enfoques distintos a los tradicionales.

Respecto de los métodos modernos, se pueden mencionar el uso de lógica difusa, algoritmos genéticos, algoritmos de colonia de hormigas, machine learning, etcétera; para funcionar tanto como elementos controladores, así como para optimizar controladores PID que necesitan del modelo matemático o de que el algoritmo funcione sobre la planta en tiempo real. Así también, ciertos métodos se vienen empleando como una alternativa a la identificación de sistemas desde sistemas simples de comportamiento lineal hasta sistemas complejos y caóticos de múltiples variables de comportamiento no lineal.

Entonces se presenta un escenario idóneo para dar soluciones nuevas y optimizadas a la sintonización de controladores, a través de la aplicación conjunta de algoritmos genéticos y machine learning.

## **Formulación del problema**

¿Cómo sintonizar un controlador PID de velocidad para un sistema motor DC – encoder, a partir de un conjunto de datos de los valores de manipulación del duty cycle de una señal PWM de entrada al sistema y de los valores de respuesta de la velocidad del motor?

## **Justificación del estudio**

### **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 tradicionalmente de forma separada y en otros contextos.

### **Relevancia Institucional**

Resolver problemas de control e identificación de sistemas, son campos todavía 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.

### **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 al tratar desde sistemas lineales de una variable de entrada y una de salida hasta complejos sistemas no lineales de múltiples variables de entrada y de salida.

### **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.

### **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.

## **Antecedentes**

## Carlos Pillajo, Paul Bonilla y Roberto Hincapié (2016) presentan la simulación de un algoritmo genético capaz de sintonizar controladores PID basado en el criterio de la integral del error absoluto. Así también, Mohd S. Saad, Hishamuddin Jamaluddin e Intan Z. M. Darus 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.

## Olalekan Ogunmolu, Xuejun Gu, Steve Jiang y and Nicholas Gans demuestran que las redes neuronales pueden proporcionar modelos estimadores efectivos a partir de datos de entrada y salida de sistemas dinámicos no lineales. Así mismo, Yu Wang realiza un estudio comparativo entre dos tipos de redes neuronales, evaluando su desempeño en la identificación de sistemas dinámicos y en el diseño de controladores.

## Por otro lado, Arthur Gretton, Amaud Douce, RalfHerbrich, Peter J. W Rayner y Bernhard Scholkop estudian técnicas de regresión de soporte vectorial para identificación de sistemas de caja negra, aplicadas a un brazo robótico hidráulico, encontrando resultados positivos respecto de publicaciones anteriores. Así también, Robert Salat, Michał Awtoniuk, Krzysztof Korpysz proponen el uso de técnicas de regresión de soporte vectorial para identificación de sistemas de caja negra, como una alternativa útil y eficiente para modelos basados en estructuras ARX y NARX para sistemas lineales y no lineales, respectivamente.

## **Objetivos**

### **General**

Abstraer un sistema motor DC – encoder en un modelo de machine learning para sintonizar un controlador PID de velocidad mediante algoritmos genéticos.

### **Específicos**

* Implementar el sistema de control.
* Elegir una señal de entrada y obtención de un conjunto adecuado de datos experimentales en tiempo discreto para la identificación del sistema de caja negra.
* Elegir un método de regresión para abstraer el sistema en un modelo de machine learning y definición de sus restricciones.
* Implementar un controlador PID discreto para definir su evaluación de desempeño en base a la integral del valor absoluto del error y su ejecución sobre el modelo de machine learning.
* Sintonizar el controlador PID aplicado al modelo de machine learning por medio de algoritmos genéticos.

## **Marco teórico**

**Sistemas LTI**

**Señales de entrada**

## **Sistema de control**

## **Controladores**

## **Desempeño de los sistemas de control**

## **Algoritmos genéticos**

## **Identificación de sistemas**

## **Arduino**

## **Jupyter Notebook**

## **Marco conceptual**

Tiempo de establecimiento

**Variable controlada**

**Variable manipulada**

**Planta**

**Proceso**

**Perturbación**

**Señal de referencia**

**Sensor**

**Actuador**

**Modelo matemático**

**Modelo de caja negra**

**Modelo de caja blanca**

# **MARCO METODOLÓGICO**

## **Hipótesis**

Es posible aplicar algoritmos genéticos para iterar sobre múltiples valores que permitan sintonizar un controlador PID discreto, cuyo desempeño será evaluado en base a la integral del valor absoluto del error, sobre la abstracción de un sistema motor DC - encoder, a través de un modelo de machine learning entrenado con datos obtenidos por muestreo digital de los valores de manipulación del duty cycle de la señal PWM de entrada y los valores de respuesta de la velocidad del motor.

## **Variables**

### **Variables independientes**

* Señal PWM.
* Frecuencia de muestreo.
* Número de muestras.

### **Variables dependientes**

* Modelo de machine learning.
* Velocidad del motor.
* Ganancia proporcional.
* Ganancia integral.
* Ganancia derivativa.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Definición conceptual** | **Definición operacional** | **Indicador** | **Escala de medición** |
| Señal de entrada | Señal de excitación sobre la planta de control. | Valor de manipulación por software del duty cycle de una señal PWM | Byte | 0 - 255 |
| Frecuencia de muestreo | Número de muestras por unidad de tiempo que se toman de una [señal continua](https://es.wikipedia.org/wiki/Se%C3%B1al_anal%C3%B3gica) para producir una señal discreta. | Número de muestras por segundo que se toman de los valores de manipulación del duty cycle de la señal PWM de entrada 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 de los valores de manipulación del duty cycle de la señal PWM de entrada y los valores de velocidad motor. | n° | > 0 |
| Velocidad del motor | Magnitud física que expresa el ángulo descrito en la unidad detiempo 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. | Nº | 0 – 1000 |
| Ganancia integral | Valor del controlador que elimina errores estacionarios. | Salida del controlador que es la integral de la señal de error. | Nº | 0 – 1000 |
| 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 – 1000 |

## **Metodología**

### **Tipo de estudio**

# Exploratorio

### **Diseño**

# Experimental.

## **Población y muestra**

### **Población**

Sistemas de motores DC – encoder.

### **Muestra**

Sistema motor de voltaje de operación nominal de 12VDC – encoder de voltaje de operación de 5V y una fase.

## **Método de investigación**

Cuantitativo.

## **Técnicas e instrumentos de recolección de datos**

### **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.

### **Instrumentos**

# Jupyter Notebook

# Plataforma Arduino

# Actuador

# Sensor

### **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.

# **DESARROLLO Y RESULTADOS**

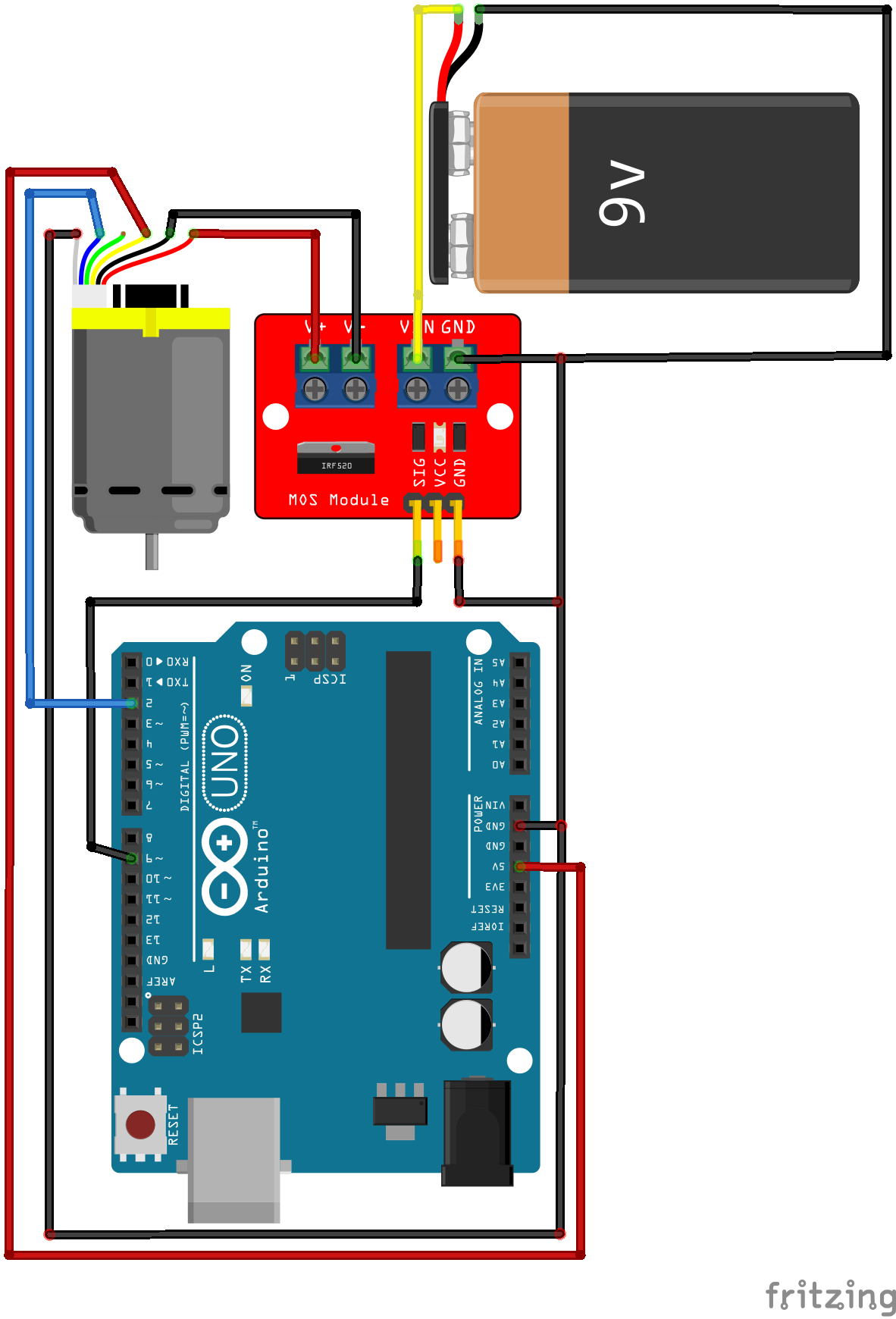
## **Implementación del sistema de control**

A continuación, se listan los elementos de software utilizados:

|  |  |
| --- | --- |
| **ELEMENTO** | **DESCRIPCIÓN** |
| Arduino IDE | Entorno Open Source de programación para la placa Arduino |
| Jupyter Notebook | Aplicación web Open Source de múltiples usos como: limpieza y transformación de datos, simulación numérica, modelado estadístico, visualización de datos y aprendizaje automático. |

A continuación, se listan los elementos de hardware utilizados:

|  |  |  |
| --- | --- | --- |
| **ELEMENTO** | **ESPECIFICACIONES** | **DESCRIPCIÓN** |
| Computadora | Procesador 4 núcleos mínimo | Elemento receptor de datos para su posterior procesamiento. |
| 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 |
| [Driver Mosfet IRF520](https://naylampmechatronics.com/drivers/239-driver-mosfet-irf520.html) | 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 | Elemento sensor |
| Fuente de voltaje | Voltajes de salida: 5 y 12 VDC | Alimentación del sistema de control |



**Fuente de alimentación**

**Motor DC y encoder**

**Placa Arduino**

**IRF520**

**12 VDC**

Disposición de los elementos de hardware del sistema de control. Elaborado en Fritzing.

### **Etapa de adquisición y procesamiento de datos:**

En esta etapa, la placa Arduino funcionará como tarjeta de adquisición de datos de las señales de entrada y salida de la planta de control. La placa se comunicará por medio del puerto serial a la laptop, donde se realizará el procesamiento de datos.



*Esquema de la disposición de elementos de software y hardware.*

**

*Funciones setup y loop del microprocesador de la placa Arduino*

**

*Interrupción del timer de la placa Arduino que se ejecuta cada 32.64 ms*

### **Etapa de control:**

En esta etapa, la placa Arduino funcionará como elemento controlador del sistema de control.



*Esquema de la disposición de elementos de software y hardware.*



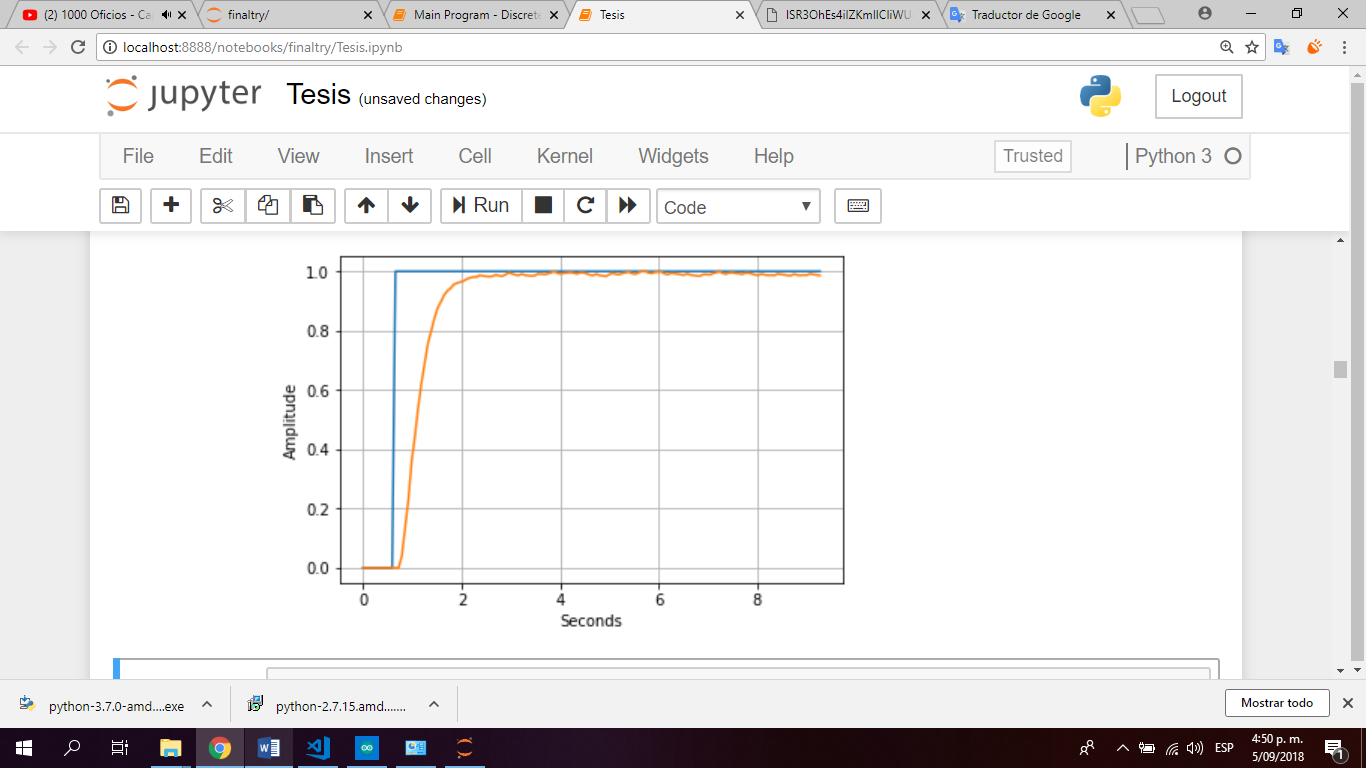


## **Elección de una señal de entrada para el sistema y obtención de un conjunto adecuado de datos experimentales en tiempo discreto para la identificación del sistema de caja negra.**

El sistema de control estudiado no estará sujeto a entradas que cambien de forma gradual en un determinado tiempo, ni tampoco a entradas de choque. Pero sí 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 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 promedio del voltaje del motor, que puede ser manipulado por una señal PWM; de esta forma, una señal PWM de 0% de duty cycle (Correspondiente al valor 0 de manipulación por software) permite obtener un valor de voltaje promedio de 0 y una señal PWM de 100% de duty cycle (Correspondiente al valor 255 de manipulación por software) permite obtener el valor de voltaje promedio y máximo de 12 V.

Así también, mediante la realización de pruebas experimentales con señales de entrada de tipo escalón, con valores distintos de amplitud (Valores de manipulación por software de 0 a 255) y duración (Número de ciclos de 32.64ms del timer del microcontrolador), fue posible determinar que el sistema se estabiliza frente a un cambio en la excitación, en menos de 4.896 s (150 ciclos del timer), y se observó que esto sucede sin importar la amplitud de la señal.



*Ejemplo de una señal de excitación para el sistema (color azul) y su respuesta (Señal - color naranja) con valores de amplitud normalizados*

Con las consideraciones anteriores, se determinaron los parámetros de diseño de la señal de entrada:

|  |  |
| --- | --- |
| **Tipo** | Escalón |
| **Amplitud** | 0 – 255 (Duty cycle de la señal PWM entre 0 y 100%) |
| **Duración** | 4.896 s |



*Señal de entrada para el sistema (color azul) y su velocidad (color naranja) en una ventana de tiempo con valores de amplitud normalizados.*

Para el proceso de muestreo, debido al ruido en la salida del sistema, fue necesario usar un filtro digital paso bajo de primer orden con una frecuencia de corte y un tiempo de muestreo :

Además, se definió un tiempo de muestreo de 0.22848 (7 ciclos del timer) y se tomó un total de 6986 muestras que fueron guardados en la hoja de cálculo “motor. xlms”.

A continuación, se muestran los diagramas de flujo del proceso de muestreo digital:

**

*Almacenamiento de datos mediante un programa en Python para Jupyter Notebook*

## **Elección del modelo de machine learning y definición de sus restricciones para abstraer el sistema**

Los datos guardados en la hoja de cálculo se ven de la siguiente manera:







En donde la columna “u” representa el vector de entradas para el sistema (Valores de manipulación por software) y la columna “y”, el vector de salidas del sistema (Rad/s).

Para el modelamiento del sistema mediante un modelo de regresión, a partir de los datos obtenidos en la hoja de cálculo, se determinaron los parámetros del conjunto de entradas en base a una ventana de tiempo correspondiente a 1.1424s (5 muestras):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **PARÁMETROS** | | | | | | | | |
| X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 | X9 |
| u(k) | u(k-1) | y(k-1) | u(k-2) | y(k-2) | u(k-3) | y(k-3) | u(k-4) | y(k-4) |

A continuación, se muestra una vista de los conjuntos de entrada y de salida generados para entrenar el modelo de regresión:







Como se puede observar, el conjunto original de datos se vio comprometido por los parámetros elegidos para el conjunto de datos de entradas para el modelo, que tiene 4 datos menos.

Debido a que los parámetros del conjunto de entradas, están correlacionados unos con otros, se utilizó el modelo de machine learning de regresión lineal “Elastic Net” de la biblioteca “Scikit Learn”.

**

A continuación, se muestra el diagrama de flujo del proceso de generación de los conjuntos de entrada y salida:



A continuación, se muestra el diagrama de flujo del proceso de entrenamiento del modelo:

Los valores máximo y mínimo de la columna “output” correspondiente a la velocidad (Rad/s) definirán las restricciones del modelo entrenado para evitar valores de salida ajenos a la planta de control real:

**Velocidad máxima:** 395.36943545427545 rad/s

**Velocidad mínima:** 0 rad/s

## **Implementación del controlador PID discreto para definir su evaluación de desempeño, en base a la integral del valor absoluto del error y su ejecución sobre el modelo de machine learning**

Se implementó un bucle que ejecuta mil veces la acción de un controlador PID discreto sobre el modelo de machine learning. Puesto que el tiempo de muestreo elegido fue de 0. 22848 s (7 ciclos del timer del microcontrolador), las mil iteraciones evalúan el desempeño del controlador durante , en base a la integral del valor absoluto del error.

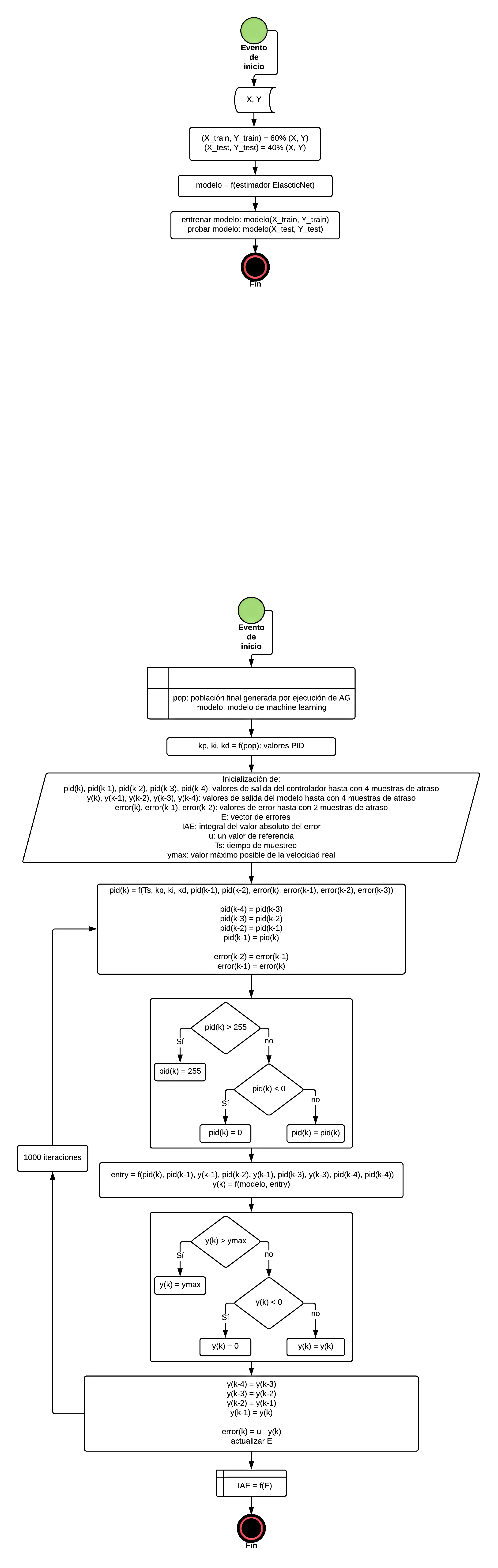
El tiempo de muestreo máximo recomendado para la implementación del controlador, según la referencia teórica, es la décima parte del tiempo requerido para la estabilización de la señal de salida del sistema: 4.896 s (150 ciclos del timer). Es decir, 0.4896 s. Por lo tanto, un tiempo de muestreo de 0.22848 es un tiempo adecuado.

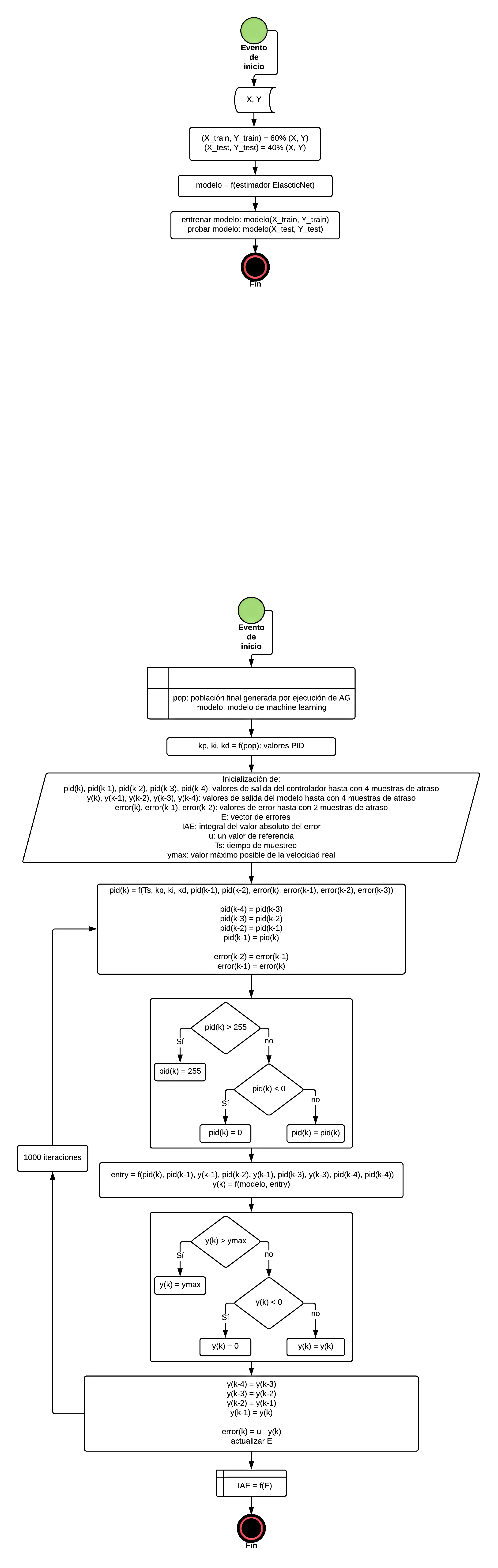
El formato de la entrada para usar el modelo de machine learning, es análoga a los parámetros del conjunto de entrada:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **u(k)** | **u(k-1)** | **y(k-1)** | **u(k-2)** | **y(k-2)** | **u(k-3)** | **y(k-3)** | **u(k-4)** | **y(k-4)** |
| pid(k) | pid (k-1) | vel(k-1) | pid (k-2) | vel (k-2) | pid (k-3) | vel (k-3) | pid (k-4) | vel (k-4) |

Donde pid se refiere al valor de salida del controlador y vel, al valor de la velocidad.

A continuación, se muestra el diagrama de flujo de la función para evaluar el desempeño del controlador:





## **Sintonización del controlador PID por medio de un algoritmo genético**

Los cromosomas (individuos) usados por el algoritmo genético serán cadenas binarias, las cuales serán decodificadas por la función de evaluación de desempeño del controlador para obtener los valores de las ganancias proporcional, integral y derivativa. Para esto, se debe definir la resolución digital de la cadena binaria de los individuos. Además, también definir los valores decimales máximos y mínimos posibles que podrían adoptar. Para este trabajo, la resolución digital escogida fue de 30 bits, con 10 bits para cada ganancia y todos los valores máximos fueron definidos en 1 y los mínimos, en 0. Por ejemplo:

**kp\_**

**ki**

**kd**

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

A continuación, se muestra el diagrama de flujo del proceso de sintonización:



# **DISCUCIÓN**

## **El sistema de control**

## **La señal de entrada para el sistema y el conjunto de datos experimentales en tiempo discreto**

## **El modelo de machine learning**

## **Implementación del controlador PID discreto sobre el modelo de machine learning y definición de su evaluación de desempeño**

## **Sintonización del controlador PID por medio de algoritmos genéticos**

# **CONCLUSIONES**

# **SUGERENCIAS**

# **BIBLIOGRAFÍA**

# **ANEXOS**