Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Diseño e implementación de un sistema de control automático para el movimiento del Kit de Robótica Costarricense: Krotic

Informe de Proyecto de Graduación para optar por el título de Ingeniera Electrónica con el grado académico de Licenciatura

Catalina Espinoza Quesada

Cartago, junio de 2018

INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PROYECTO DE GRADUACIÓN

ACTA DE APROBACIÓN

Defensa de Proyecto de Graduación
Requisito para optar por el título de Ingeniero en Electrónica
Grado Académico de Licenciatura
Instituto Tecnológico de Costa Rica

El Tribunal Evaluador aprueba la defensa del proyecto de graduación denominado "Diseño e implementación de un sistema de control automático para el movimiento del Kit de Robótica Costarricense: Krotic", realizado por la señorita Catalina Espinoza Quesada y, hace constar que cumple con las normas establecidas por la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal Evaluador

Ing. Hugo Sánchez Ortiz

Profesor lector

Ing. Leonardo Sandoval Cascante

Profesor lector

Ing. William María Moreno

Profesor asesor

Cartago, 13 de junio, 2018

Declaración de autenticidad

Declaro que el presente Proyecto de Graduación ha sido realizado, en su totalidad, por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado material bibliográfico, he procedido a indicar las fuentes mediante citas.

En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Catalina Espinoza Quesada

Cédula: 116140240

Resumen

El presente informe hace referencia al diseño e implementación de sistemas de control automático de posición y velocidad para el movimiento del kit de robótica Krotic. Se realiza una investigación acerca del diseño de codificadores de cuadratura de bajo costo y se elaboran diseños con distintas configuraciones y sensores. Posteriormente se formulan pruebas de funcionamiento para seleccionar el codificador que mejor se ajusta al sistema.

Con la elección del codificador que presenta mejor funcionamiento, se diseñan dos sistemas de control que se ajustan a instrucciones de movimiento por distancia y por tiempo. Para esto se utilizan métodos analíticos y empíricos en la obtención de los modelos numéricos del sistema y con estos modelos se realizan simulaciones con el fin de aproximar los tipos de control necesarios y los valores de las constantes de estos. Para las instrucciones de movimiento por distancia se elabora un sistema de control de posición angular, y para las instrucciones de movimiento por tiempo un sistema de control de velocidad angular.

Se rediseña el set de instrucciones de manera que se adecúe a los sistemas de control, este contiene instrucciones por tiempo y por distancia, y movimientos hacia adelante, hacia atrás, hacia la izquierda y hacia la derecha. Por último, se realizan mediciones de consumo de potencia según la velocidad de los servomotores. Se eligen velocidades de conveniencia para los estados alto, medio y bajo de batería para optimizar la duración de la misma.

Palabras clave: *codificador de cuadratura, modelo analítico, modelo empírico, posición angular, sistemas de control automático, velocidad angular.*

Abstract

This report refers to the design and implementation of automatic position and speed control systems for the movement of the robotics kit Krotic. Research is carried out on the design of low cost quadrature encoders and designs with different configurations and sensors are made. Subsequently, performance tests are formulated to select the encoder that best fits the system.

With the choice of the encoder that presents the best performance, two control systems are designed that adjust to movement instructions by distance and time. For this, analytical and empirical methods are used to obtain the numerical models of the system and with these models, simulations are carried out to approximate the types of control necessary and the values of the constants for these. For the instructions of movement by distance an angular position control system is elaborated, and for the instructions of movement by time an angular velocity control system.

The set of instructions is redesigned in a way that adapts to the control systems, this contains instructions by time and distance, and movements forward, backward, clockwise and counterclockwise. Finally, power consumption measurements are made according to the speed of the servomotors. Convenience speeds are chosen for the high, medium and low battery states to optimize battery life.

Keywords: *analytical model, angular position, angular velocity, automatic control systems, empirical model, quadrature encoder.*

Dedicatoria

Dedico este trabajo a mis padres

Arnoldo Espinoza Rojas y Floribel Quesada Alfaro,

por ser un ejemplo de esfuerzo y superación

en todos los momentos de la vida.

A Joseph Campos Porras,

por ser esa persona especial en mi vida y

demostrarme su apoyo incondicional.

A mis profesores, amigos y compañeros,

que me han ayudado a crecer como persona.

Agradecimientos

Al Ing. William Marín,

Asesor del Proyecto.

Al Ing. Hugo Sánchez Ortíz y al Ing. Leonardo Sandoval

Lectores del presente documento.

Al MSc. Milton Villegas Lemus,

Coordinador del Laboratorio LuTec.

ÍNDICE GENERAL

Capítulo 1. Introducción 1
Entorno del proyecto 1
1.2 Definición del problema 2
1.2.1 Generalidades 2
1.2.2 Síntesis del problema 3
1.3 Meta 4
1.4 Objetivos 4
1.4.1 Objetivo General 4
1.4.2 Objetivos Específicos 4
Capítulo 2. Estrategia de solución 5
Capítulo 3. Descripción del proceso de diseño 13
3.1 Codificadores de cuadratura 13
3.2 Sistema de control de posición angular 21
3.2.1 Modelo analítico 21
3.2.2 Modelo empírico 26
3.2.3 Estimación de control de posición angular por medio de simulaciones. 32
3.2.4 Estimación de control de velocidad angular por medio de simulaciones. 35
Capítulo 4. Implementación del diseño 37
4.1 Rediseño de las instrucciones de movimiento 40
Capítulo 5. Desarrollo y análisis de pruebas 44
5.1 Consumo energético 44
5.2 Pruebas sobre instrucciones de movimiento por distancia 46
5.3 Pruebas sobre instrucciones de movimiento por tiempo 48
Capítulo 6. Conclusiones y recomendaciones 50
6.1 Conclusiones 50
6.2 Recomendaciones 51
Bibliografía 52
Apéndices 53
A.1 Resultados de pruebas de configuración 1 53
A.2 Resultados de pruebas de configuración 2 55
A.3 Manual de usuario 578

ÍNDICE DE FIGURAS

Figura 1.1 Krotic 1
Figura 2.1 Diagrama de sistema de control de posición 5
Figura 2.2 Partes de un codificador 7
Figura 2.3 Señales A y B de un codificador de cuadratura 8
Figura 2.4 Sensor óptico de transmisión 9
Figura 2.5 Sensor óptico de reflexión 9
Figura 2.6 Propuestas para la configuración de la cuadratura 10
Figura 3.1 Circuito para sensor óptico de transmisión 13
Figura 3.2 Circuito para sensor óptico de reflexión 13
Figura 3.3 Disco generado por programa online 14
Figura 3.4 Disco para codificador (Versión 1) 15
Figura 3.5 Esquemático (A) y resultado (B) de disco codificador (versión 2) y referencia externa de ángulos 16
Figura 3.6 Esquemático y resultado de disco codificador (versión 3) 18
Figura 3.7 Comparación de error en pruebas 19
Figura 3.8 Circuito para obtención del modelo numérico 22
Figura 3.9 Curvas de entrada (u1) y salida (y1) 27
Figura 3.10 Comparación de las aproximaciones a la curva de posición angular 28
Figura 3.11 Diagrama de polos y ceros de modelo 'tf3' 29
Figura 3.12 Entrada vs Velocidad Angular 30
Figura 3.13 Comparación de aproximaciones a curva de velocidad angular 31
Figura 3.14 Diagrama de polos y ceros de modelo 'tf6' 32
Figura 3.15 Comportamiento de sistema tipo PID 33
Figura 3.16 Comportamiento de sistema tipo PI 34
Figura 3.17 Comportamiento de sistema tipo PD 34
Figura 3.18 Comportamiento de sistema tipo I 36
Figura 4.1 Vista frontal (A) y trasera (B) de ruedas originales de Krotic 38
Figura 4.2 Esquemático de disco codificador (A) y resultado final del corte (B) 39
Figura 4.3 Pieza interna de Krotic y colocación de sensores. 39
Figura 4.4 Vista superior de colocación de sensores frente a disco 40
Figura 4.5 Diagrama de flujo de instrucciones de movimiento por distancia 42
Figura 4.6 Diagrama de flujo de instrucciones de movimiento por tiempo 43
Figura 5.1 Gráfica de velocidad vs consumo de los servomotores 45

ÍNDICE DE TABLAS

Tabla 2.1 Tipo de movimientos y tolerancia 11
Tabla 2.2 Set actual de instrucciones de movimientos de Krotic 12
Tabla 3.1 Comparación de error de codificadores según velocidad (Configuración 1) 17
Tabla 3.2 Comparación de error de codificadores según velocidad (Configuración 2) 18
Tabla 3.3 Tabla de verdad para programación de codificador 4x 20
Tabla 3.4 Comparación de parámetros de controladores 35
Tabla 4.1 Set de instrucciones 41
Tabla 5.1 Mediciones de consumo 45
Tabla 5.2 Pruebas sobre instrucciones de movimiento por distancia (Movimiento lineal) .. 47
Tabla 5.3 Pruebas sobre instrucciones de movimiento por distancia (Movimiento lateral). 47
Tabla 5.4 Pruebas sobre instrucciones de movimiento por tiempo 49
Tabla A.1 Pruebas con configuración 1, sensor de reflexión 53
Tabla A.2 Pruebas con configuración 1, sensor de transmisión 54
Tabla A.3 Pruebas con configuración 2, sensor de transmisión 55
Tabla A.4 Pruebas con configuración 2, sensor de reflexión 56

Capítulo 1. Introducción

Entorno del proyecto

Con el objetivo de promover el interés en la tecnología en niños y jóvenes de zonas de riesgo social, nace el proyecto "Luthiers de la Tecnología (LuTec)". Este proyecto fue fundado en el Instituto Tecnológico de Costa Rica, por el Área Académica de Ingeniería en Computadores en conjunto con el Centro de investigación en Computación (CIC), y consiste en un laboratorio de investigación destinado al desarrollo de métodos y herramientas para facilitar el acercamiento y fomentar el interés de los niños en la tecnología.

Este laboratorio realiza investigaciones sobre el impacto de la tecnología en zonas de escasos recursos, mediante campamentos donde participan estudiantes de escuelas de dichas zonas. En estos campamentos se proporciona a los estudiantes proyectos que consisten en juguetes electrónicos, fabricados mayormente con materiales reutilizables, y se les instruye acerca de su uso. La interacción con dichos juguetes ayuda a incentivar el interés no solo en la tecnología, sino también en la aplicación de conceptos matemáticos y físicos. Además, se les permite a los estudiantes realizar proyectos propios para fomentar su creatividad.

Uno de los proyectos con los que cuenta este laboratorio es el kit de robótica costarricense: Krotic. Este proyecto se muestra en la Figura 1.1. Krotic se crea con la intención de desarrollar un kit de robótica de bajo costo para uso en la educación nacional, de modo que sea accesible para zonas rurales o de bajos recursos.

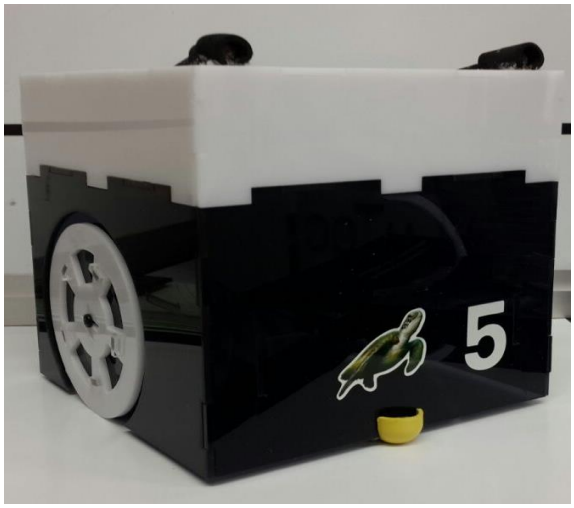

Figura 1.1 Krotic

El proyecto Krotic es un pequeño robot que cuenta con dos motores tipo servo de rotación continua para controlar el movimiento de sus ruedas, y además se le pueden acoplar distintos módulos electrónicos (luces tipo LED, sensores, etc) a conveniencia para ser programados según la actividad que desee realizar el usuario. El control de cada Krotic se realiza desde un microcontrolador ATMEGA de Atmel, compatible con la familia de dispositivos Arduino. En este dispositivo se programan distintas rutinas para ser ejecutadas por el kit, las cuales se programan ya sea mediante una Tablet o vía alámbrica. Cada kit se energiza con seis baterías tipo AA. [1]

Un servomotor de rotación continua es una variante de los servomotores normales, en los que la señal que se envía controla la velocidad de giro, en lugar de la posición angular [2]. Cuando se desea controlar la posición angular de un servomotor de rotación continua, se debe diseñar para dicho motor un sistema de control automático.

El control de servomotores se lleva a cabo mediante algoritmos de control de tipo lineal, dentro de los que pueden resaltarse, el Proporcional (P), Integral (I), y Derivativo (D), todos estos aplicados de manera independiente según sean los requerimientos del proceso a controlar, o bien; pueden funcionar de manera combinada como el PI, el PD, o el PID. [3].

1.2 Definición del problema

1.2.1 Generalidades

Como se explicó anteriormente, un servomotor de rotación continua controla únicamente la velocidad y dirección de giro. Por esta razón, cuando se trabaja con sistemas compuestos por servomotores en los que se requiere un manejo de posición, se necesita un sistema de control automático. El movimiento en el kit de robótica costarricense Krotic, es producido por servomotores de rotación continua, sin embargo, no se cuenta con un sistema de control de posición angular.

Una de las desventajas de no contar con un sistema de este tipo, es que la manera de programar instrucciones de movimiento es básicamente empírica, por lo que la precisión con la que se realizan los movimientos es muy baja y por ende se afecta todo el funcionamiento del movimiento del kit robótico.

A pesar de esta gran desventaja, el kit de robótica ha mantenido el modo actual de operación ya que, a pesar de no ser preciso, permite el funcionamiento del sistema y sus distintos módulos. Sin embargo, al encontrarse en un laboratorio de investigación, constantemente se da la oportunidad de hacer modificaciones en los proyectos, razón por la cual en este momento se busca implementar una solución, en miras de obtener un proceso de movimiento mucho más preciso.

A la hora de desarrollar un sistema de control, dentro de las muchas consideraciones que deben de tomarse en cuenta, la más importante es el cambio en el set de instrucciones que trabajarán bajo el dominio de este sistema. Esto se debe a que las mismas dejarán de ser programadas de manera empírica o experimental y ahora pasarán a ser establecidas y ejecutadas con respecto a la operación definida por el sistema de control.

Como en cualquier ocasión en la que se desarrolla un proyecto, este siempre es acompañado por una serie de limitaciones, en este caso la limitación principal a tomar en cuenta es el presupuesto disponible para el desarrollo del proyecto. El kit robótico Krotic nace y se desarrolla como una alternativa robótica de bajo presupuesto, que busca ser implementado y usado en zonas urbano-marginales de nuestro país. Esto presenta un desafío todavía mayor pues no solo se debe de diseñar e implementar un sistema de control funcional, si no que el desarrollo e implementación de este no debe de exceder los $5.

1.2.2 Síntesis del problema

¿Cómo desarrollar un sistema de control automático de posición angular, que optimice las funciones de movimiento del kit de robótica Krotic, con un presupuesto que no exceda los $5?

1.3 Meta

Optimizar el kit de robótica Krotic mediante el diseño de un sistema de control automático para la ejecución de instrucciones de movimiento, además de actualizar el set de instrucciones actual por uno que se adecue de mejor manera al sistema de control implementado.

Indicador: El kit de robótica Krotic cuenta con un sistema de control de posición angular, el cual contempla una mejora significativa en la ejecución de las funciones de movimiento, además cuenta un set de instrucciones que se adecua a este sistema de control.

1.4 Objetivos

1.4.1 Objetivo General

Diseñar e implementar un sistema de control automático encargado de controlar el movimiento kit de robótica Krotic, así como un nuevo set de instrucciones que se adecue a éste.

1.4.2 Objetivos Específicos

i. Diseñar codificadores de cuadratura de bajo costo, con sensores ópticos de reflexión y transmisión, y elegir el más adecuado para el kit de robótica.

ii. Diseñar e implementar el sistema de control de posición angular de los servomotores, que se adecue a los parámetros obtenidos de los motores del kit de robótica Krotic.

iii. Reestructurar e implementar un nuevo set de instrucciones de movimiento para el kit de robótica, para que se adecue al sistema de control previamente diseñado.

Capítulo 2. Estrategia de solución

Cuando se trabaja con servomotores, en especial cuando estos deben funcionar de forma conjunta, es común implementar un sistema de control automático para mejorar el desempeño. En el caso del kit de robótica Krotic, se utilizan dos servomotores para realizar las funciones de movimiento, y se desea implementar sobre estos un sistema de control de posición angular en lazo cerrado.

Un sistema de control de posición cuenta con un controlador, encargado de enviar señales para el funcionamiento del servomotor, así como un lazo de realimentación para medir el error del sistema. El error del sistema es la diferencia entre la posición ejecutada y la posición de referencia o deseada. Este sistema se muestra en la Figura 2.1.

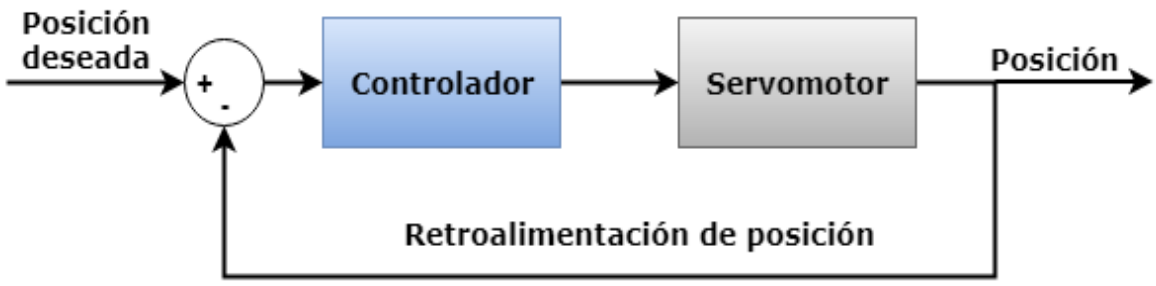

Figura 2.1 Diagrama de sistema de control de posición

A continuación, se explicarán algunos de los detalles necesarios para el diseño e implementación del sistema de control.

El controlador

El controlador es el sistema encargado de comparar el valor real de la salida con el valor de referencia (error), y con esto producir la acción de control necesaria para reducir o eliminar dicho error. Como se está trabajando con un sistema con una única entrada y una única salida (SISO), se propone utilizar un control PID. Los miembros de la familia de controladores PID, incluyen tres acciones: proporcional (P), integral (I) y derivativa (D). Estos controladores son los denominados P, I, PI, PD y PID.

• Acción de control proporcional: produce una señal de control proporcional a la señal de error. Esta acción puede reducir, pero no eliminar, el error en estado estacionario.

• Acción de control integral: proporciona una corrección para compensar las perturbaciones y mantener la variable controlada en el punto de consigna. Puede eliminar errores estacionarios, sin embargo, dependiendo del uso que se le dé puede inestabilizar el sistema.

• Acción de control derivativa: Anticipa el efecto de la acción proporcional para estabilizar más rápidamente la variable controlada después de cualquier perturbación. [4]

Los servomotores

Los servomotores que utiliza el kit de robótica Krotic, son servomotores de rotación continua de la marca Parallax. Algunas de sus características son:

- Rotación continua bidireccional
- 0 a 50 RPM, con una respuesta lineal a PWM para facilitar rampa
- Acepta cuatro tornillos de montaje
- Fácil de interconectar con cualquier microcontrolador Parallax o dispositivo con capacidad PWM
- Peso: 1.50 oz (42.5 g)
- 8 oz-in torque @ 6 V [5]

Si bien se cuenta con algunas características genéricas de los motores, se necesitan realizar pruebas para obtener características específicas del funcionamiento como parte de un sistema, tales como las velocidades apropiadas para el movimiento y el consumo energético según las velocidades elegidas.

El lazo de realimentación

Como parte del lazo de realimentación, se necesita una manera de obtener las señales de la salida, el dispositivo más común para este fin es un codificador de cuadratura. Un codificador es un dispositivo electromecánico que puede medir el movimiento o la posición. La mayoría de los codificadores utilizan sensores ópticos para proporcionar señales eléctricas en forma de trenes de impulsos, los cuales pueden a su vez, traducirse en movimiento, dirección o posición

Los codificadores rotativos se utilizan para medir el movimiento de rotación de un eje. La figura 2.2 muestra los componentes fundamentales de un codificador giratorio, que consisten en un sensor óptico ya sea de reflexión o transmisión y un disco con ranuras montado en el eje de rotación. Al girar el disco, el sensor cuenta las ranuras que pasan por él, lo que genera los pulsos de una forma de onda cuadrada, la cual puede luego ser interpretada como posición o movimiento. [6]

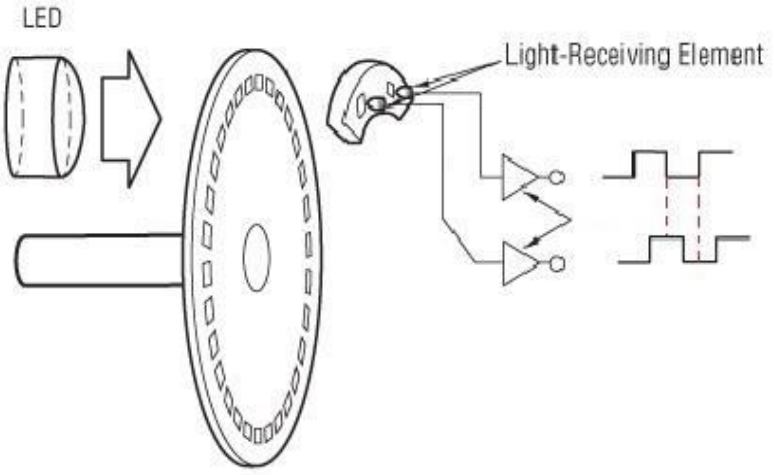

Figura 2.2 Partes de un codificador [6]

Un codificador que suministrase una simple serie de pulsos no sería útil porque no podría indicar la dirección de la rotación. Al usar dos pistas codificadas con sectores de posición desfasados 90 grados (figura 2.3) se puede indicar tanto la posición como la dirección de la rotación mediante los dos canales de salida del codificador de cuadratura. Por ejemplo, si ocurre primero un pulso en A y luego en B, el disco está girando en sentido horario. Si tiene lugar primero un pulso en B y luego en A, entonces el disco está rotando en el sentido inverso a las agujas del reloj. Por lo tanto, si se monitoriza tanto el número de pulsos como la fase relativa de las señales A y B, se puede hacer un seguimiento de la posición y de la dirección de la rotación. [6]

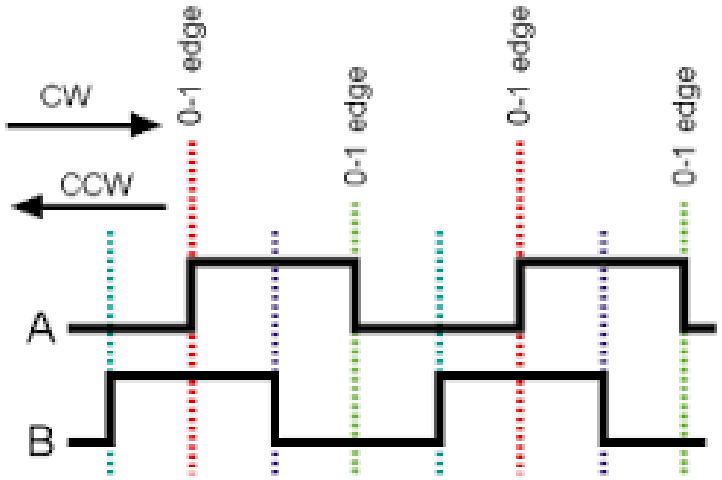

Figura 2.3 Señales A y B de un codificador de cuadratura [6]

Aquí es donde entra uno de los mayores desafíos, el presupuesto. Krotic fue diseñado como un robot de bajo costo, de manera que se pueda financiar para zonas de escasos recursos. Por esta razón, cualquier solución que se pueda plantear debe ser de un costo que no supere los $5.

Si bien en el mercado existen múltiples tipos de codificadores de cuadratura, no existe ninguno cuyo costo esté dentro del presupuesto, por lo que se necesitará diseñarlos con sensores de bajo costo. Se presentan dos propuestas para la elección de dichos sensores, ambas serán probadas de modo que se tenga de manera práctica pruebas de su funcionamiento, y se logre obtener una lista de sus ventajas y desventajas al ser implementadas en el kit de robótica.

Propuesta 1: Sensores ópticos de transmisión.

En los sensores ópticos de transmisión, el emisor y receptor del detector forman cuerpos separados. El emisor produce un haz de luz que en reposo llega al receptor creándose una especie de barrera de luz. Cuando un objeto interfiere en el haz de luz, el receptor deja de recibirlo, modificando su salida, como se puede observar en la Figura 2.4. [7]

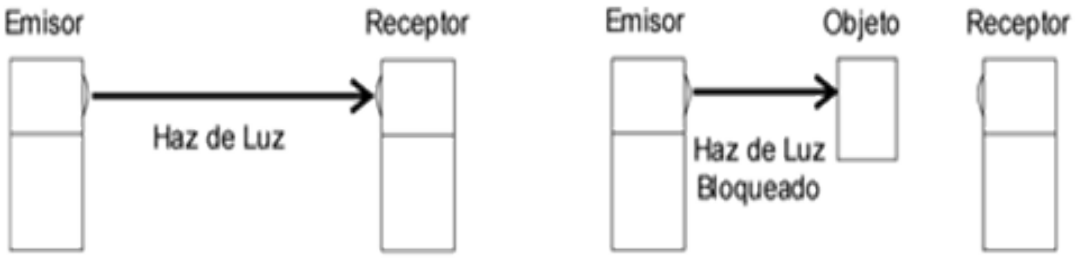

Figura 2.4 Sensor óptico de transmisión [7]

Propuesta 2: Sensores ópticos de reflexión.

Los sensores ópticos de reflexión son sensores que detectan la presencia de materiales que interfieren con la propagación de un haz de luz que genera el propio detector. El haz de luz es generado y emitido por diodos electroluminiscentes (LED) y recibido por un fototransistor, como se muestra en la siguiente figura. [7]

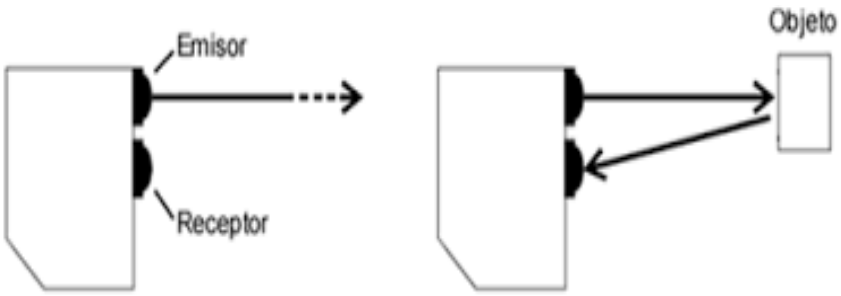

Figura 2.5 Sensor óptico de reflexión [7]

Cabe mencionar que, para el uso del sensor de reflexión, se necesitará que el disco esté ya sea hecho o forrado de un material reflectivo, para lo que se tiene pensado utilizar acrílico de espejo.

Para ambos sensores, un punto importante es la selección de la configuración para lograr la cuadratura. En la figura 2.6 se puede observar las dos propuestas que se considera favorecen de mejor manera el diseño, y evitan modificar en gran manera la estética del robot.

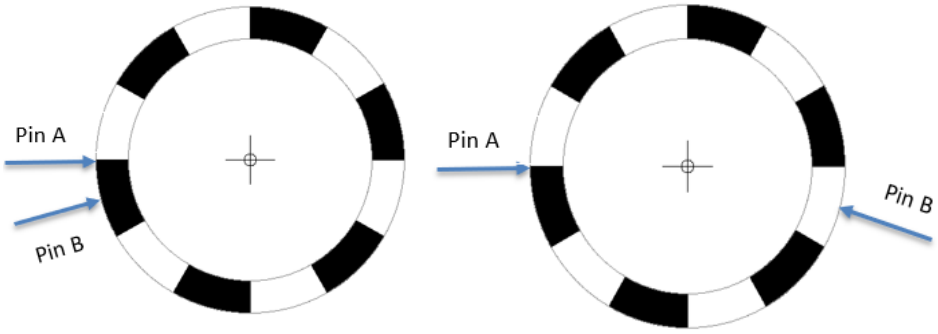

Figura 2.6 Propuestas para la configuración de la cuadratura

Una de las consideraciones a tomar en cuenta para el diseño del sistema de control de posición, es que la velocidad con la que girarán los motores será constante mientras se mantenga en uno de los niveles de batería (Alto-medio-bajo). Por ejemplo, mientras la batería se mantenga en un nivel alto, los motores trabajarán a 0.05 m/s. Estas velocidades se definirán mediante mediciones de consumo energético de los servomotores, de modo que se optimice también el rendimiento de las baterías del robot.

Para comprobar el funcionamiento del sistema de control automático, se deberán probar los tipos de movimientos que se presentan en la Tabla 1.1, y se debe cumplir con la tolerancia establecida previamente por el encargado del proyecto Milton Villegas Lemus.

Tabla 2.1 Tipo de movimientos y tolerancia

Tipo de movimiento	Tolerancia
Giro Izquierda	±5°
Giro Derecha	±5°
Avanzar hacia adelante	±2 cm
Avanzar hacia atrás	±2 cm

Una vez que se cuente con el sistema de control para los servomotores en adecuado funcionamiento, se procederá con el rediseño de las instrucciones encargadas del movimiento y la implementación de estas. El set de instrucciones de movimiento actual se muestra en la Tabla 2.2.

Se pretende diseñar un nuevo set de instrucciones, que se adapte al sistema de control implementado, pero conservando los tipos de instrucciones planteados en la Tabla 1, con ejecuciones por distancia y por tiempo. Esto se integrará como la parte de movimiento del lenguaje de primer nivel del kit de robótica Krotic.

Tabla 2.2 Set actual de instrucciones de movimientos de Krotic

Código	**Mnemónico**	**Modo direccionamiento**	**Descripción**
1	MOT_FOR DISTANCE	Inmediato	Avanza por distancia
2	MOT _FOR TIME	Inmediato	Avanza por tiempo
3	MOT _FOR100D	Inherente	Avanza 100 unidades de distancia
4	MOT_FOR1S	Inherente	Avanza por 1 segundo
5	MOT_BAC DISTANCE	Inmediato	Retrocede por distancia
6	MOT_BAC TIME	Inmediato	Retrocede por tiempo
7	MOT_BAC100D	Inherente	Retrocede 100 unidades de distancia
8	MOT_BAC1S	Inherente	Retrocede por 1 segundos
9	MOT_ROTCW ANGLE	Inmediato	Rota ciertos grados en sentido horario
10	MOT_ROTCW TIME	Inmediato	Rotar en sentido horario por determinado tiempo
11	MOT_ROTCW90D	Inherente	Rota 90 grados en sentido horario
12	MOT_ROTCW180D	Inherente	Rota 90 grados en sentido horario
13	MOT_ROTCW1S	Inherente	Rota por un segundo en sentido horario
14	MOT_ROTCCW ANGLE	Inmediato	Rota ciertos grados en sentido antihorario
15	MOT_ROTCCW TIME	Inmediato	Rotar en sentido antihorario por determinado tiempo
16	MOT_ROTCCW90D	Inherente	Rota 90 grados en sentido antihorario
17	MOT_ROTCCW180D	Inherente	Rota 180 grados en sentido antihorario
18	MOT_ROTCCW1S	Inherente	Rota por un segundo en sentido antihorario

Capítulo 3. Descripción del proceso de diseño

3.1 Codificadores de cuadratura

Se diseñan dos codificadores de cuadratura, el primero con sensores ópticos de reflexión y el segundo con sensores ópticos de transmisión. El precio de dichos sensores ronda los $0.1, por lo que son de especial conveniencia para el sistema.

Al iniciar con el diseño de los codificadores de cuadratura, se prueba el correcto funcionamiento de los sensores ópticos, para lo cual se utilizan los circuitos que se muestran en la Figura 3.1 y la Figura 3.2.

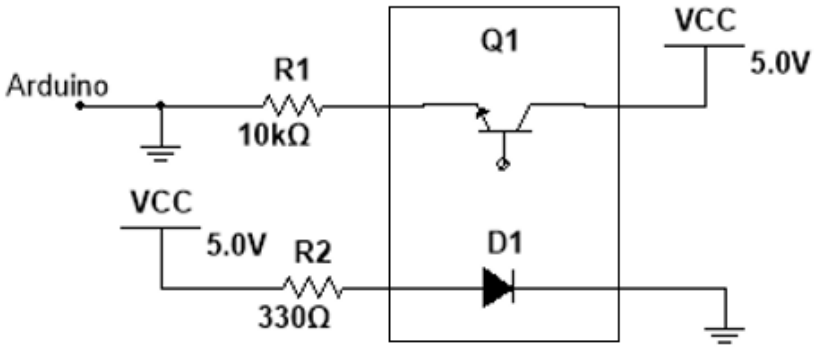

Figura 3.1 Circuito para sensor óptico de transmisión

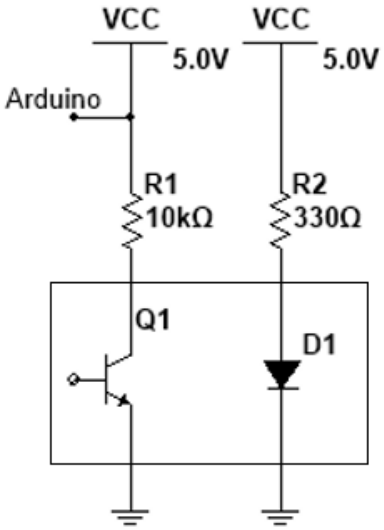

Figura 3.2 Circuito para sensor óptico de reflexión

Una vez comprobado el correcto funcionamiento de los sensores, se debe diseñar el disco con la que se planea lograr la cuadratura, para lo que se utiliza un generador de discos de cuadratura online llamado “Optical encoder wheel generator” [8]. En este programa, se ingresan las especificaciones deseadas del disco y éste genera una imagen (ver Figura 3.3). Debido a que el disco se debe cortar en un material cuyo tipo específico debe de ser acrílico, se debe crear un esquemático para la máquina encargada de realizar el corte láser. A partir de la imagen generada con el software online y el programa Adobe Illustrator, se genera dicho diseño.

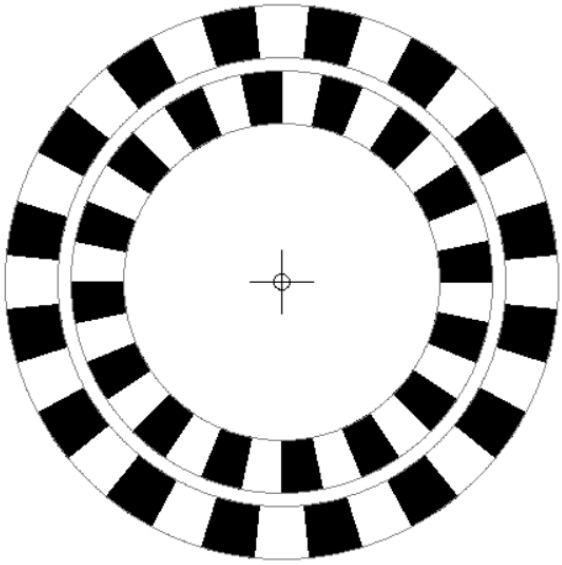

Figura 3.3 Disco generado por programa online [8]

Para el diseño inicial del disco se consideran las siguientes características.

- Dos sets de ranuras con 90° de separación
- Cada ser con 16 ranuras
- Diámetro de 7 cm

Para el corte del disco se utilizó acrílico reflectivo, debido a que éste permite un mejor funcionamiento del sensor de reflexión. En la Figura 3.4 se puede observar el resultado de disco inicial.

Figura 3.4 Disco para codificador (Versión 1)

Para la versión inicial del disco codificador se considera despreciable el grueso del láser, sin embargo, una vez que se realiza el corte se descubre que no es despreciable. Tomando esto en cuenta se realiza un nuevo diseño del disco, con ranuras cada 10° y un diámetro de 7 cm. Además, se considera que el set de ranuras interno, que se tomó en cuenta en el primer diseño para tener de referencia los 90° necesarios de cuadratura, se puede eliminar si se utiliza una referencia externa. Debido a esto, se diseña además dicha referencia. El diseño de dichas partes se realiza utilizando nuevamente el programa Adobe Illustrator y se muestra en la Figura 3.5 parte A, así como su resultado en la parte B.

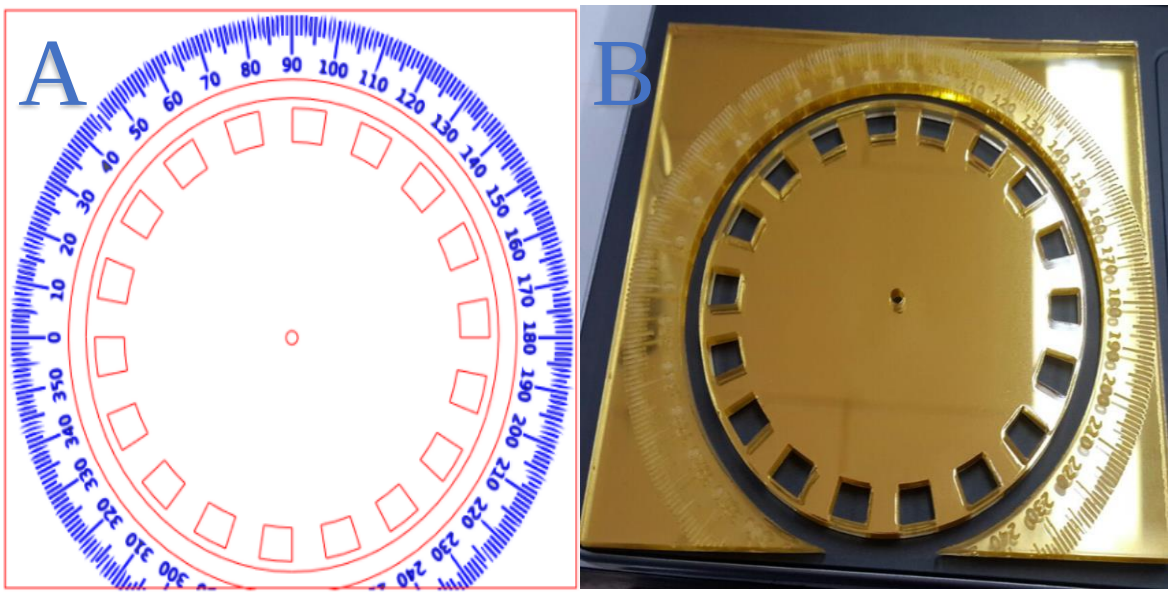

Figura 3.5 Esquemático (A) y resultado (B) de disco codificador (versión 2) y referencia externa de ángulos

Una vez que se cuenta con ambas partes, se programa un codificador sencillo para realizar pruebas de funcionamiento. Un codificador sencillo (1x), suma una unidad cada vez que detecta una ranura en sentido horario, y resta una unidad cada vez que detecta una ranura en sentido antihorario.

La primera configuración que se desea probar en la que se colocan los sensores a (180 ± 3)° de diferencia, por lo que se colocan los sensores en 0° y 183°.

Las pruebas se realizan programando el servomotor a distintas velocidades (PWM) y tomando los datos de las cuentas que realiza el codificador. Estos datos se comparan contra las cuentas que se deberían de realizar teóricamente y se obtiene el porcentaje error. Además, si se toma en cuenta que cada disco es de 7 cm de diámetro y tiene 18 ranuras, se realiza una relación matemática y se obtiene el error en centímetros. Los resultados de estos errores se pueden observar en la Tabla 3.1. (Ver resultados completos de pruebas en Apéndice A1)

Tabla 3.1 Comparación de error de codificadores según velocidad (Configuración 1)

Velocidad (PWM)	Porcentaje de Error		Error (cm)	
	Reflexión	Transmisión	Reflexión	Transmisión
0	0,906	0,719	1,525	1,220
20	0,000	1,071	0,000	1,830
40	1,642	1,268	2,745	2,135
80	1,718	2,115	2,745	3,355
86	0,781	3,571	0,610	2,745
87	0,000	1,744	0,000	0,915
95	1,923	1,359	2,135	1,525
100	0,758	2,799	1,220	4,575
110	0,543	0,725	0,915	1,220
120	1,460	1,825	2,440	3,050
140	0,362	0,899	0,610	0,915
180	0,362	0,906	1,830	2,135

Para la segunda configuración se necesita que ambos sensores estén unidos, además de esto el disco debe cumplir con un requerimiento específico. El ancho de la ranura debe ser igual al de la unión de ambos sensores, es decir, ambos sensores deben caber en cada ranura. De esta manera se aseguran los 90° de cuadratura.

Para esto se diseña un nuevo disco, esta vez con un diámetro de 7.1 cm y un total de 9 ranuras. El diseño y resultado de este se muestran en la Figura 3.6.

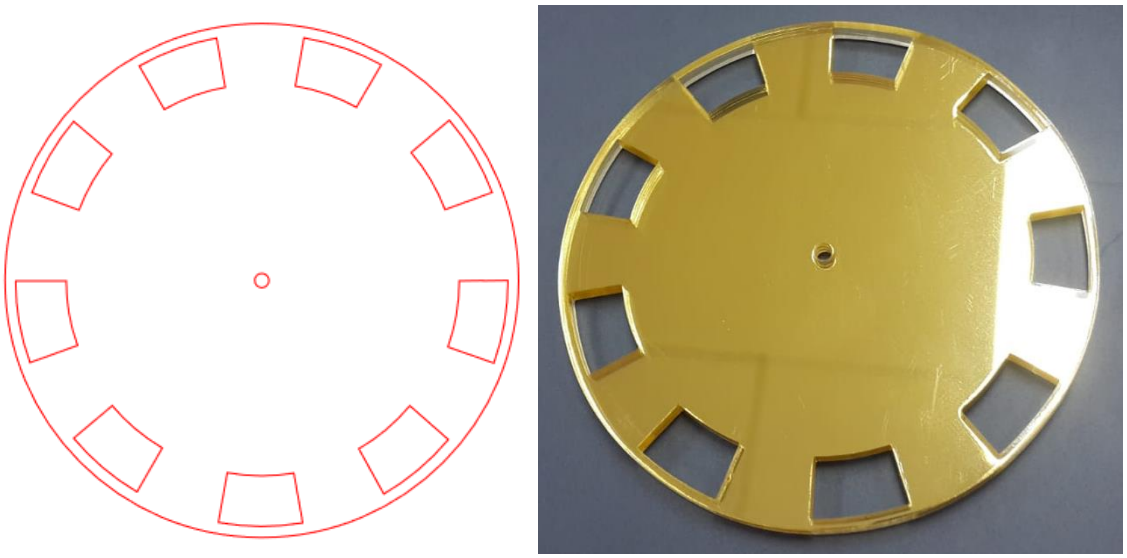

Figura 3.6 Esquemático y resultado de disco codificador (versión 3)

Cuando se cuenta con el nuevo disco se inician las pruebas con la segunda configuración. De la misma manera que las primeras pruebas, se obtiene el porcentaje de error con la diferencia entre los datos experimentales y los teóricos, y con una relación matemática entre el diámetro y la cantidad de ranuras se aproxima el error en centímetros. Los resultados de dichos errores se muestran en la tabla 3.2. (Ver resultados completos de pruebas en Apéndice A2)

Tabla 3.2 Comparación de error de codificadores según velocidad (Configuración 2)

Velocidad (PWM)	Porcentaje de Error		Error (cm)	
	Transmisión	Reflexión	Transmisión	Reflexión
0	0,746	0,758	0,610	0,610
20	1,493	0,373	1,220	0,305
40	1,119	0,000	0,915	0,000
80	0,397	0,000	0,305	0,000
85	0,595	0,000	0,305	0,000
95	1,744	1,136	0,915	0,610
100	0,781	0,769	0,610	0,610
120	0,373	0,368	0,305	0,305
140	0,000	0,735	0,000	0,610
180	0,000	0.746	0,000	0.610

Con el fin de determinar tanto la configuración como el tipo de sensores que presentan mejor desempeño, se realiza un gráfico de comparación con los errores obtenidos en las pruebas de funcionamiento. Dicho gráfico se muestra en la Figura 3.7. Como se observa, la configuración 2 muestra mejores resultados que la configuración 1, lo que se puede atribuir tanto a que los sensores son capaces de obtener una mejor lectura cuando las ranuras presentan mayor área, como a que se pueden haber presentado errores en la colocación de los sensores.

Si se toma entonces la configuración 2, se tiene los sensores que producen un menor error son los de reflexión. Además, poseen la ventaja de que no afectan la estructura externa del kit de robótica, que es uno de los factores importantes en las especificaciones iniciales. Tomando en cuenta estas consideraciones, y que el costo económico de los sensores no difiere, se elige trabajar con la configuración 1 y los sensores de reflexión para el diseño del sistema de control de posición angular.

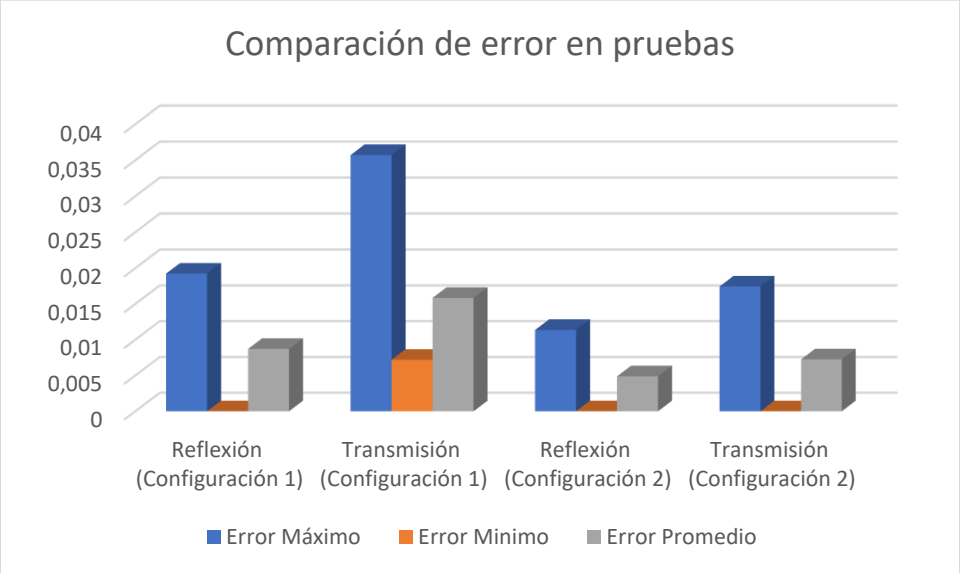

Figura 3.7 Comparación de error en pruebas

Una de las desventajas de elegir la configuración 1, es que esta disminuye a la mitad la precisión del sistema de control, ya que el disco presenta la mitad de las ranuras que el disco de la configuración 2. Entonces, para mejorar la precisión del codificador manteniendo el bajo porcentaje de error de la configuración 2, se plantea como solución la programación del codificador con precisión 4x en lugar del codificador simple que se planeaba implementar inicialmente.

Un codificador con precisión 4x toma en cuenta 4 registros, los registros de las lecturas actuales de los pines A y B, y los registros de las lecturas anteriores de estos pines. De este modo, se mejora 4 veces la precisión del codificador, ya que ahora por cada ranura se realizan 4 cuentas en lugar de 1. La tabla de verdad con la que se programan los codificadores 4x se muestra en la tabla 3.3.

Tabla 3.3 Tabla de verdad para programación de codificador 4x

Pin B	Pin A	Pin B anterior	Pin A anterior	Resultado
0	0	0	0	0
0	0	0	1	1
0	0	1	0	-1
0	0	1	1	2
0	1	0	0	-1
0	1	0	1	0
0	1	1	0	-2
0	1	1	1	1
1	0	0	0	1
1	0	0	1	-2
1	0	1	0	0
1	0	1	1	-1
1	1	0	0	2
1	1	0	1	-1
1	1	1	0	1
1	1	1	1	0

3.2 Sistema de control de posición angular

Una de las claves importantes para el correcto diseño de un sistema de control automático, es el modelo matemático de la planta. Un modelo matemático es una abstracción de un sistema real que pueda ser utilizado para propósitos de predicción y control. Este modelo debe ser capaz de analizar cuando uno o más cambios en algún aspecto del sistema modelado, puede afectar a otros aspectos del sistema o bien al sistema en conjunto. [9]

Existen dos maneras de obtener un modelo para una planta, de manera analítica (teórica) y de manera empírica. El modelado analítico busca plantear la relación a través de la aplicación sistemática de las leyes de naturaleza a los fenómenos, sin embargo, es imposible capturar perfectamente todos los detalles del comportamiento real del proceso de forma matemática [10]. Por otro lado, un estudio empírico, busca esta desconocida relación por algunas funciones matemáticas, usando la información recogida experimentalmente del sistema, especialmente cuando el proceso es demasiado complicado.

Cuando se trabaja con servomotores, la mejor manera de obtener el modelo de la planta es la empírica, debido a que de manera teórica es sumamente complicado obtener algunas de las variables del proceso. Sin embargo, es conveniente obtener el modelo teórico para tener una aproximación de la forma y el orden del sistema.

3.2.1 Modelo analítico

Para la obtención del modelo analítico se utiliza la representación del motor de la Figura 3.8.

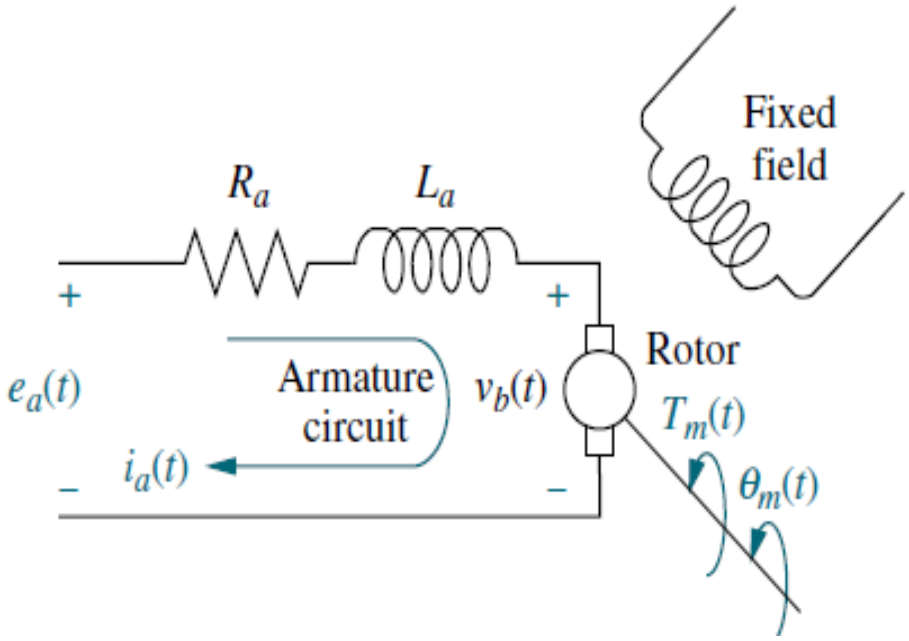

Figura 3.8 Circuito para obtención del modelo numérico

Donde:

e_a *: tensión de entrada*

e_b *: tensión de salida*

R_a *: resistencia de armadura*

L_a *: inductancia de armadura*

i_a *: corriente de armadura*

K_T *: constante del motor*

B : fricción viscosa

J : inercia

t_m *: torque del motor*

K_b *: constante fem*

ω_m *: velocidad angular*

θ_m *: posición angular*

Ecuación eléctrica:

Si se suman los voltajes de la malla marcada se obtiene:

$$e_a - e_b = V_{Ra} - V_{La} \tag{1}$$

donde:

$$V_{Ra} = R_a\, i_a \tag{2}$$

$$V_{La} = L_a \frac{di_a}{dt} \tag{3}$$

Ahora, se inserta (2) y (3) en (1):

$$e_a - e_b = R_a\, i_a + L_a \frac{di_a}{dt} \tag{4}$$

Tomando en cuenta que la tensión de salida es inducida por la velocidad angular del motor:

$$e_b = K_b \omega_m = K_b \frac{d\theta_m}{dt} \tag{5}$$

Y si se sustituye en (4):

$$e_a = R_a\, i_a + L_a \frac{di_a}{dt} + K_b \frac{d\theta_m}{dt} \tag{6}$$

Ecuación mecánica:

Se tiene que:

$$\tau_m = J \frac{d^2\theta_m}{dt^2} + B \frac{d\theta_m}{dt} \tag{7}$$

Además:

$$\tau_m = K_t\, i_a \tag{8}$$

Si se sustituye (8) en (7)*:*

$$K_t\, i_a = J \frac{d^2\theta_m}{dt^2} + B \frac{d\theta_m}{dt} \tag{9}$$

Representación en espacio de estados

Se seleccionaron como variables de estado la posición angular θ_m , la velocidad angular ω_m y la corriente de armadura i_a .

Entonces se tiene:

$$X_1 = \theta_m$$

$$\frac{dX_1}{dt} = \omega_m = X_2$$

$$X_2 = \frac{d\theta_m}{dt} = \omega_m$$

$$\frac{dX_2}{dt} = \frac{K_t}{J} i_a - \frac{B}{J} \frac{d\theta_m}{dt} = -\frac{B}{J} X_2 + \frac{K_t}{J} X_3$$

$$X_3 = i_a$$

$$\frac{dX_3}{dt} = \frac{e_a}{L_a} - \frac{R_a}{L_a} i_a - \frac{K_b}{L_a} \frac{d\theta_m}{dt} = -\frac{K_b}{L_a} X_2 - \frac{R_a}{L_a} X_3 + u$$

$$u = e_a$$

$$y = \theta_m = X_1$$

Por lo que la representación en espacio de estados queda de la siguiente manera:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & \frac{-B}{J} & \frac{K_T}{J} \\ 0 & \frac{-K_b}{L_a} & \frac{-R_a}{L_a} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L_a} \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Además, se deduce a partir de algunas de las ecuaciones del modelo matemático la función de transferencia de un servomotor:

De (6), se despeja la corriente de armadura:

$$i_a = \frac{e_a - K_b \frac{d\theta_m}{dt} - L_a \frac{di_a}{dt}}{R_a} \tag{10}$$

y se pasa al dominio de Laplace

$$I_a(s) = \frac{E_a(s) - sK_b\,\theta_m(s)}{R_a + sL_a} \tag{11}$$

Se despeja $\frac{d^2\theta_m}{dt^2}$ de (9)

$$\frac{d^2\theta_m}{dt^2} = \frac{K_t}{J} i_a - \frac{B}{J}\frac{d\theta_m}{dt} \tag{12}$$

y en el dominio de Laplace

$$s^2\theta_m(s) = \frac{K_t}{J} I_a(s) - \frac{B}{J} s\theta_m(s) \tag{13}$$

Se sustituye (3) en (4)

$$s^2\theta_m = \frac{K_t}{J}\left[\frac{E_a - sK_b\,\theta_m}{R_a + sL_a}\right] - \frac{B}{J} s\theta_m$$

$$\left(s^2\theta_m + \frac{B}{J} s\theta_m\right)(JR_a + sJL_a) = K_t(E_a - sK_b\,\theta_m)$$

$$s^2\theta_m JR_a + Bs\theta_m R_a + s^3\theta_m JL_a + Bs^2\theta_m L_a = E_aK_t - sK_b\,\theta_m K_t$$

$$\theta_m[s^3JL_a + (JR_a + BL_a)s^2 + (BR_a + K_b\,K_t)s] = E_aK_t$$

$$G(s) = \frac{\theta_m}{E_a} = \frac{K_t}{JL_a s^3 + (JR_a + BL_a)s^2 + (BR_a + K_b K_t)s}$$

Se puede reducir el grado de la ecuación si se toma en cuenta que la resistencia de armadura es mucho más grande que la inductancia:

$$\frac{L_a}{R_a} \cong 0$$

Para esto, se divide entre $R_a J$

$$\frac{\theta_m}{E_a} = \frac{\frac{K_t}{R_a J}}{\frac{\cancel{JL_a}}{\cancel{R_a J}} s^3 + \left(\frac{JR_a}{R_a J} + \frac{\cancel{BL_a}}{\cancel{R_a J}}\right) s^2 + \left(\frac{BR_a}{R_a J} + \frac{K_b K_t}{R_a J}\right) s}$$

Y simplificando queda:

$$\frac{\theta_m}{E_a} = \frac{\frac{K_t}{R_a J}}{s^2 + \left(\frac{B}{J} + \frac{K_b K_t}{R_a J}\right) s} \tag{14}$$

3.2.2 Modelo empírico

La elaboración de un modelo empírico se realiza mediante la toma de datos experimentales de la planta. Para los servomotores se recolectan datos no solo de la posición angular sino también de la velocidad angular, en caso de ser necesario un modelo de la velocidad al avanzar con el proyecto. Los datos se recolectan mediante un programa elaborado en una placa del tipo Arduino, el cual producirá un pulso de entrada de aproximadamente 10 segundos, y con un tiempo de muestreo de 10 milisegundos recolectará los datos de posición angular y velocidad angular. El programa genera con estos datos una lista separada con comas de la siguiente forma: Entrada (1/0), Posición Angular, Velocidad angular. Estos datos se procesan con el programa Excel para obtener un archivo .csv (Comma Separated Values File).

Una vez procesados los datos se consiguen importar en el programa Matlab y con ayuda de la herramienta System Identification Toolbox (ident), se pueden obtener distintas aproximaciones del modelo. System Identification Toolbox proporciona funciones MATLAB, bloques Simulink y una aplicación para construir modelos matemáticos de sistemas dinámicos a partir de datos medidos de entrada-salida. Permite crear y usar modelos de sistemas dinámicos que no se modelan fácilmente a partir de los primeros principios o especificaciones. Se puede usar datos de entrada y salida de dominio de tiempo y de dominio de frecuencia para identificar funciones de transferencia de tiempo discreto y tiempo continuo, modelos de proceso y modelos de espacio de estado. [11]

Modelo empírico de posición angular

Para el modelo de la posición, se ingresan los datos de "Entrada" y "Posición Angular" en el dominio del tiempo en la herramienta ident. La herramienta genera las curvas de entrada/salida como se muestra en la Figura 3.9, en donde "u1" representa la entrada (Entrada) y "y1" representa la salida (Posición Angular).

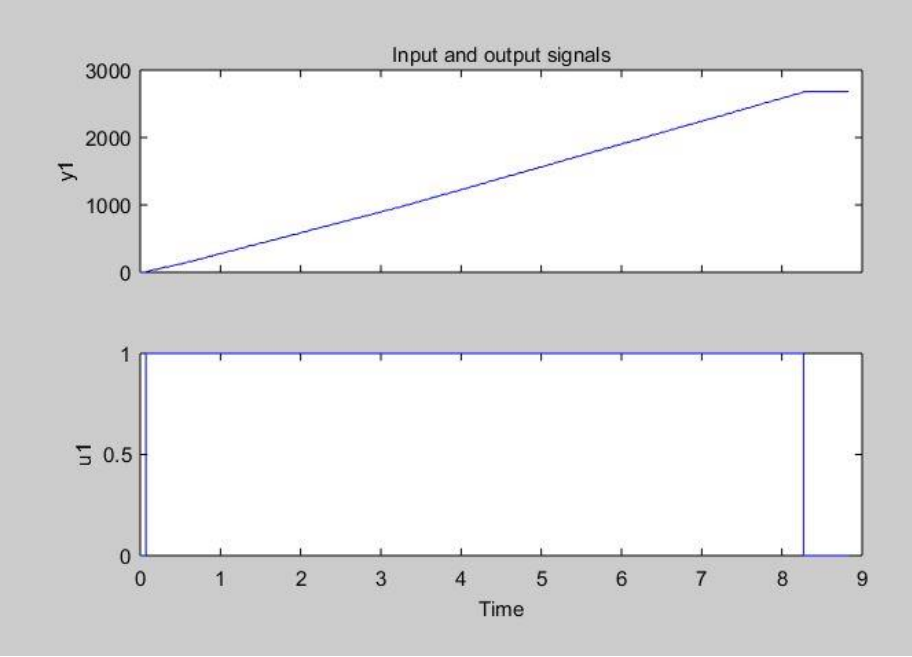

Figura 3.9 Curvas de entrada (u1) y salida (y1)

Una vez ingresados los datos, la herramienta ident cuenta con distintas funciones para la aproximación de modelos, entre las cuales están:

- Función de transferencia “tf”
- Función en espacio de estados “ss”
- Modelo polinomial “arx”
- Función de transferencia ajustable “PDZ”

Utilizando dichas funciones se calculan distintos modelos, los que más se aproximan son:

- tf3: Función de transferencia con 3 polos y 2 ceros
- ss1: Función en espacio de estados de orden 3
- ss1: Función en espacio de estados de orden 2
- P2DZU: Función de transferencia con 2 polos complejos y 1 cero
- arx221: Modelo polinomial de con 2 polos 2 ceros y constante
- arx321: Modelo polinomial de con 3 polos 2 ceros y constante

En la Figura 3.10 se muestra una comparación de las aproximaciones a la curva de la función de posición angular.

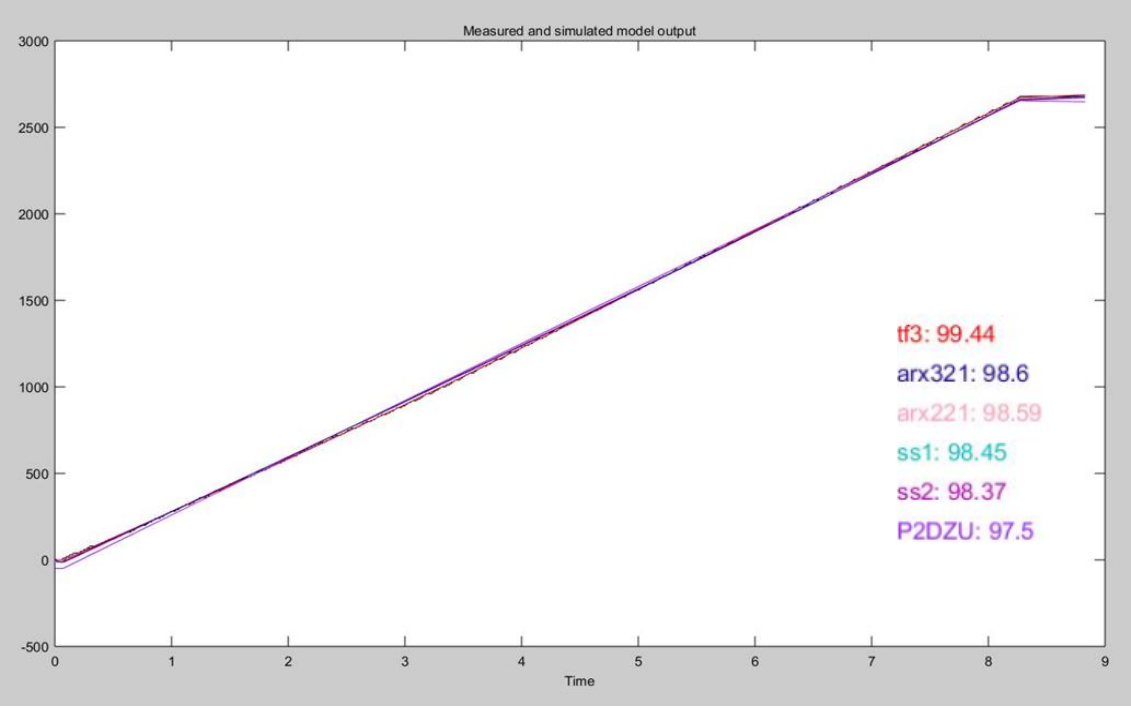

Figura 3.10 Comparación de las aproximaciones a la curva de posición angular

De las tres mejores aproximaciones se obtienen los diagramas de polos y ceros, con el fin de conocer de mejora manera el comportamiento de dichas funciones. En estos diagramas se tiene que los modelos con excepción del 'tf3' poseen polos en el lado positivo del eje real, por lo que se consideran inestables.

Sabiendo que el modelo de la función de transferencia con tres polos y dos ceros es el único que no presenta inestabilidad, se elige como modelo empírico de la posición angular. La función de transferencia de dicho modelo se presenta en la ecuación 15. El diagrama de polos y ceros del modelo tf3 se muestra en la Figura 3.11.

$$tf3 = \frac{299.6\,s^2 + 66.6\,s + 108.7}{s^3 + 0.2288\,s^2 + 0.3324\,s + 4.502x10^{-7}} \tag{15}$$

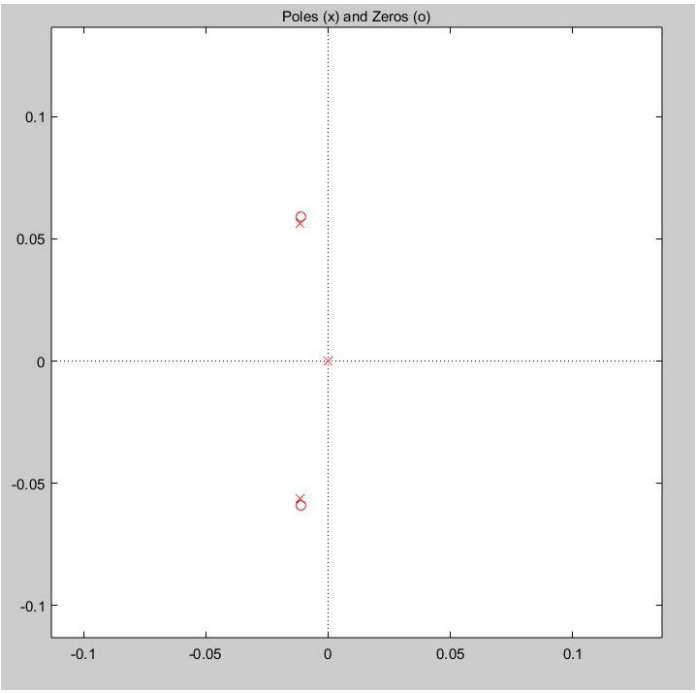

Figura 3.11 Diagrama de polos y ceros de modelo 'tf3'

Modelo empírico de velocidad angular

Para el modelo de la velocidad, de la misma manera que en el modelo de posición, se ingresan los datos de “Entrada” y “Velocidad Angular” en el dominio del tiempo en la herramienta ident. La herramienta genera las curvas de entrada/salida como se muestra en la Figura 3.12, en donde “u1” representa la entrada (Entrada) y “y1” representa la salida (Velocidad angular).

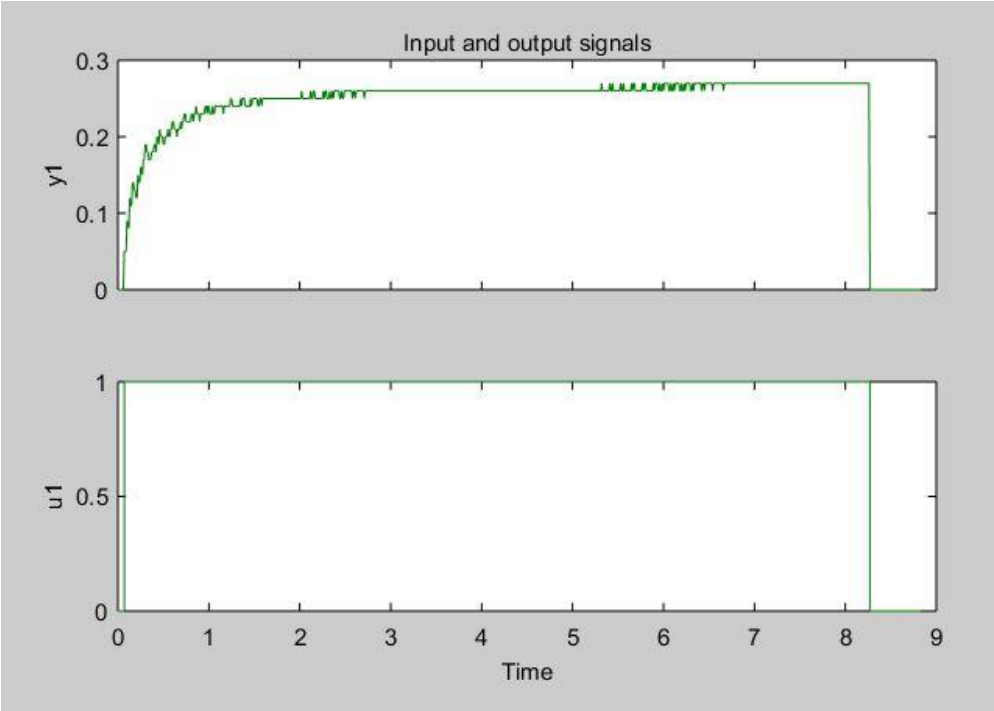

Figura 3.12 Entrada vs Velocidad Angular

Los modelos que más se aproximan a la curva de velocidad angular son:

- tf5: Función de transferencia con 2 polos y cero
- tf6: Función de transferencia con 2 polos y 2 ceros
- tf7: Función de transferencia con 3 polos y 3 ceros
- arx221: Modelo polinomial de con 2 polos 2 ceros y constante
- arx321: Modelo polinomial de con 3 polos 2 ceros y constante

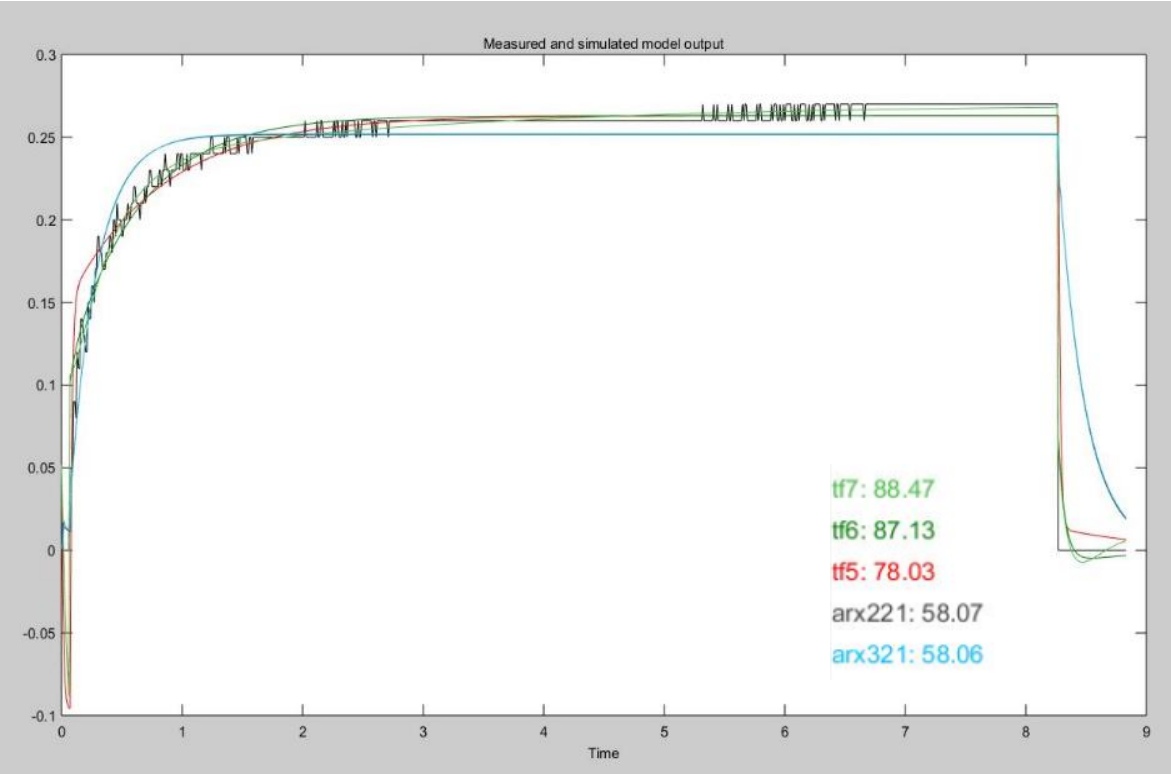

Figura 3.13 Comparación de aproximaciones a curva de velocidad angular

De igual manera, se obtienen los diagramas de polos y ceros con el fin de conocer de mejor manera el comportamiento de dichas funciones. Estos diagramas presentan estabilidad, por lo que no se descarta ningún modelo, sin embargo, se escoge el modelo de la función "tf6", debido a su alta aproximación y bajo orden (orden 2). La función de transferencia de este modelo se presenta en la ecuación 16, y su diagrama de polos y ceros en la Figura 3.14.

$$tf6 = \frac{0.1952\,s^2 + 5.276\,s + 8.221}{s^2 + 19.97\,s + 31.26} \tag{16}$$

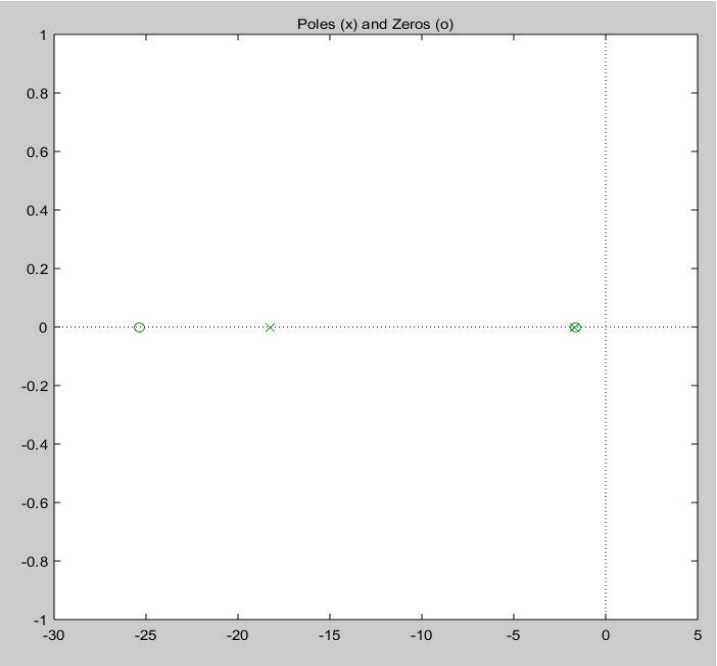

Figura 3.14 Diagrama de polos y ceros de modelo 'tf6'

3.2.3 Estimación de control de posición angular por medio de simulaciones.

Una herramienta sumamente útil en la estimación de sistemas de control de tipo PID es “PID Tuner” de Matlab. La aplicación PID Tuner sintoniza automáticamente las ganancias de un controlador PID para una planta SISO para lograr un equilibrio entre el rendimiento y la robustez. Se puede especificar el tipo de controlador, como PI, PID con filtro derivado o controladores PID de dos grados de libertad (2-DOF). Los diagramas de análisis le permiten examinar el rendimiento del controlador en los dominios de tiempo y frecuencia. Se puede refinar de forma interactiva el rendimiento del controlador para ajustar el ancho de banda del bucle y el margen de fase, o para favorecer el seguimiento del punto de referencia o el rechazo de perturbaciones. [12]

Para la estimación de los controladores, inicialmente se introduce la función de transferencia del modelo previamente seleccionado en el Workspace de Matlab. Luego se abre la herramienta PID Tuner y se importa dicha función. Ahora se elige el tipo de controlador que se desea aproximar y se ajustan los valores hasta obtener los parámetros deseados. Se considera para este sistema como parámetros deseados, un sobre impulso menor a 3% y un tiempo de estabilización de entre 0.3 y 0.5 segundos.

Se realizan simulaciones con distintos controles, de modo que se aproxime el que más se adapta al sistema. El primer control que se simula es el de tipo PID, el resultado obtenido que mejor se ajusta a los parámetros se muestra en la Figura 3.15.

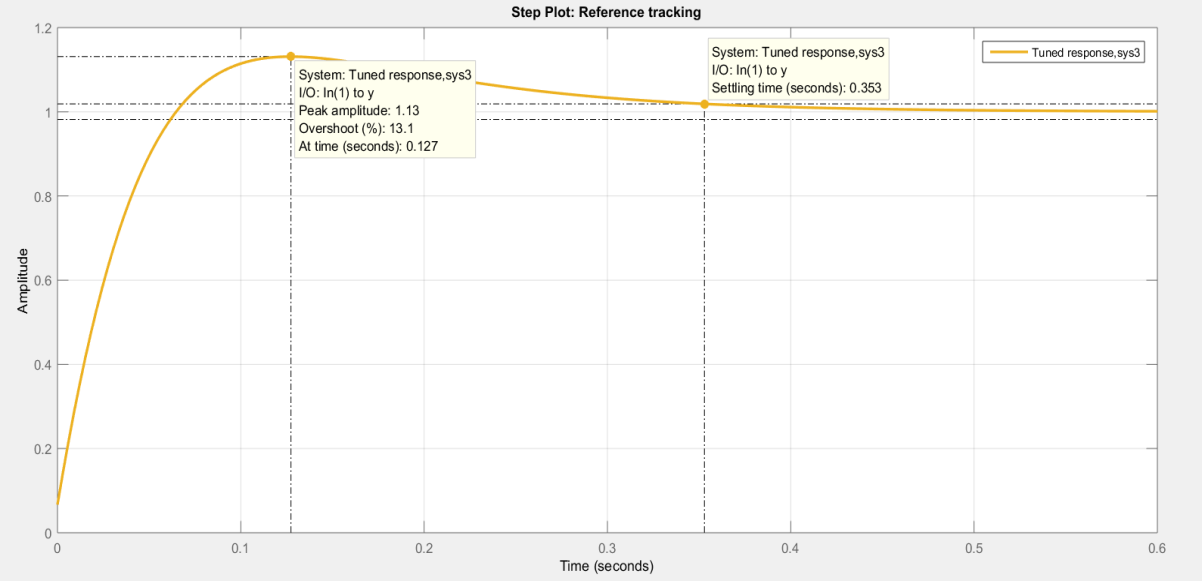

Figura 3.15 Comportamiento de sistema tipo PID

El segundo controlador que se aproxima es el de tipo PI, cuyo comportamiento se muestra en la Figura 3.16.

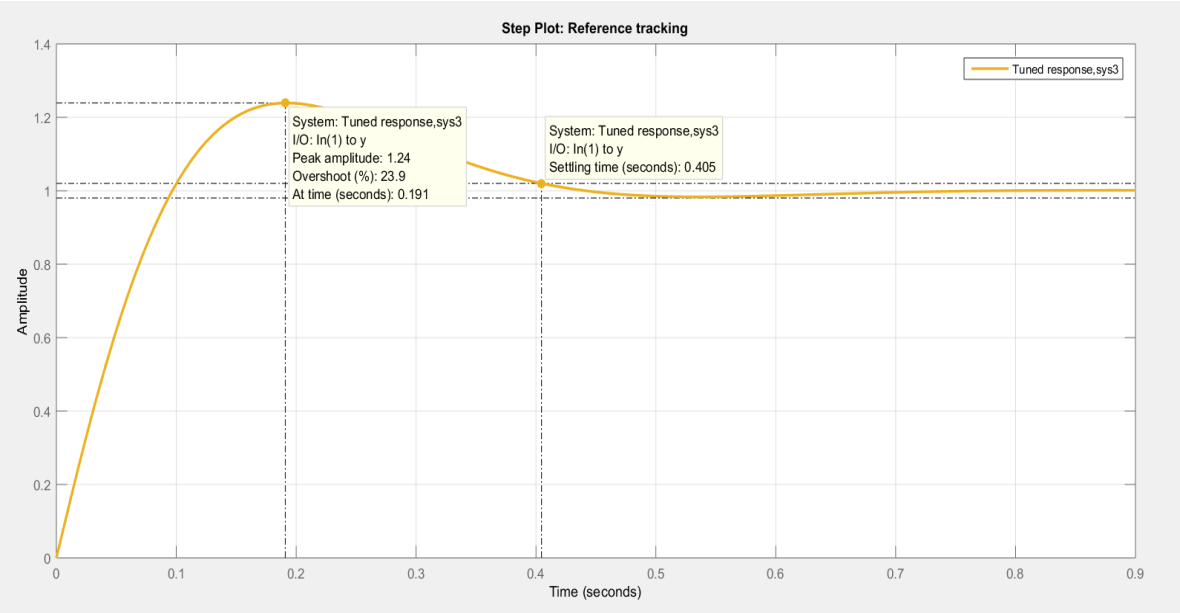

Figura 3.16 Comportamiento de sistema tipo PI

Finalmente, se aproxima el controlador de tipo PD, y su comportamiento se muestra en la Figura 3.17.

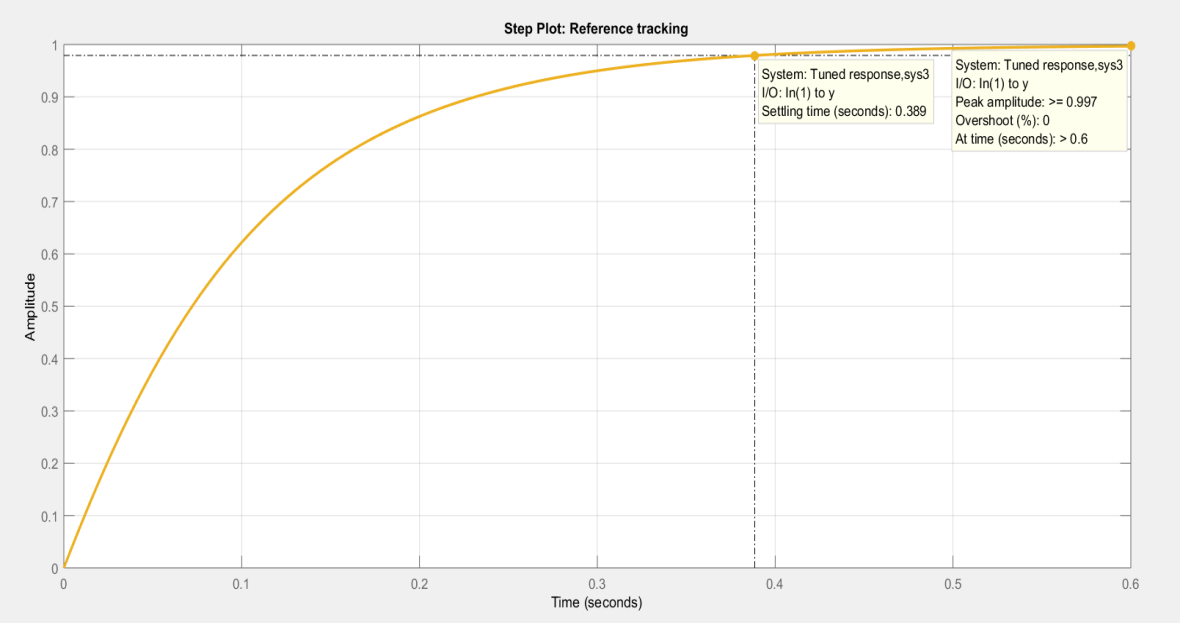

Figura 3.17 Comportamiento de sistema tipo PD

Los parámetros de funcionamiento de estos modelos se muestran en la Tabla 3.4

Tabla 3.4 Comparación de parámetros de controladores

Parámetros	**PID**	**PI**	**PD**
Kp	0.10608	0.048533	3.3243
Ki	0.77902	0.43377	n/a
Kd	0.000255	n/a	0.0315
Tiempo de subida	0.0475	0.0742	0.217
Tiempo de estabilización	0.353	0.405	0.385
Sobreimpulso	13.10%	23.90%	0%
Estabilidad	Estable	Estable	Estable

Al comparar los comportamientos de los controles obtenidos, se observa que el tipo de control que mejor se ajusta al sistema es el PD. Esto se debe a que no existe sobre impulso, de manera contraria al PID y al PI, cuyos comportamientos presentan sobre impulsos superiores a los de los parámetros deseados. Además, se logra un tiempo de estabilización acorde al sistema. Sin embargo, se probarán de manera experimental los tres controladores, para comprobar que las simulaciones se efectuaron de manera correcta.

3.2.4 Estimación de control de velocidad angular por medio de simulaciones.

Para la estimación del control de velocidad se sigue el mismo procedimiento que para el control de posición. Se considera para este sistema como parámetros deseados, un sobre impulso menor a 3% y un tiempo de estabilización de entre 0.1 y 0.3 segundos.

Se realizan simulaciones con distintos controles, de modo que se aproxime el que más se adapta al sistema. El controlador que mejor se ajusta al sistema según las herramientas de simulación es el de tipo I, o sea un integrador. El resultado obtenido que mejor se ajusta a los parámetros se muestra en la Figura 3.18.

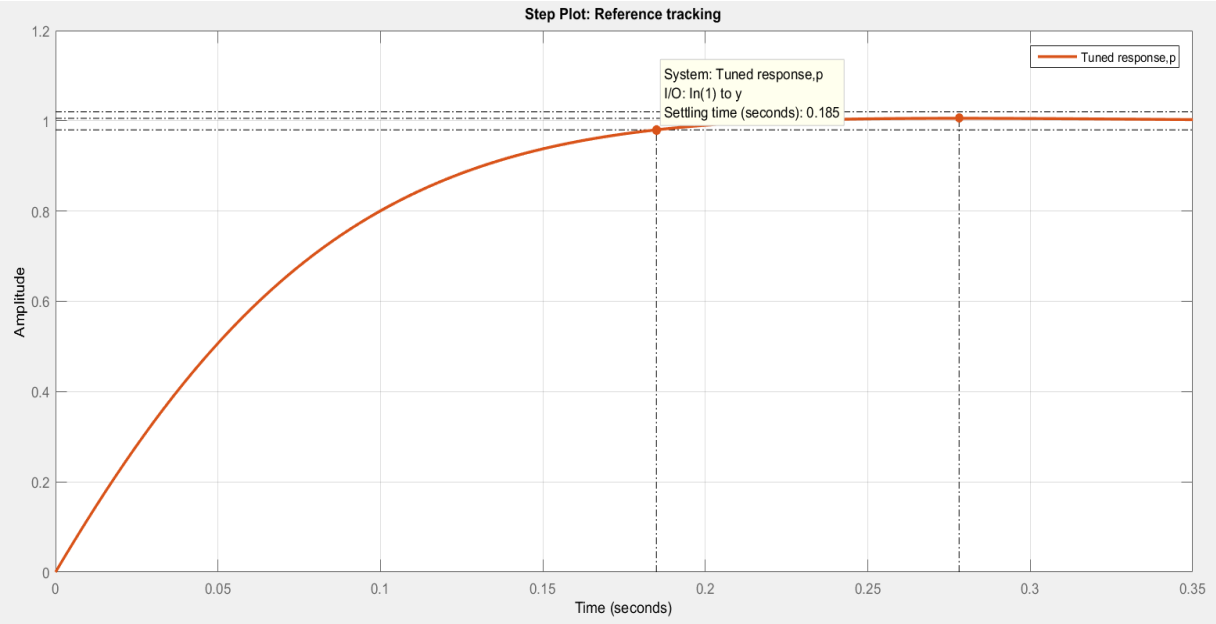

Figura 3.18 Comportamiento de sistema tipo I

Para este sistema se obtiene un tiempo de estabilización de 0.185 segundos, y un sobre impulso de 0%. La Ki de este sistema es de 62.27.

El controlador quedaría como se muestra en la ecuación 17.

$$C = 62.27 * \frac{1}{s} \tag{17}$$

De igual manera que en el sistema de control de posición, se probarán de manera experimental los distintos sistemas para comprobar la validez de las simulaciones y si se obtiene una mejor aproximación.

Capítulo 4. Implementación del diseño

La puesta en marcha del sistema de control diseñado anteriormente inicia con la elaboración de un programa en Arduino. Dicho programa es el encargado de recibir los datos provenientes de la realimentación y generar las señales de controlador (señales de tipo PWM) y enviarlas a los servomotores. Además, dentro del programa se pueden modificar las constantes del PID de manera manual, de modo que es posible probar las aproximaciones obtenidas en las simulaciones.

Para comprobar el funcionamiento tanto del programa elaborado como de los resultados de las simulaciones, se elabora un sistema externo de pruebas. Dicho sistema consta de dos servomotores colocados hacia arriba con sus respectivos codificadores y un Arduino conectado a una computadora, para ejecutar el programa cada vez que se requiera cambiar las constantes del PID.

Se realizan pruebas experimentales sobre el sistema externo con los resultados de las tres simulaciones obtenidas anteriormente, de manera que se compruebe su validez. Inicialmente se prueba con el PID y seguidamente con el PI. En ambos casos, a pesar de que en las simulaciones se observa un tiempo de estabilización bajo, en las pruebas experimentales no se logra obtener estabilización hasta al menos dos minutos después de oscilaciones. Se experimenta variando empíricamente las constantes, sin embargo, se muestra el mismo comportamiento oscilatorio, por lo que se descartan estos controles de manera definitiva.

Luego de pruebas fallidas con los controladores tipo PI y PID, se comprueba el funcionamiento de la simulación del sistema de tipo PD. Se inicia con las constantes obtenidas en las simulaciones, con las cuales se consigue un funcionamiento adecuado del sistema, no obstante, se prueba de manera empírica distintas constantes en búsqueda de un mejor resultado. Las constantes que mejor se ajustan experimentalmente al sistema son $Kp = 5$ y $Kd = 0.01$. Cabe mencionar que se mantiene un tiempo de muestreo de 10 ms.

Siguiendo el mismo procedimiento que con el sistema de control de posición, se prueba el sistema de control de velocidad. Se prueba inicialmente con el control obtenido a manera de simulación, y a pesar de que se logra eventualmente la estabilización, esto consume más tiempo del esperado. Se intenta entonces agregar una constante proporcional que ayude a mejorar el tiempo de estabilización, la cual provoca el resultado esperado. Luego de múltiples pruebas, se obtiene que las constantes que mejor se adaptan al sistema son Kp = 500 y Ki =300.

En el punto en el que ya se consiguen sistemas funcionales de control de posición y velocidad angular, se inicia con la implementación de estos en el kit de robótica. Para esto, se necesitan efectuar algunas modificaciones en la estructura interna del mismo. La modificación principal abarca un pequeño ajuste en el diseño de las ruedas, de modo que se agrega a ellas el disco del codificador.

Las ruedas originales son como se muestra en la Figura 4.1. De este diseño, se retira el disco interno (el más pequeño, color negro), y se agrega en su lugar el disco codificador. Además, para poder agregar este disco, se necesita cortar las ranuras en las que van los pines que unen los tres discos, para lo que se realiza un nuevo esquemático con la ayuda del programa Adobe Illustrator, el cual se muestra en la Figura 4.2 (A). Además, en la parte B de esta figura se muestra el resultado del corte láser una vez ensamblado en la rueda.

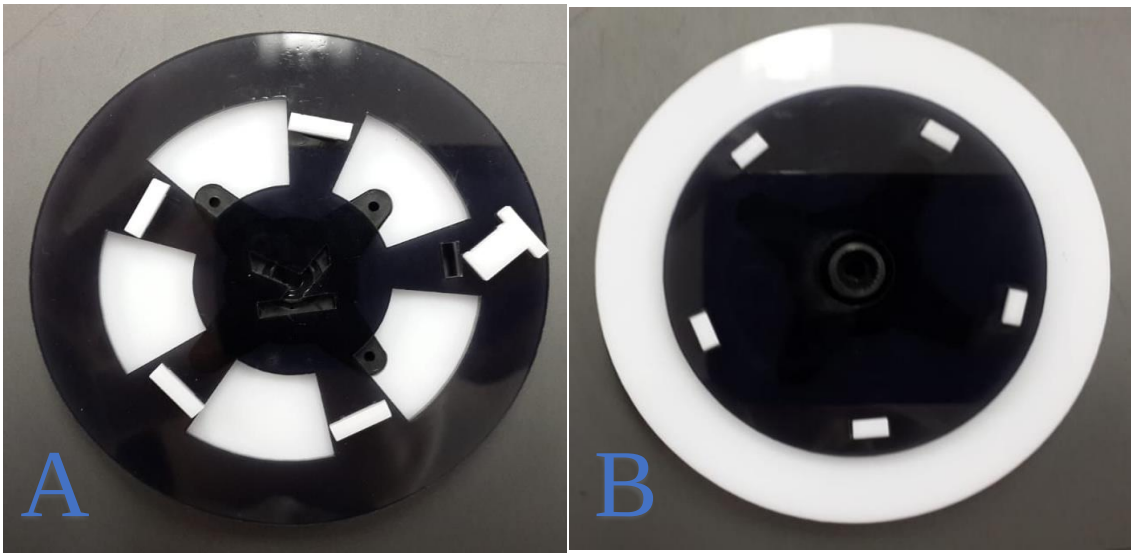

Figura 4.1 Vista frontal (A) y trasera (B) de ruedas originales de Krotic

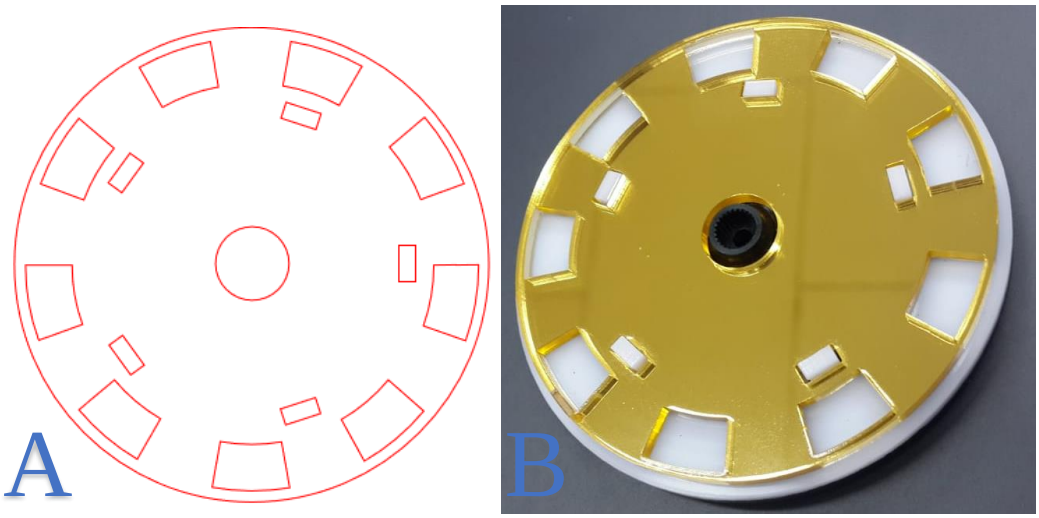

Figura 4.2 Esquemático de disco codificador (A) y resultado final del corte (B)

Además de los discos codificadores, a las ruedas del kit se le añaden ligas, con el fin de obtener mejor tracción. La segunda modificación que se debe realizar es un pequeño orificio en las paredes internas del kit que colindan con las ruedas, de modo que se pueda introducir los codificadores y queden en posición frente a los discos. La pieza y la colocación interna de los codificadores en la Figura 4.3 y la vista superior de los sensores frente a los discos se muestra en la Figura 4.4.

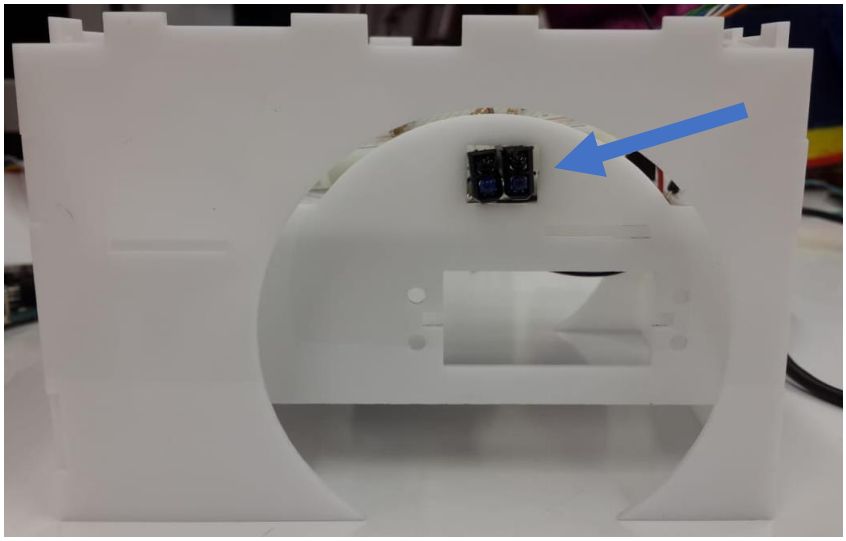

Figura 4.3 Pieza interna de Krotic y colocación de sensores.

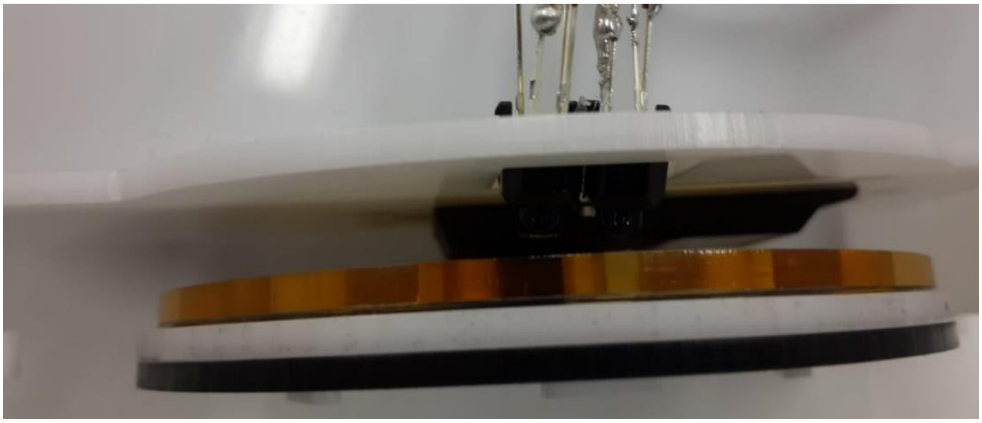

Figura 4.4 Vista superior de colocación de sensores frente a disco

El paso final luego de contar con el sistema de control adaptado en el kit de robótica es el rediseño de las instrucciones de movimiento del sistema, cuyo proceso de detalla a continuación.

4.1 Rediseño de las instrucciones de movimiento

Las instrucciones de movimiento previas al sistema de control constan de dos tipos principales, movimiento por distancia y movimiento por tiempo. Además, estos tipos de instrucción se subdividen en movimiento hacia atrás y hacia adelante y giro hacia la derecha y hacia la izquierda. La desventaja de estas instrucciones es que se utilizan distancias fijas por movimiento, por ejemplo, la instrucción 3 (Avanza 100 unidades de distancia). Esto se debe a que la manera de programar las instrucciones previas al sistema de control se realizó de manera empírica, por lo que se incurre en un mayor costo en tiempo y pruebas la programación de cada instrucción.

Al contar con el sistema de control de posición, se eliminan las pruebas empíricas al introducir la distancia deseada como “referencia” en el sistema de control. Es decir, si se desea mover el robot 30 cm, se introduce dicho valor como referencia, resultando esto en que el sistema detecte de manera automática la referencia otorgada, ejecutando de manera siguiente el movimiento.

Por lo tanto, un set de instrucciones idóneo para el sistema de control queda de la siguiente manera.

Tabla 4.1 Set de instrucciones

Código	Mnemónico	Modo direccionamiento	Descripción
1	MOT_FW_DIST	Inmediato	Avanza hacia adelante por distancia
2	MOT_FW TIME	Inmediato	Avanza hacia adelante por tiempo
3	MOT_BW_DIST	Inmediato	Avanza hacia atrás por distancia
4	MOT_BW TIME	Inmediato	Avanza hacia atrás por tiempo
5	MOT_CW_DIST	Inmediato	Gira en sentido horario por distancia
6	MOT_CW_TIME	Inmediato	Gira en sentido horario por tiempo
7	MOT_CCW_DIST	Inmediato	Gira en sentido antihorario por distancia
8	MOT_CCW_TIME	Inmediato	Gira en sentido antihorario por tiempo

Para la implementación de las instrucciones de movimiento se necesita un módulo principal que sea capaz de identificar las instrucciones y llamar a ejecución al sistema de control con los valores de referencia ingresados por el usuario. Es aquí donde el módulo principal cobra importancia, ya que este debe de ser capaz de identificar el tipo de función que se desea realizar (distancia/tiempo) y dependiendo de esto deben de existir dos submódulos capaces de identificar el tipo de movimiento (hacia adelante, hacia atrás...) y encargarse de realizar el llamado al sistema de control pertinente (posición/velocidad), haciendo uso de las referencias definidas por el usuario.

Los submódulos controladores de las instrucciones de movimiento por distancia y por tiempo funcionan según se presenta en los diagramas de flujo de las Figuras 4.5 y 4.6.

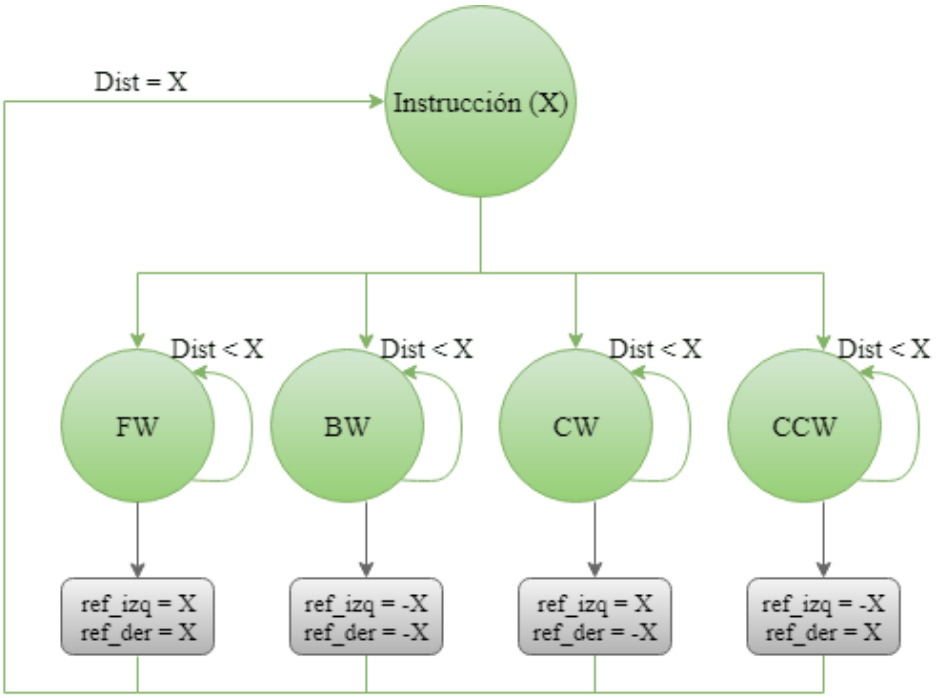

Figura 4.5 Diagrama de flujo de instrucciones de movimiento por distancia

Para las instrucciones por distancia, se utiliza como referencia para el sistema de control la distancia establecida por el usuario, siendo en este caso en particular el cumplimiento de dicha distancia lo que genera la condición de salida que finaliza la instrucción. Por otro lado, para las instrucciones por tiempo, el valor empleado como referencia para el sistema de control es definido por los valores de velocidad máximos permitidos en ese momento según la carga de la batería. Además, la condición de salida que finaliza la instrucción es el cumplimiento del tiempo establecido por el usuario. De este modo, podemos asegurar que los servomotores giren a una velocidad preestablecida hasta que dicho valor (tiempo) sea cumplido.

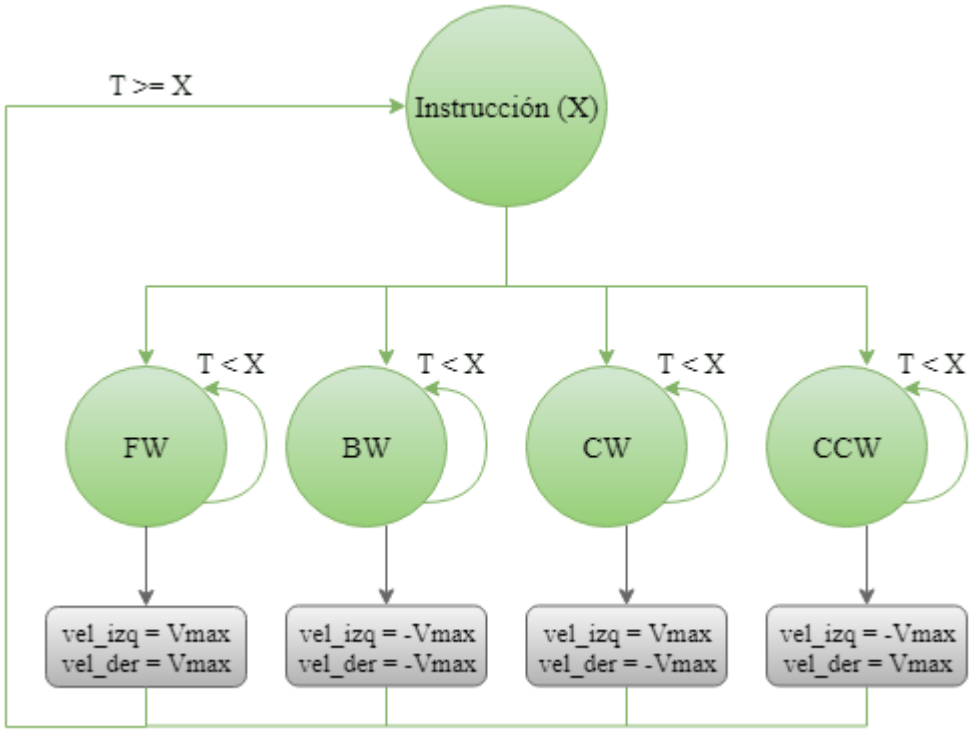

Figura 4.6 Diagrama de flujo de instrucciones de movimiento por tiempo

Como se mencionó anteriormente, otra de las funciones del módulo principal es limitar la velocidad de los servomotores según el estado de carga de la batería (ver valores establecidos en sección 5.1). Para la medición de dicho estado de carga se implementa un pequeño circuito, utilizando uno de los pines analógicos presentes en la placa Arduino, además de un divisor de tensión que limitará el voltaje suministrado a dicho pin. Los valores de carga de batería que se establecen son:

- Batería alta: Entre 6 y 4.5 voltios
- Batería media: Entre 4.5 y 3.5 voltios
- Batería baja: Menor a 3.5 voltios

Capítulo 5. Desarrollo y análisis de pruebas

5.1 Consumo energético

Para realizar las pruebas de consumo, inicialmente se desarma uno de los kit de robótica y se conectan únicamente los servomotores y el Arduino, de modo que no existan componentes que puedan afectar la medición.

Una vez conectados los servomotores se programan a diferentes velocidades para realizar las mediciones necesarias, teniendo las mismas una duración de una hora aproximadamente.

Las velocidades (PWM) con los que se realizan las pruebas son:

- 0
- 40
- 75
- 80
- 82
- 83
- 84
- 85

Se eligen estos valores de PWM debido a la conveniencia presentada para los servomotores en cuestión. Como se pudo observar en las pruebas realizadas a los codificadores, entre 0 y 75 aproximadamente la velocidad (rev/s) se mantiene constante, por lo que no resulta necesario realizar más mediciones dentro de este intervalo. Por otro lado, de 80 a 90 (punto de paro) el cambio en las velocidades es bastante notorio, por lo que es recomendado medir dichos puntos. Sin embargo, no se recomiendan PWM mayores a 85, debido a que no es suficiente para movilizar el robot de manera adecuada. Los resultados estas pruebas se presentan en la Tabla 5.1

Tabla 5.1 Mediciones de consumo

Velocidad	Duración (min)	Corriente (mA)	Voltaje inicial (V)	Voltaje final (V)	Diferencia (V)	Consumo (mW/h)
0	60	91,1	5,62	5,43	0,19	17,309
40	60	88,7	5,45	5,27	0,18	15,966
75	60	88,3	5,31	5,16	0,15	13,245
80	60	81,1	5,19	5,08	0,11	8,921
82	60	76,2	5,1	5,01	0,09	6,858
83	60	69,6	4,97	4,89	0,08	5,568
84	60	62,9	4,88	4,82	0,06	3,774
85	60	58,7	4,81	4,77	0,04	2,348

Para una mejor representación de los datos obtenidos, se realiza una gráfica de Velocidad (PWM) contra Consumo, la cual se muestra en la Figura 5.1.

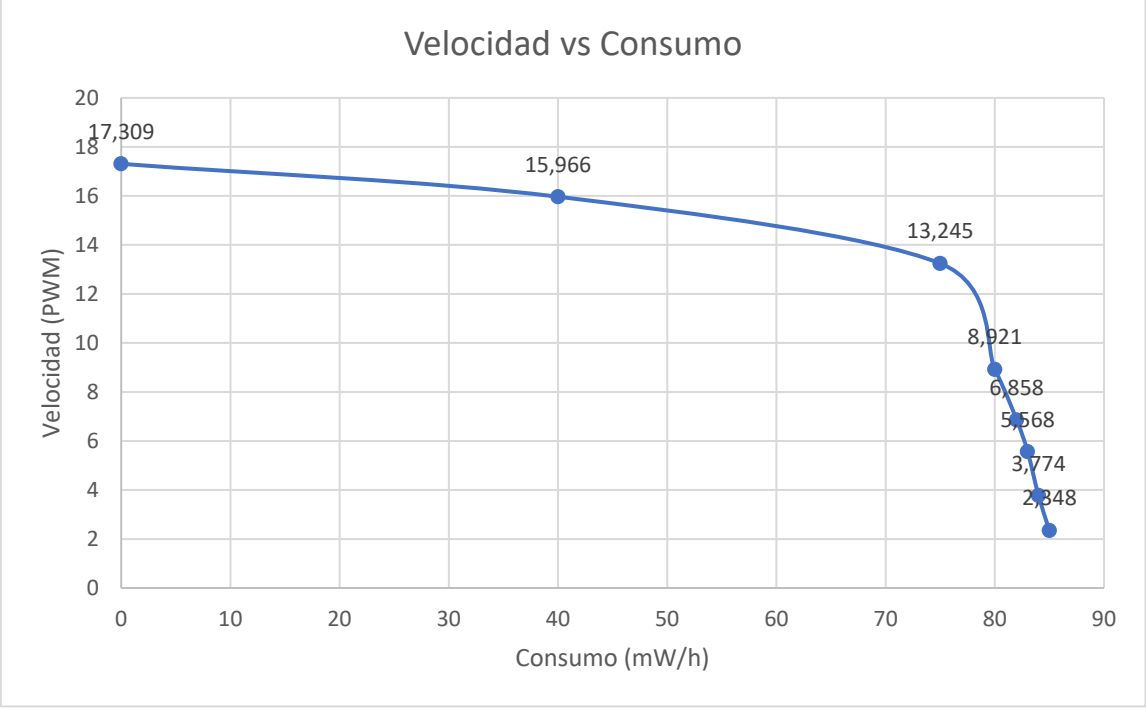

Figura 5.1 Gráfica de velocidad vs consumo de los servomotores

Con los datos obtenidos previamente, además de considerar las velocidades en las que el funcionamiento es el más adecuado, se decide establecer como velocidades según la carga de la batería las siguientes:

- 40 para batería alta
- 80 para batería media
- 83 para batería baja

5.2 Pruebas sobre instrucciones de movimiento por distancia

Las pruebas de funcionamiento sobre las instrucciones de movimiento por distancia se realizan mediante mediciones de error, de manera similar a las pruebas realizadas para la selección de los codificadores. Para obtener el error en centímetros de las instrucciones lineales (hacia adelante y hacia atrás) se utiliza la ecuación 20, sabiendo que el diámetro de las ruedas es de 7.7 cm (7.5 de las ruedas + 2mm de ligas), y que la cantidad de cuentas que lee el codificador por cada vuelta son 36 (9 ranuras en precisión 4x).

$$Distancia_{vueltarueda} = 7.7 * \pi = 24.19\ cm \tag{18}$$

$$Distancia_{cuentaencoder} = \frac{10 * 23.56}{360} = 0.672\ cm \tag{19}$$

$$E_{cm} = (Cuentasencoder_{teóricas} - Cuentasencoder_{prácticas}) * 0.672 \tag{20}$$

Se realizan pruebas con diferentes valores de cuentas teóricas, de modo que se tengan distintos resultados pudiendo así comparar los errores obtenidos en cada uno. Los resultados obtenidos para movimientos lineales se muestran en la Tabla 5.2.

Tabla 5.2 Pruebas sobre instrucciones de movimiento por distancia (Movimiento lineal)

Distancia introducida (cm)	Cuentas teóricas redondeadas	Cuentas prácticas (Codificador 1)	Cuentas prácticas (Codificador 2)	Error en cuentas (cm)	Error por redondeo (cm)
5	7	7	7	0	0.296
10	15	15	15	0	0.08
15	22	22	22	0	0.216
20	30	30	30	0	0.16
25	37	37	37	0	0.136
30	45	45	45	0	0.24
35	52	52	52	0	0.056
40	60	60	60	0	0.32

Para las instrucciones de movimiento lateral, se obtiene el error en grados que representa en cada caso la cuenta del codificador. Para esto que se utiliza la ecuación 23, teniendo en cuenta que el diámetro del robot es de 13 cm y gira sobre su eje central. Los datos de los movimientos laterales se muestran en la Tabla 5.3.

$$Distancia_{vueltarobot} = 13 * \pi = 40.84\ cm \tag{21}$$

$$Distancia_{vueltarobot^{\circ}} = \frac{40.84cm}{0.672\ cm} = 60.77\ \cong 61\ Cuentas\ encoder \tag{22}$$

$$\frac{Grados}{cuenta} = \frac{360}{61} = 5.9^{\circ} \tag{23}$$

Tabla 5.3 Pruebas sobre instrucciones de movimiento por distancia (Movimiento lateral)

Distancia introducida (°)	Cuentas teóricas redondeadas	Cuentas prácticas (Codificador 1)	Cuentas prácticas (Codificador 2)	Error en cuentas (°)	Error por redondeo (°)
15	3	3	3	0	2.7
45	8	8	8	0	2.2
50	8	8	8	0	2.8
75	13	13	13	0	1.7
90	15	15	15	0	1.5
180	31	31	31	0	2.9
270	46	46	46	0	1.4
360	61	61	61	0	0.1

Un factor importante de error se debe al redondeo de datos. Para obtener máxima precisión en los resultados, los valores ingresados deben ser múltiplos de 0.672 cm en las instrucciones lineales o de 5.9° en las instrucciones laterales. Sin embargo, los usuarios que introducen la distancias a recorrer por el robot son niños, y por lo general eligen distancias enteras. Por esta razón, si por ejemplo se introduce una distancia de 15 cm, el programa transforma la distancia a recorrer a 14.784 cm (22 cuentas del codificador).

Los resultados de las pruebas verifican el funcionamiento correcto del sistema de control de posición, ya que el error arrojado es nulo en todos los casos. El único error arrojado es el correspondiente al de redondeo en los valores a recorrer, que en el caso de las instrucciones lineales es siempre menor a 0.35 cm, por lo cual se considera despreciable. En el caso de las instrucciones hacia los lados, el máximo error es de 2.95°, que, si bien no es despreciable, cumple con las condiciones establecidas.

5.3 Pruebas sobre instrucciones de movimiento por tiempo

En el caso de las instrucciones de movimiento por tiempo, lo que se pretende probar es el correcto funcionamiento del control de velocidad, por lo que las pruebas se realizan mediante la medición del tiempo de estabilización de la velocidad. Si bien no se espera que el tiempo de estabilización concuerde con el tiempo sugerido en las simulaciones, si se espera que mejore con respecto al tiempo de estabilización del sistema sin controlador (Aproximadamente 4.5 segundos). Además, cabe mencionar que el sistema de control de velocidad se diseña con el fin de regular las velocidades de ambos servomotores a un mismo valor, no con el fin de mejorar el tiempo de estabilización.

Las pruebas se realizan utilizando los tres distintos valores de velocidad a los que se debe controlar el sistema según la carga de la batería. Los resultados se muestran en la Tabla 5.4.

Tabla 5.4 Pruebas sobre instrucciones de movimiento por tiempo

Velocidad esperada (°/ms)	Tiempo de estabilización Servomotor 1 (s)	Tiempo de estabilización Servomotor 2 (s)	Diferencia en tiempo de estabilización (ms)
0.24	2.163	2.163	0
0.24	2.212	2.231	19
0.24	2.016	1.996	20
0.22	1.718	1.735	17
0.22	1.858	1.858	0
0.22	1.645	1.645	0
0.20	2.060	2.077	17
0.20	2.141	2.159	18
0.20	2.133	2.133	0

Los resultados obtenidos en las pruebas realizadas verifican el correcto funcionamiento del sistema de control de velocidad, ya que si bien existe diferencia entre los tiempos de estabilización este tiempo es despreciable. Además, es importante recalcar que en todas las ocasiones se presenta la estabilización en el ciclo siguiente (cada ciclo dura entre 15 y 20 ms en ejecución), por lo que se puede también suponer que la estabilidad se alcanza durante este intervalo de tiempo, sin que esto signifique lograr la estabilidad en algunos de los límites del mismo. Además, el sistema de control ayuda con la mejora del tiempo de estabilización, ya que pasa de tardar alrededor de 4.5 segundos a aproximadamente 2 segundos.

Capítulo 6. Conclusiones y recomendaciones

6.1 Conclusiones

El costo económico que involucra el proyecto es de aproximadamente $1.

Según las pruebas realizadas sobre los codificadores de cuadratura, el uso de la configuración 2 presenta un porcentaje de error promedio menor comparado con el uso de la configuración 1, lo cual se debe tanto a la dificultad de lectura de los sensores en discos con ranuras más pequeñas, como a la dificultad presentada en la colocación de los sensores en las posiciones específicas de la configuración 1. Además, los sensores de reflexión presentan menor porcentaje de error comparados con los sensores de transmisión en cualquiera de las configuraciones.

Debido a que el set de instrucciones contiene tanto instrucciones de distancia como de tiempo, se necesita tanto un sistema de control de posición como un sistema de control de velocidad.

El controlador de posición que mejor se ajusta al kit de robótica Krotic es el de tipo PD, debido a que no presenta sobre impulso ni produce oscilaciones en el sistema. Además, se comprueba de manera experimental que la constante que produce movimiento oscilatorio en la planta es la constante integral.

El control de velocidad que mejor se adecúa a los servomotores el de tipo PI. Además, debido a que los valores de velocidad son menores a 0, las constantes necesarias para el adecuado funcionamiento del sistema tienden a ser grandes, en este caso se obtienen Kp = 500 y Ki = 300.

La aproximación del modelo numérico por el método analítico es útil para identificar el orden del sistema, sin embargo, es de suma dificultad encontrar las constantes tales como viscosidad o inercia del sistema. Por esta razón, se recomienda el uso del método empírico si se desea obtener una función de transferencia numérica y que asemeje el funcionamiento real de la planta

Se comprueba mediante pruebas experimentales, que el consumo de la batería es dependiente de la velocidad de giro de los servomotores.

6.2 Recomendaciones

Se recomienda limitar la velocidad de los servomotores según el estado de carga de batería, con el fin de mejorar el rendimiento de la misma.

Si se deseara mejorar la precisión de los controladores, se recomienda el uso de sensores de mejor calidad. Sin embargo, con esto se incurre en un mayor costo económico.

En miras de mejorar el funcionamiento del movimiento del kit de robótica, un mejor diseño de las ruedas sería fundamental.

Bibliografía

[1] C. A. L. Calderón, «Detección y evasión de obstáculos para la plataforma robótica educativa (KROTIC),» 2015.

[2] L. Llamas, «Controlar un servo de rotación continua con Arduino,» 2016. [En línea]. Available: https://www.luisllamas.es/controlar-un-servo-de-rotacion-continua-con-arduino/..

[3] R. Uribe, «Sintonización automática de velocidad y posición para servomotores utilizando control difuso,» 2015. [En línea]. Available: http://ri.uaemex.mx/bitstream/handle/20.500.11799/49974/TESIS_MAE_ROBERTO_URIBE.pdf?sequence=1.

[4] F. M. García, «El controlador PID,» 2007. [En línea]. Available: http://www.dia.uned.es/~fmorilla/MaterialDidactico/El%20controlador%20PID.pdf. .

[5] Parallax Inc., 2011. [En línea]. Available: https://www.parallax.com/sites/default/files/downloads/900-00008-Continuous-Rotation-Servo-Documentation-v2.2.pdf.

[6] National Instruments, «Encoder Measurements: How-To Guide - National Instruments,» 2013. [En línea]. Available: http://www.ni.com/tutorial/7109/es/.

[7] M. Escorza, «Detectores ópticos,» [En línea]. Available: http://www.mescorza.com/neumatica/sensoresweb/sensores/optico1.htm..

[8] [En línea]. Available: http://www.bushytails.net/~randyg/encoder/encoderwheel.html.

[9] Universidad de Valladolid, «Técnicas de modelado anaítico,» 2010. [En línea]. Available: https://www.infor.uva.es/~miguelv/eesi/mat/05.1-Modelado.pdf .

[10] L. Hurtado, «Modelamiento teórico y modelamiento empírico de procesos,» 2006.

[11] MatWorks, «System Identification Toolbox,» 2018. [En línea]. Available: https://es.mathworks.com/help/ident/ .

[12] MatWorks, «PID Tuner,» 2018. [En línea]. Available: https://es.mathworks.com/help/control/ref/pidtuner-app.html.

Apéndices

A.1 Resultados de pruebas de configuración 1

Tabla A.1 Pruebas con configuración 1, sensor de reflexión

Disco 18 huecos, configuración 0°-183°, Sensores reflexión							
Velocidad (PWM)	Velocidad Real (cm/s)	Cuentas Prácticas		Promedio	Cuentas Teóricas	%Error	Error centímetros
0	16.836	140	138	139.25	138	0.9057971	1.525
		140	139				
20	17.08	140	140	140	140	0	0
		140	140				
40	16.714	139	139	139.25	137	1.6423358	2.745
		140	139				
80	15.982	134	133	133.25	131	1.7175573	2.745
		133	133				
86	7.808	65	64	64.5	64	0.78125	0.61
		65	64				
87	5.124	43	42	42	42	0	0
		42	41				
95	11.102	92	93	92.75	91	1.9230769	2.135
		93	93				
100	16.104	133	133	133	132	0.7575758	1.22
		133	133				
110	16.836	140	139	138.75	138	0.5434783	0.915
		138	138				
120	16.714	139	138	139	137	1.459854	2.44
		139	140				
140	16.836	138	139	138.5	138	0.3623188	0.61
		139	138				
180	16.836	138	140	138.5	138	0.3623188	1.83
		138	138				

Tabla A.2 Pruebas con configuración 1, sensor de transmisión

Rueda 18 huecos, configuración 0°-183°, Sensores transmisión							
Velocidad	Velocidad Real (cm/s)	Cuentas Prácticas		Promedio	Cuentas Teóricas	%Error	Error centímetros
0	16.958	142	141	140	139	0.7194245	1.22
		140	137				
20	17.08	140	143	141.5	140	1.0714286	1.83
		141	142				
40	16.836	139	141	139.75	138	1.2681159	2.135
		140	139				
80	15.86	134	132	132.75	130	2.1153846	3.355
		133	132				
86	7.686	67	66	65.25	63	3.5714286	2.745
		65	63				
87	5.246	43	44	43.75	43	1.744186	0.915
		44	44				
95	11.224	92	93	93.25	92	1.3586957	1.525
		94	94				
100	16.348	128	133	130.25	134	2.7985075	4.575
		131	129				
110	16.836	140	134	137	138	0.7246377	1.22
		136	138				
120	16.714	141	138	139.5	137	1.8248175	3.05
		139	140				
140	16.958	142	141	140.25	139	0.8992806	0.915
		139	139				
180	16.836	138	141	139.25	138	0.9057971	2.135
		139	139				

A.2 Resultados de pruebas de configuración 2

Tabla A.3 Pruebas con configuración 2, sensor de transmisión

Rueda 9 huecos, configuración sensores juntos, Sensores transmisión							
Velocidad	Velocidad Real (cm/s)	Cuentas Prácticas		Promedio	Cuentas Teóricas	%Error	Error centímetros
0	16.6026	67	65	66.5	67	0.746268657	0.61
		67	67				
20	16.6026	66	67	66	67	1.492537313	1.22
		66	65				
40	16.6026	65	67	66.25	67	1.119402985	0.915
		66	67				
80	15.6114	63	64	63.25	63	0.396825397	0.305
		64	62				
85	10.4076	41	42	41.75	42	0.595238095	0.305
		42	42				
95	10.6554	43	44	43.75	43	1.744186047	0.915
		44	44				
100	15.8592	64	63	63.5	64	0.78125	0.61
		64	63				
120	16.6026	67	68	67.25	67	0.373134328	0.305
		67	67				
140	16.6026	67	67	67	67	0	0
		67	67				
180	16.6026	67	67	67	67	0	0
		67	67				

Tabla A.4 Pruebas con configuración 2, sensor de reflexión

Rueda 9 huecos, configuración sensores juntos, Sensores reflexión							
Velocidad	Velocidad Real (cm/s)	Cuentas Prácticas		Promedio	Cuentas Teóricas	%Error	Error centímetros
0	16.3548	66	66	33	66	50	40.26
		64	66				
20	16.6026	67	67	33.5	67	50	40.87
		66	67				
40	16.8504	68	69	34.5	68	49.2647059	40.87
		66	69				
80	15.8592	64	65	32.25	64	49.609375	38.735
		63	64				
85	10.6554	43	43	21.5	43	50	26.23
		43	43				
95	10.9032	43	45	22.25	44	49.4318182	26.535
		42	44				
100	16.107	65	63	32	65	50.7692308	40.26
		65	65				
120	16.8504	68	68	33.75	68	50.3676471	41.785
		68	67				
140	16.8504	68	67	33.75	68	50.3676471	41.7850
		67	68				
180	16.6026	66	67	33.5	67	50	40.8700
		66	67				

A.3 Programa de ejecución de instrucciones con sistemas de control

```
String instrucciones[]={}  //Ejemplo: {"MOT_FW_DIST_900", "MOT_CW_TIME_50"}

int Elementos=sizeof(instrucciones)/sizeof(instrucciones[0]);
String pieces[4];
String input;
int counter = 0;
int lastIndex = 0;
int value;
int valueant=0;

int Vmax;/// (alto=50, medio=10, bajo=7)
float analogPin = A3;
float myZipper = 0;
float voltaje;
float Vcv1;
float Vcv2;

////////////////////////////////////////////
 #include <Servo.h>
int x;
//Para encoder 1
Servo servo1;
int encoder0PinA = 10;
int encoder0PinB = 11;
int encoder0Pos = 0;
int PinALast = LOW;
int PinBLast = LOW;
int PinA = LOW;
int PinB = LOW;
unsigned long last_t = 0;
//Para encoder 2
Servo servo2;
int encoder1PinA1 = 13;
int encoder1PinB1 = 12;
int encoder1Pos = 0;
int PinA1Last = LOW;
int PinB1Last = LOW;
int PinA1 = LOW;
```

```
int PinB1 = LOW;
int last1=0;

//Para pid  distancia
volatile float Angulo =0;
int  Miscuentas = 0;
volatile int Output;
volatile int Output1;
float Tmuestreo = 0.01;
float E0 = 0;
float Error;
float KP = 5;
float KD = 0.01;
float KI = 0;
float Derivada;
float Integral;
//Para pid 2
volatile float Angulo1 =0;
int  Miscuentas1 = 0;
volatile int Output2;
volatile int Output3;
float E2 = 0;
float Error1;
float Derivada1;
float Integral1;

//Para pid  tiempo
float Tmuestreo2 = 0.1;
volatile float Velocidad;
float KPa = 500;
float KDa = 0;
float KIa = 300;
//Para pid 2
volatile float Velocidad1;

unsigned long tv = millis();
```

```
void setup(){
  Serial.begin(9600);
  servo1.attach(8); //der
  pinMode (encoder0PinA, INPUT);
  pinMode (encoder0PinB, INPUT);
  servo2.attach(9); //izq
  pinMode (encoder1PinA1, INPUT);
  pinMode (encoder1PinB1, INPUT);

///////////////Medición voltaje y ajustes velocidad/////////////
myZipper = analogRead(analogPin);
myZipper = map(myZipper, 0, 1023, 0, 500);
voltaje = myZipper/100;

if (voltaje>=4.5){
  Vmax=7;
  Vcv1=0.20;
  Vcv2=0.20;
}
if ((voltaje<4.5) & (voltaje>=3.5)){
  Vmax=10;
  Vcv1=0.22;
  Vcv2=0.22;
}
if (voltaje < 3.5){
  Vmax=7;
  Vcv1=0.20;
  Vcv2=0.20;
}
//////////////////////////División instrucciones//////////////
for (int j = 0; j < Elementos; j++){
   input = instrucciones[j];
 //////separar valores de instruccion////
for (int i = 0; i < input.length(); i++) {
        if (input.substring(i, i+1) == "_") {
          pieces[counter] = input.substring(lastIndex, i);
          lastIndex = i + 1;
          counter++;
        }
        if (i == input.length() - 1) {
          pieces[counter] = input.substring(lastIndex, i+1);
```

```
        }
      }
 value = pieces[3].toInt();

///////////////Instrucciones por distancia////////////////
if (pieces[2]=="DIST"){
    if (pieces[1]=="FW"){
      while ((Angulo != value)| (Angulo1 != -value)){
        unsigned long t = millis();
        if ((t - last_t) >= 10) {
        Miscuentas = encoder();
        Angulo=(Miscuentas)*10;
        Miscuentas1 = encoder1();
        Angulo1=(Miscuentas1)*10;
        x = controlador (value, -value, Vmax);
        last_t = t;
        }
      }

  }if (pieces[1]=="BW"){
    while ((Angulo != -value)| (Angulo1 != value)){
        unsigned long t = millis();
        if ((t - last_t) >= 10) {
        Miscuentas = encoder();
        Angulo=(Miscuentas)*10;
        Miscuentas1 = encoder1();
        Angulo1=(Miscuentas1)*10;
        x = controlador (-value, value, Vmax);
        last_t = t;
              }
    }
  }if (pieces[1]=="CW"){
     while ((Angulo != value)| (Angulo1 != value)){
        unsigned long t = millis();
        if ((t - last_t) >= 10) {
        Miscuentas = encoder();
        Angulo=(Miscuentas)*10;
        Miscuentas1 = encoder1();
        Angulo1=(Miscuentas1)*10;
        x = controlador (value, value, Vmax);
        last_t = t;
```

```
            }
      }
   }if (pieces[1]=="CCW"){

      while ((Angulo != -value)|(Angulo1 != -value)){
        unsigned long t = millis();
        if ((t - last_t) >= 10) {
        Miscuentas = encoder();
        Angulo=(Miscuentas)*10;
        Miscuentas1 = encoder1();
        Angulo1=(Miscuentas1)*10;
        x = controlador (-value, -value, Vmax);
        last_t = t;
            }
          }
   }
   }

////////////// Instrucciones por tiempo //////////////
if (pieces[2]=="TIME"){
     if (pieces[1]=="FW"){
        while(tv <= (value+valueant-1)){
        tv = millis()/1000;
        volatile unsigned long t = millis();
        if ((t - last_t) >= 10) {
        Miscuentas = encoder();
        Angulo=(Miscuentas)*10; //Rueda 9 huecos
        Velocidad = Angulo/(t - valueant*1000);
        Miscuentas1 = encoder1();
        Angulo1=(Miscuentas1)*10; //Rueda 9 huecos
        Velocidad1 = Angulo1/(t - valueant*1000);
        x = controlador2 (Vcv1, -Vcv2, Vmax);
         last_t = t;
        }
        }
    }if (pieces[1]=="BW"){
        while(tv <= (value+valueant-1)){
        tv = millis()/1000;
        unsigned long t = millis();
        if ((t - last_t) >= 10) {
        Miscuentas = encoder();
```

```
      Angulo=(Miscuentas)*10; //Rueda 9 huecos
      Velocidad = Angulo/(t - valueant*1000 +300);
      Miscuentas1 = encoder1();
      Angulo1=(Miscuentas1)*10; //Rueda 9 huecos
      Velocidad1 = Angulo1/(t - valueant*1000+300);
      x = controlador2 (-Vcv1, +Vcv2, Vmax);
       last_t = t;
      }
      }
   }if (pieces[1]=="CW"){
      while(tv <= (value+valueant-1)){
      tv = millis()/1000;
      unsigned long t = millis();
      if ((t - last_t) >= 10) {
      Miscuentas = encoder();
      Angulo=(Miscuentas)*10; //Rueda 9 huecos
      Velocidad = Angulo/(t - valueant*1000);
      Miscuentas1 = encoder1();
      Angulo1=(Miscuentas1)*10; //Rueda 9 huecos
      Velocidad1 = Angulo1/(t - valueant*1000);
      x = controlador2 (Vcv1, Vcv2, Vmax);
       last_t = t;
          }
      }
   }if (pieces[1]=="CCW"){
       while(tv <= (value+valueant-1)){
      tv = millis()/1000;
      unsigned long t = millis();
      if ((t - last_t) >= 10) {
      Miscuentas = encoder();
      Angulo=(Miscuentas)*10; //Rueda 9 huecos
      Velocidad = Angulo/(t - valueant*1000);
      Miscuentas1 = encoder1();
      Angulo1=(Miscuentas1)*10; //Rueda 9 huecos
      Velocidad1 = Angulo1/(t - valueant*1000);
      x = controlador2 (-Vcv1, -Vcv2, Vmax);
       last_t = t;
            }
      }
     }
}
```

```
// Limpiar variables para siguiente instrucción

    input = "";
    counter = 0;
    lastIndex = 0;
    servo1.write(90);
    servo2.write(90);
    delay(500); /// delay de medio segundo entre instrucciones
    encoder0Pos =0;
    encoder1Pos =0;
    tv=0;
    valueant=valueant+value;
  }

}


void loop() {
}


int controlador2(float Vcv1, float Vcv2, int Vmax){
////////////////////////////////PID1/////////////////////////////
//Obtener error
            Error = Vcv1 - Velocidad;

            //Calcular PID
            Integral = Integral + (Error*Tmuestreo);
            Derivada = (Error - E0)/Tmuestreo2;
            Output = (KPa * Error) + (KDa * Derivada) + (KIa * Integral);
            //Saturar PD
            Output1=Output;

            if (Output > Vmax){
               Output = Vmax;
            }
             if(Output < 0){
               Output = -1*Output;
               if(Output > Vmax){
                  Output = Vmax;
```

```
    }
  }

  if (Output1 > 0){
  servo1.write(90 - Output);
  }
  if (Output1 < 0){
  servo1.write(90 + Output);
  }
  if (Output2 = 0){
  servo1.write(90);
  }
  E0 = Error;

////////////////////////////////PID2////////////////////////////
//Obtener Error1
  Error1 = Vcv2 - Velocidad1;

  //Calcular PID
  Integral1 = Integral1 + (Error1*Tmuestreo);
  Derivada1 = (Error1 - E2)/Tmuestreo2;
  Output2 = (KPa * Error1) + (KDa * Derivada1) + (KIa * Integral1);
  //Saturar PD
  Output3=Output2;

  if (Output2 > Vmax){
    Output2 = Vmax;
  }
   if(Output2 < 0){
    Output2 = -1*Output2;
    if(Output2 > Vmax){
      Output2 = Vmax;
    }
  }
  if (Output3 > 0){
  servo2.write(90 - Output2);
  }
  if (Output3 < 0){
  servo2.write(90 + Output2);
  }
  if (Output2 = 0){
```

```
            servo2.write(90);
            }
            E2 = Error1;
}

////////////////////////////////////////////////////////////////////////////////////

int controlador(int Referencia, int Referencia1, int Vmax){    ////
//////////////////////////////////PID1//////////////////////////////
//Obtener error
            Error = Referencia - Angulo;
            //Calcular PID
            Integral = Integral + (Error*Tmuestreo);
            Derivada = (Error - E0)/Tmuestreo;
            Output = (KP * Error) + (KD * Derivada) + (KI * Integral);
            //Saturar PD
            Output1=Output;
            if (Output > Vmax){
               Output = Vmax;
            }
             if(Output < 0){
               Output = -1*Output;
               if(Output > Vmax){
                  Output = Vmax;
               }
            }

             if (Output1 > 0){
             servo1.write(90 - Output);
            }
            if (Output1 < 0){
             servo1.write(90 + Output);
            }
            if (Output2 = 0){
             servo1.write(90);
            }
            E0 = Error;

//////////////////////////////////PID2//////////////////////////////
//Obtener Error1
            Error1 = Referencia1 - Angulo1;
```

```
            //Calcular PID
            Integral1 = Integral1 + (Error1*Tmuestreo);
            Derivada1 = (Error1 - E2)/Tmuestreo;
            Output2 = (KP * Error1) + (KD * Derivada1) + (KI * Integral1);
            //Saturar PD
            Output3=Output2;

            if (Output2 > Vmax){
               Output2 = Vmax;
            }
             if(Output2 < 0){
               Output2 = -1*Output2;
               if(Output2 > Vmax){
                  Output2 = Vmax;
               }
            }
            if (Output3 > 0){
             servo2.write(90 - Output2);
            }
            if (Output3 < 0){
             servo2.write(90 + Output2);
            }
            if (Output2 = 0){
             servo2.write(90);
            }
             E2 = Error1;
 return 0;
   }

//////////////////////////////////////////ENCODER 1/////////////////////////////////////////
int encoder() {
  PinA = digitalRead(encoder0PinA);
  PinB = digitalRead(encoder0PinB);


  if (PinB == LOW){
    if (PinA == LOW){
      if((PinBLast == LOW) && (PinALast == HIGH)) {
         encoder0Pos = encoder0Pos + 1;  // 0001
     }if (PinBLast == HIGH){
```

```
      if (PinALast == LOW){
         encoder0Pos = encoder0Pos - 1; // 0010
      }if (PinALast == HIGH){
         encoder0Pos = encoder0Pos + 2; // 0011
      }
     }
   }if (PinA == HIGH){
     if((PinBLast == LOW) && (PinALast == LOW)) {
       encoder0Pos = encoder0Pos - 1;  // 0100
    }if (PinBLast == HIGH){
      if (PinALast == LOW){
         encoder0Pos = encoder0Pos - 2; // 0110
      }if (PinALast == HIGH){
         encoder0Pos = encoder0Pos + 1; // 0111
      }
     }
   }
}if (PinB == HIGH){
  if (PinA == LOW){
    if (PinBLast == LOW){
      if (PinALast == LOW){
         encoder0Pos = encoder0Pos + 1; // 1000
      }if (PinALast == HIGH){
         encoder0Pos = encoder0Pos - 2; // 1001
      }
    }if((PinBLast == HIGH) && (PinALast == HIGH)) {
       encoder0Pos = encoder0Pos - 1;  // 1011
    }
  }if (PinA == HIGH){
    if (PinBLast == LOW){
      if (PinALast == LOW){
         encoder0Pos = encoder0Pos + 2; // 1100
      }if (PinALast == HIGH){
         encoder0Pos = encoder0Pos - 1; // 1101
      }
    }if((PinBLast == HIGH) && (PinALast == LOW)) {
       encoder0Pos = encoder0Pos + 1;  // 1110
    }
   }
 }
```

```
  PinALast = PinA;
  PinBLast = PinB;

   return(encoder0Pos);
}

////////////////////////////////////////ENCODER1/////////////////////////////////////////////
int encoder1() {
  PinA1 = digitalRead(encoder1PinA1);
  PinB1 = digitalRead(encoder1PinB1);


  if (PinB1 == LOW){
    if (PinA1 == LOW){
      if((PinB1Last == LOW) && (PinA1Last == HIGH)) {
        encoder1Pos = encoder1Pos + 1;  // 0001
     }if (PinB1Last == HIGH){
        if (PinA1Last == LOW){
           encoder1Pos = encoder1Pos - 1; // 0010
      }if (PinA1Last == HIGH){
           encoder1Pos = encoder1Pos + 2; // 0011
      }
     }
   }if (PinA1 == HIGH){
      if((PinB1Last == LOW) && (PinA1Last == LOW)) {
        encoder1Pos = encoder1Pos - 1;  // 0100
     }if (PinB1Last == HIGH){
        if (PinA1Last == LOW){
           encoder1Pos = encoder1Pos - 2; // 0110
      }if (PinA1Last == HIGH){
           encoder1Pos = encoder1Pos + 1; // 0111
      }
     }
   }
 }if (PinB1 == HIGH){
    if (PinA1 == LOW){
      if (PinB1Last == LOW){
        if (PinA1Last == LOW){
           encoder1Pos = encoder1Pos + 1; // 1000
      }if (PinA1Last == HIGH){
           encoder1Pos = encoder1Pos - 2; // 1001
```

```
      }
    }if((PinB1Last == HIGH) && (PinA1Last == HIGH)) {
       encoder1Pos = encoder1Pos - 1;  // 1011
    }
  }if (PinA1 == HIGH){
     if (PinB1Last == LOW){
       if (PinA1Last == LOW){
          encoder1Pos = encoder1Pos + 2; // 1100
      }if (PinA1Last == HIGH){
          encoder1Pos = encoder1Pos - 1; // 1101
      }
    }if((PinB1Last == HIGH) && (PinA1Last == LOW)) {
       encoder1Pos = encoder1Pos + 1;  // 1110
    }
   }
 }

 PinA1Last = PinA1;
 PinB1Last = PinB1;

  return(encoder1Pos);
}
```