ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"Diseño e implementación de un prototipo académico para el control de velocidad de un motor DC, basado en un algoritmo de aprendizaje por refuerzo y hardware de código abierto”

PROYECTO INTEGRADOR

Previo la obtención del Título de:

Ingeniero en Electrónica y Automatización

Presentado por:
Ulbio Byron Alejandro Sanjinés
Anthony Martyn Maisincho Jivaja

GUAYAQUIL - ECUADOR
Año: 2021

DEDICATORIA

Dedico este proyecto a mi compañera de vida Sofía, quien estuvo incondicionalmente a mi lado toda mi vida universitaria siendo uno de los pilares fundamentales en todos los aspectos de mi vida, sin ti no lo hubiera logrado.

A mi hija Antonella, que es mi fuente de inspiración, mi vida y mayor tesoro.

A mi padre Martin, que aparte de su sacrificio y trabajo para ayudarme, me inculcó esta profesión desde mi niñez y a realizar todas mis actividades a cabalidad.

Y a mi madre Yanett, quien me brindó amor y confianza diciendo que lograría esta meta.

Anthony Martyn Maisincho Jivaja

DEDICATORIA

En primera instancia, agradecer a Dios por darme la oportunidad de culminar mis estudios de grado en conjunto con mi familia.

Mis padres fueron mi mayor motivación, así como mi hermana quiénes me daban el apoyo suficiente para que pueda desarrollar mi carrera.

A mis profesores, que supieron darme las bases necesarias para culminar con éxito este proyecto de grado.

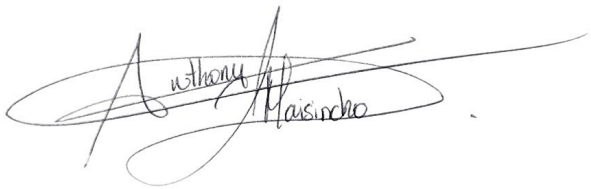
Y finalmente a mis amigos, los cuales se han preocupado por mí y en conjunto motivarme e indicarme que todo es posible, siempre y cuando se realice con perseverancia y esfuerzo.

Ulbio Byron Alejandro Sanjines

DECLARACIÓN EXPRESA

“Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; Ulbio y Anthony damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Firmado electrónicamente por:
ULBIO BYRON ALEJANDRO SANJINES



Ulbio Byron Alejandro Sanjines

Anthony Martyn Maisincho Jivaja

EVALUADORES

Firmado electrónicamente por:
DENNYS DICK CORTEZ ALVAREZ

MSc. Dennys Dick Cortez Alvarez

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
VICTOR MANUEL ASANZA ARMIJOS

MSc. Victor Manuel Asanza Armijos

PROFESOR TUTOR

RESUMEN

Los procesos industriales automatizados requieren de un controlador para obtener una señal de salida similar a la referencia indicada por el usuario. Existen controladores como el PID, el cual es eficiente si el sistema no cambia sus condiciones iniciales, caso contrario se debe ajustar sus parámetros nuevamente, sacrificando tiempos de producción. Por esto, se desarrolló un tablero de control que contiene un algoritmo de aprendizaje por refuerzo que ajusta las constantes del controlador PID en base a las experiencias obtenidas al momento de interactuar con el entorno.

El tablero de control fue desarrollado con dispositivos de hardware de código abierto, con interfaz comprendida de pulsadores, selector, pantalla HMI, y luces pilotos. Se implementaron pruebas de funcionamiento para verificar el alcance y escalabilidad del tablero. Mientras que, el algoritmo de Inteligencia Artificial se ejecutó mediante el software MATLAB.

En base al actor-crítico del Agente RL se pudo obtener las constantes de un controlador PI adaptivo que, tuvo mejor rendimiento que el controlador PI clásico. Además, que llega a la referencia con el mínimo esfuerzo del controlador determinado por una función de costos.

El controlador PI implementado por el algoritmo, permite generar las constantes adecuadas para mantener una velocidad del motor DC sin la necesidad de conocer el sistema. El tablero propuesto también permite reforzar los conocimientos teóricos de automatización y sistemas embebidos, los cuales con distintos lenguajes de programación se ajustan a las capacidades del usuario, resaltando sus características y precio competitivo con otros productos similares.

Palabras Clave: Aprendizaje por refuerzo, Inteligencia Artificial, Tablero de control, Controlador PI adaptivo, función de costos, actor-crítico, política.

ABSTRACT

Automated industrial processes require a controller to obtain an output signal like the reference indicated by the user. There are controllers such as PID, which is efficient if the system does not change its initial conditions, otherwise its parameters must be adjusted again, sacrificing production times. For this reason, a control board was developed that contains a reinforcement learning algorithm that adjusts the PID controller constants based on the experiences obtained when interacting with the environment.

The control board was developed with open-source hardware devices, with interface comprised of push buttons, selector, HMI display, and pilot lights. Functional tests were implemented to verify the scope and scalability of the board. While the Artificial Intelligence algorithm was executed using MATLAB software.

Based on the actor-critic of the RL Agent, it was possible to obtain the constants of an adaptive PI controller which, performed better than the classical PI controller. Moreover, that it reaches the benchmark with the minimum controller effort determined by a cost function.

The PI controller implemented by the algorithm, allows to generate the appropriate constants to maintain a DC motor speed without the need to know the system. The proposed board also allows to reinforce the theoretical knowledge of automation and embedded systems, which with different programming languages are adjusted to the user's capabilities, highlighting its features and competitive price with other similar products.

Keywords: Reinforcement learning, artificial intelligence, control panel, adaptive PI controller, cost function, actor-critic, policy.

ÍNDICE GENERAL

ABREVIATURAS

PID	Proporcional Integral Derivativo
RL	Reinforcement Learning
LQG	Linear Quadratic Gaussian
RELU	Unidad rectificadora lineal
DC	Corriente continua
ANN	Artificial Neural Network
DDPG	Deep Deterministic Policy Gradient
PLC	Controlador Lógico Programable
TD3	Twin Delayed
ML	Machine Learning
IA	Inteligencia artificial
R	Recompensa
J	Función de costos
HMI	Interfaz Hombre Máquina

SIMBOLOGÍA

VDC	Voltaje de corriente continua
RPM	Revoluciones por minuto
N	Newton
m	Metro
s	Segundos
A	Amperios
H	Inductancia
Kg	Kilogramos

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE PLANOS

INDICE DE FOTOS

CAPÍTULO 1

1. INTRODUCCIÓN

El presente trabajo corresponde al diseño de un tablero de control basado en hardware de código abierto, el cual será utilizado para controlar un motor DC con fines académicos. En la actualidad es importante aplicar un control para los distintos sistemas que la industria requiera, para esto se realiza una evaluación previa de las condiciones iniciales, ejerciendo así un estudio y logrando estimar un controlador del tipo PID. Esto resulta favorable siempre y cuando el sistema no haya cambiado ninguna de sus características originales, caso contrario se debe dimensionar un nuevo controlador para la planta.

Las técnicas de control clásico resultan ineficientes para este ámbito, por ese motivo se pretende utilizar un controlador donde su sintonización está basado en el aprendizaje por refuerzo, de esta manera, dicho controlador se adaptará al entorno y se ajustará ante cualquier cambio que se dé en el sistema. Para esto, el agente que en este caso es el motor DC, deberá continuamente optimizar la política con la que ejecuta una acción en base a ciertas recompensas obtenidas en función de buenas y malas acciones, el objetivo es obtener la máxima recompensa.

Además, el tablero cuenta con distintos componentes de control como son: pulsadores, relés, pantalla táctil para una interfaz hombre-máquina (HMI), etc., los mismo que son controlados mediante un microcontrolador que es el Arduino Nano y un microprocesador que es la Raspberry Pi 4. Mediante la Raspberry Pi 4, el tablero tendrá acceso a la red local que, mediante una conexión de internet virtual privado (VPN), los estudiantes podrán acceder de forma remota para realizar las distintas prácticas que se dictan en clase, con la ventaja de interactuar con una planta real,

resolviendo el problema del acceso restringido a los laboratorios por la pandemia actual.

Finalmente, se analizará los resultados obtenidos entre las técnicas de control clásico y la de aprendizaje por refuerzo, donde se mostrarán las ventajas y desventajas de cada uno cuando son aplicados en el ámbito laboral. Además, de tener un tablero de control finalizado para las distintas prácticas de los estudiantes de automatización, de esta forma aplican los conocimientos utilizando sistemas embebidos, herramientas actuales de la automatización que está en auge por la accesibilidad, adaptabilidad y reducción de costos.

1.1 Descripción del problema

En un proceso industrial se deben considerar sus variables de control, para que el producto final tenga altos estándares de calidad. Una técnica de control clásica es aplicar un controlador proporcional, derivativo e integral (PID), donde se estima que alrededor del 95% de los sistemas en lazo cerrado adoptan este tipo de familia, fundamentalmente PI (Cantarero, 2015). Una vez sintonizado sus tres términos, es importante que las condiciones iniciales de la planta industrial no cambien, esto por motivo a que resultaría ineficientes ya que la característica dinámica de la salida podría afectar por dicha modificación.

Para encontrar la dinámica del sistema es posible realizarlo por métodos matemáticos, donde considerando todas las ecuaciones que afectan a la planta, se obtiene una función de transferencia en el dominio de la frecuencia s y con el uso del software Matlab obtener las constantes del controlador. Otra alternativa es realizar una prueba en lazo abierto al sistema, adquiriendo sus datos y preprocesarlos para así con la ayuda de la herramienta Identificación de Sistemas por parte de MATLAB, encontrar una representación matemática y obtener el controlador de la familia PID.

El primer proceso descrito resulta poco práctico, ya que es posible equivocarse por todas las condiciones que se toman para que la planta se pueda modelar. La segunda opción, se debe realizar una prueba donde la planta debe someterse a niveles altos de producción, por eso se lleva a cabo con el consentimiento de los operarios y en horarios establecidos para considerar mínimas pérdidas de producción por las pruebas realizadas (Rosero, 2011).

Además, en tiempos de pandemia la educación ha sido todo un reto para las unidades educativas que tienen laboratorios para que sus estudiantes refuercen la teoría con la práctica. Los alumnos no pueden acceder al campus de la universidad por restricciones y normas a nivel mundial. Esto, debido al Centro de Operaciones de Emergencia (COE) nacional para que el virus no se esparza a la ciudadanía (Cedeño, 2021). Creando así barreras para que los estudiantes se familiaricen con los dispositivos usados en su nivel de carrera.

La misión de toda universidad es que cada uno de sus graduados, cumpla con los perfiles más altos en calidad laboral. Por esto, se implementan laboratorios de forma virtual para que sea posible simular cada una de sus prácticas en las materias dictadas. En muchos de los casos, es posible no tomar en cuentas ciertas condiciones reales por motivo de

simulación, trayendo consigo una mala práctica de aprendizaje ya que en el nivel industrial no utiliza modelos lineales ni mucho menos ideales para su implementación.

1.2 Justificación del problema

Con el objetivo de automatizar el proceso de sintonización de un control proporcional, integral, derivativo (PID) para cualquier tipo de planta y que esta siga obteniendo el mismo rendimiento ante las perturbaciones del sistema, se propone en este proyecto de tesis, un algoritmo que tome sus propias decisiones y pueda solucionar esta problemática. El desarrollo y aplicación de un algoritmo basado en el aprendizaje por refuerzo para la obtención de las contantes de un controlador PID, permitirá estudiar, analizar, evaluar y comprender este método de control moderno que está siendo ampliamente utilizado actualmente gracias a los avances en los estudios del aprendizaje profundo (Deep Learning).

Además que, ante el desafío de la modalidad virtual para las carreras de ingeniería debido a la pandemia y la poca interacción del estudiante con equipos y plantas reales provocando que los profesionales egresen con insuficiente experiencia y prácticas, el proyecto también apunta a ser presentado como un prototipo académico basado en hardware de código abierto, donde el estudiante podrá interactuar con plantas reales, conectándose al tablero didáctico, por medio de la VPN que tiene a disposición la universidad. Es decir, este proyecto pretende aplicar una práctica de control basado en aprendizaje por refuerzo en un tablero entrenador constituido de hardware de código abierto.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar e implementar un tablero de control académico constituido de hardware de código abierto para la implementación de un control PID adaptivo de velocidad de un motor DC basado en un algoritmo de aprendizaje por refuerzo.

1.3.2 Objetivos Específicos

1. Diseñar un tablero de control basado en código abierto con acceso remoto.
2. Establecer un algoritmo de control basado en aprendizaje por refuerzo.
3. Comparar los resultados entre un control clásico con el control de aprendizaje por refuerzo.
4. Fortalecer las habilidades prácticas de los estudiantes en sus simulaciones mediante los dispositivos de open hardware.

1.4 Marco teórico

1.4.1 Sistemas de control

A lo largo de la historia, el ser humano intenta controlar todos los sistemas que están a su alcance y para ello, debe tener claro sobre los componentes que lo involucre y así tener un beneficio que se ajuste a sus necesidades. Las evidencias de las primeras máquinas automáticas se remontan por el año 300 a.C. en Babilonia, donde se desarrolló el primer dispositivo de reloj basado en agua por Ktesibios o también denominado como *clepsydra* (Miranda, 2016).

La metodología aplicada fue mediante un nivel de agua en un tubo que llenaba un depósito a presión constante, mediante un flotador controlaban la apertura de una válvula según el nivel deseado, el cual se abría cuando el tanque registraba nivel bajo y se cerraba cuando llegaba a su meta. Su función automática fue implementada por el gran filósofo Platón (428-347 a.C.), ya que presentaba la dificultad que sus estudiantes no llegaban a tiempo a sus reuniones, entonces por encima del flotador colocó unas bolas, tal que al llenarse por la noche y establecer el nivel adecuado, se expulsaban del depósito cayendo sobre un plato de cobre, el mismo que serviría para levantar a las personas por medio del ruido que generaba el impacto de la caída.

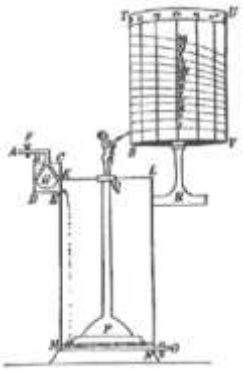


Figura 1.1 Reloj de agua de Ktesibios (Miranda, 2016)

Luego de largos años, se estableció la teoría de control en base a las ecuaciones diferenciales, que describían el comportamiento de los sistemas en el tiempo continuo, las mismas que resultó de procedimientos exhaustivos para encontrar un formato adecuado. Con el uso de la transformada de Laplace se permitió obtener soluciones exactas para los modelos por ecuaciones o sistemas de ecuaciones lineales, que implicaban problemas de valor inicial (Matilde Pilar

Legua Fernández, Luis Manuel Sánchez Ruiz., 2017), los cuales dieron las bases sólidas para realizar los sistemas de control automático.

El término sistemas podría tener distintos significados según el contexto que se interprete, en la ingeniería de control se indica como una entidad que produce una transformación de señales, donde las entradas son manipuladas por el usuario mientras que su salida es observable. Las perturbaciones pueden ser externas o internas que no son posibles manipular, tienen carácter aleatorio como es el ruido o determinista como son las interferencias (Miranda, 2016).

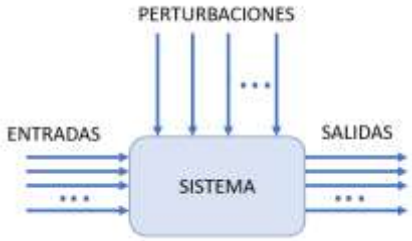


Figura 1.2 Flujo de las señales en un sistema. Autor.

Existen varios tipos de sistemas de control, los cuales depende cuantas entradas y salidas contempla, donde se clasifican los siguientes:

- SISO: Single Input-Single Output; Una entrada-Una salida.
- SIMO: Single Input-Multiple Output; Una entrada-Varias salidas.
- MISO: Multiple Input-Single Output; Varias entradas-Una salida.
- MIMO: Multiple Input-Multiple Output; Varias entradas-Varias salidas.

Para la respuesta deseada, los sistemas de control contienen diferentes componentes interconectados los cuales son previamente dimensionados en base al análisis proporcionado por la teoría de los sistemas lineales, que supone una relación entre la causa y efecto en sus elementos. La Figura 1.2 hace referencia a un bloque denominado sistema, el cual contiene un proceso especifico que, al alterar una entrada, tendríamos una salida equivalente a la acción causa-efecto antes explicada.

Se tienen dos tipos de lazos de control: abierto y cerrado. Los sistemas en lazo abierto no contemplan una retroalimentación, para que la supervisión se lleve a cabo utiliza un dispositivo de actuación el cual no emplea la observación de lo que se tiene en su salida. Los sistemas en lazo abierto están inmersos principalmente en las industrias donde requieren encontrar el funcionamiento a plena carga de

los proyectos, tal es el caso cuando se aplica un cierto valor de la entrada y obtenemos un cierto valor en la salida.

Figura 1.3 Sistema de control en lazo abierto. Autor.

Los sistemas de control en lazo cerrado requieren de un sensor que este controlando constantemente a la salida del proceso, esto lo compara con la respuesta deseada mediante una retroalimentación negativa, que estará proveyendo al controlador un cierto error para que actúe en el proceso. En estos sistemas, es importante destacar que tanto las salidas como entradas se denominan variables de proceso y están dado por unidades de medidas como son: voltaje, corriente, metros, etc., las cuales dependerá del control aplicado (Dorf, R.C; Bishop, R. H., 2005).

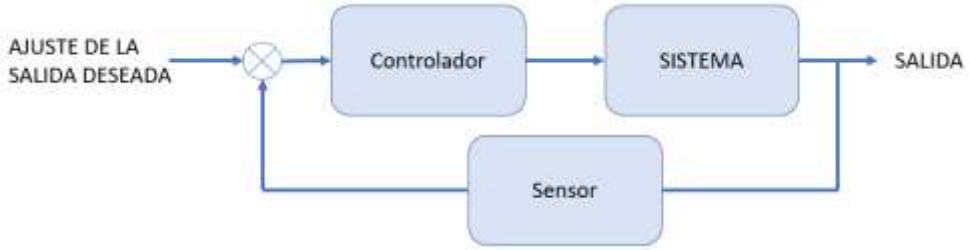


Figura 1.4 Sistema de control en lazo cerrado. Autor.

Los instrumentos de control industrial están diseñados dependiendo su área de aplicación como es: química, energética, metalúrgica, petroquímica, etc., los cuales poseen características propias y diversos elementos utilizados como: registradores, controladores, indicadores, transmisores y válvulas de control (Sole, 2005). Existen distintos controladores para los procesos, aquí destacamos de la familia PID.

1.4.2 Controladores PID

A nivel industrial, es importante controlar las variables que se encuentran en el proceso, debido a su importante relevancia con el producto final ya que, al no ajustar los parámetros correctos, es posible alterar la calidad. Para esto se utilizan distintas técnicas de control, en esta sección se basará en los controladores de la familia PID. Este tipo de controladores tiene tres parámetros a sintonizar que son: proporciona, integral y derivativo.

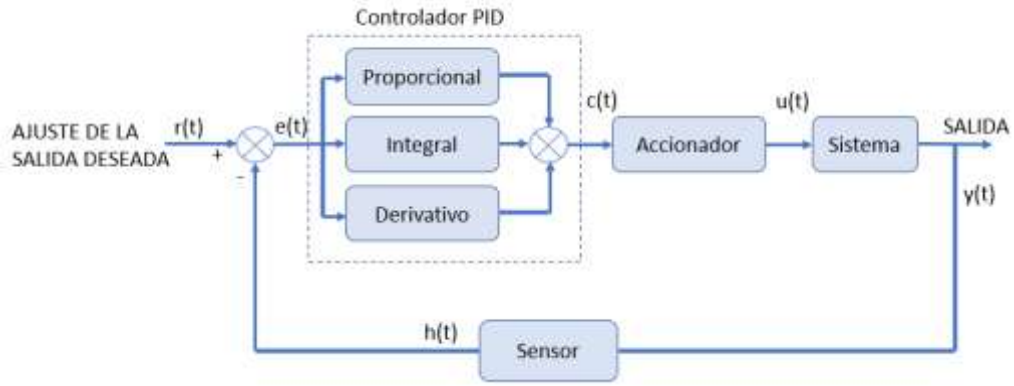


Figura 1.5 Sistema de control en lazo cerrado con control PID. Autor.

En la Figura 1.5 se presentan todas las señales que actúan en el lazo de control cerrado, el cual tiene un controlador PID, accionador, sistema y para mantener una constante supervisión en la salida, se cuenta con un sensor que está como retroalimentación negativa quién provee el error mediante el punto sumo del parámetro seteado. Las señales en cuestión se describen como:

- r(t): señal de referencia, la ajusta el usuario y debe tener la misma dimensión que la salida.
- e(t): señal de error, la misma que surge por la diferente entre la salida y su ajuste deseado.
- c(t): señal del controlador, la misma que actúa en función de cuanto es el error que presenta.
- u(t): señal del accionador, encargada de acondicionar a la entrada del sistema para que ajuste la salida.
- h(t): señal del sensor, prevista por la señal de salida en el proceso.
- y(t): señal de salida del proceso, es la variable de control que se pretende controlar.

El término ajustable denominado Proporcional, tiene una acción de control directo con la señal de error, ya que es multiplicado y se denominada por la constante **Kp**. Esta acción de control permite disminuir el error que contempla el sistema ya que, al ser grande, la constante es mayor y tiende a minimizar el error en estado estable (Ogata, 2010). Los principales efectos resultantes por este término son:

- Aumentar la velocidad de respuesta en el sistema.
- Disminuir el error en estado estable del proceso.
- Aumenta la inestabilidad para la etapa transitoria.

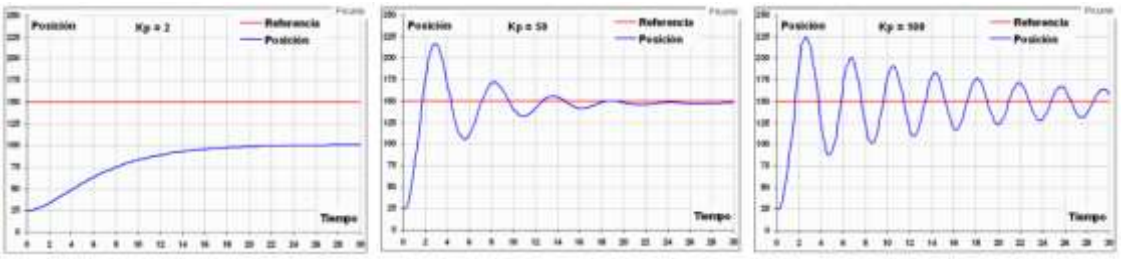


Figura 1.6 Respuesta del control modificando la constante Kp en el sistema. (Ogata, 2010)

Como se observa en la Figura 1.6, se evidencia las características antes mencionadas, los dos primeros efectos resultan positivos porque se desea que la dinámica transitoria sea rápida y se estabilice en el punto seteado, mientras que la inestabilidad resulta mayor al indicar una constante mayor, tal es el caso como la imagen de la izquierda, que resulta ser inestable en el tiempo establecido.

La acción por la componente derivativa se la denomina como una constante **Kd**, lo que resulta es derivar a la señal de error, estableciendo así su principal relación con la velocidad de la respuesta en el sistema. Los efectos resultantes al modificar la constante Kd son:

- Se aumenta la estabilidad del sistema controlado.
- Se disminuye paulatinamente la velocidad en el sistema.
- El error en estado estable no se modifica.

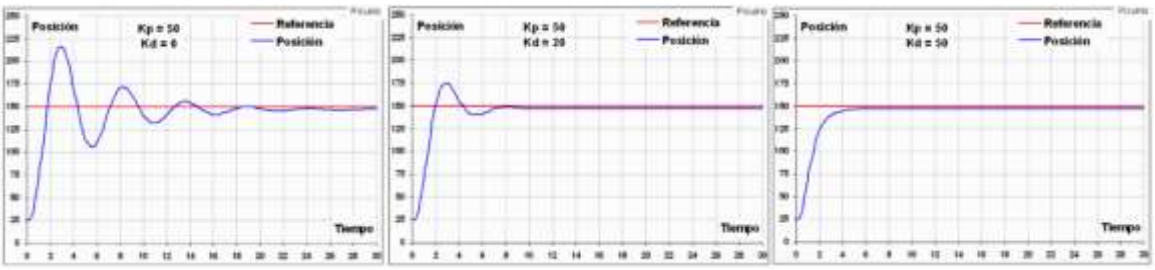


Figura 1.7 Respuesta del control, mantenimiento Kp y modificando Kd. (Ogata, 2010)

Como se muestra en la Figura 1.7, al variar la constante derivativa, la respuesta tiende hacer mucho más rápida en estabilizar, esto se logra al disminuir la velocidad en el sistema. Por lo que, se logra la estabilización disminuyendo las oscilaciones en la etapa transitoria y manteniendo el error en el sistema, es por esto que se utilizó las constante proporcional y derivativa, por ende, resulta ser un controlador del tipo PD.

La señal de error al integrarla resulta ser la suma o acumulación de esta por la facultad que se presenta la integral, esta componente se denomina **Ki**. Se reduce el error en el estado estacionario debido a que, los errores que presente en el sistema van sumándose a tal punto que la integral sea cada vez mayor, pero mientras sea este valor mucho más tiende a la inestabilidad en el sistema debido a que se añade inercia al sistema. Los efectos que contempla esta acción de control son:

- Disminuye el error en estado estacionario.
- Se aumenta la inestabilidad en el sistema.
- Aumenta levemente la velocidad en el sistema.

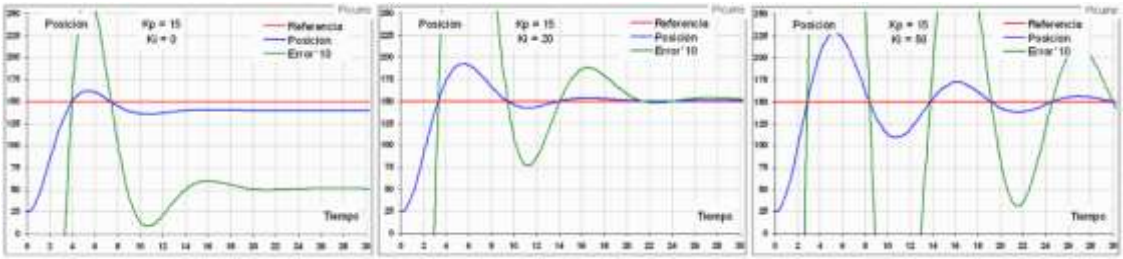


Figura 1.8 Respuesta del sistema, manteniendo Kp y variando Ki. (Ogata, 2010)

En la figura 1.8, es posible apreciar distintas señales en especial la del error de color verde, esta se ve aumentando cuando la constante incremente debido a la inestabilidad que se produce en valores altos, mientras que su velocidad es alterada para ser una respuesta mucho más rápida, el error en la posición se acerca mucho a la referencia indicando la disminución en el estado estacionario (Ogata, 2010).

Sintonizando apropiadamente cada constante que involucra el controlador de la familia PID (proporcional, integral y derivativo), es posible tener un apropiado resultado con la salida deseada en el proceso. Para esto, es importante tener las siguientes consideraciones en el ajuste manual de las mismas:

1. Acción Proporcional: Se debe aumentar poco a poco hasta lograr disminuir el error y aumentar la velocidad de respuesta en el proceso. En caso de que el sistema se vuelva inestable, se debe contrarrestar con la constante Kd.
2. Acción Derivativa: Conseguir la estabilidad del sistema se logra aumentando progresivamente esta constante, hasta llegar a la deseada.

3. Acción Integral: Cuando el error se encuentra por arriba del deseado, se debe aumentar la constante Ki para llegar al error y rapidez deseada. En caso de que el sistema se vuelva inestable, se debe modificar la constante Kd.

1.4.3 Inteligencia Artificial

Hoy en día existen muchas definiciones de la inteligencia artificial (IA) debido a que se trata de un tema complejo. Una simplificación de la definición sería: "Habilidad de los ordenadores para hacer actividades que normalmente requieren de inteligencia humana", tal como lo explicó Lasse Rouhianen (Rouhiainen, 2018). Otra definición acertada sería:

> La inteligencia artificial es la capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano. Sim embargo, a diferencia de las personas, los dispositivos basados en IA no necesitan descansar y pueden analizar grandes volúmenes de información a la vez. Asimismo, la proporción de errores es significativamente menor en las máquinas que realizan las mismas tareas que sus contrapartes humanas (Rouhiainen, 2018).

La tecnología basada en IA está beneficiando a los humanos en muchos ámbitos de la vida, uno de ellos es en el entorno industrial, donde se aprecia hoy en día sistemas de reconocimiento de imágenes en producción para clasificación y etiquetados, mejoras del desempeño de la estrategia algorítmica comercial, procesamiento y toma de decisiones en la medicina basado en un conjunto de datos, detección de objetos y muchas otras aplicaciones como la que se trata de resolver en este proyecto.

1.4.4 Aprendizaje automático

Es un subcampo de la IA, que proporciona a las computadoras la capacidad de aprender, sin ser programadas explícitamente (TechTarget, 2017).

El proceso de *machine learning* (ML) cuenta con dos etapas, una de ellas es el entrenamiento, la cual implica que el sistema trata de aprender comportamientos o patrones que se ajusten a los datos del conjunto de entrenamiento. La segunda etapa se trata de la predicción, la cual se refiere a la salida de un algoritmo luego

de que haya sido entrenado con un conjunto de datos histórico y aplicado a nuevos datos para pronosticar la probabilidad de un resultado particular (DataRobot, 2021).

Finalmente, existen tres tipos de algoritmos principales de aprendizaje automático o ML, uno de ellos es el *aprendizaje supervisado,* el cual en base a un conjunto de datos de entrada se obtiene un resultado esperado por el usuario; luego el *aprendizaje no supervisado,* que contiene un conjunto de datos de entrada pero que no se le especifica un objetivo o una tarea al algoritmo. Por último, se tiene el *aprendizaje por refuerzo,* el cual se trata de un algoritmo que aprende observando e interactuando con el entorno, donde sus decisiones son recompensadas si han sido las correctas caso contrario, recibirá penalizaciones.

Figura 1.9 Clasificación del aprendizaje automático. Autor.

1.4.5 Aprendizaje por refuerzo

El aprendizaje por refuerzo consiste en aprender que hacer y como asignar acciones a diferentes situaciones con el objetivo de maximizar una recompensa numérica (Barto, 2018).

Existen casos donde las acciones tomadas pueden afectar no solo a la recompensa inmediata, sino también, a los siguientes estados y las recompensas posteriores, por lo que, su metodología se basa en el ensayo, error y recompensa retardada.

En esta categoría no hay un instructor que indique al sistema qué hacer, sino, una simple señal de recompensa, la retroalimentación que se obtiene del entorno no

es instantáneo, sino que está retrasado y las acciones del sistema afectan a la información recibida después (Hernanz, 2019).

En este método tenemos dos elementos principales: el *agente* y el *entorno,* finalmente, se puede resumir que el objetivo es que el agente interactúe siempre con el entorno para que se pueda llegar a un objetivo concreto.

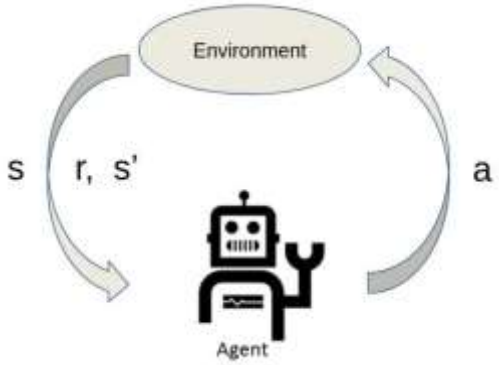


Figura 1.10 Aprendizaje por refuerzo. (Pro, 2020)

Existen otros elementos que intervienen en el aprendizaje por refuerzo: la política, función de valor y la recompensa.

El aprendizaje por refuerzo tiene dos enfoques principales, uno es el enfoque pasivo, estos algoritmos tienen una política fija, es decir, no cambia. Otro es el enfoque activo o conocido como *Q-Learning,* el cual calcula la política mediante la prueba de acciones, esto involucra un aprendizaje lento debido a que necesita la exploración (Latorre, 2019).

1.4.6 Política

La política determinará qué acciones tomará el agente en función del estado en que se encuentre. El objetivo de la política es aumentar la recompensa por cada decisión tomada. Cuaya explica lo siguiente:

> Existen dos tipos de políticas: las estacionarias y las no estacionarias. Una política estacionaria $a = \pi(s)$ especifica directamente, para cada estado, la acción a realizar independientemente del momento o tiempo en que se encuentra el proceso. Este tipo de políticas se utiliza principalmente con procesos de horizonte-infinito, puesto que, al no existir un límite de pasos, no existe ningún motivo para que el agente cambie su estrategia al elegir las acciones. Una política no estacionaria $a = \pi_t(s)$, asigna una acción u otra al mismo estado en función del momento en que se encuentra el

sistema. Este tipo de política se utiliza en procesos de horizonte finito, puesto que es conveniente modificar la estrategia a seguir en función del número de pasos que quedan para finalizar el proceso (Simbro, 2007).

Para que el agente encuentre la política óptima, prueba nuevas acciones hasta aprender, a esto se lo conoce como concepto de exploración.

1.4.7 Recompensa

La recompensa (R) define la meta, cada vez que el agente realiza una acción el entorno envía una señal de recompensa, entonces, el objetivo del agente será maximizar la recompensa obtenida al final del episodio. La recompensa definirá si la acción realiza por el agente es buena o mala, pero para evaluar las recompensas se necesita de otro elemento importante, denominado función de valor.

1.4.8 Función de valor

La función de valor permite cuantificar la recompensa que obtendrá el agente al final desde cada estado. Esta función de valor V(s) produce una estimación de la recompensa que obtendrá el agente hasta el final del episodio, empezando desde el estado *s.* Si se consigue estimar este valor de forma correcta, podremos decidir ejecutar las acciones que nos lleve al estado *s+1* con la mejor recompensa (Ausin, 2020).

1.4.9 Agente

El agente de aprendizaje por refuerzo es el que recibe observaciones y una recompensa del entorno. La función de este agente es realizar una acción mediante su política a partir de las observaciones y la recompensa obtenida, la acción escogida es enviada al entorno y va a alterar el estado de este. El agente durante el entrenamiento va a actualizar continuamente los parámetros de su política en función de las recompensas, hasta lograr la política óptima (MathWorks, 2020).

Existen varios tipos de agentes con diferentes algoritmos, los más populares o comunes son, SARSA, DQN, DDPG y PPO.

1.4.10 Redes Neuronales

Para crear agentes es necesario crear redes neuronales que representarán al actor y al crítico.

Las redes neuronales artificiales (ANN) son modelos computacionales que tratan de replicar, de manera simplificada, el complejo funcionamiento del cerebro humano. Su capacidad de aprendizaje se basa en la repetición de ensayos o episodios (Aristizábal, 2006).

Una red neuronal profunda está formada por múltiples capas. Cada capa recibe la entrada de la capa anterior, realiza una operación en la entrada y pasa la salida a la siguiente capa (Matlab Academy, 2021).

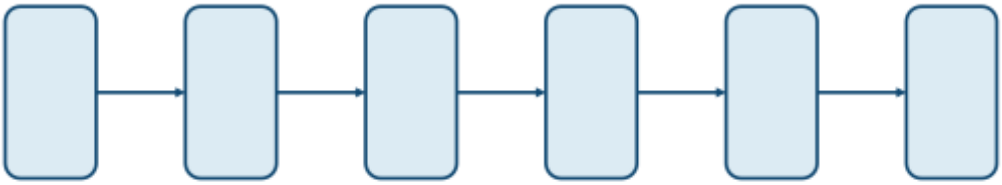


Figura 1.11 Capas de red neuronal profunda. (Matlab Academy, 2021)

Para las redes del actor o crítico, las capas internas u ocultas son capas totalmente conectadas o capas de función de activación.

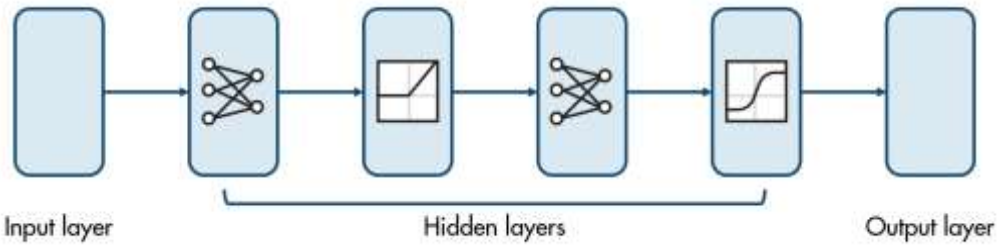


Figura 1.12 Capas internas o ocultas en una red neuronal. (Matlab Academy, 2021)

Una capa totalmente conectada consiste en un numero de neuronas, donde cada neurona representa una operación matemática.

Se multiplica la entrada con una constante denominada peso, y la suma a otra constante denominada sesgo, el resultado de esta operación pasará a la siguiente capa (Matlab Academy, 2021).

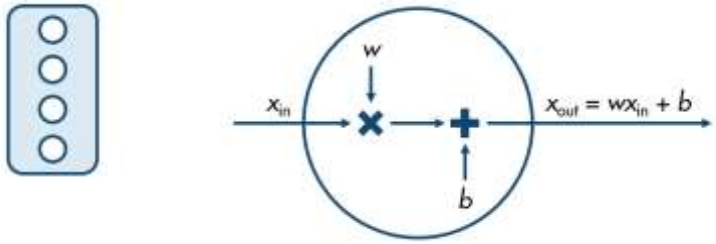


Figura 1.13 Representación matemática de una neurona de una capa totalmente conectada. (Matlab Academy, 2021)

Dentro de la capa totalmente conectada, cada neurona toma como entrada el resultado de la operación ejecutada por la neurona anterior y pasa la salida a cada neurona de la siguiente etapa (Matlab Academy, 2021).

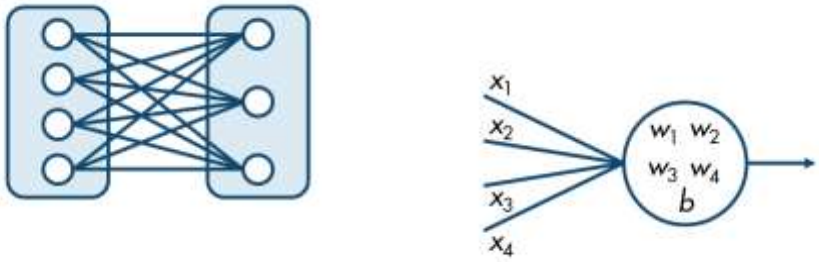


Figura 1.14 Operaciones entre neuronas de una capa totalmente conectada. (Matlab Academy, 2021)

Dentro de las capas totalmente conectadas se puede emplear capas de función de activación, las cuales introducen no linealidad al resultado anterior.

Las funciones de activación comúnmente usados son unidad de rectificador linear (RELU) o tangente hiperbólico (TANH), donde la RELU mantiene en su salida el mismo valor de la entrada siempre y cuando esta sea positiva, caso contrario, la deja en cero, y en el caso de TANH mapea el valor de entrada entre un rango de [-1,1] (Matlab Academy, 2021).

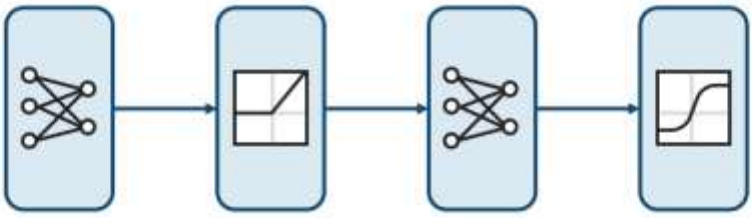


Figura 1.15 Representación de una red neuronal. (Matlab Academy, 2021)

1.4.11 Criterio Control Lineal Cuadrático Gausiano

El criterio lineal cuadrático gaussiano (LQG: *Linear Quadratic Gaussian*) es una técnica en control óptimo para equilibrar entre la regulación de una variable de proceso y el esfuerzo que realiza el controlador para lograr su objetivo de error aproximadamente cero. Dicha técnica, se basa de la representación de espacios de estado en el sistema los cuales son utilizados para el diseño de reguladores dinámicos óptimos y su acción integral que permite regular el punto de referencia (MathWorks, 2022).

Su análisis se resalta en los sistemas lineales que contienen una componente de perturbación impulsados por un ruido blanco gaussiano aditivo (Athans, 1971), los cuales no contienen información completa, sin embargo, son evaluados sus costos mediante el control de sujetos cuadráticos. En base a lo anterior, es posible el diseño de controladores LQG que permiten regular sus señales de entradas para un control óptimo en los sistemas no lineales perturbados.

Un controlador LQG es la composición de un estimador lineal cuadrático (LQE: *Linear Quadratic Estimate*) y un regulador lineal cuadrático (LQR: *Linear Quadratic Regulate*). El LQE consiste en un filtro Kalman, que es base para conocer el estado oculto entre los sistemas dinámicos lineales el cual es sometido a un ruido blanco aditivo (Kalman, 1960).

El principal objetivo para el criterio LQG es encontrar una matriz K que permita minimizar el índice del rendimiento J, dada en la función de costos representada en la ecuación (1), donde Q y R son las matrices de pesos simétricos definidas en base a la investigación en función del grado de penalización que son definidos para los estados y las entradas en el sistema, donde minimizan la energía utilizada por el controlador y en cada estado que es representado.

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{1}$$

Donde:

- J: Función costos
- x: Señal de retroalimentación en el sistema
- Qx: Peso simétrico de la señal de retroalimentación en el sistema
- u: Señal de acción en el control
- Ru: Peso simétrico en la señal de acción en el control

La matriz K se obtiene en base al problema analizado por las ecuaciones Euler-Lagrange y aplicando cálculo de varias variables, teniendo así la ley de control en función a la retroalimentación del estado dado en la ecuación (2).

$$K = -R^{-1}B^{T}P \tag{2}$$

Donde P, es la solución algebraica de la ecuación Ricatti, descrita en la ecuación (3).

$$A^{T}P + PA - PBR^{-1}B^{T}P + Q = 0 \tag{3}$$

Donde:

- A: Subespacio invariante
- P: Matriz de la covarianza del error
- B: Matriz de peso simétrico
- R: Matriz de covarianza del ruido en las mediciones
- Q: Matriz de peso simétrico

En consecuencia, el estado óptimo en el lazo de control sigue la formulación $u = -Kx$, donde K es la matriz de ganancia óptima previamente analizada (D. I. Zabala-Benavides, 2021).

1.4.12 Agente Actor-Crítico

El agente actor crítico es un algoritmo que según *Mathworks*, su objetivo es optimizar la política del actor y capacitar a un crítico para estimar el retorno o recompensa futuras (MathWorks, 2020).

El agente recibe observaciones y recompensa, a su vez, envía acciones al entorno para modificar su estado, dicho agente contiene una política y un algoritmo de aprendizaje.

La política es un mapeo que selecciona acciones basadas en las observaciones del entorno mientras que el algoritmo de aprendizaje actualiza constantemente los parámetros de la política en función de las observaciones, acciones y recompensa, el objetivo del algoritmo de aprendizaje es encontrar una política óptima que maximice la recompensa acumulada esperada a largo plazo (MathWorks, 2020).

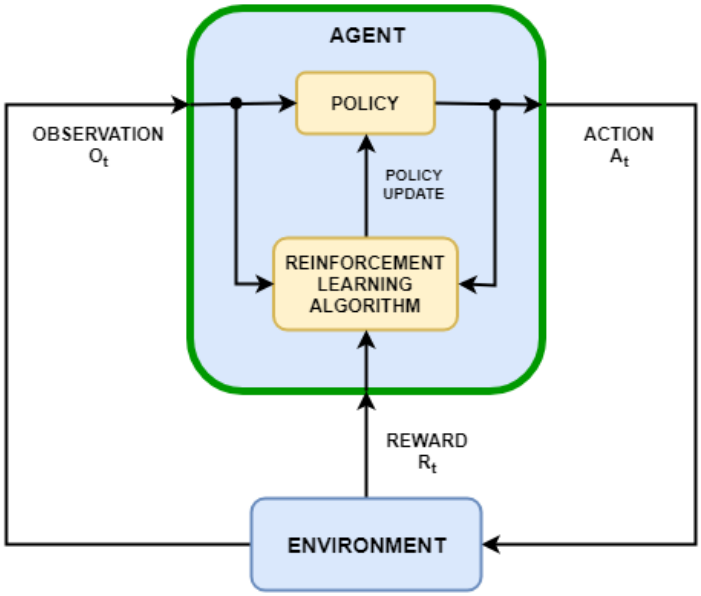


Figura 1.16 Agente general del aprendizaje por refuerzo. (MathWorks, 2020)

Dependiendo del algoritmo de aprendizaje, un agente mantiene uno o más parámetros para el entrenamiento de la política, los agentes que solo utilizan un crítico, para seleccionar sus acciones se basan en una representación indirecta de la política, este tipo de agente está basado en valores y utiliza aproximaciones para representar una función de valor basado en la función Q-valores, estos agentes son recomendados para sistemas con acciones discretas debido a que requieren altos recursos computacionales para espacios continuos.

En el caso de los agentes que solo utilizan un actor para seleccionar sus acciones se basan en una representación directa, también se denominan agentes basados en políticas, donde esta puede ser determinista o estocástica, tiene como característica que son agentes más simples y pueden manejar espacios de acción continua con la desventaja de que el algoritmo de entrenamiento puede ser sensible a la medición ruidosa provocando convergencia a mínimos locales en el gradiente de la política.

Por último tenemos a los agentes basados en ***actor-crítico*** los cuales utilizan, un actor y un crítico para el algoritmo de aprendizaje, durante el entrenamiento el actor aprende la mejor acción a tomar usando como retroalimentación al crítico, en lugar de usar la recompensa directamente, al mismo tiempo, el crítico aprende la función de valor de las recompensas para que pueda cuestionar las acciones

del actor de forma correcta, este tipo de agente maneja espacios de acción continuos y discretos (MathWorks, 2020).

Según el tipo de acción de un sistema, un agente de actor crítico tiene los siguientes algoritmos de aprendizaje.

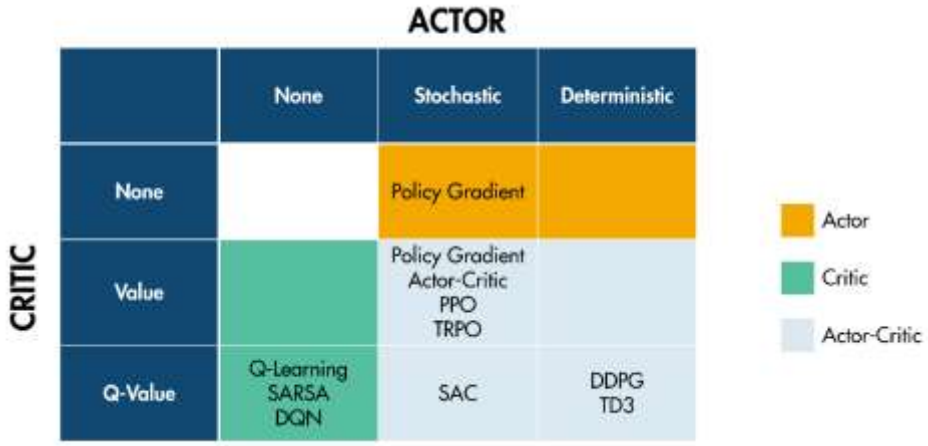


Figura 1.17 Algoritmos de aprendizaje para los agentes actor-crítico. (Matlab Academy, 2021)

1.4.13 Agente DDPG TD3

Un agente de *política de gradiente determinista profunda* (DDPG) son agentes tipo actor-crítico donde el actor emplea acciones deterministas y el crítico se basa en la función Q-Valores.

Los actores deterministas son usados para acciones continuas, donde el actor provee un mapeo directo de los valores de acciones a partir de las observaciones. El crítico recibe como entrada las acciones tomadas y las observaciones para estimar el valor de seguimiento de la política actual en la situación actual (MathWorks, 2020).

Figura 1.18 Actor y crítico. (Matlab Academy, 2021)

El algoritmo DDPG funciona de la siguiente manera: el agente recibe las observaciones del sistema, este lleva las observaciones al actor determinista y al crítico, el actor realiza una acción, la cual también será enviada al crítico, el crítico estima un valor, y el agente recibe una recompensa del entorno, tal cual se muestra en la *figura 1-19.*

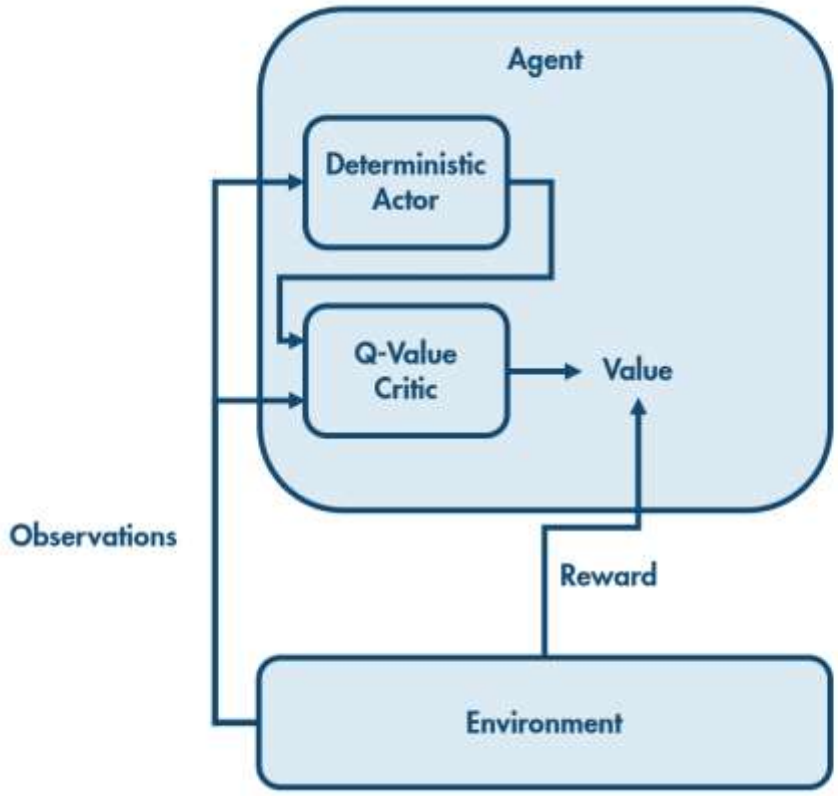


Figura 1.19 Agente DDPG. (Matlab Academy, 2021)

El entorno proporciona una recompensa al agente y éste observa el nuevo estado del entorno. El agente utiliza el nuevo estado para determinar una nueva acción. Esta acción y el nuevo estado se pasan al crítico, que determina el valor. Es decir, el crítico estima cuánta recompensa recibirá el agente en el futuro por esta situación. Combinando esto con la recompensa observada se obtiene una estimación actualizada del valor inicial del estado original (Matlab Academy, 2021).

El algoritmo de entrenamiento utiliza la estimación original y la estimación actualizada para actualizar tanto el actor como el crítico. El crítico se actualiza para producir estimaciones más cercanas a las obtenidas con las recompensas reales. El actor se actualiza para producir acciones que lleven a estados con valores más altos de recompensa.

Un agente DDPG retardo de gemelo o TD3, tiene en el agente dos redes neuronales para el crítico, recordar que el crítico busca aprender los valores Q óptimos que luego se utilizan para el ascenso de gradiente y actualizar los parámetros del actor, la ventaja de utilizar dos críticos es que mejora el error de aproximación, ya que ahora se tiene dos Q valores, uno por cada crítico (Foy, 2019).

CAPÍTULO 2

2. METODOLOGÍA

2.1 Implementación del tablero de control

Para que los estudiantes tengan la experiencia de implementar sus prácticas, se utilizaron distintos componentes donde tendrían las siguientes características como:

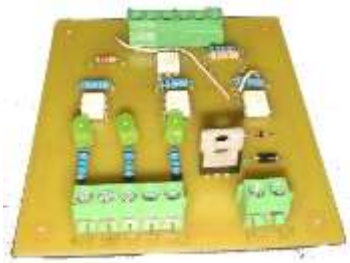
- Interfaz HMI: Permite que el usuario tenga acceso a las variables de control en el proceso, así como crear una interfaz gráfica para mejor entendimiento por el operario.
- Procesador: Este se considera como el CPU del tablero de control, el cual guardará los códigos realizados por los estudiantes.
- Pulsadores y Selectores: Su funcionamiento es de controlar, dar marcha o paro, así como seleccionar configuraciones previamente programadas por el usuario.
- Fuente de voltaje: Permite que todos los dispositivos eléctricos y electrónicos funcionen para ser utilizados por los usuarios.
- Protecciones: El tablero cuenta con protección eléctrica mediante un disyuntor, así mismo su gabinete metálico para servicio liviano ofrece una protección de grado IP 41, el cual no permite el acceso a objetos sólidos con diámetro inferior a 1mm además del goteo de agua en forma vertical.
- Riel Din: Los montajes industriales contienen los rieles que permite colocar los dispositivos a la altura y espacio adecuado en el riel.
- Indicadores: Estos se traducen a focos pilotos que están energizados a 120 V, con el fin de simular a los actuadores con el tiempo de encendido según la programación.
- Protocolo de comunicación: El protocolo usado es el Ethernet para configurar el microprocesador, mientras que el microcontrolador es a través de un USB conectada al CPU máster.

2.1.1 Lista de equipos a utilizar

La lista a continuación detalla cuales son fueron los componentes eléctricos y electrónicas utilizados para su implementación:

Ítem	Cantidad	Imagen	Descripción
1	1		**Gabinete metálico** - Referencia: GSL-403020 - Dimensiones: 400x300x200 [mm](AltoxAnchoxProfundidad) - Peso aprox.: 4.64 [kg]
2	1		**Pantalla Raspberry Pi 4** - Case plástico, ajustable por tornillos. - Pantalla touch de 7 " - Resolución 1024x600 HD - Tipo de entrada: HDMI
3	1		**Pulsador metálico Verde** - Modelo: XB2-BA31 - Marca: CNC - Código: ICO2100
4	1		**Pulsador metálico Rojo** - Modelo: XB2-BA42 - Marca: CNC - Código: ICO2105
5	1		**Selector metálico corto 2P** - Modelo: XB2-BD21 - Marca: CNC - Código: ICO2115
6	2		**Luz Piloto Verde** - Modelo: AD16-22 - Marca: CNC - Código: ICO0235 - Alimentación: 12-450 Vac
7	1		**Luz Piloto Rojo** - Modelo: AD16-22 - Marca: CNC - Código: ICO0240 - Alimentación: 12-450 Vac

8	2		**Luz Piloto Amarillo** - Modelo: AD16-22 - Marca: CNC - Código: ICO0245 - Alimentación: 12-450 Vac
9	1		**Luz Piloto Azul** - Modelo: AD16-22 - Marca: CNC - Código: ICO0250 - Alimentación: 12-450 Vac
10	1		**Shield Riel Din para Arduino Nano con comunicación Modbus RTU** - Alimentación: 12-24 Vdc - Comunicación: Serial y RS482 - Tamaño PCB: 68x72 mm - Tamaño del producto: 71x86x40 mm
11	1		**Raspberry Pi 4 Model B** - Memoria RAM: 2 GB - CPU: ARM-Cortex-A72 - Velocidad CPU: 1.5 GHz - 4 Puertos USB - 1 Puerto Ethernet - 2 Puertos micro-HDMI - Bluetooth 5.0 - Memoria: 32 GB
12	1		**Disyuntor** - Montaje: Riel Din - 2 Polos - Modelo: NXB-63 - Código: CHI0462DS - Corriente: 20 A

13	1		**Fuente de voltaje** - Montaje Riel Din - Protección: Cortocircuito, Sobrecarga, Sobre voltaje, Sobre temperatura. - Entrada: 120Vac - Salida: 24 Vdc - Modelo: EDR-75-24 - Corriente: 3.2 A - Potencia: 76.8 W
14	1		**Canal Riel Din** - Dimensiones: 35x7.5x1.0 mm - Longitud: 1 m - Material: Acero galvanizado
15	1		**Módulo de placa relé para Raspberry** - 8 Canales - Compatible: Raspberry 4 B/3B+/3B - Montaje para Riel Din - Led de activación para cada canal - Señal de disparo: 3.3 V/5 V - Dimensiones 232x72 mm
16	1		**Placa Motor Driver** - Dimensiones 72x72.2 mm - Entrada de 24 Vdc – Regulada a 5 Vdc - Interfaz de entrada para los pulsadores de comunicación con el Arduino - Control por medio de PWM al motor - Leds indicadores Alto/Bajo de pulsadores
17	1		**Motor DC** - Voltaje alimentación: 12 Vdc - Velocidad nominal: 5000 [RPM] - Encoder por sensor Efecto Hall

			- Soporte con amortiguación y ajuste en el tablero del control
18	8		**Bornera para riel din** - Voltaje 600 VAC - Rango de operación -60°C a 130°C - Corriente nominal 20A - Cable máximo 12AWG - Cable mínimo 26AWG - Dimensiones 1.87x1.88x0.21 in

Tabla 1 Materiales que constituyen el tablero de control.

2.1.2 Diseño del tablero

Luego de tener todas las herramientas y equipos se procede a realizar el diseño del tablero de prototipo académico basado en hardware de código abierto, donde se ubican de manera estratégica los diferentes módulos a utilizar.

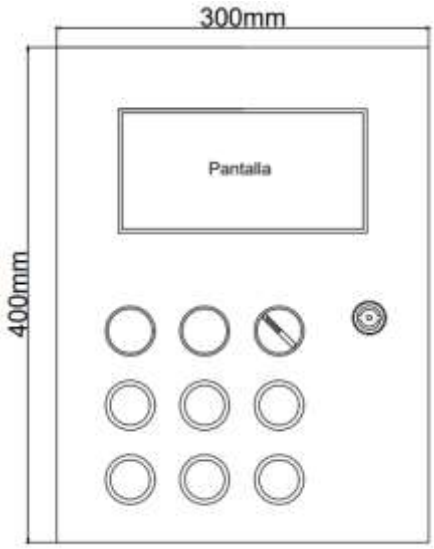


Figura 2.1 Vista frontal del tablero. Autor

En la vista frontal del tablero se encuentra la pantalla de forma que sea visible y accesible para el usuario, además se incluyen los pulsadores, selector y luces pilotos.

También se presenta en la Figura 2-1 las dimensiones reales del tablero en milímetros.

En la Figura 2-2 se puede observar los demás componentes, se encuentra en el lado superior izquierdo el *MOTOR DRIVER* el cual se encargará de accionar al

motor DC por medio de una señal PWM proveniente del microcontrolador Atmega 328P.

En el lado superior derecho tenemos al microcontrolador dentro de su *SHIELD* el cual nos brinda muchas posibilidades y ventajas como, por ejemplo, la comunicación mediante el protocolo MODBUS. Además, presenta borneras de conexión para las entradas y salidas digitales del microcontrolador.

En el centro tenemos el módulo de relés para la Raspberry Pi, el cual nos permite accionar las luces pilotos que son alimentadas con la fuente DC de 24 voltios.

En la esquina inferior izquierda tenemos el disyuntor del tablero, seguido de la fuente DC de 24 voltios y finalmente los bornes de conexiones.

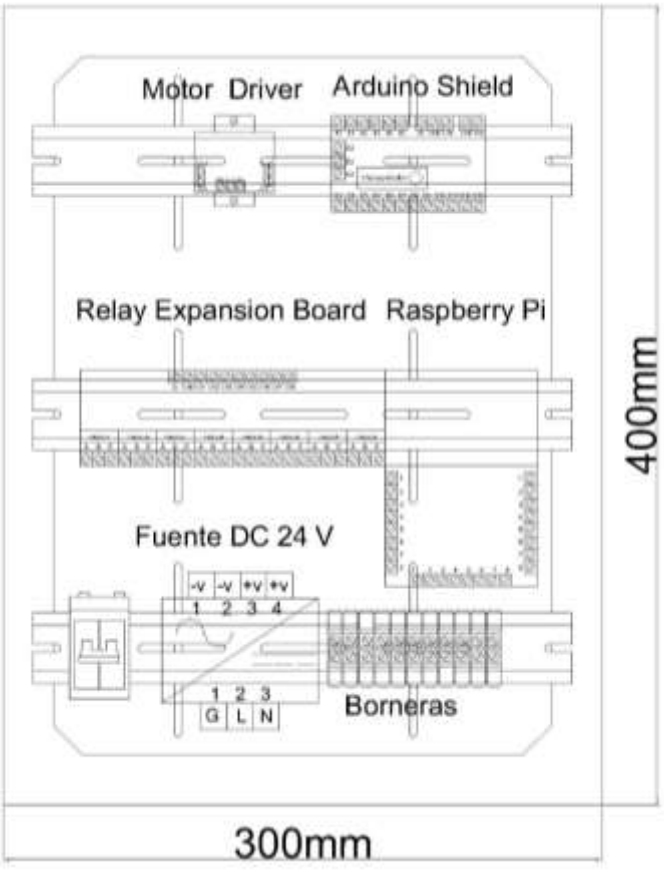


Figura 2.2 Vista Interior. Autor

En la Figura 2-3 se presenta una vista lateral de tablero junto a sus dimensiones reales en milímetro.

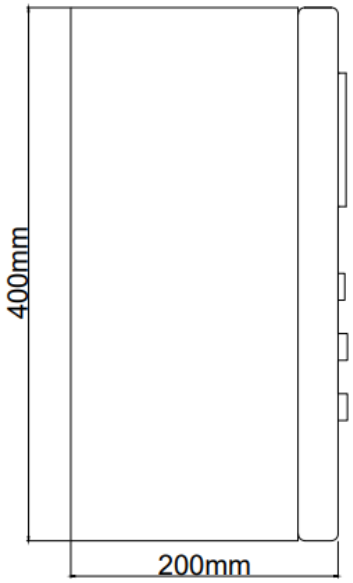


Figura 2.3 Vista Lateral. Autor

2.1.3 Diagrama esquemático

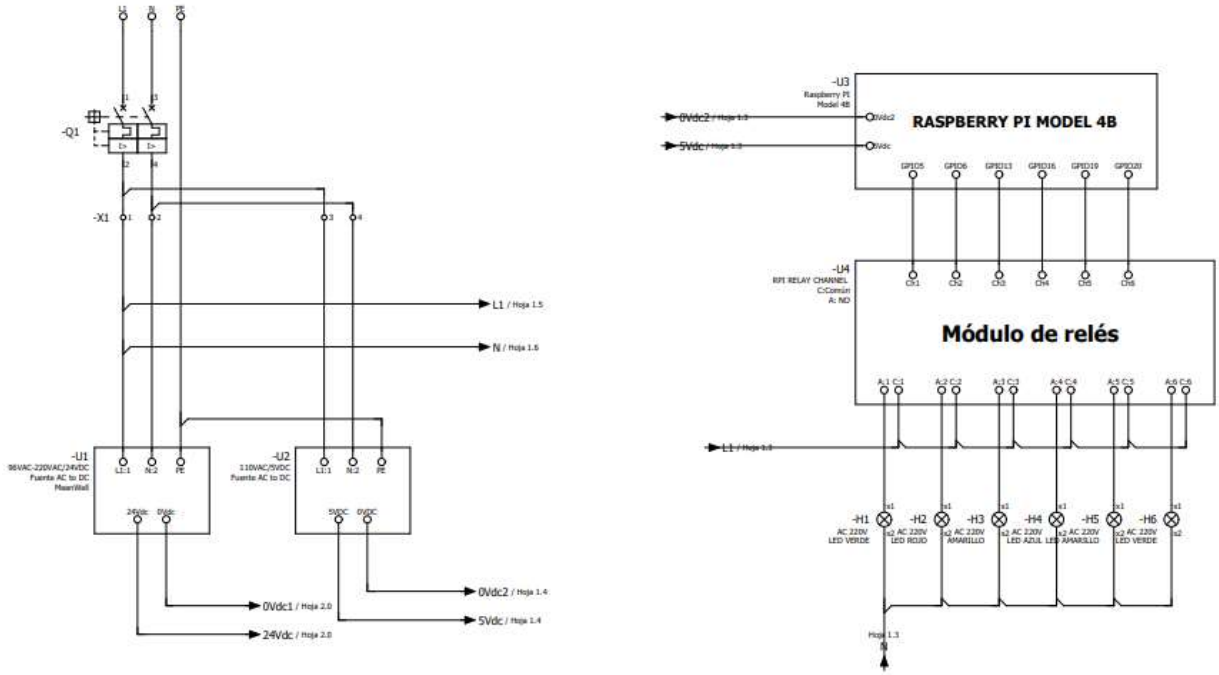


Figura 2.4 Plano Tablero. Autor

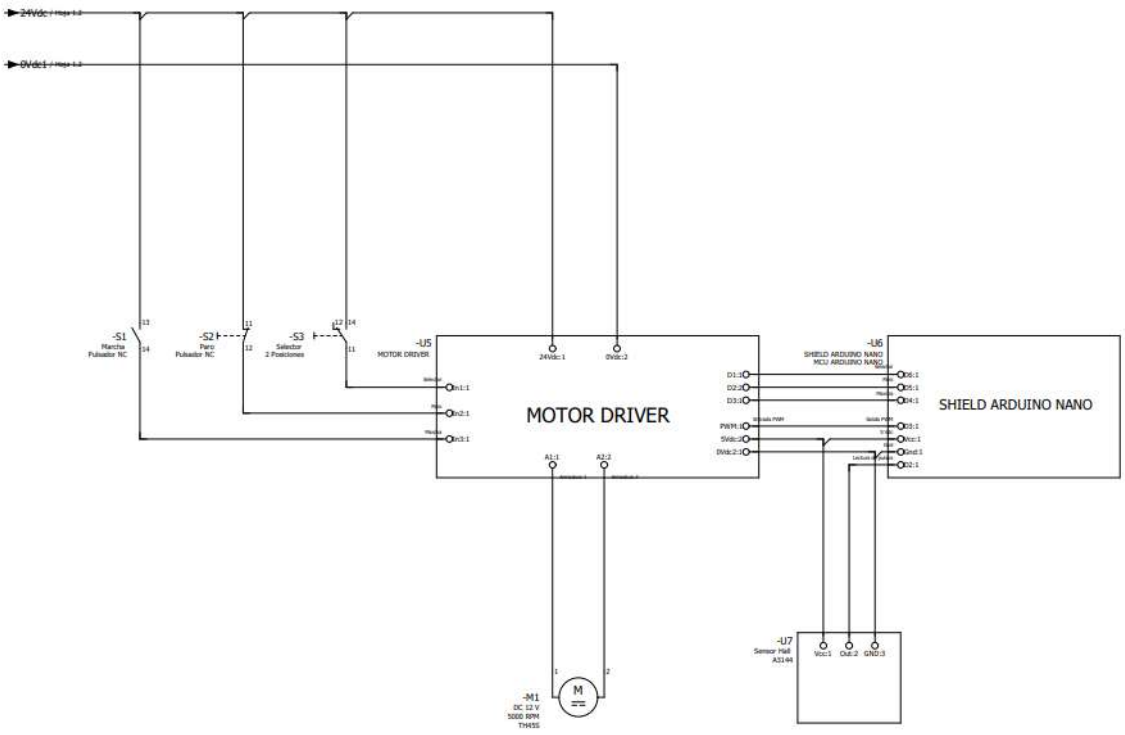


Figura 2.5 Plano tablero. Autor

Figura 2.6 Resultado final. Autor

2.2 Pruebas del tablero de control

Las pruebas realizadas en el tablero de control permiten corroborar que todos los equipos instalados presentaron una excelente conexión, además que las operaciones de las interfaces entre la pantalla y los pulsadores ejecuten ciertas acciones como es las luces pilotos o el motor DC. A continuación, se presentan las metodologías aplicadas como parte de funcionamiento y alcance que contiene el tablero de control.

2.2.1 Prueba #1: Uso de la Interfaz HMI y módulo de relé Raspberry

Para su implementación, se utilizó el software CODESYS el mismo que es pionero en la automatización industrial para los procesos que requieren una supervisión o lenguajes de programación adaptables para los estudiantes de la carrera electrónica y automatización. Por este motivo, es posible relacionarlo con el microprocesador Raspberry el cual permite programarlos mediante técnicas y uso de la interfaz gráfica del software.

2.2.1.1 CODESYS

Es considerado como software de automatización que se rige de la norma IEC 61131-3 el cual resulta independiente de los fabricantes con tal que sean compatibles con el protocolo de comunicación que permita programarlos. Es importante destacar, que Raspberry se considera de código abierto, por ello que es posible configurarlo mediante el protocolo Ethernet vinculado en una red local. Los lenguajes de programación disponibles para el usuario son de tipo texto y gráfico, cuales se detallan a continuación

Lenguajes de texto	Lenguajes gráficos
IL: Lista de instrucciones.	LD: Diagrama Ladder.
ST: Texto estructurado.	FDB: Diagrama de bloques de funciones.
	SFC: Bloques de función secuencial.
	CFC: Gráfico de función continua.

Tabla 2 Tipos de lenguaje de programación en CODESYS.

Además, cuenta con una pila de protocolos de bus de campo para CANopen, J1939, EtherCAT, EtherNet/IP, Profinet, Modbus (TCP y Serie) y Sercos III. A continuación, se enlistas los que son compatibles con el software:

- PROFIBUS
- PROFINET
- EtherCAT
- CanOpen
- J1939
- EtherNet/IP
- Sercos
- Modbus
- IO-Link
- BACnet

Raspberry previamente se enlazó a la red local para ser actualizada y descargar el programador proporcionado por el software CODESYS, el mismo que se utiliza Wifi quién previamente se ajusta su dirección IP para programarlo con el lenguaje de programación Ladder muy utilizado en aplicaciones industriales por los estudiantes.

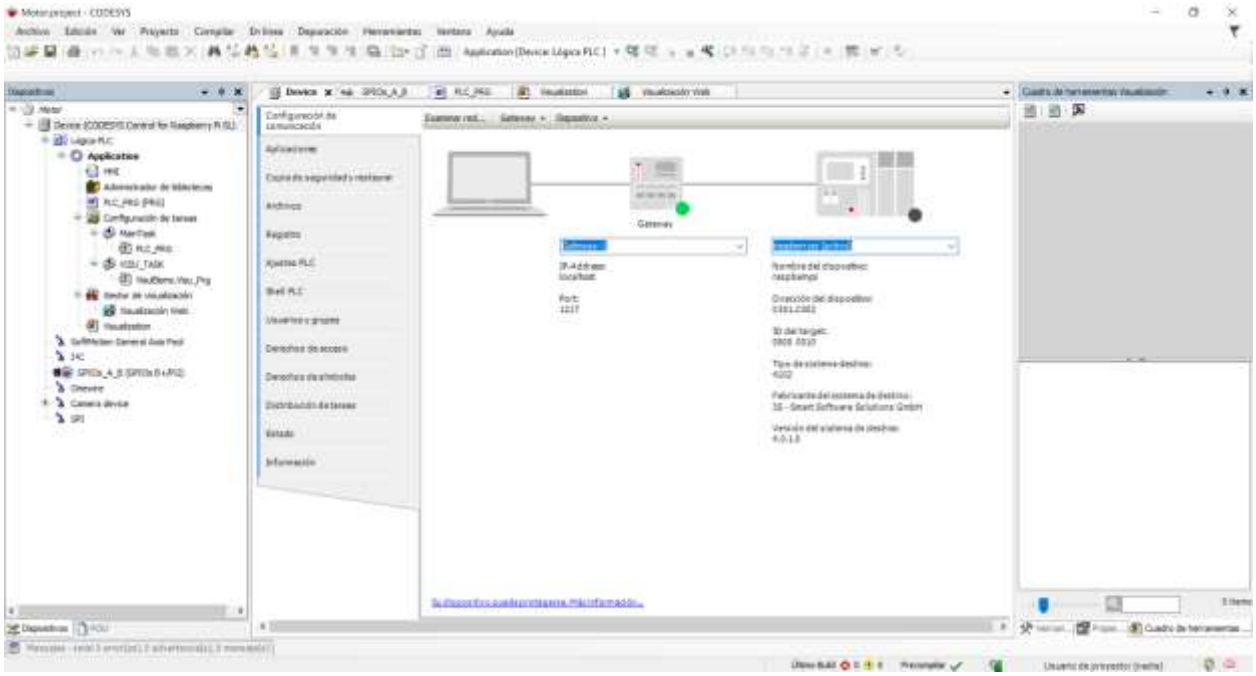


Figura 2.7 Interfaz del software CODESYS. Autor

La versión utilizada para este ejemplo fue CODESYS V3.5 SP16 Patch 4, en el proyecto se visualiza los dispositivos como: configuración, aplicación, interfaz HMI, puertos GPIO los mismo que permitirán ser entradas o salidas físicas de la Raspberry, etc. Se configura la comunicación mediante la dirección IP del dispositivo y ajusta al software para que su enlace sea exitoso.

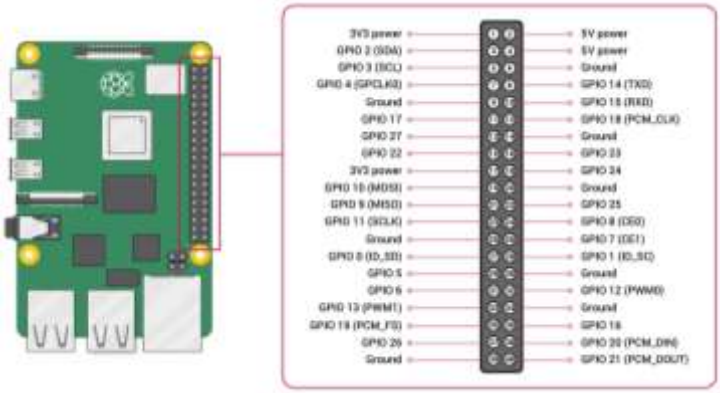


Figura 2.8 Puertos GPIO Raspberry Pi 4. (Jecrespom, 2020)

Es importante destacar que los GPIO de la placa Raspberry se encuentra conectados con un módulo de relés, los mismos que consideran los pines:

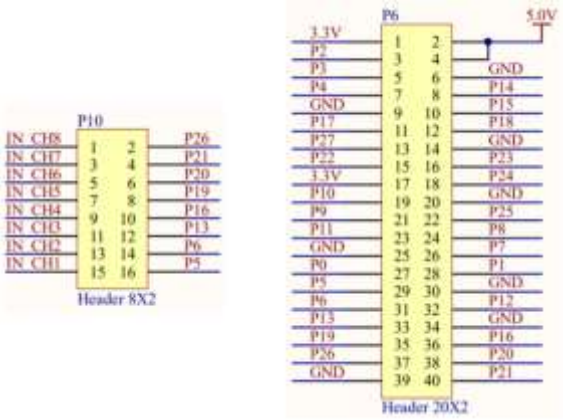


Figura 2.9 Pines ocupados desde la Raspberry al módulo relé. (ROBOTSHOP, s.f.)

Destacando de esta forma que los pines configurados para las luces leds son:

Canal	Color Luz Piloto	GPIO Raspberry
Canal 1	Verde	GPIO 5 – Pin 29
Canal 2	Rojo	GPIO 6 – Pin 31
Canal 3	Amarillo	GPIO 13 – Pin 33
Canal 4	Azul	GPIO 16 – Pin 36
Canal 5	Amarillo	GPIO 19 – Pin 35
Canal 6	Verde	GPIO 20 – Pin 38
Canal 7	Sin utilizar	GPIO 21 – Pin 40
Canal 8	Sin utilizar	GPIO 26 – Pin 37

Tabla 3 Definición entre pines del módulo y Raspberry.

2.2.1.2 Ejercicio de prueba 1 en el tablero de control

Una empresa requiere automatizar su proceso de mezclado correspondiente a dos ingredientes A y B. Para ello, es importante considerar los tiempos que tendrán tanto las electroválvulas que permiten el paso de los ingredientes como el motor que realizará la mezcla. Los requerimientos la siguiente tabla:

Modos de operación	Tiempo Electroválvula Ingrediente A [s]	Tiempo Electroválvula Ingrediente B [s]	Tiempo Motor Mezcladora [s]
Mezcla 1	5	3	7
Mezcla 2	7	6	2
Mezcla 3	9	4	5

Tabla 4 Tiempos para la implementación del ejercicio.

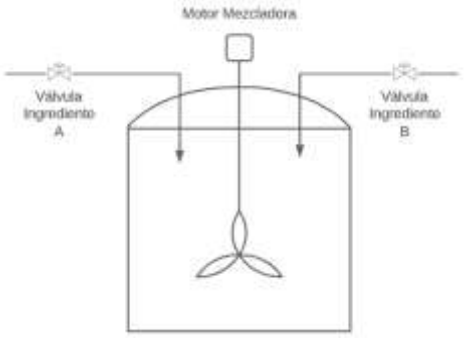


Figura 2.10 Aplicación del problema. Autor

El proceso industrial deberá contar con un HMI, donde el operario podrá establecer la marcha y parada, además que por cada mezcla seleccionada mostrar los tiempos equivalentes para cada actuador. Es importante tener indicadores pilotos para demostrar su efectividad.

2.2.1.3 Solución del ejercicio de prueba 1

Se utiliza la programación gráfica del tipo escalera, donde en primera instancia se utiliza los puertos GPIO 5, 6, 13 y 16, los cuales corresponden al sistema encendido, motor de la mezcladora, electroválvula A y electroválvula B respectivamente.

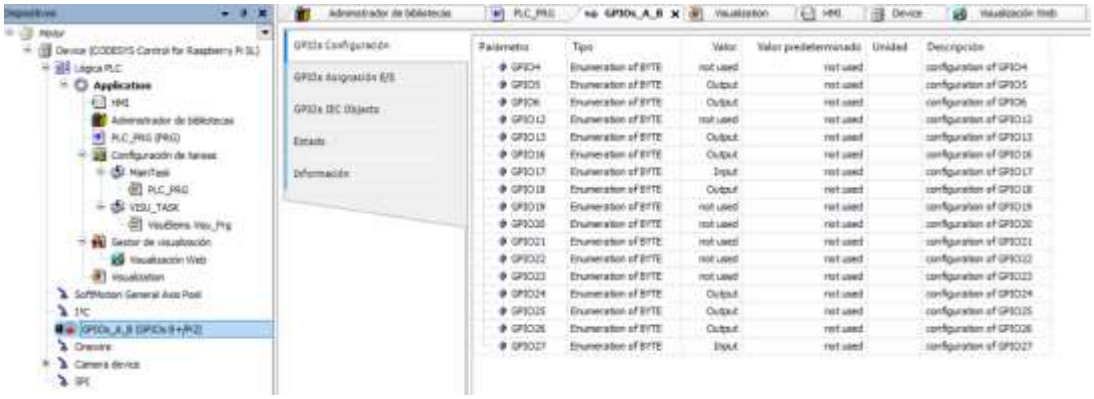


Figura 2.11 Puertos GPIO establecidos como entrada y salida de la aplicación. Autor

Las entradas usadas son parte de los pulsadores para dar marcha y parada. Mientras que las salidas relacionan a los GPIO de la Raspberry que contienen las luces pilotos respectivamente. Existen salidas del GPIO que fueron utilizadas como parte de variables en el HMI, para que estas puedan ser simuladas por medio de la programación escalera.

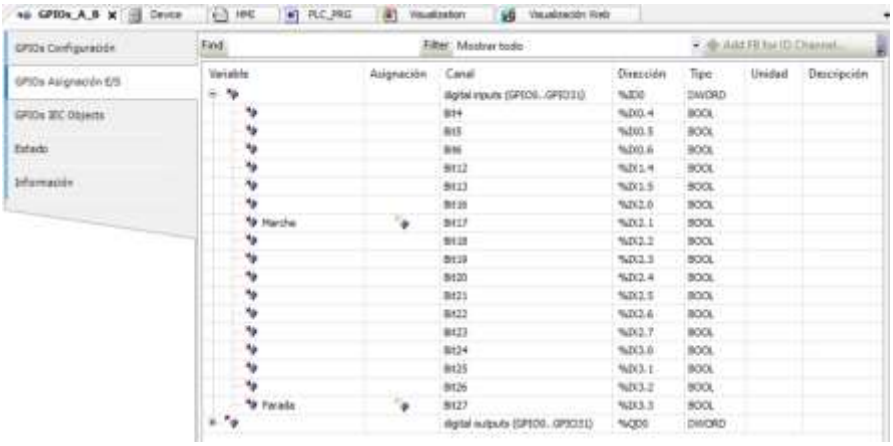


Figura 2.12 Asignación de las entradas. Autor

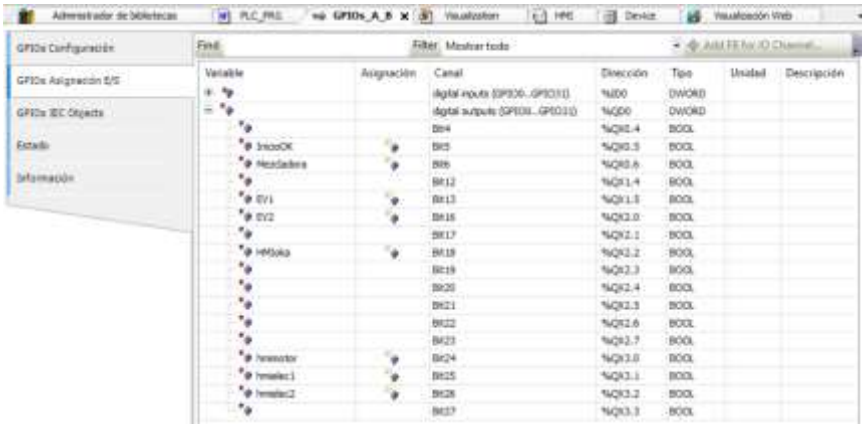


Figura 2.13 Asignación de las salidas. Autor

Luego, fue posible realizar la programación la cual se detalla a continuación.

Figura 2.14 Peldaño 1, enclavamiento del sistema. Autor

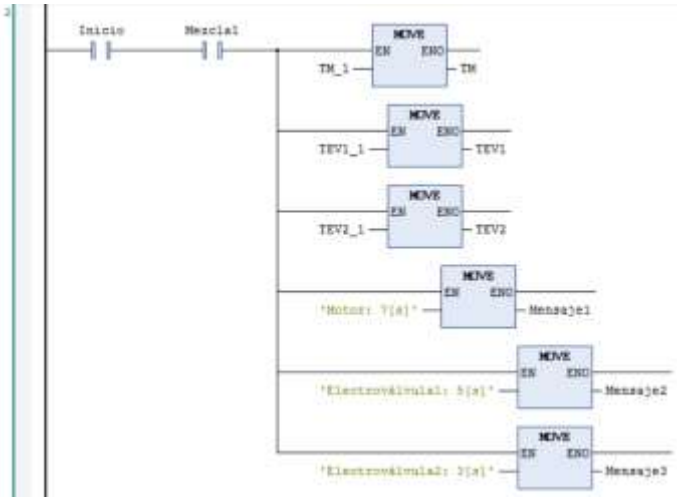


Figura 2.15 Condiciones para Mezcla 1. Autor

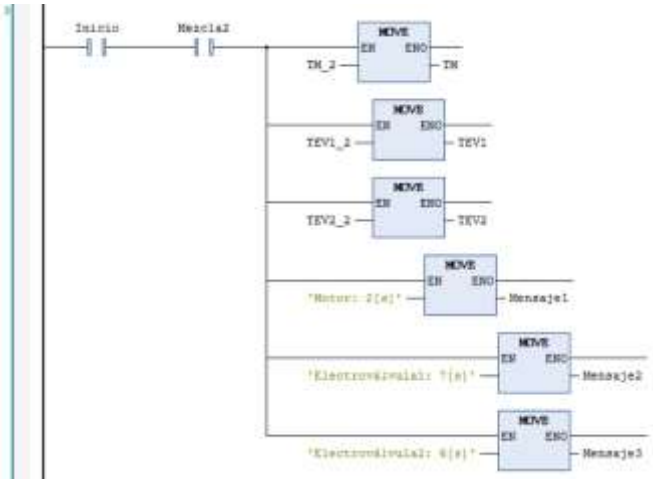


Figura 2.16 Condiciones para Mezcla 2. Autor

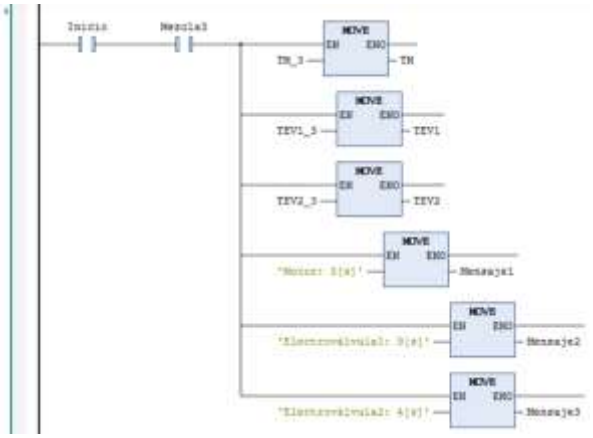


Figura 2.17 Condiciones para Mezcla 3. Autor

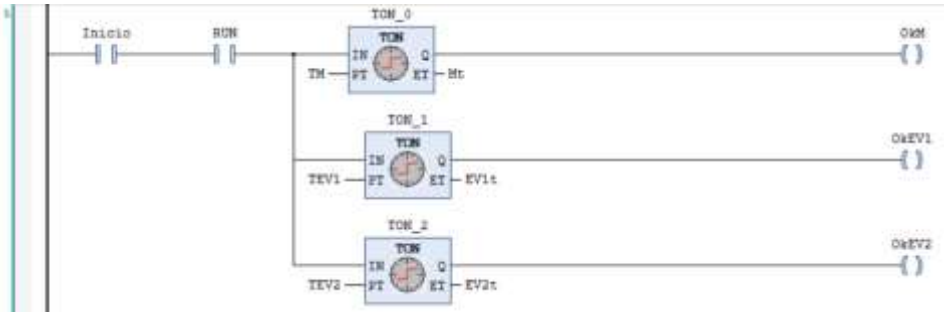


Figura 2.18 Establece los tiempos para cada mezcla. Autor

Figura 2.19 Activación de las salidas físicas. Autor

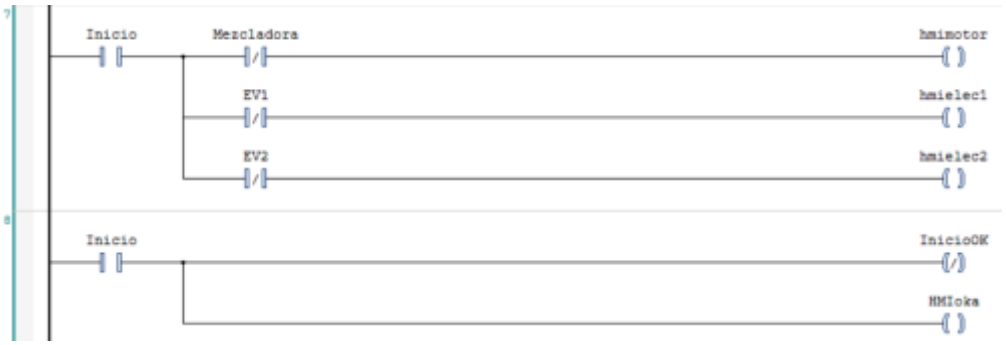


Figura 2.20 Activación de las variables en el HMI: activación, motor, electroválvula A y electroválvula B. Autor

En la figura 2-14, tiene programado el enclavamiento eléctrico el cual es activado por el pulsador de marcha tanto como una marca (variable HMI) como de las entradas físicas. En las figuras 2-15, 2-16 y 2-17 representan las condiciones de

tiempos que pertenecen según la selección del tipo de mezcla los cuales resaltan el bloque *Move*, se le asigna un valor a una variable representativa del actuador. En la figura 2-18 se tiene los bloques TON que son activados al retardo de la conexión. Finalmente, la figura 2-19 y 2-20 representan la activación tanto de las salidas físicas como las marcas representativas en el HMI.

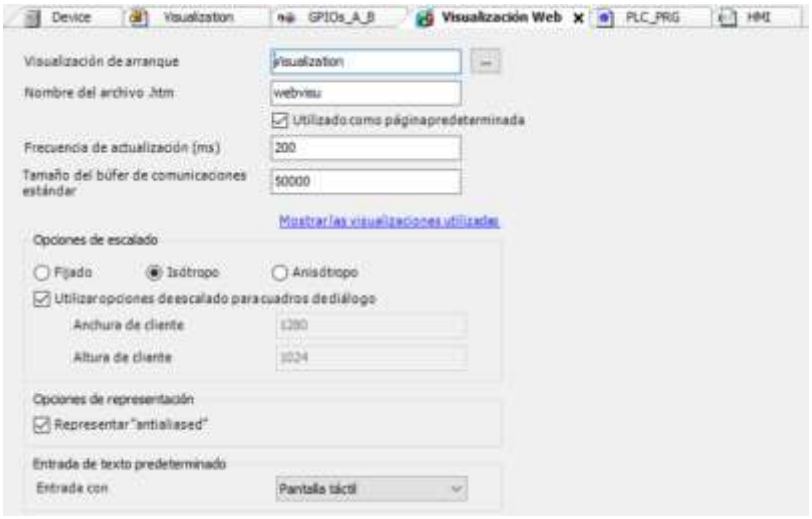


Figura 2.21 Se crea la interfaz de visualización. Autor

La interfaz de visualización web permite que el usuario a través de la Raspberry pueda tener acceso al HMI mediante la configuración de la red local, donde en su navegador se debe escribir lo siguiente: https://192.168.100.145:8080/webvisu.htm la dirección IP corresponde a la Raspberry, el puerto de enlace es fijo y a continuación se indica el mismo nombre del archivo ".htm" configurado en CODESYS. Es posible además tener la opción que en cualquier navegador visualizarlo siempre y cuando se encuentre en la red local.

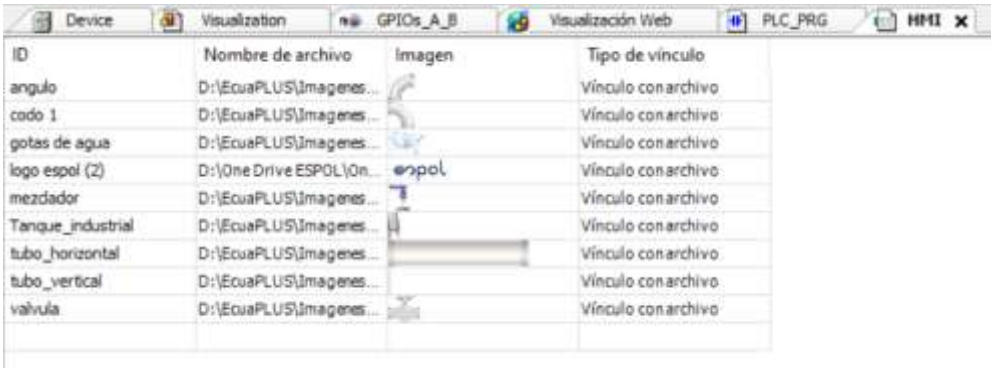


Figura 2.22 Adjuntar imágenes en el software. Autor.

Figura 2.23 Herramienta visualización software. Autor

CODESYS contiene varios elementos que permiten programar la visualización con distintos gráficos catalogados por literales vistos en la figura 2.20, además en caso de no encontrar el gráfico deseado, es posible agregarlo desde un archivo del PC, destacando así el uso de visualización hechas por los usuarios vistos en la figura 2.19.

Figura 2.24 Implementación HMI. Autor.

Finalmente, se debe crear una visualización en el software donde contiene todas las variables seteadas en la programación, de esta forma destacamos tantos los actuadores que son las salidas físicas como marcas en el sistema, las mismas que toman de referencia entre la conmutación del on/off. Se presentan cuadro de textos del tipo flotante con el fin de observar los tiempos destinados en cada

mezcla, además de conservar los pulsadores de marcha y parada, así como los de selección. A continuación, se presentan varias imágenes representativas como parte de la implementación del tablero de control, reflejando así el uso del HMI como de las salidas físicas en el sistema.

Figura 2.25 Implementación de la programación. Autor.

2.2.2 Prueba # 2: Comunicación entre sistemas embebidos y MQTT

Esta prueba plantea dos objetivos, la comunicación serial entre sistemas embebidos y el uso del protocolo MQTT para la publicación de datos en la nube.

Al realizar dicha prueba demostramos la funcionalidad total del tablero, ya que se realiza comunicación serial entre el Arduino Nano y la Raspberry Pi 4B+, se utiliza la pantalla táctil para mostrar el monitoreo en tiempo real y se usan las luces pilotos para corroborar el funcionamiento del módulo de relé que va conectado a los pines GPIO de la Raspberry.

En la ***Figura 2-26*** se explica de forma general como fluye la información desde el sensor de efecto Hall, hasta la publicación en el *broker* de *Thingsboard Cloud*, notamos que consta de las etapas: Adquisición de datos, Comunicación serial, extracción de datos, control de GPIO y publicación en la nube.

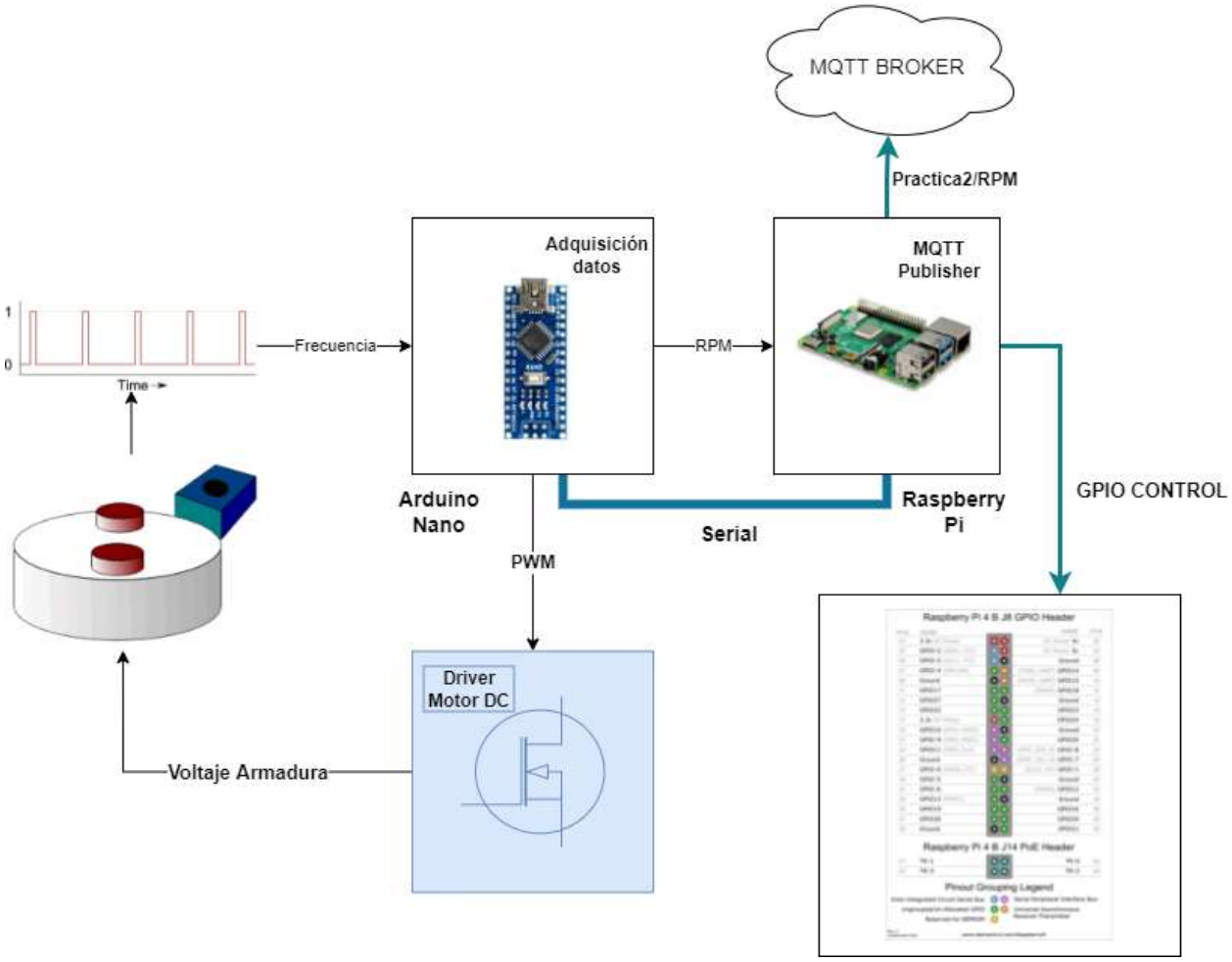


Figura 2.26 Diagrama general práctica #2. Autor

2.2.2.1 Arduino Nano

El microcontrolador Arduino Nano tiene la función de adquirir los datos del entorno, como se explicó anteriormente, se tiene un sistema de motor DC y un sensor de efecto Hall el cual se encarga de enviar flancos positivos al Arduino Nano cada vez que detecte el imán en la rueda del motor. Estos flancos de subidas serán contados y procesados en el código de Arduino para transformarlos a RPM.

El Arduino Nano también receptará las señales de los pulsadores y el selector, el algoritmo empleado lo explica el diagrama de flujo de la ***Figura 2-27.***

Según el pulsador accionado se aumentará o disminuirá en 5% el tiempo de encendido de la señal PWM, con el objetivo de variar la velocidad de giro del motor DC.

Se usaron dos tipos de interrupciones, la externa que es accionada por un flanco de subida en el pin digital 2, y la interrupción por el temporizador 1 de Arduino, el

cual esta seteado por 100 ms, de esta forma, podemos simular un sistema multitarea de una forma eficiente.

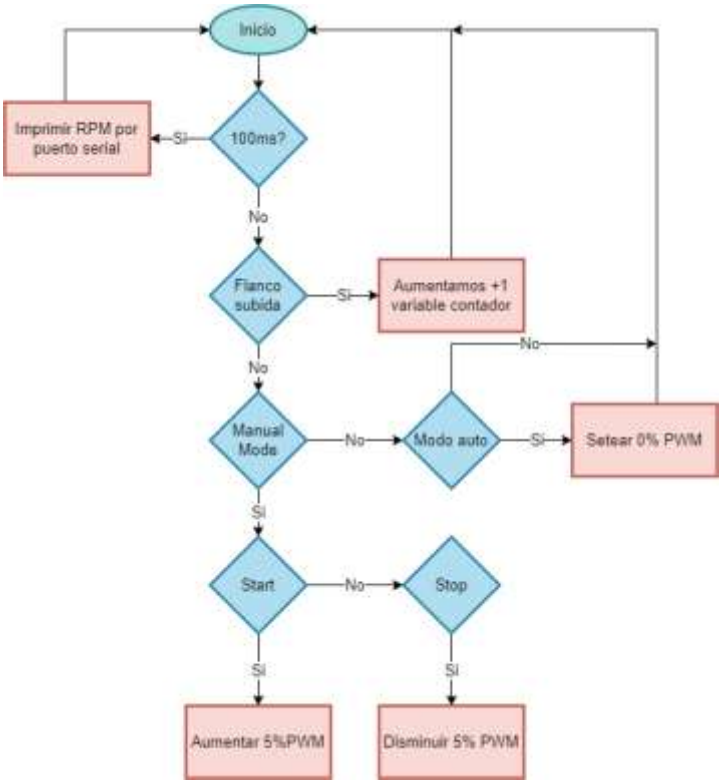


Figura 2.27 Diagrama de flujo del Arduino para la Práctica # 2

2.2.2.2 Placa PCB

Se diseñó e implementó una placa electrónica que maneja el motor DC, aísla y baja el nivel de voltaje de las señales proveniente de los pulsadores y selector del tablero. El diagrama esquemático de la placa se puede observar en la ***Figura 2-28,*** la primera etapa consta de 3 circuitos *"Pull down"* que están en la salida de un optoacoplador, para que el usuario pueda manejar altos valores de voltajes DC en los dispositivos de accionamiento y que el Arduino los pueda leer sin comprometerlo.

Luego se tiene un circuito controlador de PWM donde el elemento principal es el dispositivo semiconductor MOSFET, el cual, al recibir una señal por compuerta, acciona al motor DC entregando la potencia que dicho motor requiere en función de la fuente de alimentación del circuito controlador de motor.

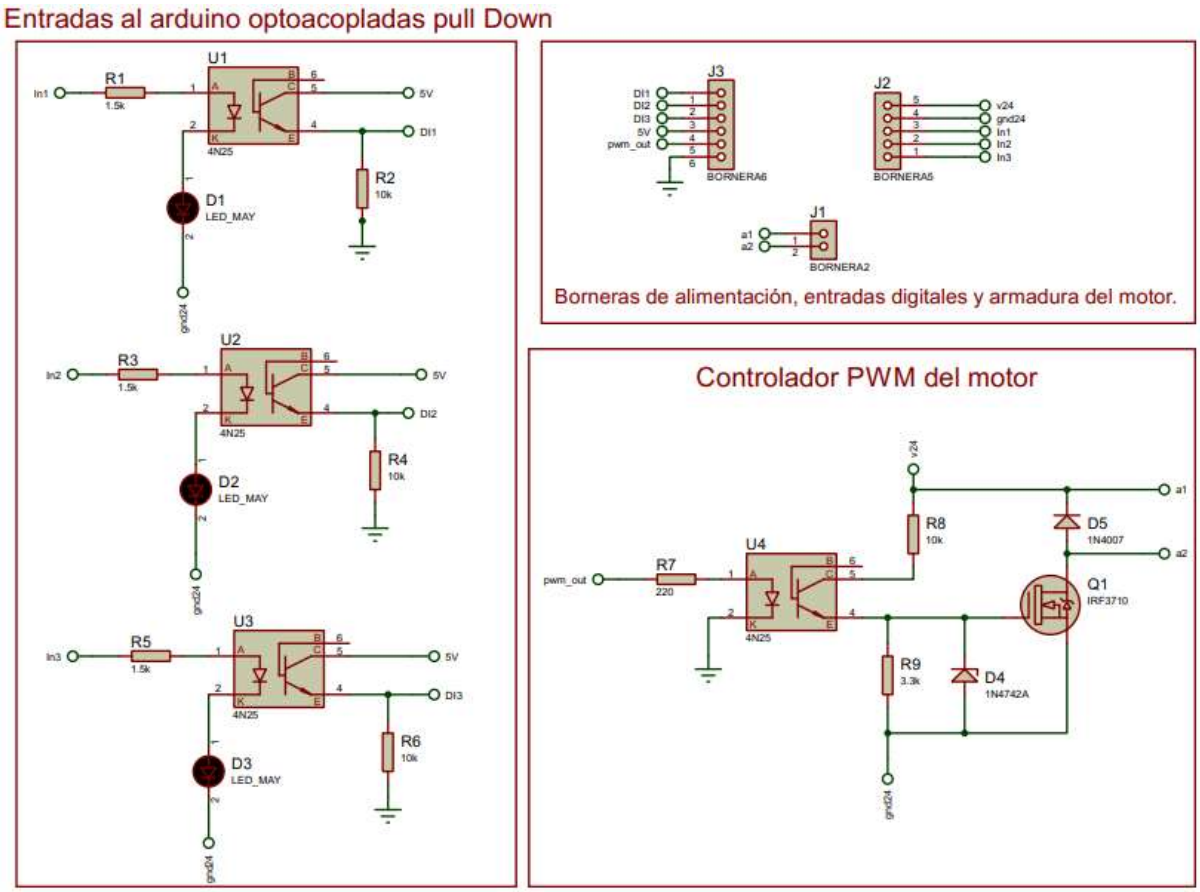


Figura 2.28 Diagrama esquemático placa MOTOR DRIVER. Autor

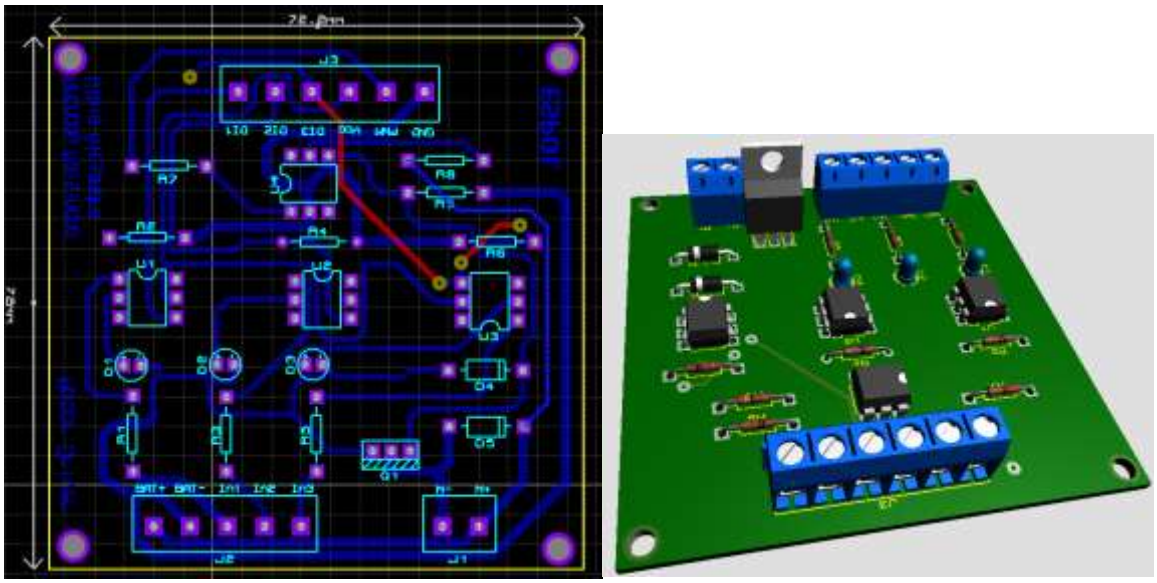


Figura 2.29 Diseño placa PCB Motor Driver. Autor

2.2.2.3 Raspberry Pi

La Raspberry Pi es la encargada de la recolección de datos, tanto de las RPM como el porcentaje de PWM enviado, controlar las luces pilotos del tablero y establecer la conexión como cliente publicador de datos en la nube en la plataforma de *ThingsBoard*.

El código desarrollado consta de importar librerías, definirse como cliente, acceso al token, conexión al host por medio de MQTT, extracción de datos y validación para condiciones de las salidas GPIO.

Para correr el script de Python se utilizó el programa MobaXterm, el cual por medio del túnel SSH podemos manejar de forma remota a la Raspberry PI 4b.

2.2.2.4 Thingsboard

Se agregó un dispositivo y se realizó un SCADA para leer las variables RPM y porcentaje de PWM.

Figura 2.30 Dashboard en Thingsboard. Autor

Figura 2.31 Funcionamiento del tablero con la prueba #2. Autor

2.3 Obtención de parámetros experimentales del motor DC.

2.3.1 Modelado del motor DC

El modelado matemático consiste en la representación del sistema (Motor DC) a través de ecuaciones, estas pueden ser diferenciales dependiendo de cuáles son los elementos que lo constituyen. Para esto, se debe tener un alto grado de entendimiento físico, así como sus perturbaciones, por ese motivo se considera un proceso riguroso y que requiere de varias condiciones para obtener finalmente con la conversión de Laplace una función de transferencia en el dominio de la frecuencia s.

Con la función de transferencia es posible analizar el comportamiento dinámico y estático en el sistema, de esta forma es posible obtener características de sistemas de control donde resaltan: sobre nivel porcentual, tiempo en estado estable, salida en estado estable, tiempo pico, tau, etc. La respuesta del sistema es importante ya que la función de transferencia relaciona la entrada y la salida, de esta forma es posible dimensionar el proyecto ante de ejecutarlo, así como el caso del motor ubicar las protecciones o el control adecuado.

Además, con la función de transferencia mediante la teoría de control fue posible realizar el dimensionamiento los controladores PID, los cuales se ajustan dependiendo del tipo de sistema previamente estudiados. Sin embargo, se presenta cuáles son las condiciones y las variables que presenta actualmente el motor DC.

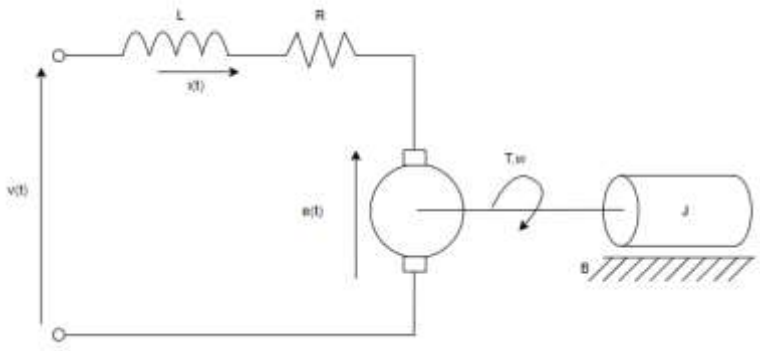


Figura 2.32 Planteamiento del problema Motor DC. Autor

Donde las constantes del sistema se definen como:

- **J:** Momento de inercia del rotor $[Kg \cdot m^2]$.
- **B:** Constante de fricción viscosa del motor $[N * m * s]$.
- **R:** Resistencia eléctrica $[\Omega]$.
- **L:** Inductancia eléctrica $[H]$.
- **Ka:** Constante de la fuerza contraelectromotriz $\left[\frac{V \cdot s}{rad}\right]$.
- **Km:** Constante del par del motor $\left[\frac{N \cdot m}{A}\right]$.

Se considera la malla de la armadura:

$$v(t) = Ri(t) + L\frac{di(t)}{dt} + e_a(t)$$

$$L\frac{di(t)}{dt} = v(t) - Ri(t) - e_a(t) \quad (4)$$

Ecuación de la sección mecánica:

$$T_m(t) = J\frac{d\omega(t)}{dt} + B\omega(t)$$

$$J\frac{d\omega(t)}{dt} = T_m(t) - B\omega(t) \tag{5}$$

La fuerza contraelectromotriz se encuentra definida por una constante, además que es directamente proporcional a la velocidad angular con la que gira el motor, por ende:

$$e_a(t) = f\{\omega(t)\} = K_a\omega(t) \tag{6}$$

La relación mecánico-electromecánico se define como una constante multiplicada con la corriente de armadura:

$$T_m(t) = f\{i(t)\} = K_m i(t) \tag{7}$$

Las cuatro ecuaciones antes mencionadas definen el comportamiento del sistema. Ahora, se aplica la transformada de Laplace, obteniendo lo siguiente:

$$LsI(s) = V(s) - RI(s) - E_a(s) \tag{8}$$

$$Js\omega(s) = T_m(s) - B\omega(s) \tag{9}$$

$$E_a(s) = K_a\omega(s) \tag{10}$$

$$T_m(s) = K_m I(s) \tag{11}$$

En base a las ecuaciones anteriores, se establecen las funciones de transferencias, por ese motivo se sustituye la ecuación (10) y (11) en la (8).

$$I(s) = \frac{T_m(s)}{K_m}$$

$$Ls\frac{T_m(s)}{K_m} = V(s) - R\frac{T_m(s)}{K_m} - K_a\omega(s)$$

$$V(s) = \frac{(Ls + R)T_m(s)}{K_m} + K_a\omega(s) \tag{12}$$

De la ecuación (9) se despeja $\omega(s)$ para reemplazarla en la ecuación 9:

$$\omega(s) = \frac{T_m(s)}{Js + B}$$

$$V(s) = \frac{(Ls + R)T_m(s)}{K_m} + K_a \frac{T_m(s)}{Js + B}$$

$$V(s) = \left(\frac{(Ls + R)}{K_m} + \frac{K_a}{Js + B}\right) T_m(s)$$

$$V(s) = \left(\frac{(Ls + R)(Js + B) + K_a K_m}{K_m(Js + B)}\right) T_m(s)$$

$$V(s) = \left(\frac{JLs^2 + (BL + RJ)s + RB + K_a K_m}{K_m(Js + B)}\right) T_m(s)$$

$$\frac{T_m(s)}{V(s)} = \frac{K_m(Js + B)}{JLs^2 + (BL + RJ)s + RB + K_a K_m} \quad (13)$$

Función de transferencia Torque - Velocidad

Ahora, a partir de las ecuaciones (8), (9), (10)$E_a(s) = K_a\omega(s)$ y (11) se encuentra la función de transferencia Fuerza Contraelectromotriz – Voltaje. Se sustituyó (11) en (8):

$$\frac{LsT_m(s)}{K_m} = V(s) - \frac{RT_m(s)}{K_m} - E_a(s)$$

$$V(s) = E_a(s) + \frac{R + Ls}{K_m} T_m(s) \quad (14)$$

Con la ecuación (9), se despeja $T_m(s)$ y se sustituye en (14).

$$T_m(s) = Js\omega(s) + B\omega(s)$$

$$V(s) = E_a(s) + \frac{(R + Ls)(Js + B)}{K_m} \omega(s) \quad (15)$$

De la ecuación (10), se despeja $\omega(s)$ sustituyéndolo en (14).

$$\omega(s) = \frac{E_a(s)}{K_a}$$

$$V(s) = E_a(s) + \frac{(R + Ls)(Js + B)}{K_m K_a} E_a(s)$$

$$V(s) = \left(1 + \frac{(R + Ls)(Js + B)}{K_m K_a}\right) E_a(s)$$

$$V(s) = \left(\frac{K_m K_a + LJs^2 + (LB + RJ)s + RB}{K_m K_a}\right) E_a(s)$$

$$\frac{E_a(s)}{V(s)} = \frac{K_m K_a}{LJs^2 + (LB + RJ)s + K_m K_a + RB} \tag{16}$$

Función de transferencia Contraelectromotriz - Voltaje

Mientras que son las ecuaciones (8), (9), (10)$E_a(s) = K_a\omega(s)$ y (11) es posible encontrar la función de transferencia Corriente de Armadura – Voltaje. Sustituyendo la ecuación (10) en (8).

$$LsI(s) = V(s) - RI(s) - K_a\omega(s) \tag{17}$$

Con la ecuación (9), se despeja $\omega(s)$ y sustituir en (17).

$$LsI(s) = V(s) - RI(s) - \frac{K_a}{Js + B}T_m(s) \tag{18}$$

Se reemplazó la ecuación (11) en (18).

$$LsI(s) = V(s) - RI(s) - \frac{K_a K_m}{Js + B}I(s)$$

$$V(s) = LsI(s) + RI(s) + \frac{K_a K_m}{Js + B}I(s)$$

$$V(s) = \left(\frac{(Ls + R)(Js + B) + K_a K_m}{Js + B}\right)I(s)$$

$$V(s) = \left(\frac{LJs^2 + (BL + RJ)s + RB + K_a K_m}{Js + B}\right)I(s)$$

$$\frac{I(s)}{V(s)} = \frac{Js + B}{LJs^2 + (BL + RJ)s + RB + K_a K_m} \tag{19}$$

Función de Transferencia Corriente de Armadura - Voltaje

A partir de las ecuaciones (8), (9), (10)$E_a(s) = K_a\omega(s)$ y (11) se encuentra la Función de Transferencia Velocidad Angular – Voltaje. Para lo cual se sustituyó la ecuación (11) en (9).

$$Js\omega(s) = K_m I(s) - B\omega(s)$$

$$I(s) = \frac{(Js + B)}{K_m}\omega(s) \tag{20}$$

Se sustituye la ecuación (20) en (8):

$$\frac{Ls(Js + B)}{K_m}\omega(s) = V(s) - \frac{R(Js + B)}{K_m}\omega(s) - E_a(s) \tag{21}$$

Se sustituye la ecuación (10) en (21):

$$\frac{Ls(Js+B)}{K_m}\omega(s) = V(s) - \frac{R(Js+B)}{K_m}\omega(s) - K_a\omega(s)$$

$$V(s) = \left(\frac{(Ls+R)(Js+B)}{K_m}\right)\omega(s) + K_a\omega(s)$$

$$V(s) = \left(\frac{LJs^2 + (LB+RJ)s + RB + K_aK_m}{K_m}\right)\omega(s)$$

$$\frac{\omega(s)}{V(s)} = \frac{K_m}{LJs^2 + (LB+RJ)s + RB + K_aK_m} \quad (22)$$

Función de Transferencia Velocidad Angular – Voltaje

Para obtener la función de transferencia del Desplazamiento – Voltaje, se debe conocer que:

$$\omega(t) = \frac{d\theta(t)}{dt}$$

Aplicando la transformada de Laplace:

$$\omega(s) = s\theta(s) \quad (23)$$

Reemplazamos (23) en (22), se tiene que:

$$\frac{\theta(s)}{V(s)} = \frac{K_m}{s(LJs^2 + (LB+RJ)s + RB + K_aK_m)} \quad (24)$$

Función de Transferencia Desplazamiento - Voltaje

En resumen, se encontraron las siguientes funciones de transferencias del sistema:

- (13): Función de transferencia Torque – Voltaje
- (16): Función de transferencia Contraelectromotriz – Voltaje
- (19): Función de transferencia Corriente de Armadura- Voltaje
- (21): Función de Transferencia Velocidad Angular – Voltaje
- (24): Función de Transferencia Desplazamiento - Voltaje

2.3.2 Medición experimental de los parámetros del motor DC.

El objetivo de la obtención de los parámetros experimentales del motor DC es tener una referencia del comportamiento del controlador basando en aprendizaje por refuerzo y la dinámica de la planta, ya que así tendremos un modelo simulado y el sistema real.

A pesar de que el algoritmo a utilizar no necesita conocer la planta o la función de transferencia del sistema, se la estimó mediante pruebas experimentales para verificar ambos comportamientos y poder generar datos de la dinámica del sistema simulado probando diferentes tipos de controladores.

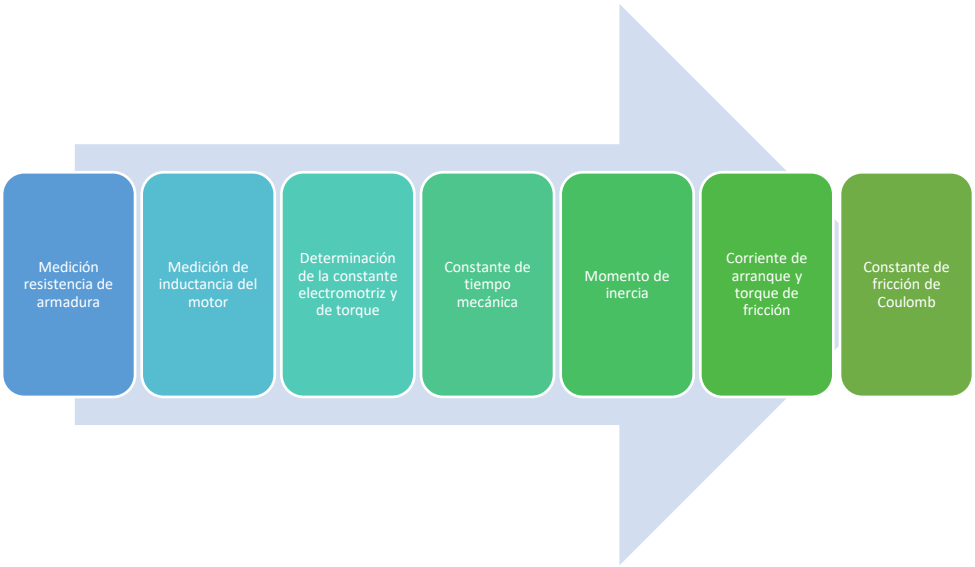


Figura 2.33 Procesos empleados para la obtención de los parámetros experimentales del motor DC. Autor

El motor DC utilizado tiene las siguientes características:

- **Modelo:** TH45S
- **Marca:** TYHE
- **Voltaje alimentación:** 12 [Vdc]
- **Torque:** 100[mN·m]
- **Velocidades desde:** 2000 rpm hasta 6000 rpm
- **Corriente nominal:** 1-3.5 [A]
- **Eficiencia:** IE2

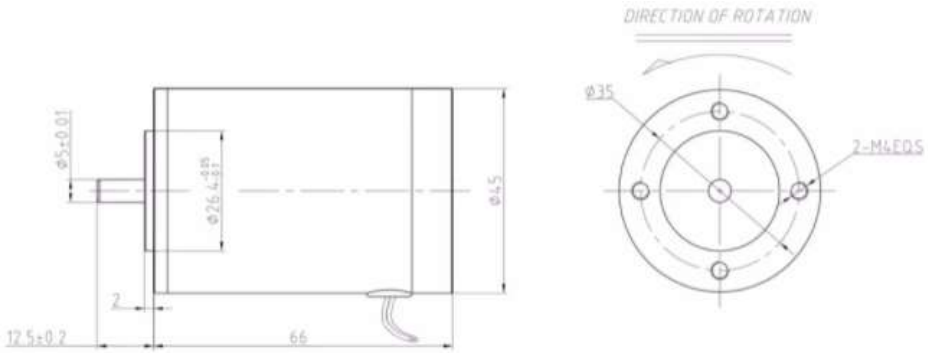


Figura 2.34 Dimensiones del motor DC. (TYHE, s.f.)

Para determinar la ***resistencia de armadura*** se puede emplear dos métodos.

El primer método es utilizar un multímetro y configurarlo en ohmios para medir la resistencia de armadura, la cual podemos acceder desde los dos cables que posee el motor.

El segundo método experimental es utilizar una fuente de alimentación variable, setear el voltaje de alimentación a cero e ir incrementando gradualmente hasta que el motor quiera salir de la inercia, en ese momento, mediante medición indirecta y por ley de Ohm se puede determinar la resistencia, ya que; en este instante el voltaje contraelectromotriz es cero (Ortega, 2009) .

Se utilizó el primer método, para esto se utilizó el multímetro de alta precisión *RIGOL DM3058E*:

Figura 2.35 Medición de resistencia de armadura. Autor

La ***inductancia de armadura***, al igual que la resistencia de armadura se mide en los devanados del motor DC, utilizando un instrumento que sirve para medir inductancias llamado LCR meter (Ortega, 2009).

Para la medición se utilizó el instrumento *UNI-T UT612* el cual nos permite medir inductancia, capacitancia, resistencia y otros parámetros con alta precisión.

Figura 2.36 Medición de inductancia de armadura. Autor

Para la obtención de la ***constante electromotriz y de torque*** tenemos que utilizar el modelamiento matemático del motor DC, las ecuaciones a emplear se obtienen del siguiente circuito.

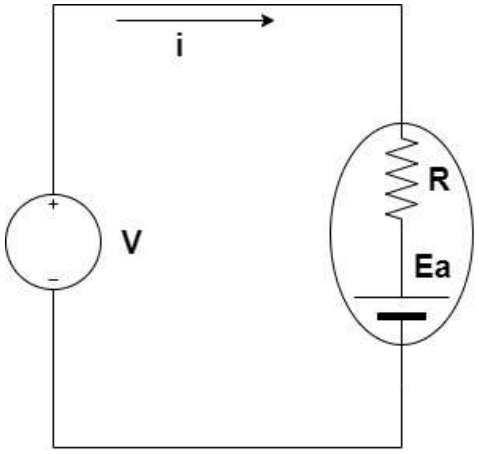


Figura 2.37 Modelado eléctrico motor DC. Autor

Utilizando la ley de voltaje de Kirchoff se obtuvieron las siguientes relaciones:

$$K_a = \frac{E_a}{\omega(t)}$$

$$V = iR + Ea$$

$$\therefore\ K_a = \frac{V - iR}{\omega(t)}\ (22)$$

Donde las variables mencionadas representan:

- K_a: Constante electromotriz
- V: Voltaje aplicado
- i: Corriente de armadura
- R: Resistencia de armadura
- $\omega(t)$: Velocidad en radianes por segundo

La **ecuación 22** nos indica que debemos medir el voltaje aplicado a la armadura, corriente y la velocidad en radianes por segundo, para esto se planteó el siguiente sistema.

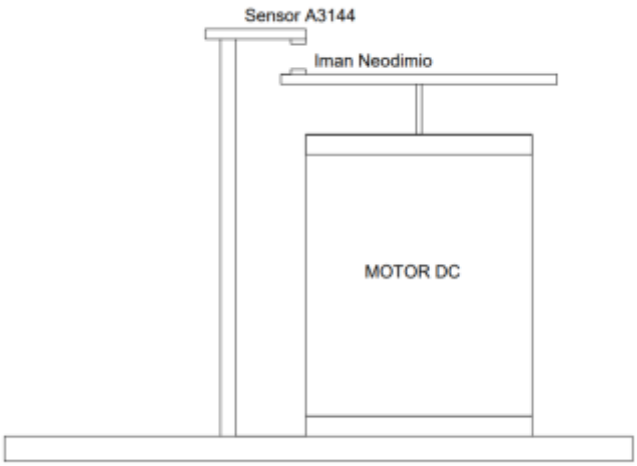


Figura 2.38 Sistema de medición RPM. Autor

Se utilizó el sensor *A3144* el cual es de efecto Hall, su función principal es detectar campos magnéticos y tiene la ventaja de que retiene estados, es decir, cambia de estado de un valor booleano 0 a 1 en función del estado actual, permitiendo detectar de manera correcta los pulsos de subida que entrega el sensor al momento de detectar campos magnéticos.

Especificaciones técnicas

- Voltaje de operación: 4.5V ~ 24V DC
- Corriente máxima: 25mA.
- Rango de temperatura: -40～85℃

Figura 2.39 Sensor de Efecto Hall. (ElectronicLab, s.f.)

La figura nos indica como se realizará las mediciones de estas variables para poder determinar la constante electromotriz y de torque.

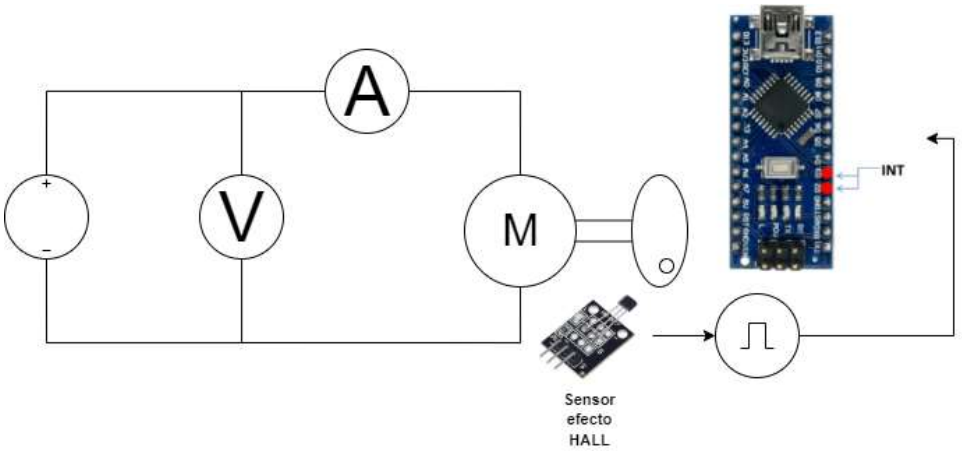


Figura 2.40 Diagrama de bloque para la obtención de la constante electromotriz y de torque. Autor

Los valores de voltaje y corriente de esta prueba fueron obtenidos con el multímetro, sin embargo, para la velocidad, se utilizó un código que permitió obtener dicha variable mediante el conteo del tiempo entre pulsos que llegan al pin 2. Es importante utilizar esta técnica, ya que si contamos el número de vueltas que ha dado nuestro motor en un determinado tiempo, se obtienen lecturas erróneas de la velocidad debido a que solo se toma en cuenta las vueltas completas.

Este algoritmo al detectar un pulso comienza a contar el tiempo mediante una función de Arduino hasta que detecta otro pulso, de esta forma se obtiene un periodo, que puede ser convertido a frecuencia y posteriormente a revoluciones por minuto, a continuación, el siguiente diagrama de flujo explica el algoritmo.

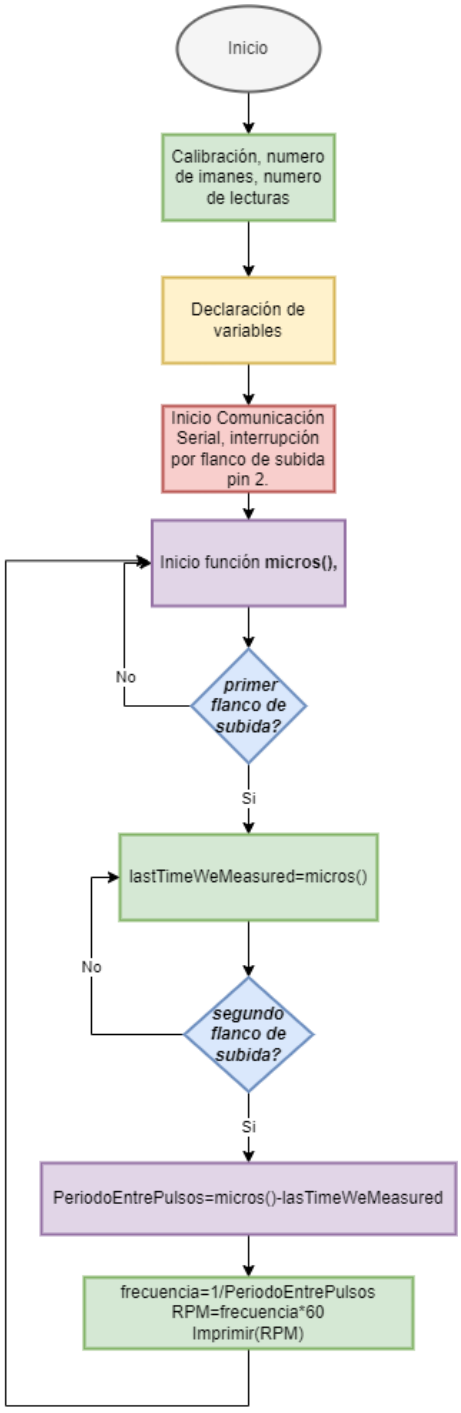


Figura 2.41 Diagrama de flujo Lectura de velocidad. Autor

Ahora se procederá a determinar la constante de tiempo mecánica (T_m), la cual representa el 63.2% del tiempo de estabilización del sistema. Para poder realizar esta prueba se realizó un código que mantenga encendido el motor durante 5

segundos y lo mantenga apagado por 5 segundos, esto tiene como objetivo observar la dinámica del sistema y poder medir el tiempo de estabilización.

El equipo empleado es el *PC-OSCILLOSCOPE* modelo *HANTEK6022BE,* el cual nos mostrará el comportamiento de la curva de voltaje generada durante el transiente, aprovechando que la velocidad guarda relación directa con la tensión aplicada a la armadura.

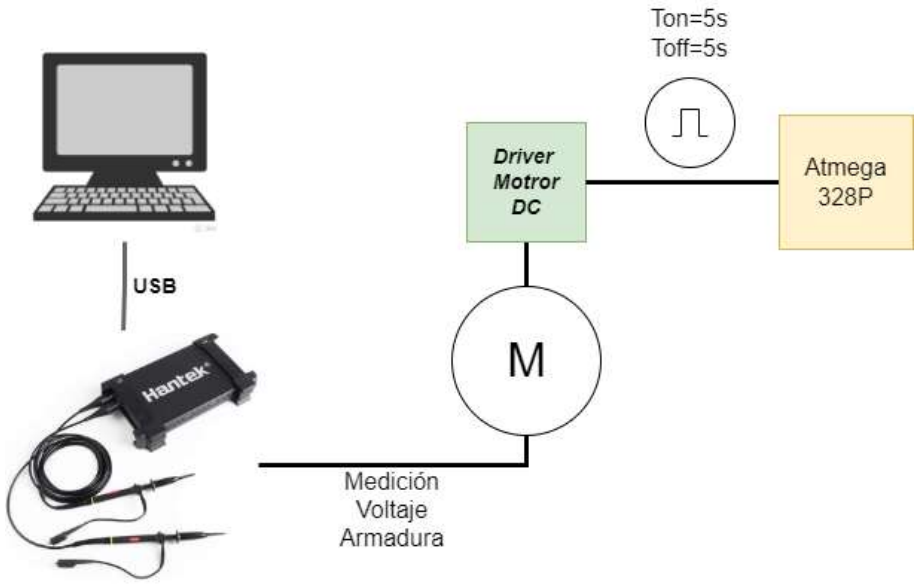


Figura 2.42 Diagrama de bloque para determinar la constante de tiempo mecánica. Autor

Con todos los parámetros obtenidos hasta ahora es posible determinar el momento de inercia, variable que depende de la constante de tiempo mecánica, contante electromotriz, de torque y resistencia de armadura.

$$J = \frac{T_m \cdot K_a \cdot k_m}{R} \quad (23)$$

Es posible también determinar la corriente de arranque y torque de fricción empleando una fuente de alimentación de voltaje DC variable, se aplica la tensión gradualmente en la armadura del motor DC y se determina la corriente necesaria, con la cual el rotor del motor comienza a girar.

Con la corriente de arranque se determina la constante de torque provocada por la fricción en el eje, sin embargo, no se puedo realizar esta prueba, debido a que no se tiene disponible el instrumento que me permita regular dicha tensión.

$$T_f = k_m i_{arr} (24)$$

Finalmente se determina la constante de fricción de Coulomb, la cual puede ser obtenida de forma rápida aplicando la siguiente ecuación.

$$B = \frac{T_m}{\omega(t)} (25)$$

La velocidad empleada es la alcanzada en estado estacionario y se puede utilizar la constante de tiempo mecánica que se halló en el anterior experimento.

2.3.3 Adquisición de datos entre Simulink y Arduino

Se procedió a realizar la conexión serial entre Arduino y Simulink utilizando "Instrument Control Toolbox", el objetivo es enviar las RPM del Arduino y visualizarlos en Simulink. En el Arduino se ejecutó el algoritmo que me permite obtener las RPM a partir del tiempo entre pulsos que detecta el sensor de efecto Hall. En Simulink se realizó las siguientes configuraciones.

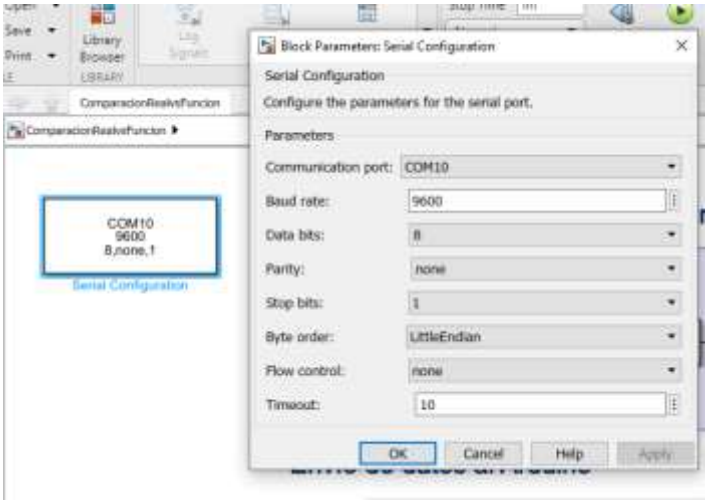


Figura 2.43 Configuración del bloque serial. Autor

Se estableció la conexión por el puerto "COM10", se estableció un "baud rate" de 9600 tanto como para el microcontrolador y la configuración del bloque. Los datos por transmitir o recibir serán de 8 bits, sin bit de paridad y con un tiempo máximo de 10 segundos para realizar la conexión.

Luego de establecer esta configuración se utiliza el bloque "Serial Receive", el cual se encarga de receptar datos desde el Arduino a Simulink. Los datos enviados por el Arduino son tipo bytes, con un encabezado para detectar el inicio del dato y con un salto de línea para detectar el final del dato.

Los datos enviados por el Arduino tienen formato “Single” el cual tiene un tamaño de 4 bytes, sin embargo, Simulink solo trabaja con datos tipo “Double” por lo tanto se debe realizar este casteo en el diagrama de bloques.

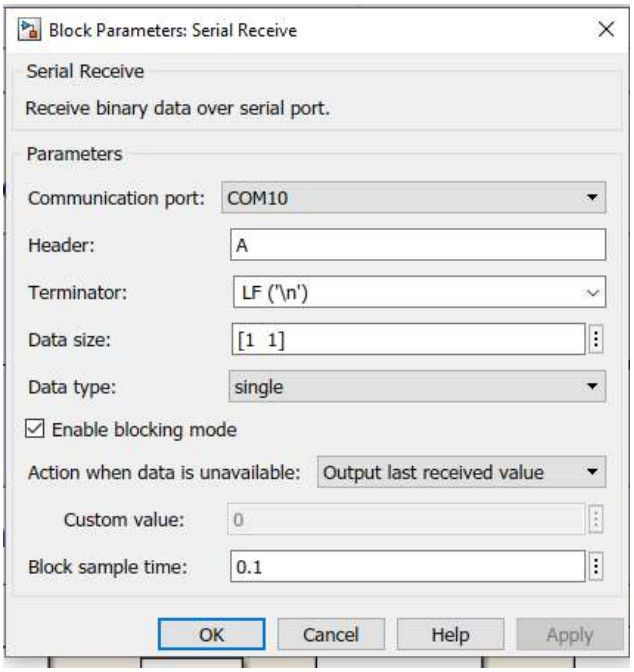


Figura 2.44 Configuración del bloque “Serial Receive”. Autor

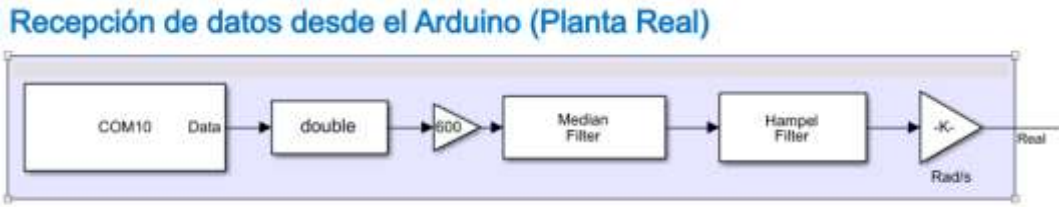


Figura 2.45 Diagrama de bloques parala recepción de datos desde el Arduino. Autor

Al igual que se lee datos enviados por el Arduino, se enviará la velocidad en PWM desde Simulink al Arduino, por lo tanto, se empleó el bloque “Serial send” para esta tarea, las configuraciones se presentan en las **Figuras 2-37 y 2-38.**

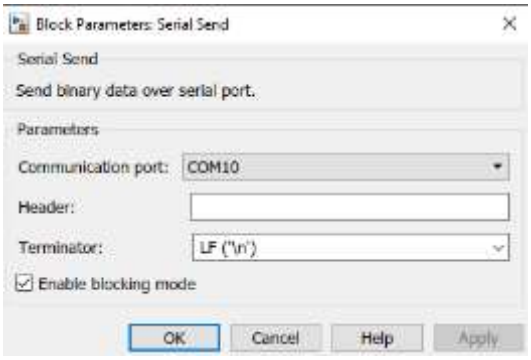


Figura 2.46 Configuración bloque "Serial send". Autor

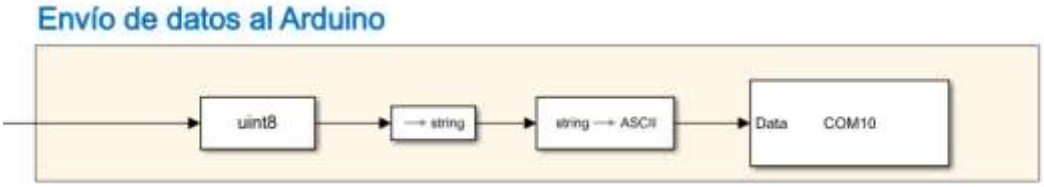


Figura 2.47 Diagrama de bloques para el envío de datos al Arduino. Autor

Se envían los datos en formato tipo ASCII para que el Arduino luego los castee internamente a formato tipo entero y posteriormente se realiza el mapeo de 0-255 para obtener el PWM deseado.

Los parámetros experimentales obtenidos fueron puestos a prueba con la función escalón, donde se compara la función de transferencia y fenomenológica cargados con dichos parámetros.

2.4 Control de Velocidad de un motor DC basado en un algoritmo de aprendizaje por refuerzo

En este proyecto se muestra cómo sintonizar un controlador PI utilizando el algoritmo de aprendizaje por refuerzo de política determinista profunda de doble retardo (TD3). El rendimiento del controlador ajustado se compara con el de un controlador ajustado con la aplicación Sisotool de MATLAB.

Para tareas de control relativamente sencillas con un pequeño número de parámetros sintonizables, las técnicas de sintonización basadas en modelos como: Sisotool, PID Tuner, etc., pueden obtener buenos resultados con un proceso de sintonización más rápido en comparación con los métodos basados en RL sin modelos. Sin embargo, los métodos RL pueden ser más adecuados para sistemas altamente no lineales o para la sintonización de controladores

adaptativos, los cuales cambian según las modificaciones en el sistema como es el caso de mayor carga en el eje.

Para facilitar la comparación de los controladores, ambos métodos de ajuste utilizan una función objetivo lineal cuadrática gaussiana (LQG).

Este proyecto utiliza un agente de aprendizaje por refuerzo (RL) para calcular las ganancias de un controlador PI.

2.4.1 Requisitos mínimos del software MATLAB

Para realizar este proyecto, se utilizó la versión de MATLAB 9.11 R2021b, los cuales considera los siguientes toolbox:

- Simulink versión 10.4 (R2021b).
- Deep Learning versión 14.3 (R2021b).
- Reinforcement Learning versión 2.1 (R2021b).

Además, que, desde MATLAB es posible ajustar el controlador PID en base a técnicas de control por medio de: Sisotool, PID Tuner, Control System Tuner, etc.

2.4.2 Modelo del Entorno

El modelo de entorno para este proyecto es un modelo de motor DC. El objetivo de este sistema de control es mantener la velocidad del motor para que coincida con un valor de referencia. El modelo matemático del sistema antes mencionado, permitió modelarlo en Simulink y así obtener la entrada de voltaje y la salida en la velocidad RPM.

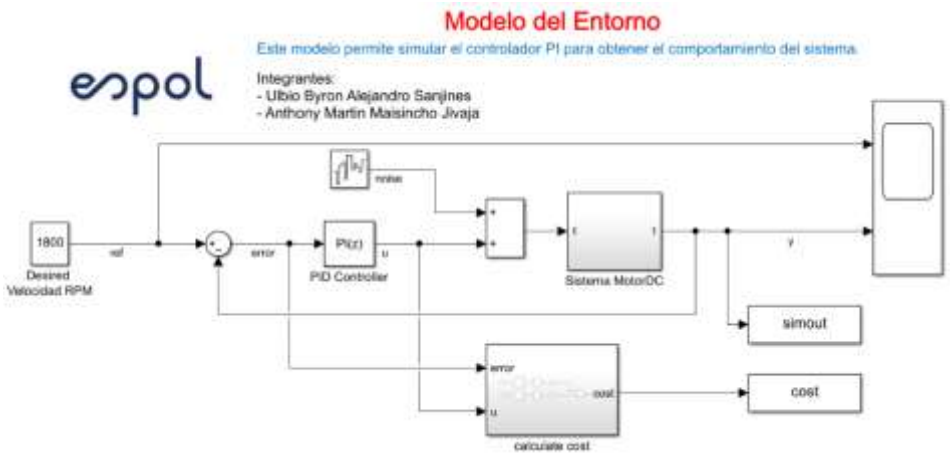


Figura 2.48 Modelo de Simulink del entorno para el sistema de control. Autor.

En el modelo, se puede observar que contiene el bloque PID el cual se ajusta las constante PI definidas por el usuario o del algoritmo RL. Además, contiene un ruido en la señal de entrada que se suma con la de acción por parte del

controlador, con el fin de que con o sin perturbaciones, el sistema tenga un comportamiento estable, el cual es definido con varianza de:

$$E\left(n^2(t)\right) = 1 \tag{25}$$

Dicho ruido se lo define como Ruido Blanco y es representada como un proceso estocástico, es decir no guarda ninguna correlación estadística conforme a su comportamiento.

Para el control óptimo en este sistema se aplica el criterio LQG, el cual permite el análisis de los costos con respecto a cuanto esfuerzo le toma la señal de control en llegar al punto en estado estable ajustado por el usuario, en la operación dinámica de la planta controlada. Lo cual se define a continuación:

$$J = \int_{T0}^{Tf} e(t)dt \tag{26}$$

Donde:

- J: Función costo.
- E(t): Error conforme transcurre el tiempo de simulación.

La ecuación (26), representa la función costo en el sistema, además que el error corresponde a la ecuación (27).

$$e(t) = r - v(t) \tag{27}$$

Donde:

- r: Referencia de velocidad deseada.
- v(t): Velocidad variable conforme transcurre el tiempo (velocidad del sistema).

La ecuación de costos es representa por índices cuadrados:

$$J = \int_{To}^{Tf} e(t)^2 dt \tag{28}$$

Como dicha función, no solo depende de la salida del sistema, sino de la entrada, entonces se rige a la señal de acción de control, dando así la siguiente ecuación:

$$J = \int_{To}^{Tf} (e(t)^2 - pu(t)^2)dt \tag{29}$$

Donde:

p: Ponderación de la entrada.

u(t): Señal de acción variante en el tiempo del control.

La ponderación en la entrada es que nos permita que cuando existan señales de control elevadas y el error es mínimo, entonces no afecte negativamente con el

aprendizaje y el análisis de costos en el sistema. Entonces, para mantener la velocidad del motor minimizando el esfuerzo de control u, los controladores de este proyecto utilizaron el siguiente criterio LQG:

$$J = \lim_{T\to\infty} E\left(\frac{1}{T}\int_0^T ((r-v)^2(t) + 0.01u^2(t))dt\right) \quad (30)$$

Donde:

- Se utilizó una ponderación de 0.01 para las acciones de la señal de control.
- T: Tiempo de simulación en la función de costo.

Los tiempos utilizados para el muestre es de 0.1 segundos y el tiempo de simulación corresponde a 10 segundos.

2.4.3 Sintonización controlador PI utilizando Sisotool Toolbox de MATLAB

Una vez obtenida la función de transferencia del sistema y ubicarla en el código de MATLAB, se procede ajustar el controlador mediante el comando *sisotool(G)*, donde G representa a ficha función. Se ejecute y mediante su interfaz se ajusta las ganancias del controlador, del cual se debe agregar en primera instancia un integrador, para que la señal de salida en el sistema sea muy aproximada a cero.

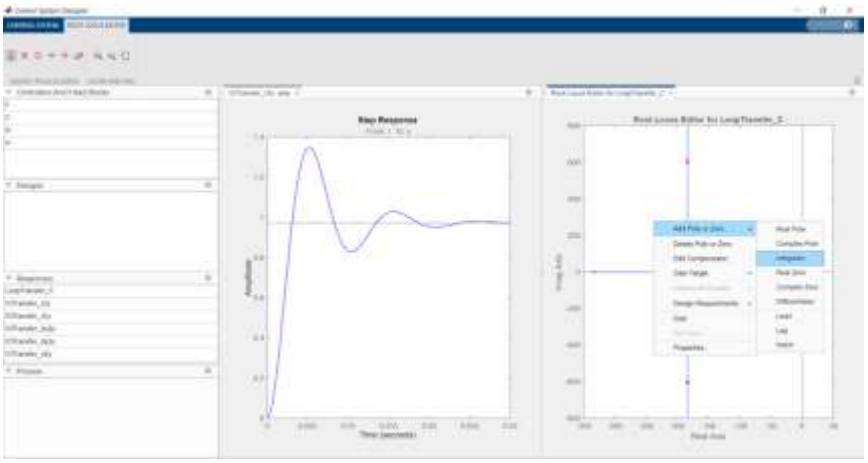


Figura 2.49 Agregar un integrador en la función de transferencia. Autor.

Para ajustar el controlador, se recomienda agregar un integrador para así obtener un error de cero en el estado estacionario. Es posible indicar los diseños de los requerimientos en el sistema de control, para ello se debe dirigir en el editor de localización de raíces en la planta, clic derecho y colocar dichas etiquetas donde resaltan: tiempo en estado estable, sobre nivel porcentual, frecuencia natural, radio de amortiguación y limitación de la región.

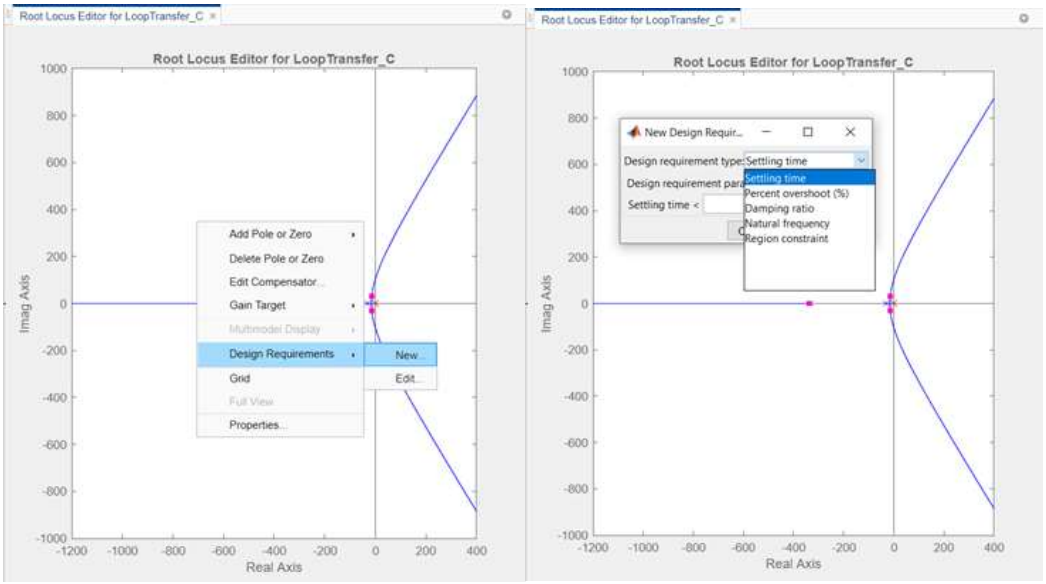


Figura 2.50 Ajuste de los requerimientos en el diseño de control. Autor.

Una vez indicado algún requerimiento en el diseño, se van agregando ceros o polos según corresponda al controlador, el cual se identifica con la variable C.

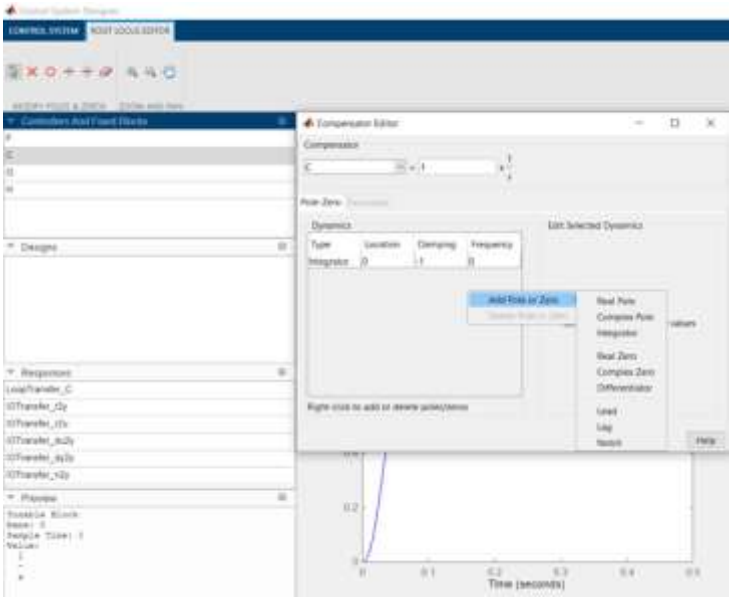


Figura 2.51 Agregar polos y ceros en el controlador. Autor.

Para observar cómo esto se encuentra afectando al sistema en la respuesta escalón unitario en lazo cerrado, se activaron las características dando clic derecho y seleccionado todas las variables de interés.

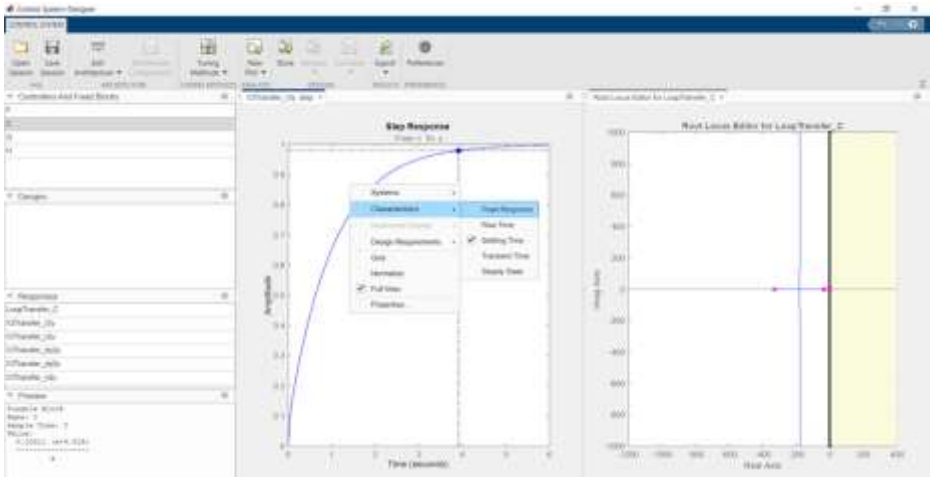


Figura 2.52 Agregando características en la respuesta escalón. Autor.

Una vez definida la respuesta deseada, se procede a exportar el controlador, navegando en la pestaña superior denominada *export*:

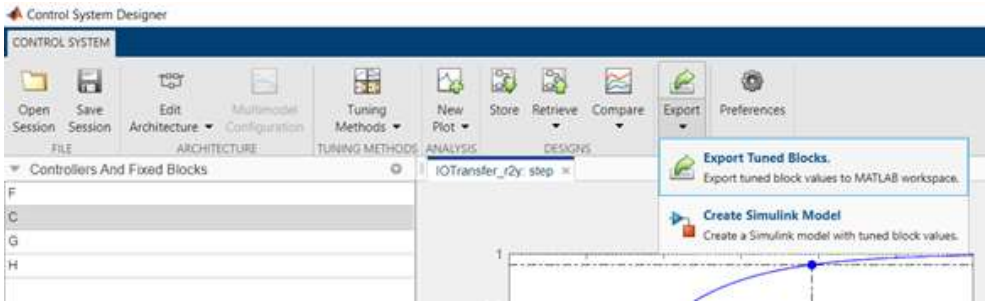


Figura 2.53 Opción para exportar las funciones diseñadas en Sisotool. Autor.

Finalmente se define un nombre y se ubica en el espacio de trabajo de MATLAB.

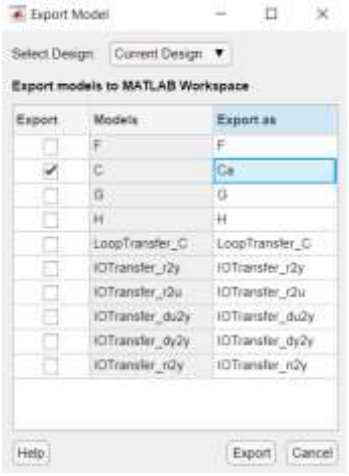


Figura 2.54 Definir el nombre de la variable a exportar. Autor.

Este modelo de Sisotool es posible acceder en el archivo denominado SISOcontrol.mat, para que el usuario pueda verificar y mejorar las ganancias en el controlador. Con el siguiente comando load('SISOcontrol.mat'), es posible cargar dichas configuraciones.

En el código, se escribe la función de transferencia del controlador encontrada para obtener así las ganancias del mismo.

$$Controlador = K_p + K_i \frac{1}{s} \tag{31}$$

Donde:

- K_p: Ganancia proporcional del controlador.
- K_i: Ganancia integral del controlador.
- s: Dominio de la frecuencia.

Teniendo los siguientes resultados:

$$K_p = 0.00311 \tag{32}$$

$$K_i = 0.0299 \tag{33}$$

2.4.4 Creación del entorno para entrenamiento del Agente RL

Para definir el modelo entrenamiento del agente RL, se modifica el *Simulink* que contiene el controlador PID del motor DC con los siguientes pasos:

- Eliminar el PID Controller.
- Insertar un bloque de RL Agent.
- Crear el vector de observación $[\int e\, dt \quad e]^T$ donde $e = r - v$, v es la velocidad del motor DC, y r es la referencia de velocidad. Conectar la señal del observador en el bloque de RL Agent.
- Definir la función de recompensa para el agente RL como el negativo de los costos del criterio LQG, esto es, $Recompensa = -\big((ref - v)^2(t) + 0.01u^2(t)\big)$. El agente RL maximiza esta recompensa, minimizando así el coste LQG.

El resultado del modelo es *rlmotorDCPIDTune.slx* como archivo de simulación encontrado en el GitHub adjunto.

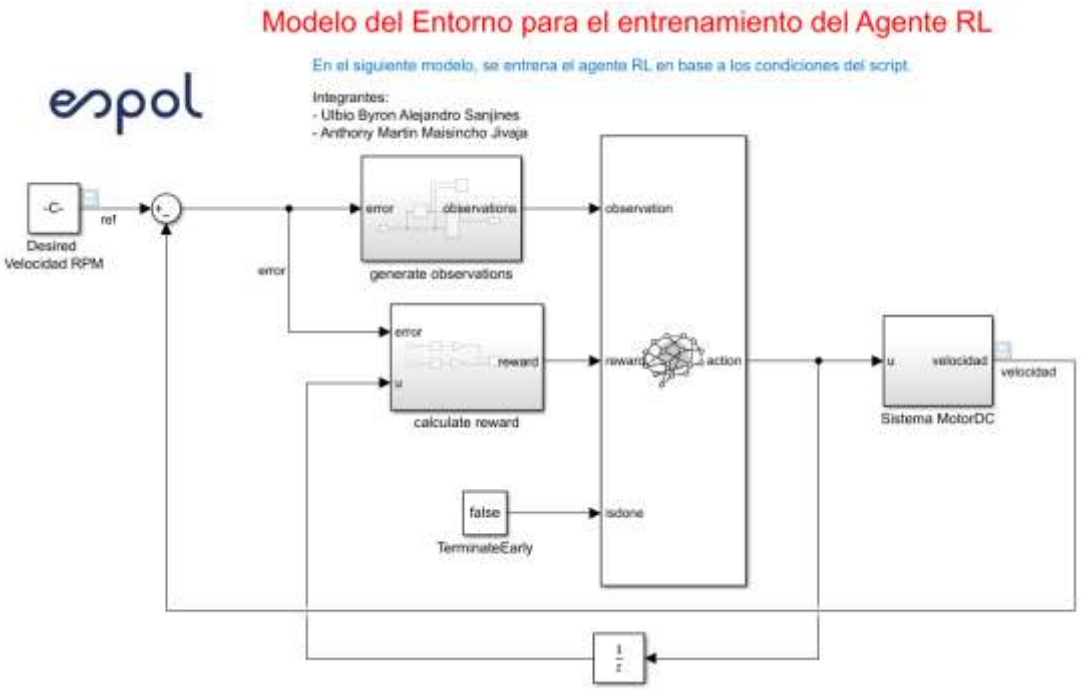


Figura 2.55 Modelo de entrenamiento del agente RL. Autor.

Se creó la interfaz del entorno, para ello se utilizó la función *localCreatePIDEnv* definida en el script de MATLAB, extrayendo así las dimensiones de observación y acción de este. El generador aleatorio de reproducibilidad es el tipo *Twister*, esencial para variar el valor de referencia en el aprendizaje del modelo.

2.4.5 Creación del Agente TD3

Dadas las observaciones, un agente TD3 decide qué acción realizar utilizando una representación de actor. Para crear el actor, primero hay que crear una red neuronal profunda con la entrada de la observación y la salida de la acción.

Se puede modelar un controlador PI como una red neuronal con una capa totalmente conectada con observaciones de error e integral de error.

$$u = \left[\int e\,dt \quad e\right] * [K_i \quad K_p]^T \tag{34}$$

Donde:

- u: Salida de la red neuronal del actor.
- K_i y K_p: Valores absolutos de los pesos en la red neuronal.
- $e = r - v$: v es la velocidad del motor DC, y r es la señal de referencia de velocidad deseada.

La optimización del descenso gradual puede llevar los pesos a valores negativos. Para evitar los pesos negativos, se sustituyó la capa totalmente conectada *fullyConnectedLayer* normal por una *fullyConnectedPILayer*. Esta capa asegura

que los pesos sean positivos implementados en la función $Y = abs(WEIGHTS) * X$, dicha capa se definió como *fullyConnectedPILayer.m*.

Un agente TD3 aproxima la recompensa a largo plazo dadas las observaciones y las acciones utilizando dos representaciones de función de valor crítico. Para crear los críticos, primero hay que crear una red neuronal profunda con dos entradas, la observación y la acción, y una salida.

Para crear los críticos, se utilizó la función *localCreateCriticNetwork* definida al final del script de MATLAB. Se utilizaron la misma estructura de red para ambas representaciones de críticos. A continuación, se presenta la red neuronal del crítico.

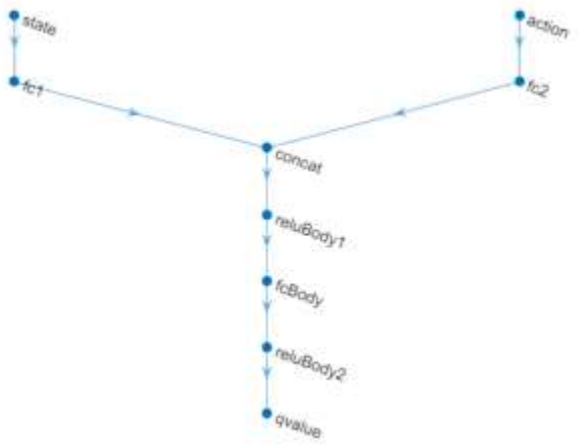


Figura 2.56 Red neuronal del crítico. Autor.

Se utilizan dos críticos debido a que cada uno, podrá aprender sobre las ganancias tanto de la constante proporcional e integral individualmente, ya que son valores muy distintos en el ajuste de sus parámetros y deben tener su propio aprendizaje en el sistema.

Además, se debe configurar el agente que para este proyecto se consideró lo siguiente:

- Configuración del agente para utilizar el tiempo de muestreo del controlador Ts.
- Ajustar el tamaño mini-batch a 128 muestras de experiencia.
- Establecer la longitud del buffer de experiencia en 1e6.
- Establecer el modelo de exploración y el modelo de suavización de la política de objetivos para utilizar el ruido gaussiano con varianza 0,1.

Es posible utilizar las funciones propuestas por MATLAB como es rlTD3AgentOptions, el cual configura las opciones del agente TD3. Mientras que

rlTD3Agent permite crear el agente TD3 utilizando la representación del actor, la representación del crítico y las opciones del agente propuestas anteriormente.

2.4.6 Opciones para el entrenamiento del Agente

Para entrenar el agente, primero se especifican las opciones del entrenamiento:

- Se ejecutó cada entrenamiento durante un máximo de 1000 episodios, con una duración de cada episodio de un máximo de 100 pasos de tiempo.
- Mostrar el progreso del entrenamiento en el Gestor de Episodios (establecer la opción Plots) y desactivar la visualización de la línea de comandos (establecer la opción Verbose).
- Se detiene el entrenamiento cuando el agente reciba una recompensa media acumulada superior a -355 en 100 episodios consecutivos. En este punto, el agente puede controlar la velocidad del motor DC.

Todas estas opciones permiten obtener una gráfica de mejor análisis con respecto a los tiempos de entrenamiento, las cuales contienen señales tanto del promedio como del objetivo Q, esta última indica las experiencias del entrenamiento que en cada episodio intentar direccionar una ganancia del controlador PI.

Todos los archivos generados en la metodología como parte del uso del tablero o el entrenamiento del aprendizaje por refuerzo se encuentran en el GitHub compartidos, para que cualquier usuario tenga acceso y pueda realizar modificaciones o incluso realizar pruebas del proyecto de tesis. El enlace de dichos archivos se encuentra en anexo.

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS

3.1 Parámetros del motor.

Como se había mencionado en la metodología, los parámetros del motor son sumamente importantes para realizar comparaciones de la dinámica del sistema. En la siguiente tabla se presentan los resultados de dicho experimento.

Tabla 5 Parámetros del motor DC.

Parámetros de Motor DC de Imán Permanente			
Parámetro	**Símbolo**	**Valor**	**Unidad**
Momento de Inercia	J	4,971E-10	Kg.m^2
Constante de Fricción viscosa	B	1,9562E-05	N.m.s
Constante de Fuerza Electromotriz	Ka	0,02176438	V/rad s
Constante del Par del Motor	Km	0,02176438	N.m/A
Resistencia de Armadura	R	1,234	Ohms
Inductancia eléctrica	L	0,0004671	H

La comparación que se realizó fue entre la planta real que está embebida en el tablero de control y las funciones de transferencia que dependen de los parámetros hallados en la **Tabla 5.**

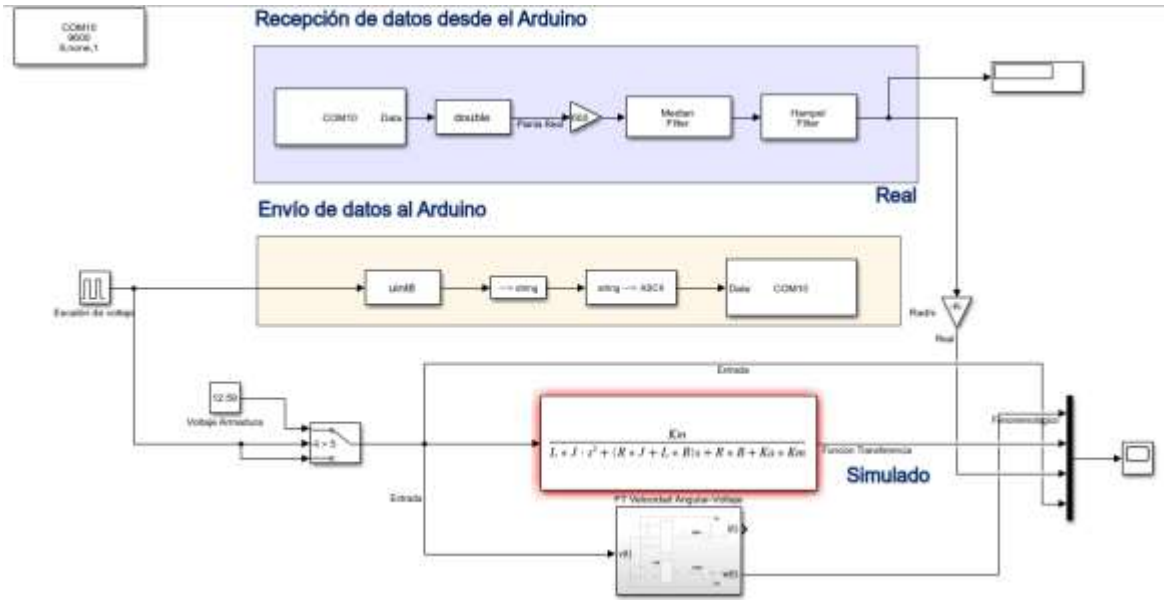


Figura 3.1 Modelo Simulink para la comparación Real vs Simulado. Autor

En la Figura 3.1 se tiene la planta real, la cual corresponde a la adquisición de datos entre Simulink y Arduino, y la planta simulada utiliza una función de transferencia o un modelo fenomenológico basado en los parámetros del motor DC.

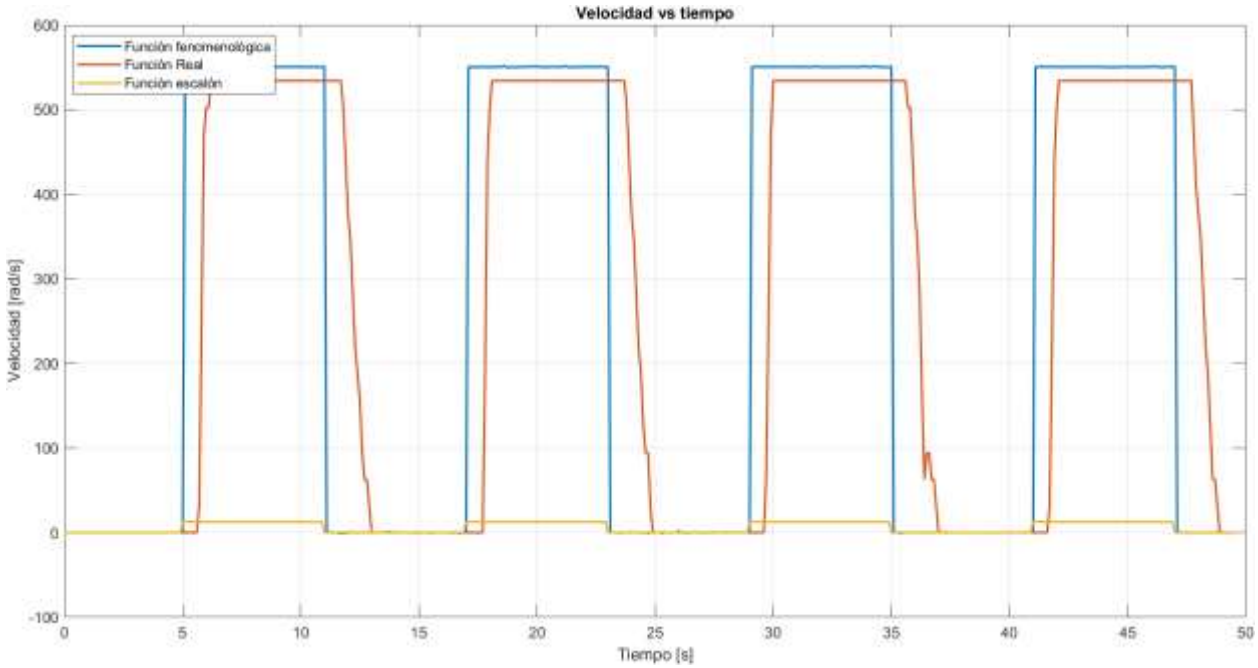


Figura 3.2 Gráfica Función Real, Fenomenológica y escalón vs tiempo. Autor

La gráfica naranja representa a la función real mientras que la azul a la del modelo fenomenológico, notamos un cierto desfase al momento de responder a la entrada escalón, esto debe a dos factores.

El primero es el tiempo de muestreo empleado, que en este caso fue de 100 [ms], teniendo en cuenta que el motor en su punto de operación alcanza las 5000 RPM, se pierden datos, comprometiendo a la gráfica del transiente.

El segundo factor corresponde a los recursos computacionales empleados por Simulink, los cuales guardan relación directa con el tiempo de muestreo.

Es aceptable este retardo que se presenta en el transiente por todas las justificaciones presentadas, sin embargo, no fue posible realizar esta prueba con un menor tiempo de muestreo para obtener una mejor respuesta debido a las limitaciones técnicas del CPU empleado.

Sistema Operativo	Windows 10 Pro-64 bits
Procesador	AMD Ryzen 5 3400G
Núcleos del procesador	8
Frecuencia del procesador	3.7 [Ghz]
Tarjeta de Vídeo	AMD Radeon RX Vega 11 Graphics
Memoria VRAM	2032 Mb
Memoria RAM	16384 Mb
Disco Duro	256 GB Tipo SDD

Tabla 6 Especificaciones técnicas del computador empleado para la adquisición de datos.

Con respecto al valor en estado estacionario de la velocidad, notamos una gran diferencia entre el real y simulado, por lo que se puede decir que estos parámetros hallados no representan de manera adecuada a la dinámica del motor.

Se procedió a optimizar los parámetros obtenidos mediante un modelo de regresión, este modelo utiliza una función objetivo la cual presenta la siguiente ecuación.

$$\sum_{i=0}^{i=m} (X_{i\_} Y_i)^2 \tag{35}$$

Donde:

m: Número total de muestras

i: Muestra actual

X: Velocidad del modelo de función de transferencia

Y: Velocidad real de la adquisición de datos

Se obtuvo un valor alto de la función objetivo antes de la optimización, posteriormente, aplicando el modelo, se pudo reducir dicho valor como se muestra en la ***Figura 3.3***.

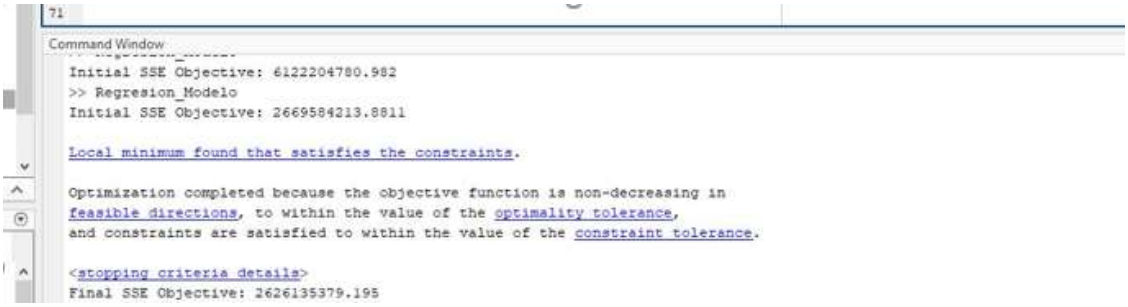


Figura 3.3 Resultados iniciales y finales de la función objetivo del modelo de regresión. Autor

El algoritmo nos entregó los siguientes parámetros optimizados:

Parámetros de Motor DC de Imán Permanente			
Parámetro	**Símbolo**	**Valor**	**Unidad**
Momento de Inercia	J	4,81E-4	Kg.m^2
Constante de Fricción viscosa	B	2,26E-03	N.m.s
Constante de Fuerza Electromotriz	Ka	0,2276438	V/rad s
Constante del Par del Motor	Km	0,2276438	N.m/A
Resistencia de Armadura	R	4,08	Ohms
Inductancia eléctrica	L	0,011307	H

Tabla 7 Parámetros del motor DC optimizados.

Con estos parámetros se realizará el entrenamiento del agente DDPG TD3 ya que son los que mejor representan al motor DC de la planta real.

3.2 Agente TD3 Entrenado

Se entrena el agente utilizando la función de MATLAB *train* como parte de la librería de *Reinforcement Learning Toolbox*. El entrenamiento de este agente es un proceso computacionalmente intensivo que tarda varios minutos en completarse (incluso horas). Para ahorrar tiempo al ejecutar este proyecto, es posible utilizar un agente pre-entrenado cargándolo en el espacio de trabajo de MATLAB y así podemos obtener los pesos del controlador PI.

Dicho entrenamiento se lo realizó en una laptop, la cual contiene las siguientes características técnicas de software.

Sistema Operativo	**Windows 10 Pro-64 bits**
Procesador	AMD Ryzen 7 3700U
Núcleos del procesador	8
Frecuencia del procesador	2.3 [Ghz]

Tarjeta de Vídeo	Radeon Vega Mobile Gfx
Memoria VRAM	2034 Mb
Memoria RAM	16384 Mb
Disco Duro	256 GB Tipo SDD + 1 TB Tipo HDD

Tabla 8 Características de la computadora para el entrenamiento del RL.

Al realizar este proyecto, los gráficos que representan continuamente el aprendizaje según el número de episodios es el siguiente.

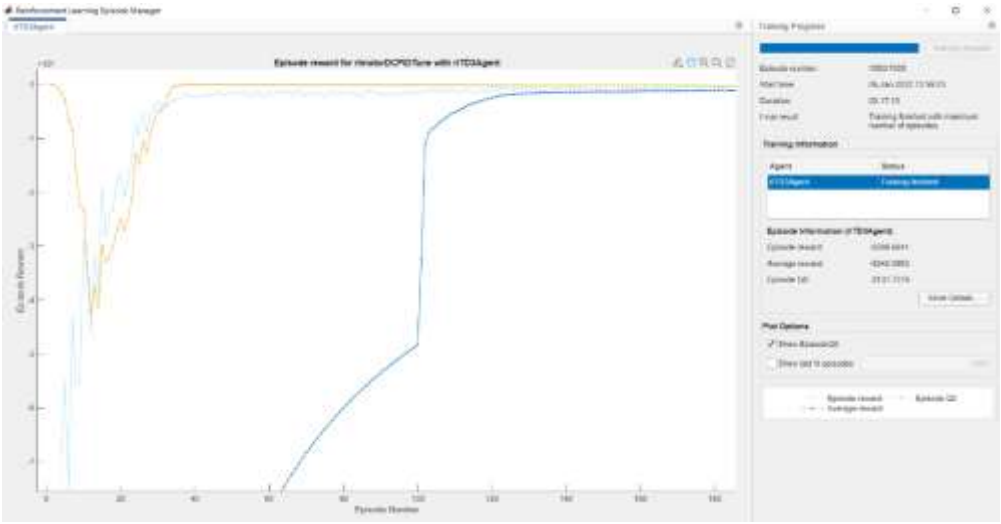


Figura 3.4 Episodios de recompensa del entrenamiento del agente RL. Autor.

En la interfaz del aprendizaje por refuerzo resalta tres tipos de gráficas que son: Episodio de la recompensa, Promedio de la Recompensa y el Episodio Q0. La curva de color celeste son los episodios de la recompensa, que determina si las acciones tomadas por el entorno son las más eficientes para que sean actuadas en el sistema y así tener la referencia deseada de velocidad, dando pequeñas valoraciones en caso de que llegue a su objetivo, por lo contrario, recibirá castigos las cuales le indica al agente RL que sus acciones no son las correctas y debe realizar distintas operaciones. Se evalúa el comportamiento de las recompensas mediante el promedio a través del número de episodios, la cual destaca del color azul.

Mientras que la gráfica de los episodios Q0 es el progreso de aprendizaje consecutivo que realiza las acciones del agente, para que el crítico aprenda del sistema, este crítico es el que define cuales son las ganancias del controlador para ser ajustados en el Control PID Adaptativo. El tiempo de entrenamiento fue de 3h:17m:10s con 1000 episodios de simulación, donde se considera que el agente se encuentra totalmente entrenado y los pesos de los críticos resultan los valores de las

ganancias del controlador PI, que son comparados con las ganancias del controlador ajustados por el usuario previamente, obteniendo lo siguiente:

	Control Sisotool Tuner	Agente RL
Constante proporcional	0.0031	0.0061
Constante integral	0.0299	0.0525

Tabla 9 Constantes del controlador PI. Autor.

Luego, con el primer modelo de Simulink, se realiza la simulación de los dos controladores para ser comparados entre ellos.

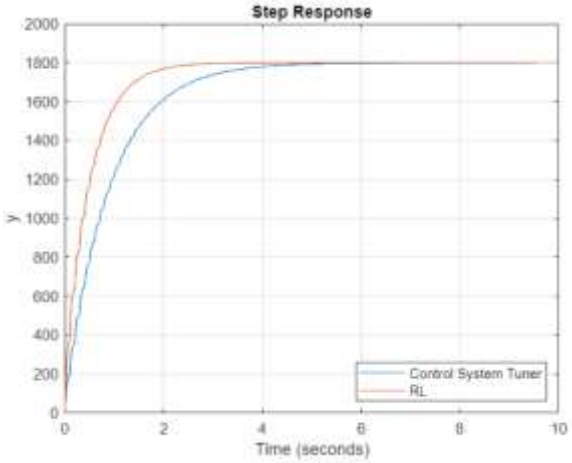


Figura 3.5 Respuesta escalón sin considerar Ruido blanco. Autor.

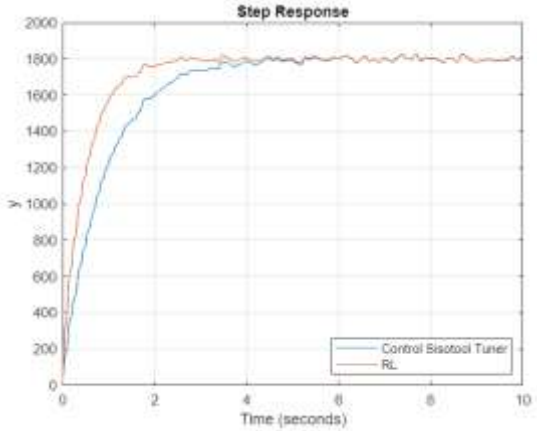


Figura 3.6 Respuesta escalón considerando Ruido Blanco. Autor.

En la gráfica anterior presenta los resultados sobre las constantes ajustadas por el usuario y las obtenidas por medio del algoritmo de RL. El sistema contiene un ruido blanco, por ese motivo es que existen pequeñas variaciones durante su recorrido en la estabilización que es el punto de referencia de 1800 RPM. Gráficamente se observa que el sistema tiene un comportamiento sobre amortiguado y que el algoritmo RL tiene una mejor respuesta que la ajustada por

el usuario, debido a que tiene un tiempo de estabilización menor y por tanto la constante de amortiguación es menor. Las respuestas en estado estable son muy similares y cumplen que las dos llegan a su objetivo que es el punto de referencia deseado.

	Tiempo pico	Tiempo transiente	Tiempo en estado estable	Sobrenivel porcentual	Valor pico
Control Sisotool Tuner	1.9552	3.5020	3.5020	0.0016	1.8000e+3
Agente RL	1.0973	1.9246	1.9246	0.0017	1.8000e+3

Tabla 10 Respuesta del sistema sin considerar ruido gaussiano.

	Tiempo pico	Tiempo transiente	Tiempo en estado estable	Sobrenivel porcentual	Valor pico
Control Sisotool Tuner	2.0450	5.2015	5.2015	1.2945	1.8314e+3
Agente RL	1.1064	2.1700	2.1700	1.1825	1.8310e+3

Tabla 11 Respuesta del sistema considerando ruido gaussiano.

En la tabla anterior, se especifican cuáles son las características del sistema obtenidas con la simulación en lazo cerrado de las constantes PI, las obtenidas por el Sisotool y del algoritmo RL. Como se indicó gráficamente, el algoritmo RL presenta un mejor desempeño en la respuesta obtenida ya que le toma menor tiempo en llegar al punto referencia de velocidad.

Sin considerar el ruido, podemos observar que su valor en estado estable tiende a llegar a la velocidad ajustada por el usuario. Se cumple que el tiempo en estado

estable resulta mejor con el algoritmo RL, teniendo así mejores variaciones entre la señal de acción, mientras que su sobre nivel porcentual son muy similares.

Con respuesta al ruido blanco, se obtuvo margen de alrededor 3.0315 segundos de diferencia, de igual forma en el tiempo pico el cual existe un margen de 0.9386 segundos. Mientras que las perturbaciones por el ruido blanco incorporado, hace que el sistema presente ciertos picos que conducen a un sobre nivel porcentual, el mismo que es inferior con 1.1825% por parte del algoritmo óptimo. Finalmente, los valores máximos de picos en el sistema resultan ser iguales.

	Con Ruido Blanco	**Sin Ruido Blanco**
Costos con Sisotool Tuner	-1.6170e+7	-1.6201e+7
Costos con RL	-9.4182e+6	-9.4434e+6

Tabla 12 Costos del criterio LQG. Autor.

Ambos controladores producen respuestas estables, mientras que el controlador ajustado por Sisotool produce una respuesta más lenta. Sin embargo, el método de sintonización RL produce un mayor margen de ganancia y una solución óptima, con el tiempo menor teniendo así una velocidad deseada mucho más rápida. Las ganancias encontradas por el algoritmo RL le toma un inferior esfuerzo en comparación con las ganancias ajustadas en Sisotool, esto podemos notarlo en los valores de costos que resultan más cercanos a cero que con la propuesta por el usuario, tanto con o sin ruido blanco.

3.3 Agente TD3 entrenado con distintos episodios

Con el fin de observar la efectividad de las ganancias del controlador PI conformen transcurren cierta cantidad de episodios, se evalúa su efectividad teniendo distintas simulaciones, las cuales se detalla el desempeño de cada una de ellas.

3.3.1 Agente TD3 entrenado con 20 episodios

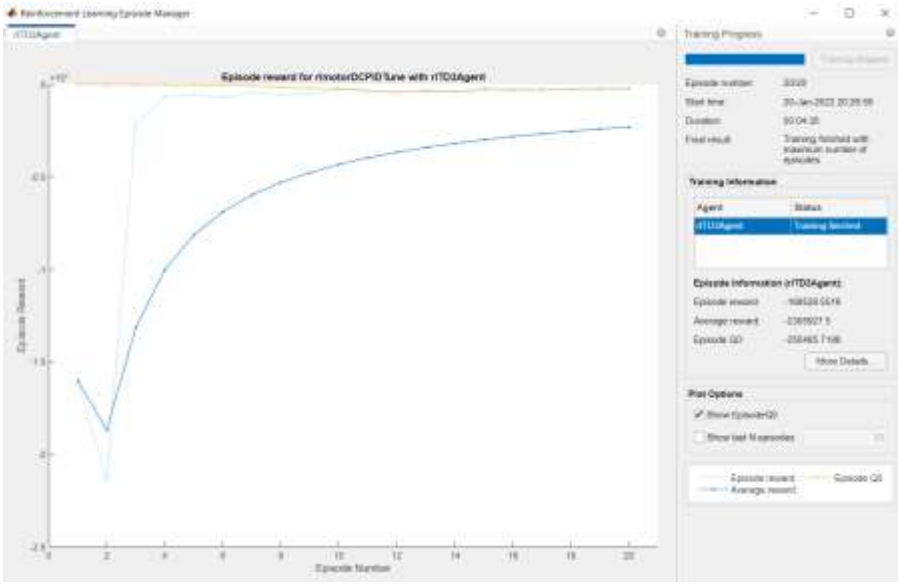


Figura 3.7 20-Episodios de recompensa del entrenamiento del agente RL. Autor.

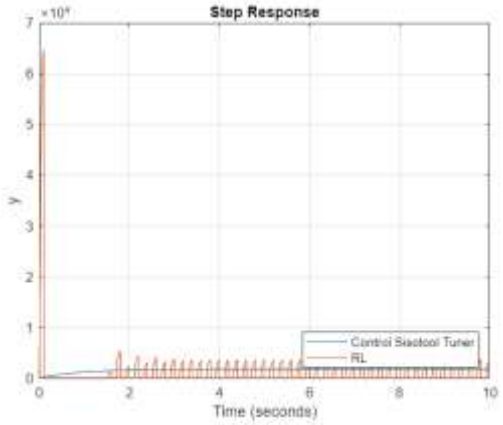


Figura 3.8 Respuesta escalón considerando ruido blanco y 20 episodios de entrenamiento. Autor.

	Control Sisotool Tuner	Agente RL
Constante proporcional	0.0031	1.0642
Constante integral	0.0299	0.4973

Tabla 13 Constantes del controlador con 20 episodios de entrenamiento. Autor.

	Tiempo pico	Tiempo transiente	Tiempo en estado estable	Sobrenivel porcentual	Valor pico
Control Sisotool Tuner	1.9552	3.5020	3.5020	0.0016	1.8000e+3
Agente RL	1.0973	1.9246	1.9246	0.0017	1.8000e+3

Tabla 14 Características del sistema con 20 episodios de entrenamiento. Autor.

Función costo criterio LQG	
Costos RL	-4.3595e+9
Costos Sisotool Tuner	-1.617e+7

Tabla 15 Costos con 20 episodios de entrenamiento. Autor.

El entramiento con 20 episodios resultó ineficiente, el mismo presenta un sistema inestable y las características del sistema no se acercan a las que fueron requeridas por el usuario cuando estas fueron ajustadas en Sisotool. El diagrama de los episodios resultantes conforma muy pocos episodios para generar un aprendizaje completo por el agente, se observa que las recompensas están convergiendo a los episodios Q0.

3.3.2 Agente TD3 entrenado con 60 episodios

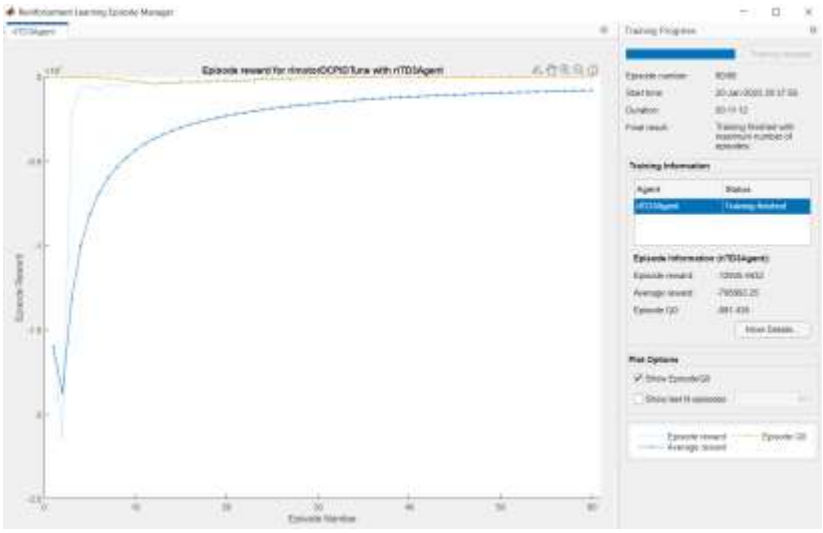


Figura 3.9 60-Episodios de recompensa del entrenamiento del agente RL. Autor.

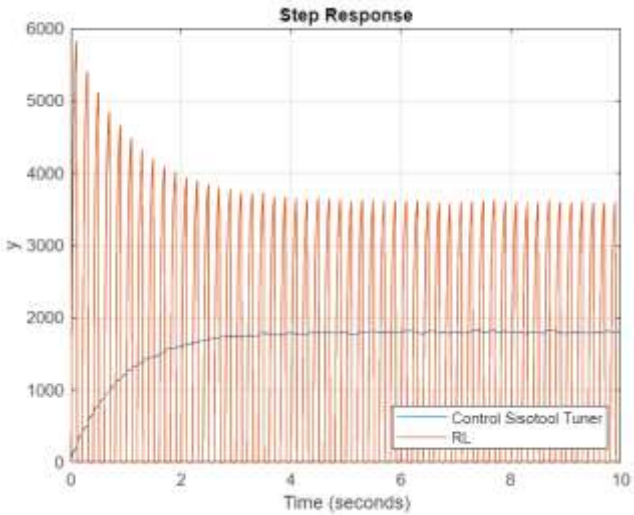


Figura 3.10 Respuesta escalón considerando ruido blanco y 60 episodios de entrenamiento. Autor.

	Control Sisotool Tuner	Agente RL
Constante proporcional	0.0031	0.0955
Constante integral	0.0299	0.0497

Tabla 16 Constantes del controlador con 60 episodios de entrenamiento. Autor.

	Tiempo pico	Tiempo transiente	Tiempo en estado estable	Sobrenivel porcentual	Valor pico
Control Sisotool Tuner	2.0450	5.2015	5.2015	1.2945	1.8314e+3
Agente RL	0	9.9140	9.9144	inf	5.8221e+3

Tabla 17 Características del sistema con 60 episodios de entrenamiento. Autor.

Función costo criterio LQG	
Costos RL	-3.9003e+9
Costos Sisotool Tuner	-1.617e+7

Tabla 18 Costos con 60 episodios de entrenamiento. Autor.

El sistema continua siendo inestable, las caracsteristicas técnicas no coinciden y se puede interpretar que las constantes son mejor ajustadas por Sisotool que el agente entrenado. Además, los episodios de la recompensa le tomo un tiempo de 11m:12s para llegar a los 60 episodios, sin embargo las recompensas y los episodios Q0 no son suficientes para obtener los pesos de los críticos.

3.3.3 Agente TD3 entrenado con 120 episodios

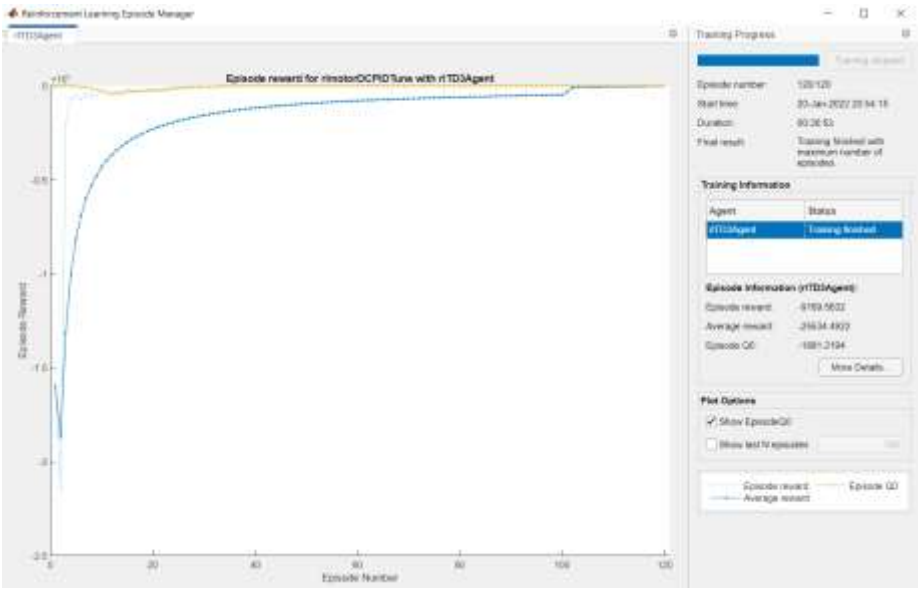


Figura 3.11 120-Episodios de recompensa del entrenamiento del agente RL. Autor.

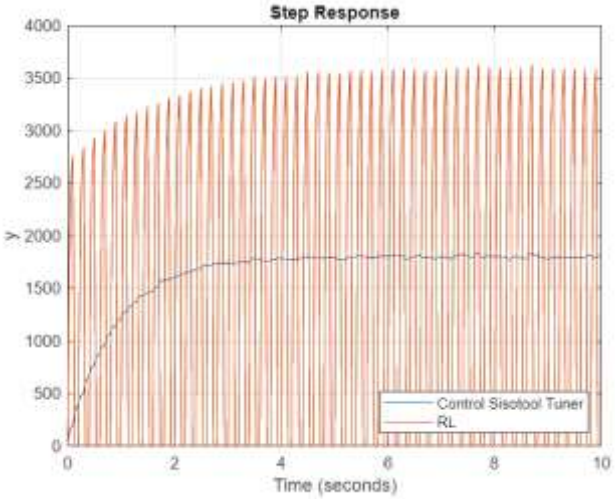


Figura 3.12 Respuesta escalón considerando ruido blanco y 120 episodios de entrenamiento. Autor.

	Control Sisotool Tuner	Agente RL
Constante proporcional	0.0031	0.0452
Constante integral	0.0299	0.0334

Tabla 19 Constantes del controlador con 120 episodios de entrenamiento. Autor.

	Tiempo pico	Tiempo transiente	Tiempo en estado estable	Sobrenivel porcentual	Valor pico
Control Sisotool Tuner	2.0450	5.2015	5.2015	1.2945	1.8314e+3
Agente RL	0	9.9359	9.9371	inf	3.6239e+3

Tabla 20 Características del sistema con 120 episodios de entrenamiento. Autor.

Función costo criterio LQG	
Costos RL	-3.0365e+8
Costos Sisotool Tuner	-1.617e+7

Tabla 21 Costos con 120 episodios de entrenamiento. Autor.

El sistema continúa siendo inestable, sin embargo, con respecto a los episodios anteriores, ahora conserva una función de costo mucho más cerca que la de Sisotool. Este procedimiento de entrenamiento le tomó 30m:53s, las características de la respuesta escalón aún siguen siendo ineficientes y por tanto las ganancias ajustadas por el usuario, permiten tener mejor rendimiento.

3.3.4 Agente TD3 entrenado con 500 episodios

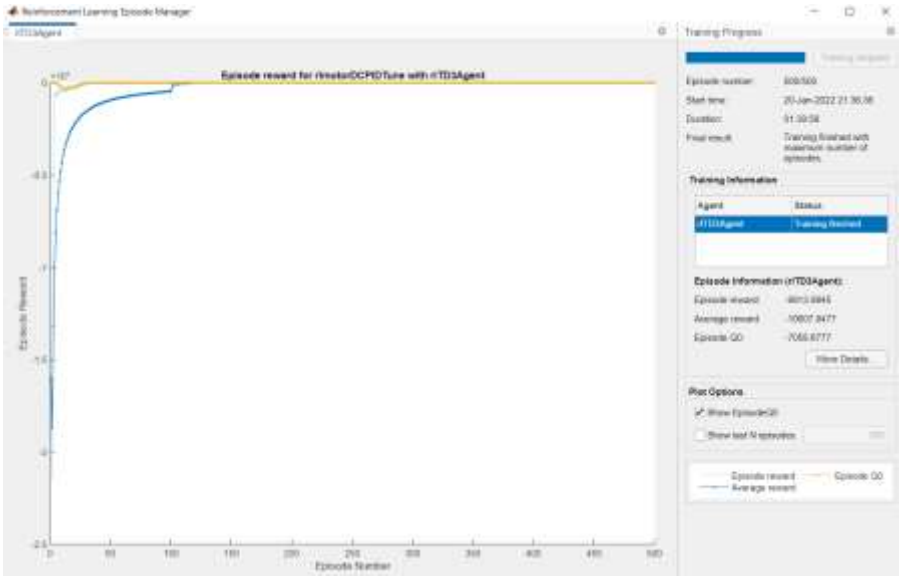


Figura 3.13 500-Episodios de recompensa del entrenamiento del agente RL. Autor.

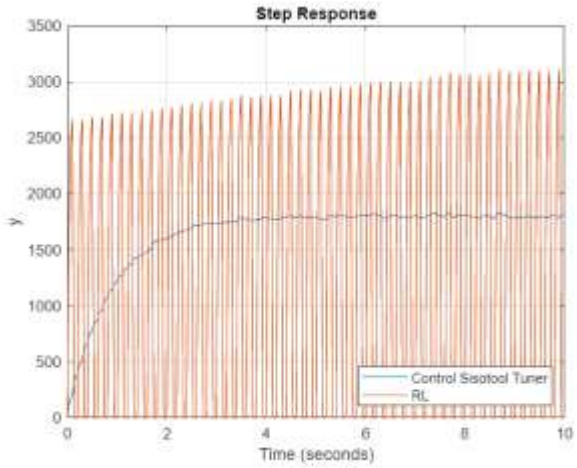


Figura 3.14 Respuesta escalón considerando ruido blanco y 500 episodios de entrenamiento. Autor.

	Control Sisotool Tuner	Agente RL
Constante proporcional	0.0031	0.0436
Constante integral	0.0299	0.0041

Tabla 22 Constantes del controlador con 500 episodios de entrenamiento. Autor.

	Tiempo pico	Tiempo transiente	Tiempo en estado estable	Sobrenivel porcentual	Valor pico
Control Sisotool Tuner	2.0450	5.2015	5.2015	1.2945	1.8314e+3
Agente RL	0	9.9362	9.9374	inf	3.1147e+3

Tabla 23 Características del sistema con 500 episodios de entrenamiento. Autor.

Función costo criterio LQG	
Costos RL	-2.2855e+8
Costos Sisotool Tuner	-1.617e+7

Tabla 24 Costos con 500 episodios de entrenamiento. Autor.

La respuesta escalón con 500 episodios continúa siendo inestable, aunque las características del sistema demuestran en conformidad con las anteriores, tender hacia los valores antes ajustados por plataformas como Sisotool. Con respecto al tiempo de entrenamiento, este fue mayor que los anteriores con 1h:39m:56s, el mismo donde las recompensas tienden a los episodios de los valores Q0, ya que el agente continúa entrenándose para obtener los pesos del crítico correspondientes al controlador PI. Las funciones de costos continúan mejorando, las cuales tienden acercarse al valor constituido por Sisotool.

3.4 Análisis de Costos

A continuación, se muestra una tabla con el presupuesto del tablero de control basado en hardware de código abierto.

Artículo	Cantidad	Descripción	Costo Unitario	Total
1	1	Gabinete metálico Beacoup	$ 60,00	$ 60,00
2	1	Pantalla Raspberry Pi 4 con case	$ 100,00	$ 100,00
3	1	Pulsador Metálico verde	$ 3,50	$ 3,50
4	1	Pulsador Metálico rojo	$ 3,50	$ 3,50
5	1	Selector Metálico corto 2P	$ 5,00	$ 5,00
6	2	Luz piloto verde	$ 0,97	$ 1,94

7	2	Luz piloto amarillo	$ 0,97	$ 1,94
8	1	Luz piloto rojo	$ 0,97	$ 0,97
9	1	Luz piloto azul	$ 0,97	$ 0,97
10	1	Placa controladora MCU+KIT RS485 Modbus Arduino	$ 50,00	$ 50,00
11	1	Raspberry Pi 4 Model B	$ 120,00	$ 120,00
12	1	Disyuntor 2 polos	$ 6,25	$ 6,25
13	1	Fuente de voltaje MW 24VDC	$ 75,00	$ 75,00
14	1	canal Riel Din por metro	$ 5,00	$ 5,00
15	1	Módulo RPI Raspberry 8 canales	$ 55,00	$ 55,00
16	1	Placa Motor Driver	$ 15,85	$ 15,85
17	1	Sistema Motor DC+Sensor y base	$ 25,00	$ 25,00
18	6	Bornera Riel Din	$ 0,30	$ 1,80
19	8	Cable #14 AWG	$ 0,50	$ 4,00
			Total	$ 535,72

Tabla 25 Presupuesto del tablero de control. Autor.

Para los costos variables se incluyen los siguientes rubros:

- Mano de obra por implementación de tarjetas Motor Driver
- Mano de obra por implementación de tablero de control
- Presupuesto total por tablero

La mano de obra de un técnico eléctrico que ejecute las actividades indicadas previamente ronda los $10,00 dólares americanos por hora, por lo tanto, se tienen los siguientes costos variables.

Horas	Precio//hora	Trabajo realizado	Total
17	$ 10,00	Armado del tablero de control	$ 170,00
1	$ 10,00	Implementación PCB	$ 10,00
		Presupuesto del tablero	$ 535,72
		Total	$ 715,72

Tabla 26 Tabla de costos Variables. Autor.

En el caso de los costos fijos se incluyen los siguientes rubros:

- Diseño de hardware de la tarjeta motor Driver
- Diseño de plano eléctrico del tablero de control
- Herramientas de trabajo (Taladro, Ponchadora, Cortadora, Destornilladores, etc.)
- Documentación de pruebas del tablero

En esta ocasión el diseño intelectual y la ingeniería empleada para el diseño del tablero y del hardware será valorizada por el número de horas invertidas en la actividad. El valor promedio por hora de servicios de un Ingeniero ronda los $20,00.

Horas	Precio/hora	Trabajo realizado	Total
6	$ 20,00	Diseño tarjeta PCB	$ 120,00
4	$ 20,00	Diseño del plano eléctrico del tablero de control	$ 80,00
12	$ 20,00	Documentación de pruebas del tablero	$ 240,00
		Herramientas empleadas para la implementación del tablero	$ 200,00
		Total	$ 520,00

Tabla 27 Tabla de costos fijos. Autor.

Con estos valores podemos obtener el costo total, pero antes, habría que simular una producción de 100 tableros obtenidos con todos estos costos y poder obtener un costo total unitario razonable.

Cantidad	Producción de tableros	Subtotal	Total
100	Tableros	$ 715,72	$ 71.572,00
1	Costos Fijos	$ 520,00	$ 520,00
	Total		$ 72.092,00

Tabla 28 Producción tableros. Autor.

El costo unitario total será de $720,92 pero este aún no sería el precio de venta al público, para esto determinamos un porcentaje de utilidad razonable, que va a depender de factores externos como, por ejemplo, competencia con otros productos similares en el mercado.

Para esto se analizará dos productos similares al propuesto en el mercado.

3.4.1 Producto 1: Kit de entrenamiento Mechatronic

El siguiente kit de entrenamiento contiene como controlador a un S7-1200 de marca Siemens y tiene los siguientes parámetros técnicos:

Dimensión: 500mm x 340mm x 160mm

Potencia de funcionamiento: voltaje AC220V monofásico, frecuencia 50/60Hz, potencia <150VA

Fuente de alimentación: Fuente de voltaje DC24V 1 vía, capacidad 1A

Peso: alrededor de 5kg

PLC: Siemens S7-1200

Figura 3.15 Kit Mechatronic. (GTEE, s.f.)

3.4.2 Producto 2: TSC-LAB

Se trata de un hardware de desarrollo con aplicaciones de sistemas de control embebido, con el módulo ESP32. Presenta una planta de temperatura y una de velocidad.

Figura 3.16 Laboratorio de temperatura y velocidad. (A, s.f.)

3.4.3 Comparación de los productos

Características	TSC-LAB	Kit Mechatronic	Tablero propuesto
HMI	NO	NO	SI
Wi-fi	SI	NO	SI
Grado de Protección al entorno industrial	NO	IP-65	IP-41

Sistema de control lazo cerrado	SI	NO	SI
Tablero didáctico	NO	SI	SI
Protocolos de comunicación Industrial	NO	Profinet	• Modbus • Profinet
Accesibilidad de los equipos	Alta	Baja	Alta
GPIO	SI	SI	SI
Lenguajes de programación	• C++ • Micro-Python • C	• Ladder • STL • Grafcet • FUP	• Ladder • Python • C++ • Grafcet • FDB
Hardware Open Source	SI	NO	SI

Tabla 29 Características de los productos. Autor.

Con esta comparación podríamos estimar un precio de venta al público del producto propuesto con un margen de utilidad del 20%.

$$Precio\ venta = \frac{CTU}{1 - \%\ Utilidad} = 901{,}15 \quad (36)$$

	TSC-LAB	KIT MECHATRONIC	Tablero de Control propuesto
Precio de venta	$ 150,00	$ 1200,00	$ 901,15

Tabla 30 Comparación de precios. Autor.

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- Se diseñó un tablero de control conformado por hardware de código abierto, entre los que resalta: Arduino y Raspberry. Dicho tablero, cuenta con el grado de protección industrial IP41 el cual evita la penetración de elementos >1mm de diámetro y protección contra el goteo vertical del agua. La interfaz de comunicación con el usuario es a través de pulsadores, selector, pantalla táctil e indicadores pilotos. El mismo cuenta con acceso presencial por SSH o remoto por medio de Anydesk. Además, contiene un sistema de velocidad conformado por un motor DC y un sensor de efecto Hall para el control a lazo abierto o cerrado.

- Se utilizó el software de MATLAB 2021b para realizar un algoritmo de control de velocidad de un motor DC, que permita obtener las constantes PI adaptativo basándose en el aprendizaje por refuerzo. La política utilizada fue del tipo DDPG TD3, quiénes contenían el actor y crítico conformado por redes neuronales capaces de establecer el agente RL. Las redes fueron dirigidas por 300 neuronas que constantemente aprendían de las recompensas y se valoraba, cuán óptima fue la señal de control mediante la función de costos del criterio LQG, indicando el menor esfuerzo del controlador para llegar a la señal de referencia. Dos críticos permitieron obtener los pesos del controlador proporcional e integral respectivamente, luego que se entrenó el agente RL.

- El controlador PI generado por el algoritmo de aprendizaje por refuerzo obtuvo un mayor rendimiento en comparación al controlador diseñado por Sisotool, que compromete el control clásico. La característica de la prueba escalón del algoritmo RL representaban una respuesta sobre amortiguada,

con un menor tiempo de estabilización y error de estado estacionar cercano al cero. La función de costos LQG indicó que el controlador RL ejecuta una señal acción con el menor esfuerzo para llegar a su punto de referencia. Con Ruido o sin Ruido blanco, tienen una diferencia alrededor de 6.7392e+6 entre las funciones de costos, destacando así que el control PI adaptativo basado en RL presenta mayor eficiencia.

- Se realizó dos pruebas distintas con el tablero para ejemplificar la versatilidad del uso del equipo. La primera prueba se enfocó al uso de los lenguajes de programación empleados en la industria de la automatización como el lenguaje Escalera y también a la programación de un HMI, teniendo finalmente un tablero que permite visualizar e interactuar a los estudiantes con una herramienta real, aumentando sus habilidades prácticas. En la segunda prueba se realizó la comunicación serial entre los dos sistemas embebidos, a su vez se usó el protocolo MQTT para publicar datos en la nube y poder visualizarlos en una pantalla mediante un Dashboard. Este tablero tiene un precio al mercado de $901,15 inferior a los tableros didácticos de alta gama que se encuentran en el mercado, pero es superior con respecto a las características y posibilidades que ofrece, convirtiéndolo en un producto altamente competitivo.

Recomendaciones

- Desarrollar el entrenamiento del agente RL en tiempo real, mediante la configuración de la comunicación serial con los bloques entre Simulink y Arduino. De esta forma se evita el proceso de identificación del sistema, ya que el Agente RL aprende de las observaciones del entorno, que constituye la velocidad del motor DC donde, el sistema podrá ser modificado, para demostrar el controlador PI adaptivo.

- Se recomienda aumentar la resolución del encoder del motor DC, extendiendo el número de imanes por revolución, de esta forma se podrían obtener parámetros experimentales del motor DC que representen mejor al sistema y a su vez, las mediciones del número de vueltas en cierto periodo

serán más exactas, generando un valor de RPM confiable. Además, utilizar un computador con mejores características computacionales para que el desfase de la comunicación entre el Arduino y Simulink sea mínimo y se pueda establecer un menor tiempo de muestreo en la adquisición.

- Para mejorar el tiempo de aprendizaje con respecto al Agente RL, es posible aumentar la capacidad neuronal del crítico, destacando así su habilidad de interpretar las recompensas y los distintos episodios Q0 en su ejecución. Además, se debe recordar, que los costos computacionales influyen en su capacidad de aprender, por ello al intentarlo con mejores características que las usadas en esta tesis, es posible optimizar dicho tiempo.

- Para realizar las pruebas en el tablero de control, se deben descargar los archivos del repositorio GibHub ubicado en anexos. Las mismas, pueden ser modificadas para ser adaptadas con algún tipo de control requerida por el programador. Con respecto al aprendizaje por refuerzo, se debe tener la versión de MATLAB 2021b para que las funciones del Reinforcement Learning puedan ser ejecutadas.

BIBLIOGRAFÍA

A, V. A. (s.f.). *TSC-LAB*. Obtenido de https://tsc-lab.blogspot.com/p/code.html

Alicia Marcela Printista, M. L. (2020). Una implementación paralela del algoritmo de Q-Learning basada en un esqueda de comunicación caché. En M. L. Alicia Marcela Printista, *Una implementación paralela del algoritmo de Q-Learning basada en un esqueda de comunicación caché.* (pág. 2). San Luis-Argentina: ANPCyT.

Aristizábal, M. C. (2006). Evaluación simétrica de una red neuronal artificial: Aplicación al caso de la inflación en Colombia. En M. C. Aristizábal, *Evaluación simétrica de una red neuronal artificial: Aplicación al caso de la inflación en Colombia* (pág. 4). Medellín.

Athans, M. (1971). The role and use of the stochastic linear-quadratic-gaussian problem in control system desgin. *IEEE Transactions on Automatic Control*, 529 - 552.

Ausin, M. S. (29 de Marzo de 2020). *Medium*. Obtenido de https://markelsanz14.medium.com/introducci%C3%B3n-al-aprendizaje-por-refuerzo-parte-2-q-learning-883cd42fb48e

Barto, R. S. (2018). *Reinforcement Learning: An introduction.* London : Westchester Publishing Services.

Cantarero, T. Á. (2015). *Diseño del Controlador PID.* Sevilla: Departamento de Ingeniería de Sistemas y Automática - Universidad de Sevilla.

Cedeño, R. U. (2021). *Resoluciones COE Nacional.* Quito: Gobierno de la república del Ecuador.

D. I. Zabala-Benavides, J. F.-C. (2021). Design and implementation of LQG controller in Ball & Bem system. *Visión Electrónica vol. 15(2)*, https://revistas.udistrital.edu.co/index.php/visele/article/view/17594.

DataRobot. (2021). *DataRobot*. Obtenido de https://www.datarobot.com/wiki/prediction/#content

Dorf, R.C; Bishop, R. H. (2005). *Sistemas de control moderno.* Madrid: Pearson Educación.

ElectronicLab. (s.f.). *ElectronicLab*. Obtenido de ElectronicLab: https://electronilab.co/tienda/a3144-sensor-de-efecto-hall/

Foy, P. (2019). *MLQ.ai*. Obtenido de https://www.mlq.ai/deep-reinforcement-learning-twin-delayed-ddpg-algorithm/

GTEE. (s.f.). Obtenido de https://spanish.alibaba.com/product-detail/vocational-siemens-plc-trainer-educational-equipment-training-box-mechatronics-training-kit-60660260160.html

Hernanz, S. R. (2019). *Aprendizaje por refuerzo en sistemas robóticos.*

Jecrespom. (4 de Marzo de 2020). *Aprendiendo arduino*. Obtenido de Aprendiendo arduino: https://aprendiendoarduino.wordpress.com/2020/03/04/manejar-gpio-raspberry-pi/

Kalman, R. E. (1960). A new approach to Linear Filtering and Prediction Problems, transactions of the ASME. *Journal of Basic Engineering Vol. 82*, 35 - 45.

Latorre, A. S. (2019). Aprendizaje por refuerzo para control de sistemas dinámicos. En A. S. Latorre, *Aprendizaje por refuerzo para control de sistemas dinámicos.* (pág. 28). Santiago de Cali.

MathWorks. (2020). *MathWorks*. Obtenido de MathWorks: https://la.mathworks.com/help/reinforcement-learning/agents.html

MathWorks. (2020). *MathWorks* . Obtenido de MathWorks .

MathWorks. (19 de Enero de 2022). *Centro de ayuda - MathWorks - Diseño lineal cuadrático gaussiano*. Obtenido de Centro de ayuda - MathWorks - Diseño lineal cuadrático gaussiano: https://www.mathworks.com/help/control/getstart/linear-quadratic-gaussian-lqg-design.html

Matilde Pilar Legua Fernández, Luis Manuel Sánchez Ruiz. (2017). *Ecuaciones diferenciales y transformadas de Laplace con aplicaciones.* Valencia: Universidad Politécnica de Valencia.

Matlab Academy. (2021). *Matlab Academy*. Obtenido de Matlab Academy: https://matlabacademy.mathworks.com/R2021b/portal.html?course=reinforcementlearning#chapter=3&lesson=3&section=1

Miranda, C. V. (2016). *Sistemas de control continuos y discretos.* Madrid: Paraninfo.

Ogata, K. (2010). *Ingeniería de control moderna.* Madrid: Pearson Educación S.A.

Ortega, M. Q. (2009). Obtención experimental de los parámetros del motor que se utilizará en el sistema de locomoción de una esfera rodante. . BUCARAMANGA: Universidad Pontificia Bolivariana.

Pro, I. (2020). *Ichi Pro*. Obtenido de https://ichi.pro/es/introduccion-al-aprendizaje-por-refuerzo-parte-2-q-learning-85297949015614

ROBOTSHOP. (s.f.). Obtenido de https://www.robotshop.com/media/files/pdf/RPi-Relay-Board-Schematic.pdf

Rosero, P. P. (2011). Identificación de procesos: uso de algoritsmos en Matlab para encontrar un modelo por identificación para un proceso de medición de pH. *INGENIUS - Revista de Ciencia y Tecnología*, 4.

Rouhiainen, L. (2018). *Inteligencia artificial: 101 cosas que debes saber hoy sobre nuestro futuro .* Barcelona: Planeta S.A.

Simbro, G. C. (2007). Procesos de decisión de Markov aplicados a la locomoción de robots hexápodos. En G. C. Simbro, *Procesos de decisión de Markov aplicados a la locomoción de robots hexápodos.* (pág. 203). Tonantzintla, Puebla: INAOE.

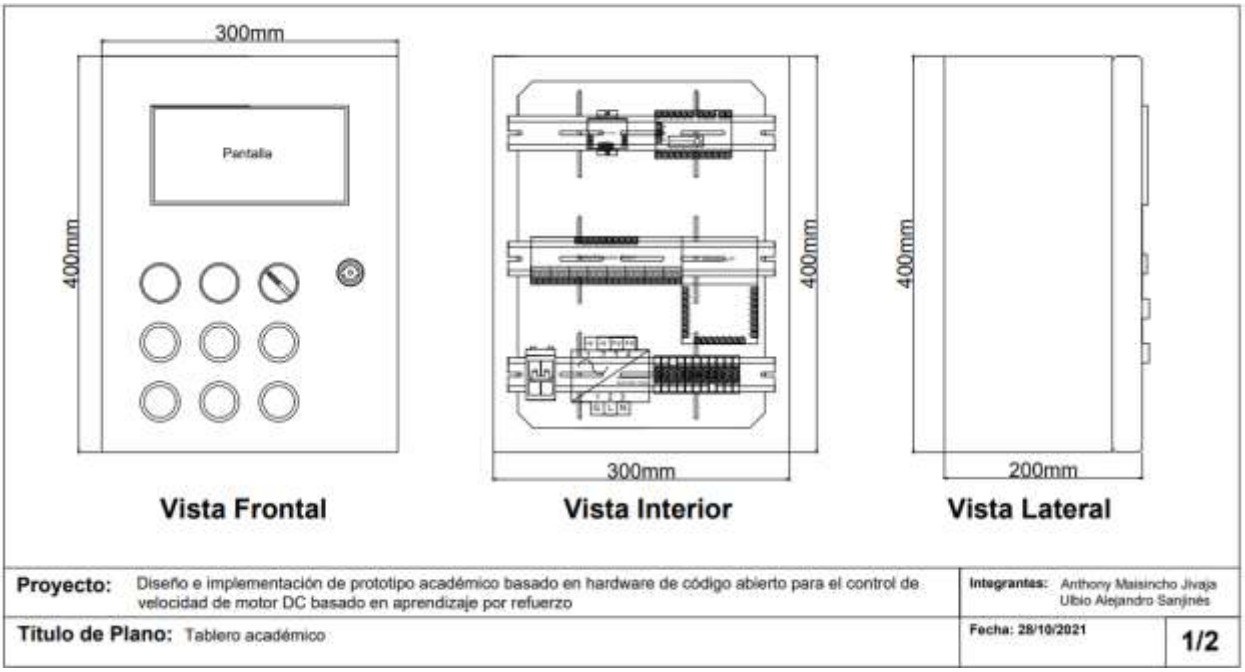
Sole, A. C. (2005). *Instrumentación Industrial.* Madrid: Marcombo S.A.

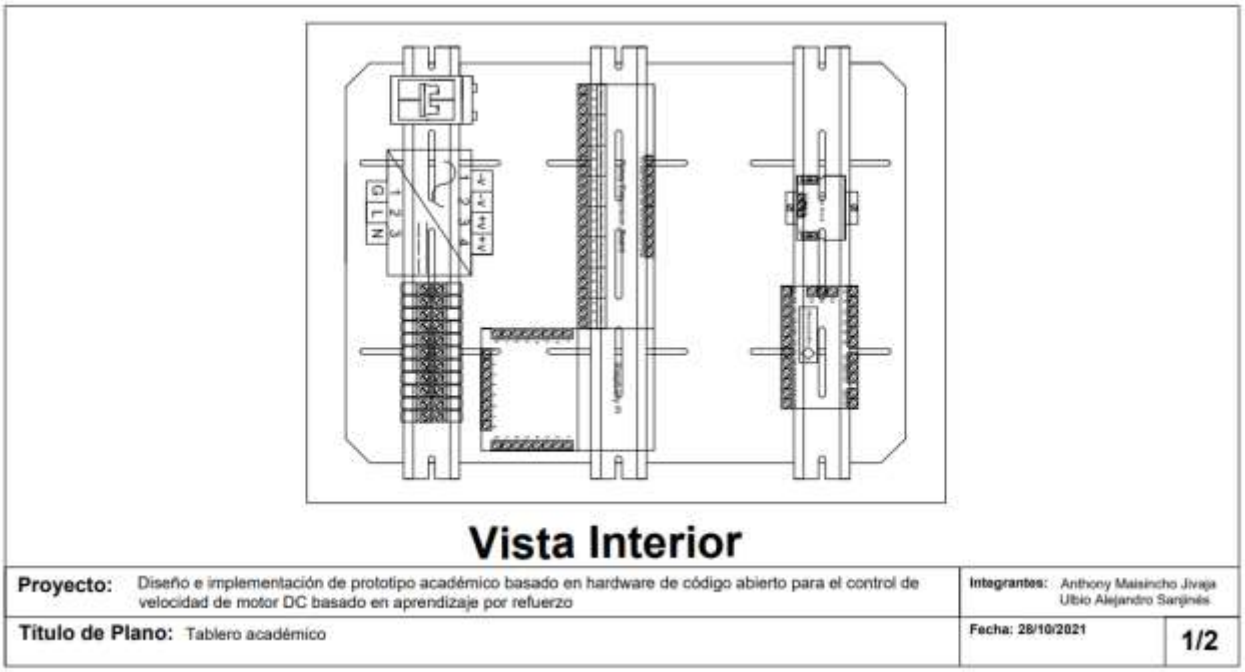
TechTarget. (Enero de 2017). *ComputerWeekly*. Obtenido de https://www.computerweekly.com/es/definicion/Aprendizaje-automatico-machine-learning

TYHE. (s.f.). *Tyhe*. Obtenido de Tyhe: https://www.alibaba.com/product-detail/Miniature-micro-45mm-diameter-12v-24v_60405149408.html

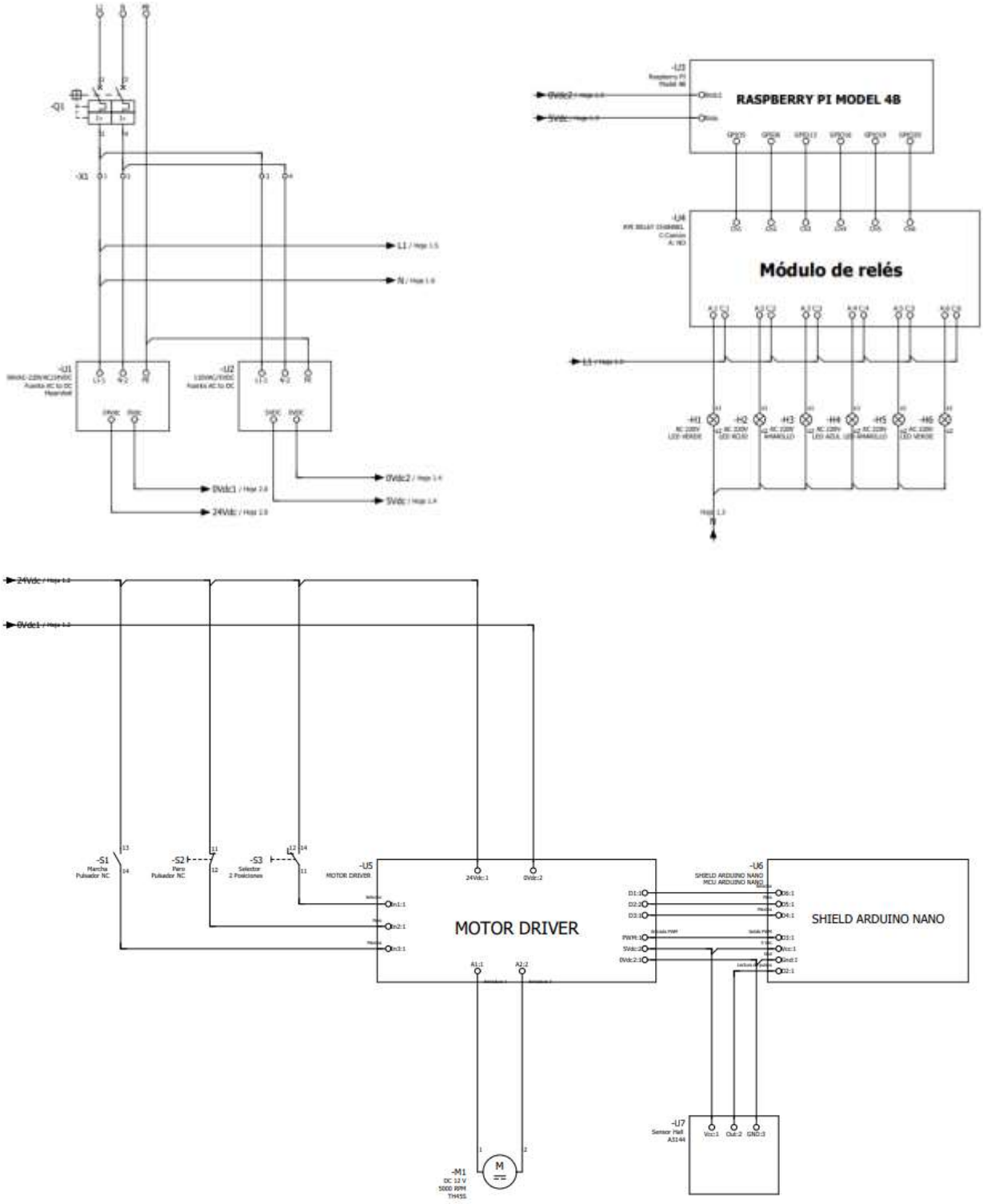
APÉNDICES

APÉNDICE A





Plano 1 Tablero académico vista Interior, exterior y lateral. Autor



Plano 2 Plano eléctrico del tablero académico. Autor

APÉNDICE B

Enlace al repositorio de GitHub con todas las pruebas realizadas en el tablero académico: https://github.com/Maydrummer/Tablero-RL

APÉNDICE C

Foto 1 Esbozo de la interfaz del tablero.

Foto 2 Acople de Riel Din.

Foto 3 Ajuste de las luces piloto.

Foto 4 Cableado de los equipos que conforman el tablero.

Foto 5 Colocación de equipos en Riel Din.

Foto 6 Implementación de Placa PCB.

Foto 7 Prueba de Sistema Motor DC-Encoder

Foto 8 Ajuste de cableado.

Foto 9 Armado de tablero de control.

Foto 10 Pruebas de funcionamiento del tablero.

Foto 11 Resultado final del tablero.

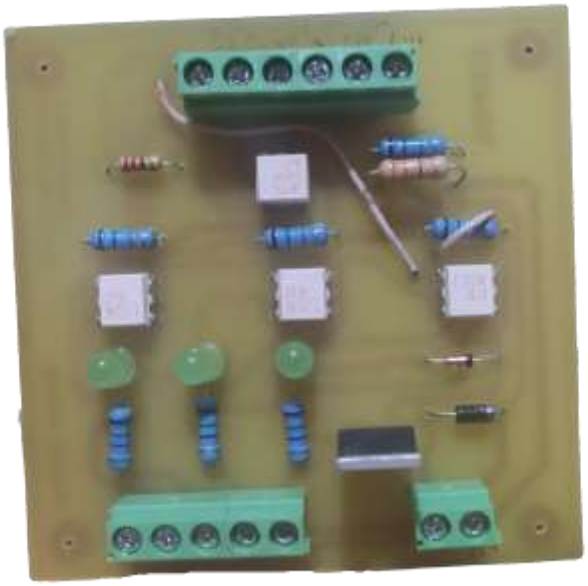


Foto 12 Resultado final Placa Motor Driver.

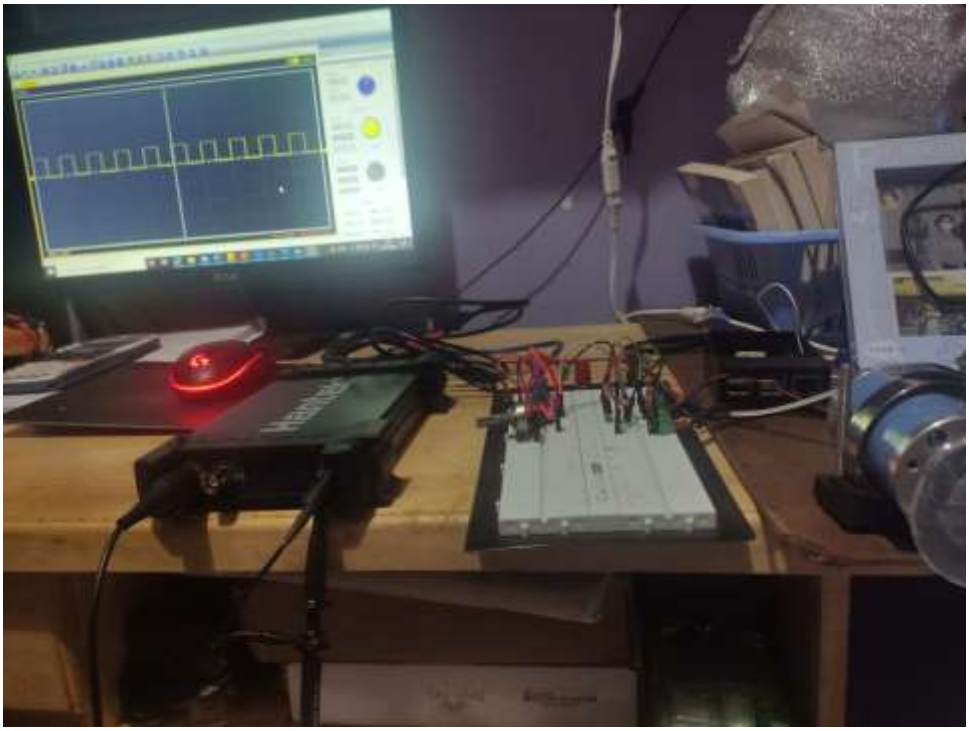


Foto 13 Medición de constante de tiempo mecánica.