PLATAFORMA DE CONTROL CON INTERFAZ HMI DE UN BALANCÍN CON MOTOR BRUSHLESS

ALFONSO ADRIAN BERRIO RESTREPO

INGENIERIA EN ELECTRONICA

DOCENTE ASESOR

JULIAN PELAEZ RESTREPO

INSTITUTO TECNOLÓGICO METROPOLITANO

Fecha

21 noviembre de 2018

RESUMEN

Diseño e implementación de un control digital con ayuda de un Arduino UNO donde se pretende controlar la posición de un sistema mediante la herramienta Simulink de Matlab (control de posición de un balancín de 1 eje (1 solo grado de libertad)), El sistema consta de una barra anclada a modo de balancín, en uno de los extremos de la barra se encuentra un motor Brushless con una hélice que será el encargado de producir la fuerza de empuje proporcional al giro de su hélice que será la responsable de los cambios de posición de nuestro sistema. Para el control del motor se realiza una sintonización de un PI vía USB en comunicación con el software grafico de Matlab APPDESIGNER donde se diseña una Interfaz gráfica que permite realizar la toma de datos en tiempo real, la sintonización del control y la simulación del comportamiento del sistema.

Palabras clave: Motor Brushless, Control PID, APP Designer, Arduino, Potenciómetro, planta, interfaz, Matlab, Simulink.

RECONOCIMIENTOS

A MI FAMILIA.

Por su apoyo incondicional y valores para continuar y adquirir grandes logros, a mi hermana por sus consejos que me dieron fortaleza cada día, al Profesor Julián Peláez Restrepo que me apoyo y orientó y a todos los que de una u otra manera hicieron parte de este trabajo para el logro del objetivo.

¡Gracias a todos por hacerlo posible!

ACRÓNIMOS

ABREVIATURAS	***SIGNIFICADO***
CC	Corriente Continua
e	Error
GNU	Sistema Operativo de tipo Unix
GPL	General Public License
KD	Ganancia derivativa
KI	Ganancia integral
KP	Ganancia proporcional
P	Proporcional
PD	Proporcional derivativo
PI	Proporcional integral
PID	Proporcional integral derivativo
RPM	Revoluciones por minuto
SISO	Single Input-Single Output
T	Periodo de muestreo
V	Voltaje

SIMBOLO	***SIGNIFICADO***
A	Amperio
cm	Centímetro
Hz	Hertzio
I	Corriente
mA	miliamperio
ms	milisegundo
Ω	Ohmio

τ	Constante de tiempo(s)
μ	Señal de control

GLOSARIO

BALANCÍN: Pieza de distintos mecanismos, que consiste en una barra móvil alrededor de un eje para regular el movimiento.

FUNCIÓN DE TRANSFERENCIA: es un modelo matemático que a través de un cociente relaciona la respuesta de un sistema con una señal de entrada o excitación.

INRUNNER: Se refiere a un motor eléctrico donde el rotor está dentro del estator. El término se usa en particular para motores sin escobillas para diferenciarlos de los corredores que tienen su rotor fuera del estator.

OUTRUNNER: Se refiere a un tipo de motor eléctrico de CC sin escobillas que hace girar su cubierta exterior alrededor de sus devanados.

PI: algoritmo de control que contiene parte proporcional, integral.

PROCESSING: Es un lenguaje de programación y entorno de desarrollo basado en Java, de código abierto y bajo una licencia GNU GPL.

WIRING: Es una plataforma que nos permite programar y generar prototipos con electrónica.

1 TABLA DE CONTENIDO

LISTA DE FIGURAS.

LISTA DE TABLAS.

1. INTRODUCCIÓN

1.1 GENERALIDADES

1.1.1 PLANTEAMIENTO DEL PROBLEMA

El programa de ingeniería en electrónica de la institución ITM, cuenta con un laboratorio que trabaja constantemente en mantenerse a la vanguardia de la tecnología y de esta manera generar un ambiente en el cual el estudiante pueda desarrollar las competencias adecuadas que lo lleven a una formación integral.

Este trabajo de laboratorio se enfrenta al problema de diseñar una interfaz gráfica con el entorno de programación visual de Matlab Simulink y la APP DESIGNER donde permita Sintonizar, visualizar el comportamiento en tiempo real y además de eso permite realizar la toma de datos para la caracterización del sistema,

¿Es posible diseñar e implementar un sistema de control de bajo costos que cuente con una interfaz gráfica que pueda ser utilizada para realizar prácticas en los laboratorios del ITM y en clases de control?

1.1.2 OBJETIVOS

1.1.2.1 General

Diseñar e implementar un control digital con ayuda de un Arduino, realizar la sintonización del PID y el control vía USB en comunicación con algún software grafico que puede ser Matlab, labView o MyOpenLab, además dejar las guías de laboratorio que permitan caracterizar la planta, diseñar el control y finalmente sintonizar y realizar el ajuste si es necesario.

1.1.2.2 Específicos

- Diseñar y construir una interfaz gráfica con APP DESIGNER que permita Sintonizar el control del sistema.
- Conseguir la regulación de la posición de la barra, lograr que en todo momento sea la deseada.
- Lograr que el sistema alcance la referencia con el menor sobre pico posible.

- Obtener graficas de la respuesta del sistema ante las perturbaciones y valores de referencia en el tiempo.
- Buena respuesta ante perturbaciones, el sistema deberá ser capaz de corregir su posición ante la acción de fuerzas externas que puedan provocar una desviación de la posición de la barra respecto a la referencia.
- Encontrar el modelo matemático que representa el sistema.
- Controlar la interfaz para la comunicación de Matlab con Arduino.
- Aplicar el control convencional PID (proporcional integral y derivativo) con el entorno de programación visual de Matlab Simulink.
- Simular la planta de control en Simulink y verificación del funcionamiento del sistema.
- Dejar la guía de funcionamiento para futuras prácticas en el laboratorio y en clases de control.

En la institución se vio la necesidad de desarrollar un control digital que permita a futuros estudiantes observar el funcionamiento en tiempo real de una planta en este caso un balancín de bajo costo que permite obtener de forma clara y concisa conocimientos que permita desarrollar nuevas habilidades y estrategias a los estudiantes por medio de prácticas que afianzan sus conocimientos.

Para obtener los resultados se debe de revisar y seleccionar uno del software gráficos propuestos ya que puede que no todos tengan las características que se necesitan para el funcionamiento del sistema o sea más complejo su funcionamiento.

Implementar y ajustar el algoritmo de control convencional PI (proporcional integral), PID (Proporcional integral derivativo) para regular la velocidad del motor Brushless, Los controladores PID son los más usados en los sistemas de control de la industria, pero se aprecia más su utilidad cuando el modelo de la planta a controlar no se conoce y los métodos analíticos no pueden ser empleados, el controlador PID recibe una señal de entrada generalmente es el error e(t) y proporciona una salida (acción de control u(t)).

Se debe simular la planta de control y verificación del funcionamiento del sistema. El módulo Arduino Uno se comunica con un computador mediante un sistema de adquisición de datos, para poder leer el sensor (Potenciómetro Lineal) y actuar sobre la planta, esto mediante una interfaz desarrollada en APP DESIGNER de Matlab, el cual permite implementar estrategias de control como PI y PID.

Los diseños, simuladores e implementaciones presentados en este trabajo han demostrado la efectividad de la construcción de una planta, que permite al estudiante afianzar sus conocimientos y experiencias con base en la implementación de sus diseños de técnicas de control.

2. MARCO TEÓRICO

2.1 DISEÑAR UNA PLATAFORMA DE CONTROL CON INTERFAZ HMI DE UN BALANCIN CON MOTOR BRUSHLESS.

2.2 DESCRIPCIÓN DE LA PLANTA.

La planta que se usara en este trabajo, se encuentra confeccionada en madera, posee un acabado muy bien elaborado, y en cuanto a peso, resulta relativamente ligero para trabajar. Consta con tres piezas principales, que se encuentran unidas por un eje, y una base plana sobre la cual se encuentran empernadas de manera completamente fija. Las dimensiones de las piezas son las siguientes: El balancín cuenta con una longitud de 52cm x 3cm, mientras que los soportes que lo contienen miden 70 cm x 3 cm. La base mide 40cm x 30cm. Perfectamente alineado al eje, se encuentra un potenciómetro lineal, en cual se encarga de sensor de la posición angular del balancín, y es accionado por perfiles de madera que se encuentran unidos de manera paralela al balancín, con lo que se logra una medida más precisa, cabe resaltar que este valor es muy importante en el control, porque mientras más preciso sea, podemos realizar mediciones con mayor precisión también Consta de un motor Brushless, un ESC y una hélice.

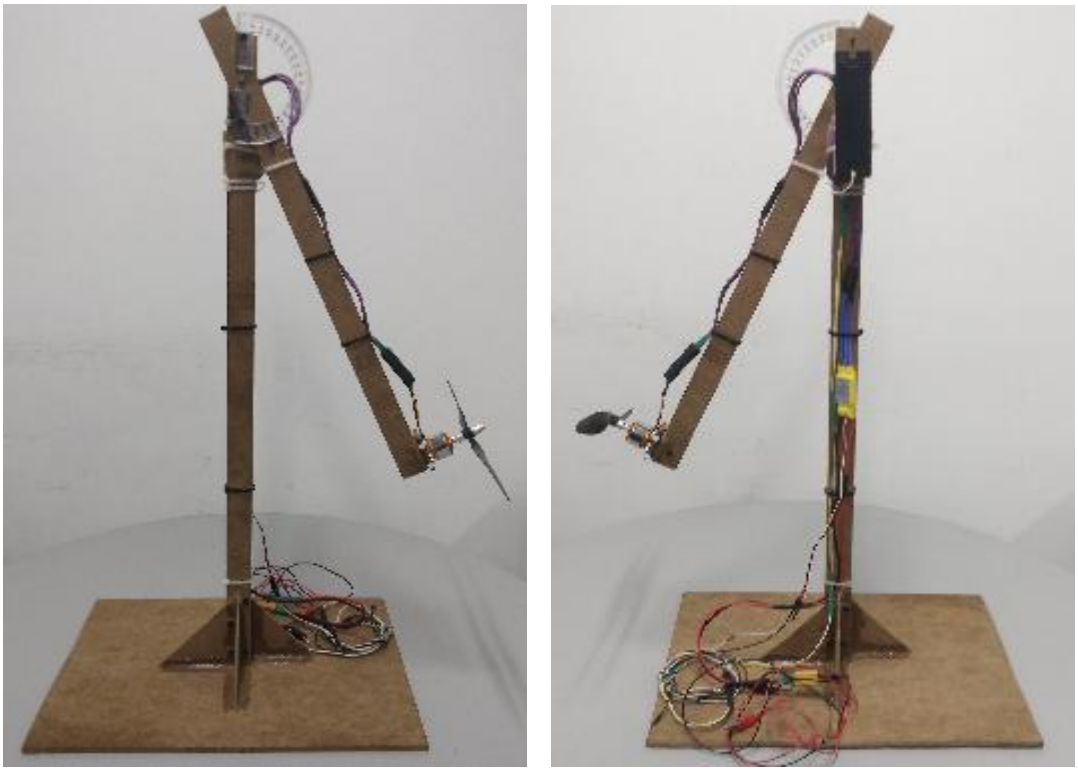

Figura 52 Planta Balancín

Fuente: Autor

2.3 SENSOR DE POSICION ANGULAR O SENSOR POTENCIOMETRICO.

Un potenciómetro es un dispositivo electrónico compuesto por una resistencia variable en la que se puede cambiar su valor de forma manual. El valor de la resistencia irá variando en función de la posición del potenciómetro. Se trata de un componente sencillo y robusto, que proporciona una señal de precisión en función de la calidad del potenciómetro que se tenga. Dispone de tres terminales: X, Y y Z. Los terminales X y Z se conectan a una fuente de tensión (terminales positivo y neutro) mientras que el tercer terminal Y está conectado a una resistencia. El terminal Y es el que controla el usuario y le permite ajustar la posición del mismo a lo largo de la resistencia, proporcionando una tensión variable en función del valor de la resistencia.

Para registrar la posición de la barra, como ya se había comentado anteriormente, se eligió un potenciómetro que es el responsable de recopilar y entregar un valor de posición angular, se seleccionó este resistor variable de manera que la variación de resistencia en función de la posición del cursor sea lineal, de esta manera tendremos una ley de variación matemáticamente sencilla ya que para cada posición del cursor habrá un valor diferente de resistencia R(Ω) y con ello de tensión, este último parámetro será el registrado mediante la lectura de las entradas analógicas del microcontrolador, se encuentra alineado al eje del balancín, y que varía a medida el balancín cambie su posición angular. El potenciómetro seleccionado tiene un valor nominal de 1kΩ, si dividimos la tensión de alimentación empleada para dar energía a este sensor entre valor nominal de resistencia, obtendremos en cuanto variará la tensión leída por cada ohm que se incremente debido al desplazamiento del cursor, en otras palabras, la resolución, que indicará cuanto debe variar la posición de la barra para notar algún cambio en la lectura del valor indicado por el sensor, si hacemos el cálculo:

$$Resistencia = \frac{V_{cc}}{R} = \frac{5V}{1K\Omega} = {}^{0.005}/_{K\Omega}$$

Trabajar con este valor de Resolución resulta conveniente debido a que los mínimos cambios registrados en el potenciómetro podrán ser registrados.

Figura 53 Potenciómetro como sensor de posición angular.

Fuente: Autor

2.4 PLACA ARDUINO.

Figura 54 Placa Arduino Uno Utilizada.

Fuente: Autor

Es una plataforma de prototipos electrónica con hardware y software flexibles de código abierto, basada en una placa con un microcontrolador y un entorno de desarrollo, facilita el uso de la electrónica en proyectos interactivos. El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida.
Arduino permite de una forma rápida y económica desarrollar un proyecto con un microcontrolador ya que la placa está totalmente montada, con un bootloader listo para conectar por USB al ordenador. También están disponibles en internet una gran cantidad de librerías para trabajar con Arduino, lo cual ayuda a que no sea necesario un gran conocimiento previo de programación para poder utilizarlo. Debido a su popularidad y gran uso, es fácil encontrar diferentes placas de Arduino de todos los precios, lo que lo hace un dispositivo de fácil acceso para cualquier persona.

Por otro lado, el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring, el lenguaje estándar es C++, aunque es posible programarlo en otros lenguajes. No es un C++ puro, sino que es una adaptación que proveniente de avr-libc que provee de una librería de C de alta calidad para usar con GCC en los microcontroladores AVR de Atmel y muchas funciones específicas para los MCU AVR de Atmel y el cargador de arranque que es ejecutado en la placa. Se programa en el ordenador para que la placa controle los componentes electrónicos. Arduino puede tomar información del entorno a través de sus entradas analógicas y digitales, puede controlar luces, motores y otros actuadores.

El microcontrolador en la placa se programa mediante el lenguaje de programación (basado en Wiring) y el entorno de desarrollo (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador. También cuenta con su propio software que se puede descargar de su página oficial que ya incluye los drivers de todas las tarjetas disponibles lo que hace más fácil la carga de códigos desde el computador. se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software, Una tendencia tecnológica es utilizar la placa como tarjeta de adquisición de datos desarrollando interfaces, El entorno de desarrollo integrado libre se puede descargar gratuitamente.

2.5 MOTOR BRUSHLESS 1000 KV.

El motor brushless o motor sin escobillas es un motor eléctrico de corriente continua que, como su propio nombre indica, no utiliza escobillas para el cambio de polaridad, sino que emplea un circuito electrónico externo conocido como variador electrónico o ESC (Electronic Speed Controller). En este tipo de motores, los imanes permanentes se encuentran en la parte móvil o rotor, mientras que el bobinado se encuentra en la parte fija o estator, justo al contrario a como ocurre en los motores de corriente continua con escobillas. El hecho de no usar escobillas hace que no sea necesario el rozamiento entre el rotor y la parte fija del motor, de tal forma que aumenta la eficiencia ya que se produce menor pérdida de calor, aumentando con ello el rendimiento con un menor consumo de potencia, y permite un rango de velocidad elevado al no tener limitaciones mecánicas. La ausencia de escobillas disminuye también el mantenimiento, así como el ruido electrónico (menor interferencia en otros circuitos). Como desventaja se tiene un mayor costo de construcción, así como la necesidad para su funcionamiento y control del ESC, un circuito caro y complejo, aumentando el precio del conjunto. Al ser un tipo de motor muy común en el mundo del radiocontrol y el aeromodelismo, se dispone de una gran variedad tanto de motores brushless como de ESC en el mercado, con diferentes características y precios.

En su interior están compuestos por dos partes diferenciables, la parte móvil que gira, conocida como rotor, compuesta por imanes permanentes, y una parte fija, denominada estator, donde se encuentran las bobinas de hilo conductor por el que circula la electricidad. Dependiendo de la posición relativa entre estator y rotor puede haber dos tipos de motores brushless: inrunner o outrunner. En los outrunner la parte que gira o rotor es la parte exterior del motor, la cual porta los imanes permanentes, y es en la zona interior donde se encuentran las bobinas conductoras fijas formando el estator. En el caso de los inrunner ocurre, al contrario, el bobinado fijo se encuentra en la parte externa del motor rodeando al rotor, que se encuentra en la parte interna junto con los imanes.

El funcionamiento del motor brushless se basa en que la corriente eléctrica que pasa por el bobinado del estator genera un campo magnético que interacciona con el campo magnético de los imanes permanentes del rotor, y con ello se produce el movimiento de giro de éste.

El bobinado está formado por tres bobinas colocadas en un desfase en la posición de 120 grados. La corriente eléctrica que le llega al motor procede del variador electrónico o ESC y le llega a través de tres cables que conectan con las tres bobinas, de tal forma que se aplica tensiones en cada bobina con polaridades desfasadas 120 grados de forma secuencial, haciendo girar el motor a la velocidad que genera el variador. Es decir, el ESC aplica voltajes secuencialmente sobre las bobinas y los imanes del rotor siguen el campo magnético que generan las bobinas. La velocidad de la secuencia se denomina frecuencia y se mide en Hz, de tal forma que, a mayor frecuencia, mayor velocidad de giro del rotor el motor que se utilizó.

Motor: A2212/13T
KV:1000
Corriente sin carga:10V-0.5A.
Corriente máxima: 12A/60segundos
ESC: 30 A.
RPM/V: 1000RPM/V.
Eficiencia Máxima: 80% en 4-10A (>75%).
Resistencia Interna: 90mΩ.
Dimensiones: 27.5x30mm.
Diámetro del eje 3.175mm.

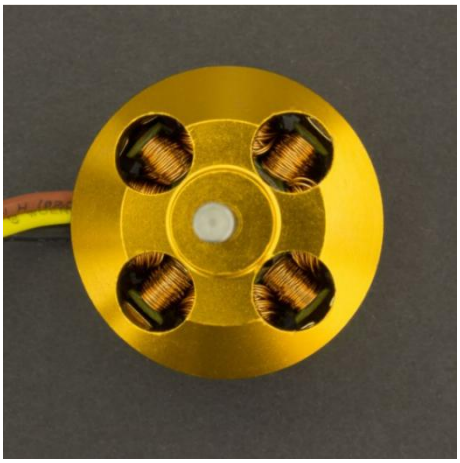

Figura 55 Motor Brushless.

Fuente: Wikipedia

Figura 56 Motor Brushless utilizado.

Fuente: Autor

2.6 ESC (Electronic Speed Control).

las características del control de un motor brushless es que requiere de un circuito electrónico que lo haga funcionar. Este circuito electrónico se conoce como variador o ESC (Electronic Speed Controller), al cual se le envía una señal de información de tipo modulación por ancho de pulso, o PWM (Pulse Width Modulation), para controlar la velocidad de giro del motor. El ESC se conecta por un lado al motor a través de tres cables, y por otro a la fuente de alimentación y al dispositivo que le suministra la señal de información PWM de velocidad de motor deseada (en este caso, el dispositivo que genera la señal PWM es un Arduino). Al ser un motor brushless, la fuente de alimentación va conectada al ESC y no directamente al motor, como ocurre en motores de corriente continua, de forma que el ESC es el encargado de proporcionar la potencia necesaria al motor, en función de la señal de control que recibe.

El ESC es un control de velocidad electrónico de alta potencia, que controla y regula la velocidad de un Motor eléctrico, también puede proporcionar la inversión del motor y el frenado dinámico mediante una señal de control, que vendrá del microcontrolador, y una alimentación, que será de hasta 12 V, cabe resaltar que, al tratarse de un alto número de revoluciones y torque, el ESC debe tener la capacidad de aguantar hasta 30 A de corriente.

Un ESC suele contar con siete conexiones necesarias para el correcto funcionamiento del motor: tres cables de salida que van conectados directamente al motor y son los que le proporcionan la señal de control, dos cables de alimentación usados para conectar la fuente de alimentación o batería (positivo y negativo), un cable de señal de información por donde recibe la señal PWM de control de velocidad del motor, y otro cable para la toma de tierra.

Sin entrar en el funcionamiento de las señales de control tipo PWM, se indica que la señal que debe recibir el ESC para que se produzca movimiento del motor se caracteriza por tener un pulso de duración que puede variar entre 1 y 2 milisegundos y una amplitud de 0 o 5 Voltios. La variación de la longitud del ancho de pulso dentro de este rango es lo que controla la velocidad del motor, trasmitiéndole más o menos potencia. Cuando la duración del ancho de pulso es menor o igual a 1ms el motor permanece inmóvil ya que corresponde a la velocidad mínima del motor, y conforme aumenta la duración del pulso aumenta la velocidad de giro del motor, hasta un máximo cuando alcanza los 2 ms. Esta señal PWM la genera la placa de Arduino y es recibida por el ESC y no el motor. La señal de potencia que recibe el motor tiene unas características diferentes y dependerá de la señal PWM y del voltaje de la fuente de alimentación o batería que se conecte al ESC. En definitiva, el variador o ESC es el dispositivo electrónico que se encarga de suministrar la señal de potencia necesaria al motor en función de la señal PWM recibida y de la potencia que le suministra la fuente de alimentación, haciendo que éste gire con mayor o menor velocidad.

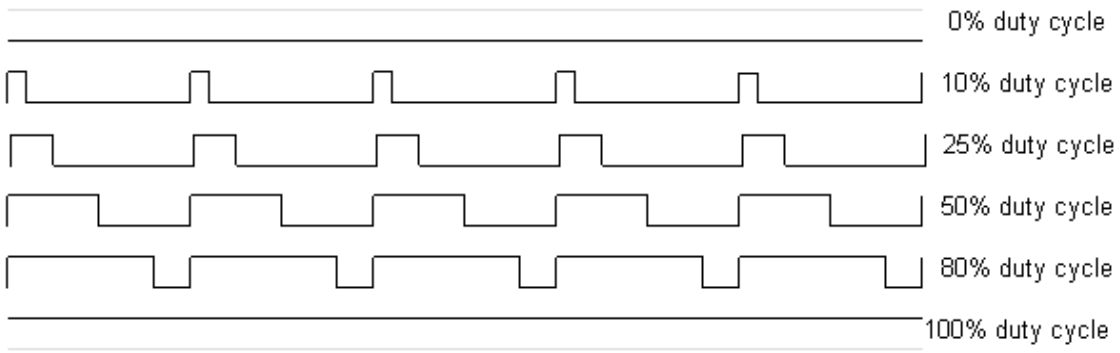

Figura 57 Diferentes señales de Duty Cycle.

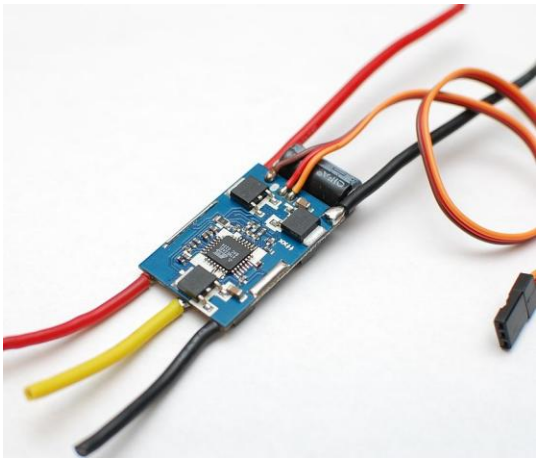

Figura 58 Modulo ESC Genérico clasificado a 35 Amperios.

Fuente: Wikipedia

Figura 59 Módulo ESC Utilizado de 30Amperios.

Fuente: Autor.

2.7 SIMULINK.

Es una toolbox especial de MATLAB que sirve para simular el comportamiento de los sistemas dinámicos. Puede simular sistemas lineales y no lineales, modelos en tiempo continuo y tiempo discreto y sistemas híbridos de todos los anteriores. Es un entorno gráfico

de programación visual en el cual el modelo a simular se construye arrastrando los diferentes bloques que lo constituyen, Es un entorno de programación de más alto nivel de abstracción que el lenguaje interpretado Matlab (archivos con extensión .m). Simulink genera archivos con extensión. mdl (de "model"). Al ejecutar un modelo se genera un código en C que el ordenador reconoce y ejecuta.

Para implementar el control de la planta será necesario utilizar varios bloques de Simulink que permitirán la comunicación y la ejecución de algoritmos de control, veamos cuales son eso bloques a continuación:

El paquete de soporte de Simulink® para Arduino® Hardware le permite crear y ejecutar modelos de Simulink en placas Arduino. El paquete de soporte incluye una biblioteca de bloques de Simulink para configurar y acceder a los sensores, actuadores e interfaces de comunicación de Arduino. También le permite monitorear y ajustar interactivamente los algoritmos desarrollados en Simulink mientras se ejecutan en Arduino.

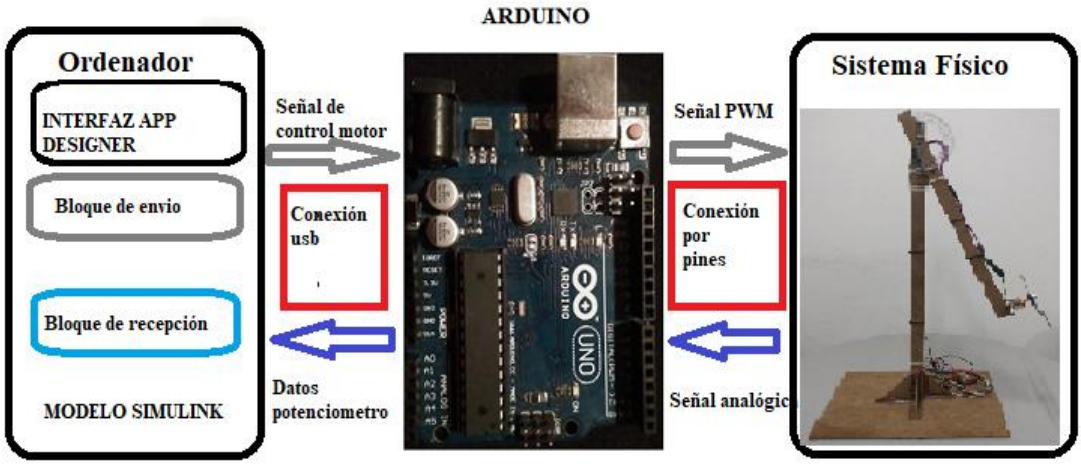

Figura 60 Esquema de Información entre PC-Arduino y Sistema.

Fuente: Autor.

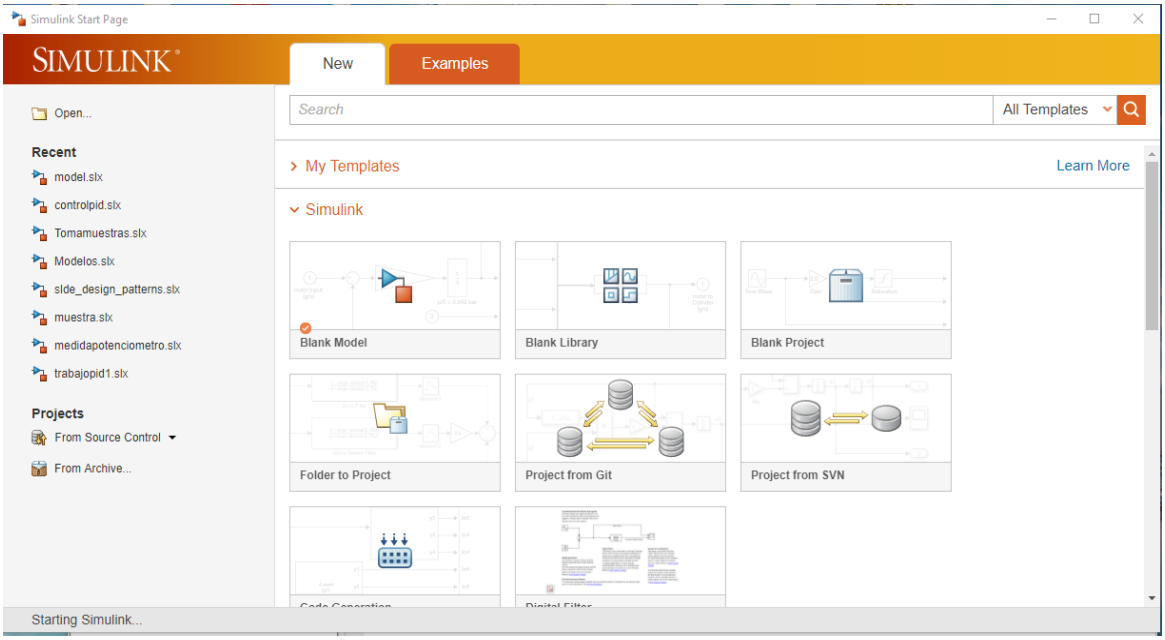

Figura 61 Entorno Simulink de Matlab R2018b.

Fuente: Matlab-Simulink.

2.8 VERSIÓN DE MATLAB UTILIZADA.

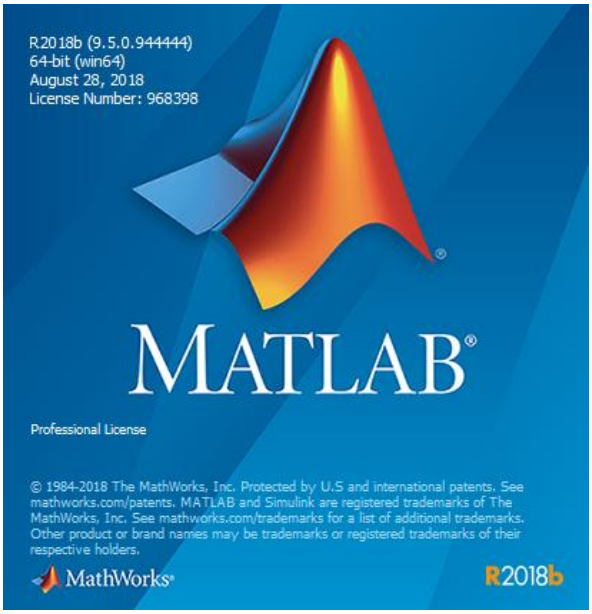

Figura 62 Versión Matlab R2018.

Fuente: Matlab

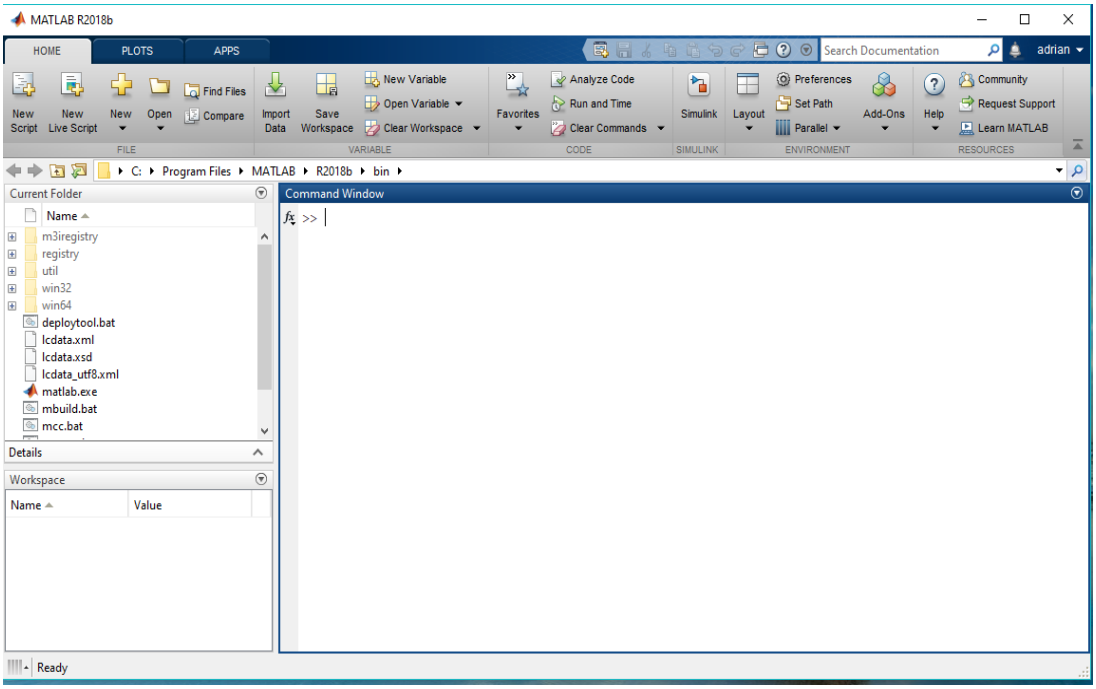

Figura 63 Entorno Matlab Utilizado.

Fuente: Matlab.

2.8.1 MATLAB.

Es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M).

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets).

2.9 FUENTE DE ALIMENTACION.

Para suministrar al motor la potencia eléctrica necesaria se necesita una batería o fuente de corriente continua que irá conectada al ESC. Se puede utilizar una fuente de corriente continua en vez de una batería lipo o similar. Aunque en este caso se utilizara una Batería para que se pueda utilizar de forma portable, la fuente de corriente tiene como ventaja que proporciona una corriente continua estable y constante durante todo el tiempo, pudiendo

regular el voltaje que se le suministra al ESC a través de un interruptor, mientras que la desventaja de las baterías es que pueden perder potencia conforme se van descargando. Esta batería está conectada al ESC a través de dos cables, positivo y negativo.
Se utilizó una fuente de Alimentación MTEK MT12180HR 12V20A

Figura 64 Fuente de Alimentación.

Fuente: Autor.

2.10 APP DESIGNER.

Es un entorno de desarrollo que proporciona vistas de diseño y código, una versión totalmente integrada del editor MATLAB® y un gran conjunto de componentes interactivos.

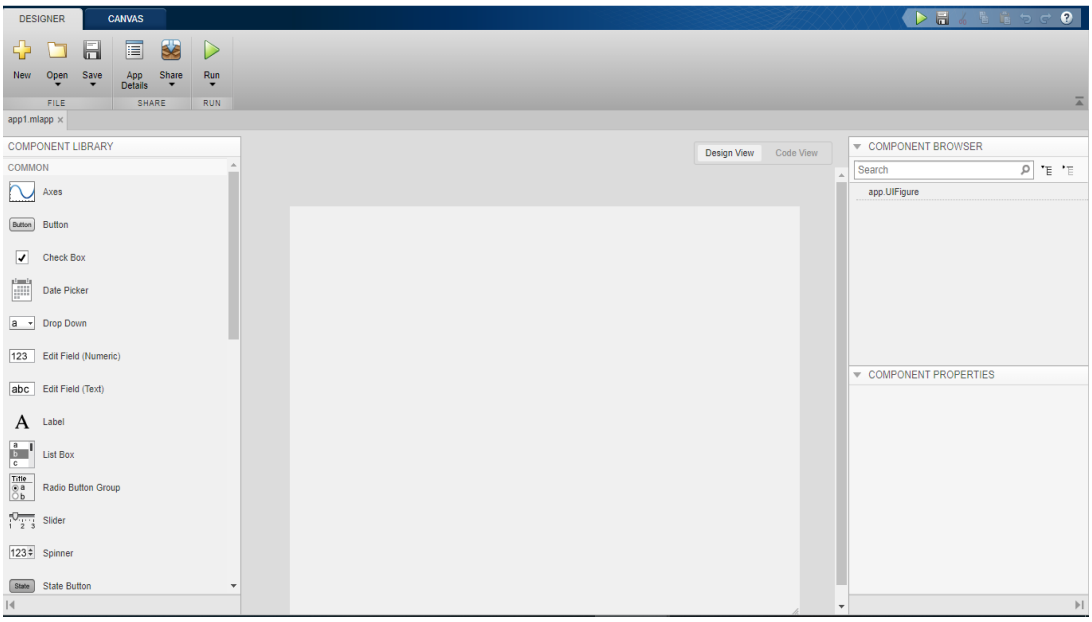

Figura 65 Entorno APP DESIGNER.

Fuente: Matlab.

2.10.1 Componentes de APP.

Cuenta con componentes estándar como botones, casillas de verificación y listas desplegables, ofrece elementos de control como medidores, indicadores luminosos, controles y conmutadores que le permiten replicar el aspecto y las acciones de los paneles de instrumentación. Utilice gráficos 2D y 3D, así como tablas, para presentar resultados en su app. También puede utilizar componentes de contenedor, como pestañas y paneles, para organizar la interfaz de usuario.

2.10.2 la App.

Este diseñador de aplicaciones integra las dos tareas principales de la creación de una aplicación: la organización de los componentes visuales y la programación del comportamiento de la aplicación. Simplemente arrastre los componentes visuales y colóquelos en el diseño y utilice las guías de alineación para lograr un diseño preciso. La App genera automáticamente código orientado a objetos que especifica la distribución y el diseño de la aplicación.
Se puede ver un ejemplo: Tutorial – Creación de una app sencilla con App Designer.

2.11 CONEXIONES DE CONTROL MOTOR BRUSHLESS.

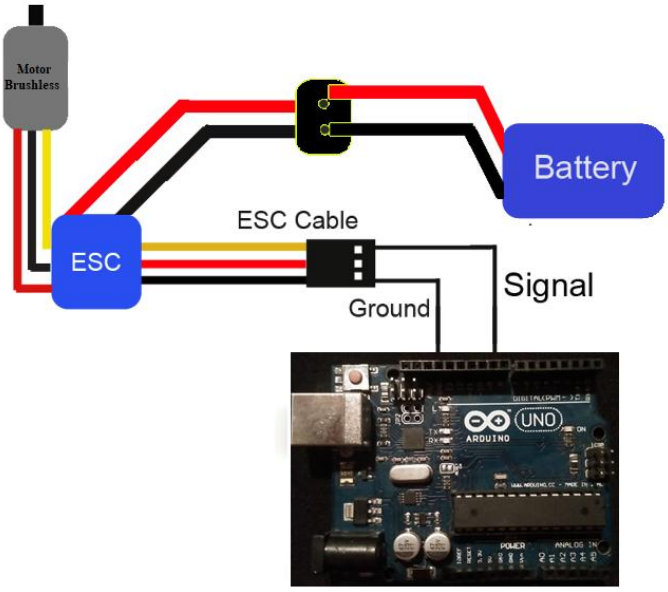

Figura 66 Conexiones Control Motor.

Fuente: Autor.

2.12 SISTEMAS DE CONTROL.

Se considerarán dos tipos de clasificación para los sistemas de control.
Los sistemas de control en lazo abierto y los sistemas de control en lazo cerrado:

- Sistemas de control en lazo abierto (sistemas no realimentados): Aquellos en los que la variable de salida (variable controlada) no tiene efecto sobre la acción de control (variable de control).
- Sistemas de control en lazo cerrado (sistemas realimentados): Aquellos en los que la señal de salida del sistema (variable controlada) tiene efecto directo sobre la acción de control (variable de control).

La realimentación, como se puede observar en la Figura 16, consiste básicamente en una conexión desde la salida hacia la entrada del sistema para eliminar parte de las carencias que tienen los sistemas de control no realimentado. Para obtener un control más exacto, la señal controlada y debe ser realimentada y comparada con la referencia, y se debe enviar una señal actuante a través del sistema para corregir el error, e.

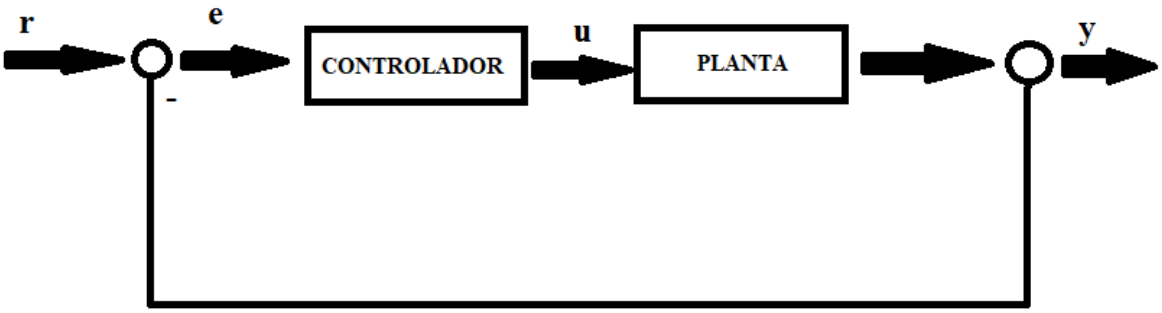

Figura 67 Sistema de control lazo cerrado.

Fuente: Autor.

El tipo de controlador en lazo cerrado más utilizado en la industria es el controlador PID.

2.13 SISTEMAS DE CONTROL PID.

El controlador PID es un controlador realimentado cuyo propósito es hacer que el error en estado estacionario, entre la señal de referencia y la señal de salida de la planta, sea cero.

Un PID consta de 3 parámetros o constantes:

- Proporcional: Se puede ajustar como el valor de la ganancia del controlador.
- Integral: Indica la velocidad con la que se repite la acción proporcional.
- Derivativa: Se manifiesta cuando hay un cambio en el valor absoluto del error.

Las salidas de estos tres términos son sumadas para calcular la salida del controlador PID.

$$y(t) = k_p e(t) + k_i \int_0^t e(t)dt + k_d \frac{de(t)}{dt} \qquad (1)$$

La función de transferencia se puede expresar, en el dominio s, como:

$$G_c(s) = k_p + \frac{k_i}{s} + k_d s \qquad (2)$$

En la tabla 1 se puede observar cómo afecta el aumento de cada una de las ganancias en lazo cerrado a aspectos tan importantes como son el tiempo de subida, sobre-pico, tiempo de establecimiento y error.

Respuesta lazo cerrado	Tiempo de subida	Sobre-pico	Tiempo de establecimiento	Error
k_p	Disminuye	Aumenta	Poco cambio	Disminuye
k_i	Disminuye	Aumenta	Aumenta	Elimina
k_d	Poco cambio	Disminuye	Disminuye	Poco cambio

Tabla 2 Características de las constantes del control PID.

2.13.1 Acción Proporcional.

La ley de un controlador proporcional está definida por:

$$u(t) = u_b + Ke(t)$$

donde u_b se relaciona con el offset considerando la ley de control de la ecuación (1). Se suele fijar como:

$$u_b = \frac{u_{max} + u_{min}}{2}$$

Y generalmente toma un valor de cero. Así, en el dominio de Laplace, en condiciones iniciales nulas, la función de transferencia del controlador proporcional es:

$$G(s) = K = \frac{U(s)}{E(s)} \qquad (3)$$

Un aumento del valor fijo de la ganancia proporcional con llevará los efectos siguientes sobre el lazo de realimentación:

- Reducción del crecimiento de la respuesta hacia la consigna si el proceso es lineal con ganancia canónica positiva
- Reducción del error en estado estable u offset, ya que, cuanto mayor es la ganancia proporcional, menor será la señal de error que actúa.

2.13.2 Acción Integral.

La acción de control integral genera una señal de control proporcional a la integral de la señal de error:

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

que en términos de la variable compleja implica una función de transferencia:

$$G(s) = \frac{U(s)}{E(s)} = \frac{k_i}{s}$$

Esta acción elimina el offset, pero se obtiene una mayor desviación de la señal de referencia, la respuesta es más lenta y el periodo de oscilación es mayor que en el caso de la acción proporcional.

2.13.3 Acción Derivativa

La acción de control derivativa genera una señal de control proporcional a la derivada de la señal de error:

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

que en términos de la variable compleja implica una función de transferencia:

$$G(s) = \frac{U(s)}{E(s)} = k_d s$$

La acción de control derivativa tiene el efecto de incrementar la estabilidad relativa del sistema, al reducir al sobre-impulso y mejorar la respuesta transitoria, pero en cambio no actúa un régimen permanente ya que la derivada de una señal de error constante es nula. Si tiene demasiada influencia sobre la acción global de regulación, entonces la respuesta del sistema puede ser excesivamente lenta.

2.14 ELECCIÓN DEL CONTROLADOR.

La primera decisión en el diseño de un sistema de control PID es la elección del controlador. Posteriormente se ajustarán los parámetros del mismo. A una buena elección del tipo de controlador a emplear (P, PI, PD o PID) ayudarán las consideraciones siguientes:

- **Controlador P:** En ciertos tipos de procesos es posible trabajar con una ganancia elevada sin tener ningún problema de estabilidad en el controlador. Muchos procesos que poseen una constante de tiempo dominante o son integradores puros caen en esta categoría. Una alta ganancia en un controlador P significa que el error en estado estacionario será pequeño y no se necesitará incluir acción integral.

- **Controlador PD**: El control PD puede ser apropiado cuando el proceso a controlar incorpore ya un integrador. También son válidos procesos en que es posible trabajar con ganancias elevadas en el controlador sin que sea necesario introducir la acción integral. La acción derivativa es sensible al ruido, ya que a altas frecuencias tiene una ganancia relativamente elevada. Por lo tanto, en presencia de altos niveles de ruido se debe limitar dicha ganancia, o prescindir de la acción derivativa.

- **Controlador PI:** Es la estructura más usual del controlador. La introducción de la acción integral es la forma más simple de eliminar el error en régimen permanente. Otro caso en el que es común utilizar la estructura PI es cuando el desfase que introduce el proceso es moderado. La acción derivativa, más que una mejora en esta situación es un problema ya que amplifica el ruido existente. También se recomienda la acción PI cuando hay retardos en el proceso, ya que, es este tipo de procesos la acción derivativa no resulta apropiada en este tipo de sistemas. Un tercer caso en el que se debería prescindir de la acción derivativa es cuando el proceso está contaminado con niveles de ruido elevados.

- **Controlador PID:** La acción derivativa suele mejorar el comportamiento del controlador, ya que permite aumentar las acciones proporcional e integral. Se emplea para mejorar el comportamiento de procesos que no poseen grandes

retardos pero que sí presentan grandes desfases. Este es el caso típico de procesos con múltiples constantes de tiempo.

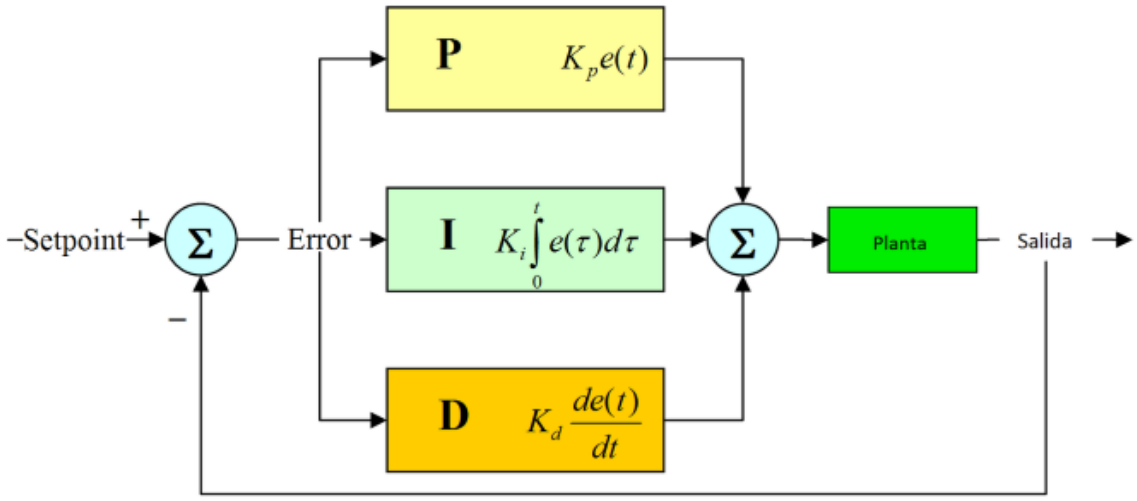

Figura 68 Control PID.

Fuente: Wikipedia.

3. METODOLOGÍA

5.4 INSTALAR SOPORTE PARA HARDWARE ARDUINO

Puede agregar soporte para hardware Arduino® al producto Simulink®. Cuando complete este proceso y reemplace el firmware, puede ejecutar los modelos de Simulink en el hardware de Arduino.
El proceso de instalación agrega los siguientes elementos a su computadora host:
Herramientas de desarrollo de software de terceros, como el software Arduino con Mega 2560, Uno, Nano 3.0, Due, Leonardo, Mega ADK, Micro, Tarjeta de control de robot, Tarjeta de motor de robot y Tarjeta MKR1000.
Una biblioteca de bloques de Simulink para configurar y acceder a los sensores, actuadores e interfaces de comunicación de Arduino.

En la pestaña Inicio de MATLAB®, en la sección Entorno, seleccione Complementos> Obtener paquetes de soporte de hardware.

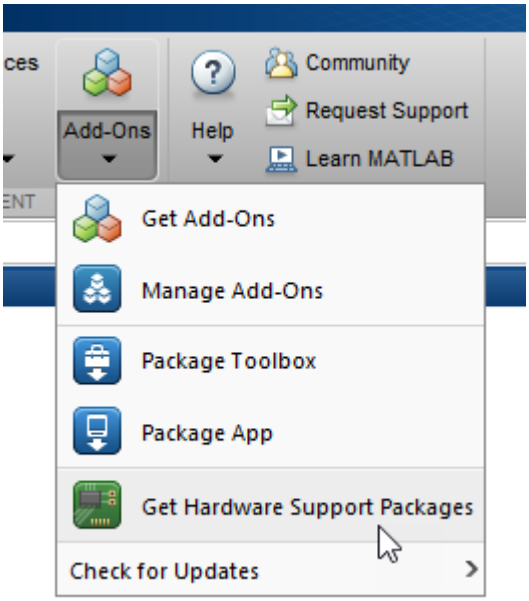

Figura 69 Get Hardware Support Package Pestaña de inicio Matlab.
Fuente: Matlab.

En la ventana del explorador de complementos, haga clic en el paquete de soporte y luego haga clic en Instalar.

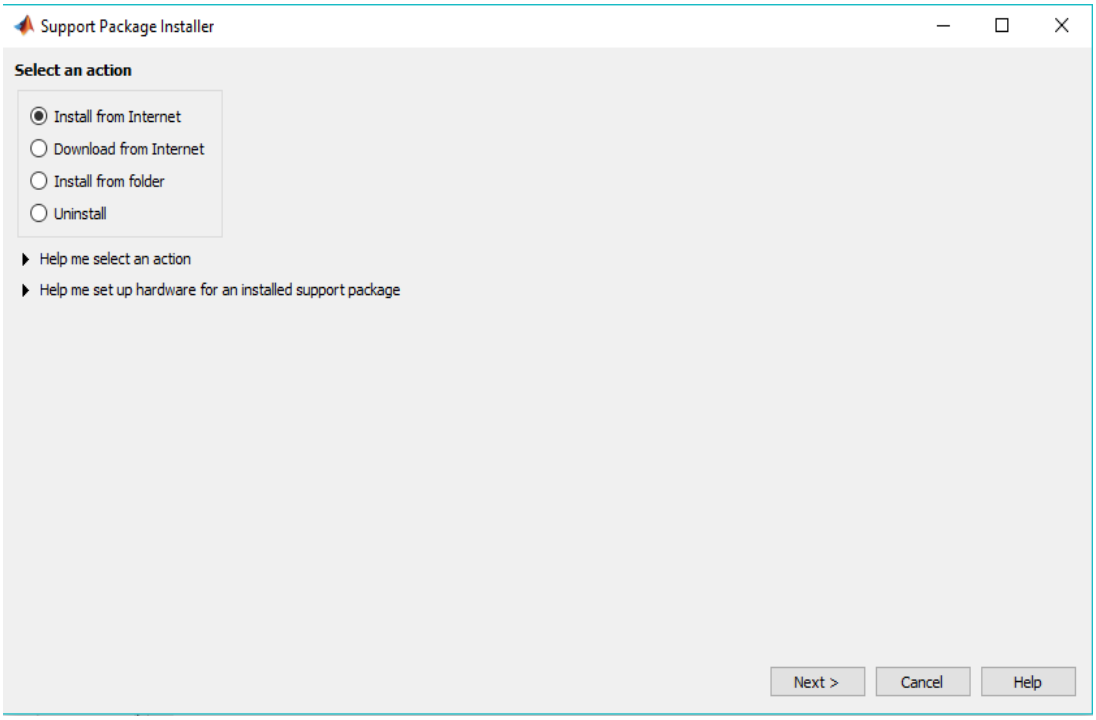

Figura 70 instalar Simulink Support Package for Arduino Hardware.

Fuente: Matlab.

Las tarjetas de hardware y los dispositivos compatibles con MathWorks® requieren pasos adicionales de configuración para conectarse a MATLAB y Simulink. Cada paquete de soporte proporciona un proceso de configuración de hardware que lo guía a través del registro, la configuración y la conexión a su placa de hardware.
Si el paquete de soporte ya está instalado, puede iniciar la configuración del hardware abriendo el Administrador de complementos.

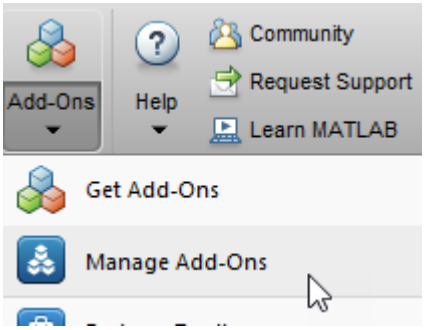

Figura 71 Manage Add-Ons Barra Superior Matlab.

Fuente: Matlab.

En el Administrador de complementos, inicie el proceso de configuración del hardware haciendo clic en el botón Configurar, después de comenzar, la ventana Configuración de

hardware proporciona instrucciones para configurar el paquete de soporte para que funcione con su hardware.

Figura 72 Add_Ons Manage.

Fuente: Matlab.

5.4.1 Biblioteca de bloques abierta para hardware Arduino:

Puede abrir la biblioteca de bloques para su hardware Arduino® desde la ventana de comandos de MATLAB® o desde el navegador de la biblioteca Simulink®.

Los bloques en esta biblioteca de bloques brindan soporte para varios periféricos disponibles en el hardware Arduino, desde la línea de comando.

También se puede utilizar el comando `arduinorootlib` desde Matlab y este abrirá la biblioteca de bloques correspondiente.

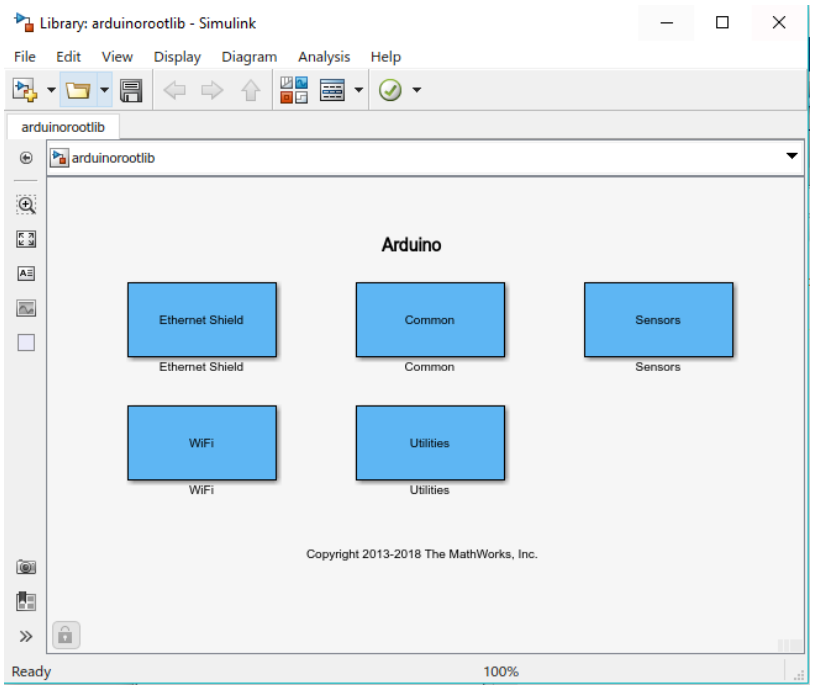

Figura 73 Biblioteca de bloques hardware Arduino.

Fuente: Matlab.

Desde el navegador de la biblioteca Simulink
Para abrir la biblioteca de bloques desde el navegador de la biblioteca Simulink:
Ingrese Simulink en la ventana de comandos de MATLAB, o haga clic en el siguiente icono en la barra de herramientas de MATLAB.

Figura 74 Icono Librería de Simulink.

Fuente: Matlab.

En el Explorador de bibliotecas de Simulink, haga clic en Paquete de soporte de Simulink para hardware Arduino.
El navegador de la biblioteca Simulink muestra la biblioteca de bloques correspondiente.

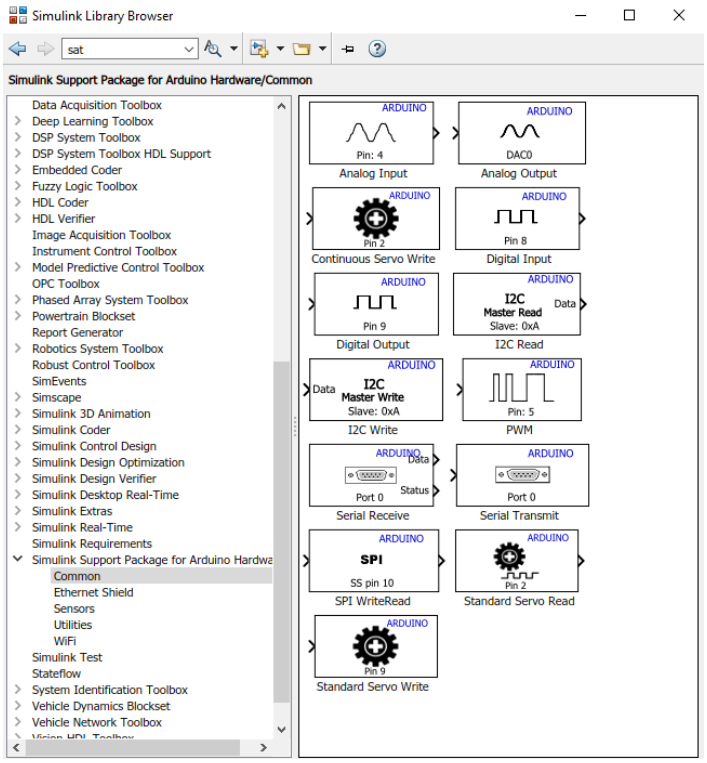

Figura 75 Paquete Common Arduino en simulink.

Fuente: Matlab.

La librería de Soporte de Simulink para Hardware Arduino cuenta con cuatro paquetes más como Ethernet Shield, Sensores, Utilities y Wifi los cuales no se tomarán en cuenta en este trabajo, pero sirve para trabajos futuros.

5.4.2 Como Configurar el puerto COM del host manualmente:

El software Simulink® detecta automáticamente la configuración del puerto COM de la conexión USB entre su computadora host y el hardware Arduino®. Opcionalmente, también puede configurar estos ajustes manualmente.

5.4.2.1 Configurar el puerto COM en Windows Manualmente:

Haga clic en el menú Herramientas en el modelo y vaya a Ejecutar en hardware de destino> Preparar para ejecutar ...

En el panel Implementación de hardware, vaya a Recursos de hardware de destino> Conexión de la placa de host. Cambie el parámetro de conexión Host-board a Manualmente y deje abierto el cuadro de diálogo Parámetros de configuración.

Abre Administrador de dispositivos en Windows.

Navegue a Puertos (COM y LPT), expanda la lista y localice su dispositivo de hardware Arduino. El nombre del tablero aparece como <nombre de su tablero> <(COMx). x es el número de puerto en su computadora. Este número debe coincidir con el número de puerto COM.

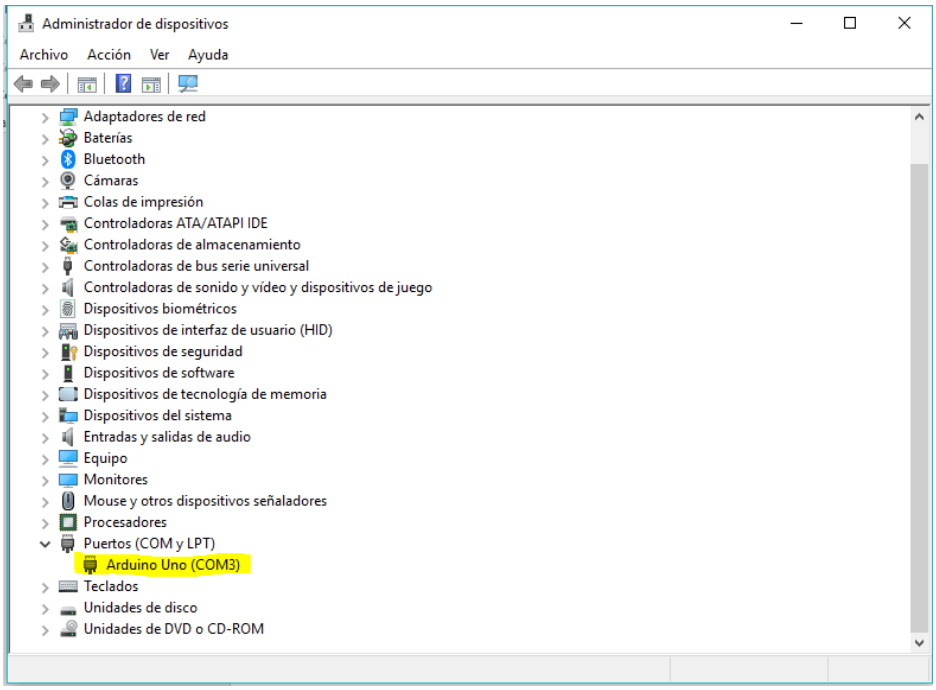

Figura 76 Administrador de dispositivos en Windows.

Fuente: Windows.

Haga clic en la barra de herramientas Tools y seleccione Run on Target Hardware.
Seleccione Options y el abrirá la configuración de parámetros.
Haga click en la pestaña Hardware Implementation, abre Target Hardware resources en Host-board connection-Set host COM port y cambiar a manual. Se ingresa el COM port number.

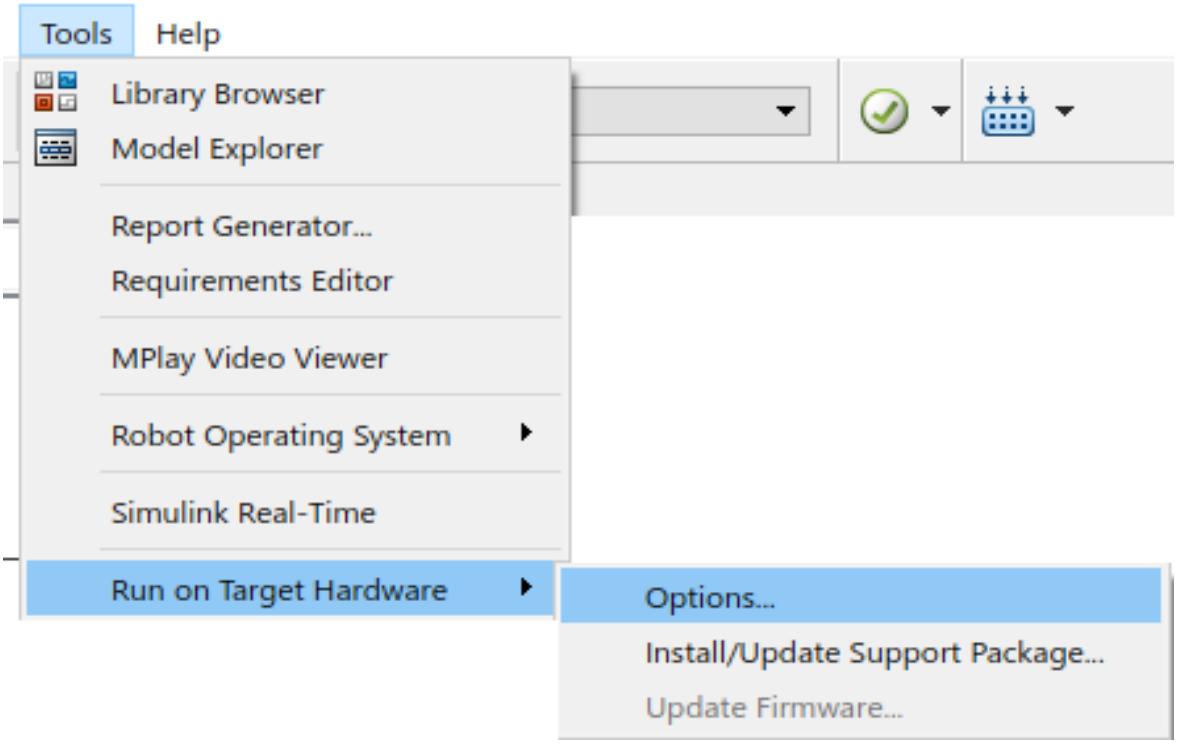

Figura 77 Tools en la Barra de herramientas de Matlab.

Fuente: Simulink-Matlab.

En la ventana de configuración de Parámetros actualice la conexión de la placa de host> Establezca el puerto COM del host y las propiedades del puerto serie> Velocidad en serie 0 para que coincida con el dispositivo Arduino en Windows.

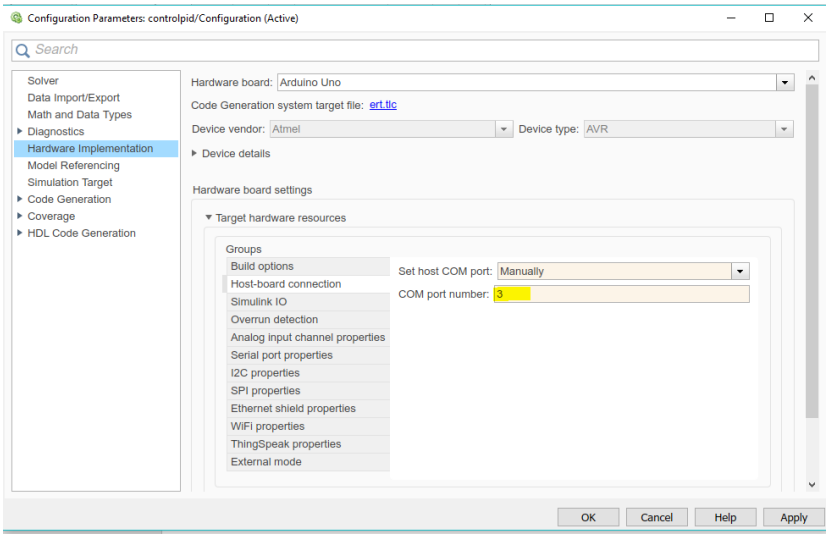

Figura 78 Configuración de parámetros para la comunicación manual con Arduino.

Fuente: Fuente: Simulink-Matlab

.

En la ventana de configuración de Parámetros actualice en Solver el tiempo de la simulación o ejecución, el tipo, el estado y el tiempo de muestreo.

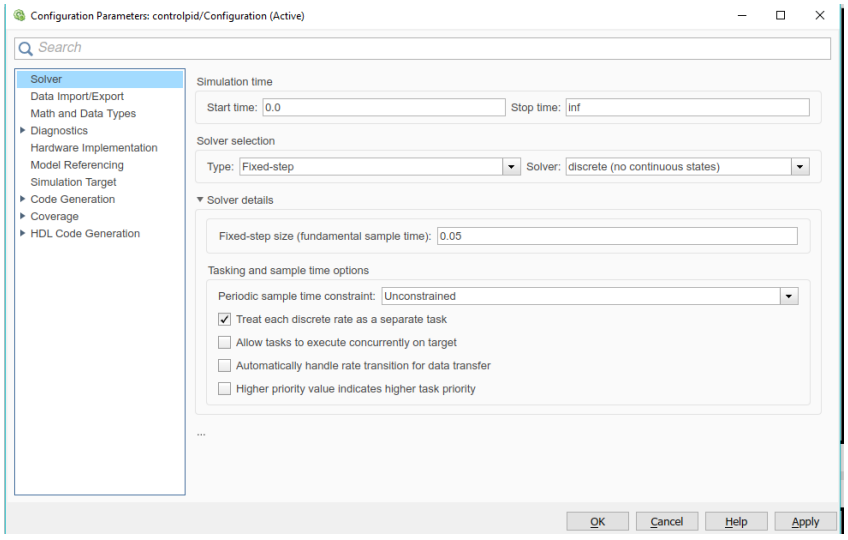

Figura 79 configuración de parámetros tiempo de ejecución y de muestreo.
Fuente: Simulink-Matlab

5.4.3 Ejecutar modelo en tiempo real Arduino.

Puede preparar, configurar y ejecutar un modelo en el Arduino.
Antes de iniciar este procedimiento:
Conecte el hardware Arduino a la computadora host mediante un cable USB.
Crea o abre un modelo Simulink.
Use el Archivo Guardar como para crear una copia de trabajo del modelo.
En su modelo, seleccione Herramientas> Ejecutar en hardware de destino> Preparar para ejecutar. Esta acción cambia los parámetros de configuración del modelo.
Cuando se abra el panel Implementación de hardware, establezca el parámetro de la placa de hardware en la placa Arduino específica que está utilizando.
Haga clic en el botón Implementar en hardware, modelo de sintonización y monitor que se ejecuta en el hardware Arduino.

Haga clic en el botón Deploy to Hardware para enviar el programa a Arduino y quedara grabado en el Arduino.

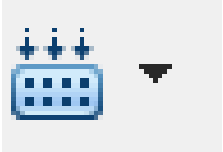

Figura 80 Deploy To Hardware.

Fuente: Simulink-Matlab

Se puede usar el modo externo para ajustar los parámetros y monitorear los datos del modelo mientras se ejecuta en el Arduino.
El modo externo le permite ajustar los parámetros del modelo y evaluar los efectos de diferentes valores de parámetros en los resultados del modelo en tiempo real, con el fin de encontrar los valores óptimos para lograr el rendimiento deseado. Este proceso se llama ajuste de parámetros.

Figura 81 Modo Simulación en Externo.

Fuente: Simulink-Matlab

El modo External permite modificar los parámetros y visualizar los datos del modelo de Simulink mientras se está ejecutando en el hardware Arduino en tiempo real. Esto permite un ajuste de parámetros más cómodo y rápido, ya que se puede visualizar en tiempo real cómo reacciona el sistema antes cambios de las variables durante la simulación.

Establezca el parámetro de tiempo de detención de Simulación, ubicado a la izquierda del modo Simulación en la barra de herramientas del modelo. El valor predeterminado es 10.0 segundos. Para ejecutar el modelo por un período indefinido, ingrese inf.
Haga clic en el botón Ejecutar.

5.5 MODELO DE ENVIO DE SEÑAL CONTROL DEL MOTOR.

En este apartado se detallan las características que debe tener la parte del modelo de Simulink que se encarga de enviar la señal que aporta el movimiento de giro del motor. Gracias al bloque Standard Servo Write que viene en la librería de bloques de Arduino se puede enviar por el pin que se indique del Arduino la señal de información tipo PWM que debe recibir el ESC para hacer girar el motor. En la Figura 26 se muestra un modelo básico con los bloques necesarios para hacer girar el motor desde Simulink. Se trata de un modelo bastante sencillo y que, como se puede ver en la ventana que aparece al hacer doble click sobre el bloque Standard Servo Write (a la derecha de la Figura 26), genera una señal PWM cuyo duty cycle depende del valor de entrada al bloque, que debe estar comprendido entre 0 y 255. Es decir, el bloque necesita una señal de entrada con valor entre 0 y 255 donde 0 corresponde a duty cycle de 0% y 255 a un duty cycle del 100%. El porqué de que el valor máximo sea 255 es porque al Arduino se le transmite 1 byte (8 bits), cuyo valor máximo en decimal es 255, para esto se realiza una conversión de datos y se utiliza el bloque se Data Type Conversión que nos proporciona Simulink y se pone en dato tipo int8 para que tenga una salida de 8bits y un bloque de Referencia para manipular, para realizar las pruebas iniciales se recomienda hacer la referencia con un Slider y así tener el punto inicial y final de control del motor para acondicionar el funcionamiento.
Se debe indicar en el bloque de parámetros de Standard Servo Write el número del pin de salida en este caso como es el Arduino Uno se utilizará el pin 9 y se conecta el cable del ESC indicado.

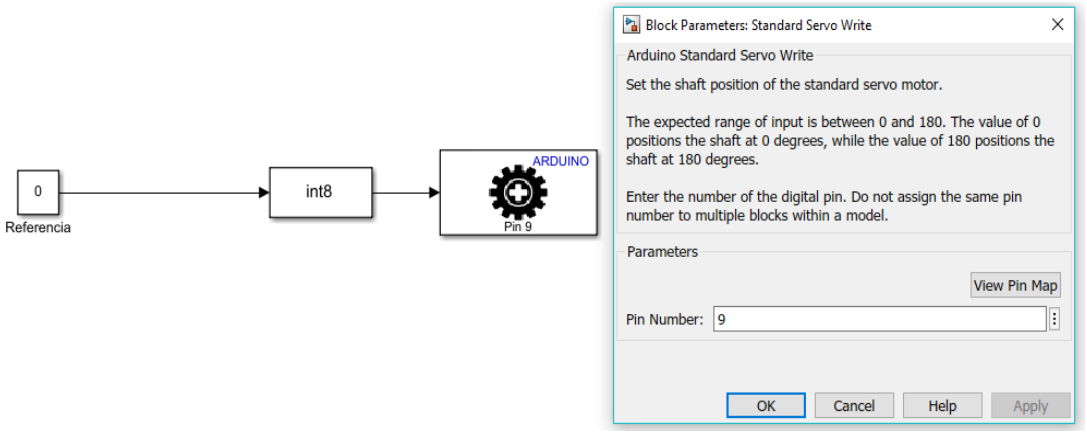

Figura 82 Modelo control de motor.

Fuente: Simulink-Matlab

5.5.1 Arrancar del motor

Hay que tener en cuenta varios aspectos para encender el, primero es que hay que armar el ESC antes de arrancar el motor, se alimenta el ESC con la fuente de alimentación si suena un pitido es que el ESC. se ha armado correctamente. Una vez listo el ESC hay que ir aumentando progresivamente el valor de la señal hasta que el motor comience a funcionar. Cuando el ESC se ha armado no hace falta volver a armarlo hasta que no se desconecte la fuente de alimentación. Cada motor arranca con un valor diferente. En este caso el motor utilizado comienza a moverse a partir del valor de señal igual 50 desde Simulink, estando el ESC alimentado con una fuente a 12V.

5.5.2 Modelo de recepción de datos del Sensor de Posición Angular.

En este caso que se utiliza un potenciómetro como sensor de medición del ángulo, al modelo se le debe incluir determinados bloques dedicados a la recepción de la información que aporta el potenciómetro. Como se ha comentado anteriormente el potenciómetro aporta una señal analógica por uno de sus terminales cuyo valor varía según la posición del potenciómetro. El valor de esta señal analógica puede ser medido desde uno de los pines de entrada analógica de la placa Arduino. Para ello se ha de configurar el pin correspondiente como entrada y visualizar el valor en Simulink. Todo ello se puede hacer usando los bloques disponibles en la librería sin tener que escribir ningún código en Arduino.

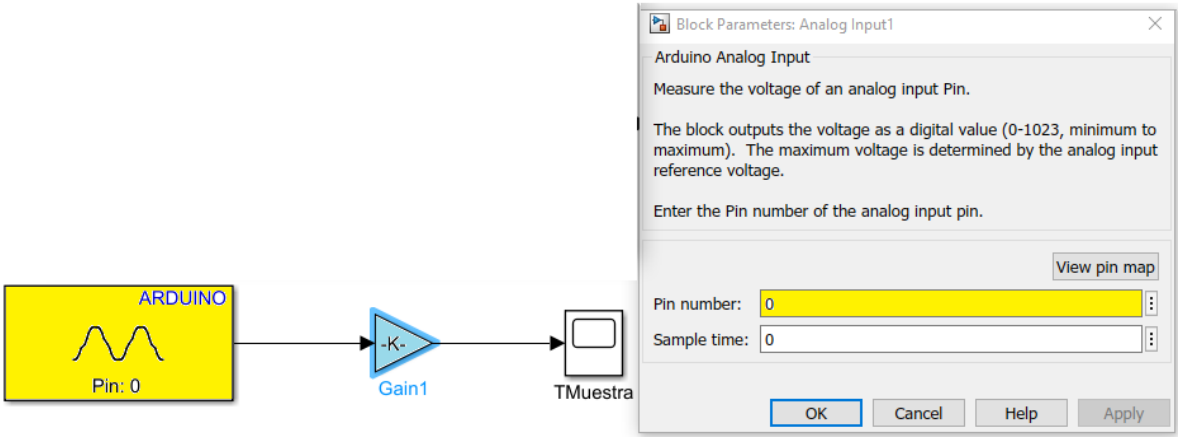

Figura 83 Modelo de recepción señal Análoga, ventana de parámetros Gain y ventana parámetros Analog input.

Fuente: Simulink-Matlab

El modelo está formado por varios tipos de bloques disponibles en la librería de Simulink:
El Bloque Analog Input: proporciona la señal recibida por el Arduino desde el potenciómetro. Hay que indicar el pin que se va utilizar como entrada haciendo doble click sobre el bloque e indicándolo en el apartado Pin number 0 en este caso Ver Figura (30) lado derecho.
Bloque Gain: Multiplica una señal de entrada y entrega una señal escalonada se utilizará en este caso para convertir la señal análoga recibida por el bloque Analog Input en una señal que varié desde 20 ha diferentes valores en grados.
Bloque Scope: representa gráficamente la señal de entrada en el tiempo. Muy útil para ver su evolución.

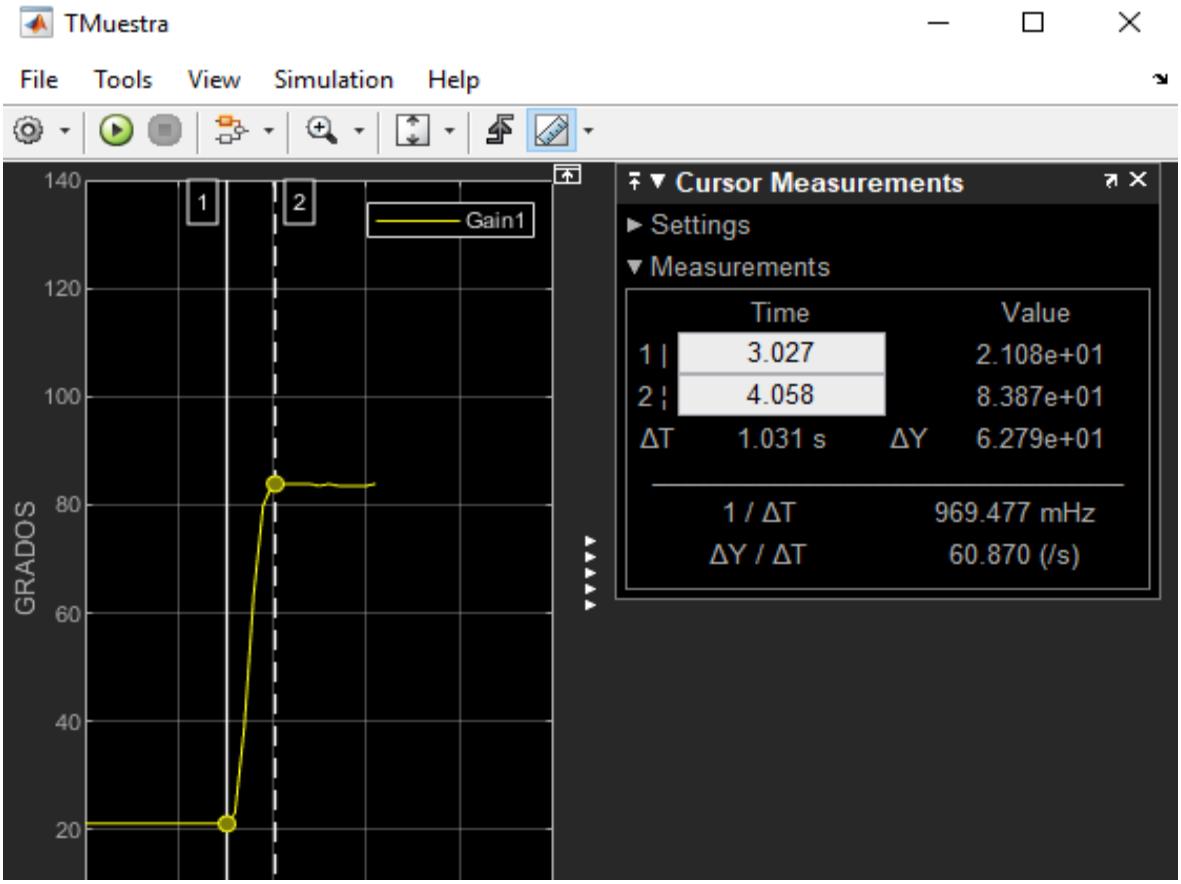

Figura 84 Muestra grafica de datos rampa ,ganancia y tiempo de muestreo.

Fuente: simulink-appdesigner.

Para obtener una buena base de datos necesarios para el control se debe de realizar varias muestras con el mismo valor de referencia y observar su comportamiento sacar las ganancias y tiempos para luego promediarlos y adquirir una mejor exactitud del sistema.

$K_1 = 57.351$ $T_1 = 1.166ms$

$K_2 = 65.35$ $T_2 = 1.0315ms$

$K_3 = 60.692$ $T_3 = 1.0765ms$

$K_4 = 60.870$ $T_4 = 1.031\ ms$

$K_5 = 61.983$ $T_5 = 1.4ms$

$K_6 = 53.862$ $T_6 = 0.94ms$

Promedio. $K_P = 60.006$ $T_P = 1.1075ms$

5.6 SIMULACIÓN EN LAZO CERRADO.

Para la simulación del modelo en lazo cerrado utilizare la herramienta Simulink, añadiremos un integrador discreto como planta ya que el comportamiento del balancín es totalmente integral al aumentar el Duty Cycle el comportamiento del balancín es una Rampa por esto será sintonizado mediante el método de prueba y error, pero con ayuda de la interfaz App Designer podremos sacar los datos de la gráfica de toma de muestras, el lazo queda de la siguiente manera.

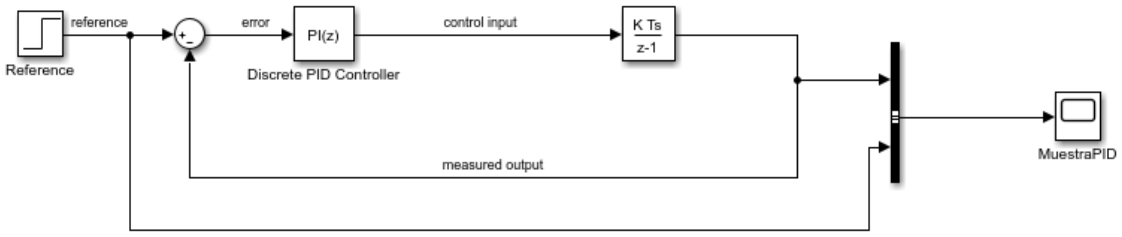

Figura 85 Modelo control PI lazo cerrado.

Fuente: Simulink-Matlab.

El bloque Discrete PID Controller implementa un controlador PID (PID, PI, PD, P solamente, o solo I), en este caso se utiliza el PI.

La salida de bloque es una suma ponderada de la señal de entrada, la integral de la señal de entrada y la derivada de la señal de entrada. Los pesos son los parámetros de ganancia proporcional, integral y derivada. Un polo de primer orden filtra la acción derivada.

El bloque soporta varios tipos de controladores y estructuras. Las opciones configurables en el bloque incluyen:

- Forma del controlador (paralelo o ideal): En este trabajo se realizará en formato paralelo.
- Dominio de tiempo (continuo o discreto): En este trabajo se realiza en tiempo discreto.
- Condiciones iniciales y activación del restablecimiento: consulte los parámetros de restablecimiento de fuente y externos.

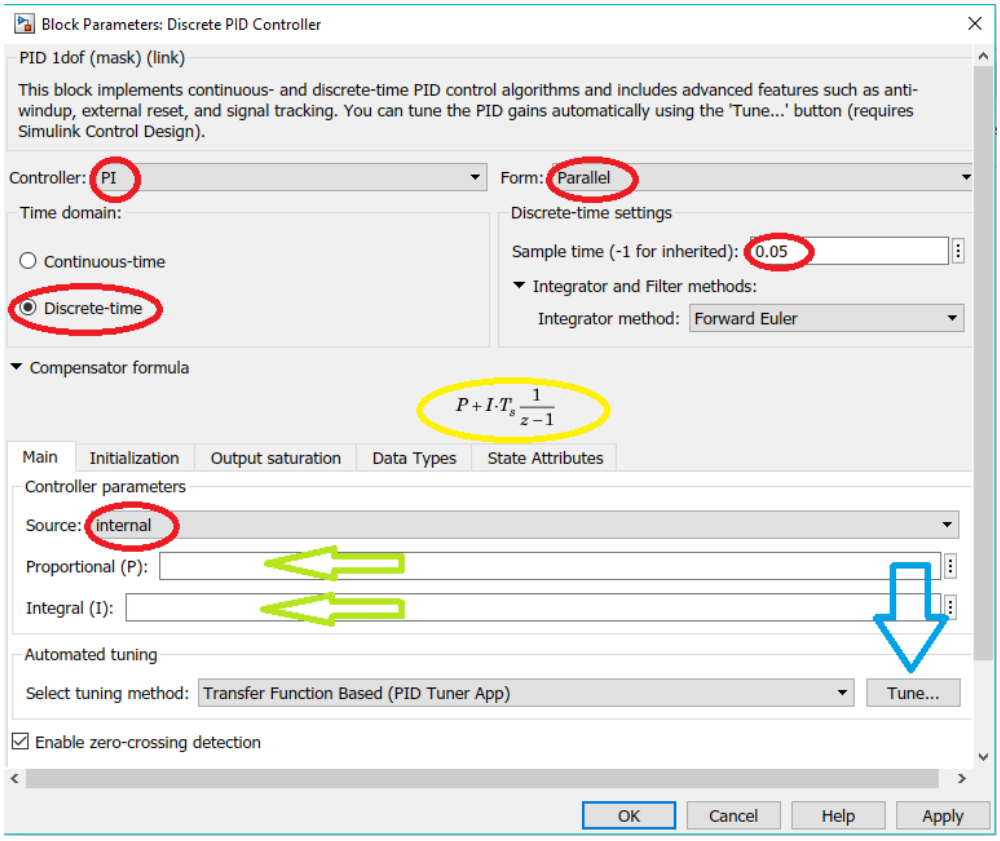

Figura 86 Block de Parámetros del controlador PID del modelo control PI.

Fuente: Matlab-Simulink.

Como se ve en la figura 34 los parámetros que se utilizan son el controlador PI en tiempo discreto de forma paralela con un tiempo de respuesta de 0.05 el cual se va utilizar durante todo el trabajo, se utiliza el PID de modo interno (señalados con Rojo), tal que nos pueda arrojar los valores del control (flecha verde) con ayuda de la APP PID Tuner (flecha azul) y poder compensar el control con la fórmula que se ve en amarillo.

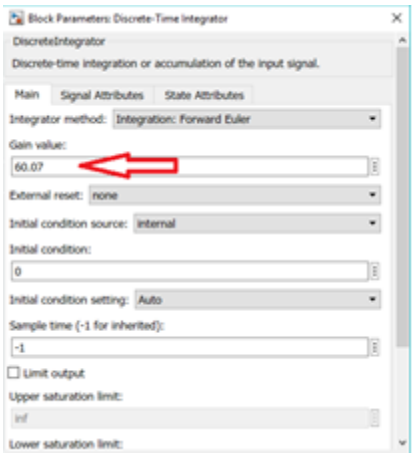

Figura 87 Block de parámetros del integrador discreto.

Fuente: Matlab-Simulink

En el bloque de integrador discreto se inserta el valor de constante proporcional $K = \frac{\Delta Y}{\Delta x}$ que en este caso fue tomada de la gráfica Figura 33 para poder obtener un comportamiento del sistema modelado a la del sistema real y adquirir los datos del controlador y obtener los valores para realizar la sintonización del control PID en el modelo de Lazo de control Figura 38 .

Los valores los obtenemos con ayuda de La APP PID Tuner incorporada dentro del controlador PID figura 34 flecha azul.

PID Tuner es la aplicación del sintonizador PID que ajusta 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 derivado o controladores PID. Los gráficos de análisis le permiten examinar el rendimiento del controlador en los dominios de tiempo y frecuencia. Puede refinar interactivamente el rendimiento del controlador para ajustar el ancho de banda del bucle y el margen de fase, o para favorecer el seguimiento de los puntos de ajuste o el rechazo de perturbaciones.

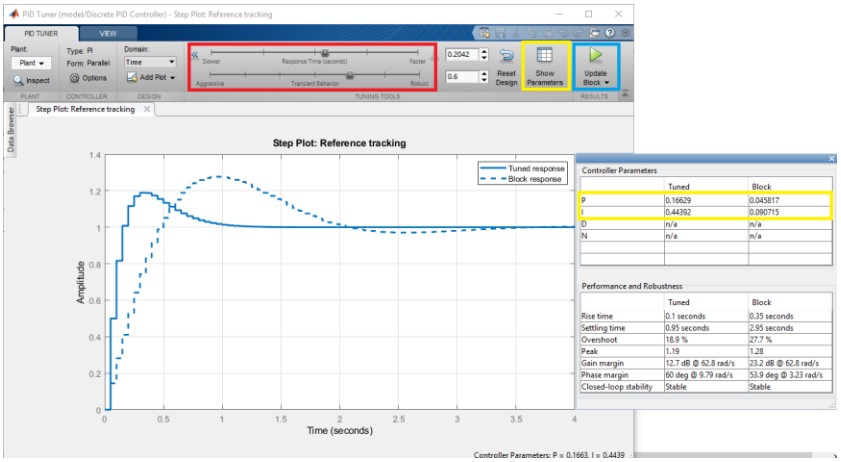

Figura 88 APP PID Tuner.

Fuente: Simulink

En esta grafica se visualiza el ajuste para el controlador, se puede hacer desde las barras superiores en rojo para compensar y después se envía los datos desde Update block recuadro azul, se puede visualizar los datos en Show Parameter recuadro amarillo. Estos valores son los utilizados en el controlador, también son arrojados al modelo PID.

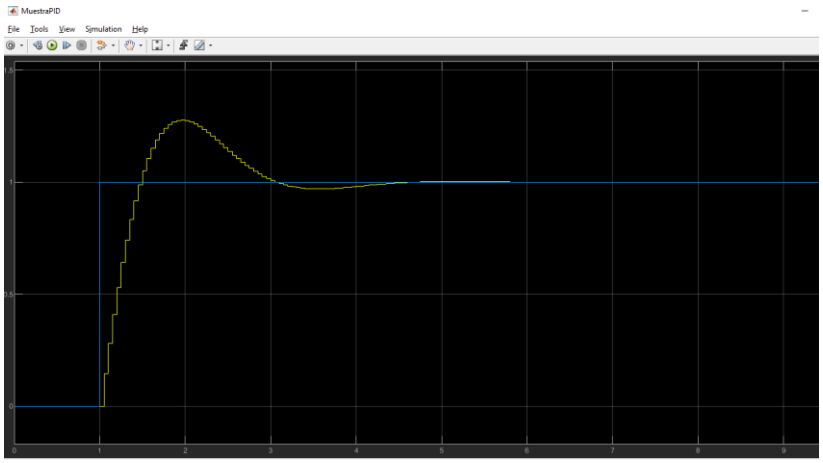

Figura 89 Grafica del control del modelo de lazo cerrado.

Fuente: Matlab-Simulink

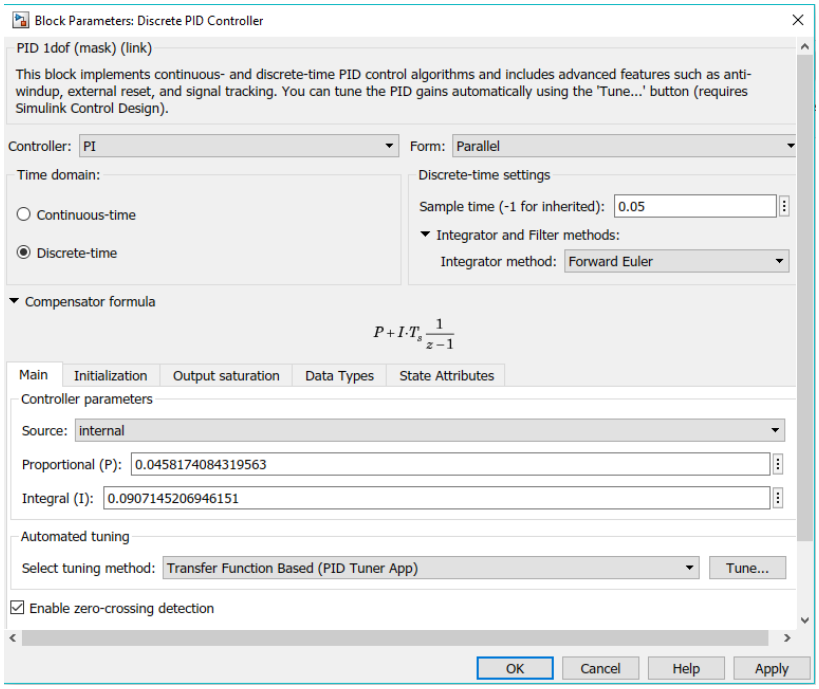

Figura 90 Block de Parámetros del controlador PID con los Valores de Sintonización.

Fuente: Matlab-Simulink.

5.7 LAZO DE CONTROL. SIMULINK.

En la siguiente figura se puede observar el lazo de control creado mediante la herramienta de Matlab, Simulink.

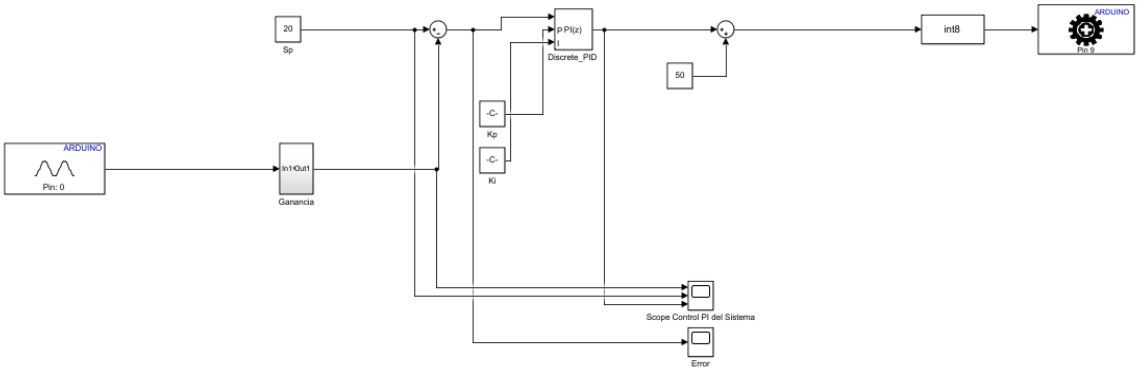

Figura 91 Modelo Final Control PID.

Fuente: Simulink-Matlab

5.7.1 Sintonización del controlador PI.

Se ha elegido un regulador PI ya que este nos permite reducir el error de estado estacionario a cero, tiene buen comportamiento ante saltos en la referencia, no introduce ruidos, hecho que estaría presente si el controlador tuviese acción derivativa.

Para sintonizar el PI se utilizará el método de prueba y error, teniendo en cuenta que se quiere logra una respuesta sin sobre pico y lo más lineal posible. A continuación, veamos la respuesta del sistema para diferentes valores de la ganancia proporcional y el tiempo integral.

En la figura 38 se tienen los valores de la ganancia proporciona e integral que será ingresada en el modelo Control PID en las constantes Externas del controlador.

En la Figura 39 se muestra el modelo en Simulink definitivo, en el que se recibe la posición de sistema y se controla la velocidad del motor. Se parte de los modelos descritos la recepción de datos del Sensor de Posición Angular y envío de señal del control del motor, y se añaden los bloques que permite realizar el control.

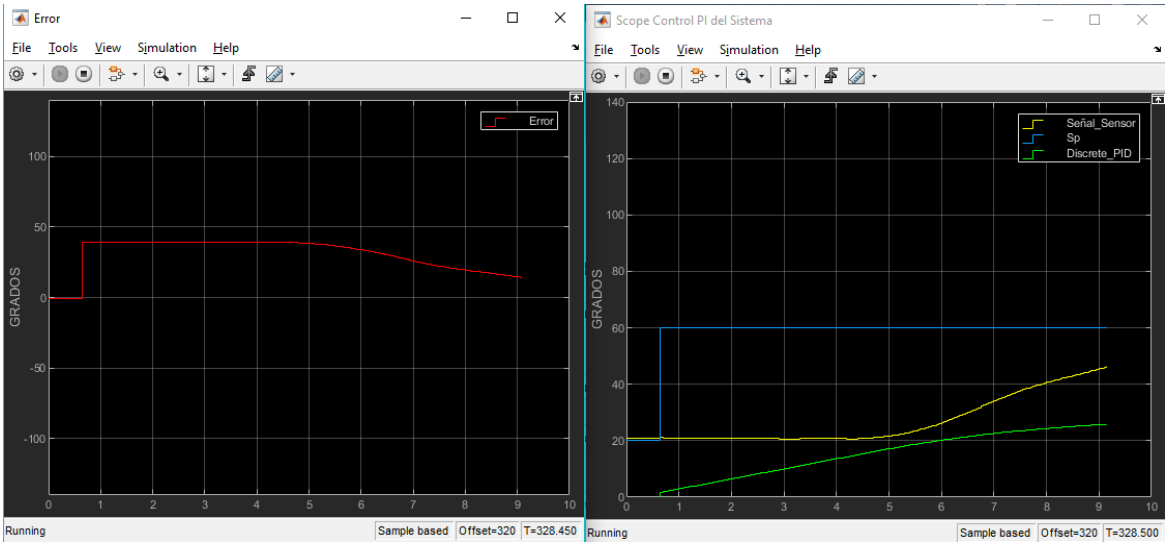

Figura 92 Grafica del control comportamiento de 20 a 60 Grados.

Fuente: Simulink-appdesigner.

En la gráfica se puede ver la señal del balancín (línea amarilla) como busca estabilizarse en el punto de referencia (línea Azul), el comportamiento del control (línea verde) y el error (línea roja).

5.8 INTERFAZ GRAFICA DEL CONTROL.

El modelo de envió de señal control del motor y el modelo de recepción de datos del Sensor de Posición Angular están en un solo Block Model de Simulink llamado Tomamuestras, el Modelo de lazo cerrado está en un Block Model llamado Model y el controlador PID en un Block Model llamado controlpid esto lo necesitaremos para crear la Interfaz en la APP DESIGNER de Matlab.

La interfaz se elabora simplemente arrastrando los componentes visuales y colocándolos en el diseño hasta completarlo como se muestra en la Figura 93.

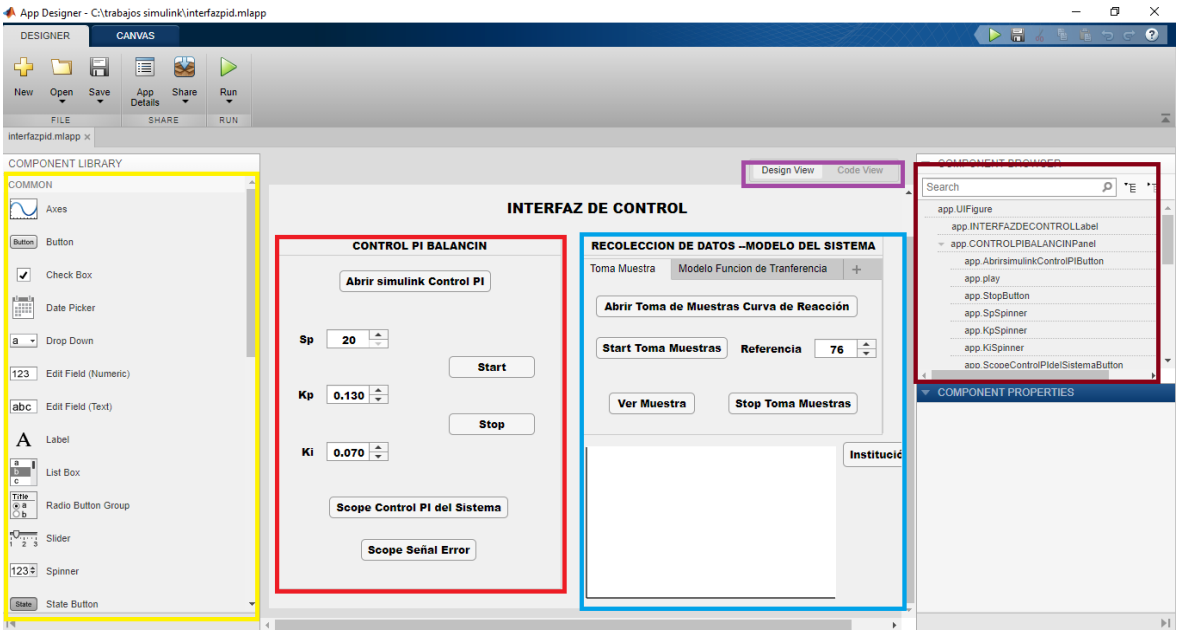

Figura 42 Interfaz Gráfica en APP DESIGNER.

Fuente: Mablab.

Podemos ver al lado Izquierdo de la Figura 94 que se encuentras los componentes

Que se pueden utilizar (recuadro amarillo) al arrastrarlos al diseño este automáticamente crea una app en el navegador de componentes al lado derecho del diseño (recuadro café), se puede modificar el nombre y también crea automáticamente un código (recuadro morado) donde se puede manipular para que funcione nuestra interfaz.

La interfaz de control se realizó en dos partes la primera es la recolección de datos (recuadro azul) que cuenta con dos partes.

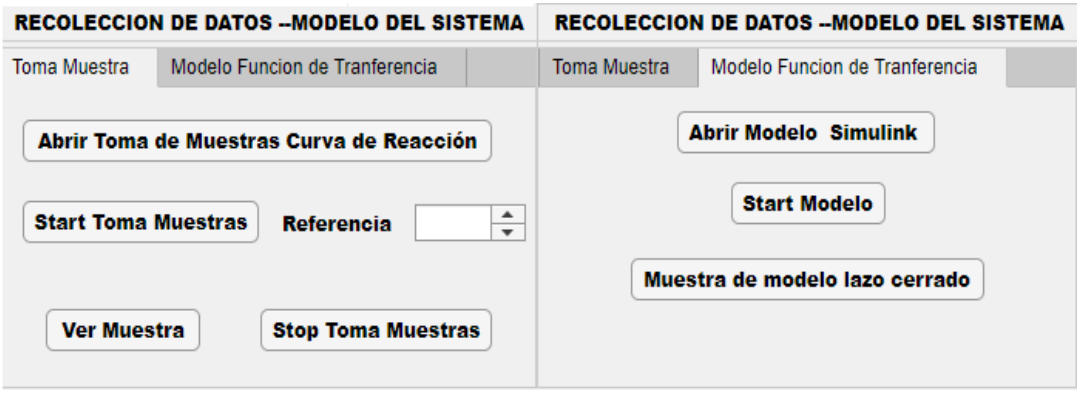

Figura 43 recolección de datos.

Fuente: Appdesigner.

En la parte izquierda de la figura se tiene la toma de muestra con cinco componentes los cuales están divididos así.

- Abrir Toma de Muestras Curva de Reacción: Este botón lo que hace es abrir el modelo de Simulink que se llama Tomamuestras, el código empleado para poder acceder a él es:

```
function AbrirTomadeMuestrasCurvadeReaccinButtonPushed (app, event)
open_system('Tomamuestras')
end
```

- Start Toma Muestras: Nos permite poner en funcionamiento en modelo de tomar muestras de simulink, para poner en funcionamiento este modelo se debe de tener conectado el Arduino y Matlab debe de estar cargado con la carpeta donde están ubicados los modelos, el código empleado para poder acceder a él es:

```
function StartTomaMuestrasButtonPushed (app, event)
        set_param ('Tomamuestras','Simulation Command','Start')
        end
```

- Ver Muestra: Permite abrir el Scope del modelo de recepción de datos del Sensor de Posición Angular que envía la señal de control del motor, el código empleado para poder acceder a él es:

```
function VerMuestraButtonPushed (app, event)
        open_system('Tomamuestras/TMuestra')

        end
```

- Referencia: Nos permite poner una referencia al motor la cual está entre un rango de 50 a 80 en Simulink 50 como dato mínimo para que le motor comience a girar y 80 como dato máximo para que el motor este en su punto máximo. El rango que se envió como escalón fue de 76 para ver el comportamiento de la planta, el código empleado para poder acceder a él es:

```
function ReferenciaSpinnerValueChanged (app, event)
set_param('Tomamuestras/Referencia','Value',num2str(app.ReferenciaSpinner.Va
lue));

end
```

- Stop Toma Muestras: Nos permite parar el modelo y de ahí obtener una gráfica en el Scope de ver muestra para ´poder realizar el análisis que deseemos, el código empleado para poder acceder a él es:

```
function StopTomaMuestrasButtonPushed(app, event)
    set_param('Tomamuestras','SimulationCommand','Stop')
    end
```

En la parte derecha de la figura se tiene el modelo de lazo cerrado con la función de trasferencia de integral pura, cuenta con tres componentes:

- Abrir Modelo Simulink: Nos permite abrir el modelo de Simulink llamado Model donde se encuentra el modelo de lazo cerrado del sistema, el código empleado para poder acceder a él es:

```
function AbrirModeloSimulinkButtonPushed (app, event)
   open_system('model')
 end
```

- Start Modelo: Permite poner en funcionamiento el modelo, el código empleado para poder acceder a él es:

```
function StartModeloButtonPushed(app, event)
    set_param('Model','SimulationCommand','Start')
  end
```

- Muestra de modelo lazo cerrado: Permite obtener la gráfica del comportamiento del modelo de la planta en lazo cerrado, el código empleado para poder acceder a él es:

```
function MuestrademodelolazocerradoButtonPushed (app, event)
    open_system('Model/MuestraPID')
  end
```

Adicional en la parte inferior se coloca una imagen de la institución con un botón para mostrarla:

```
function InstitucinButtonPushed (app, event)
        imshow('ITM.png','Parent',app.UIAxes);
    end
```

Figura 95 Imagen institución en Interfaz Grafica

Fuente: Appdesigner.

La segunda parte de la interfaz Figura 41 (Recuadro Rojo) es la interfaz del control PID del balancín que cuenta con ocho componentes descritos así:

- Abrir simulink Control PI: Nos permite abrir el modelo de Simulink donde se encuentra el control llamado controlpid, el código empleado para poder acceder a él es:

```
function AbrirsimulinkControlPIButtonPushed(app, event)
open_system('controlpid.slx')
end
```

- Start: Nos permite poner en funcionamiento el modelo de controlpid, para poner en funcionamiento este modelo se debe de tener conectado el Arduino y Matlab debe de estar cargado con la carpeta donde están ubicados los modelos, el código empleado para poder acceder a él es:

```
function KpSpinnerValueChanged(app, event)
set_param('controlpid/Kp','Value',num2str(app.KpSpinner.Value));
end
```

- Scope Control PI del Sistema: Nos permite visualizar el comportamiento del control PID en tiempo real abriendo el Scope de Simulink, el código empleado para poder acceder a él es:

```
function ScopeControlPIdelSistemaButtonPushed(app, event)
open_system('controlpid/Scope Control PI del Sistema')
end
```

- Scope Señal Error: Nos permite visualizar el comportamiento del error en el sistema del control PID, el código empleado para poder acceder a él es:

```
function ScopeSealErrorButtonPushed(app, event)
open_system('controlpid/Error')
end
```

- Sp: nos permite manipular el setpoint del sistema, la referencia donde queremos que se nos posicione el balancín comienza con un valor en grados de 20 ya que el balancín está posicionado en 20 grados para comenzar, el código empleado para poder acceder a él es:

```
function SpSpinnerValueChanged(app, event)
set_param('controlpid/Sp','Value',num2str(app.SpSpinner.Value));
end
```

- Kp: Nos permite manipular la constante proporcional del modelo de control enviando el dato a un bloque de simulink constante que se encuentra en la parte externa del controlador PI, el código empleado para poder acceder a él es:

```
function KpSpinnerValueChanged(app, event)
set_param('controlpid/Kp','Value',num2str(app.KpSpinner.Value));
end
```

- Ki: Nos permite manipular la constante Integral del modelo de control enviando el dato a un bloque de simulink constante que se encuentra en la parte externa del controlador PI, el código empleado para poder acceder a él es:

```
function KiSpinnerValueChanged(app, event)
set_param('controlpid/Ki','Value',num2str(app.KiSpinner.Value));
end
```

El código de la interfaz se puede visualizar en la APP DESIGNER en la parte superior derecha Figura 40 (recuadro morado),

6 RESULTADOS Y DISCUSIÓN

6.1 RESULTADOS OBTENIDOS DEL CONTROLADOR PI EN LA PLANTA BALANCIN.

Es aplicar a la planta, el control PI desarrollado en Simulink y por medio de la interfaz Gráfica manipular sus variables, aquí se implementan los diseños presentados en este trabajo verificando el comportamiento del control del balancín y además permite al estudiante afianzar sus conocimientos y experiencias con base en la implementación de sus diseños de control.

Luego de realizar varias pruebas del control se llega a la conclusión de que para tener una respuesta que sea la deseada, sin ruido en la señal de control, sin sobre picos en la evolución de la posición y con buenos tiempos de respuesta, los valores de las ganancias proporcional e integral debían ser de 0.0458174084319563 y 0.0807145206946151 respectivamente. En las siguientes imágenes se puede observar la respuesta del sistema para estos valores.

En la Figura 45 se puede observar una referencia o setpoint de 30 con un tiempo de muestreo de 0.05 el cual presenta una evolución de la posición de la barra hasta el punto deseado que es a 30 grados, como se puede apreciar el comportamiento fue el deseado con los valores de ganancia proporcional e integral y con un buen tiempo de respuesta.

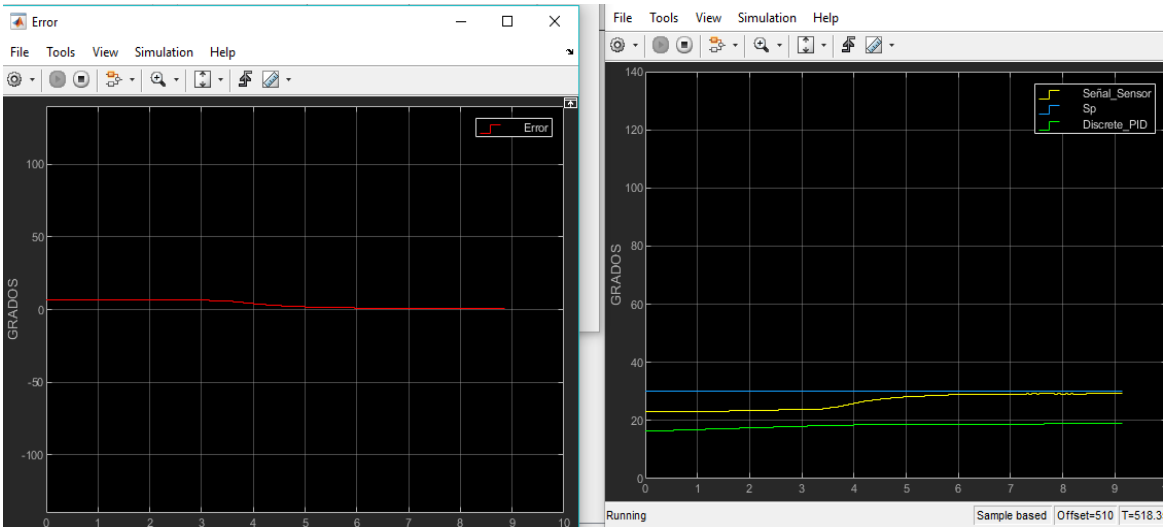

Figura 96 Control PI 30 Grados.
Fuente: Simulink.

Como se observa el sistema tiene una buena respuesta, la referencia se alcanza sin que haya sobre picos en la evolución de la posición, la cual se incrementa prácticamente de forma lineal.

En la Figura 46 se puede observar un setpoint de 40 el cual presenta una evolución de la posición de la barra hasta el punto deseado que es a 30 grados, como se puede apreciar el comportamiento fue el deseado con los valores de ganancia proporcional e integral y con un buen tiempo de respuesta.

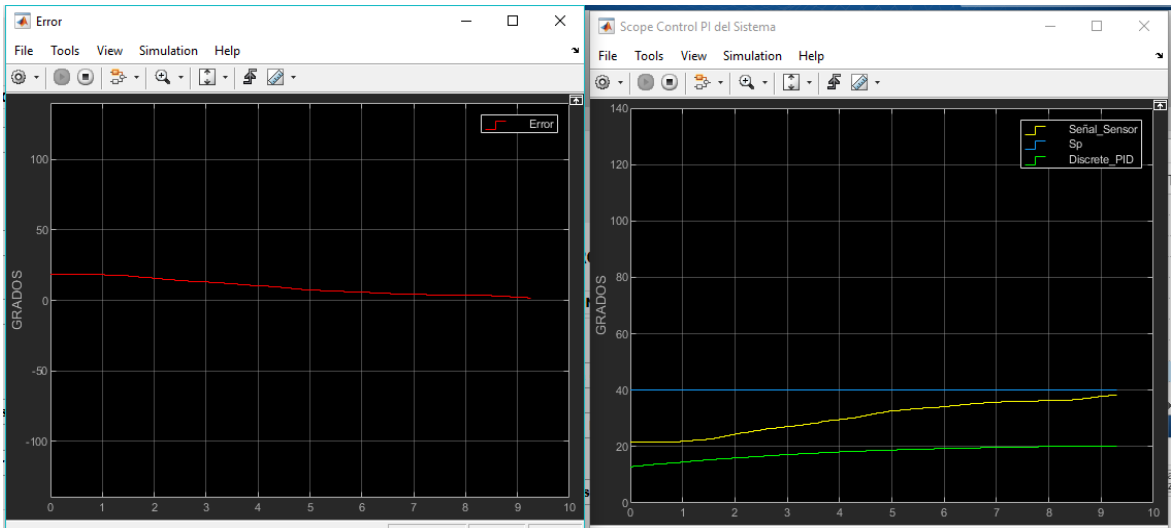

Figura 97 Control PI 40 Grados.
Fuente: Simulink.

El sistema continuo con buena respuesta, la referencia se alcanza sin que haya sobre picos en la evolución de la posición.

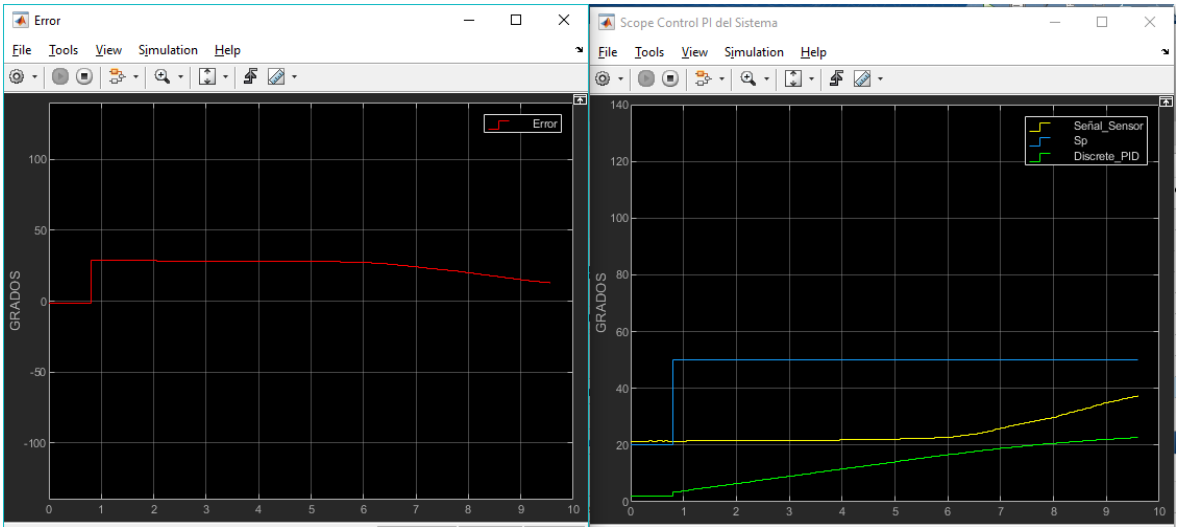

Figura 98 Control PI 50 Grados.
Fuente: Simulink.

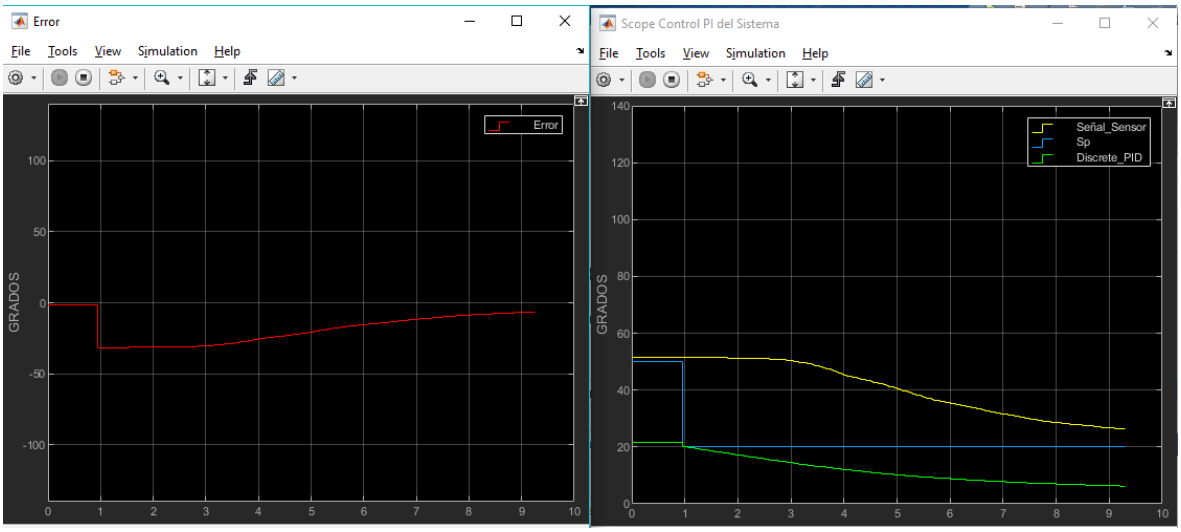

Figura 99 Control PI 50 Grados descendiendo a 20.
Fuente: Simulink.

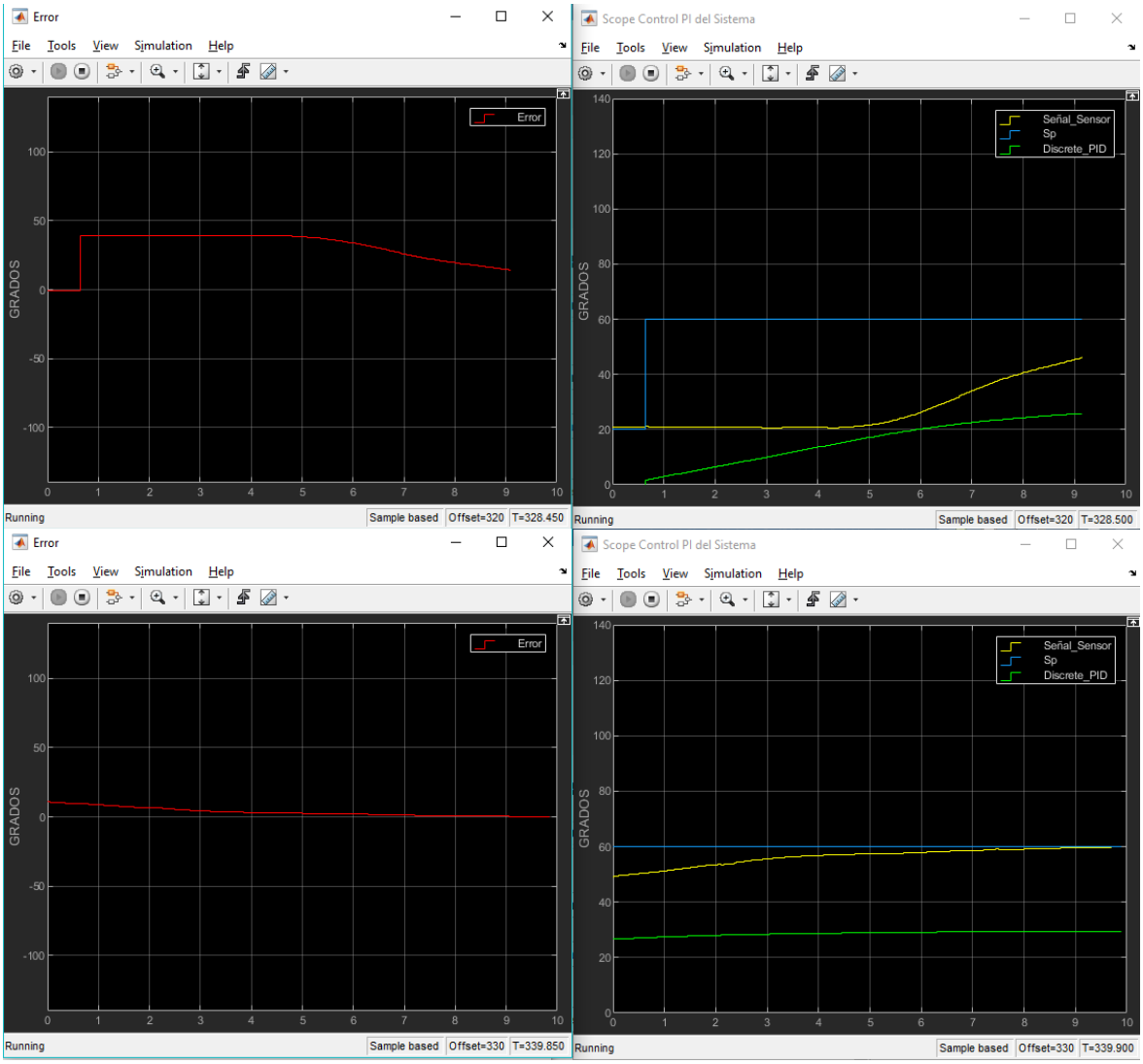

Figura 100 Control PI 60 Grados
Fuente: Simulink.

El control continúa siendo eficiente la figura 49 muestra cómo llega al punto de referencia y en la figura 50 muestra como desciende al Angulo 20 que es donde el balancín está en su posición inicial.

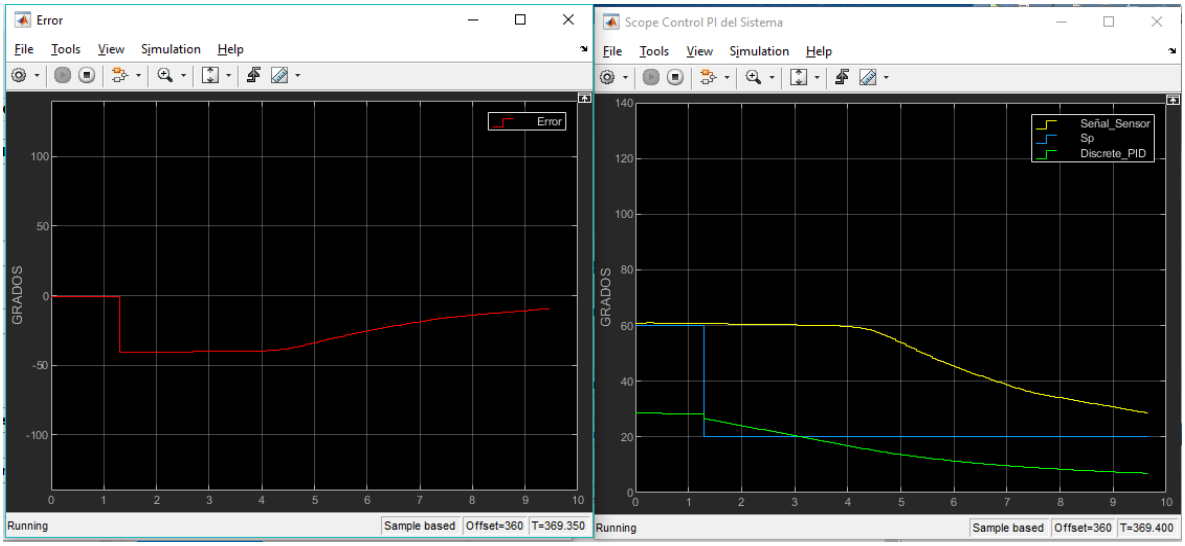

Figura 101 Control PI 60 Grados descendiendo.
Fuente: Simulink.

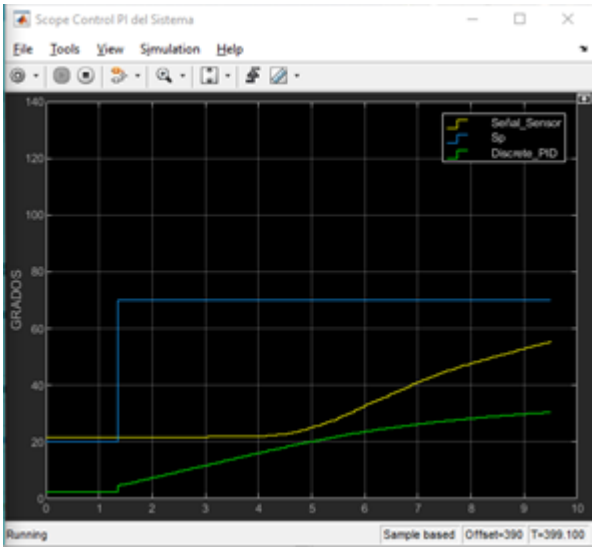

Figura 102 Control PI 70 Grados.
Fuente: Simulink.

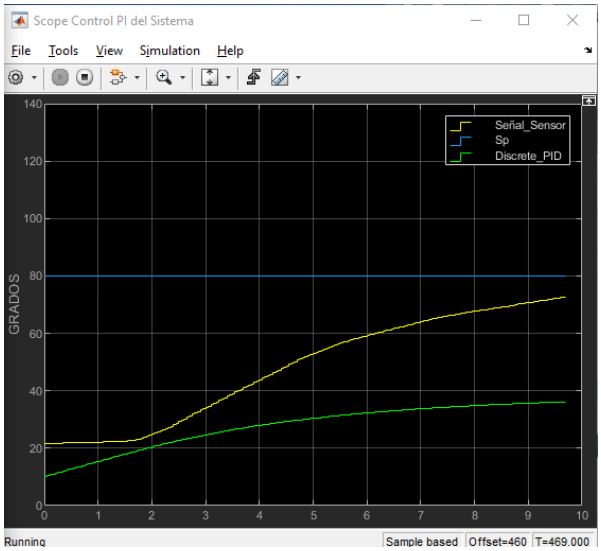

Figura 103 Control PI 80 Grados.
Fuente: Simulink.

En las figuras 51 y 52 continúa siendo estable, aunque en los 80 grados se demora un poco más para estabilizarse en el punto de referencia.

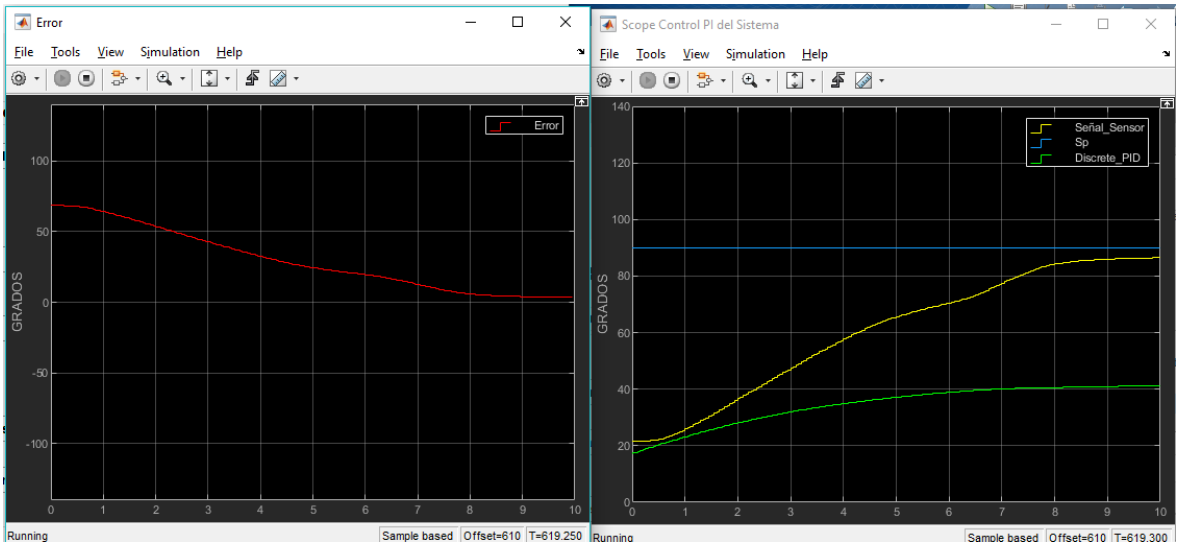

Figura 104 Control PI 90 Grados.
Fuente: Simulink.

El control en 90 grados comienza a mostrar inestabilidad no siempre encuentra el punto de referencia y en referencia 100 grados en adelante ya es inestable, por tanto, la respuesta del controlador PI es satisfactoria de 20 a 80 grados ya que fue casi instantánea la detección de la señal de error, y con un pequeño retardo la acción el control integral I, que es el encargado de anular totalmente la señal de error, en referencias mayores a ese valor presenta inestabilidad.

7 CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

7.1 CONCLUSIONES

- La implementación, el diseño y el desarrollo de la interfaz en el proceso de control para manipular los datos es de gran ayuda además permite identificar todo el proceso ya que direcciona el punto al que se debe llegar.

- Los resultados conseguidos con estos proyectos han sido satisfactorios. Se han cumplido los objetivos propuestos en el proyecto, obteniendo un correcto funcionamiento y un buen control de la planta hasta 90 grados, así como un gran conocimiento de todo lo que ha permitido llegar a ello.

- Este proyecto ha permitido poner en práctica algunos conocimientos adquiridos durante la estadía en la institución y desarrollar nuevos en especial los relacionados con el control y el software implementado.

- El software Matlab con su toolbox simulink y su APP Designer es uno de los grandes aportes que he descubierto con la realización de este proyecto y con la implementación en Arduino me ha hecho consiente de la amplia posibilidad y gran potencial que tiene a la hora de hacer cualquier otro tipo de proyectos relacionados.

- Con la realización de este proyecto he aprendido a adaptar el ordenador para trabajar con la placa de Arduino desde la herramienta simulink de Matlab, lo cual conlleva la instalación de librerías en Matlab.

- Se ha realizado un control en lazo cerrado del sistema con un controlador PID, con la ayuda de Matlab y su toolbox simulink y para su posterior ajuste he identificado los parámetros del controlador PID, que hacen que el proceso funcione de una manera correcta. Esto me ha permitido llegar a la conclusión que el controlador PID es idóneo para este sistema.

- Todo esto ha permitido cumplir nuestro objetivo principal que era el de control de posición de un balancín.

- Este proyecto tiene amplio uso didáctico, ya que su uso sencillo permitirá a los futuros alumnos trabajar con conceptos como lazo abierto, lazo cerrado, controlador PID… viendo reflejado sus resultados en un medio físico.

7.2 RECOMENDACIONES

- Se debe tener en cuenta los ajustes de la planta ya que algún desajuste pude tener fallas considerables a la hora de la implementación del control.

- Cuando se hagan pruebas se debe de tener cuidado con la potencia del motor ya que pude causar daño o lecciones.

- Para alimentar el motor se debe de tener precaución y verificar que si sea la fuente o batería indicada para su funcionamiento para no causar daños a los elementos utilizados o equipos.

- Se debe de verificar bien la estabilidad y la ubicación de la planta no puede estar cerca de objetos o personas por que pueden sufrir algún impacto, la planta debe de estar sujeta desde la base ya que es de un material liviano y el motor puede moverla.

- Se debe de verificar bien las conexiones antes de comenzar algún proceso de control, que si estén bien ubicadas y sin ningún riesgo.

- La ubicación de las aplicaciones y el software de control debe de ser la misma para no generar un error de dirección.

- Se debe de verificar bien los puertos del ordenador donde se valla a implementar, que tengan los drivers y los puertos habilitados para su buen funcionamiento.

7.3 TRABAJOS FUTURO

- Se implementó y desarrollo este trabajo en Matlab con su toolbox simulink y con ayude de la APP DESIGNER ya que permite trabajar con buenos tiempos de muestreo, pero también puede hacerse en otro tipo de software y obtener buenos resultados.

- Se pude continuar con la elaboración de un balancín con más motores y dispositivos como sensor de posición diferentes y aplicarlo en un proceso más robusto.

- Se puede adquirir fácilmente los conocimientos de este trabajo y acondicionarlo a sistemas de control diferente donde se puedan implementar los motores brushless.

REFERENCIAS

https://la.mathworks.com/

http://ctms.engin.umich.edu/CTMS/index.php?aux=Home

http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID

https://la.mathworks.com/help/simulink/slref/discretepidcontroller.html?s_tid=doc_ta

https://la.mathworks.com/help/matlab/app-designer.html?searchHighlight=app%20designer&s_tid=doc_srchtitle

https://la.mathworks.com/products/matlab/app-designer.html

https://la.mathworks.com/videos/arduino-with-matlab-and-simulink-part-3-programming-arduino-with-simulink-99404.html

Arduino:

https://www.arduino.cc/

Potenciómetro:

https://curiosoando.com/que-es-un-potenciometro

http://www.areatecnologia.com/electronica/potenciometro.html

https://www.youtube.com/watch?v=2RhzXuETnwg

Motor Brushless:

https://es.wikipedia.org/wiki/Motor_el%C3%A9ctrico_sin_escobillas

https://es.wikipedia.org/wiki/Motor_eléctrico_sin_escobillas

http://housecomputer.ru/rest/hobby/rc_models/brushless_device/brushless02/brushless02.html

http://www.quadruino.com/guia-2/materiales-necesarios-1/motores-brushless

http://www.asifunciona.com/electrotecnia/af_motor_cd/af_motor_cd_8.htm

https://1mecanizadoelarenal.files.wordpress.com/2013/11/motores-brushless.pdf

http://www.electrosector.com/wp-content/ftp/descargas/brushless.pdf?64b304

http://motores.nichese.com/brushless.htm

Variador Esc:

https://en.wikipedia.org/wiki/Electronic_speed_control

http://tallerdedalo.es/web/ESC

file:///C:/Users/PERSONAL/Downloads/30A_BLDC_ESC_Product_Manual.pdf

control PID:

https://es.wikipedia.org/wiki/Controlador_PID

http://www.dia.uned.es/~fmorilla/MaterialDidactico/El%20controlador%20PID.pdf

https://la.mathworks.com/help/supportpkg/arduino/examples/drive-with-pid-control.html

https://wechoosethemoon.es/2011/07/21/arduino-matlab-simulink-controlador-pid/

APÉNDICE

Los apéndices deben ser nombrados con letras para diferenciarse unos de otros (p. ej: Apéndice A, Apéndice B, etc.). Estos hacen extensiva la información del contenido del trabajo realizado tales como cálculos matemáticos extensos, códigos de programación, etc. El contenido de los apéndices debe permitir a alguien externo al desarrollo del trabajo, llegar a los mismos resultados siguiendo la misma metodología complementada con la información que en este aparte reposa.

FIRMA ESTUDIANTES _____

_____

_____

FIRMA ASESOR _____

FECHA ENTREGA: _____

FIRMA COMITÉ TRABAJO DE GRADO DE LA FACULTAD _____

RECHAZADO___ ACEPTADO____ ACEPTADO CON MODIFICACIONES_____

ACTA NO._____

FECHA ENTREGA: _____

FIRMA CONSEJO DE FACULTAD_____

ACTA NO._____

FECHA ENTREGA: _____