BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

MAESTRÍA EN CIENCIAS DE LA ELECTRÓNICA
OPCION EN AUTOMATIZACIÓN

“Desarrollo de una red neuronal convolucional para la identificación paramétrica de un brazo robot.”

T E S I S

PRESENTADA PARA OBTENER EL TITULO DE:

MAESTRO EN CIENCIAS DE LA ELECTRONICA
OPCION EN AUTOMATIZACION

PRESENTA

ING. GENGIS CRUZ CORONEL*

DIRECTORES DE TESIS

DRA. MARÍA AURORA DIOZCORA VARGAS TREVIÑO

(FCE-BUAP)

DR. SERGIO VERGARA LIMÓN (FCE-BUAP)

DR. CARLOS LEOPOLDO CARREÓN DÍAZ DE LEÓN (FCC-BUAP)

*Becario SECIHTI

Puebla, Puebla, agosto 2025

Resumen

En esta tesis se presenta el diseño de una red neuronal convolucional para realizar la identificación paramétrica de un brazo robot de 2 gdl, esta técnica ha sido implementada en esta tesis debido a que esta técnica se ha usado previamente en otros trabajos y ha demostrado una alta similitud en la obtención de los parámetros usando el modelo dinámico del robot a identificar. En este trabajo se propone una estructura de red neuronal convolucional para el caso de estudio presentado, se usan las imágenes generadas usando las señales de posición, velocidad y torque obtenidas al aplicar una trayectoria predefinida. La red propuesta será entrenada usando el gradiente descendente del error, además el repositorio de imágenes de entrenamiento se realizará usando el modelo dinámico del brazo robot, el cual también es presentado, este repositorio de imágenes será generado usando valores pseudoaleatorios en el modelo dinámico. Una vez entrenada la red se mostrará su validación y la identificación de los parámetros del robot usando una imagen generada para cada grado de libertad usando los torques obtenidos experimentalmente. En los resultados se mostrará la validación del modelo dinámico a un control de posición para verificar que el comportamiento es el deseado. Los resultados obtenidos alcanzan en el segundo grado de libertad más del 80% de similitud aplicando una métrica que considera la señal en el dominio del tiempo y de la frecuencia. La principal contribución de esta tesis es el desarrollo de una metodología de identificación paramétrica para el brazo robot de 2 gdl que permita usar como entrada imágenes generadas a partir de las señales del robot, ofreciendo una identificación más precisa en el ámbito de la robótica.

Agradecimientos

A Dios, por la vida y por todas las oportunidades que me ha brindado.

A mis padres Isabel y Albaro, por el apoyo que siempre me han brindado, el cual me ha permitido seguir avanzando hasta llegar a donde estoy. A mis hermanos Alba, Araceli y Leonel por haberme apoyado en todo momento, y por ser parte de todo este proceso.

A mis asesores, la Dra. María Aurora Diozcora Vargas Treviño, el Dr. Sergio Vergara Limón, y el Dr. Carlos Leopoldo Carreón Díaz de León por todo su conocimiento y el apoyo brindado.

A mi jurado revisor compuesto por el Dr. José Eligio Moisés Gutiérrez Arias, Dr. Jaime Cid Monjaraz y el Dr. Jesús López Gómez por las observaciones y comentarios que permitieron desarrollar y culminar este trabajo de tesis.

A mis amigos de la maestría, por todos los buenos momentos que pasamos en esta etapa, les agradezco por su amistad.

Al M.C. Michelle Guerra Marín y el M.C. Juan Francisco Pintor Michimani por el apoyo proporcionado en el desarrollo de este trabajo.

A la Benemérita Universidad Autónoma de Puebla y a la Facultad de Ciencias de la Electrónica, que a través de toda la infraestructura y docentes me permitieron ser parte de un posgrado de calidad.

Y, por último, pero no menos importante, agradezco al SECIHTI, por el apoyo económico que permite que muchas más personas puedan seguir preparándose académicamente, y que contribuye al avance en la investigación y desarrollo del país.

Dedicatoria

Para A. E. M. R.

Índice general

Índice de Figuras

Índice de Tablas

Introducción

La identificación paramétrica de un robot consiste en hallar los parámetros asociados a un modelo dinámico que describe su comportamiento. Realizar la identificación paramétrica de un robot permite que este pueda ser simulado, por lo que obtener el valor de estos parámetros más cercanos a los valores reales permite obtener un modelo dinámico más preciso. En este trabajo se propone usar una red neuronal convolucional (RNC) para realizar la identificación paramétrica, esto con el fin de obtener los parámetros del robot con mayor precisión y un modelo dinámico usando el método de parámetros agrupados.

Tal como su nombre lo indica las redes neuronales convolucionales se basan en el uso de la convolución, la cuál es una operación usada en el procesamiento digital de señales, comúnmente se refiere a estas mediante su acrónimo en ingles CNN (Convolutional Neural Network) [1]. Este tipo de redes han sido ampliamente usadas en el procesamiento de imágenes ya que gracias a esta se pueden resolver problemas de clasificación de patrones, segmentación, identificación y clasificación de imágenes, sin embargo, no solo están enfocadas a las imágenes, sino también a nuevas áreas como son la robótica, el lenguaje, etc.

Actualmente ya se ha trabajado en la identificación paramétrica de robots, en [2] se muestra el diseño de una red neuronal convolucional para un brazo robot de dos grados de libertad, con el cual se trabajará en esta tesis, en esta se muestra una propuesta de red neuronal convolucional para la parametrización del brazo robot. De igual forma en [3] se muestra el desarrollo de cuatro propuestas de red neuronal convolucional para la identificación paramétrica usando un robot cartesiano y un brazo robot, en esta se muestra una similitud superior al 97% en la identificación paramétrica con el modelo dinámico.

Una descripción de la estructura de las redes convolucionales puede verse en [4] donde se detalla las partes más importantes de las redes convolucionales. En [5] se muestra una comparación entre los diferentes métodos de la operación de reducción “pooling” proporcionando las ventajas y desventajas de cada uno de estos métodos. En el área de la medicina en [6] se implementó una red neuronal convolucional de 9 capas para detectar arritmias y ataques del corazón usando señales de electrocardiograma. De igual forma en [7] se usó una red R-CNN para la detección e identificación de tumores en el hígado, e identificar si estos son benignos o malignos, usando imágenes de resonancia magnética en adición al diagnóstico médico. En [8] se aplicó una red neuronal para calcular y predecir la rotación, trayectoria y la velocidad de una pelota de tenis de mesa usando varios algoritmos de inteligencia artificial. Dentro del área industrial existen diferentes clases de robots y diferentes marcas, un ejemplo son los robots de la Marca Kuka [9], los cuales son usados para diferentes aplicaciones como son soldadura, ensamble, inspección, montaje, etc. En el ámbito de la investigación en robótica se tiene a los robots Sophia [10] desarrollado por la

empresa Hanson Robotics, este robot puede generar gestos, seguir una conversación como lo haría una persona real, la empresa Engineered Arts [11] ha desarrollado al robot Ameca el cual es una plataforma de desarrollo para nuevas tecnologías, basado principalmente en la interacción humano-robot , lo cual cada vez es más necesario ya que se pueden construir robots que puedan interactuar con personas y servir de asistencia, también existe al robot Atlas [12] desarrollado por la empresa Boston Dynamics, este robot puede saltar, evadir obstáculos y cargar objetos pesados. Estos robots han sido desarrollados aprovechando diferentes técnicas de inteligencia artificial, principalmente en el lenguaje para poderse comunicar de forma oral, así como en emular los movimientos y gestos humanos. En el ámbito de la inteligencia artificial ha tomado mucha importancia, tal que han surgido librerías y plataformas para el desarrollo en inteligencia artificial como Numpy, TensorFlow, Keras y Matlab, lo que ha permitido desarrollar redes neuronales de manera más rápida.

Objetivo General:

"Diseñar e implementar una red neuronal convolucional para que realice la identificación paramétrica de un brazo robot"

Objetivos particulares:

1. Estudiar el método de parámetros agrupados
2. Obtener el modelo dinámico del brazo robot usando el método de parámetros agrupados
3. Estudiar los fundamentos de redes neuronales
4. Diseñar una red neuronal convolucional
5. Diseñar un algoritmo de identificación paramétrica que utilice la red neuronal convolucional propuesta
6. Implementar el algoritmo de identificación paramétrica
7. Realizar pruebas experimentales para evaluar el rendimiento de la red neuronal.
8. Comparar los resultados obtenidos con la red neuronal y los resultados con el modelo dinámico
9. Publicación de los resultados.
10. Escritura de la tesis.

La presente tesis está estructurada en siete capítulos, abarca desde la descripción del brazo robot hasta la identificación paramétrica usando su modelo dinámico. En esta tesis se plantea usar redes neuronales convolucionales para realizar la identificación paramétrica, por lo que se mostraran los resultados usando esta técnica.

Se comienza con la descripción de todos los elementos mecánicos que integran el brazo robot de 2 grados de libertar, posteriormente se mostrará la obtención del modelo dinámico del robot, con su correspondiente representación en variables de estado. Posteriormente en el capítulo 4 se mostrarán los elementos electrónicos usados, la caracterización de los motores para volverlos de quasi transmisión directa. Se mostrarán los cambios y actualizaciones para terminar con la integración de todo el sistema.

En el capítulo 4 y 5 se mostrará el firmware y software que fueron usados para la adquisición y salida de las señales del sistema, como son posición, velocidad y torque, así como del software para él envió, y captura de los datos obtenidos.

Finalmente, en el sexto capítulos de esta tesis se muestra la teoría de las redes neuronales convolucionales, y finalmente el último capítulo, se muestran los resultados obtenidos del modelo dinámico y de la identificación paramétrica, se mostrarán algunos ejemplos de imágenes obtenidas del entrenamiento, y las gráficas obtenidas de posición, velocidad y torque sin y con la identificación paramétrica.

Capítulo 1. Brazo robot de 2 grados de libertad

El robot usado para el desarrollo de esta tesis es un brazo robot articular de dos grados de libertad (GDL), este robot se caracteriza por tener articulaciones rotatorias, estos son accionados por motores de corriente directa, algunas aplicaciones de este tipo de robots son procesos de soldadura, montaje, mecanizado, etc.

En la Figura 1.1 se muestra el diagrama general del brazo robot de dos grados de libertad, la comunicación que existe entre la PC y el microprocesador (el cual se encuentra implementado en el FPGA) es realizada usando un módulo Wi-Fi, este es usado para enviar la configuración del sistema, así como para recibir los datos capturados. Dentro del microprocesador se tienen los bloques de memoria ROM y RAM, en estos se guarda el programa para controlar el robot, así como de las variables usadas en el programa, estas pueden ser de posición, velocidad, aceleración, etc. También se tienen puertos de entrada y salida, en los puertos de entrada se encuentran los decoders los cuales se encargan de procesar las señales de los encoders para determinar el sentido de giro y la posición de los motores del robot, también se incluyen los fines de carrera, estos al implementarse pueden detener el robot en caso de que se detecte que ha alcanzado una posición determinada, en los puertos de salida se encuentran los generadores de PWM, los cuales se usan para mover el robot en ambas direcciones y para variar su velocidad. En la etapa de potencia se encuentran los puentes H, los cuales se encargan de conmutar usando las señales de PWM proporcionadas por el microprocesador, estos son importantes ya que las señales proporcionadas del FPGA son de 3.3V y 1 mA, lo cual hace necesario el uso de estos.

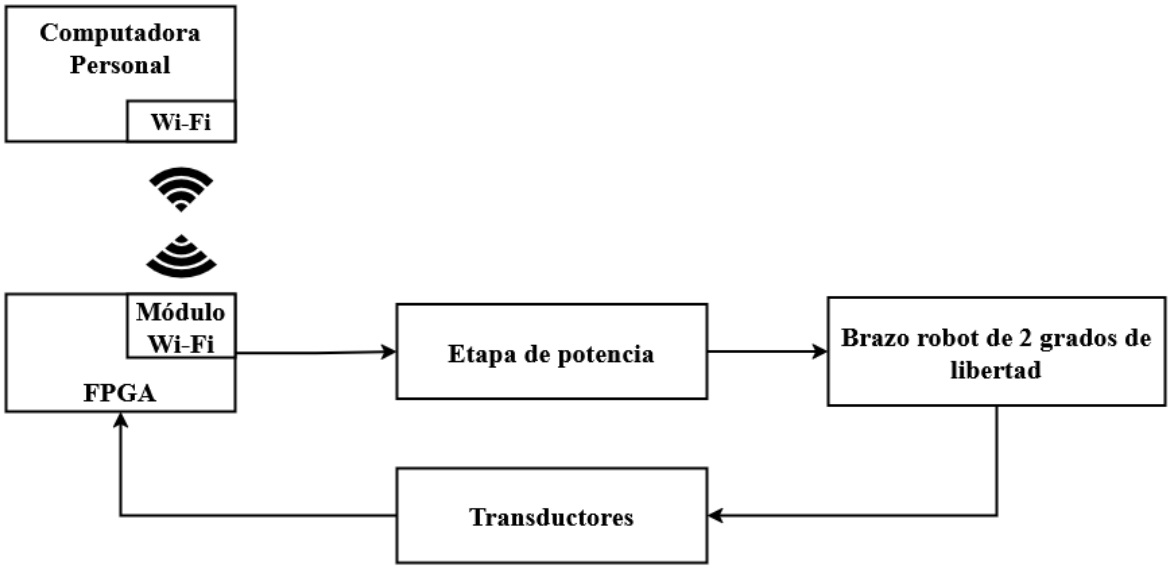


Figura 1.1: Diagrama a bloques del sistema.

1.1 Sistema mecánico

El sistema con el que se trabajó está conformado por el brazo robot constituido por 2 barras de aluminio para los eslabones, 2 acopladores de aluminio para unir el eje de cada motor con el siguiente eslabón y dos motores de corriente directa de la marca Pololu modelo GB37-131, así como su etapa de potencia, la cual contiene dos puentes H, además de una etapa para el acondicionamiento de las señales de los encoders para la conexión con una tarjeta FPGA DE0-CV con un FPGA Cyclone V, la cual será necesaria para la adquisición de datos del sistema.

En la Figura 1.2, pueden verse cada uno de los elementos mecánicos del sistema en el estado previo en el que se encontraba el robot, puede verse que el sistema no se encontraba armado ni funcionando, por lo que se realizaron las mejoras correspondientes para tener el robot funcionando de manera correcta para lograr realizar la identificación paramétrica.

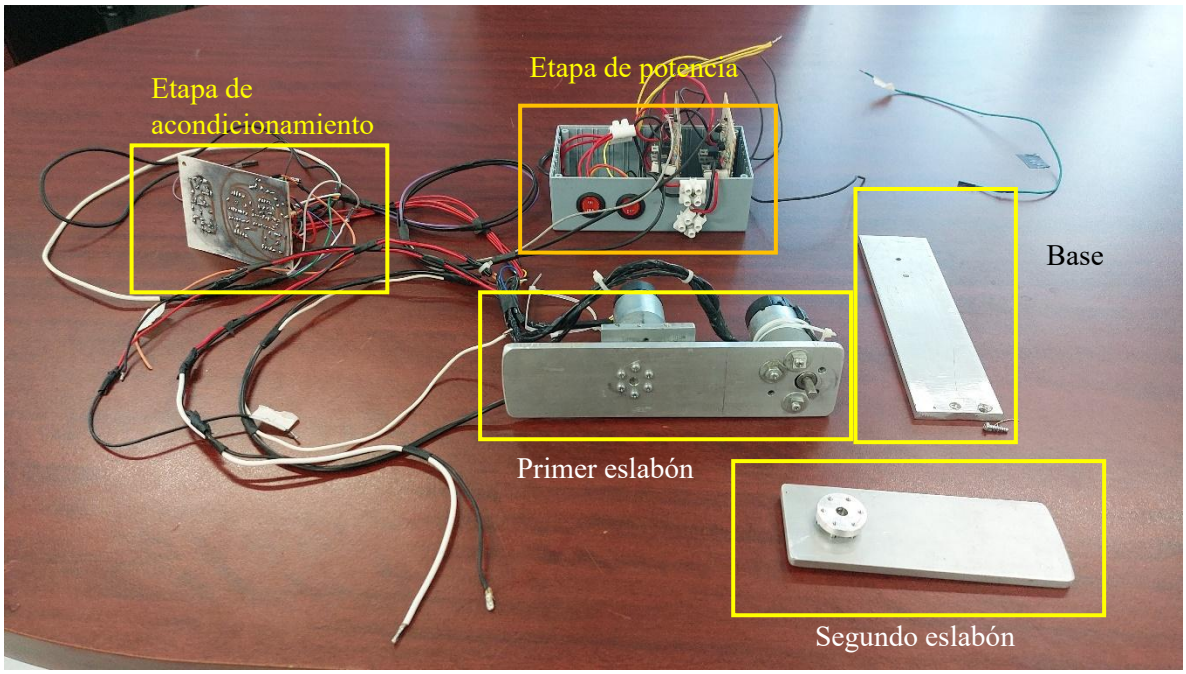


Figura 1.2: Componentes del brazo robot.

En el siguiente apartado se muestran los planos de cada uno de los elementos que componen el robot.

1.1.1 Planos

En este apartado se muestran los planos de los diferentes elementos del robot, como son los eslabones, la base del robot, del soporte del robot, así como de la estructura diseñada para la electrónica del robot, estos planos sirvieron para realizar el ensamble de todos los elementos del robot.

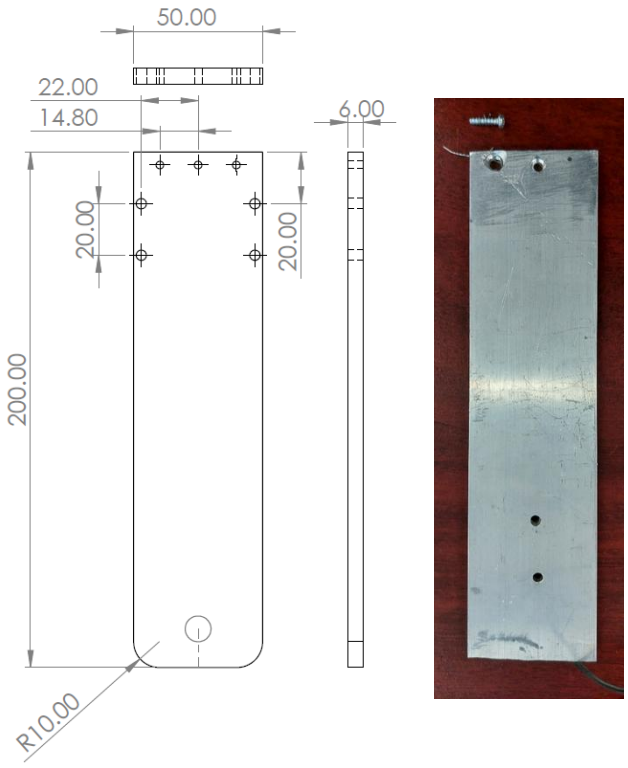


Figura 1.3: Base del robot con su respectivo plano.

En la Figura 1.3 puede observarse la base del robot, fue necesario realizar las perforaciones pertinentes de acuerdo con el plano diseñado, para lograr sujetar la base al soporte que se mostrará más adelante.

En la Figura 1.4 y la Figura 1.5 puede verse los planos de ambos eslabones con su respectivo elemento mecánico, los ajustes que se realizaron en estos fue cambiar los tornillos que se tenían previamente por los tornillos con las dimensiones adecuadas para lograr un ensamble funcional, ya que anteriormente algunos tornillos sobresalían más de lo deseado.

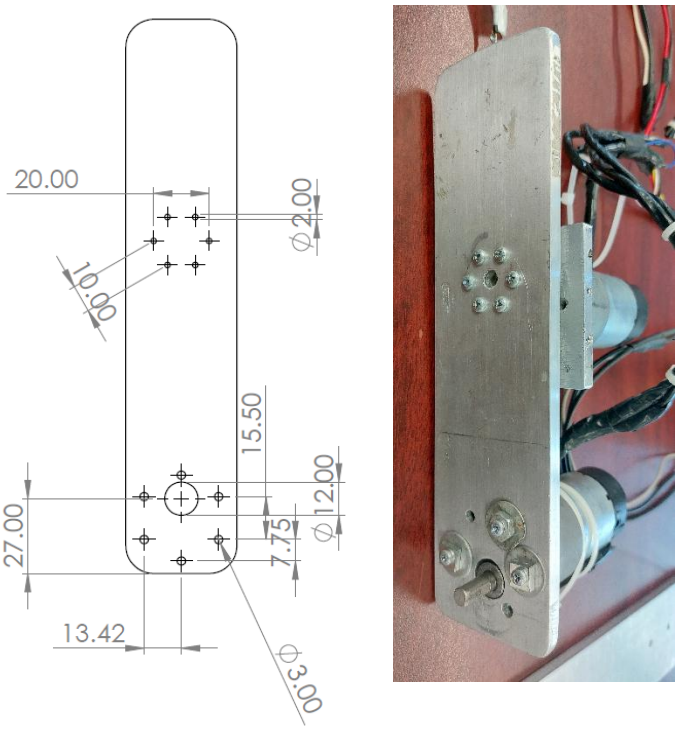


Figura 1.4: Primer eslabón con su respectivo plano.

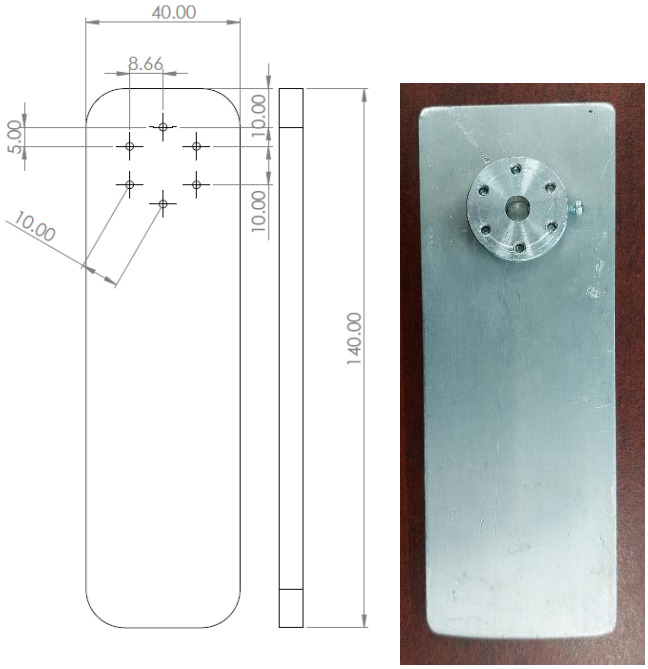


Figura 1.5: Segundo eslabón con su respectivo plano.

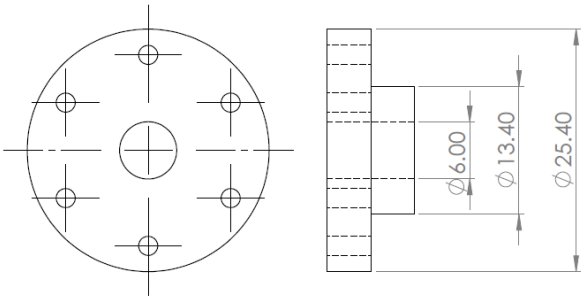
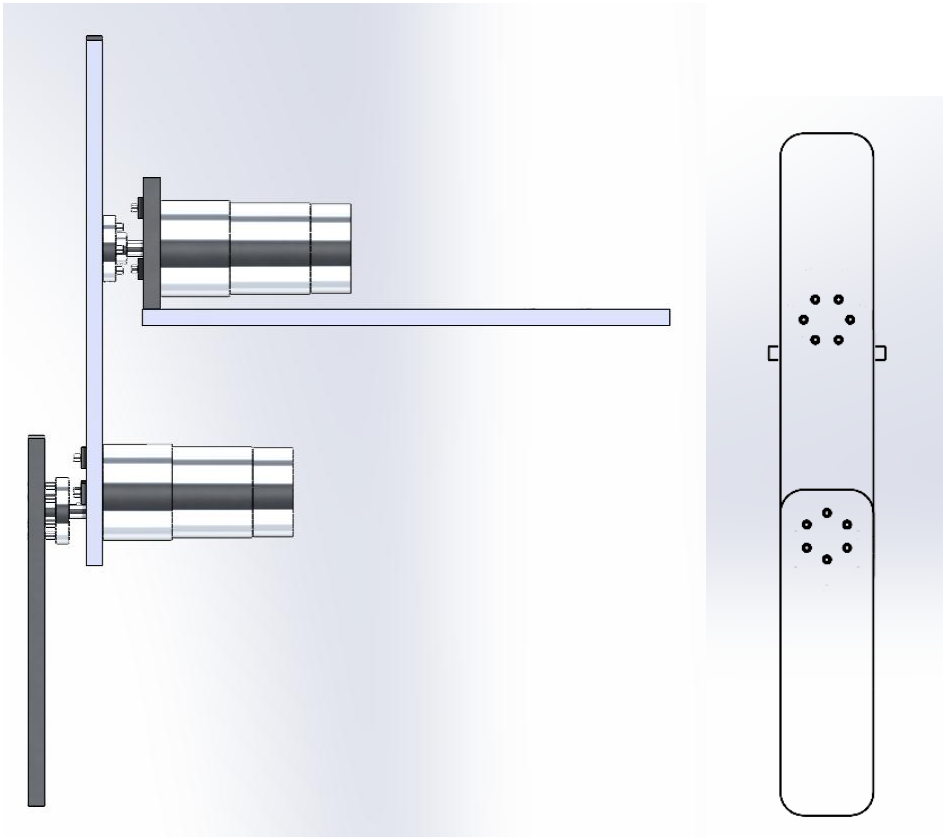


Figura 1.6: Cople de aluminio de 6 mm

En la Figura 1.6 puede observarse el plano del cople de 6 mm, este debe ir fijado al eje de cada motor, para esto fue necesario colocar opresores de 3 mm en cada uno de los acopladores de aluminio para poder unir el eje del motor, el eslabón y el cople.

Usando los planos del brazo robot, se muestra en ensamble en la Figura 1.7 tal como se debe construir el robot y la posición de cada uno de los elementos del sistema.



a) b)

Figura 1.7: a) Vista lateral del robot armado, b) Vista frontal del robot armado.

Como se puede observar en la Figura 1.7, el robot no contaba con una base de apoyo, por lo que se diseñó un soporte para todo el robot, en la siguiente Figura 1.8 se muestra el plano de la base diseñada, así como del robot completo.

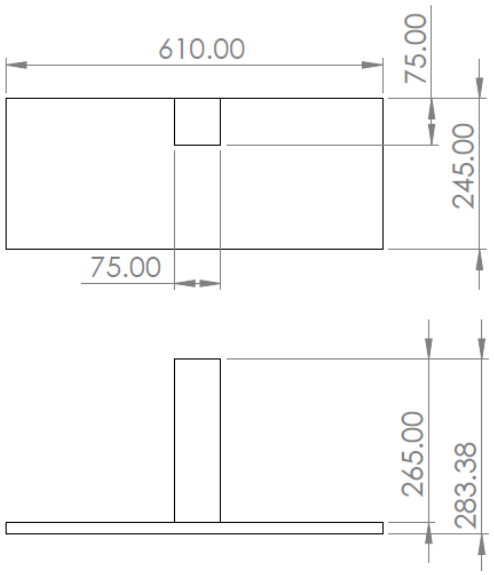


Figura 1.8: Plano de la base.

Con el fin de tener todos los componentes electrónicos en el mismo lugar, se diseñó una estructura fabricada con lámina de acrílico de 6 mm y perfil cuadrado de ½ pulgada, esta se fabricó con corte laser para darle un mejor acabado, en la siguiente Figura 1.9 se muestra el plano de la estructura.

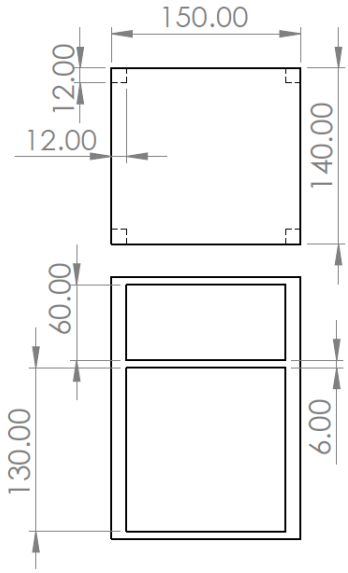


Figura 1.9: Plano de la estructura de acrílico.

1.2 Ensamble

En la Figura 1.10 se observa la estructura de acrílico con los componentes ya montados, para sujetar esta estructura se usaron tornillos M3, con sus respectivos opresores, los cuales se usaron para unir las partes intermedias, esta estructura se montó sobre la fuente de alimentación de computadora de igual manera aprovechando los orificios de sujeción.

Estructura de acrílico

Fuente de alimentación

Figura 1.10: Estructura de acrílico montada sobre la fuente de alimentación.

En la siguiente Figura 1.11 se muestra la estructura realizada conectada al robot, se realizaron pruebas para verificar que los cables no obstruían el movimiento de ambos eslabones.

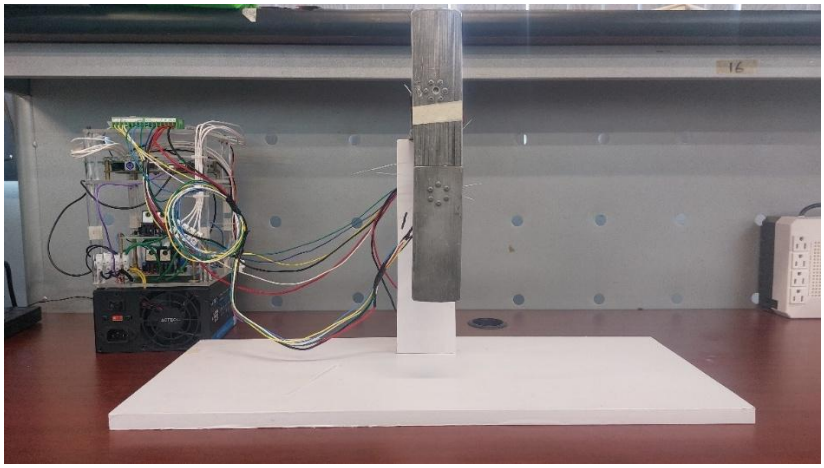


Figura 1.11: Vista frontal del brazo robot armado con su respectivo soporte.

En la Figura 1.12 se muestra la vista lateral del sistema armado, como puede observarse quedó armado de la misma forma en la que se muestra el plano de la Figura 1.7, para hacerlo fue necesario realizar las perforaciones necesarias en la placa de aluminio de la base del robot, para poder atornillarlo al soporte diseñado.

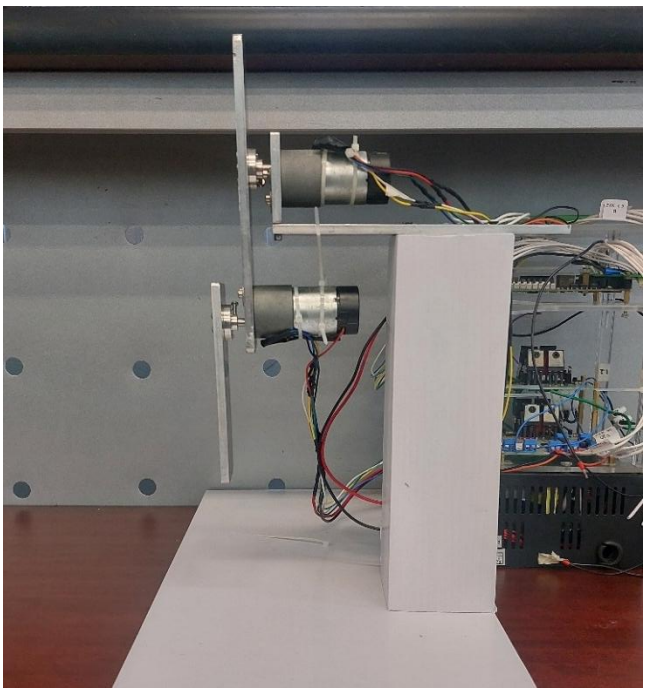


Figura 1.12: Vista lateral del brazo robot armado.

Finalmente, en la Figura 1.13, se muestra el sistema mecánico conectado a la electrónica, la cual ya se encuentra conectada con la PC, esto se explicará más adelante en sus respectivos capítulos.

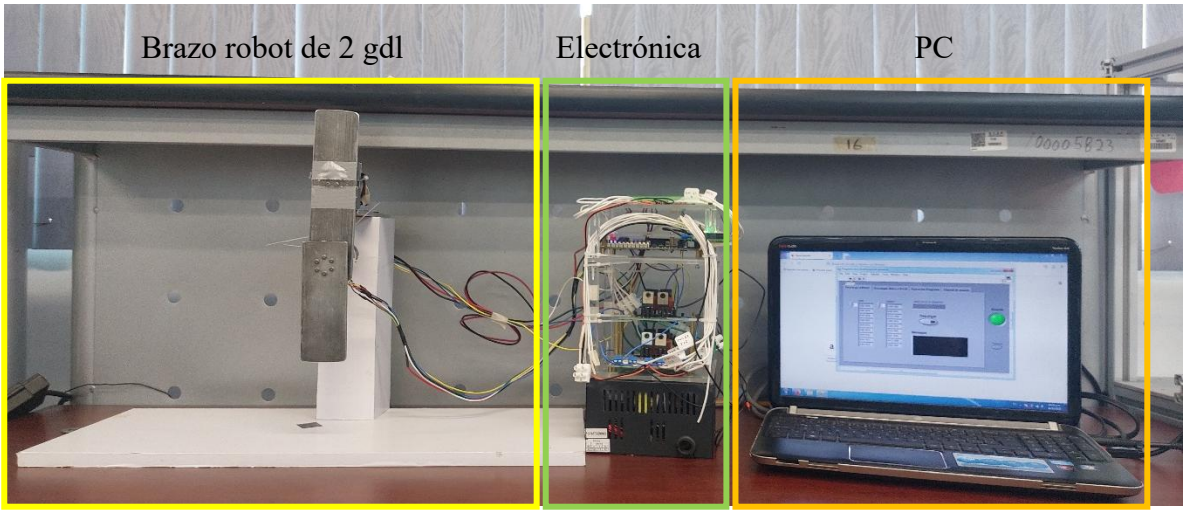


Figura 1.13: Sistema completo en funcionamiento.

Conclusiones

En este capítulo se realizó una descripción de los elementos mecánicos que componen el brazo robot, empezando por los planos, aunque el robot fue construido previamente, este no se encontraba de manera funcional, por lo que fue necesario realizar el ensamble nuevamente tomando las consideraciones necesarias como fue el ajuste de los tornillos, opresores y la base.

En la parte mecánica fue necesario hacer las adecuaciones necesarias para el montaje del brazo robot, como se observó, fue necesario diseñar un soporte para el brazo robot, una vez que se tuvo el soporte fue necesario hacer las perforaciones para el montaje de los elementos mecánicos, de igual forma fue importante el diseño de la estructura que contiene la electrónica del robot, ya que antes se tenían que realizar las conexiones con jumpers, por lo que en ocasiones había falsos contactos entre los cables. La estructura facilitó el montaje de los puentes H, de la tarjeta FPGA, del circuito de protección y del conector para las señales de los motores.

Las actualizaciones mostradas son muy importantes para el desempeño del robot, ya que al estar en movimiento es necesario que toda la estructura que lo sostiene sea muy estable, además de que facilita el traslado y su manipulación.

Capítulo 2. Modelo dinámico

Para conocer el comportamiento del brazo robot de dos grados de libertad es necesario obtener el modelo dinámico de este, por lo que este se obtuvo usando la técnica de parámetros agrupados usando la ecuación de movimiento de Euler-Lagrange.

Desplazamientos virtuales

Un desplazamiento virtual es un pequeño desplazamiento del sistema que es consistente con las restricciones, y que tiene lugar simultáneamente. Es el segundo requisito que distingue los desplazamientos virtuales de los diferenciales. desplazamientos. (Aquí, un desplazamiento diferencial es un pequeño desplazamiento del sistema que es consistente con las restricciones.)

Trabajo Virtual

El trabajo virtual es el trabajo realizado por los esfuerzos en llevar el sistema. a través de un desplazamiento virtual,

$$\delta \mathcal{W} = \sum_{i=1}^{n} e_i \delta q_i \tag{1}$$

Aquí, δW es el trabajo virtual, δq_i son los desplazamientos virtuales, e_i son los esfuerzos aplicados, i = 1; 2; …; n siendo n el número de grados de libertad. Teniendo en cuenta que q_i son los desplazamientos generalizados y e_i son los desplazamientos correspondientes. esfuerzos generalizados.

La Tabla 2.1 mostrada a continuación muestra los trabajos virtuales más comunes para las diferentes áreas en las que se puede aplicar [15].

Tabla 2.1: Trabajos virtuales para diferentes disciplinas

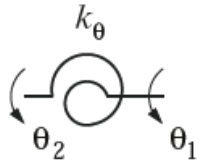
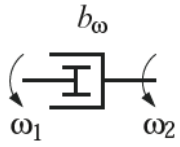
Disciplina	Trabajo virtual	
	$e\delta q$	$f\delta p$
Traslación mecánica	$F\delta x$	$v\delta p$
Rotación mecánica	$\tau\delta\theta$	$\omega\delta H$
Eléctrica	$v\delta q$	$i\delta\lambda$
Fluidos	$P\delta V$	$q\delta T$
Térmica	$T\delta S$	$-$

2.1 Análisis de parámetros agrupados

Movimiento Rotacional

En la Tabla 2.2 se muestran las ecuaciones para calcular las energías cinéticas y potenciales para elementos torsionales, a partir de estos elementos básicos se realiza el diagrama de parámetros agrupados.

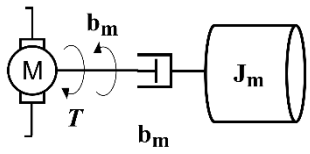
Tabla 2.2: Energía en movimiento rotacional

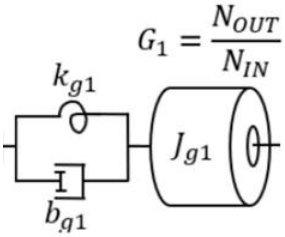
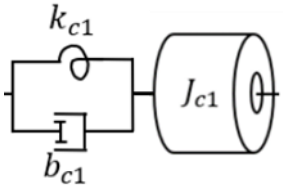
Elemento	Fuerza	Energía	Símbolo
Resorte	$\mathrm{F} = \mathrm{k}\theta$	$E_p = \frac{1}{2}k\theta^2$	
Amortiguador	$F = B\frac{d\dot{\theta}}{dt}$	$E_c = \frac{B\dot{\theta}^2}{2}$	
Masa		$E_c = \frac{1}{2}Iw^2$	

Donde E_c es la energía cinética y E_p es la energía potencial.

En la Tabla 2.3 se muestra cada bloque que compone el diagrama del sistema, en este se incluyen los bloques del diagrama esquemático obtenido para el primer motor y eslabón, los cuales representan el primer grado de libertad del robot completo. Es importante notar que en el análisis propuesto la rigidez del eje de ambos motores es considerada como 0.

Tabla 2.3: Bloques del sistema dinámico

Bloque	Diagrama	Elementos
Motor		J_m es la inercia del motor T es el torque aplicado al motor b_m es el coeficiente de fricción del motor

Reductor		k_{g1} es la rigidez del reductor b_{g1} es el coeficiente de fricción del reductor J_{g1} es la inercia del reductor G_1 es la relación de reducción N_{out} es el número de vueltas en la salida del reductor N_{in} es el número de vueltas en la entrada del reductor
Eslabón 1		k_{c1} es la rigidez del eslabón b_{c1} es el coeficiente de fricción del eslabón 1 J_{c1} es la inercia del eslabón 1

Con lo antes descrito es posible dibujar un diagrama esquemático que represente ambos grados de libertad del brazo robot. De acuerdo con la técnica de parámetros agrupados se obtuvieron las ecuaciones diferenciales que definen el comportamiento del sistema.

2.2 Diagrama de parámetros agrupados

Para el desarrollo del modelo dinámico se realizó el diagrama de la Figura 2.1, en el cual ya se considera ambos grados de libertad, se revisó de manera analítica el desarrollo matemático a fin de obtener las ecuaciones del modelo dinámico de forma correcto.

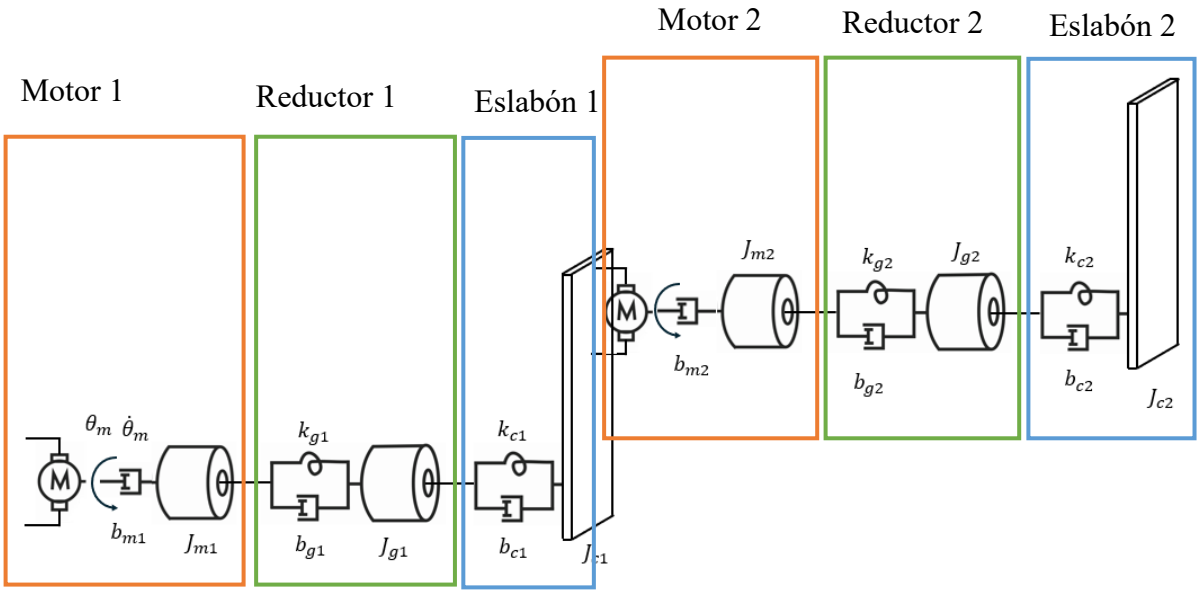


Figura 2.1: Diagrama de parámetros agrupados.

Una vez que se obtuvo el diagrama de parámetros agrupados, el siguiente paso es usar la ecuación generalizada de Lagrange tal como se muestra a continuación.

2.3 Ecuación generalizada de Lagrange

La ecuación de Euler-Lagrange generalizada está representada por

$$\frac{d}{dt}\frac{\partial T^*}{\partial \dot{q}_i} - \frac{\partial T^*}{\partial q_i} = -\frac{\partial D}{\partial \dot{q}_i} - \frac{\partial V}{\partial q_i} + e_i^s, \tag{2}$$

Donde T^* es la energía cinética del sistema dinámico

D son las energías disipativas del sistema dinámico

V es la energía potencial del sistema dinámico

e son los trabajos virtuales del sistema dinámico

2.4 Ecuaciones del modelo dinámico

La energía cinética y potencial del sistema dependen del tipo de movimiento del sistema, el cual puede ser traslacional o rotacional, en las Tablas 2.2 y 20.3 se muestran las ecuaciones para calcular la energía cinética de cada elemento, los cuales son resortes torsionales, amortiguadores torsionales, y los elementos mecánicos involucrados en el sistema, como son los eslabones y los acopladores [30].

Considerando el esquema de un brazo robot manipulador de 2 gdl tal como se muestra en la Figura 2.2. Se usará la posición inicial sobre el eje -y del plano. Usando el siguiente diagrama de la Figura 2.2 y considerando las diferentes longitudes de los centros de masa y los eslabones, así como las masas correspondientes podemos iniciar el análisis cinemático para encontrar las ecuaciones correspondientes al modelo dinámico.

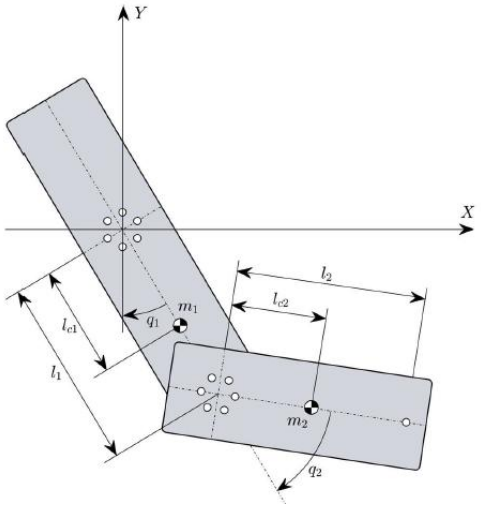


Figura 2.2: Diagrama del brazo robot de 2 gdl [3].

Donde $\theta_{c1} = q_1$ y $\theta_{c2} = q_2$, estos ángulos son considerados a partir del eje -y del plano.

Usando el diagrama de parámetros agrupados se pueden identificar los parámetros del sistema, los cuales se muestran en la Tabla 2.4

Tabla 2.4: Parámetros del sistema.

Parámetro	Significado	Unidades
$\boldsymbol{J_{m_n}}$	Inercia del motor	$Kg \cdot m^2 / rad$
$\boldsymbol{b_{m_n}}$	Coeficiente de fricción del motor	$N \cdot m \cdot s/rad$
$\boldsymbol{b_{g_n}}$	Coeficiente de fricción del reductor	$N \cdot m \cdot s/rad$
$\boldsymbol{b_{c_n}}$	Coeficiente de fricción del eslabón	$N \cdot m \cdot s/rad$
$\boldsymbol{J_{g_n}}$	Inercia del reductor	$N \cdot m \cdot s/rad$
$\boldsymbol{k_{g_n}}$	Rigidez del reductor	$N \cdot m/rad$
$\boldsymbol{k_{c_n}}$	Rigidez del eslabón	$N \cdot m/rad$

Usando el diagrama de parámetros agrupados de la Figura 2.1 se pueden identificar todas las variables que están involucradas en el modelo dinámico, las cuales se muestran en la Tabla 2.5 con su respectivo significado.

Tabla 2.5: Variables del sistema.

Variables	Unidades
$\boldsymbol{\theta_{g_n}}$ **es la posición del reductor**	rad
$\boldsymbol{\theta_{m_n}}$ **es la posición del eje del motor**	rad
$\boldsymbol{\theta_{c_n}}$ **es la posición del primer eslabón**	rad
$\boldsymbol{\dot{\theta}_{m_n}}$ **es la velocidad del eje del motor**	rad/s
$\boldsymbol{\dot{\theta}_{g_n}}$ **es la velocidad del eje del reductor**	rad/s
$\boldsymbol{\dot{\theta}_{c_n}}$ **es la velocidad del primer eslabón**	rad/s
$\boldsymbol{\ddot{\theta}_{m_n}}$ **es la aceleración del eje del motor**	rad/s^2
$\boldsymbol{\ddot{\theta}_{g_n}}$ **es la aceleración del eje del reductor**	rad/s^2
$\boldsymbol{\ddot{\theta}_{c_n}}$ **es la aceleración del primer eslabón**	rad/s^2

Con $n = 1{,}2$ correspondiente al primer o segundo grado de libertad.

La relación de reducción del reductor está dada por

$$G = \frac{Entrada}{Salida} = \frac{131.25}{1} \tag{3}$$

Usando la ecuación general de movimiento de Lagrange

$$\frac{d}{dt}\left[\frac{\partial Ec}{\partial \dot{\theta}}\right] - \left[\frac{\partial Ec}{\partial \theta}\right] + \left[\frac{\partial Ep}{\partial \theta}\right] + \left[\frac{\partial Ed}{\partial \dot{\theta}}\right] = \frac{\partial}{\partial q}(\delta w - f_R) \tag{4}$$

donde

E_c es la energía cinética total del sistema.

E_p es la energía potencial total del sistema.

E_d es la energía disipativa total del sistema.

θ es la variable sobre la cual se aplica la ecuación $\theta_{m_n}, \theta_{g_n}, \theta_{c_n}$ con $n = 1,2$

Se calculan la energía cinética, potencial y disipativa del sistema usando la Tabla 2.3 se tienen las siguientes ecuaciones

$$\begin{aligned} E_c = {} & \frac{1}{2}J_{m1}(\dot{\theta}_{m1})^2 + \frac{1}{2}J_{g1}(\dot{\theta}_{g1})^2 + \frac{1}{2}J_{c1}(\dot{\theta}_{c1})^2 + \frac{1}{2}J_{m2}(\dot{\theta}_{m2})^2 + \frac{1}{2}J_{g2}(\dot{\theta}_{g2})^2 + \frac{1}{2}J_{c2}(\dot{\theta}_{c2})^2 \\ & + \frac{1}{2}m_1 l_{c1}^2 \dot{\theta}_{c1}^2 + \frac{1}{2}[l_1^2 + l_{c2}^2 + 2l_1 l_{c2}\cos(\theta_{c2})]\dot{\theta}_{c1}^2 \\ & + \frac{1}{2}[2l_1 l_{c2}\cos(\theta_{c2}) + 2l_{c2}^2]\dot{\theta}_{c1}\dot{\theta}_{c2} + \frac{1}{2}m_2 l_{c2}^2 \dot{\theta}_{c2}^2 \end{aligned} \tag{5}$$

$$\begin{aligned} E_p = {} & \frac{1}{2}K_{g1}(\theta_{m1} - G\theta_{g1})^2 + \frac{1}{2}K_{c1}(\theta_{g1} - G\theta_{c1})^2 + \frac{1}{2}K_{g2}(\theta_{m2} - G\theta_{g2})^2 + \frac{1}{2}K_{c2}(\theta_{g2} - G\theta_{c2})^2 \\ & + m_1 g l_{c1}[1 - \cos(\theta_{c1})] + m_2 g[l_1 + l_{c2} - l_1\cos(\theta_{c1}) - l_{c2}\cos(\theta_{c1} + \theta_{c2})] \end{aligned} \tag{6}$$

$$\begin{aligned} E_d = {} & \frac{1}{2}B_{m1}\dot{\theta}_{m1}^2 + \frac{1}{2}B_{g1}(G\dot{\theta}_{g1} - \dot{\theta}_{m1})^2 + \frac{1}{2}B_{c1}(\dot{\theta}_{c1} - \dot{\theta}_{g1})^2 + \frac{1}{2}B_{g2}(G\dot{\theta}_{g2} - \dot{\theta}_{m2})^2 \\ & + \frac{1}{2}B_{c2}(\dot{\theta}_{c2} - \dot{\theta}_{g2})^2 \end{aligned} \tag{7}$$

Usando la ecuación de movimiento se obtienen las ecuaciones correspondientes del modelo dinámico

$$J_{m1}\ddot{\theta}_{m1} + K_{g1}(\theta_{m1} - G\theta_{g1}) + B_{m1}\dot{\theta}_{m1} - B_{g1}(G\dot{\theta}_{g1} - \dot{\theta}_{m1}) = \tau_1 \tag{8}$$

$$J_{g1}\ddot{\theta}_{g1} - GK_{g1}(\theta_{m1} - G\theta_{g1}) + K_{c1}(\theta_{g1} - \theta_{c1}) + B_{g1}G(G\dot{\theta}_{g1} - \dot{\theta}_{m1}) - B_{c1}(\dot{\theta}_{c1} - \dot{\theta}_{g1}) = 0 \tag{9}$$

$$\left[J_{c1} + J_{c2} + m_1 l_{c1}^2 + m_2[l_1^2 + l_{c2}^2 + l_1 l_{c2}\cos(\theta_{c2})]\right]\ddot{\theta}_{c1} - 2m_2 l_2 l_{c2} sen(\theta_{c2})\dot{\theta}_{c1}\dot{\theta}_{c2}$$

$$- m_2[l_1 l_{c2} sen(\theta_{c2})]\dot{\theta}_{c2}^2 + m_2[l_1 l_{c2}\cos(\theta_{c2} + l_{c2}^2 + J_{c2}]\ddot{\theta}_{c2} + 0$$

$$+ K_{c1}(\theta_{g1} - \theta_{c1}) + m_1 g l_{c1} sen(\theta_{c1}) + m_2 g l_1 sen(\theta_{c1})$$

$$+ m_2 g\, l_{c2} sen(\theta_{c1} + \theta_{c2}) + B_{c1}(\dot{\theta}_{c1} - \dot{\theta}_{g1}) = 0 \qquad (10)$$

$$J_{m2}\ddot{\theta}_{m2} + K_{g2}(\theta_{m2} - G\theta_{g2}) + B_{m2}\dot{\theta}_{m2} - B_{g2}(G\dot{\theta}_{g2} - \dot{\theta}_{m2}) = \tau_2 \qquad (11)$$

$$J_{g2}\ddot{\theta}_{g2} - GK_{g2}(\theta_{m2} - G\theta_{g2}) + K_{c2}(\theta_{g2} - \theta_{c2}) + GB_{g2}(G\dot{\theta}_{g2} - \dot{\theta}_{m2}) - B_{c2}(\dot{\theta}_{c2} - \dot{\theta}_{g2}) = 0 \qquad (12)$$

$$\left[m_2[l_1 l_{2c}\cos(\theta_{c2}) + m_2 l_{c2}^2 + J_{c2}]\right]\ddot{\theta}_{c1} + [J_{c2} + m_2 l_{c2}^2]\ddot{\theta}_{c2} + -m_2 l_1 l_{c2} sen(\theta_{c2})\dot{\theta}_{c1}^2$$

$$+ K_{c2}(\theta_{g2} - \theta_{c2}) + m_2 g l_{c2}\sin(\theta_{c1} + \theta_{c2}) + B_{c2}(\dot{\theta}_{c2} - \dot{\theta}_{g2}) = 0 \qquad (13)$$

Usando las ecuaciones 8, 9, 10, 11, 12, 13 es posible escribir el sistema usando matrices para obtener una representación más clara y manipulable (Ecuación 14)

$$\overbrace{\begin{bmatrix} J_{m1} & 0 & 0 & 0 & 0 & 0 \\ 0 & J_{g1} & 0 & 0 & 0 & 0 \\ 0 & 0 & I_{11} & 0 & 0 & I_{12} \\ 0 & 0 & 0 & J_{m2} & 0 & 0 \\ 0 & 0 & 0 & 0 & J_{g2} & 0 \\ 0 & 0 & I_{21} & 0 & 0 & I_{22} \end{bmatrix}}^{M} \begin{bmatrix} \ddot{\theta}_{m1} \\ \ddot{\theta}_{g1} \\ \ddot{\theta}_{c1} \\ \ddot{\theta}_{m2} \\ \ddot{\theta}_{g2} \\ \ddot{\theta}_{c2} \end{bmatrix}$$

$$+ \overbrace{\begin{bmatrix} B_{m1} + B_{g1} & -GB_{g1} & 0 & 0 & 0 & 0 \\ -GB_{g1} & B_{c1} + G^2 B_{g1} & -B_{c1} & 0 & 0 & 0 \\ 0 & -B_{c1} & B_{c1} & 0 & 0 & 0 \\ 0 & 0 & 0 & B_{m2} + B_{g2} & -GB_{g2} & 0 \\ 0 & 0 & 0 & -GB_{g2} & B_{c2} + G^2 B_{g2} & -B_{c2} \\ 0 & 0 & 0 & 0 & -B_{c2} & B_{c2} \end{bmatrix}}^{B} \begin{bmatrix} \dot{\theta}_{m1} \\ \dot{\theta}_{g1} \\ \dot{\theta}_{c1} \\ \dot{\theta}_{m2} \\ \dot{\theta}_{g2} \\ \dot{\theta}_{c2} \end{bmatrix}$$

$$+\overbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2m_2 l_1 l_{c2} sen(\theta_{c2})\dot{\theta}_{c2} & 0 & -m_2 l_1 l_{c2} sen(\theta_{c2})\dot{\theta}_{c2} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & m_2 l_1 l_{c2} sen(\theta_{c2})\dot{\theta}_{c1} & 0 & 0 \end{bmatrix}}^{C} \begin{bmatrix} \dot{\theta}_{m1} \\ \dot{\theta}_{g1} \\ \dot{\theta}_{c1} \\ \dot{\theta}_{m2} \\ \dot{\theta}_{g2} \\ \dot{\theta}_{c2} \end{bmatrix}$$

$$+\overbrace{\begin{bmatrix} K_{g1} & -GK_{g1} & 0 & 0 & 0 & 0 \\ -GK_{g1} & G^2K_{g1} + K_{c1} & -K_{c1} & 0 & 0 & 0 \\ 0 & -K_{c1} & K_{c1} & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{g2} & -GK_{g2} & 0 \\ 0 & 0 & 0 & -GK_{g2} & G^2K_{g2} + K_{c2} & -K_{c2} \\ 0 & 0 & 0 & 0 & -K_{c2} & K_{c2} \end{bmatrix}}^{K} \begin{bmatrix} \theta_{m1} \\ \theta_{g1} \\ \theta_{c1} \\ \theta_{m2} \\ \theta_{g2} \\ \theta_{c2} \end{bmatrix}$$

$$+\overbrace{\begin{bmatrix} 0 \\ 0 \\ m_1 g l_{c1} sen(\theta_{c1}) + m_2 g[l_{c1} sen(\theta_{c1}) + l_{c2} sen(\theta_{c1} + \theta_{c2})] \\ 0 \\ 0 \\ m_2 g l_{c2} sen(\theta_{c1} + \theta_{c2}) \end{bmatrix}}^{g} = \overbrace{\begin{bmatrix} \tau_{m1} \\ 0 \\ 0 \\ \tau_{m2} \\ 0 \\ 0 \end{bmatrix}}^{\tau} \qquad (14)$$

con

$$I_{11} = m_1 l_{c1}^2 + J_{c1} + J_{c2} + m_2 l_1^2 + m_2 l_{c2}^2 + 2m_2 l_1 l_{c2} \cos(\theta_{c2}) \qquad (15)$$
$$I_{12} = m_2 l_1 l_{c2} \cos(\theta_{c2}) + m_2 l_{c2}^2 + J_{c2} \qquad (16)$$
$$I_{21} = m_2 l_1 l_{c2} \cos(\theta_{c2}) + m_2 l_{c2}^2 + J_{c2} \qquad (17)$$
$$I_{22} = J_{c2} + m_2 l_{c2}^2 \qquad (18)$$

El sistema anterior también puede escribirse de la siguiente forma

$$\ddot{q} = M^{-1}[\tau - C(q, \dot{q}) - B\dot{q} - g(q) - K(q)] \qquad (19)$$

Donde $M, K, C, \in \mathbb{R}^{nxn}, g(q) \in \mathbb{R}^n$

El sistema anterior también es posible escribirlo en espacio de estados como se muestra en la siguiente tabla 2.6.

Tabla 2.6: Variables de estado del sistema

$x_1 = \theta_{m1}$ $x_2 = \theta_{g1}$ $x_3 = \theta_{c1}$ $x_4 = \theta_{m2}$ $x_5 = \theta_{g2}$ $x_6 = \theta_{c2}$	$x_7 = \dot{\theta}_{m1}$ $x_8 = \dot{\theta}_{g1}$ $x_9 = \dot{\theta}_{c1}$ $x_{10} = \dot{\theta}_{m2}$ $x_{11} = \dot{\theta}_{g2}$ $x_{12} = \dot{\theta}_{c2}$	$\dot{x}_1 = \dot{\theta}_{m1}$ $\dot{x}_2 = \dot{\theta}_{g1}$ $\dot{x}_3 = \dot{\theta}_{c1}$ $\dot{x}_4 = \dot{\theta}_{m2}$ $\dot{x}_5 = \dot{\theta}_{g2}$ $\dot{x}_6 = \dot{\theta}_{c2}$	$\dot{x}_7 = \ddot{\theta}_{m1}$ $\dot{x}_8 = \ddot{\theta}_{g1}$ $\dot{x}_9 = \ddot{\theta}_{c1}$ $\dot{x}_{10} = \ddot{\theta}_{m2}$ $\dot{x}_{11} = \ddot{\theta}_{g2}$ $\dot{x}_{12} = \ddot{\theta}_{c2}$

El sistema descrito por la ecuación 19 se convierte a espacio de estados, de esta forma es posible resolverlo usando la función "ode45" de Matlab, la cual nos permite solucionar las ecuaciones diferenciales correspondientes al sistema, y así conocer la respuesta a diferentes entradas y condiciones iniciales.

$$\begin{bmatrix} \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \\ \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \end{bmatrix} = M^{-1}\left[-C\begin{bmatrix} x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} - B\begin{bmatrix} x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} - g - K\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix}\right] \qquad (20)$$

Donde $M, K, C \in \mathbb{R}^{nxn}, g \in \mathbb{R}^n$

Control de posición PD con función tangente hiperbólica

El esquema de control aplicado en este trabajo es un control PD con tangente hiperbólica [16] y está descrito en el diagrama de la Figura 2.3, se decidió usar este control ya que es un control saturado, es decir el valor máximo está limitado a un valor fijo, que en este caso es el valor de torque máximo aplicado a cada motor.

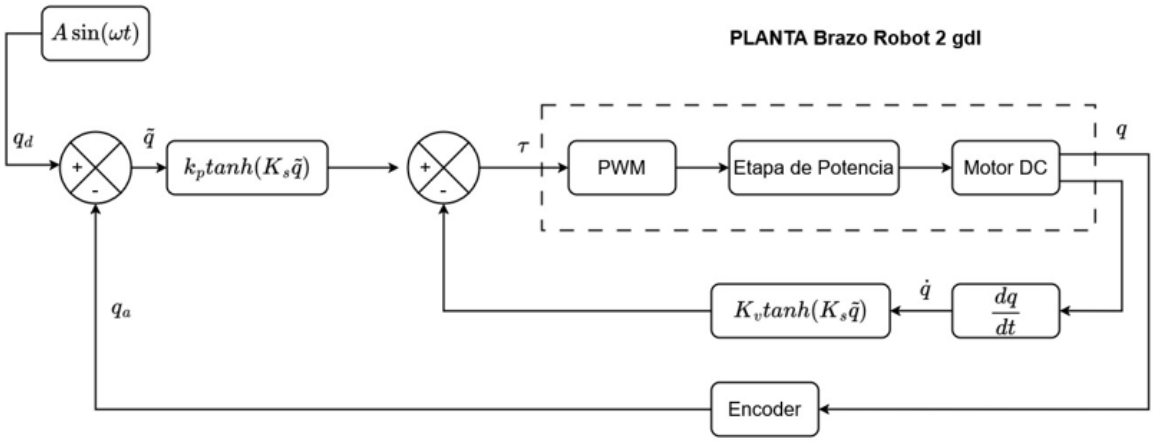


Figura 2.3: Lazo cerrado de control de posición con retroalimentación de posición y de velocidad.

$$\tau = K_p \tanh(K_s\tilde{q}) - K_v \tanh\left(K_s\tilde{\dot{q}}\right) \tag{21}$$

donde:

K_p es la ganancia proporcional

K_v es la ganancia derivativa

K_s es la pendiente del control tangente hiperbólica

$\tilde{q}$ es el error de posición, es decir, $\tilde{q} = q_d - q_a$

$\tilde{\dot{q}}$ es el error de velocidad

Los valores usados en este trabajo son $K_p = .450$ y $K_s = .1$, para el primer grado de libertad y $K_p = .200$ y $K_s = .1$ para el segundo grado de libertad.

Se comenzará con esta ley de control, ya que el fin es tener un control de trayectoria de forma experimental, sin embargo, al no conocer los parámetros del brazo robot, no es posible realizar la compensación para implementarlo en el control del brazo robot, a continuación, se muestra el diagrama del control de trayectoria que se desea implementar.

Control de trayectoria

El control de trayectoria consiste en determinar una función para τ tal que las posiciones y velocidades de cada articulación sigan con precisión las posiciones y velocidades deseadas [16], en este trabajo se plantea una trayectoria senoidal como trayectoria deseada.

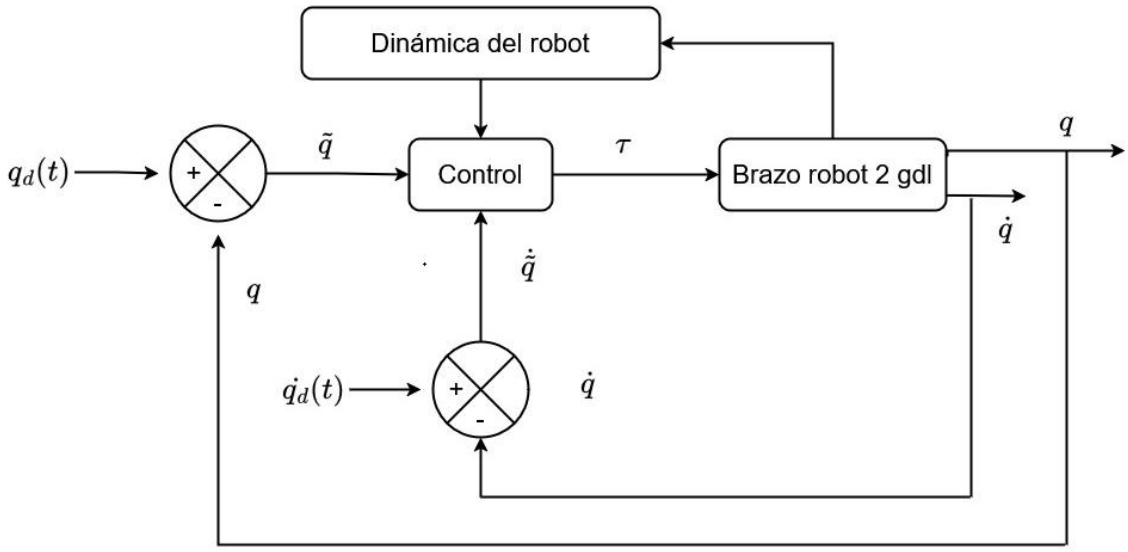


Figura 2.4: Diagrama a bloques del control de trayectoria de robots manipuladores

Donde $\tilde{\boldsymbol{q}}, \dot{\tilde{\boldsymbol{q}}} \in \mathbb{R}^n$ representan el error de posición definido como la resta entre la posición deseada $\boldsymbol{q}_d(t)$ y la posición actual del brazo robot $\boldsymbol{q}(t)$, igual a $\tilde{\boldsymbol{q}} = \boldsymbol{q}_d(t) - \boldsymbol{q}(t)$y el error de velocidad definido por $\dot{\tilde{\boldsymbol{q}}} = \dot{\boldsymbol{q}}_d(t) - \dot{\boldsymbol{q}}(t)$, que es la diferencia entre la velocidad deseada y la velocidad articular [Reyes].

El algoritmo de control planteado en este trabajo está el dado por la ecuación

$$\tau = K_p \tanh(K_s \tilde{\boldsymbol{q}}) - K_v \tanh\left(K_s \dot{\tilde{\boldsymbol{q}}}\right) + M(\boldsymbol{q})\ddot{\boldsymbol{q}}_{\boldsymbol{d}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})B\dot{\boldsymbol{q}} + g(\boldsymbol{q}) + K(\boldsymbol{q}) \quad (22)$$

Donde $M, K, C, \in \mathbb{R}^{nxn}, g(q) \in \mathbb{R}^n$ y fueron descritas por la ecuación (14)

Como se puede notar en la ecuación anterior, es necesario conocer las matrices $M, K, C, \in \mathbb{R}^{nxn}, g(q) \in \mathbb{R}^n$, por lo que es necesario conocer los parámetros para poder implementar el control de trayectoria, en los próximos capítulos se presentará la técnica de identificación paramétrica usando redes neuronales convolucionales usada para hallarlos.

Conclusiones

En este capítulo se llevó a cabo un análisis para poder obtener el modelo dinámico del brazo robot, el enfoque usado para obtenerlo fue usando la técnica de parámetros agrupados, la cual consiste en modelar el brazo robot usando cada elemento que compone cada grado de libertad, como son el motor, el reductor y el eslabón. Una vez que se obtiene el diagrama de parámetros agrupados lo siguiente fue usar la ecuación de Lagrange, ya que al desarrollarla se obtienen las ecuaciones del modelo dinámico del brazo robot.

Al usar la técnica de parámetros agrupados se consideran los esfuerzos generalizados del sistema, por lo que el modelo obtenido nos permite tener una representación más completa del sistema.

Con las ecuaciones del modelo dinámico correspondientes al brazo robot obtenidas es posible analizar su comportamiento con diferentes valores de los parámetros y para diferentes entradas del sistema. Con el modelo dinámico obtenido en este apartado se obtuvo el repositorio de imágenes de entrenamiento para la red neuronal correspondiente a cada grado de libertad. Usando el modelo dinámico en espacio de estados fue posible resolver este sistema usando “ode45” con lo cual se obtendrá la respuesta del sistema con diferentes parámetros.

Capítulo 3. Electrónica del brazo robot

La electrónica toma un papel muy importante en el control del robot, ya que sin esta solo se tendría un sistema mecánico que no se puede controlar, por lo que para poder tener el robot de manera funcional fue necesario tener toda la electrónica para su funcionamiento. En el diseño de la electrónica del brazo robot se consideró una parte de baja potencia y una de potencia, en las cuales se encuentra la adquisición de las señales, la cual nos permite mediante los encoders poder determinar la posición de los motores en todo momento, los circuitos de protección de los puentes h, los cuales se encargan de proteger al motor y finalmente los puentes H, los cuales son necesarios para hacer posible el cambio de dirección de giro y de entregar al motor la energía necesaria para mover ambos eslabones. A continuación, se hará una explicación de todos los circuitos y elementos electrónicos que fueron usados para tener el robot funcional, también se explicará la caracterización de los motores de corriente directa para volverlos de quasi transmisión directa.

3.1 Electrónica de baja potencia

Para evitar que en el puente H ambas ramas estén en funcionamiento al mismo tiempo se construyó el circuito de la Figura 3.1, este se diseñó como circuito de protección para los transistores del puente H, este se implementó en una placa perforada para después conectarla a los puentes H y la tarjeta FPGA

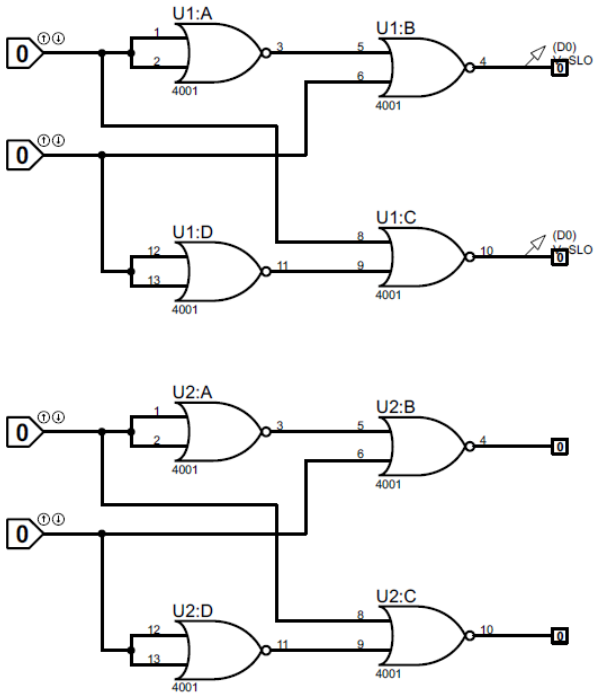


Figura 3.1: Circuito de protección de los puentes H.

En la siguiente Figura 3.2 se muestra la tarjeta diseñada usando el diagrama esquemático anterior.

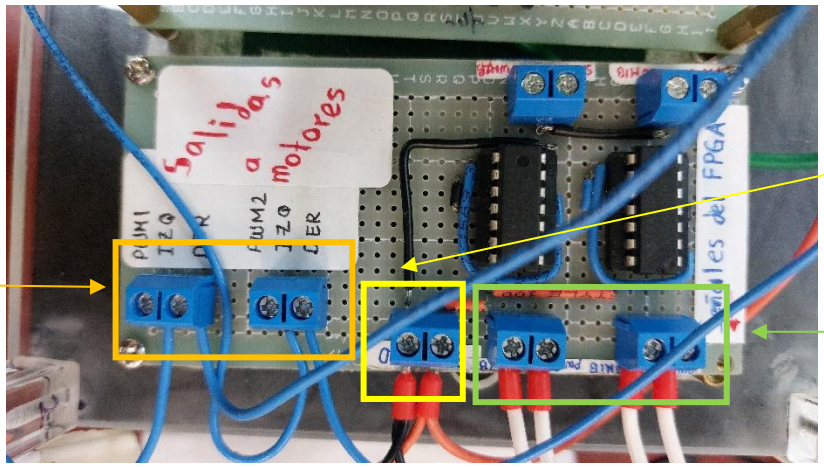


Figura 3.2: Tarjeta de protección para los puentes H.

3.2 Electrónica de potencia

El sistema cuenta con dos etapas, una etapa de potencia y una etapa para el acondicionamiento de las señales del motor. Para la etapa de potencia del robot, se cuenta con dos puentes H con transistores BJT para controlar el movimiento de cada motor Pololu GB37-131 (Figura 3.3) en cual corresponde a cada grado de libertad.

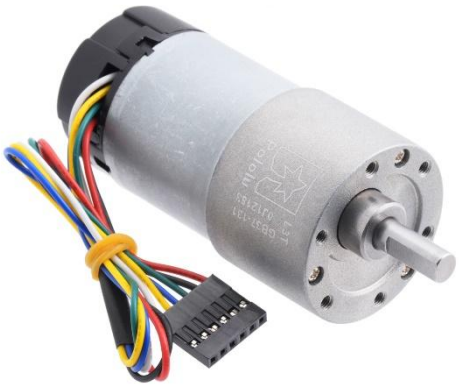


Figura 3.3: Motor GB37-131[17].

En la siguiente Tabla 3.1 se muestras las características de los motores usados, el motor cuenta con reductor metálico 131.25:1 y un encoder de cuadratura que provee 64 cuentas por revolución del eje del motor, lo cual corresponde a 8400 cuentas por revolución a la salida del reductor. Estos motores cuentan con un eje con un diámetro de 6 mm tipo D, lo que permite adaptarlo a un cople con estas especificaciones.

Tabla 3.1: Datos del motor GB37-131.

Voltaje	Corriente	Relación de transmisión	Velocidad sin carga (RPM)	Torque
12 V	5.5 A	131.25:1	76	45 kg*cm

Una vez que se cuenta con los datos del motor, el siguiente paso es obtener la caracterización de ambos motores, esto para obtener un comportamiento lineal en la respuesta del motor, a continuación, se mostrará la caracterización de ambos motores.

Es posible calcular la resolución máxima que puede ser leída, en este caso se calcula usando siguiente fórmula

$$\Delta\theta = \frac{360°}{8400} = 0.042857 \tag{23}$$

Donde 8400 es el número de cuentas totales por revolución de la salida del reductor.

3.3 Puente H

En la Figura 3.4 se muestra el diseño de los puentes H, estos fueron diseñados para las corrientes mayores a 5.5 A, ya que esta es la corriente máxima que consumen los motores según el fabricante. El diagrama esquemático de cada uno de los motores se muestra a continuación, se construyeron dos Puentes H usando tarjetas perforadas, esto se realizó por la facilidad del montaje, ya que adicionado a esto fue posible colocar tornillos separadores para el montaje de las tarjetas sobre la estructura de acrílico.

Una vez que se tuvieron los puentes H construidos, fue necesario realizar las conexiones necesarias al circuito de protección mencionado anteriormente, esto debido a que así se evita que ambas ramas del puente H se encuentren activas al mismo tiempo, y así evitar algún corto eléctrico.

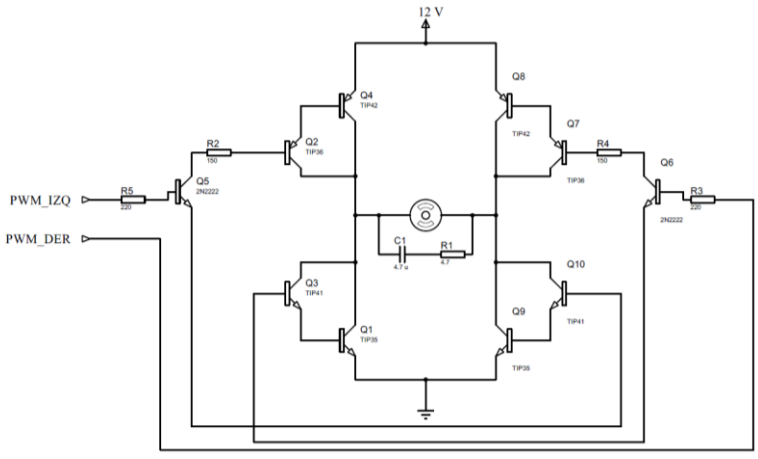


Figura 3.4: Diagrama esquemático de los puentes H.

3.4 Caracterización de los motores

Usando la metodología descrita a en este apartado se caracterizaron los dos motores correspondientes a los dos grados de libertad del brazo robot, la caracterización del motor se realizó con los componentes que se observan en la Figura 3.5, fue necesario contar con un generador de funciones para generar las señales de PWM, una fuente de voltaje de 10 A, y un circuito conmutador, fue necesario montar el motor sobre una estructura de aluminio para poder medir la fuerza con un dinamómetro para posteriormente hacer los cálculos necesarios para obtener el torque.

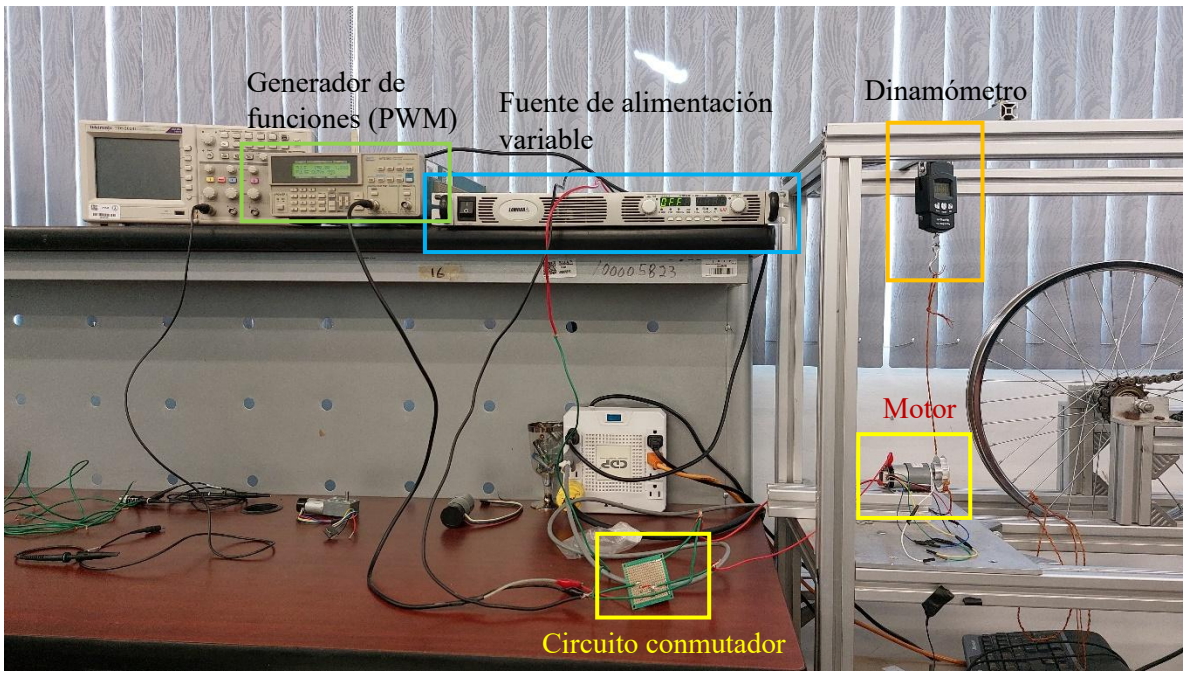


Figura 3.5: Caracterización del motor de corriente directa.

3.5 Metodología para la caracterización de los motores de CD

1. Primero se debe determinar el torque máximo del motor, para lo cual se realiza una variación del voltaje de alimentación con que se conecta el motor (sin exceder el límite de voltaje de trabajo recomendado en la hoja de datos).

2. Realizar el cálculo correspondiente para obtener la mitad del torque máximo al que trabaja el motor al alimentarlo con una señal de PWM al 50% del ciclo de trabajo. En orden de poder lograr este paso, es necesario contar con el programa que te genere la señal de variación de frecuencia del PWM, la cual debe ir desde los 50 Hz hasta la frecuencia determinada por el usuario.

3. El siguiente paso consiste en seleccionar las frecuencias o frecuencia que proporcione el valor que sea más cercano a la mitad del torque máximo que se calculó en el paso anterior.

4. Se debe probar cada una de las secuencias seleccionadas al ir variando el ciclo de trabajo del 10% al 90% de su rango. Esto se realiza sin variar la alimentación del motor, empleando para ello el voltaje del torque máximo. Cuando se varía el ciclo de trabajo, se debe determinar el torque del motor para cada variación del PWM y con los datos obtenidos graficar lo que corresponde a la respuesta del motor.

5. Al final, se deben revisar las gráficas obtenidas y analizarlas para determinar la ecuación de la curva característica. Con esto, se tiene que seleccionar la frecuencia que proporcione la respuesta más cercana posible a un comportamiento lineal para el motor. [17,18]

Una vez realizados los pasos anteriores se obtiene una gráfica similar a la Figura 3.7, una vez hecho esto es necesario obtener la ecuación de la recta que más se aproxima a la curva obtenida, esto para tener la relación entre el voltaje vs el torque del motor.

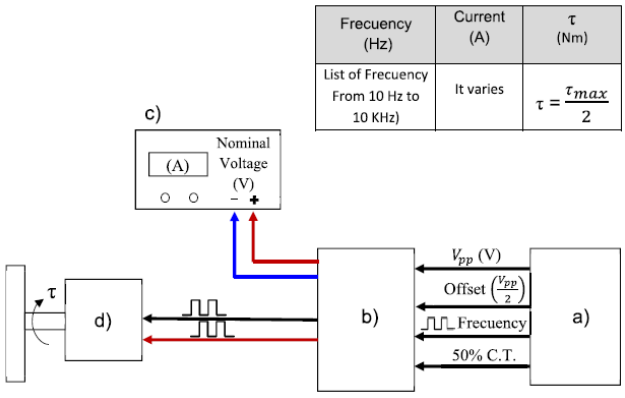


Figura 3.6: Ejemplo de la configuración experimental para suministrar las frecuencias al motor. a) Generador de funciones, b) Puente H, c) Fuente de alimentación, d) Motor de DC [18].

Una vez que se hizo todo el proceso anterior se tendrá la caracterización del motor, con esto tendremos un motor de quasi transmisión directa.

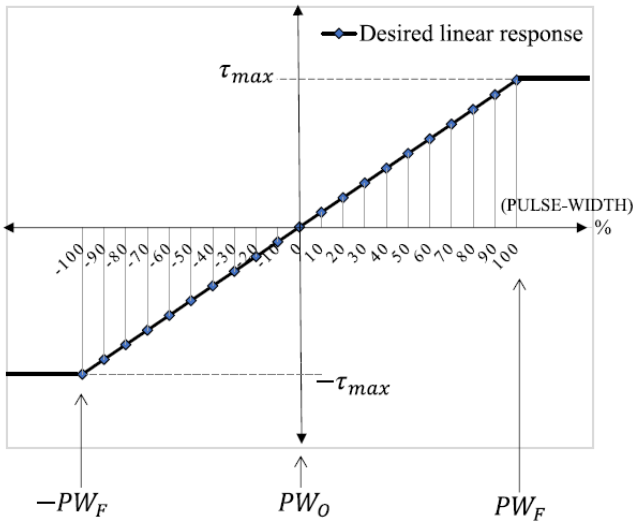


Figura 3.7: Comportamiento deseado del torque del motor [18].

Se caracterizaron los motores correspondientes a cada grado de libertad siguiendo la metodología antes explicada, a continuación, se muestran los resultados obtenidos

Motor 1

La medición del torque se realizó usando un dinamómetro y una polea con un diámetro de 1.8 cm. Se realizó la variación de voltaje aplicado al motor 1, variándolo de 0-12 V, se obtuvieron los datos mostrados en la Tabla 3.2, los cuales muestran los valores de torque obtenidos por cada medición.

Tabla 3.2: Torque obtenido con diferentes voltajes

Voltaje [V]	Corriente [A]	Fuerza [Kg]	Torque [Nm]
0	0	0	0
1	0.68	1.75	0.31759875
2	1.36	4.2	0.762237
3	2	5.2	0.943722
4	2.59	8.36	1.5172146
5	3.09	9.86	1.7894421
6	5.56	12.1	2.1959685
7	4.14	13.52	2.4536772
8	4.4	14.3	2.5952355
9	4.73	15.9	2.8856115
10	5.04	16.86	3.0598371
11	5.32	17.4	3.157839
12	5.4	17.65	3.20321025

Para hacer la conversión de Kg a Nm usó la siguiente ecuación

$$\tau = \frac{F * 9.81 m/s^2}{0.018\, m} \tag{24}$$

Una vez obtenida la tabla anterior se graficó para ver el comportamiento del torque respecto a voltaje aplicado al motor (Figura 3.8).

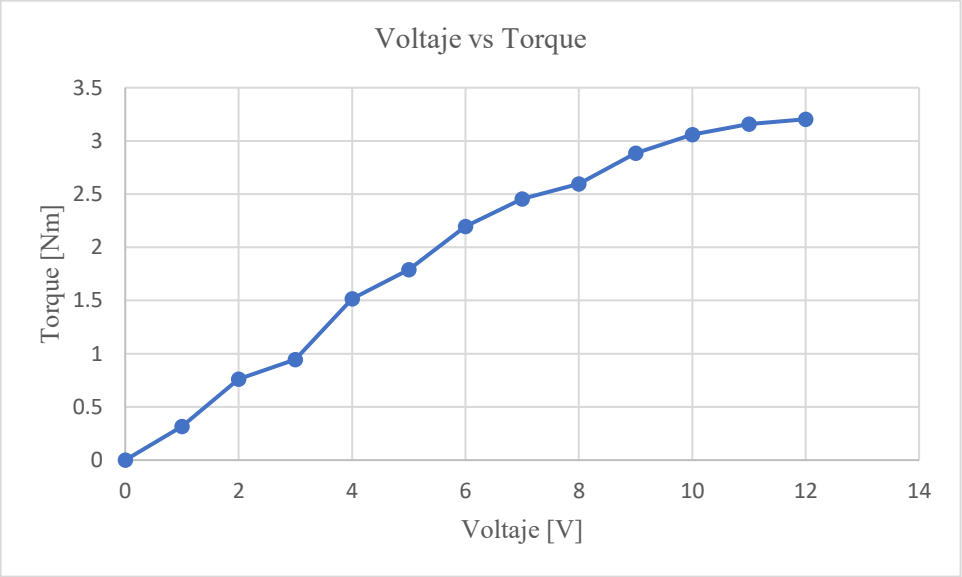


Figura 3.8: Voltaje vs Torque Motor 1.

Una vez que se obtuvo la tabla 3.2 anterior se seleccionó el 50% del valor de torque medido a 12 V con lo que se obtuvo un torque de 1.6 Nm, el siguiente paso fue buscar las frecuencias de PWM en las que al 50% del ciclo de trabajo se obtenga este torque.

Al realizar el barrido de frecuencias se obtuvo la siguiente Tabla 3.3

Tabla 3.3: Barrido de frecuencias

Frecuencia [Hz]	Torque [Nm]
20	2.31756345
50	2.19778335
100	2.1741903
500	1.0453536
1000	0.8747577
2000	0.78945975
5000	1.1796525
10000	1.51902945

Al realizar el barrido se encontró que las frecuencias que cumplen el criterio especificado anteriormente son 4.9 kHz y 352 Hz, por lo que se usaron estas frecuencias para el PWM de entrada usando el circuito mostrado anteriormente.

Tabla 3.4: Respuesta a un PWM con frecuencia de 352 Hz

% CT	Corriente [A]	Fuerza [Kg]	Torque [Nm]
0	0	0	0
10	0.1	0	0
20	0.34	2.13	0.38656305
30	0.69	5.26	0.9546111
40	1.11	7.9	1.4337315
50	1.96	9.6	1.742256
60	2.11	11	1.996335
70	2.9	12.39	2.24859915
80	3.82	13.65	2.47727025
90	4.2	15.08	2.7367938
100	6	18.9	3.4300665

Al graficar los datos obtenidos se obtuvo la gráfica de la Figura 3.9.

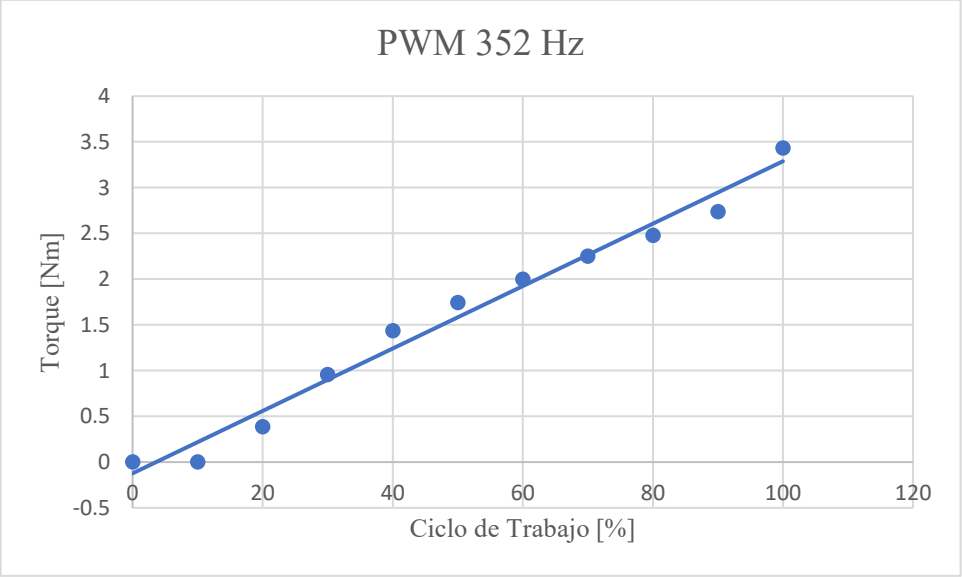


Figura 3.9: Respuesta a un PWM con frecuencia de 352 Hz.

Al usar la siguiente frecuencia de 4.9 KHz se obtuvo la siguiente Tabla 3.5, esta se construyó variando el ciclo de trabajo del PWM.

Tabla 3.5: Respuesta a un PWM con frecuencia de 4.9 kHz

% CT	Corriente [A]	Fuerza [Kg]	Torque [Nm]
0	0	0	0
10	0.27	1.97	0.35752545
20	0.47	3.56	0.6460866
30	0.68	5.86	1.0635021
40	0.92	6.8	1.234098
50	1.25	8.07	1.46458395
60	1.75	10.5	1.9055925
70	2.49	11.7	2.1233745
80	3.6	13.92	2.5262712
90	5	16	2.90376
100	6.25	18.4	3.339324

Con los datos obtenidos al variar el ciclo de trabajo se obtuvo la respuesta del torque al variar el ciclo de trabajo, en la Figura 3.10 se muestran los resultados obtenidos

Figura 3.10: Respuesta a un PWM con frecuencia de 4.9 kHz.

Observando la Figura 3.9 y la Figura 3.10, se decidió usar una frecuencia de 4.9 kHz para el primer motor.

Motor 2

Se realizó el mismo procedimiento antes mencionado para la caracterización del segundo motor. Se obtuvo la gráfica de voltaje vs torque, y una vez que se obtuvo, se buscó

la mitad del torque nominal, el cual fue de aproximadamente 1.26 N, de aquí se hizo el barrido para encontrar la frecuencia óptima, quedando 445 Hz y 5.09 kHz,

Tabla 3.6: Torque obtenido con diferentes voltajes

Voltaje [V]	Corriente [A]	Fuerza [Kg]	Torque [Nm]
0	0	0	0
1	0.3	0.56	0.1016316
2	0.79	1.86	0.3375621
3	1.33	3.14	0.5698629
4	2.42	4.4	0.798534
5	3.03	5.92	1.0743912
6	3.47	7.18	1.3030623
7	3.94	6.68	1.2123198
8	4.45	10.28	1.8656658
9	4.95	10.9	1.9781865
10	4.98	11.9	2.1596715
11	6.39	12.58	2.2830813
12	6.7	13.9	2.5226415

En la Figura 3.11 se muestran los resultados obtenidos al variar el voltaje de entrada aplicado al segundo motor.

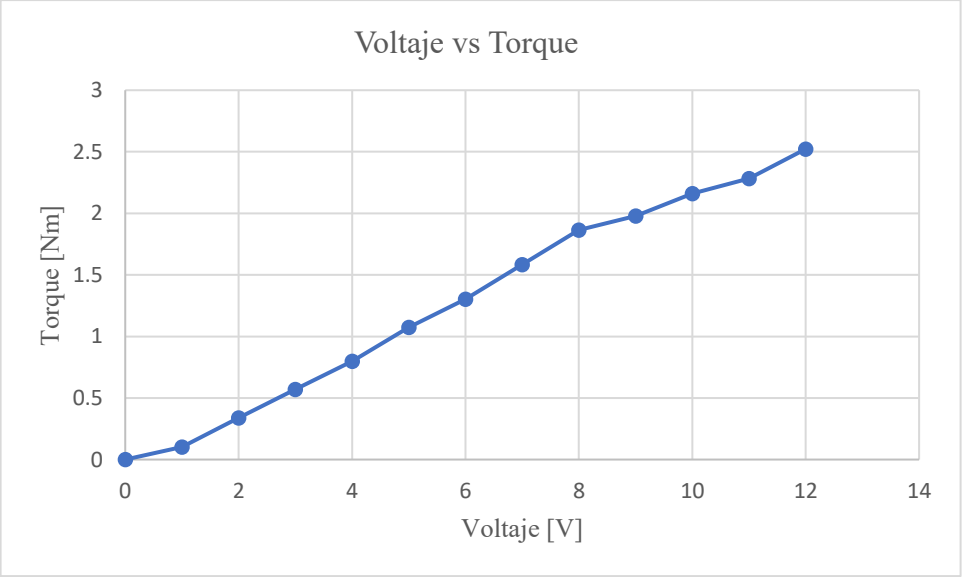


Figura 3.11: Voltaje vs Torque Motor 2.

Al variar el ciclo de trabajo a una frecuencia de 445 Hz se obtuvo la Tabla 3.7, en la que se muestran los valores de torque obtenidos con esta frecuencia.

Tabla 3.7: Respuesta a un PWM con frecuencia de 445 Hz

% CT	Corriente [A]	Fuerza [Kg]	Torque [Nm]
0	0	0	0
10	0	0	0
20	0.28	1.66	0.3012651
30	0.58	3.63	0.65879055
40	0.99	5.42	0.9836487
50	1.35	6.69	1.21413465
60	2.41	7.32	1.3284702
70	2.38	10	1.81485
80	2.88	10.5	1.9055925
90	4.5	11.14	2.0217429
100	6	13.9	2.5226415

Una vez que se obtuvieron los datos a esta frecuencia, se graficó para ver su comportamiento, los resultados obtenidos pueden verse en la Figura 3.12.

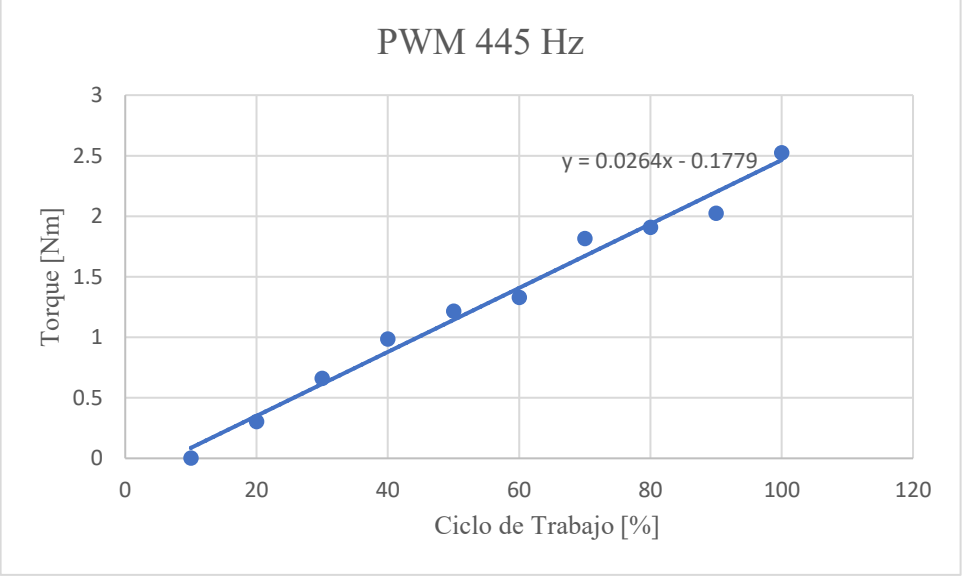


Figura 3.12: Respuesta a un PWM con frecuencia de 352 Hz.

Al usar la frecuencia la otra frecuencia de trabajo de 5.09 kHz se obtuvieron los datos de la Tabla 3.8, donde pueden observarse los torques obtenidos al variar el ciclo de trabajo.

Tabla 3.8: Respuesta a un PWM con frecuencia de 5.09 kHz

% CT	Corriente [A]	Fuerza [Kg]	Torque [Nm]
0	0	0	0
10	0.28	1.52	0.2758572
20	0.48	3.5	0.6351975
30	0.69	5.45	0.98909325
40	0.92	6.82	1.2377277
50	1.29	7.57	1.37384145
60	1.79	8.8	1.597068
70	2.54	9	1.633365
80	3.64	11.56	2.0979666
90	5.7	13.18	2.3919723
100	6.2	14.59	2.64786615

Al graficar los datos obtenidos se obtuvo la siguiente Figura 3.13.

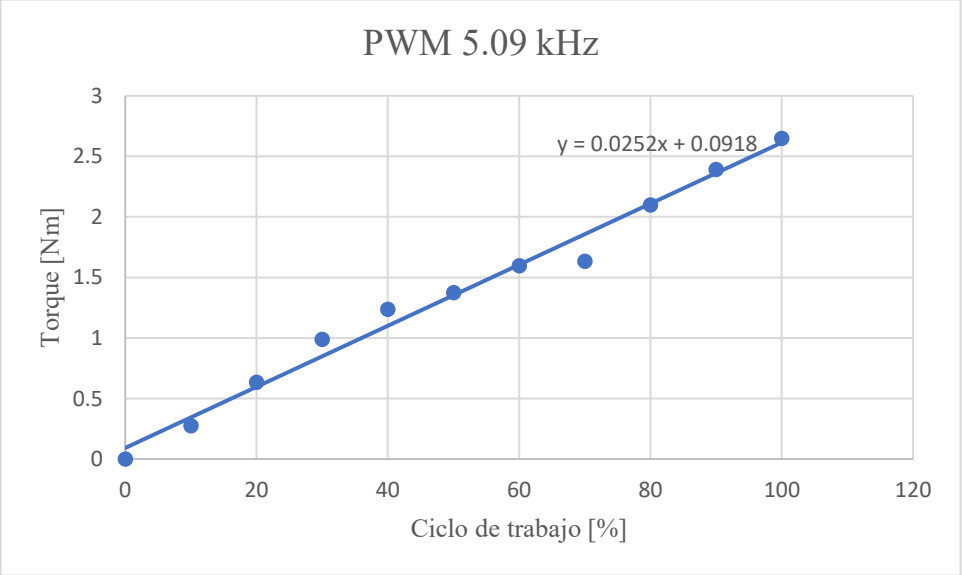


Figura 3.13: Respuesta a un PWM con frecuencia de 5.09 kHz.

Observando las gráficas de la Figura 3.12 y la Figura 3.13 para ambas frecuencias, se eligió la segunda ya que la respuesta del torque al variar el ancho de pulso se obtiene los puntos más aproximados a una recta, por lo que la frecuencia a usar para el segundo motor es de 5.09 kHz.

3.6 Actualización de la electrónica

Se construyó una estructura que contenga todo el sistema, se diseñó de manera que se puedan separar las conexiones del robot a la tarjeta, ya que resulta más práctico para hacer las pruebas del sistema, con las conexiones de esta forma solo es necesario realizar las conexiones de alimentación y unir ambos lados del conector, el plano de la estructura se mostró en un apartado anterior.

Se realizó una actualización en la electrónica, en la siguiente Figura 3.14 se puede observar todo el sistema, en la parte superior se observa la tarjeta FPGA, la parte media los puentes H junto con su circuito de protección para las salidas de PWM, y finalmente en la parte inferior se encuentra la fuente de alimentación de todo el sistema, la cual nos proporciona los voltajes necesarios para cada componente.

Para esto se realizaron las conexiones necesarias, para alimentar todo el sistema se decidió implementar una fuente de voltaje de PC ya que esta los voltajes a usar por el brazo robot, 3.3V para la alimentación para el FPGA, el circuito de protección y la alimentación de los encoders, en color blanco se muestran las señales que salen o entran del FPGA y para la alimentación de los puentes H se usaron 12V proporcionados igual por la misma fuente.

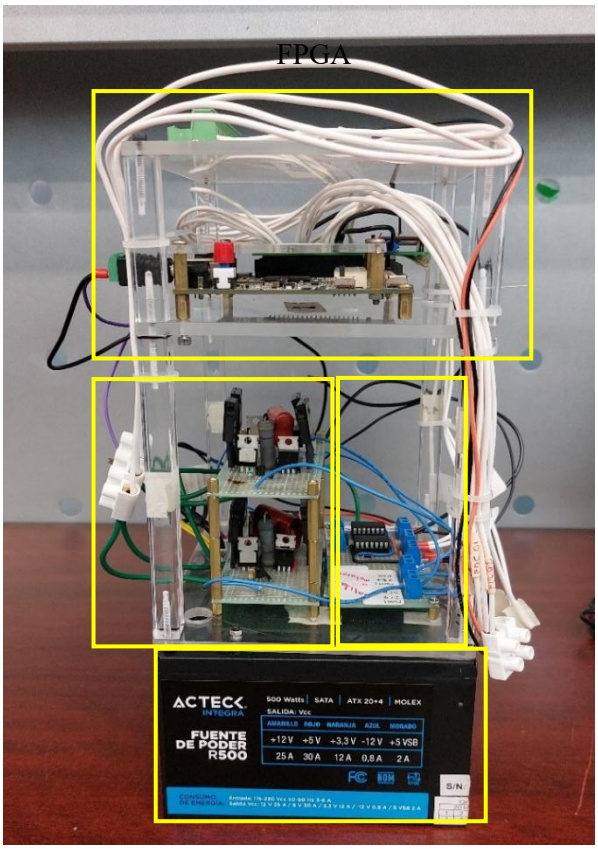


Figura 3.14: Componentes de la estructura de acrílico y la fuente de alimentación.

En la Tabla 3.9 se muestra el código de colores usados, estos fueron elegidos ya que un código de colores ayuda a reducir los errores que pudieran existir al realizar las conexiones eléctricas, además de que permite ubicar las conexiones de una manera más rápida y así poder realizar inspecciones de manera más rápida en caso de detectar una falla.

Tabla 3.9: Código de colores para los motores

Color	Significado
Negro	GND (22 AWG), Terminales de los motores
Rojo	Alimentación 5 (V), Terminales de los motores
Naranja	3.3V (CD)
Azul	Alimentación encoders
Verde	GND de los encoders
Amarillo	Señal A Encoder 1, 2
Blanco	Señal B Encoder 1, 2

Para realizar las conexiones entre los motores y el FPGA se usó un conector extraíble de 16 entradas, para esto se eligieron las terminales de acuerdo con el diagrama de conexiones de la Figura 3.15, en la terminal 1-4 se encuentras las señales A y B de ambos encoders, de 6-8 se encuentras los voltajes de alimentación 3.3 V, de 9-11 se encuentra GND de la alimentación de ambos encoders, y finalmente en las terminales 12-16 se encuentras las salidas de PWM, las cuales van conectadas directamente a los motores, el orden de esta distribución se eligió para evitar que las señales de alimentación de los motores creen ruido en las señales de los encoders.

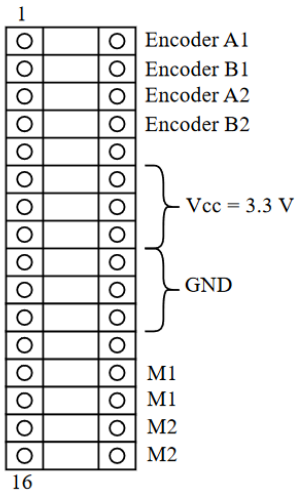


Figura 3.15: Diagrama de conexiones conector 16 entradas.

Usando el diagrama anterior se realizaron las conexiones pertinentes tal como se muestra en la Figura 3.16, una vez que se tienen este conector es posible separar la estructura que contiene la electrónica.

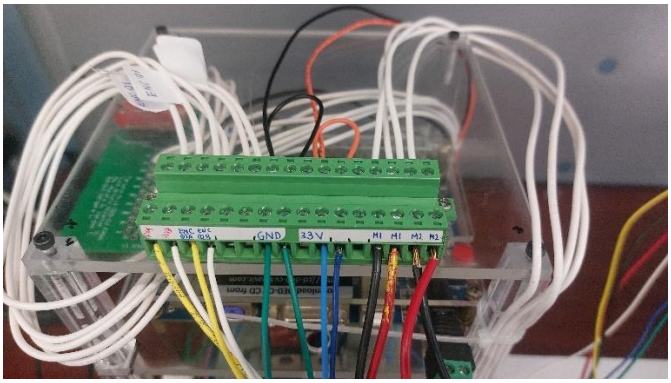
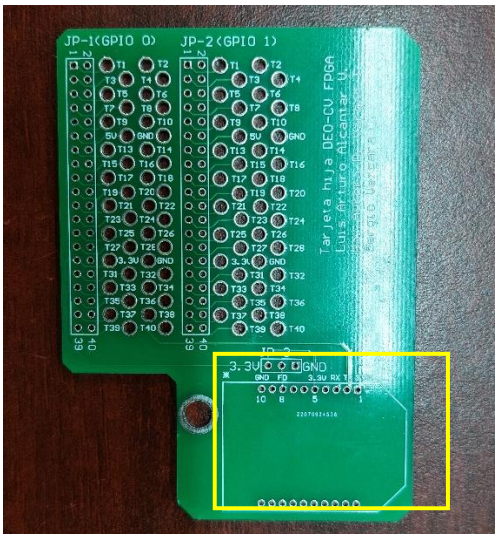


Figura 3.16: Conector entre el FPGA, el sistema eléctrico y mecánico.

También se realizó la conexión de la tarjeta hija (Figura 3.17), esta fue diseñada en la Maestría de Ciencias de la Electrónica, Opción Automatización [20], se soldaron los cables de acuerdo con la configuración descrita en la Tabla 5.2 y 5.3, para esto fue necesario colocar dos tiras de pines de 2x40 para poder acoplar la tarjeta hija con la tarjeta FPGA.



Espacio Módulo Wi-Fly

Figura 3.17: Tarjeta hija para la tarjeta FPGA.

En la Figura 3.18 se muestra la distribución de los pines de la tarjeta FPGA, usando esta imagen como referencia se soldaron los pines que son usados para el robot, entre estos se encuentras los puertos de entrada de las señales de los encoders, las salidas de PWM, y los fines de carrera, estos últimos fueron aterrizados a tierra, ya que no se implementaron, sin embargo, son necesarios para el funcionamiento del brazo robot.

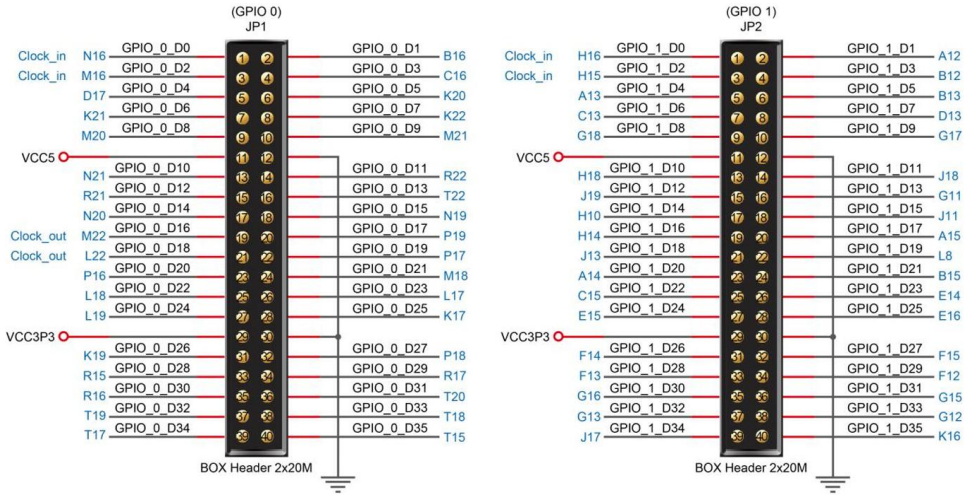


Figura 3.18: Distribución de pines de la Tarjeta DE0-CV.

Una vez que se soldaron los cables, se colocaron las tiras de pines en la parte inferior de la tarjeta para montarla sobre la tarjeta FPGA, también se soldaron dos tiras de pines hembra para montar el módulo Wi-Fly, en la Figura 3.19 se muestra el módulo Wi-Fly sin montar y conectada con jumpers, mientras que en la Figura 3.20 ya se muestra montada con sus respectivos cables dentro de la estructura de acrílico.

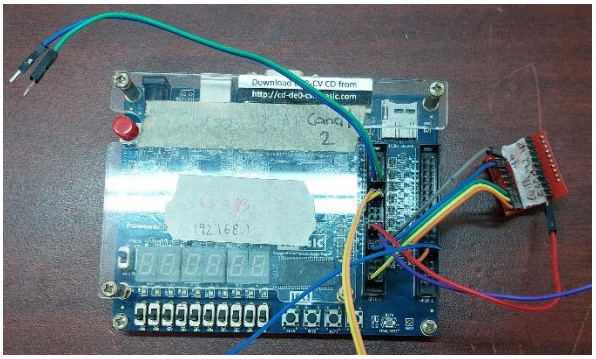


Figura 3.19: Conexión del módulo Wi-Fly con jumpers.

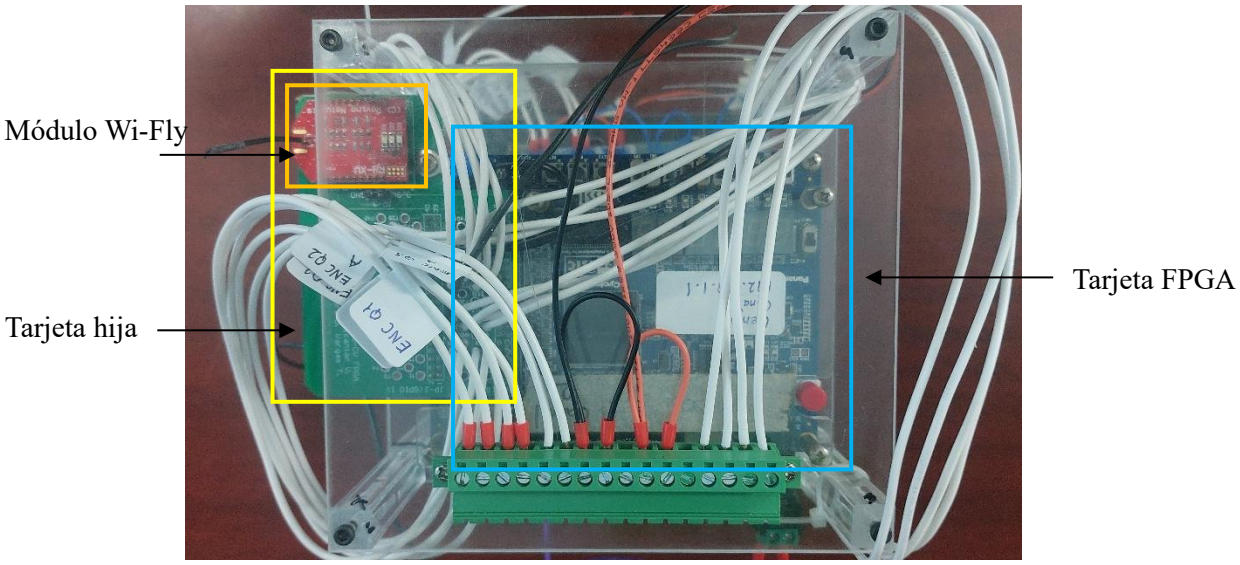


Figura 3.20: Tarjeta hija conectada con el módulo Wi-Fly y la tarjeta FPGA.

Conclusiones

En este capítulo se mostró la electrónica que conforma el brazo robot, la implementación de la interconexión de la parte eléctrica y la mecánica que fue realizada para tener el robot de manera funcional. El diseño de todos los elementos electrónicos que forman parte de la electrónica toma vital importancia en el control del brazo robot, los puentes H y de los circuitos de protección permiten que el robot pueda cambiar de dirección de giro para mover los eslabones, además de evitar que se activen las dos ramas del puente H, lo que provocaría un corto eléctrico en el motor haciendo que este pueda llegar a quemarse.

En la caracterización de ambos motores, las mediciones de torque permitieron obtener los torques máximo que puede proporcionar cada motor, tomando esto en cuenta fue posible seleccionar la mejor ubicación para cada motor, ya que el que proporciona el mayor torque fue seleccionado para el primer grado de libertad al ser el que tiene que cargar la masa del segundo grado de libertad, el cual está conformado por el segundo motor y eslabón.

También es importante destacar que usar la estructura de acrílico permite una mejor integración entre todos los elementos electrónicos del robot, permitiendo que se pueda usar un conector extraíble para el control de ambos motores, lo que permitió una conexión y desconexión eléctrica más fácil al usar el robot para pruebas, Gracias a esto fue posible realizar las pruebas experimentales para mover los dos grados de libertad del robot, en los próximos capítulos se mostrarán los resultados obtenidos, estos resultados fueron obtenidos con ayuda de la interfaz de usuario desarrollado en la Maestría de Ciencias de la Electrónica,el compilador de lenguaje D y el uso del módulo Wifly para la conexión inalámbrica entre el FPGA y la PC.

Capítulo 4. Firmware

En este capítulo se muestran todos los elementos que componen el firmware del brazo robot, este es el que permite la comunicación entre el hardware y el software, se muestran a continuación los usados para realizar esta comunicación.

4.1 CPU

El procesador usado fue diseñado en la Maestría en Ciencias de la Electrónica, este fue programado dentro de una tarjeta de desarrollo FPGA de Altera DE0 CV (Figura 4.1), una tarjeta de desarrollo que cuenta con diversos periféricos tales como puertos de propósito general, leds, display 7 segmentos, puerto PS/2, VGA, memoria RAM, entre otros. Las características principales se detallan en la Tabla 4.1.

La FPGA es un dispositivo de compuerta lógicas programables, es decir: dentro del empaquetado tenemos un arreglo de compuertas lógicas, mismas que son interconectadas unas con otras con el fin de desarrollar un proceso, ya sea combinacional o secuencial [24].

La idea de implementar una FPGA destaca en el hecho de que es posible implementar módulos de propósito específico (IP por sus siglas en inglés), es decir, firmware que desarrolle una tarea específica, pero además correr diversos de estos módulos en paralelo, esto a nivel sistemático es considerado como control en tiempo real. Para nuestro dispositivo se destinará parte de la FPGA en emular un microprocesador que se comunique con los módulos de propósito específico.

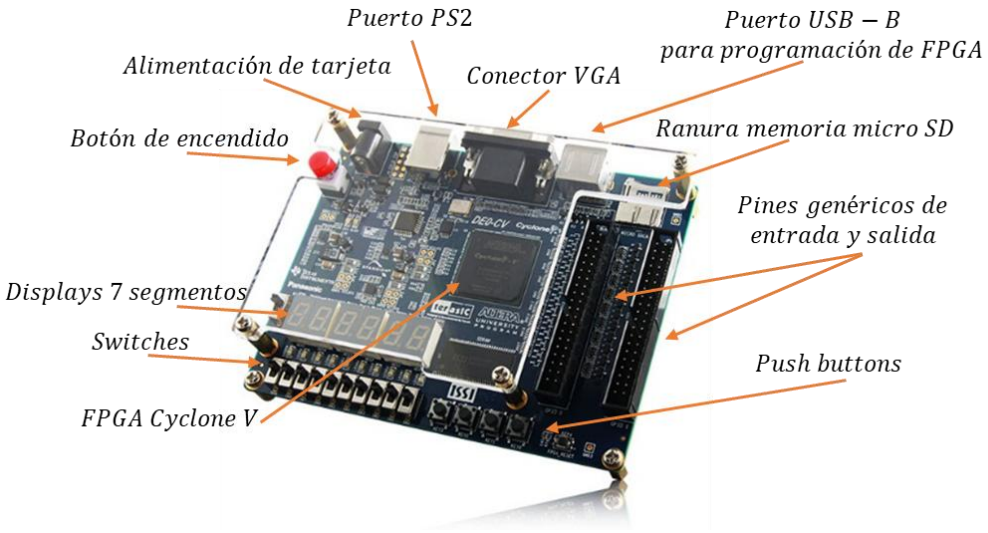


Figura 4.1: Tarjeta de desarrollo DE0-CV [24].

Tabla 4.1: Tabla de especificaciones de tarjeta de desarrollo implementada.

Procesador	Cyclone V 5CEBA4F23C7N de Altera.
Elementos lógicos disponibles	49000
Memoria SDRAM	64MB, en grupos de 16 bits.
Pines de propósito general	2 peinetas de 40 pines, de las cuales: 36 pines de entrada y salida, 2 de alimentación (3.3V y 5V) y 2 de tierra (GND).
Periféricos genéricos de entrada-salida	4 push buttons 10 dip switches 10 leds 6 displays siete (7) segmentos
Otros periféricos	Interfaz para LCD Touch Screen. Salida VGA. Ranura Micro SD. Ps2 / teclado / ratón.
Método de programación	Programable a través del software Quartus II y/o Nios II (Aplicaciones embebidas).

En la Figura 4.2, se puede ver el diagrama general con el que se cuenta, por lo que, para hacer uso de este, se procedió a obtener el diagrama para el brazo robot de 2 gdl.

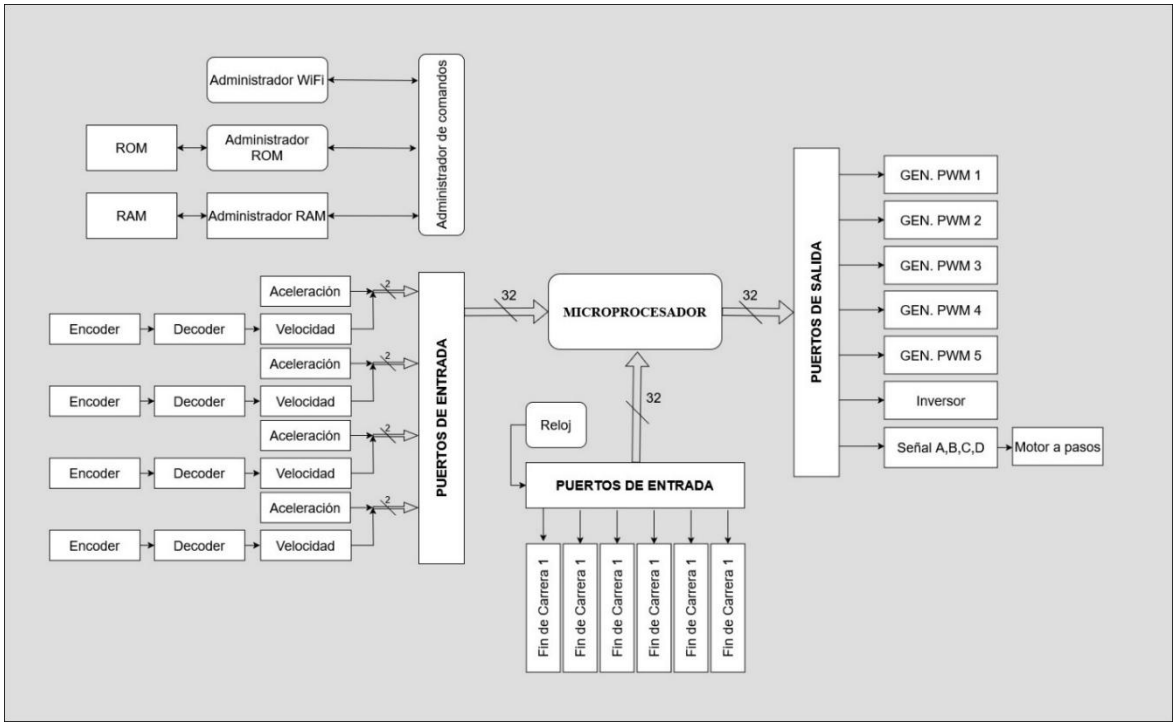


Figura 4.2: Arquitectura de microprocesador emulado dentro de FPGA.

El procesador otorgado por el laboratorio de robótica posee 16 puertos de entrada y 16 puertos de salida, cada puerto es de 32 bits y es posible acceder a ellos, ya sea para conectar dispositivos de forma externa, o en su caso, conectar módulos internos IP diseñados en VHDL o AHDL. La Tabla 4.2 representa el conjunto de puertos de entrada disponibles, donde el puerto cero está reservado para la entrada de datos derivada del módulo WIFI, mientras que la Tabla 4.3 representa los puertos de salida con los que cuenta el procesador para administrar recursos variados, de igual forma, el puerto cero es utilizado para la transferencia de datos a través del módulo WIFI. El bit 32 es un bit de signo por lo que típicamente éste no es utilizado cuando se envían datos tipo entero sin signo [20],[21],[22].

Tabla 4.2. Distribución de puertos de entrada.

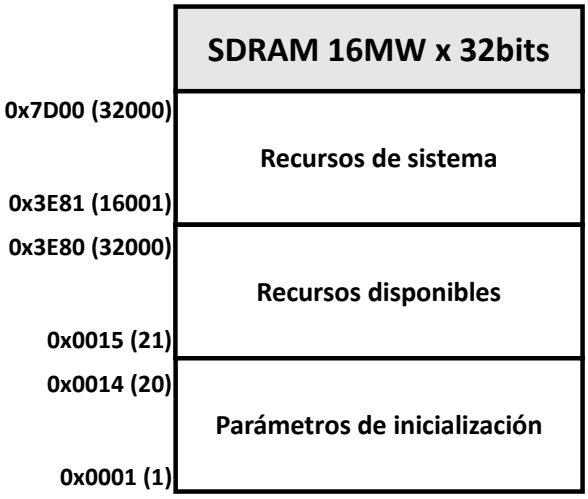
N° PTO	Disposición de bits de puerto	
PT0	No usar PT0[32]	Reservado para la lectura datos WIFI PT0[31-1]
PT1	No usar PT1[32]	Bits de usuario disponible PT1[31-1]
PT2	No usar PT2[32]	Bits de usuario disponible PT2[31-1]
.	.	.
PT15	No usar PT5[32]	Bits de usuario disponible PT15[31-1]
PT16	No usar PT6[32]	Bits de usuario disponible PT16[31-1]

Tabla 4.3. Distribución de puertos de salida.

N° PTO	Disposición de bits de puerto	
PT0	No usar PT0[32]	Reservado para la escritura datos WIFI PT0[31-1]
PT1	No usar PT1[32]	Bits de usuario disponible PT1[31-1]
PT2	No usar PT2[32]	Bits de usuario disponible PT2[31-1]
. . .	. . .	. . .
PT15	No usar PT5[32]	Bits de usuario disponible PT15[31-1]
PT16	No usar PT6[32]	Bits de usuario disponible PT16[31-1]

En la Tabla 4.4 se muestran los recursos con los que se cuenta, estos son importantes ya que es necesario conocerlos al momento de programar en lenguaje D.

Tabla 4.4 Recursos disponibles y mapeo de memoria RAM.



4.2 Mapeo de entradas y salidas

En la Tabla 4.5 se muestran los puertos disponibles, para el primer motor se usó el puerto PT2 y el segundo motor se eligió el puerto PT3, estos son los puertos que se usaron en la programación del lenguaje D, estos se usan para determinar la posición de los eslabones.

Tabla 4.5: Mapeo de puertos para las señales de los encoders

MAPA DE ENTRADA

Puerto de entrada	**Dispositivo**			
PT1	PT1[32] No usar	PT1[31..1] Posición Encoder	A	B
FPGA Pins			N16	M16
Header Pins			GPIO0_1	GPIO0_3
PT2	PT2[32] No usar	PT2[31..1] Posición Encoder	A	B
FPGA Pins			M20	M21
Header Pins			GPIO0_9	GPIO0_10
PT3	PT3[32] No usar	PT3[31..1] Posición Encoder	A	B
FPGA Pins			K21	K22
Header Pins			GPIO0_7	GPIO0_8

Tabla 4.6: Mapeo de puertos para las señales PWM

MAPA DE SALIDA							
PT1	PT1[32] No usar	1 (PWM1, PT1[31..26])	PT1[25] Dir Giro	PT1[24..11] TAT	PT1[10-1] AP	Derecha	Izquierda
FPGA Pins						F14	F13
FPGA Headers						GPIO1_31	GPIO1_33
PT1	PT1[32] No usar	2 (PWM2, PT1[31..26])	PT1[25] Dir Giro	PT1[24..11] DELTAT	PT1[10-1] AP	Derecha	Izquierda
FPGA Pins						F15	F12
FPGA Headers						GPIO1_32	GPIO1_34
PT1	PT1[32] No usar	3 (PWM3, PT1[31..26])	PT1[25] Dir Giro	PT1[24..11] DELTAT	PT1[10-1] AP	Derecha	Izquierda
FPGA Pins						M18	P16
FPGA Headers						GPIO0_24	GPIO0_23
PT1	PT1[32] No usar	4 (PWM4, PT1[31..26])	PT1[25] Dir Giro	PT1[24..11] DELTAT	PT1[10-1] AP	Derecha	Izquierda
FPGA Pins						L17	L18
FPGA Headers						GPIO0_26	GPIO0_25

En la Tabla 4.6 se mostraron los puertos disponibles para las salidas PWM, estas son generadas usando el puerto de salida PT1, en la misma tabla puede verse la configuración de los pines usados, en estos es donde se realizan las conexiones de salida de las señales, las cuales van conectadas a la tarjeta de protección mostrada anteriormente.

4.3 Encoder y decoder

Para controlar la posición del robot es necesario leer los datos proporcionados por un sensor, para obtener esta se usa un encoder acoplado al motor integrado a cada uno de los motores del robot, este nos permite conocer la posición del motor.

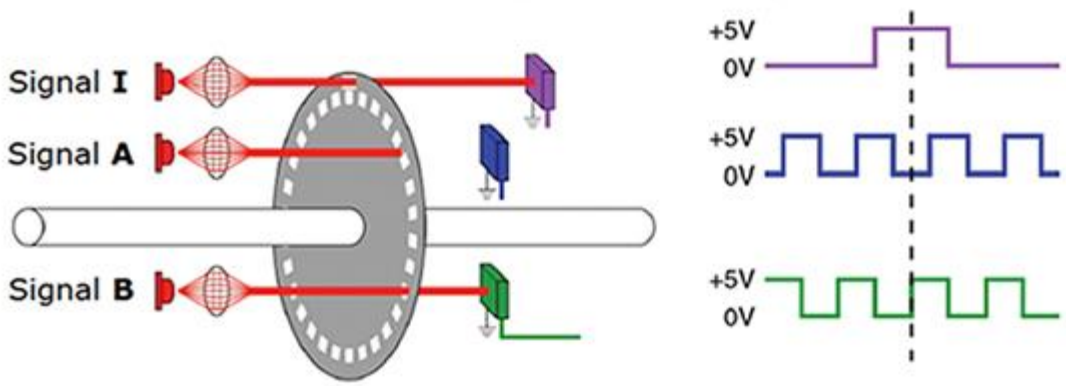


Figura 4.3: Señales de cuadratura estándar A y B más una señal de índice, para un codificador óptico [19].

El encoder funciona de manera que no solo nos permite saber cuánto ha avanzado, sino también en qué dirección, esto es posible gracias a que cuenta con 2 salidas en forma de un tren de pulsos, para conocer en qué dirección está girando el motor es necesario comparar ambas, algo importante a considerar es que no proporciona una posición inicial, sino que esta depende de la forma en que esta sea interpretada.

Las señales entregadas son A y B que se encuentran desfasadas por 90° esto nos permite realizar la siguiente tabla. 4.7.

Tabla 4.7: Casos del Decoder.

	Caso 1	Caso 2	Caso 3	Caso 4
B	0	0	1	1
A	0	1	1	0

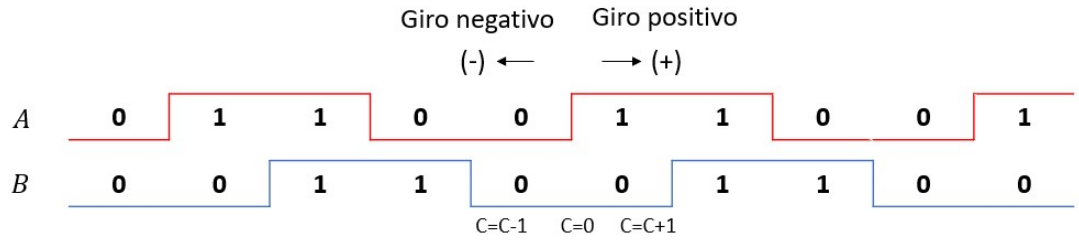


Figura 4.4: Señales A y B del Encoder.

La frecuencia de estas señales puede variar según la velocidad del motor, pero siempre conservan la misma relación de 90°, esto nos permite usar información, sin embargo, es necesario generar un bloque con la programación que decodifique la señales al desplazamiento recorrido.

La primera parte es conseguir un ciclo de muestreo que nos permita recuperar los datos en cualquier instante, esto dado que si se llega a perder información de cualquiera de los 2 canales la información recolectada esta errónea y dado que no se mide más información tendremos una implementación errónea.

Para generar el ciclo de muestreo de manera que nunca se pueda perder la información hay que cumplir 2 cosas, la primera es que sea lo suficientemente rápido, esto quiere decir que la frecuencia de muestreo sea más rápida que la frecuencia de la señal de los encoders, esto con el fin de que sea capaz de obtener la posición de efectiva sin perder información de la posición.

En la Figura 4.5 se pueden observar los pasos para obtener el reloj, primero se tomarán las señales de las entradas A y B, usando están señales como referencia se generará un retraso

haciendo uso de 3 flip-flop, las nuevas señales obtenidas son denominadas $A_{\Delta t}$ y $B_{\Delta t}$ ya que presentan un retraso respecto a las señales originales.

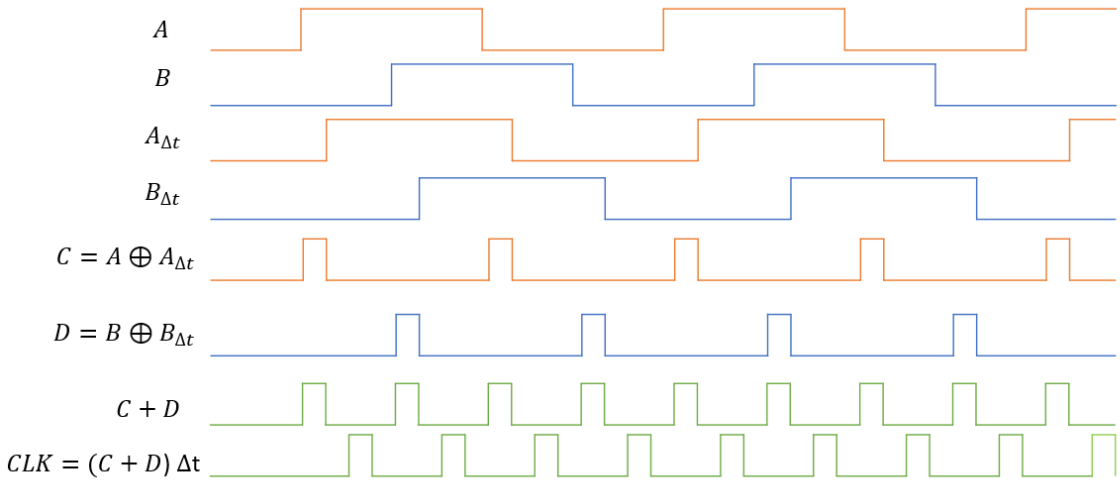


Figura 4.5: Diagrama de tiempos, donde se muestra la recuperación de la señal de reloj a partir de las señales A y B de un Encoder.

Posteriormente se realiza una operación XOR de las señales A con $A_{\Delta t}$, y B con $B_{\Delta t}$, cuyos resultados son denominados C y D, a lase señales C y D se les aplica una operación OR, y finalmente se realiza un último atraso Δt.

Para obtener el periodo usamos la ecuación, usando el inverso de la frecuencia máxima del motor.

$$\mathrm{T}_{min} = \frac{1}{\mathrm{f}_{\mathrm{maxAB}}} \tag{24}$$

Usando el tiempo mínimo se obtiene Δt a partir de

$$\Delta \mathrm{T} = \frac{1}{8}\mathrm{T}_{min} \tag{25}$$

En la siguiente Figura 4.6 se puede observar el circuito para obtener el reloj con el fin de poder recuperar la información.

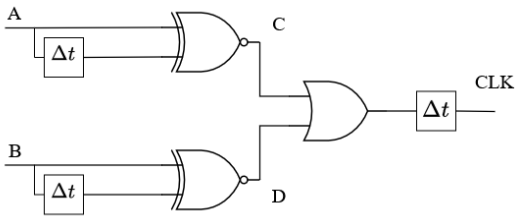


Figura 4.6: Circuito para recuperación del reloj.

Para poder determinar la posición del motor, se usan ambas señales A y B, las cuales correspondes a las salidas proporcionadas por el encoder de cada motor de corriente directa, aquí toma importancia ya que es necesario que los motores usados tengan un encoder incorporado para poder determinar la posición, usando el diagrama de la Figura 4.7 es posible determinar la posición del motor, tomando en cuenta ambas señales y los cambios que existen entre estas al variar la velocidad y el sentido de giro, podemos determinar la posición de cada grado de libertad en todo momento, así como llevar el conteo de los pulsos para llevar un control de la posición .

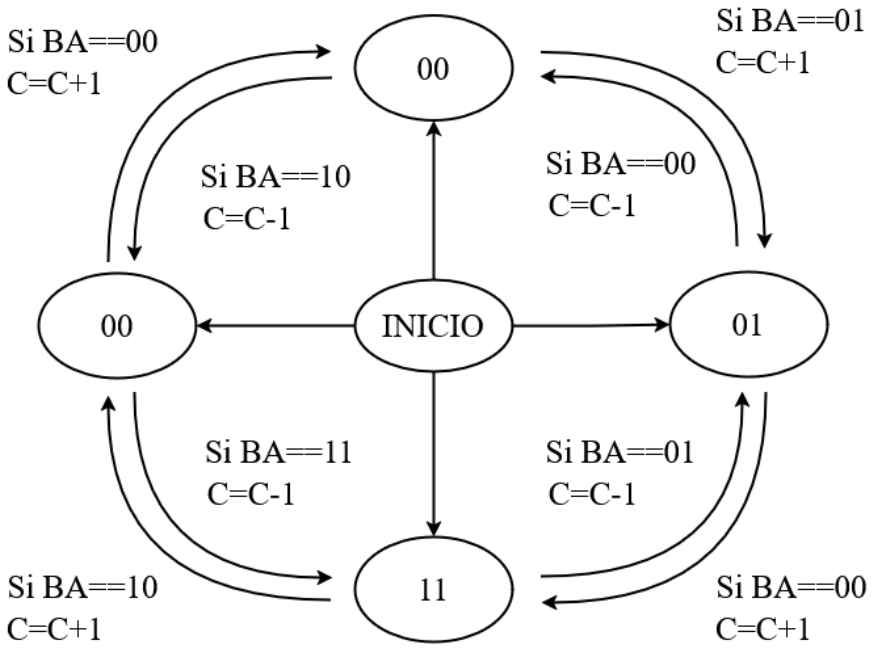


Figura 4.7: Diagrama de estados del Decoder.

4.4 Efecto del muestreo

Se trata del proceso en el cual somos capaces de convertir una señal analógica a digital, en este proceso de conversión se realiza la cuantificación de la señal analógica y el muestreo. Si la conversión se hace de forma correcta el proceso contrario será posible, de tal forma que en base a los datos digitales se podría reconstruir la señal analógica nuevamente [20].

El proceso de digitalización se puede representar matemáticamente, para esto es importante conocer la función delta de Dirac $[\delta(t)]$ en la Figura 4.8, la cual tiene varias propiedades, pues se trata de una función que solo existe un instante de tiempo, pero su amplitud tiende al infinito, por lo que su integral es 1, Ecuación 26.

$$\int_{-\infty}^{\infty} \delta(\mathrm{t})dt = 1 \tag{26}$$

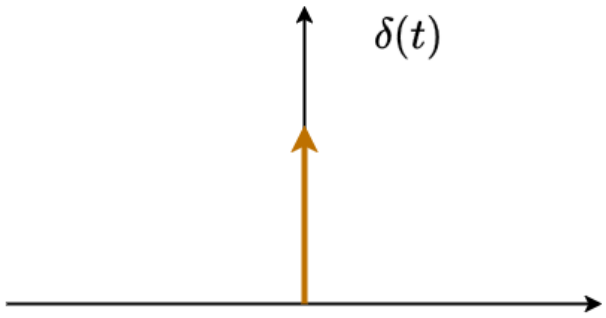


Figura 4.8: Delta de Dirac.

Para poder realizar un muestro también es importante el uso del tren o peine de Dirac $(Comb(t))$, ver Figura 4.9.

$$Comb(\mathrm{t}) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \tag{27}$$

donde: *T* es el periodo del tren de Dirac.

Comb(t)

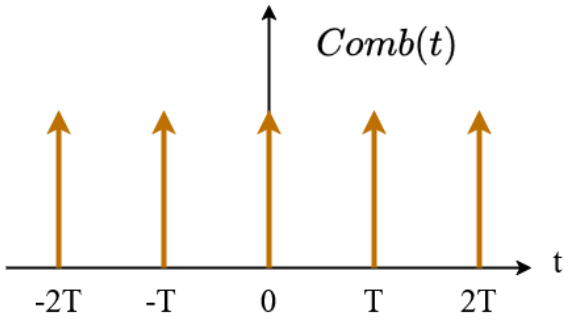


Figura 4.9: Tren de delta de Dirac.

El muestreo de la función *f(t)* matemáticamente se puede expresar con la siguiente ecuación:

$$f_m(t) = f(t)Comb(\mathrm{t}) = f(t)\sum_{n=-\infty}^{\infty}\delta(f - nT) \tag{28}$$

La ecuación 28 se puede representar mediante la Figura 4.10.

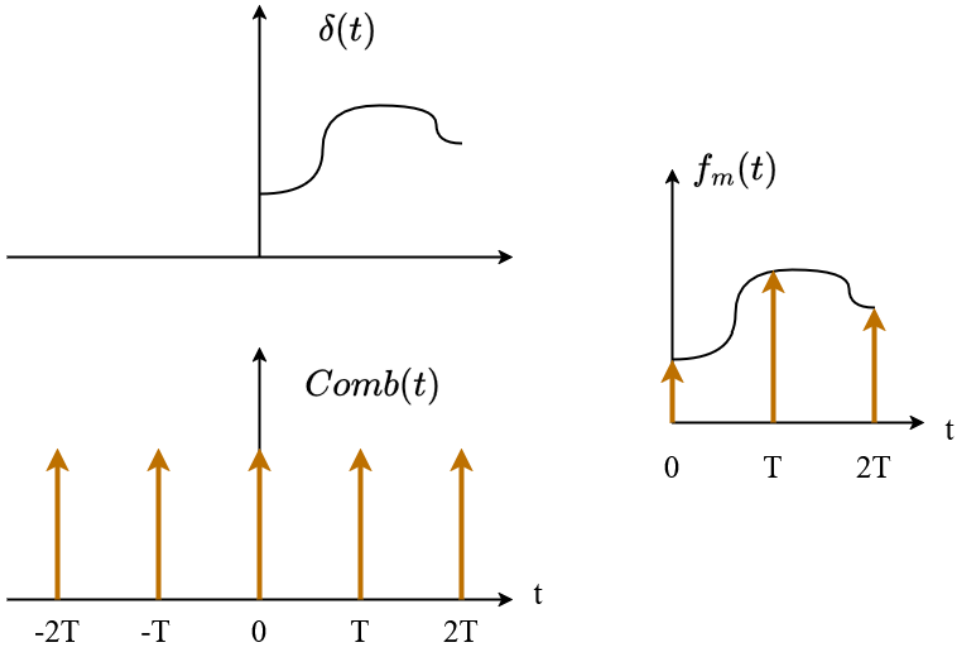


Figura 4.10: El muestreo de una señal *f(t)* es igual a multiplicar la función *f(t)* por el peine de Dirac. (a) Señal *f(t),* (b) Peine de Dirac y (c) Señal muestreada.

Usando las propiedades de la transformada de Fourier se sabe que la multiplicación de dos funciones en el tiempo es igual a la convolución de sus transformadas de Fourier en frecuencia, Ecuación 29.

$$F(w) * \omega_0 \sum_{n=-\infty}^{\infty} \delta(\omega - n\omega_0) = \frac{\omega_0}{2\pi} \int_{-\infty}^{\infty} F(\tau) Comb(\omega - \tau)\, d\tau \qquad (29)$$

Con: $$\mathcal{F}[Comb(t)] = \omega_0 \sum_{n=-\infty}^{\infty} \delta(\omega - n\omega_0) \qquad (30)$$

y $$F[f(t)] = F(w) \qquad (31)$$

La transformadas de Fourier para *f(t)* y para el peine de Dirac *comb(t),* se muestran en la siguiente Figura 4.11:

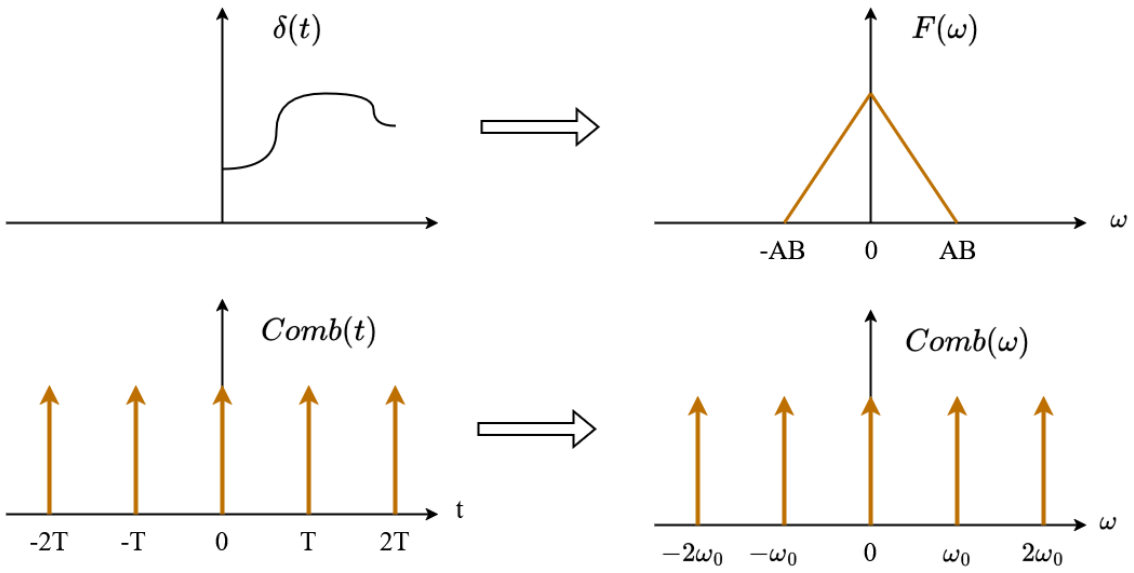


Figura 4.11: Representación gráfica de la función *f(t)* y *Comb(t)* con respectivas transformadas de Fourier. (a) *f(t)*, (b) *Comb(t)*, (c) Transformada de *f(t)* y (d) Transformada de *Comb(t)*.

La transformada de Fourier de un peine de Dirac es otro peine de Dirac pero en el dominio de la frecuencia, tal como se observa en la Figura 4.11.

La convolución puede presentar tres diferentes resultados según el valor del Ancho de Banda (AB) de la señal f(t) respecto de la frecuencia de muestreo (ω_0), los tres posibles resultados muestran en a la siguiente Figura 4.12.

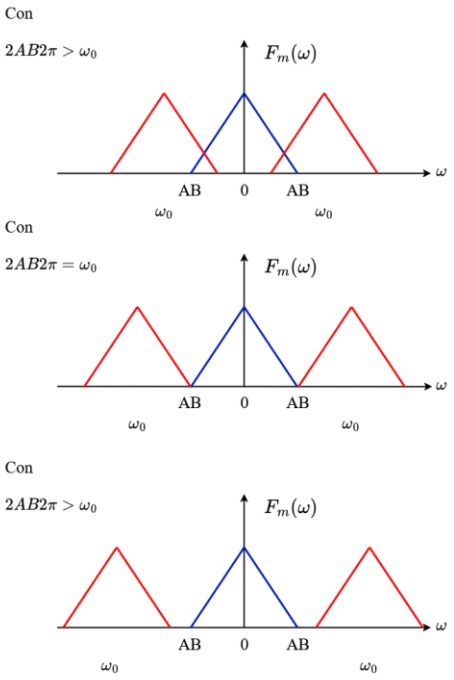


Figura 4.12: Resultados de la convolución entre $F(\omega)$ con el peine de Dirac $Comb(\omega)$. (a) Cuando la frecuencia angular de muestreo ω_0 es menor a 2 veces el ancho de banda (AB) multiplicado por 2π, (b) Cuando la frecuencia angular de muestreo ω_0 es igual a 2 veces el ancho de banda (AB) multiplicado por 2π y (c). Cuando la frecuencia angular de muestreo ω_0 es mayor a 2 veces el ancho de banda (AB) multiplicado por 2π.

Analizando la Figura 4.12, en el inciso (a), la información se pierde al encontrarse traslapados los espectros $F(\omega)$ y no hay forma de recuperar la señal $f(t)$.

En el inciso (b) de la Figura 4.12, la información se puede recuperar únicamente empleando un filtro pasa bajas, esto es seleccionando 2 veces el ancho de banda requerido para poder reconstruir la señal original.

En el caso (c) de la Figura 4.12, es posible recuperar la información muestreada seleccionando un ancho de banda mayor a 2 veces la frecuencia a muestrear.

Usando lo explicado anteriormente se puede concluir que es necesario que se cumpla el teorema de Nyquist donde la frecuencia de muestreo debe ser al menos dos veces la frecuencia máxima de nuestra señal, esto para poder reconstruir la señal original.

El sistema de control usado en los robots desarrollados en el laboratorio de automatización de la MCEA es totalmente digital, por lo que este filtro es implementado dentro del firmware del FPGA [20.]

4.5 Filtro pasabajas

Los filtros son elementos diseñados para dejar pasar una señal a una cierta frecuencia deseada y atenuar las que no se encuentren en ese rango de frecuencia, estos dispositivos pueden modificar la amplitud de la señal de entrada como su fase.

Existen distintos tipos de filtros, en este caso particular se necesita emplear un filtro de tipo pasa-bajas, el cual permite el paso de frecuencias muy bajas hasta un valor determinado, como se muestra en la Figura 4.13, donde f_c es la frecuencia de corte del filtro a usar.

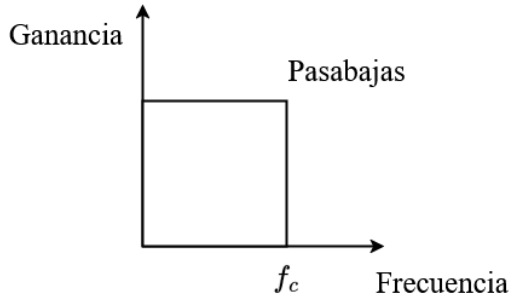


Figura 4.13: Grafica de Ganancia del Filtro Pasa bajas.

Como observamos en la Figura 4.14, el espectro de nuestra señal muestreada se va repitiendo cada cierto tiempo, por lo que para recuperar la señal original sin que esas repeticiones tengan efecto, se utiliza un filtro pasa bajas que se diseña de tal forma que solo deje pasar el espectro de la señal original y no las repetidas, como se muestra en la Figura 4.14 [20].

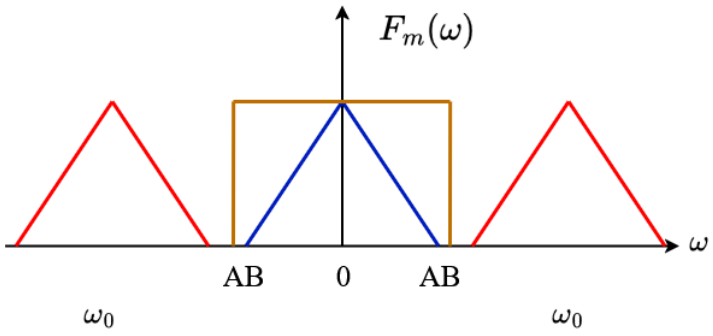


Figura 4.14: Filtrado de la señal recuperada.

El filtro usado en este trabajo ya se encuentra implementado en lenguaje D, y se toma en considera que la frecuencia de muestreo es mayor a dos veces la frecuencia máxima que está dada en términos de la frecuencia máxima del motor.

4.6 Cuantificación de la velocidad angular en función de la posición angular.

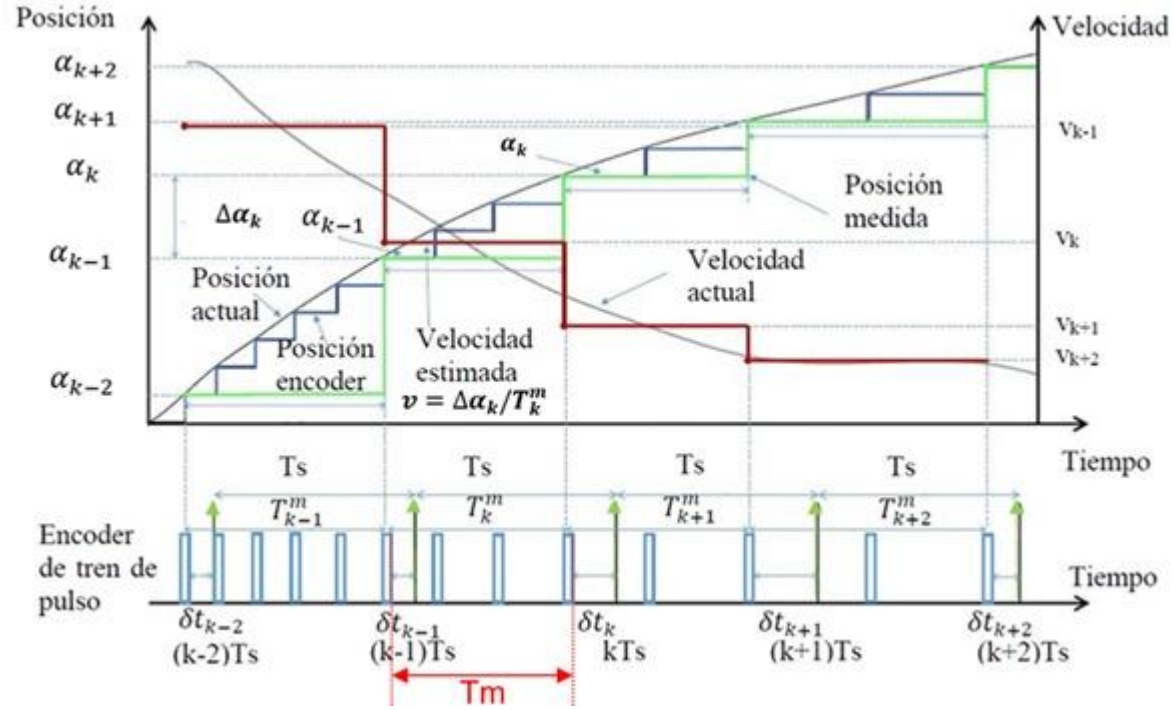


Figura 4.15: Cuantificación de la velocidad

La siguiente ecuación es aplicada dentro del firmware en el muestreo del encoder, para realizar la cuantificación, tomando en cuenta el tren de pulsos del FPGA.

$$v_m^{MT} = \frac{\alpha_k - \alpha_{k-1}}{T_m} = \frac{\alpha_k - \alpha_{k-1}}{T_s + \delta t_k - \delta t_{k-1}} \tag{32}$$

4.7 Generador PWM

Los tiempos empleados en el generador de PWM se definen mediante contadores que registran la duración en ciclos de reloj del dispositivo utilizado. En este esté trabajo se hizo uso de un FPGA que opera a una frecuencia de 100 MHz. Para poder diseñar un bloque que nos genere una señal de PWM, el cual puede variarse de acuerdo con la frecuencia de cada motor a controlar, este periodo es de 10 ns, dado que la frecuencia del FPGA es de 100 MHz (Ecuación 33).

Además, el ancho del pulso será un conjunto de bits de entrada definido por N_{AP}, un bit de entrada correspondiente al sentido de giro, que indica la dirección a la cual está girando el motor del motor. En cuanto a las salidas, tendremos PWM+ y PWM-, cuyo estado dependerá de la dirección de giro que se busque. A partir de ello, en la siguiente Figura 4.16 se encuentra el diagrama del bloque de PWM implementado en firmware dentro del FPGA

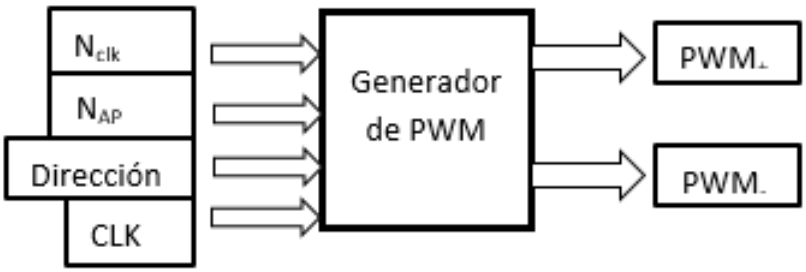


Figura 4.16: Diagrama del PWM.

En el cual tenemos las siguientes variables.

$$N_{clk} = \frac{T_{pwm}}{T_{clk}} \quad , \tag{33}$$

donde: N_{clk} es el número de ciclos que corresponde a un periodo del PWM, T_{pwm} representa el periodo del PWM, T_{clk} es el periodo del reloj del FPGA.

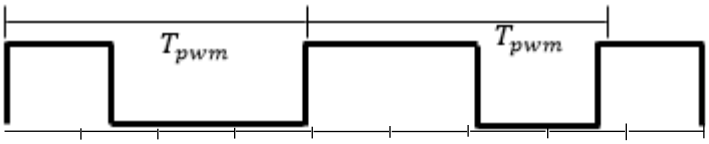


Figura 4.17: Diagrama de tiempos del PWM.

$$\Delta AP = \frac{T_{pwm}}{N_{APmax}} \tag{34}$$

En este caso, ΔAP representa el incremento mínimo utilizado para definir el ancho de pulso. Se calcula dividiendo el periodo del PWM(T_{pwm}) entre la cantidad máxima de subdivisiones permitidas ($N_{APm\,ax}$), que en este caso es 1000.

El tiempo del ancho de pulso está definido por la ecuación:

$$N_{AP} = \frac{AP}{\Delta AP} \tag{35}$$

Donde N_{AP} es el número que representa el ancho de pulso solicitado, AP es el tiempo que el pulso estará en alto y es el número que corresponde a la partición mínima del periodo del PWM.

En este caso, N_{AP} representa el valor asignado al ancho de pulso solicitado, mientras que AP indica la duración durante la cual el pulso permanecerá en estado alto. Además, AP corresponde a la unidad mínima de división dentro del periodo del PWM.

La dirección es determinada por el controlador.

4.8 Módulo Wi-Fly

El microprocesador emulado dentro de la FPGA tiene la capacidad de comunicarse con diversos dispositivos a través del protocolo WIFI, dicho módulo IP es capaz de administrar los comandos necesarios para comunicar el microprocesador con el módulo de conexión inalámbrica llamado "Wifly", mismo que incorpora el circuito integrado RN-XV, capaz de lograr toda la comunicación por este protocolo (Figura 4.18) [20],[21].

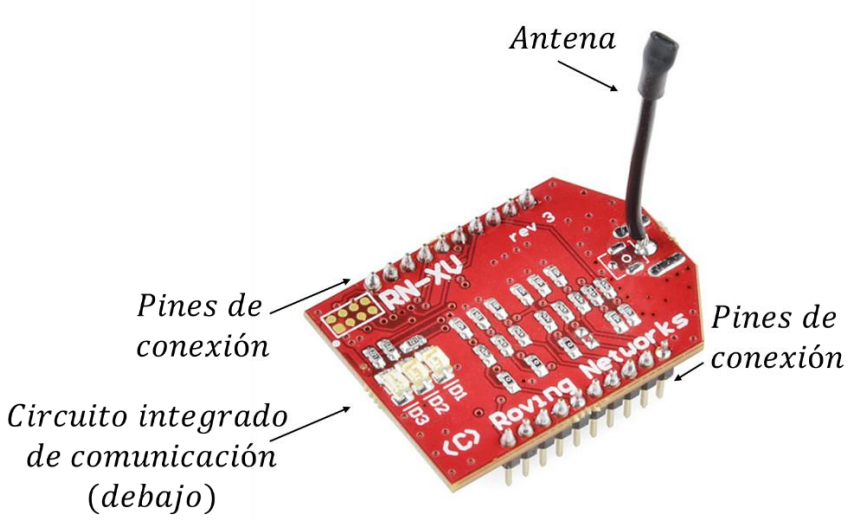


Figura 4.18: Módulo de comunicación inalámbrica Wifly [25].

Como se ha mencionado antes, es posible accesar al módulo WIFI a través del puerto cero (Tabla 4.8) utilizando la instrucción "EPOR Rx PT0" para escribir algún dato o "LPOR PT0 Rx" para leer los datos provenientes del módulo WIFI.

Tabla 4.8: Uso del puerto cero para acceso a datos WIFI.

N° PTO	Disposición de bits de puerto	
PT0	No usar PT0[32]	Reservado para la lectura/escritura datos WIFI PT0[31-1]

El puerto de entrada y salida PT0 está intercomunicado con un módulo interno de comunicación serial UART, el protocolo de comunicación serial tiene como objetivo entablar comunicación entre dos dispositivos a través de dos señales, Rx y Tx, donde la señal Rx de un dispositivo puede leer los datos recibidos a través de la señal Tx de otro dispositivo y viceversa (Figura 4.19) [20].

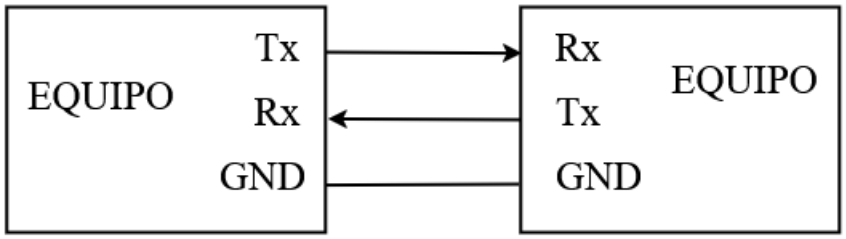


Figura 4.19: Comunicación serial entre dispositivos a través de señal Rx y Tx.

Para utilizar el módulo Wifi es necesario configurarlo a los parámetros de operación del módulo ip serial del procesador, estos consisten en:

- Velocidad de transferencia (baudios): 460800
- Tamaño de palabra: 5

Los demás parámetros (canal de operación, frecuencia, etc) pueden mantenerse con los valores de fábrica.

Para lograr la configuración, usamos el programa “Teraterm”, y una vez abierto, es necesario indicar la dirección IP del módulo (192.168.1.1 para este dispositivo), seleccionar servicio Telnet e indicar el puerto TCP 2000 (Figura 4.20).

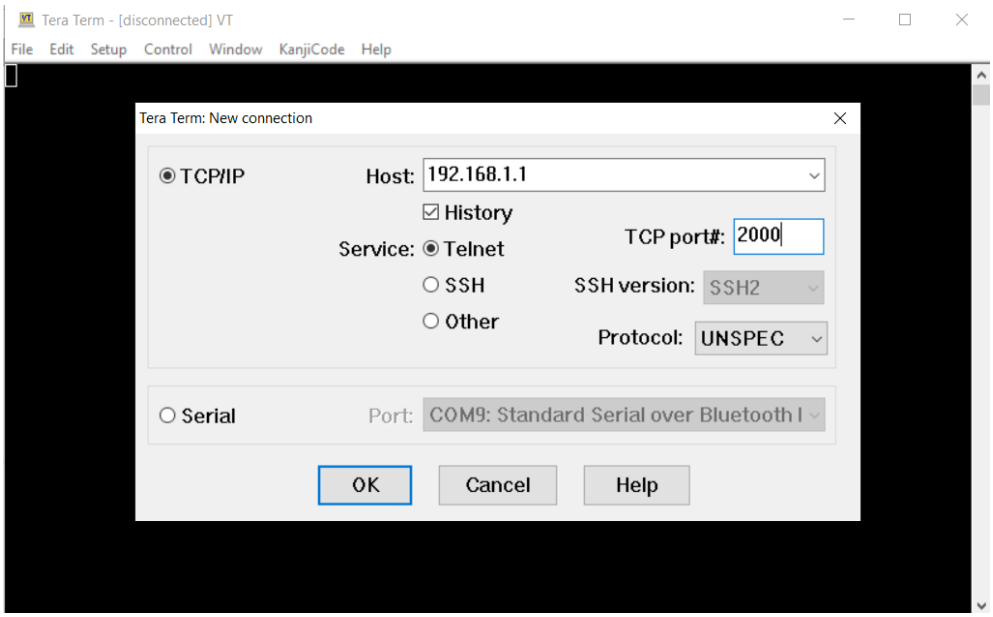


Figura 4.20: Ventana de configuración de módulo Wifi con Teraterm.

Una vez abierta, se habilitará la ventana de comandos del terminal y eso significa que el dispositivo está entablando comunicación con la computadora, posteriormente es necesario añadir los comandos correspondientes, estos se enlistan a continuación:

- **$$$**: este comando nos permite entrar a modo configuración, al dar click "enter", recibimos el mensaje "CMD"
- **set uart baudrate 460800**: Con este comando cambiaremos la velocidad de transferencia de datos serial a 460800 baudios por segundo, el haber ejecutado este comando adecuadamente devuelte "AOK" como respuesta.
- **set comm size 5:** De esta forma se cambia el tamaño de la palabra a 5 palabras de 8 bits a transferir por paquete, de igual forma se recibe "AOK" como respuesta si se configura adecuadamente.
- **save:** Este comando guarda los cambios hechos en la configuración del módulo.
- **Reboot:** Finalmente, se reinicia el dispositivo con este comando y así salir de modo programación, con ello será suficiente la configuración del dispositivo para operar con el microprocesador.

Conclusiones

En este apartado se mostraron los elementos que se encuentran en el firmware, estos son importantes ya que son los que permiten que pueda existir comunicación entre el microprocesador dentro del FPGA, así como la comunicación entre el FPGA y la PC, esta integración facilita el desarrollo de esta tesis ya que permite la implementación de todas las partes que conforman el sistema.

Es importante notar que estos elementos fueron diseñados previamente en la Maestría de Ciencias de la electrónica, por lo que el entendimiento de estos bloques permite entender la forma en la que funciona el generador de PWM y la frecuencia de muestreo, ya que en este caso la frecuencia de las salidas de PWM de ambos motores está dada por la frecuencias encontradas en la caracterización de los motores, donde estas frecuencias son las que proporcionan un torque lineal, para lograr que los motores de corriente directa se comporten como motores de quasi transmisión directa.

Finalmente se concluye que la comunicación entre la PC y el Módulo Wi-fly permite realizar él envió de los datos capturados por el FPGA a la PC, facilitando la transmisión de los datos de forma inalámbrica, por lo que no es necesario realizar conexiones físicas entre el FPGA y la computadora, gracias a esta comunicación se pudieron obtener los datos de posición de ambos motores, además de las señales de los torques aplicados a los motores en las diferentes pruebas realizadas.

Capítulo 5. Software

5.1 Software de adquisición

En la siguiente Figura 5.1 se observa la interfaz principal desarrollada en la Maestría de Ciencias de la Electrónica, como se puede observar esta fue modificada para usarla en el brazo robot, por lo que se modificó para que se puede modificar la posición deseada θ_{c1}, θ_{c2} esto con el fin de poder realizar los cambios en la interfaz sin tener que modificar el código, ya que la posición deseada se vuelve una variable del código, por lo que esta se puede cambiar a lo largo de la ejecución, en esta pantalla, una vez que se descargó el programa es posible ir cambiando los parámetros del ángulo deseado y las constantes del control.

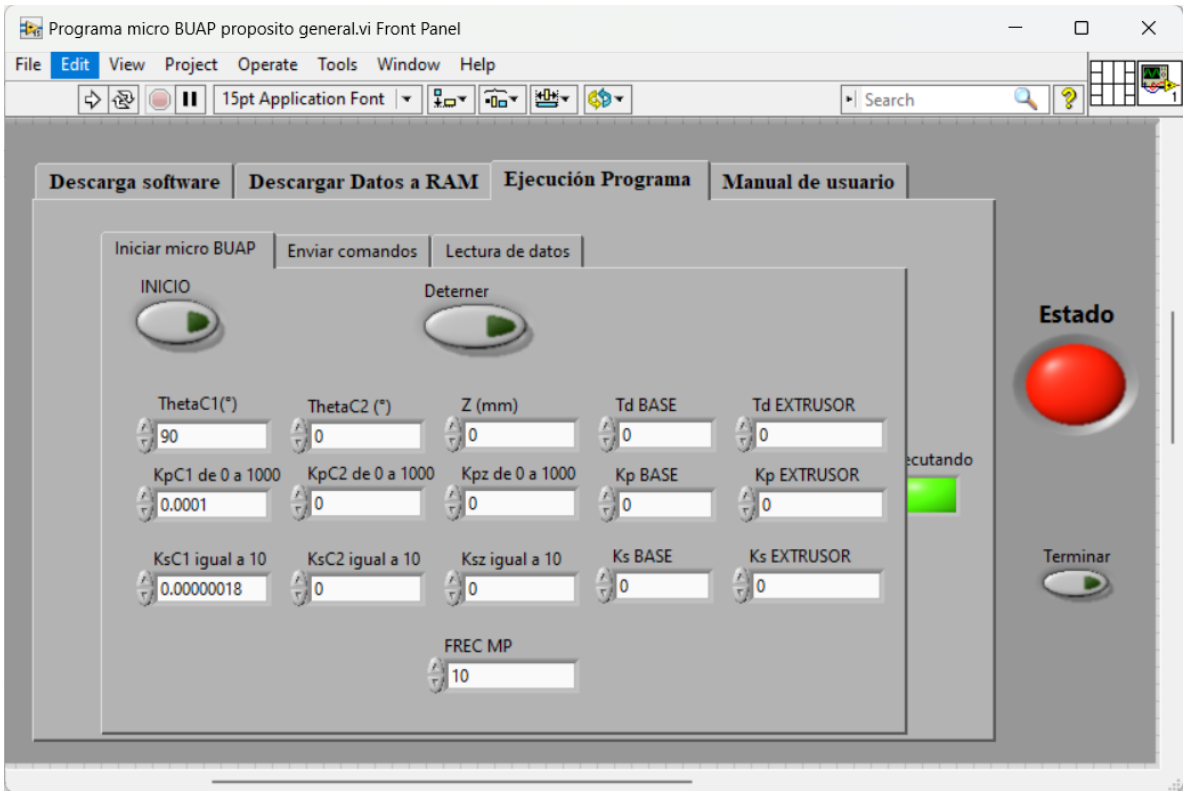


Figura 5.1: Interfaz principal de la ejecución del programa.

En la Figura 5.2 se observa la subpestaña “Lectura de datos”, en esta pestaña se puede observar la variable que está siendo capturada en la computadora, también en esta pestaña se encuentra un botón donde se pueden grabar los datos, primero es necesario escribir el nombre del archivo, para esto también es necesario modificar la ruta de la carpeta donde se guardarán los archivos creados, en este se grabarán los datos que se encuentran en el buffer del sistema, los cuales pueden verse en “Datos”

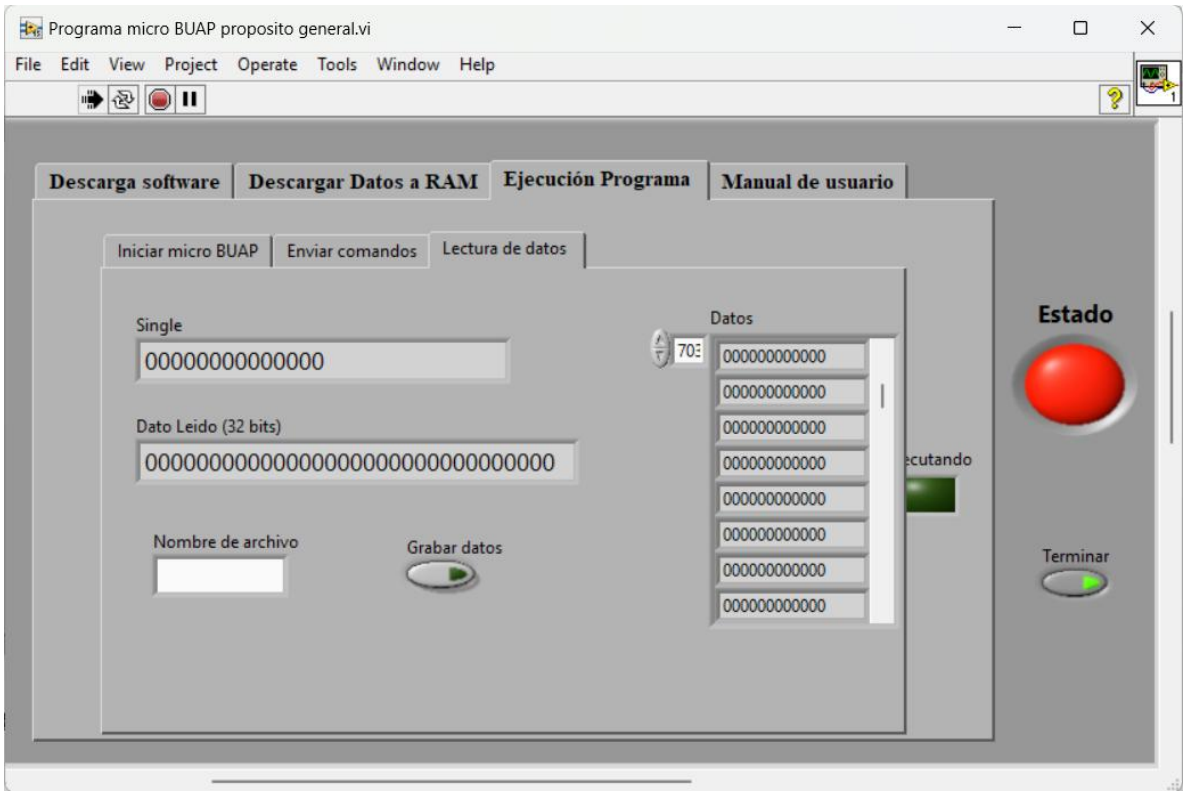


Figura 5.2: Interfaz de usuario, pestaña “Lectura de datos”.

5.2 Software de control

Se hizo uso del compilador de lenguaje D, el cual fue diseñado en la Maestría de Ciencias de la electrónica, en la Tabla 5.1 se muestran las instrucciones de este lenguaje.

El microprocesador emulado dentro de la tarjeta de desarrollo FPGA contiene una unidad aritmética lógica (ALU, por sus siglas en inglés) capaz de ejecutar, entre otros, funciones matemáticas básicas y trigonométricas, operadores de salto de línea de instrucción, comparadores (IF, WHILE, FOR), operación bit a bit (instrucción AND, OR), entre otros.

Es posible ejecutar las diferentes funciones del microprocesador acorde a las necesidades de nuestro algoritmo a implementar, para lograrlo, se ha desarrollado un compilador que “traduce” el lenguaje de programación de alto nivel a lenguaje máquina, las instrucciones con las que se cuentan se describen en la Tabla 5.1. El lenguaje de programación posee instrucciones similares al lenguaje “C”, las diferencias radican principalmente en que este lenguaje de programación está orientado a registros, es decir, está limitado a apuntadores dentro de la memoria RAM del microprocesador, (en el lenguaje de programación, los registros son invocados con la letra R_i). Las instrucciones principales de este lenguaje de programación son en el poder hacer sumas, productos, funciones

trigonométricas, el resultado es almacenado en un registro cualesquiera y el argumento de los operadores pueden ser números naturales o registros, según lo necesario.

Para acceder a la memoria RAM se han implementado las instrucciones LRAM y ERAM, la primera permite leer la memoria RAM la última la escritura en memoria RAM. La sintaxis de lectura de memoria RAM es "$LRAM \quad Z \quad R_i$" donde Z es la dirección de la memoria RAM a la cual se desea acceder y R_i representa el registro en el cual se desea almacenar el dato leído en la memoria RAM de la dirección Z; para la escritura en la memoria RAM se implementa la instrucción "$ERAM \quad R_i \quad Z$" donde R_i es el registro que posee el dato que deseamos almacenar en memoria RAM y Z es la dirección en la cual se almacena el dato.

El procesador trabaja de forma nativa en punto flotante, lo que significa que para hacer operaciones matemáticas y comparaciones, estas se ejecutarán en punto flotante, no obstante, existe la posibilidad de convertir un dato de punto flotante a entero y viceversa, esta posibilidad se da debido a que los apuntadores de memoria RAM, por ejemplo, deben hacerse en formato entero, además, para transferir resultados de diferentes procesos dentro del microcontrolador a la computadora es recomendable hacerlo en dato tipo entero por la estabilidad que implica esta instrucción. Para convertir un dato tipo entero a punto flotante se utiliza la instrucción "$R_i = INT2FP\ R_j$" donde R_i almacena el valor R_j en tipo flotante y éste último debería tener un formato tipo entero, para obtener un dato tipo entero a partir de un dato tipo punto flotante recurrimos a "$R_j = FP2INT\ R_i$", donde R_j almacena el dato R_i en formato tipo entero, cuando R_i tiene un formato tipo punto flotante [20],[21],[22].

Tabla 5.1: Instrucciones disponibles para programación de microprocesador.

INSTRUCCIÓN Y SINTAXIS Lenguaje D				
IF	R	>,<,>=,<=	R o N	
CODIGO A REALIZAR SI LA CONDICION ES CIERTA				
ENDIF				
WHILE	R	>,<,>=,<=	R o N	
CODIGO A REALIZAR REPETIDAMENTE SI LA CONDICION ES CIERTA				
ENDWHILE				
FOR	N (VALOR DE INICIO)	N (VALOR DE AUMENTO	N (VALOR FINAL)	
CODIGO A REALIZAR REPETIDAMENTE				
ENDFOR				
DELAY				
R	=	N		
R	=	R	+	R o N
R	=	R	-	R o N

R	=	R	*	R o N
R	=	R	/	R o N
R	=	R o N	^	R o N
R	=	LN	R o N	
R	=	EXP	R o N	
R	=	SQRT	R o N	
R	=	SIN	R o N	
R	=	COS	R o N	
R	=	TAN	R o N	
R	=	CSC	R o N	
R	=	SEC	R o N	
R	=	COT	R o N	
R	=	SINH	R o N	
R	=	COSH	R o N	
R	=	TANH	R o N	
R	=	CSCH	R o N	
R	=	SECH	R o N	
R	=	COTH	R o N	
R	=	ARCSIN	R o N	
R	=	ARCCOS	R o N	
R	=	ARCTAN	R o N	
LRAM	DIRECCION	R		
ERAM	R	DIRECCION		
LPOR	P	R		
EPOR	R	P		

5.3 Compilador Lenguaje D

En la Figura 5.3 se observa la pantalla de la interfaz gráfica de usuario del compilador, , una vez que se realizó el código en lenguaje D usando un editor de texto plano se procede a realizar una nueva compilación, donde se genera un archivo ensamblador que posteriormente se convierte a un archivo en hexadecimal, el cual ya puede ser interpretado por el microprocesador implementado en el FPGA.

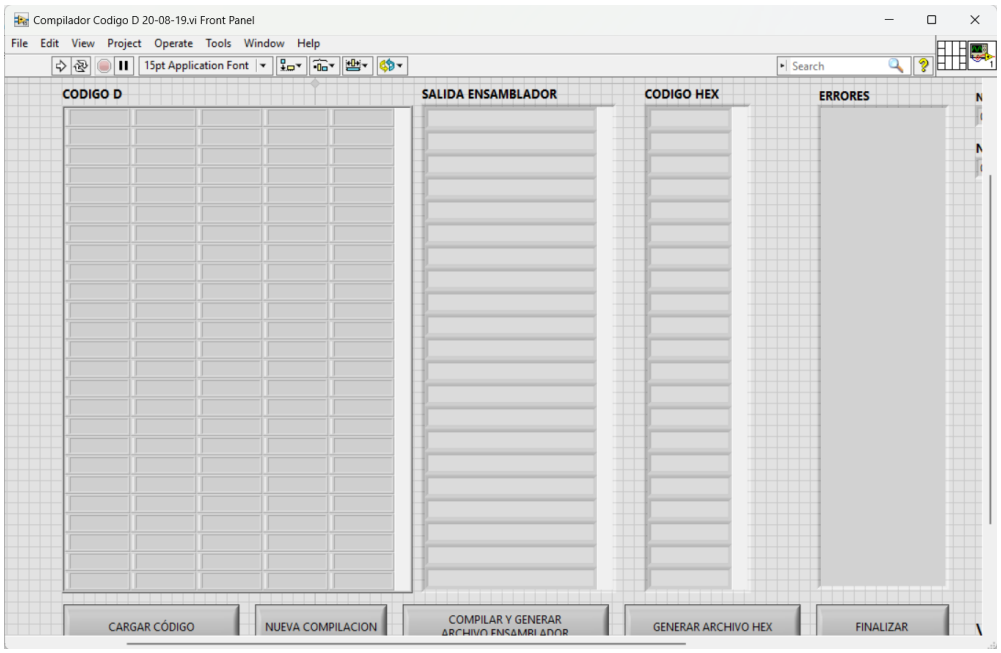


Figura 5.3: Interfaz de usuario para el compilador en Lenguaje D.

Una vez que se cuenta con diagrama general es posible asignar los puertos necesarios de entrada y salida como se muestra a continuación en la Figura 5.4.

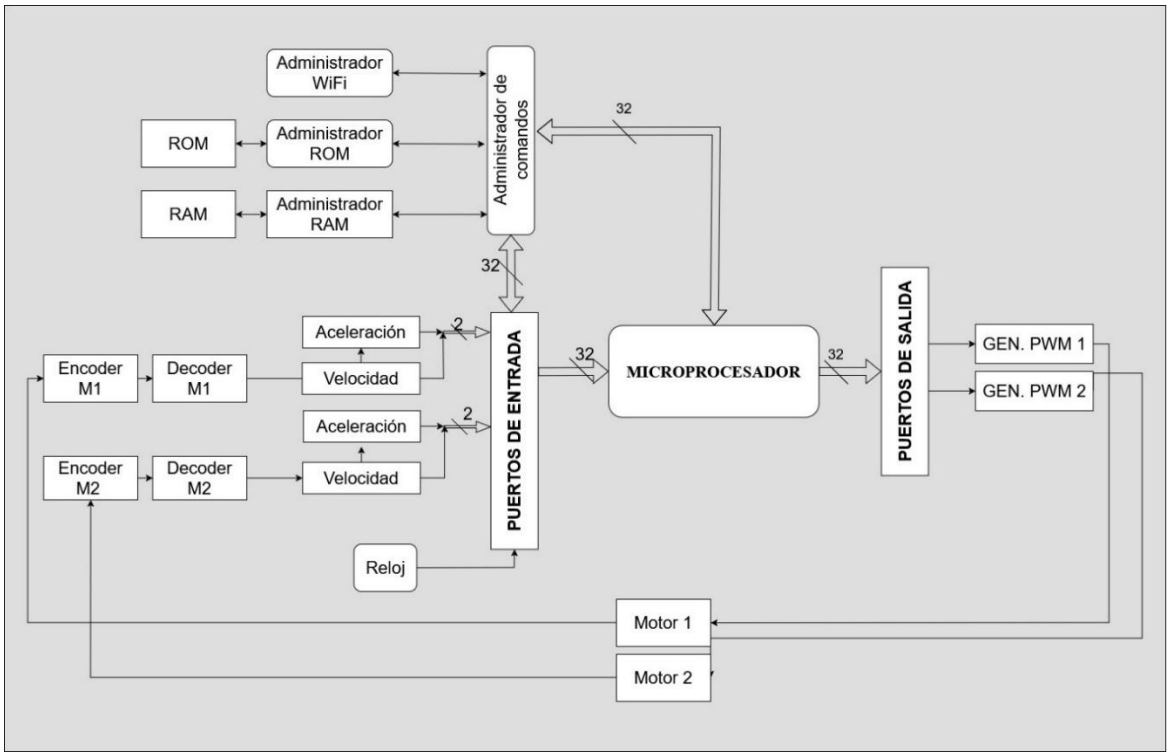


Figura 5.4: Diagrama a bloques del sistema del brazo robot.

Usando el diagrama anterior como referencia se eligieron los puertos correspondientes a usar dentro del robot, estos se eligieron respecto a las características necesarias dadas por el sistema, en la tabla 5.2 se eligieron los puertos PT2 y PT3, esto es importante ya que en la programación estos son los puertos donde se lee la información de cada uno de los encoders, posteriormente haciendo uso del lenguaje D, se realiza la conversión de pulsos a su posición angular.

Tabla 5.2: Mapa de entrada

MAPA DE ENTRADA

Puerto de entrada	**Dispositivo**			
PT1	PT1[32] No usar	PT1[31..1] Posición Encoder	A	B
FPGA Pins			N16	M16
Header Pins			GPIO0_1	GPIO0_3
PT2	PT2[32] No usar	PT2[31..1] Posición Encoder	A	B
FPGA Pins			M20	M21
Header Pins	**Motor 1**		GPIO0_9	GPIO0_10
PT3	PT3[32] No usar	PT3[31..1] Posición Encoder	A	B
FPGA Pins			K21	K22
Header Pins	**Motor 2**		GPIO0_7	GPIO0_8
PT4	PT4[32] No usar	PT4[31..1] Posición Encoder	A	B
FPGA Pins			D17	K20
Header Pins			GPIO0_5	GPIO0_6

Para el mapeo de las señales de los encoders se usaron los puertos PT3, y PT4 tal como se muestran en la siguiente Tabla 5.2, estos de igual forma son usados para realizar la programación usando lenguaje D.

Tabla 5.3: Mapeo de puertos para las señales PWM

MAPA DE SALIDA							
PT1	PT1[32] No usar	1 (PWM1, PT1[31..26])	PT1[25] Dir Giro	PT1[24..11] TAT	PT1[10-1] AP	Derecha	Izquierda
FPGA Pins						F14	F13
FPGA Headers						GPIO1_31	GPIO1_33
PT1	PT1[32] No usar	2 (PWM2, PT1[31..26])	PT1[25] Dir Giro	PT1[24..11] DELTAT	PT1[10-1] AP	Derecha	Izquierda
FPGA Pins						F15	F12
FPGA Headers	**PWM Motor 1 (Brazo)**					GPIO1_32	GPIO1_34
PT1	PT1[32] No usar	3 (PWM3, PT1[31..26])	PT1[25] Dir Giro	PT1[24..11] DELTAT	PT1[10-1] AP	Derecha	Izquierda
FPGA Pins						M18	P16
FPGA Headers	**PWM Motor 2(Codo)**					GPIO0_24	GPIO0_23
PT1	PT1[32] No usar	4 (PWM4, PT1[31..26])	PT1[25] Dir Giro	PT1[24..11] DELTAT	PT1[10-1] AP	Derecha	Izquierda
FPGA Pins						L17	L18
FPGA Headers						GPIO0_26	GPIO0_25
PT1	PT1[32] No usar	5 (PWM5, PT1[31..26])	PT1[25] Dir Giro	PT1[24..11] DELTAT	PT1[10-1] AP	Derecha	Izquierda
FPGA Pins						K17	L19
FPGA Headers						GPIO0_28	GPIO0_27

Finalmente, una vez que se ha escrito el programa, y se ha compilado, se tiene la ventana de “Descarga de Software” tal como se muestra en la Figura 5.5, esta es la ventana donde se descarga el programa en ensamblador usando conexión Wifi, de esta manera es posible mandar el programa al PFGA sin realizar conexiones físicas con la computadora, una vez que se realizó la descarga del programa ya es posible seguir con el siguiente paso que es la ejecución y captura de los datos necesarios.

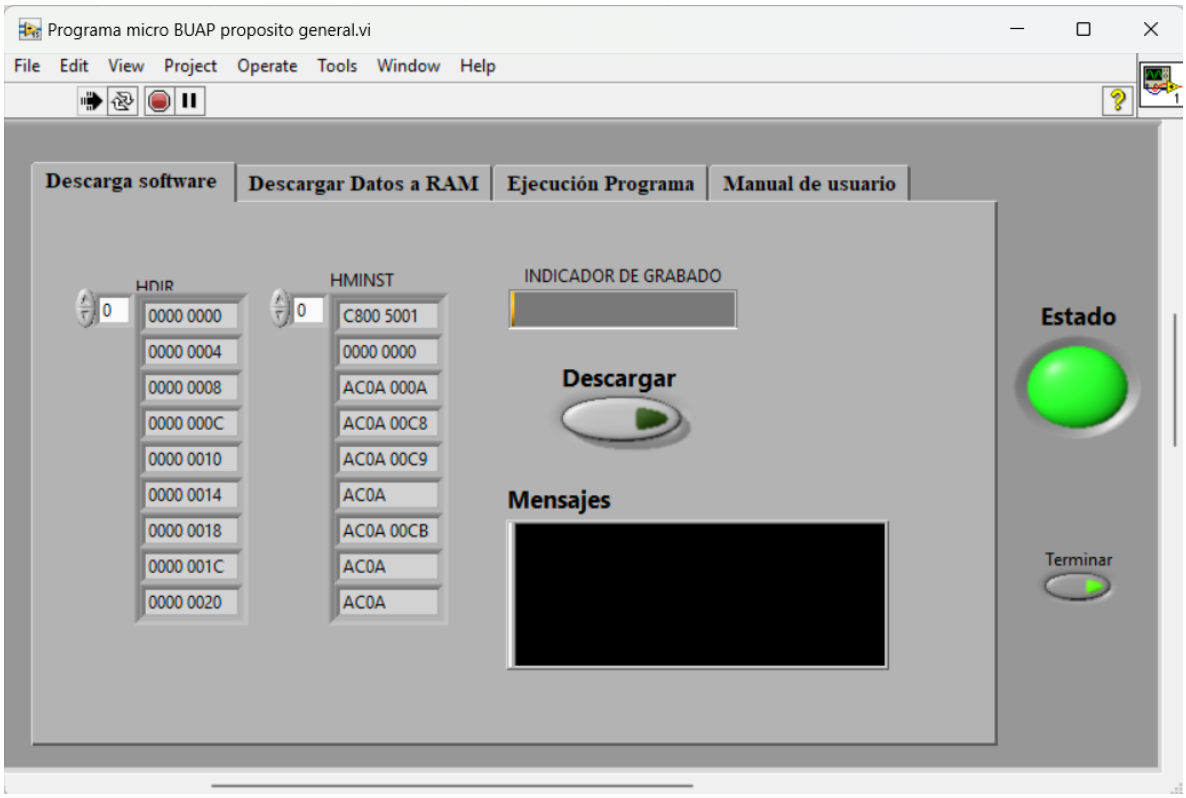


Figura 5.5: Interfaz de usuario pestaña “Descarga de software”.

Conclusiones

Se aprendió a usar la interfaz desarrollada en la maestría y del conjunto de instrucciones del Lenguaje D, por lo que ya fue posible escribir los programas necesarios para realizar pruebas experimentales usando el brazo robot, además se usaron las tablas para conocer los puertos usados por lo que ya es posible configurar las entradas y salidas del robot de acuerdo con el mapeado realizado, con el uso del Lenguaje D ya es posible realizar el control del brazo robot.

Capítulo 6. Redes neuronales convolucionales (CNN)

El origen de las redes neuronales convolucionales se remonta a los primeros modelos de redes neuronales artificiales propuestas en los años 70, tomando en base los estudios sobre la David Hubel y Torsten Wiesel sobre el córtex visual. Hubel y Wiesel estudiaron el córtex estriado, la primera parte del córtex visual que interviene en el procesamiento de la información visual, la que recibe información directamente del tálamo. En la misma década Kuhiniko Fukushima, propone un modelo de red neuronal artificial inspirado biológicamente: conocido como cognitrón, este tipo de arquitectura de las redes neuronales de Fukushima es muy similar a las redes convolucionales actuales, años después publicaría una versión mejorada de su modelo: el neocognitrón. La desventaja del neocognitrón yace en que no dispone de un algoritmo genérico para el entrenamiento, contrario a las redes convolucionales que se pueden entrenar utilizando el gradiente descendente y backpropagation.

Otro pionero fue el psicólogo Frank Rosenblatt de la Universidad de Corell. En 1959, Rosenblatt construyó una máquina neural simple que llamó perceptrón. Ésta tenía una matriz con 400 fotoceldas que se conectaban aleatoriamente a 512 unidades tipo neurona. Cuando se representaba un patrón a las unidades sensoras, éstas enviaban una señal a un banco de neuronas que indicaba la categoría del patrón. El perceptrón de Rosenblatt reconoció todas las letras del alfabeto. Al final de los años setenta Minsky y Papert demostraron que los perceptrones eran incapaces de hacer tareas simples tales como sintetizar la función lógica XOR.

En 1986 McClelland y Rumelhart publicaron un libro en dos volúmenes titulado: Parallel Distributed Processing: Explorations in the Microstruture of Cognition. Este libro se considera un clásico en el área de redes neurales y se puede decir que su aparición significó un nuevo impulso a la investigación en sistemas neurales al mostrar las ventajas y desventajas de las redes neurales artificiales (RNA) [1].

A continuación, se mostrará un breve resumen de los elementos de las redes neuronales convolucionales, las partes que la conforman, así como el entrenamiento que se usó en este trabajo.

Elementos de una red neuronal convolucional

Una red neuronal se caracteriza por los siguientes elementos:
1. Un conjunto de unidades de procesamiento o neuronas.
2. Un estado de activación para cada unidad, equivalente a la salida de la unidad.
3. Conexiones entre las unidades, generalmente definidas por un peso que determina el efecto de una señal de entrada en la unidad.

4. Una regla de propagación, que determina la entrada efectiva de una unidad a partir de las entradas externas.
5. Una función de activación que actualiza el nuevo nivel de activación basándose en la entrada efectiva y la activación anterior.
6. Una entrada externa que corresponde a un término determinado como bias para cada unidad.
7. Un método para reunir la información, correspondiente a la regla del aprendizaje
8. Un ambiente en el que el sistema va a operar, con señales de entrada e incluso señales de error.

La ecuación que define a una neurona como el elemento básico es la siguiente

$$y = f\left(\sum_i w_i x_i\right) \tag{36}$$

Donde
x_i: señales de salida de otros nodos o entradas externas.
w_i: pesos de las ligas de conexión.
$f(\cdot)$: función no lineal.

La función *f* puede ser sigmoidal, tangente hiperbólica, escalón, entre otras. En MATLAB® se tiene diferentes funciones de activación como tansig, hardlim y purelin, entre otras, lo cual facilita las aproximaciones que se requieran hacer, empleando RNA (Redes neuronales artificiales).

Cada unidad de proceso tiene una tarea simple: recibe la entrada de otras unidades o de fuentes externas y procesa la información para obtener una salida que se propaga a otras unidades.

Una red puede tener una estructura arbitraria, pero las capas que contienen estas estructuras están definidas de acuerdo con su ubicación en la topología de la red neuronal. Las entradas externas son aplicadas en la primera capa, y las salidas se consideran la última capa. Las capas internas que no se observan como entradas o salidas se denominan capas ocultas. Por convención, las entradas no se consideran como capa porque no realizan procesamiento.

La entrada total *y* de una unidad *k* es la suma de los pesos de las entradas conectadas, más un bias *b*:

$$y = \sum_i w_i x_i + b \tag{37}$$

Si el peso ω es positivo se habla de una excitación y si el peso es negativo se considera una inhibición de la entrada.

Redes neuronales convolucionales

Una red neuronal convolucional es una arquitectura de red para Deep Learning que aprende directamente a partir de datos. Este tipo de redes es muy usado para identificar patrones en imágenes lo que permite realizar clasificaciones de estás dado un conjunto de imágenes de aprendizaje, en este tipo de redes se aplican filtros o máscaras para generar nuevas imágenes para la siguiente capa de la red.

Las redes convolucionales son, simplemente, redes multicapa en las que algunas capas realizan una operación de convolución en lugar de la tradicional multiplicación matricial de entradas por pesos.

Matemáticamente, la convolución es una operación matemática que se realiza sobre dos funciones para producir una tercera que se suele interpretar como una versión modificada (filtrada) de una de las funciones originales. Gracias a la convolución, así como los filtros usados, se pueden hacer operaciones como detección de bordes Figura 6.1. [26].

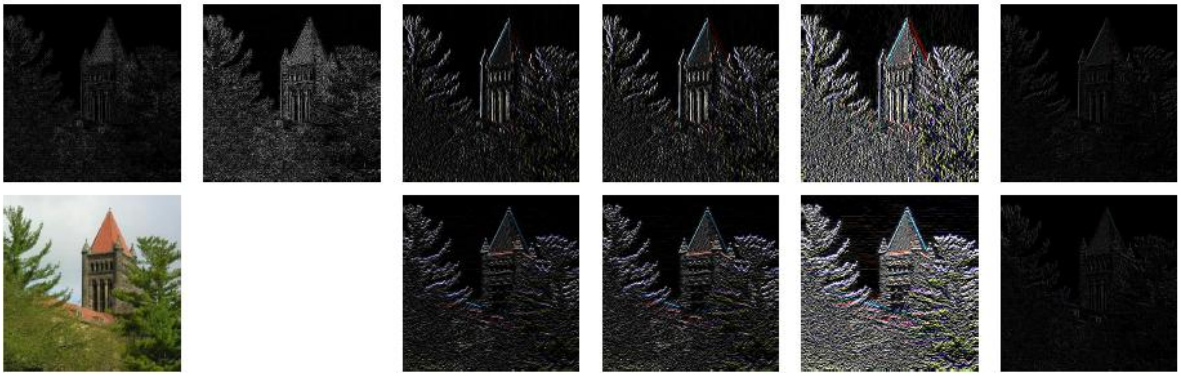


Figura 6.1: Detección de bordes.

Aprendizaje de características, capas y clasificación

Una CNN consta de una capa de entrada, una capa de salida y varias capas ocultas entre ambas.

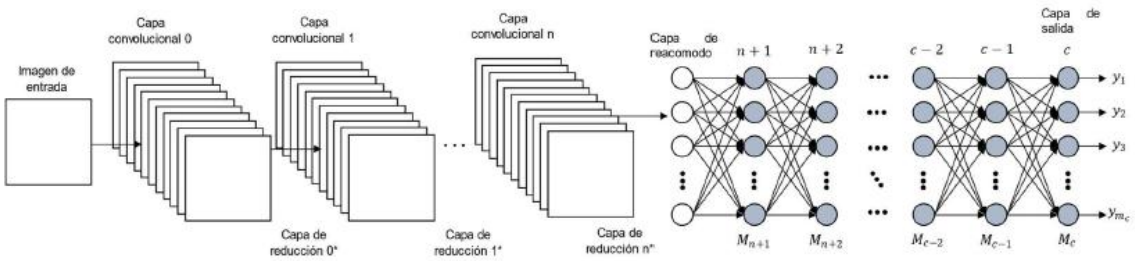


Figura 6.2: Estructura de una red neuronal convolucional [3].

Estas capas realizan operaciones que modifican los datos, con el propósito de comprender sus características particulares. Las 3 capas más comunes son: convolución, activación o ReLU, y agrupación.

Convolución: Aplica un conjunto de filtros convolucionales a las imágenes de entrada; cada filtro activa diferentes características de las imágenes.

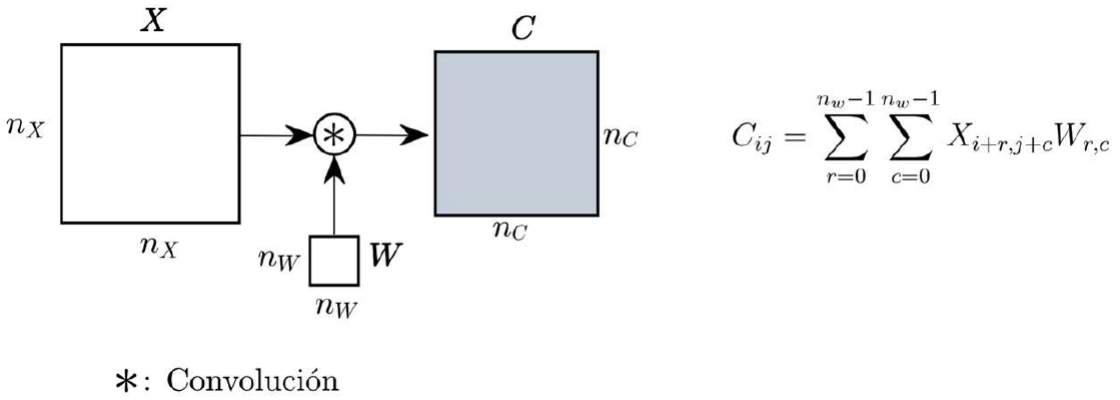


Figura 6.3: Operación de convolución [3].

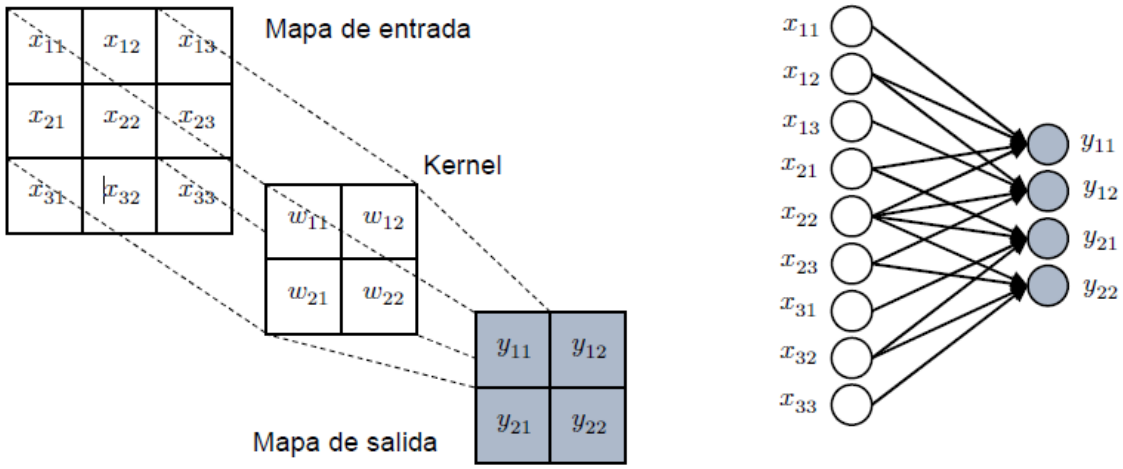


Figura 6.4: Operación de convolución aplicada con un filtro de 2x2 [3].

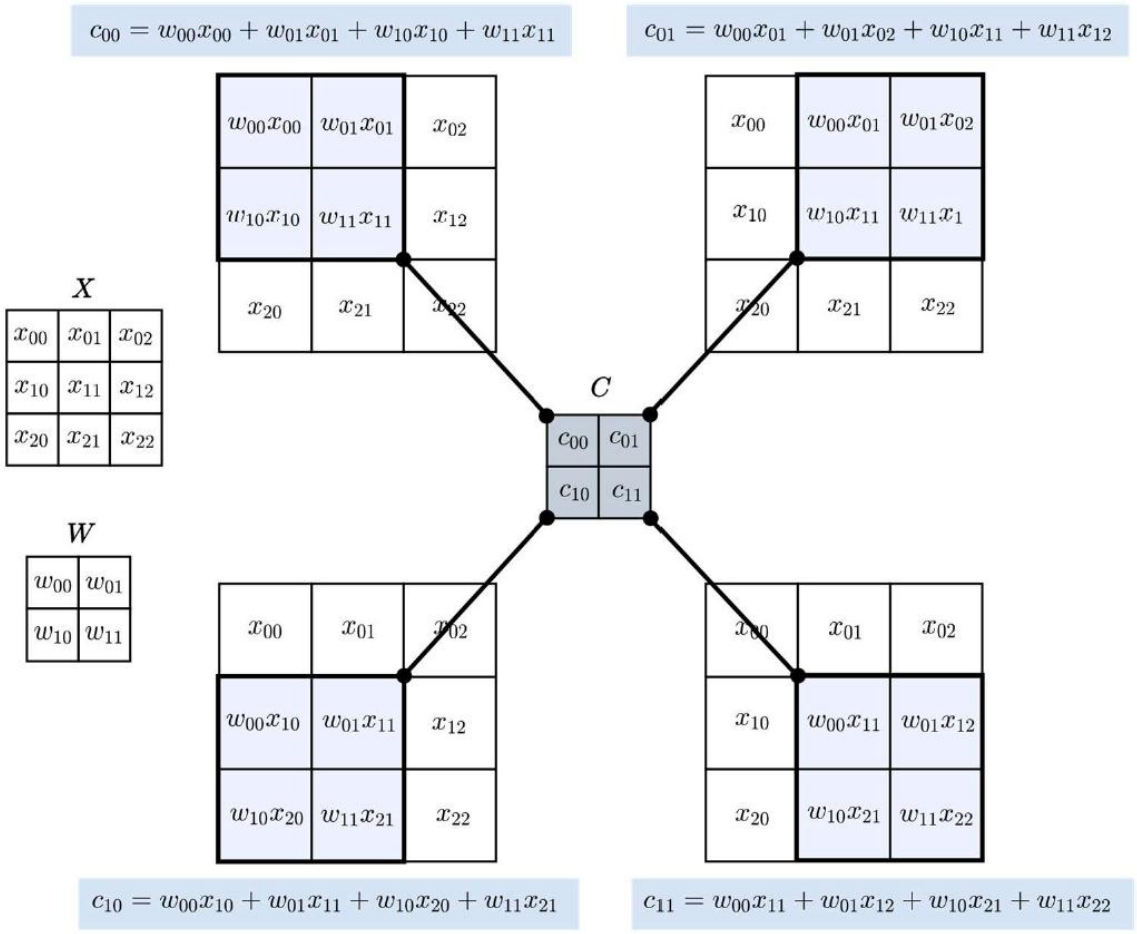


Figura 6.5. Operación de la convolución en dos dimensiones utilizando un filtro convolucional de 2 × 2 [2].

Las redes neuronales convolucionales están basadas en la operación de convolución descrita en la ecuación 2.4, donde X es la entrada y W es el filtro convolucional. Es posible ajustar el paso L de la convolución para abarcar un área más grande con el mismo filtro convolucional.

Para un solo mapa de entrada, se pueden obtener varios mapas de salida utilizando varios kernels:

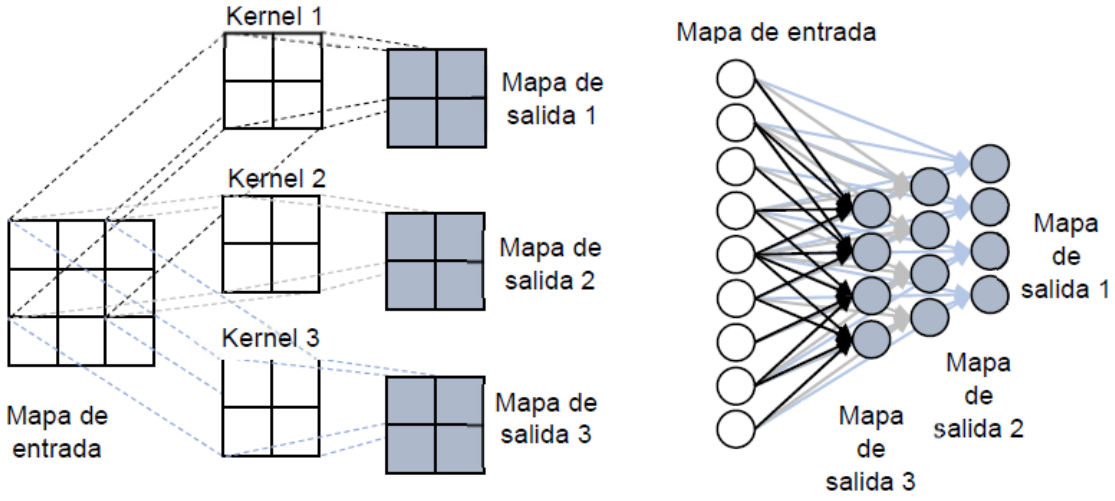
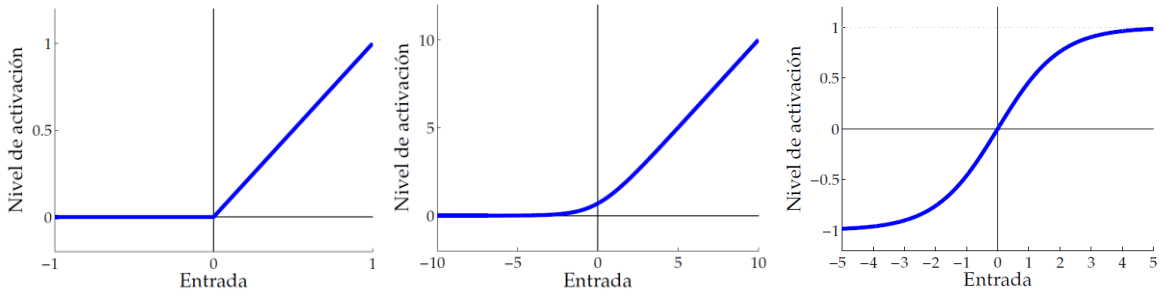


Figura 6.6: Capa convolucional de varios mapas de salida [33].

Unidad lineal rectificada (ReLU): Mantiene los valores positivos y establece los valores negativos en cero, lo que permite un entrenamiento más rápido y eficaz. También se lo conoce como activación, ya que solo las características activadas prosiguen a la siguiente capa.

Existen también otras funciones de activación como:



a) b) c)

Figura 6.7: Funciones de activación a) Función de activación lineal rectificada b) Función de activación softplus c) Función de activación sigmoide bipolar [30].

Agrupación: Simplifica la salida mediante reducción no lineal de la tasa de muestreo, lo que disminuye el número de parámetros que la red debe aprender.

Reducción por valor máximo

En la reducción por valor máximo solo se considera el valor máximo al aplicar el filtro.

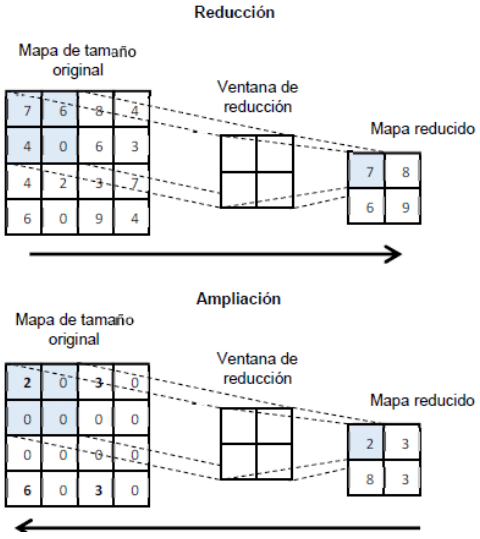


Figura 6.8: Reducción por valor máximo [33].

Reducción por promedio

En la reducción promedio se obtiene el promedio de todos los valores de acuerdo con el tamaño del filtro.

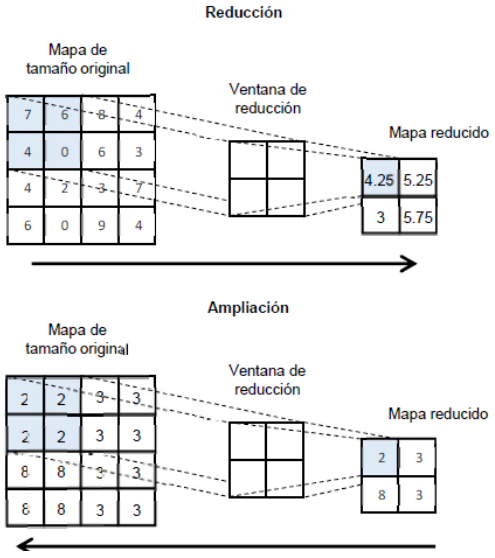


Figura 6.9: Reducción por promedio [33].

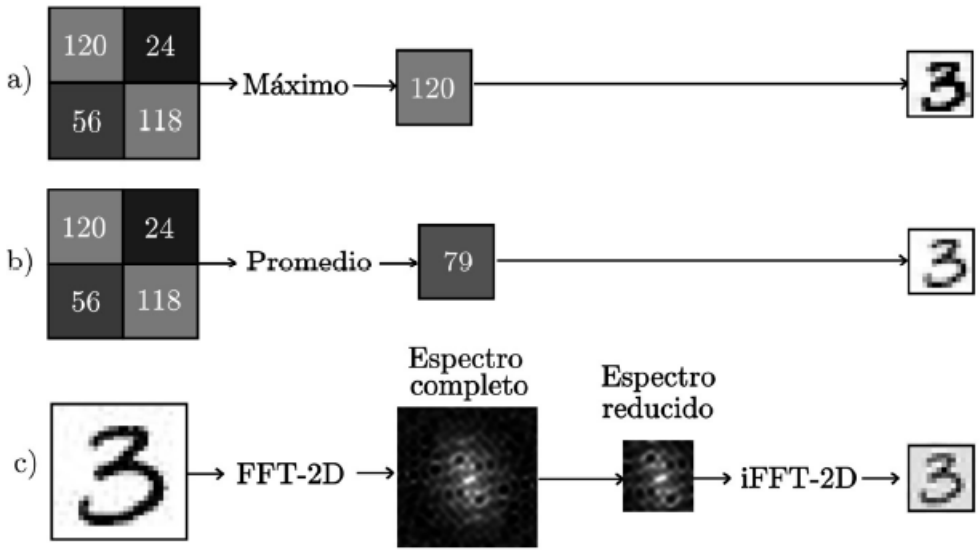
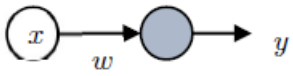


Figura 6.10: Capas de reducción para redes convolucionales. (a) Valor máximo para una ventana de 2x2, (b) Valor promedio para una ventana de 2x2, (c) Reducción de espectro de frecuencia con transformada rápida de Fourier (FFT). iFFT: transformada inversa de Fourier [30].

Estas operaciones se repiten en decenas o cientos de capas.

Entrenamiento de redes neuronales

A continuación, se describe la implementación del gradiente descendente del error para redes neuronales convolucionales.



$$y = xw$$

Figura 6.11: Neurona con una sola entrada y salida

El elemento básico de las RNC es la neuronal artificial mostrada en la Figura 6.12-a donde x_i son las entradas, b es una constante que ayuda al ajuste de la desviación de las señales de entrada. Las entradas son ponderadas matemáticamente con constantes ajustables denominadas pesos sinápticos y son denotadas por w_i. Una función no lineal de activación f determina la salida de la neurona artificial: El comportamiento de la neurona artificial es descrito por la ecuación 1.

$$y = f\left(\sum_{i=1}^{n} x_i w_i + b\right) \tag{38}$$

Una de las primeras arquitecturas de redes neuronales fue el perceptrón que consiste en una capa de entrada y una capa de salida [32]. En la Figura 6.12-b se aprecia un perceptrón de 7 entradas y 3 salidas: las neuronas se muestran en color gris. El perceptrón ha sido utilizado para determinar soluciones no lineales a problemas de clasificación y regresión considerando que las funciones de activación utilizadas en las neuronas son no lineales: Función sigmoide, Rectificación lineal, tangente hiperbólica entre otras [26]. Las redes neuronales actuales consisten en varias capas de neuronas que son interconectadas entre ellas, como se muestra en la Figura 6.12-c.

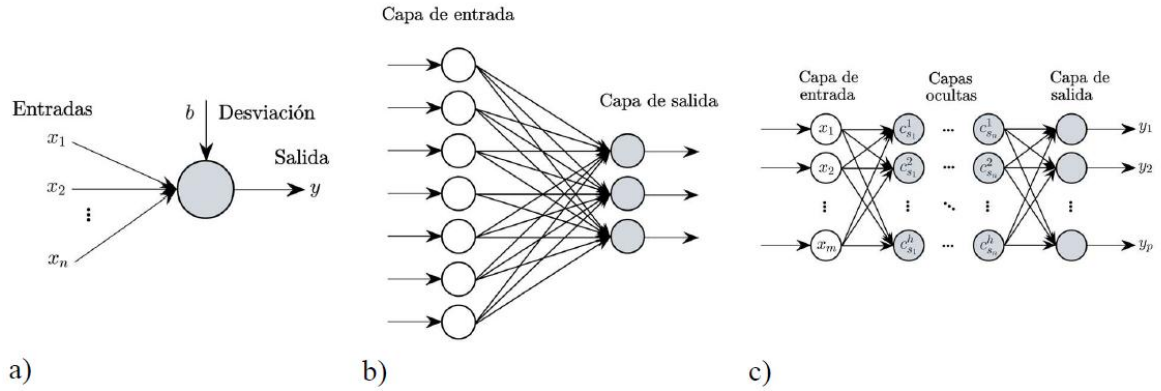


Figura 6.12: Partes de una red neuronal artificial: a) neurona con n entradas, una desviación b y una función de activación f, b) perceptrón simple, c) red neuronal artificial completamente conectada de varias capas ocultas [3].

La Figura 6.13 muestra un ejemplo de una capa convolucional de una RNC. Los datos de entrada son denominados mapa de entrada, los pesos sinápticos son denominados filtro convolucional y también se utilizan funciones no lineales para las salidas. En color azul se muestran las conexiones neuronales parciales: observe que no todas las entradas son asociadas a las neuronas de salida. La operación de convolución es descrita por la ecuación 2, donde $c(i,j)$ es el resultado de salida, S_w es el tamaño del filtro convolucional W, el mapa de entrada es denotado por X, $\{i, j\}$ son los índices donde se colocará el resultado de la convolución, y $\{u, v\}$ son los índices para realizar la convolución.

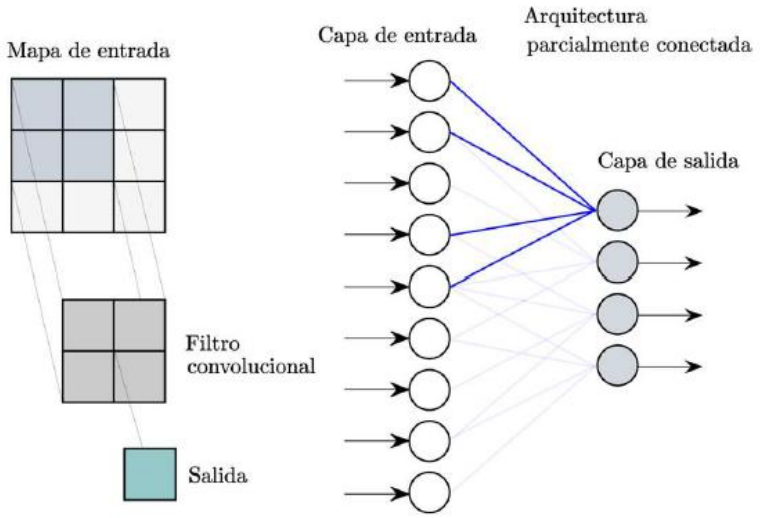


Figura 6.13: Conexión capa de entrada y capa de salida [31].

$$c(i,j) = \sum_{u=0}^{S_w} \sum_{v=0}^{S_w} W(u,v)X(u+i, v+j) \tag{39}$$

El entrenamiento de una red multicapa se suele realizar utilizando un algoritmo de propagación de errores hacia atrás denominado backpropagation

Aprendizaje supervisado (gradiente descendente y backpropagation):

La red recibe una realimentación directa para cada uno de los ejemplos del conjunto de entrenamiento. Esa señal, en forma de salida deseada, se puede utilizar directamente para ajustar los parámetros de la red, tal como hacen las técnicas de optimización usadas con backpropagation.

La retro propagación del error es un entrenamiento supervisado que se emplea para redes multicapa, donde se ajusta el valor de los pesos en función del error generado. Esta técnica es muy empleada ya que permite tener un método de optimización que se encuentra al definir el gradiente del error y minimizarlo con respecto a los parámetros de la red neural.

El algoritmo de propagación del error entrena a la red neuronal eficientemente. Consta de dos pasos:
1. Ejecución de la red neuronal (desde la capa de entrada hasta la de salida).
2. Propagación del error (dese la capa de salida hasta la de entrada).
3. Ajuste de pesos sinápticos.
4. Repetir desde 1 hasta que el error converja a cero o hasta un determinado número de veces.
Se considera la red neuronal FF multicapa para describir los pasos del entrenamiento.

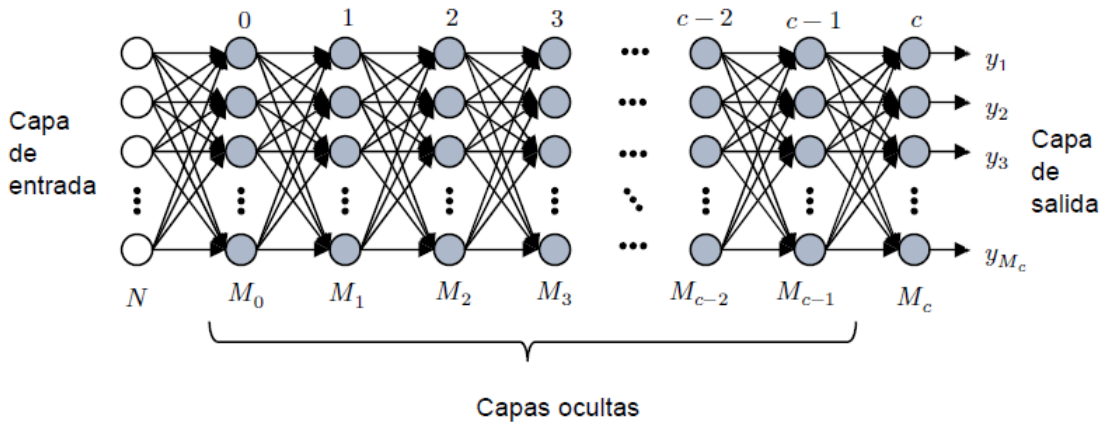


Figura 6.14: Red neuronal FF de c capas [31].

Entrenamiento de redes multicapa

Las redes multicapa de tipo feed-forward en ocasiones se denominan backpropagation networks porque el algoritmo de entrenamiento que se suele utilizar con ellas está basado en la propagación hacia atrás del error; esto es, en el uso de backpropagation.

Las redes multicapa usadas en deep learning tienen una capa de entrada, múltiples capas ocultas y una capa de salida. Desde un punto de vista formal, podemos verlas como una función matemática f que, a partir de un vector de entrada x, obtiene un vector de salida y = f(x).

Dado un conjunto de entrenamiento en forma de pares (x, y), el objetivo del algoritmo de entrenamiento es ser capaz de aproximar la función f de forma que para cada posible entrada x se obtenga una salida ˆ y = f(x)

Entrenamiento de neuronas

Una neurona lineal, también conocida como filtro lineal, se limita a realizar el producto escalar de los vectores de pesos y entradas:

$$y = f(x) = \sum_i w_i x_i = w^T x = w \cdot x \qquad (40)$$

Si queremos ajustar los parámetros de una neurona lineal, su vector de pesos w, tendremos que definir una función de error, coste o pérdida que nos permita decidir qué conjuntos de pesos son mejores y cuáles son peores. Por ejemplo, podemos recurrir al error cuadrático medio [MSE: Mean Squared Error] medido sobre el conjunto de entrenamiento:

$$MSE = \frac{1}{n}\sum_j \left(t_j - y_j\right)^2 \qquad (41)$$

donde y_j es la salida proporcionada por la neurona lineal y t_j es el valor deseado, nuestro objetivo, tal como aparece en el conjunto de entrenamiento.

Cálculo del error mediante el gradiente

Si optamos por resolver analíticamente el problema de optimización, calculamos la derivada de la función de coste y la igualamos a cero. En el ejemplo anterior, para el que hemos supuesto una función de error cuadrática, su derivada será lineal y nos dará como resultado un sistema de ecuaciones lineales, con una ecuación para cada parámetro de la red.

En la Figura 6.15 puede verse el diagrama general del entrenamiento de las redes neuronales, para implementar el gradiente descendente del error en el entrenamiento de la red neuronal convolucional, se hizo uso del algoritmo de propagación hacia atrás [3] que propaga el error que hay entre la salida de la red y la salida deseada. Primero se extraen las imágenes A junto con sus etiquetas E del repositorio de entrenamiento correspondiente. La imagen extraída del repositorio es introducida a la red neuronal que aún no ha sido entrenada y se toma su salida para compararla con la etiqueta correspondiente. Posteriormente se calcula la función del costo del error y se propaga hasta la primera capa. Todo esto se calcula usando la regla delta y se repite el proceso las N veces.

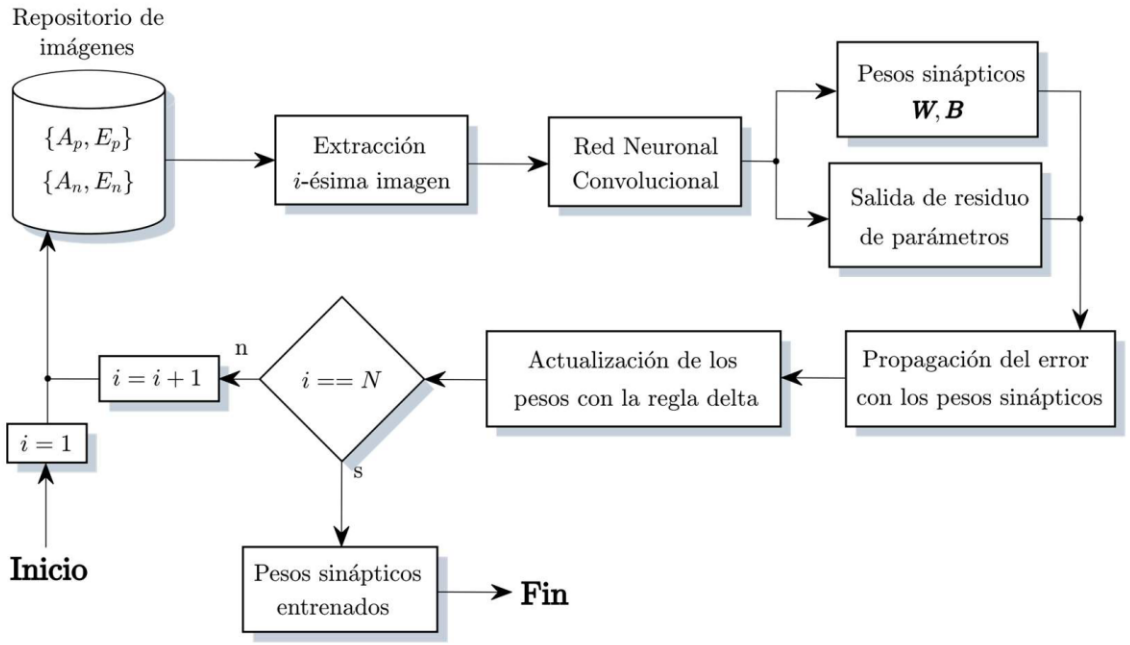


Figura 6.15: Entrenamiento de redes neuronales convolucionales.

Se calcula el gradiente de E con respecto a cada peso sináptico w, si solo existe un w, por lo que el gradiente es:

$$\frac{\partial E}{\partial w} = \left(\frac{\partial E}{\partial y}\right)\left(\frac{\partial y}{\partial w}\right) \tag{42}$$

Donde para la función de costo cuadrática, se tiene que el gradiente es:

$$\frac{\partial E}{\partial w} = -(y_d - y)x \tag{43}$$

La variación de los pesos sinápticos se realiza con la regla delta que utiliza un parámetro η denominado tasa de aprendizaje:

$$w_n = w_{n-1} - \eta\left(\frac{\partial E}{\partial w}\right) \tag{44}$$

$$E_j = \frac{1}{2}\left(y_{dj} - y_j\right)^2, \qquad \forall j \in \{1,2,\dots,M\} \tag{45}$$

Donde

$$\nabla E_j = \left[\frac{\partial E_j}{\partial w_{1j}}, \frac{\partial E_j}{\partial w_{2j}}, \dots, \frac{\partial E_j}{\partial w_{3j}}\right]^T \tag{46}$$

La variación de la función de costo con el gradiente descendente y la regla delta siempre es igual o menor que cero:

$$\Delta w_j = -\eta \nabla E_j$$

$$\Delta E_j \cong \nabla \mathrm{E_j} \cdot \Delta w_j$$

$$\Delta E_j \cong -\eta \left|\left|\nabla E_j\right|\right|^2 \tag{47}$$

Cuando se utiliza una función de activación $f(z_j)$

$$z_j = \sum_{i=1}^{N} w_i x_i, \qquad y_j = f\left(z_j\right) \tag{48}$$

El gradiente de E se determina con la regla de la cadena para las derivadas:

$$\nabla E_j = \left[\left(\frac{\partial E_j}{\partial y_{1j}}\right)\left(\frac{\partial y_j}{\partial z_{1j}}\right)\left(\frac{\partial z_j}{\partial w_{1j}}\right), \left(\frac{\partial E_j}{\partial y_{2j}}\right)\left(\frac{\partial y_j}{\partial z_{2j}}\right)\left(\frac{\partial z_j}{\partial w_{2j}}\right), \dots, \left(\frac{\partial E_j}{\partial y_{Nj}}\right)\left(\frac{\partial y_j}{\partial z_{Nj}}\right)\left(\frac{\partial z_j}{\partial w_{Nj}}\right)\right]^T \tag{49}$$

En la Figura 6.16 se puede observar de manera gráfica la interconexión para la propagación del error entre las capas completamente conectadas.

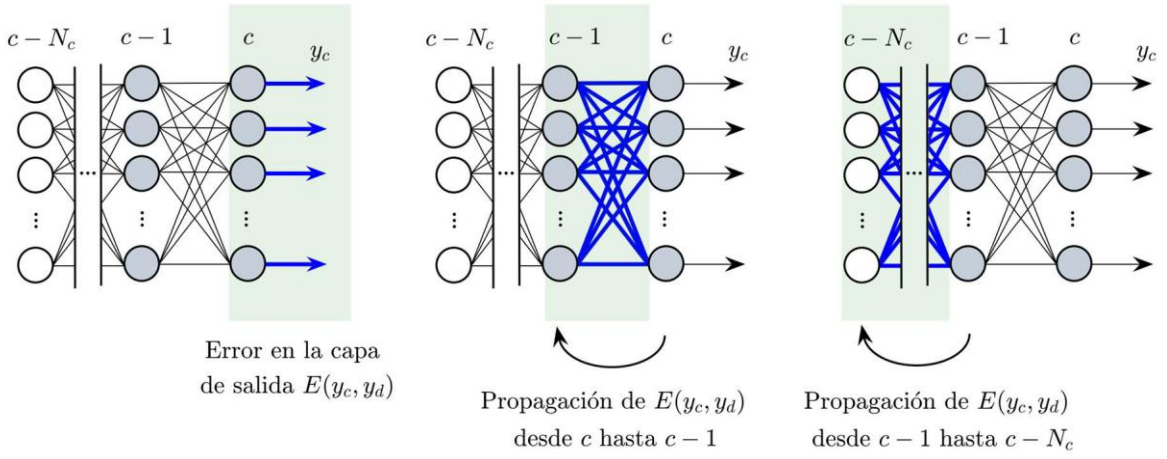


Figura 6.16: Propagación del error entre capas completamente conectadas.

En la Figura 6.16 se puede observar como el error de la capa de salida se propaga hacia atrás usando los pesos sinápticos, en este caso se muestra una representación para un mapa de entrada de 3x3 y un filtro de 3x3; estas salidas usan la variable delta para determinar las ecuaciones para poder actualizar los pesos W y los valores de b, en la Figura 6.17 se observan las conexiones de $w_{11}, w_{12}, w_{21}, w_{22}$. Para poder encontrar ΔW se van sumando las contribuciones de los δ de la salida de las entradas $x_{i,j}$

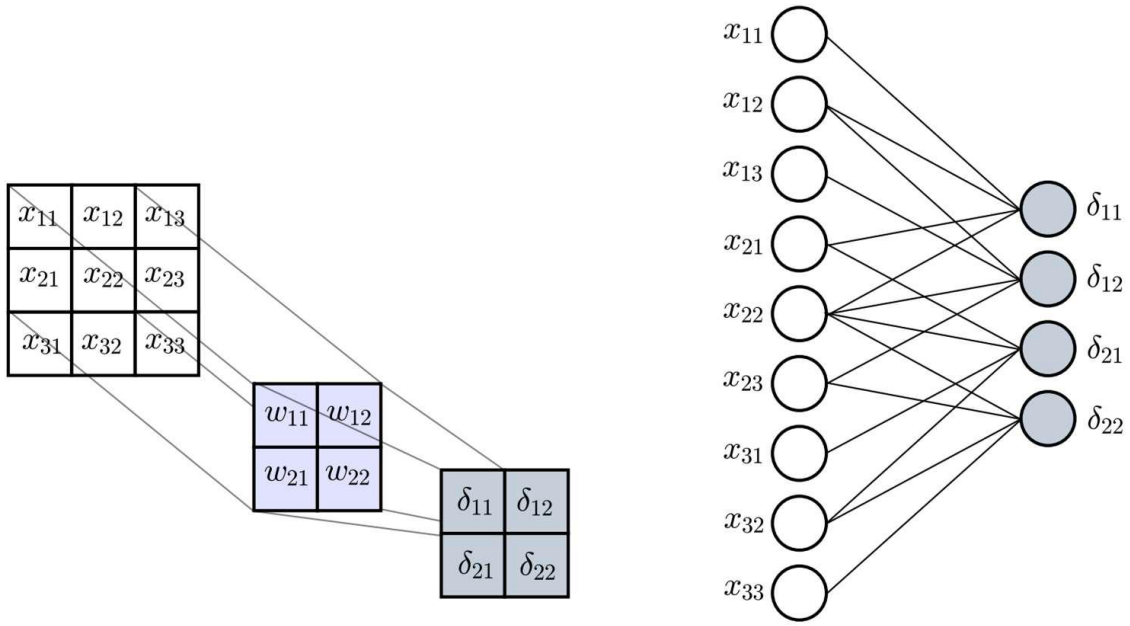


Figura 6.17: Ejemplo para determinar las ecuaciones de actualización de los filtros convolucionales en una capa convolucional.

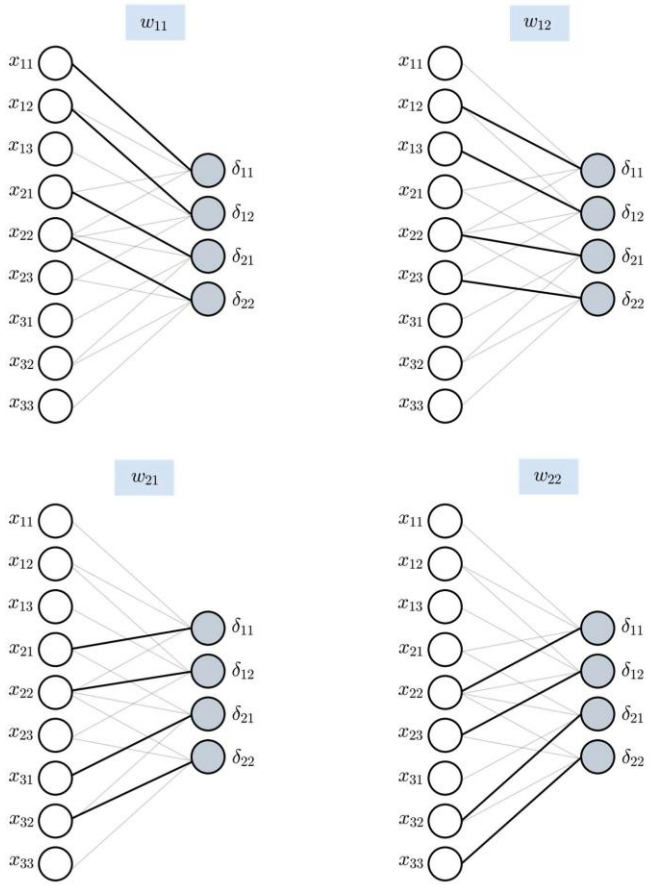


Figura 6.18: Conexiones ponderadas por el filtro convolucional de la Figura 6.17

$$\Delta w_{11} = \eta(\delta_{11}x_{11} + \delta_{12}x_{12} + \delta_{21}x_{21} + \delta_{22}x_{22})$$

$$\Delta w_{12} = \eta(\delta_{11}x_{12} + \delta_{12}x_{13} + \delta_{21}x_{22} + \delta_{22}x_{23})$$

$$\Delta w_{21} = \eta(\delta_{11}x_{21} + \delta_{12}x_{22} + \delta_{21}x_{31} + \delta_{22}x_{32})$$

$$\Delta w_{22} = \eta(\delta_{11}x_{22} + \delta_{12}x_{22} + \delta_{21}x_{32} + \delta_{22}x_{33})$$

$$\Delta W = \eta \text{conv}(\text{X}, \delta) \tag{50}$$

En el caso de la propagación del error hacia atrás entre las capas convolucionales, la representación gráfica puede verse en la Figura 6.19, con sus respectivas ecuaciones para su cálculo.

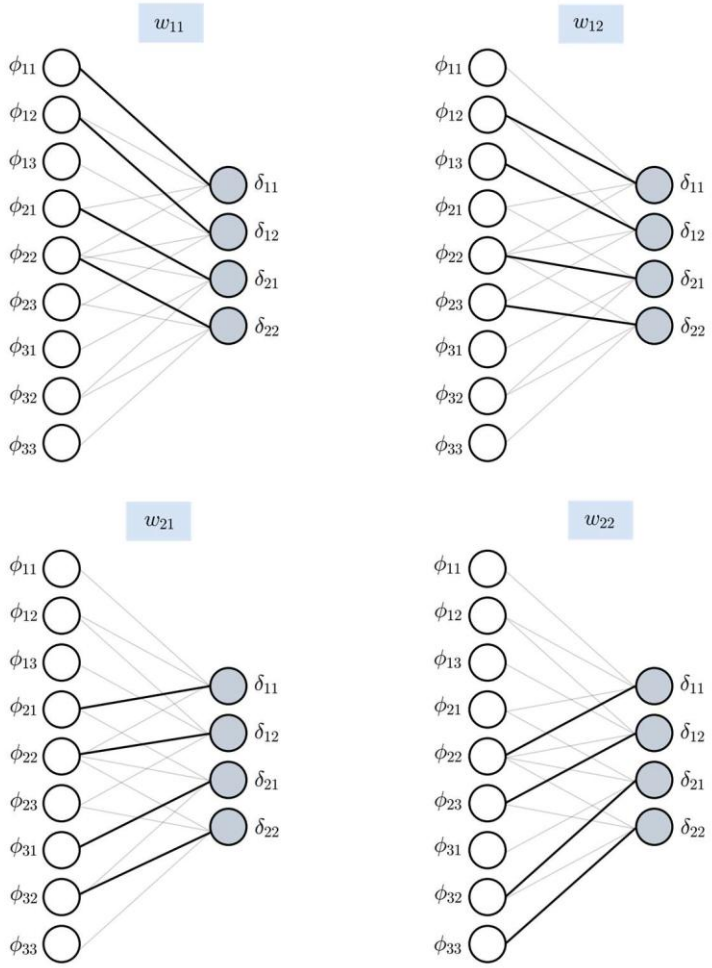


Figura 6.19: Propagación del error entre capas convolucionales

$$\phi_{11} = \delta_{11} w_{11}$$

$$\begin{aligned}
\phi_{12} &= \delta_{12} w_{11} + \delta_{11} w_{12} \\
\phi_{13} &= \delta_{12} w_{12} \\
\phi_{21} &= \delta_{21} w_{11} + \delta_{11} w_{21} \\
\phi_{22} &= \delta_{22} w_{11} + \delta_{21} w_{12} + \delta_{12} w_{21} + \delta_{11} w_{22} \\
\phi_{23} &= \delta_{22} w_{12} + \delta_{12} w_{22} \\
\phi_{31} &= \delta_{21} w_{21} \\
\phi_{32} &= \delta_{22} w_{21} + \delta_{21} w_{22} \\
\phi_{33} &= \delta_{22} w_{22}
\end{aligned} \qquad (51)$$

La correspondiente representación gráfica de las ecuaciones anteriores puede verse en la Figura 6.20.

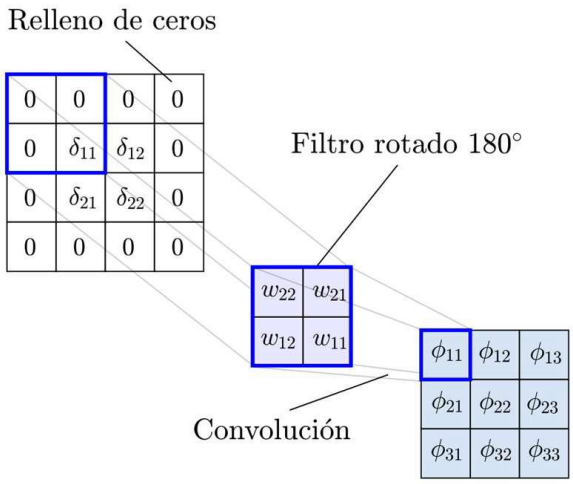


Figura 6.20: Operación de convolución completa para la propagación del error entre las capas convolucionales.

6.2 Algoritmo de identificación paramétrica

En la Figura 6.21, se describe la metodología para realizar la identificación paramétrica del brazo robot, esta se basa en usar las posiciones, velocidades y aceleraciones para generar un repositorio de imágenes de entrenamiento de la red neuronal convolucional. Es necesario que el robot siga una trayectoria predefinida usando controlador, una vez que se ha conseguido que el robot siga la trayectoria, lo siguiente es estimar la velocidad y la posición de cada uno de los elementos del sistema para construir el torque aplicado. Una vez que se conoce este, lo siguiente es la creación de las imágenes que corresponderán al repositorio de entrenamiento de la red neuronal convolucional. Para la generación de múltiples imágenes se variarán los diferentes parámetros de forma seudo aleatoria. La generación de cada imagen corresponderá se explicará más adelante, resultando una matriz de n x n correspondiente a cada una de las imágenes generadas.

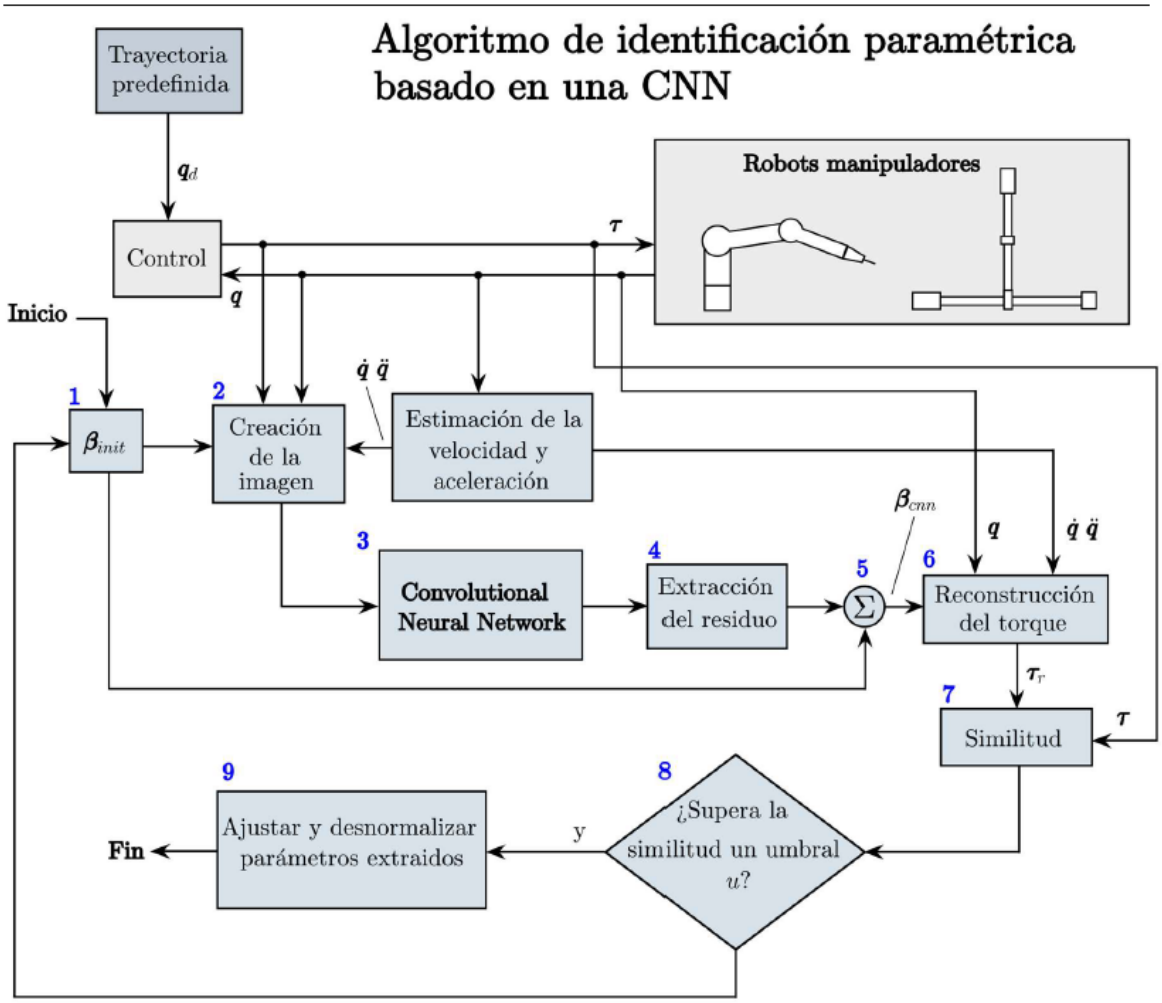


Figura 6.21: Diagrama general de la metodología propuesta de identificación paramétrica para los dos casos de estudio [3].

Generación de imágenes con las señales del brazo robot.

Para generar las imágenes de entrada se usarán las funciones ψ y α, donde $\psi = \alpha = \tau_{motor}$, estos al ser vectores, y ser multiplicados generan una matriz expresada en la Figura 6.22. Estos vectores son el resultado obtenido al realizar la simulación del modelo dinámico, es decir los torques aplicados a ambos grados de libertad para lograr la trayectoria deseada en este trabajo. Los torques obtenidos en la simulación se procesaron a fin de considerar 100 elementos del torque obtenido para cada articulación, estos cien elementos corresponden a la respuesta obtenida para un 1.5 periodos de la trayectoria senoidal.

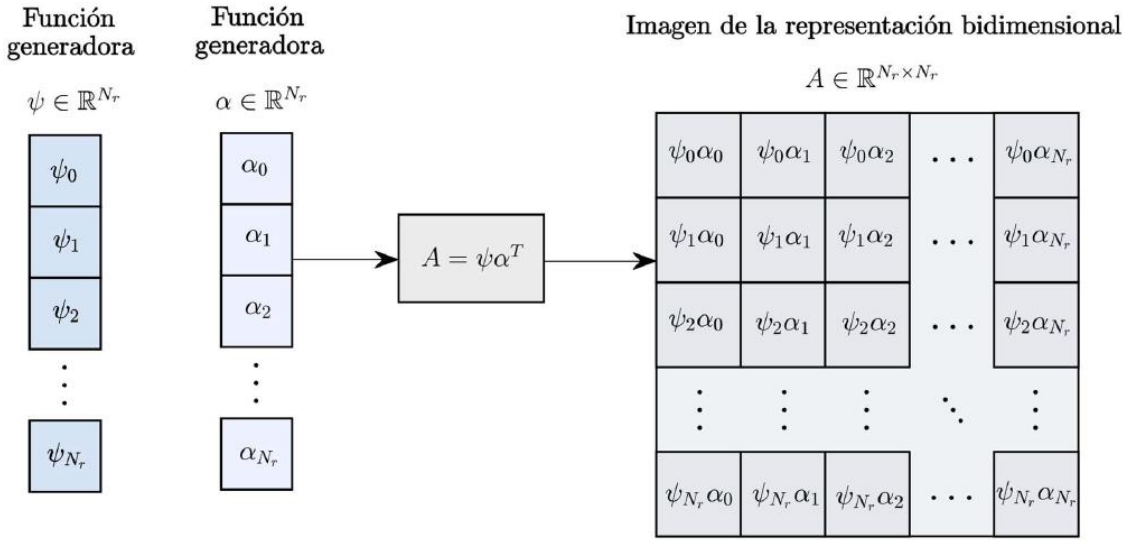


Figura 6.22: Generación de las imágenes de entrada [31].

Descrito de forma detallada se tiene que la ecuación que describe los torques del brazo robot están descritos por la ecuación

$$M\ddot{q} + C(q,\dot{q}) + B\dot{q} + g(q) + K(q) = \tau \tag{52}$$

donde M, $C,\ B, K,\ \in \mathbb{R}^{nxn}, g(q) \in \mathbb{R}^n$

El vector $\tau = \begin{bmatrix}\tau_{m1}\\ \tau_{m2}\end{bmatrix}$ este compuesto a su vez por dos vectores donde por $\tau_{mn} = \begin{bmatrix}\tau_{mn_1}\\ \tau_{mn_2}\\ \vdots \\ \tau_{mn_{100}}\end{bmatrix}$, siendo los vectores de esta forma, construimos la matriz correspondiente a cada imagen del repositorio usando la multiplicación del vector de torques con su correspondiente vector transpuesto, estos dos vectores son la respuesta obtenida al implementar el control de trayectoria.

$$\begin{bmatrix}\tau_{mn_1}\\ \tau_{mn_2}\\ \vdots \\ \tau_{mn_{100}}\end{bmatrix}\begin{bmatrix}\tau_{mn_1} & \tau_{mn_2} & \cdots & \tau_{mn_{100}}\end{bmatrix} =$$

$$\begin{bmatrix}\tau_{mn_1}\tau_{mn_1} & \tau_{mn_1}\tau_{mn_2} & \cdots & \tau_{mn_1}\tau_{mn_{100}}\\ \tau_{mn_2}\tau_{mn_2} & \tau_{mn_2}\tau_{mn_2} & \cdots & \tau_{mn_2}\tau_{mn_2}\\ \vdots & \vdots & \ddots & \vdots \\ \tau_{mn_{100}}\tau_{mn_1} & \tau_{mn_{100}}\tau_{mn_2} & \cdots & \tau_{mn_{100}}\tau_{mn_{100}}\end{bmatrix} = A \tag{54}$$

Donde A es la imagen generada, el siguiente paso es normalizar la imagen obtenida de acuerdo con la ecuación

$$Z = \frac{A - \min(A)}{\max(A) - \min(A)} \tag{55}$$

Donde Z es la normalización de cada una de las imágenes que conforman el repositorio de imágenes, ya que las redes neuronales convolucionales tienen un mejor desempeño cuando los datos de entrada se encuentran normalizados [3].

Algoritmo para la generación de imágenes.

En este apartado se revisará la metodología para la generación de imágenes a partir de las señales del robot, la señal de entrada está conformada por las señales de posición, velocidad, aceleración y torque del robot, usando este algoritmo se tendrá un repositorio de imágenes para realizar la identificación paramétrica

Algoritmo 1. Algoritmo para generar imágenes a partir de las señales del robot cartesiano.

Input: Posición q, torque τ , número de iteraciones N.
$\dot{q} \leftarrow$ **dct2dev**(q, h, c_f)
$\ddot{q} \leftarrow$ **dct2dev**$(\dot{q}, h, c_f)$
$\{q_s, \dot{q}_s, \ddot{q}_s, \tau_s\} \leftarrow$ dct2dev$(\dot{q}, \ddot{q}, \tau, 100)$
for $k = 1 : N$ **do**
$\beta \leftarrow$ **rand**(1, 14)
$\tau_r \leftarrow$ **Dynmod**$(\ddot{q}, \dot{q}, \beta)$
$\psi \leftarrow 2 * \tau - \tau_r$
$\alpha \leftarrow \tau$
$A \leftarrow \psi * \alpha$
$A_{p,n} \leftarrow norm(A)$
$L_{p,n} \leftarrow \beta r - \beta$
end for
Output: Imágenes $A_{p,n}$ con etiquetas $L_{p,n}$

El algoritmo 1 describe el proceso para aproximar la velocidad y la aceleración del robot, en el ciclo **for** se generan el conjunto de datos aleatorios para reconstruir la señal de torque con las señales de posición y velocidad obtenidas del robot. Posteriormente se usa la multiplicación de vectores para generar cada una de las imágenes del repositorio, finalmente se normaliza esta imagen y se guarda el residuo paramétrico en una etiqueta para su posterior referencia.

En la Figura 6.23 se puede ver el diagrama general para la generación de las imágenes, en este puede verse el uso del modelo dinámico obtenido para poder generar las señales necesarias como son posición, velocidad y aceleración, las cuales son fundamentales para generar cada una de las imágenes.

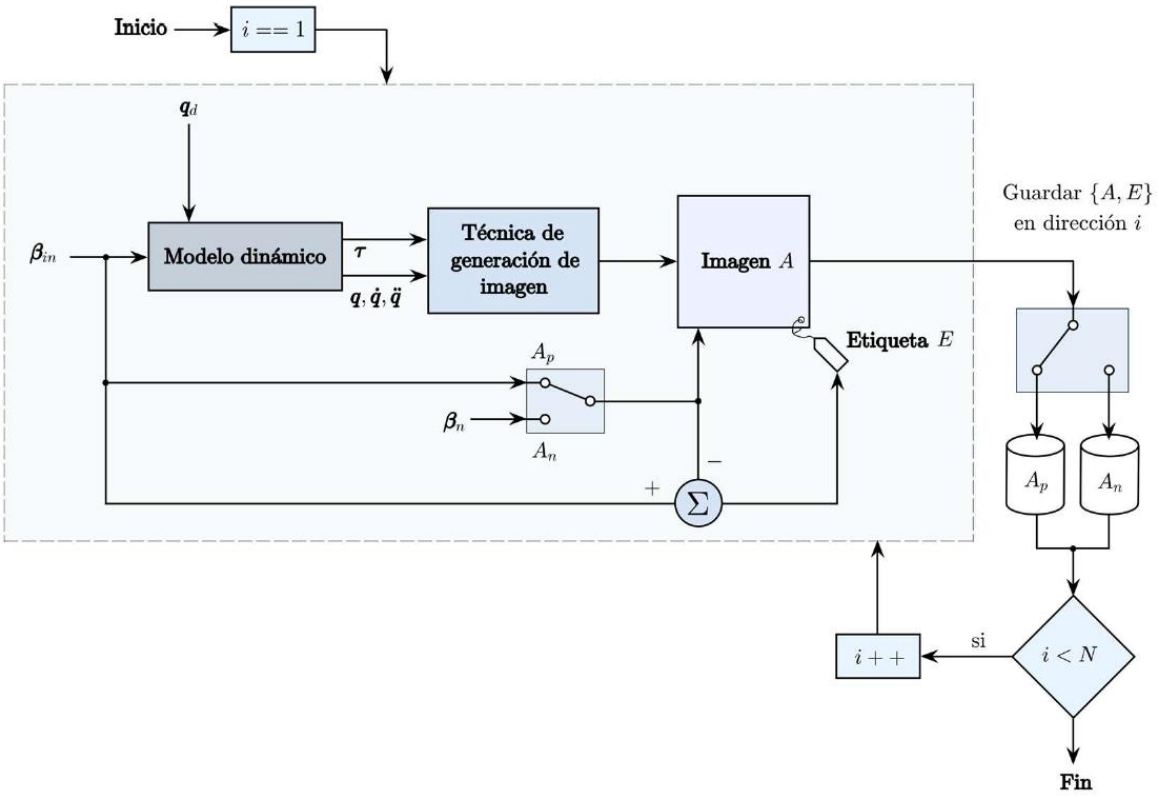


Figura 6.23: Diagrama general de la generación del repositorio de imágenes.

Métrica de evaluación

La metodología usada para evaluar la similitud entre el torque de entrada y el que es reconstruido usando los parámetros identificados por la metodología que se presenta esta descrita por la ecuación (56), donde se considera la señal en el tiempo y su espectro en frecuencia con la transformada discreta del coseno

$$a = 1 - \frac{\left|\sum_{i=1}^{N} x_i - \sum_{i=1}^{N} y_i\right|}{\left|\sum_{i=1}^{N} (|x_i| + |y_i|)\right|}$$
$$X = dct_{II}(x)$$
$$Y = dct_{II}(y)$$

$$X_n = \frac{X}{\max(X)}$$

$$Y_n = \frac{Y}{\max(Y)}$$

$$b = 1 - \frac{\left|\sum_{i=1}^{N} |X_{ni} - Y_{ni}|\right|}{\left|\sum_{i=1}^{N} (|X_{ni}| + |Y_{ni}|)\right|}$$

$$\mathbf{metr2ev} = (\mathrm{ab})^{0.4} \tag{56}$$

Donde z es la similitud con un rango entre 0 y 1, esta salida es ajustada por un valor de 0.4.

Red neuronal propuesta para la identificación paramétrica

La propuesta de red neuronal convolucional usada en este trabajo, esta red consta de 3 etapas de convolución, para la primer capa de convolución se propone como entrada imágenes de 100 x100 pixeles, ya que existe trabajos previo donde esta resolución contiene la suficiente información para ser procesada dentro de una imagen con dimensiones pequeñas, este conjunto de imágenes será generado usando las señales de torque de los motores del robot, esto es propone ya que es tanto en la simulación del modelo dinámico como en la parte experimental se puede construir nuevamente el torque usando los datos del robot, como son su posición, velocidad y aceleración, a continuación se detalla cada capa de la red neuronal convolucional.

Para la primera capa de convolución tenemos imágenes de 100x100 pixeles, posteriormente se aplicarán 10 filtros de un tamaño de 9x9, posteriormente se aplicará un bias de 10 valores diferentes, para después pasar por una función de activación ReLU, de esta primera capa resultarán imágenes de 92x92x10 pixeles, para este trabajo se propone una reducción por valor máximo usando filtros de 2x2, resultando en imágenes de 46x46 pixeles que serán la entrada de la segunda capa de convolución.

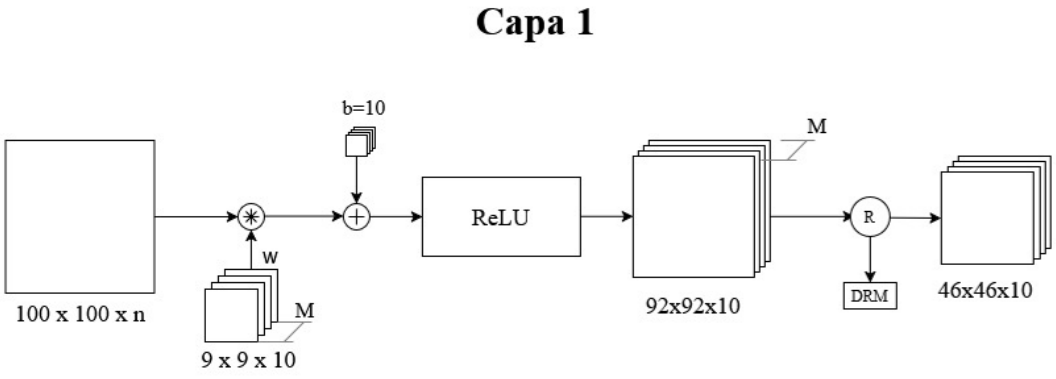


Figura 6.24: Primer capa de convolución.

Para la segunda etapa se proponen 10 filtros de 5x5, bias de 10 valores y con función de activación ReLU, por lo que resultará una salida de 42x42x10 y al hacer nuevamente una reducción de 2x2 tendremos un conjunto de imágenes de 21x21x10.

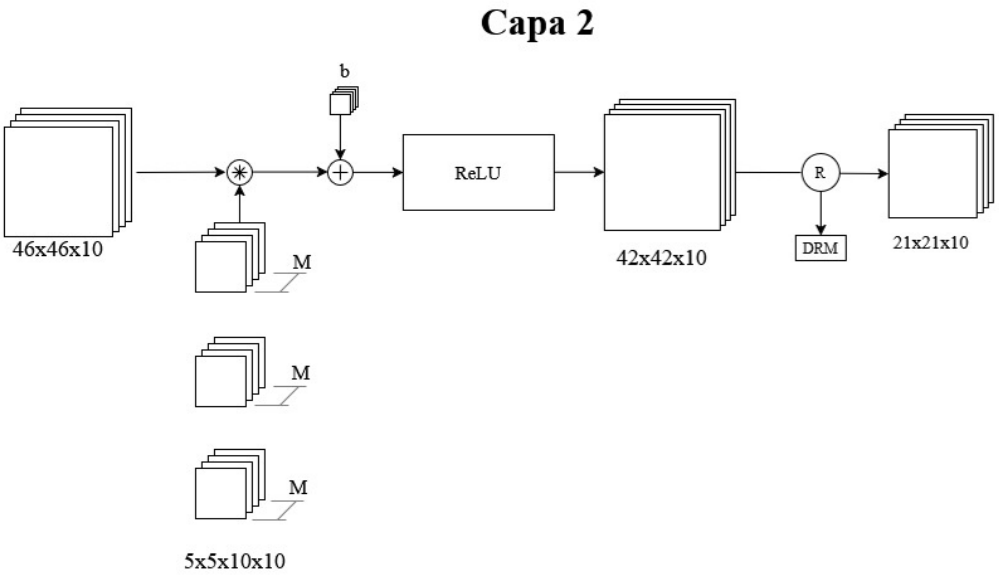


Figura 6.25: Segunda Capa de convolución.

Finalmente se aplicará una tercera capa de convolución al que se le aplicará un filtro de 3x3x10 con un bias de 10 valores y función de activación ReLU, resultando finalmente en un conjunto de imágenes de 21x21x10 pixeles, en la etapa de salida no se hará reducción sino una operación diferente llamada transformación o aplanamiento.

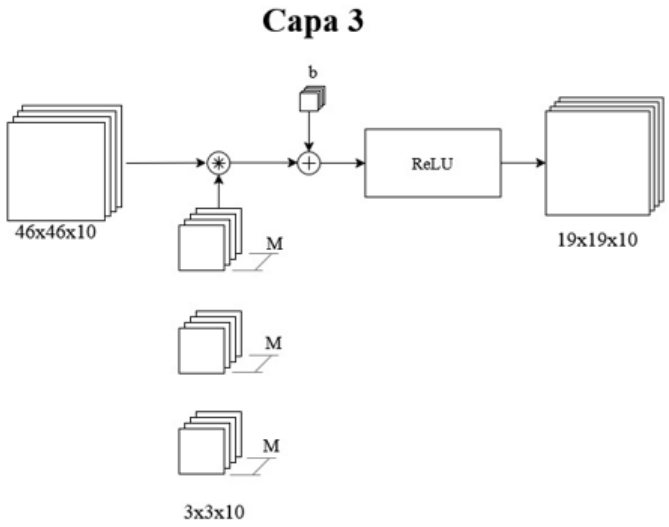


Figura 6.26: Tercera capa de convolución.

En la Capa 4 tenemos la operación de transformación, la cual se encarga de convertir los mapas de características en un vector de 3610 neuronas y una salida de 100 neuronas, estas están completamente conectadas. En esta capa la función de activación es la función ReLU.

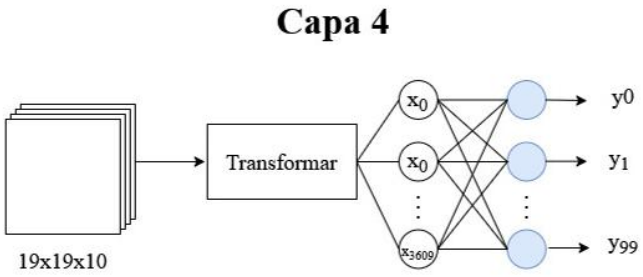


Figura 6.27: Cuarta capa completamente conectada.

En la Figura 6.28 tenemos la Capa 5 igual completamente conectada, en esta se tienen 100 neuronas de entrada y 100 neuronas de salida y se usa la función de activación ReLU.

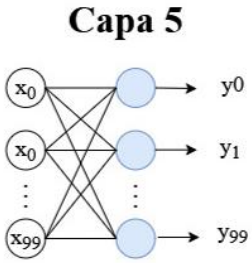


Figura 6.28: Quinta capa completamente conectada.

En la Figura 6.29 se tiene la última capa de la red neuronal convolucional, en esta se tienen 100 neuronas de entrada y 7 neuronas de salida completamente conectadas las cuales dan como salida el valor de los parámetros deseados, en ambos casos, para el primer y segundo grado de libertad se usó la misma configuración de red, en el siguiente capítulo se mostrarán los resultados obtenidos al realizar la identificación paramétrica.

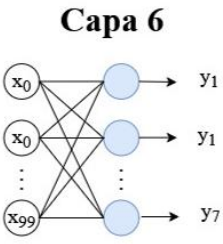


Figura 6.29: Capa de salida.

En la Figura 6.30 se muestra la red neuronal convolucional completa, esta se pueden ver todas las capas de convolución, las capas completamente conectadas y la capa de salida, los valores obtenidos en la capa de salida corresponden a los valores de los parámetros que se desean identificar.

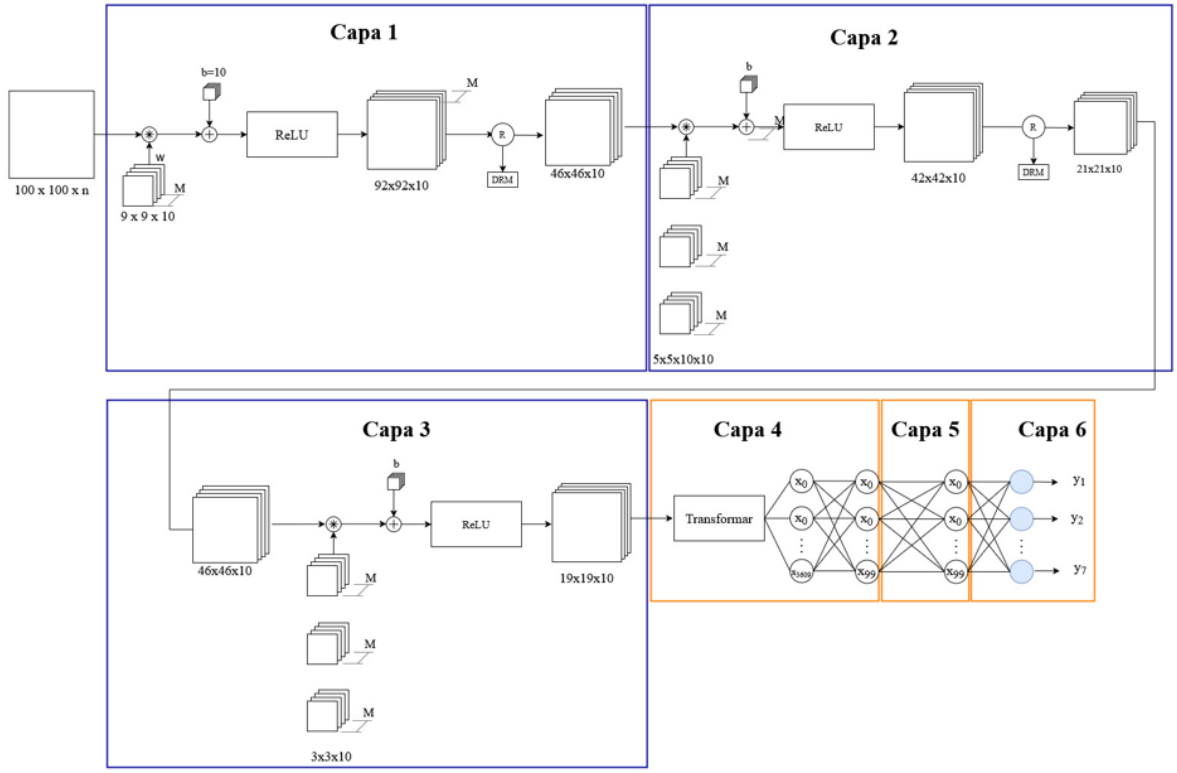


Figura 6.30: Propuesta de red neuronal convolucional para la identificación paramétrica.

Finalmente se puede construir una tabla 6.1 que corresponde a las neuronas de cada capa que conforma la red neuronal convolucional, la cantidad de neuronas que se usan se muestran en la Tabla 6.1

Tabla 6.1: Dimensión de la red neuronal convolucional propuesta para la identificación paramétrica del brazo robot de 2 gdl.

Tamaño	Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Capa 6
Entrada	100x100	46x46x10	21x21x10	3610x1	100x1	100x1
Salida	46x46x10	21x21x10	3610x1	100x1	100x1	7x1
Filtro	9x9x1x10	5x5x10	3x3x10	3610x100	100x100	100x7
Neuronas	84600	17640	3610	100	100	7
Pesos	820	2510	910	361100	10100	707

El total de neuronas usadas es de 106,057 y el número de pesos sinápticos es de 376,147.

Conclusiones

En este capítulo se dieron los fundamentos de las redes neuronales convolucionales, así como de su entrenamiento, se dio un pequeño resumen sobre sus partes, así como de su entrenamiento, en este trabajo se usa la retro propagación del error hacia atrás usando la ecuación de costo para el entrenamiento y su validación.

Se presentó la estructura de red neuronal convolucional a usar en este trabajo, así como la explicación de cada una las capas que la componen, para finalmente tener el diseño completo de la red, por lo que el siguiente paso es usar esta red para realizar el entrenamiento, tal como se explicará en los siguientes capítulos

Capítulo 7. Resultados

En este capítulo se mostrarán los resultados obtenidos para el modelo dinámico y la identificación paramétrica usando redes neuronales convolucionales, se partirá con un control de posición para verificar que el modelo dinámico cumpla con los resultados esperados como son llegar a una posición definida con las consideraciones necesarias, también se mostrarán los resultados obtenidos para un torque constante, con su correspondiente trayectoria.

Posteriormente se mostrarán los resultados obtenidos para la identificación paramétrica usando redes neuronales convolucionales, se comenzará con una muestra simbólica de las imágenes obtenidas usando el algoritmo mostrado anteriormente, posteriormente se mostrarán las gráficas del entrenamiento con su correspondiente análisis, de igual forma se darán dos ejemplos para la validación del entrenamiento logrado con la red neuronal convolucional, para cada grado de libertad fue usada la misma red neuronal convolucional por lo que se observarán los resultados obtenidos antes y después de la identificación paramétrica.

7.1 Resultados de la simulación del modelo dinámico

Una vez que se obtuvo el modelo dinámico se procedió a realizar simulaciones con el fin de validarlo, en las siguientes Figuras 7.1 y 7.2 se aplicó un torque constante en ambos motores, por lo que las gráficas resultantes muestran una recta, lo cual confirma lo esperado ya que al aplicar un torque constante el motor girará con una velocidad constante, por lo que la posición irá aumentando.

Los valores usados para realizar esta simulación son los siguientes, se usaron como referencia de [18], donde se usó la técnica de parámetros agrupados para el modelado dinámico, dado que se usó el mismo tipo de motor se tomaron valores aproximados ya que cada motor tiene parámetros distintos sin importar que sea el mismo modelo y fabricante.

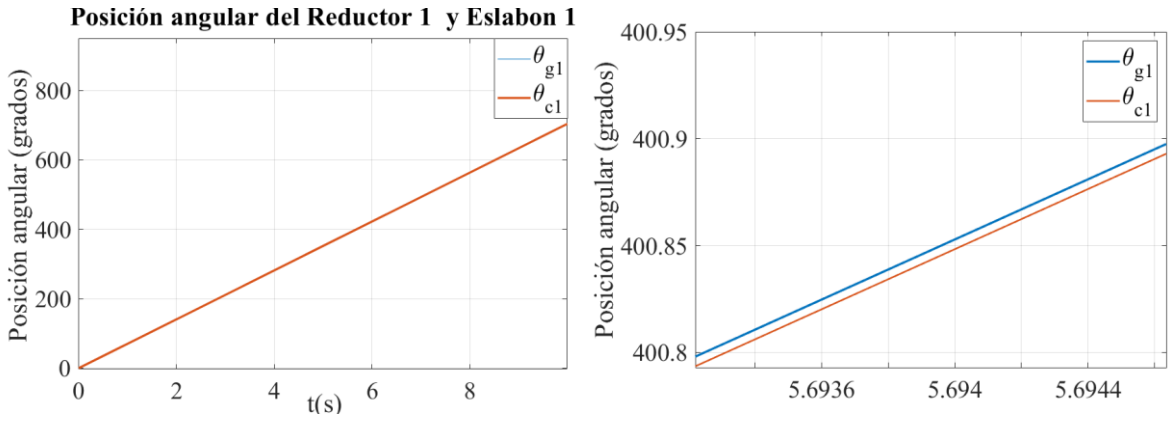


Figura 7.1: Posiciones angulares θ_{g1} , θ_{c1}.

Como se puede observar en la Figura 7.1 anterior, las posiciones angulares del reductor y del eslabón parecen ser una sola línea, sin embargo, al realizar un acercamiento en la Figura 7.1 de la derecha se observa que existe un pequeño desplazamiento entre ellas, al observar esto se confirma que la respuesta del modelo dinámico es correcta ya que ambas posiciones de deben de seguir la misma trayectoria.

En la siguiente Figura 7.2 se observa la respuesta de la posición de θ_{g2} y θ_{c2} los cuales corresponden a la posición angular del reductor y eslabón del segundo grado de libertad.

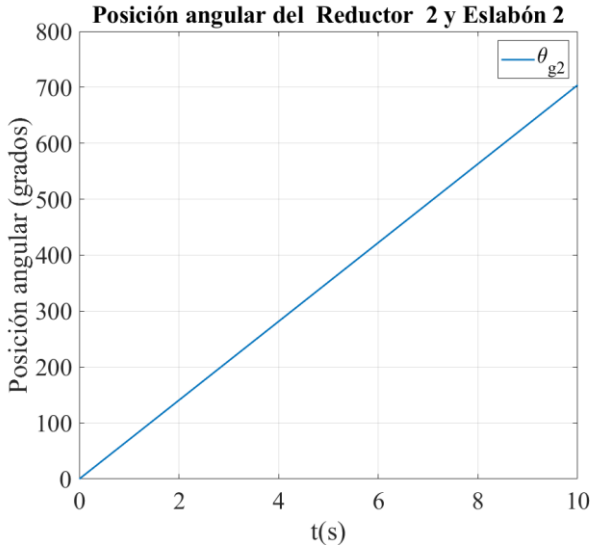


Figura 7.2: Posición angular θ_{g2}, θ_{c2}.

Una vez que se aplicó un torque constante se graficó la respuesta de la trayectoria obtenida, como se puede observar en la línea azul se muestra la posición del extremo final del primer eslabón, el cual muestra una trayectoria circular, lo cual confirma lo esperado, y

en el caso del segundo eslabón se forma una trayectoria diferente lo cual se debe a que este está girando al mismo tiempo que el primer eslabón, por lo que no describe una trayectoria completamente circular.

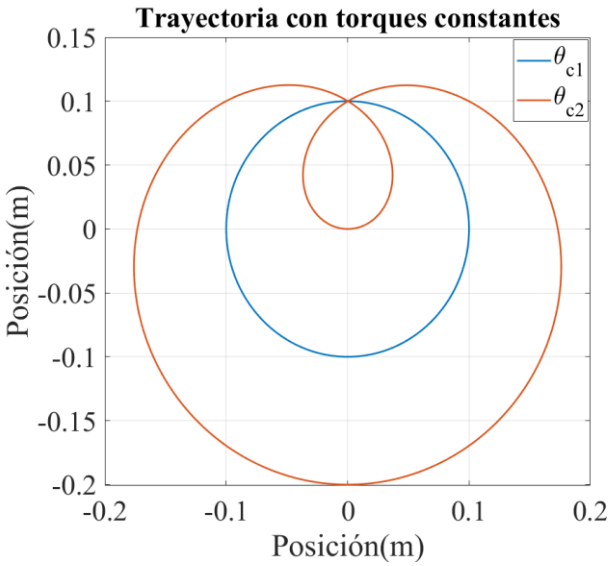


Figura 7.3: Trayectoria realizada al aplicar torques constantes.

Posteriormente se aplicó un control de posición con el controlador $\tau_n = k_{p_n}\varepsilon_n$ donde $\varepsilon_n = q_{dn} - q_n$, siendo q_{dn} la posición deseada y q_n la posición actual considerando $n = 1{,}2$ para cada eslabón, en la siguiente Figura 7.4 puede observarse la respuesta, para $\tau_1 = 2.73 * (\frac{\pi}{2} - \theta_{c1})$ y $\tau_2 = 2 * (\frac{\pi}{2} - \theta_{c2})$, donde $\frac{\pi}{2}$ es la posición deseada para ambos grados de libertad.

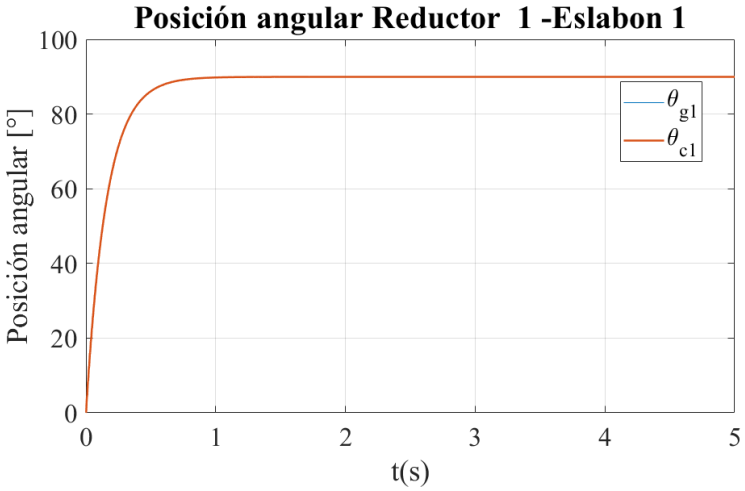


Figura 7.4: Control de posición con $\theta_{c1} = 90°$

Se calculó también el error de posición respecto al control de posición para 90°, en la Figura 7.5 se muestra como el error tiende a cero.

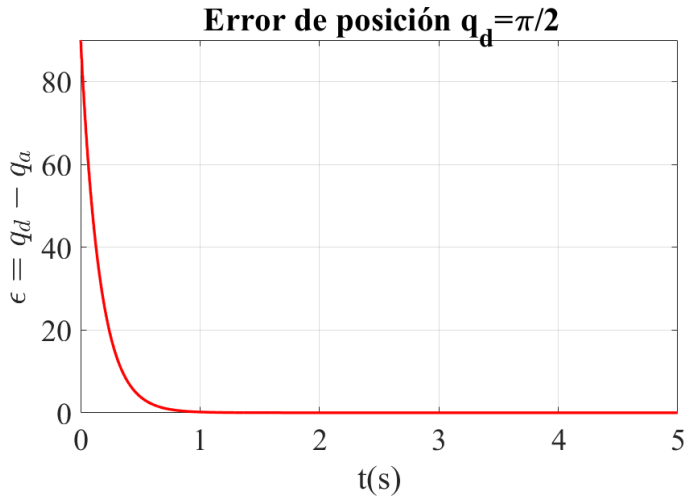


Figura 7.5: Error de posición con $\theta_{c1} = 90°$ 1 gdl.

La respuesta obtenida para la posición de 90° para el segundo grado de libertad se muestra en la Figura 7.6.

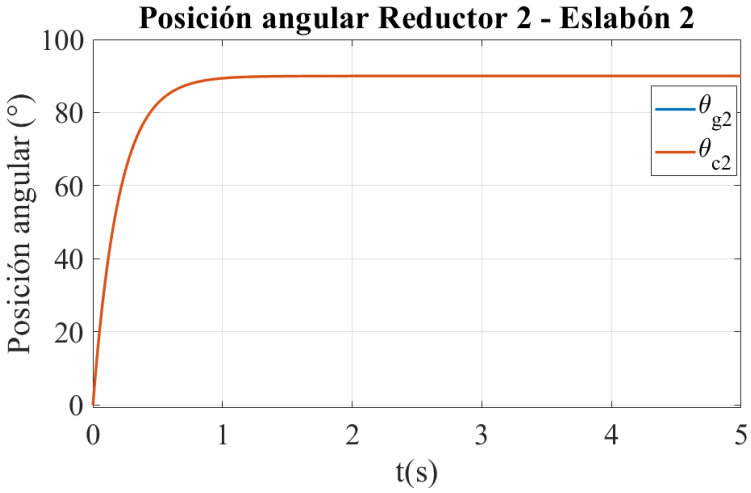


Figura 7.6: Control de posición con $\theta_{c2} = 90°$.

En la Figura 7.7 se puede observar el error de posición obtenido para el segundo grado de libertad con la posición deseada de 90°.

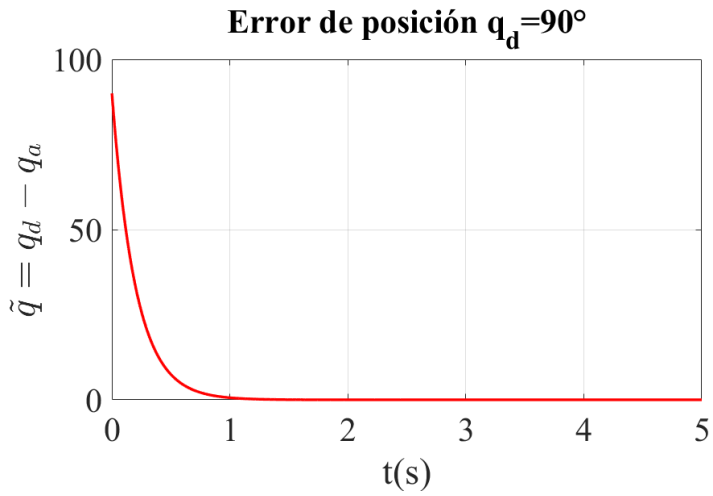


Figura 7.7: Error de posición para la posición deseada de 90°.

Los torques obtenidos para cada grado de libertad se pueden observar en la Figura 7.8

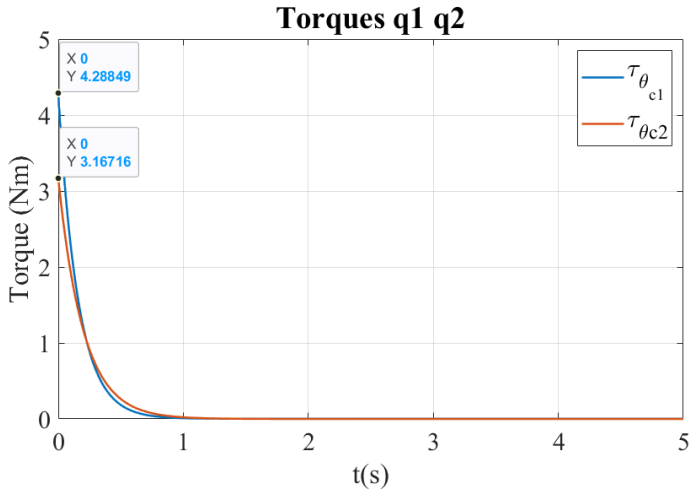


Figura 7.8: Torques aplicados.

Como se puede observar en la Figura 7.8, los torques alcanzan valores de 4.2 Nm y 3.1 Nm correspondientes al primer y segundo motor, sin embargo, estos valores exceden los torques máximos que puede proporcionar cada uno de los motores, por lo que es necesario ajustar las ganancias del controlador k_p, aquí es donde toma importancia la identificación paramétrica ya que al obtener los parámetros del sistema es posible analizar su comportamiento en simulación.

Respuesta a un torque constante

Para verificar el modelo dinámico se realizaron dos pruebas, en ambas se aplicó un torque constante a cada uno de los motores, para hacer esto, en el caso del primer eslabón, se

sujetó el segundo eslabón, esto para evitar que el movimiento del segundo eslabón afecte al primero. En la segunda prueba solo se consideró la posición del segundo eslabón, por lo que no fue necesario realizar alguna sujeción extra, en la siguiente Figura 7.9 se puede observar la posición del robot aplicando un 100% del PWM aplicado al primer grado de libertad. En ambas pruebas se consideró que ambos eslabones giran libremente, por lo que se hicieron los cambios necesarios para que las conexiones del segundo grado de libertad no interfieran en el giro de la primera.

En la siguiente Figura 7.9 se observa el cambio de la posición en el tiempo, aplicando un PWM del 100% en el primer grado de libertad, en la imagen se puede ver cómo va aumentando la posición con el tiempo.

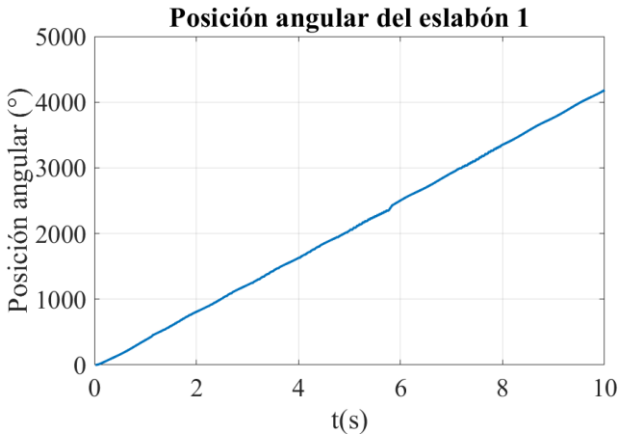


Figura 7.9: Respuesta del 1 gdl a un torque constante.

En el segundo eslabón se aplicó de igual forma un torque constante, para esto se sujetó el primer eslabón para evitar que interfiera en el segundo grado de libertad, en la Figura 7.10 se muestra su respuesta

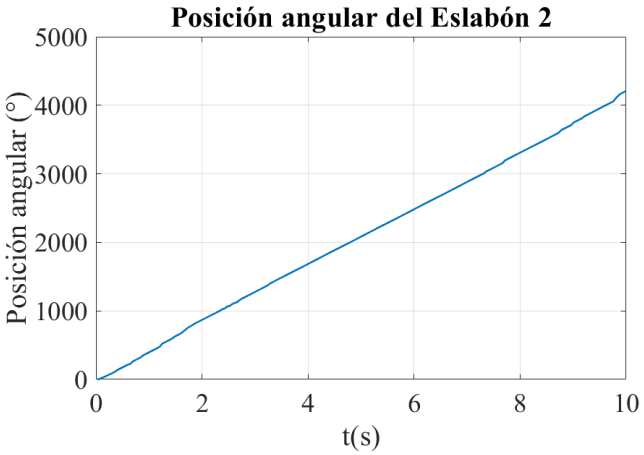


Figura 7.10: Respuesta del 2 gdl a un torque constante.

Trayectoria Senoidal

Primer grado de libertad

Los resultados obtenidos para una trayectoria senoidal con amplitud de 90 °, un periodo de 20 s y usando el mismo esquema de control planteado por la ecuación (19) se muestran a continuación. En la Figura 7.11, se pueden observar los resultados de posición angular para el primer eslabón, y en la Figura 7.12 se observa la velocidad obtenida, estos resultados fueron conseguidos antes de la identificación paramétrica,

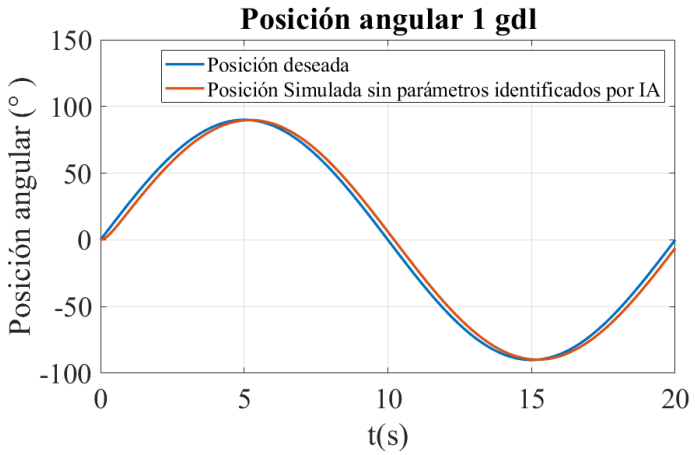


Figura 7.11: Posición para una trayectoria senoidal.

La similitud obtenida para entre la posición simulada y la deseada es de 98.35%, en la Figura 7.12, se puede observar que, si bien ambas gráficas muestran la misma trayectoria, la posición simulada se encuentra desfasada respecto a la deseada.

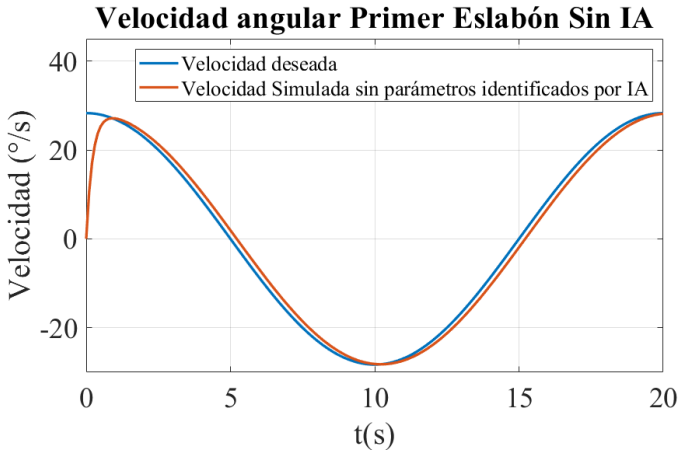


Figura 7.12: Velocidad obtenida en simulación sin parámetros identificados por IA

Se puede observar que la velocidad obtenida en la simulación está desfasada en la simulación, lo cual no corresponde exactamente a la velocidad angular deseada, mostrando una similitud de 83.18% entre la velocidad deseada y la simulada.

En la Figura 7.13 se pueden ver los torques aplicados en el primer eslabón aplicando un control PD con tangente hiperbólica,

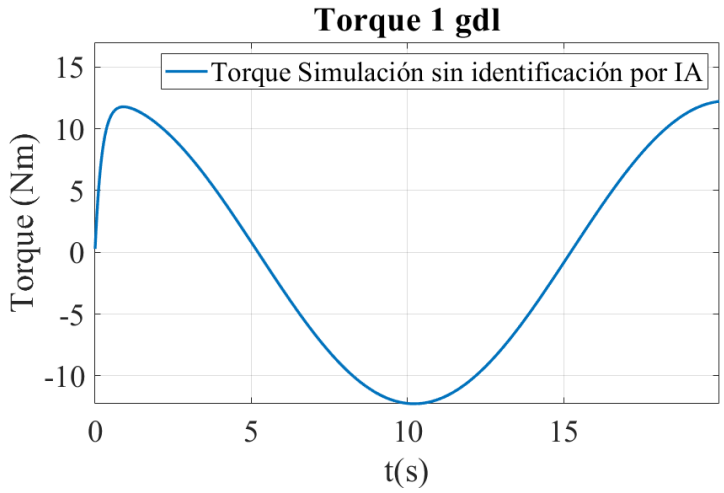


Figura 7.13: Torque aplicado al primer eslabón.

Como se puede observar en la Figura 7.13, se puede notar que el torque máximo aplicado al robot es superior a 10 N, lo cual excede los torques nominales del primer motor, más adelante con los parámetros identificados se observará que el torque obtenido usando los parámetros encontrados no exceden los límites físicos del motor correspondiente al primer eslabón.

Segundo grado de libertad

En la Figura 7.14 se puede observar la posición obtenida en la simulación, de igual forma se usó la trayectoria senoidal con 90° de amplitud y un periodo de 20 segundos.

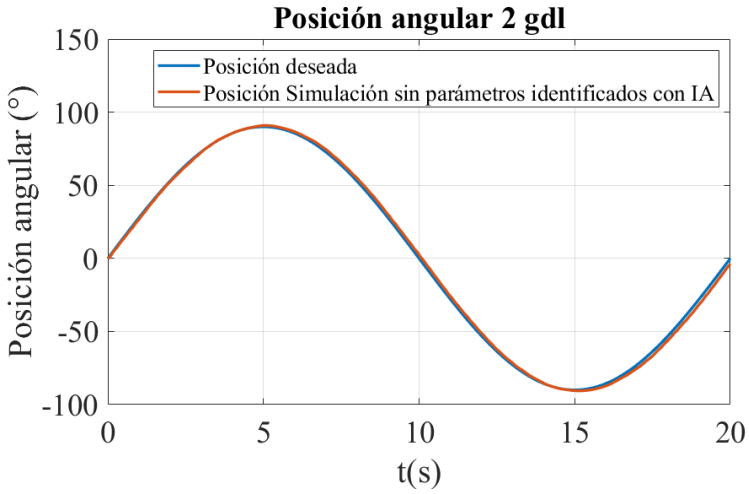


Figura 7.14: Posición angular del 2 gdl.

En la Figura 7.15 se observa la posición obtenida para el segundo grado de libertad, aplicando la métrica planteada se encontró una similitud de 99.28%.

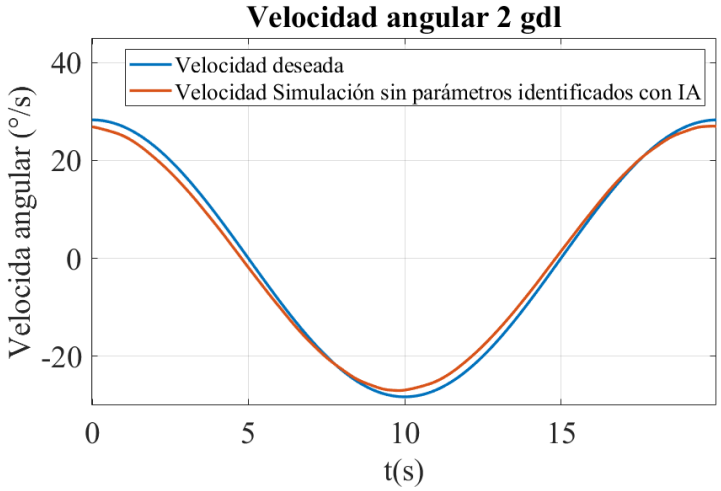


Figura 7.15. Velocidad del 2 gdl.

Se puede observar en la Figura 7.15 la velocidad obtenida en la simulación y la deseada, se puede notar que existe un pequeño desfase en la velocidad simulada, y pequeño desplazamiento en la amplitud deseada, usando la métrica usada en este trabajo se encontró una similitud de 97.51%.

Finalmente se obtuvo el torque aplicado para seguir la trayectoria planteada, en la Figura 7.16 se muestra el torque obtenido en la simulación

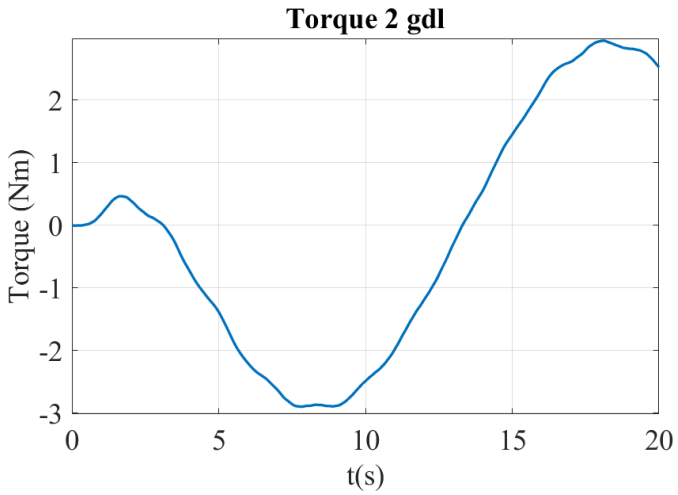


Figura 7.16: Torque aplicado al 2 gdl.

Se puede observar que la solución dada para el segundo grado de libertad empieza con un torque positivo para luego entrar al ciclo positivo de torque, esto es algo a analizar de forma experimental, ya que se espera que, al ir en dirección positiva, el torque sea también positivo, este efecto se debe a los parámetros usados en la simulación, por lo que finalmente se realizará la identificación paramétrica en el siguiente apartado.

7.2 Resultados: Identificación paramétrica

En este capítulo se describen los resultados obtenidos de la identificación paramétrica usando la red neuronal propuesta, para esto se planteó una trayectoria senoidal con el fin de obtener las señales de torque, y usar las señales correspondientes para generar dos imágenes sobres las cuales se realizará la identificación paramétrica. Se presentan también las gráficas obtenidas del entrenamiento y validación de la red neuronal convolucional. Finalmente se mostrarán los resultados obtenidos de la identificación paramétrica mostrando las gráficas de torque obtenidas de forma experimental y la simulación obtenida usando los parámetros encontrados por la red neuronal convolucional.

7.2.1 Repositorio de imágenes

En este trabajo se usaron dos repositorios de imágenes, cada uno con 500,000 imágenes correspondientes a cada grado de libertad, se simuló la misma trayectoria senoidal con valores pseudoaleatorios para los parámetros que se desean encontrar.

Para facilitar la obtención de imágenes algunos parámetros se pueden determinar usando SolidWorks, los parámetros conocidos se tienen en la siguiente Tabla 7.1.

Tabla 7.1: Parámetros conocidos del brazo robot

Parámetro	Descripción	Valor	Unidades
J_{c1}	Inercia del 1 gdl	0.00116865393	$Kg * m^2/rad$
J_{c2}	Inercia del 2 gdl	0.00023822675	$Kg * m^2/rad$
m_1	Masa del primer eslabón	.387	Kg
m_1	Masa del segundo eslabón	.128	Kg
l_1	Longitud del primer eslabón	0.10	m
l_{c1}	Distancia al centro de masa del primer eslabón	0.05726	m
l_2	Longitud del segundo eslabón	.11	m
l_{c2}	Distancia al centro de masa del segundo eslabón	0.03973	m
G	Relación de reducción	131.25	
g	Gravedad	9.81	m/s^2

Primer grado de libertad

Para generar las imágenes fue necesario usar super cómputo, para esto se solicitó acceso al LNS de la BUAP, en la Tabla 7.2 se muestra el tiempo aproximado que sería necesario para generar cada imagen del repositorio de imágenes de entrenamiento usando una computadora personal.

Tabla 7.2: Tiempos de ejecución

Equipo	Cantidad de imágenes	Tiempo aproximado por imagen	Tiempo Total
Personal	500,000	15 min	≈14.26 años

En la Figura 7.17 y la Figura 7.18 se pueden observar algunos ejemplos de las imágenes obtenidas que forman parte del repositorio de entrenamiento, en el caso de la Figura 7.17, se muestran los parámetros usados para construir la imagen. Para el repositorio de imágenes del primer grado se generaron 500,000 imágenes de entrenamiento.

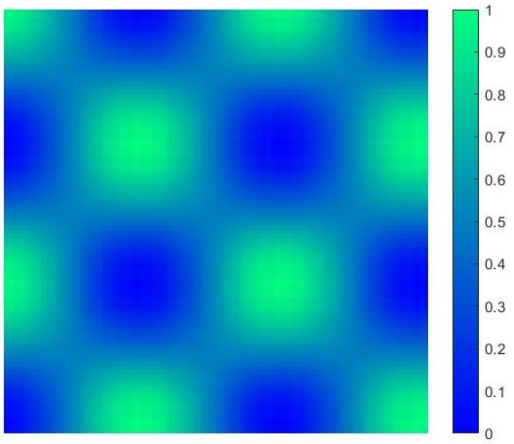
Imagen tipo A_n	
Parámetros dinámicos	
$J_m = 0.0015\ Kg * m^2$ $B_m = 0.0035\ Nms/rad$ $B_g = 4.3910x10^{-4} Nms/rad$ $K_g = 0.0423\ Nm/rad$ $J_g = 0..0015\ Kg * m^2$ $B_c = 0.0048\ Nms/rad$ $K_c = 0.3679\ Nm/rad$	
	Índice: 1000 de 500,000;

Figura 7.17: Ejemplo de imagen tipo A_n.

En la Figura 7.18 se muestran dos ejemplos de imágenes tipo A_p, en estos los parámetros usados para construir la imagen son diferentes, por lo que no se muestran los valores de los parámetros aleatorios.

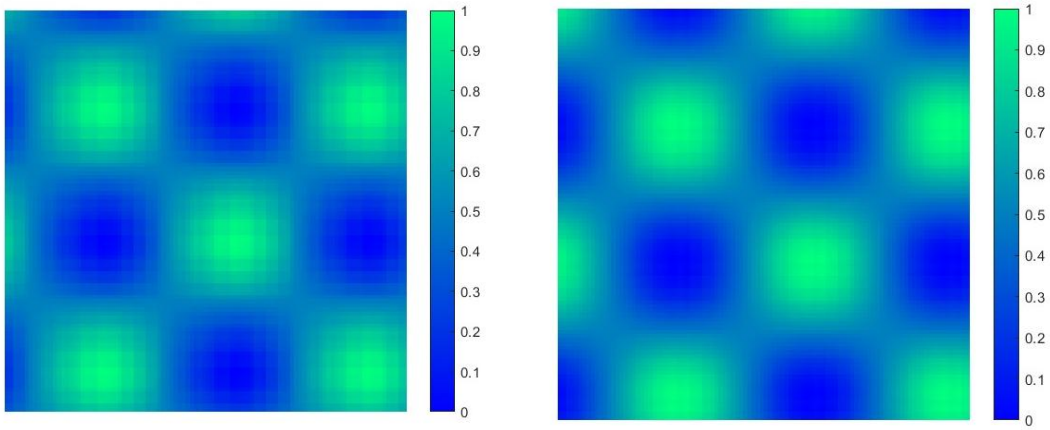
Imagen tipo A_p

Índice: 10000, 250000 de 500,000;

Figura 7.18: Ejemplo de imagen tipo A_p.

Segundo grado de libertad

En la Figura 7.19 y la Figura 7.20 se pueden observar los dos tipos de imágenes de entrenamiento, imágenes de tipo A_n, donde el residuo paramétrico es igual a cero e imágenes de tipo A_p, donde el residuo paramétrico es diferente de cero, esto es importante ya que es necesario estos dos tipos de imágenes para realizar el entrenamiento. Se obtuvieron 500,000 imágenes de cada tipo.

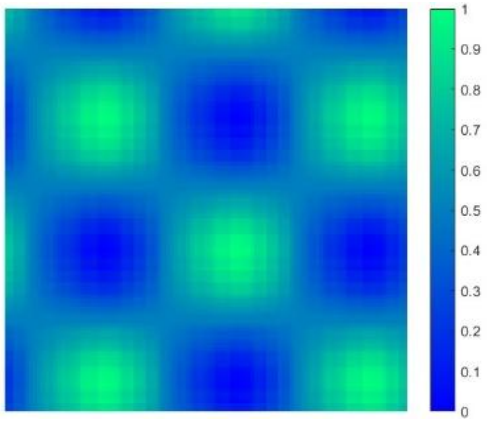
Imagen tipo A_n	
Parámetros dinámicos	
$J_m = 1.4229x10^{-4}\, Kg * m^2$ $B_m = 2.3245x10^{-5} Nms/rad$ $B_g = 1.066x10^{-4} Nms/rad$ $K_g = 0.0054\, Nm/rad$ $J_g = 1.3581x10^{-4} Kg * m^2$ $B_c = 1.5851x10^{-4} Nms/rad$ $K_c = 0.0022\, Nm/rad$	
	Índice: 1000 de 500,000;

Figura 7.19: Ejemplo de imagen tipo A_n.

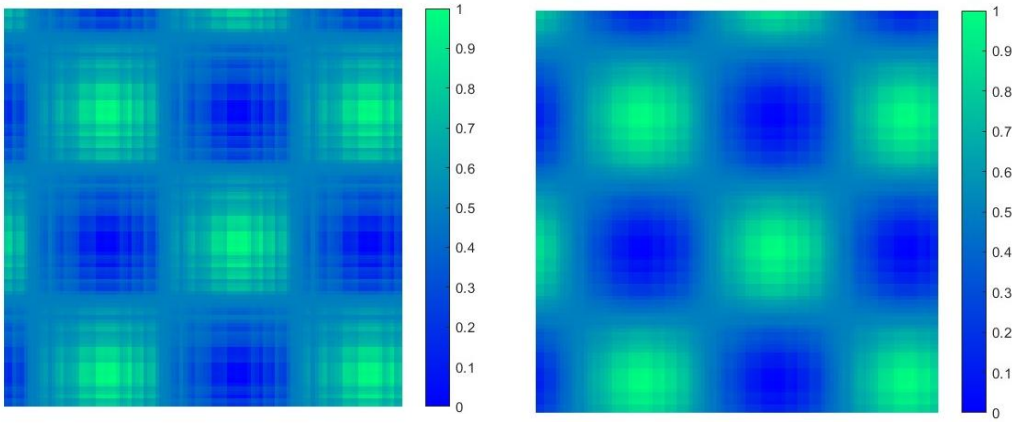
Imagen tipo A_p

Índice: 7500,10000 de 500,000;

Figura 7.20: Ejemplo de imagen tipo A_p.

7.2.3 Resultados del entrenamiento

Entrenamiento Primer grado

En la Figura 7.21 se puede observar el entrenamiento de la red neuronal correspondiente al primer grado de libertad, como se puede ver la tendencia va disminuyendo conforme aumenta el número de iteraciones, lo cual garantiza que la red está siendo entrenada correctamente y no solo está memorizando, siendo la memorización un efecto a evitar ya que al proporcionarle una imagen diferente al conjunto de imágenes del repositorio la red neuronal no es capaz de identificarla.

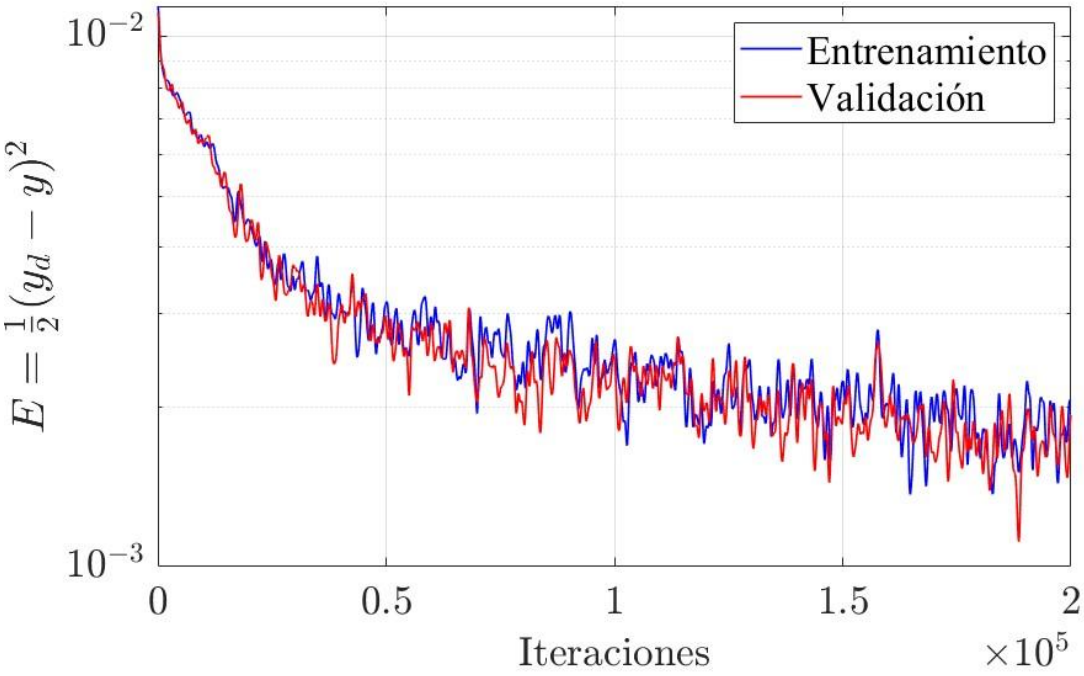


Figura 7.21: Curva de entrenamiento para la red neuronal correspondiente al primer gdl.

Para realizar el entrenamiento de la red neuronal convolucional se usó la ecuación

$$E = \frac{1}{2}\sum_{j=1}^{N}\left(y_{jd} - y_j\right)^2 \tag{57}$$

Donde y es el vector de salidas de la red neuronal convolucional, y_d es el vector de salidas deseadas, j indica la dirección de las salidas y_d y N es el número de salidas de la red.

Para realizar la modificación de los pesos sinápticos W y los valores de bias (desviaciones) se usa el gradiente para la función de error usando la regla delta mencionada anteriormente.

En la Figura 7.21 se mostró el entrenamiento de la red neuronal convolucional para el primer grado de libertad, se puede observar que ambas curvas, la de entrenamiento como la de validación van disminuyendo conforme va aumentando el número de iteraciones, observando esto se puede decir que la red neuronal convolucional está aprendiendo de forma correcta y no está memorizando.

En la Figura 7.22 se pueden observar los pesos sinápticos de una red neuronal, estos juegan un papel muy importante para que la red neuronal convolucional realice la tarea de identificar los parámetros a identificar con la red neuronal convolucional, estos pueden verse como la analogía al funcionamiento del cerebro humano, esta similitud puede verse como el cerebro humano procesa la información, seleccionando características de manera jerárquica donde primer se aprenden ciertas características como las encontradas en las primeras capas de convolución, y así sucesivamente en la siguientes capas donde se van identificando diferentes características al ir variando el tamaño de los filtros.

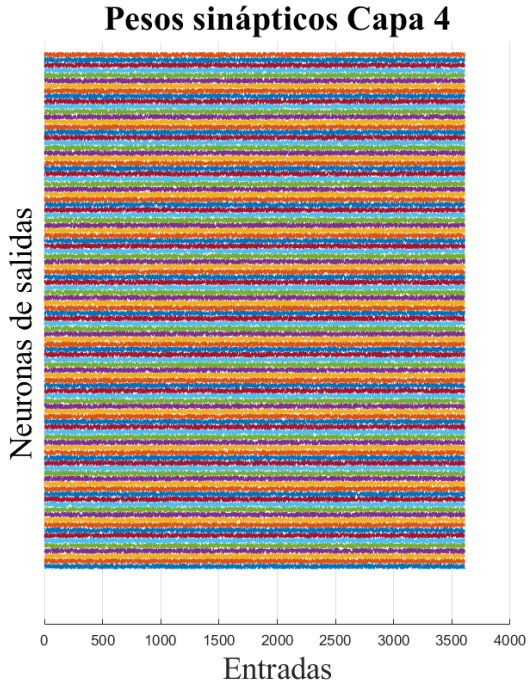


Figura 7.22: Pesos sinápticos para la Capa 4.

En la Figura 7.23 se pueden observar los pesos sinápticos para la Capa 5, estas salidas son a su vez entradas para la siguiente capa (Figura 7.24) donde se muestran los pesos sinápticos para la capa de salida correspondiente a los parámetros identificados

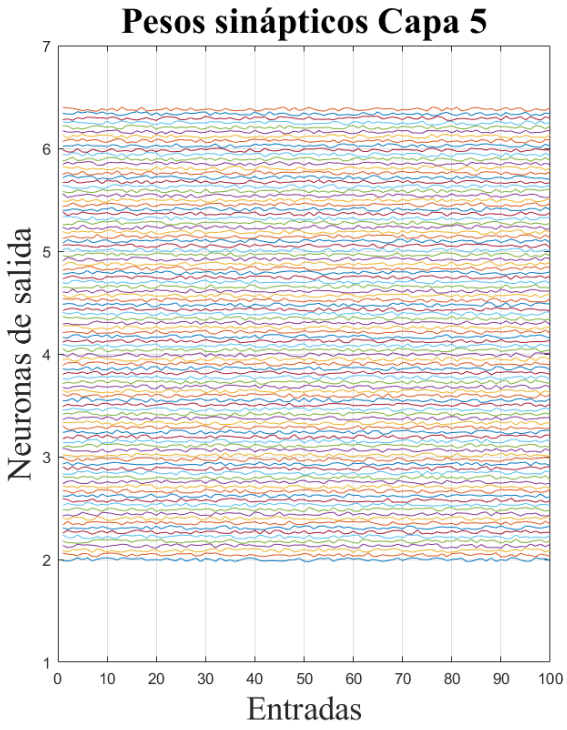


Figura 7.23: Pesos sinápticos para la Capa 5.

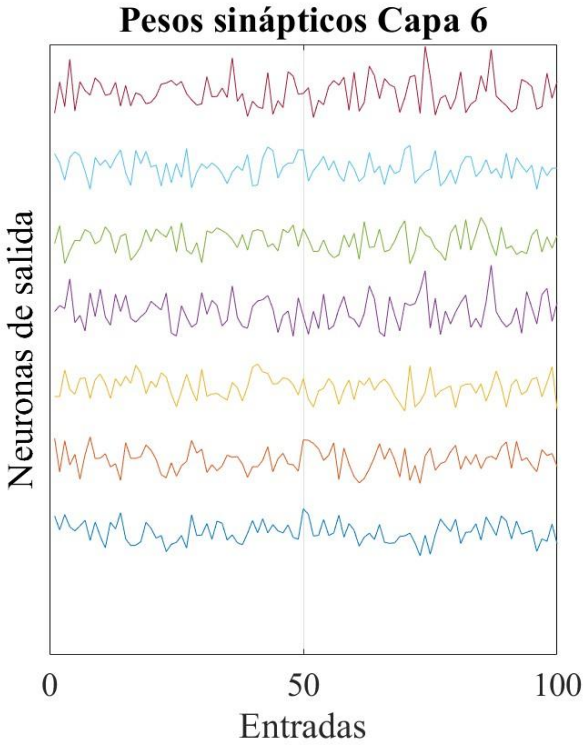


Figura 7.24: Pesos sinápticos Capa 6.

Entrenamiento segundo grado

En la Figura 7.25 se observa el entrenamiento para el segundo grado de libertad, como se puede observar ambas curvas van disminuyendo progresivamente, esto significa que la red neuronal convolucional está aprendiendo y generalizando bien los datos. En ambos casos, para el primer y segundo grado se hizo el cálculo para 200,000 iteraciones,

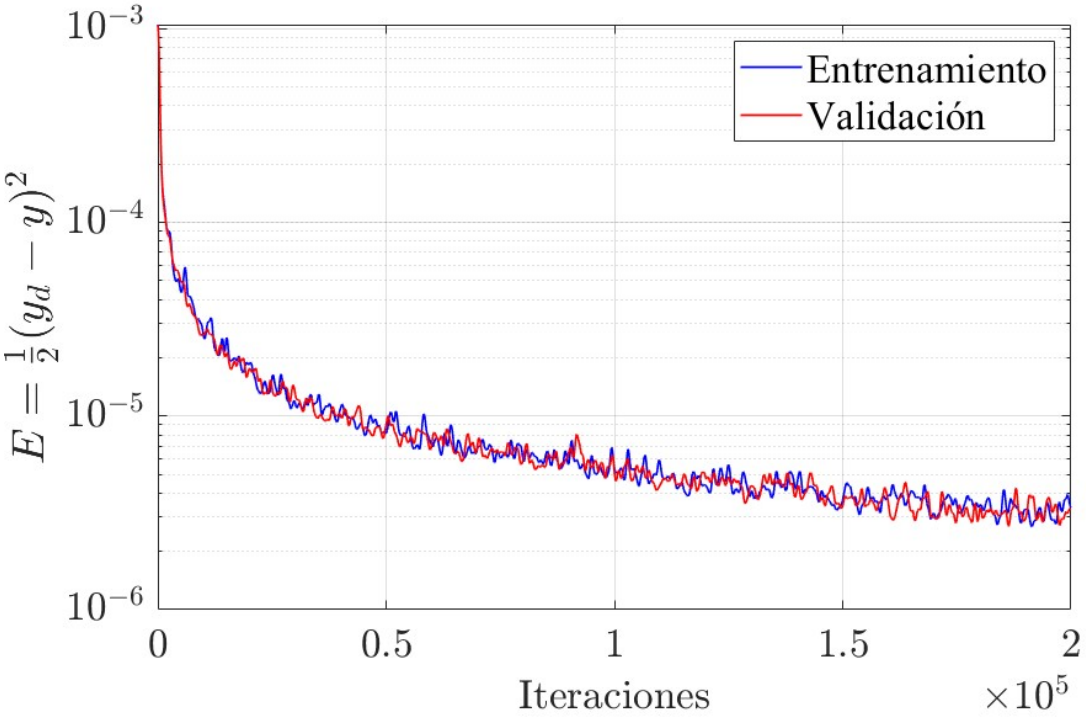


Figura 7.25: Curva de entrenamiento de la red neuronal convolucional para el 2 gdl.

Se puede observar que entre la Figura 7.21 y la Figura 7.25 donde se ve el entrenamiento para ambos grados de libertad, en el primero existe una mayor oscilación en el entrenamiento, lo que indica que puede ser posible probar una tasa de aprendizaje menor para estabilizar el entrenamiento, sin embargo, por la forma de la gráfica se puede decir que si está entrenando de forma correcta.

En la Figura 7.26 pueden verse los pesos sinápticos para la Capa 4 correspondiente al segundo grado de libertad, en la Figura 7.27 los pesos sinápticos de la Capa 5 y finalmente en la Figura 7.28 los pesos sinápticos de la Capa 6.

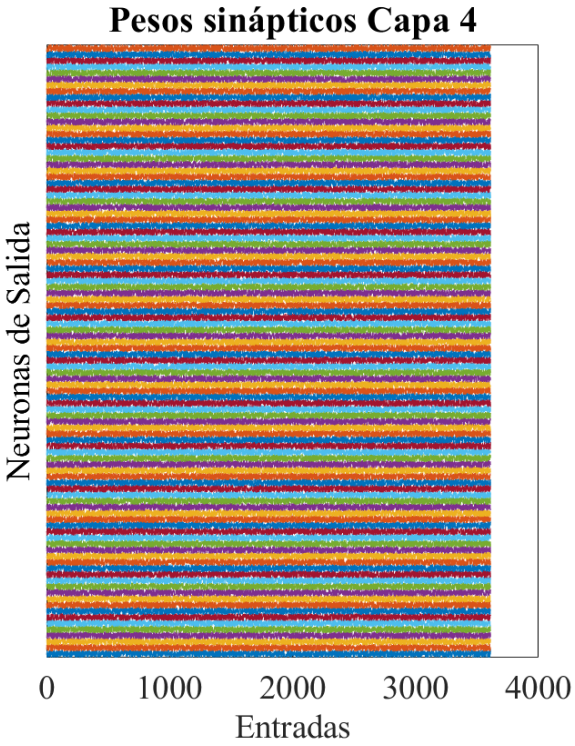


Figura 7.26: Pesos sinápticos Capa 4.

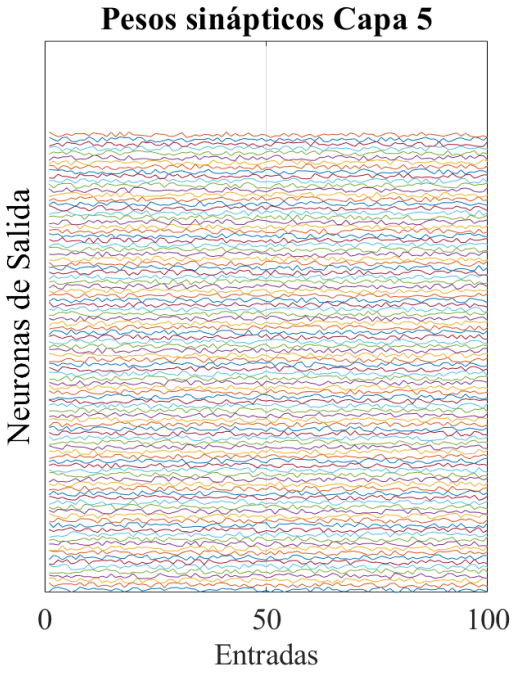


Figura 7.27: Pesos sinápticos Capa 5.

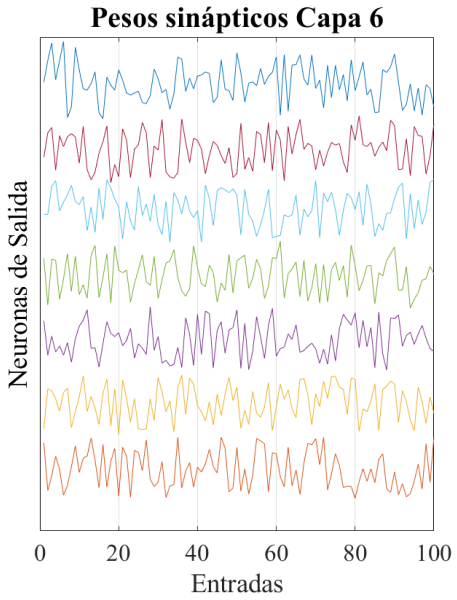


Figura 7.28: Pesos sinápticos para la Capa.

7.4 Resultados de la identificación paramétrica

Para verificar que la red ha sido entrenada correctamente se puede comprobar usando una imagen generada con valores aleatorios diferente a los usados en el conjunto de entrenamiento, en la Figura 7.29 se muestra la identificación obtenida para el primer grado de libertad con una coincidencia de 99.29 %, una vez que la red neuronal convolucional ha sido entrenada correctamente para el primer grado de libertada, el siguiente paso es evaluar el torque experimental para encontrar los parámetros deseados del primer grado de libertad.

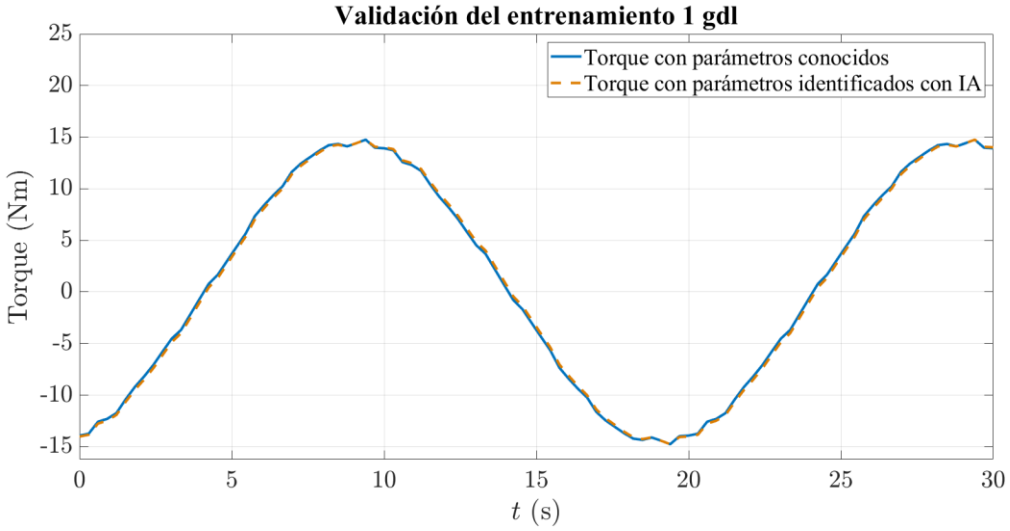


Figura 7.29: Resultados de la validación del 1 gdl.

En la Tabla 7.3 se muestran los valores aleatorios usados para la señal de torque de la Figura 7.29, y los valores identificados por la red neuronal convolucional propuesta.

Tabla 7.3: Valores aleatorios propuestos y valores identificados por la RNC

Parámetro	Valor	CNN1	Unidades
J_m	1.3474x10-04	1.3692x10-04	$Kg \cdot m^2/rad$
B_m	1.9103x10-04	1.9412x10-04	$N \cdot m \cdot s/rad$
B_g	7.1567x10-07	7.2726x10-07	$N \cdot m \cdot s/rad$
K_g	0.0152	0.0155	$N \cdot m/rad$
J_g	7.3113x10-04	7.4297x10-04	$Kg \cdot m^2$
B_c	0.0019	0.0019	$N \cdot m \cdot s/rad$
K_c	0.0059	0.0059	$N \cdot m/rad$

En la Figura 7.30 se evaluó la red neuronal convolucional para el segundo grado de libertad, se usó una imagen usando parámetros aleatorios para el segundo grado de libertad diferentes a los del conjunto de entrenamiento, se encontró una similitud de 99.32%, por lo que podemos concluir que al existir una similitud cercana al 100 %, la red neuronal convolucional ha sido entrenada exitosamente para el segundo grado de libertad, por lo que el siguiente paso es evaluar la red con los torques experimentales obtenidos.

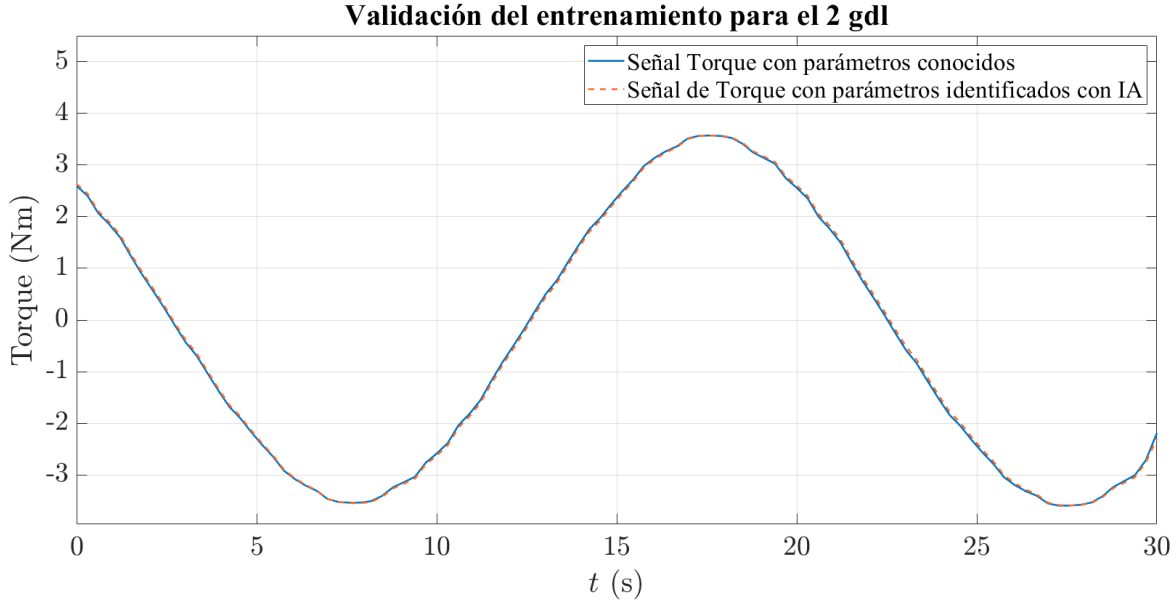


Figura 7.30: Resultados de la simulación de la identificación paramétrica del segundo grado.

En la Tabla 7.4 se muestran los valores propuestos del torque de la Figura 7.30 y los valores hallados al realizar la identificación

Tabla 7.4: Valores aleatorios propuestos y valores identificados por la RNC

Parámetro	Valor	CNN2	Unidades
J_m	2.6792x10-05	2.9608x10-05	$Kg \cdot m^2/rad$
B_m	4.8211x10-05	5.3289x10-05	$N \cdot m \cdot s/rad$
B_g	1.5414x10-05	1.7035x10-05	$N \cdot m \cdot s/rad$
K_g	0.0199	0.0220	$N \cdot m/rad$
J_g	0.0019	0.0021	$Kg \cdot m^2$
B_c	0.0018	0.0020	$N \cdot m \cdot s/rad$
K_c	0.0048	0.0053	$N \cdot m/rad$

Los valores obtenidos, para el primer grado se obtuvo una similitud de 99.29% y para el segundo una similitud de 99.32%, analizando los resultados obtenidos se puede notar que el algoritmo de identificación paramétrica, la red propuesta y el repositorio de imágenes es válido para realizar la identificación paramétrica de forma experimental, ya que se puede notar la similitud en la forma de ambos torques, como en los valores de los parámetros hallados. Los valores identificados de la Figura 7.29 y la Figura 7.30 mostrados en la Tabla 7.3 y 7.4, denotan la similitud entre los valores propuestos de la simulación con los valores obtenidos al usar la red neuronal propuesta, más adelante se mostrarán los resultados obtenidos al realizar la identificación paramétrica usando los torques experimentales obtenidos al aplicar una trayectoria senoidal al robot.

Resultados experimentales de torque

Para obtener los resultados experimentales de torque, se aplicó una señal senoidal para ambos grados de libertad con una amplitud de 90° y un período de 20 segundos, se obtuvieron los datos de torque de un ciclo y medio con un tiempo total de 30 segundos. Se hizo uso del lenguaje D y de la interfaz mostrada anteriormente para que el robot realizara la trayectoria senoidal propuesta usando un control de trayectoria usando en el diagrama de la Figura 2.4, de esta forma se obtuvieron los datos necesarios, también fue necesario realizar un procesamiento en ambas señales para obtener un vector de torque de 100 muestras.

Primer grado de libertad

Se usó el vector de torques obtenido experimentalmente para el primer grado de libertad con el fin de generar la imagen a evaluar dentro de la red neuronal, los parámetros obtenidos de la identificación paramétrica se muestran en la Tabla 7.5.

Tabla 7.5: Parámetros obtenidos de la identificación paramétrica del 1 GDL

Parámetro	Valor	Unidades
$\boldsymbol{J_{m1}}$	0.0049	$Kg \cdot m^2/rad$
$\boldsymbol{B_{m1}}$	1.0526x(10^-4)	$N \cdot m \cdot s/rad$
$\boldsymbol{B_{g1}}$	0.0018	$N \cdot m \cdot s/rad$
$\boldsymbol{K_{g1}}$	0.2271	$N \cdot m/rad$
$\boldsymbol{J_{g1}}$	9.8071x(10^-4)	$Kg \cdot m^2$
$\boldsymbol{B_{c1}}$	0.0018	$N \cdot m \cdot s/rad$
$\boldsymbol{K_{c1}}$	0.1596	$N \cdot m/rad$

Con los valores obtenidos de la identificación paramétrica para el primer grado de libertar, se realizó la simulación del control de trayectoria con estos parámetros, en la Figura 7.31, se puede observar la posición obtenida en la simulación y de forma experimental, se encontró una similitud de 99.28 % de coincidencia entre la posición real y la obtenida por el control de trayectoria.

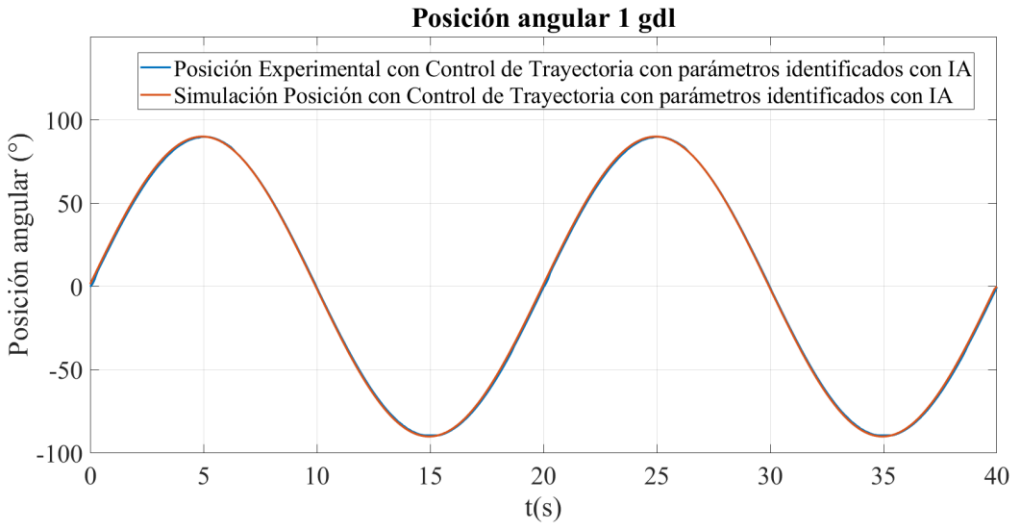


Figura 7.31: Posición angular 1 gdl.

En la Figura 7.32 se puede observar el error de posición correspondiente al primer grado de libertad, como se puede notar, existe un error entre la posición deseada y la posición experimental, se puede observar también que el error siempre tiende a cero,

Figura 7.32. Error de posición 1 gdl.

En la gráfica de la Figura 7.33 se muestra la velocidad obtenida usando el control de trayectoria con los parámetros obtenidos con IA y la velocidad obtenida de forma experimental, se obtuvo una similitud de 94.50% entre estas.

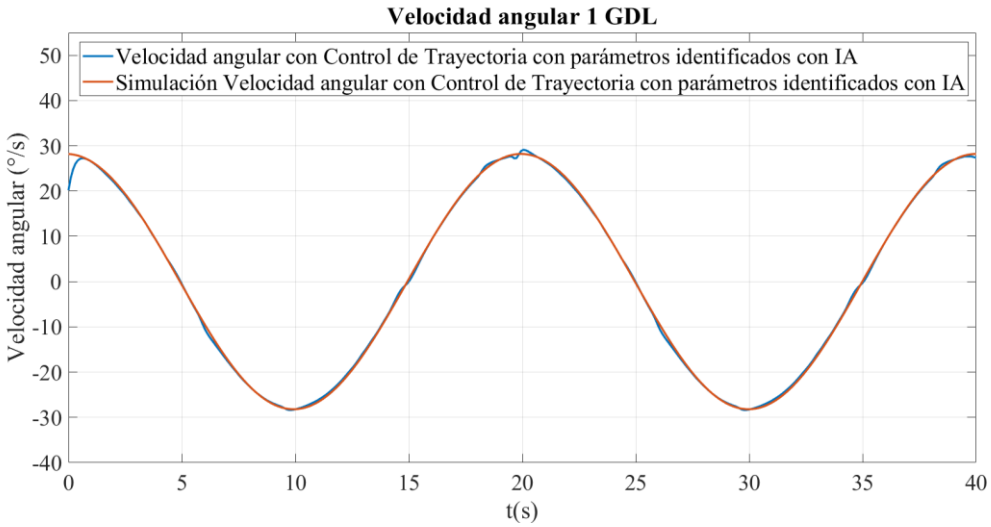


Figura 7.33: Velocidad obtenida con Control de trayectoria Simulada con los parámetros obtenidos con IA y la experimental.

Con los valores obtenidos de la identificación paramétrica se procedió a realizar la comparación de los torques obtenidos en la simulación y de forma experimental, en la Figura 7.34 se observa el torque experimental y el obtenido de la simulación con los parámetros identificados por la red neuronal, se obtuvo una similitud de 80.58% con los parámetros obtenidos de la red neuronal convolucional.

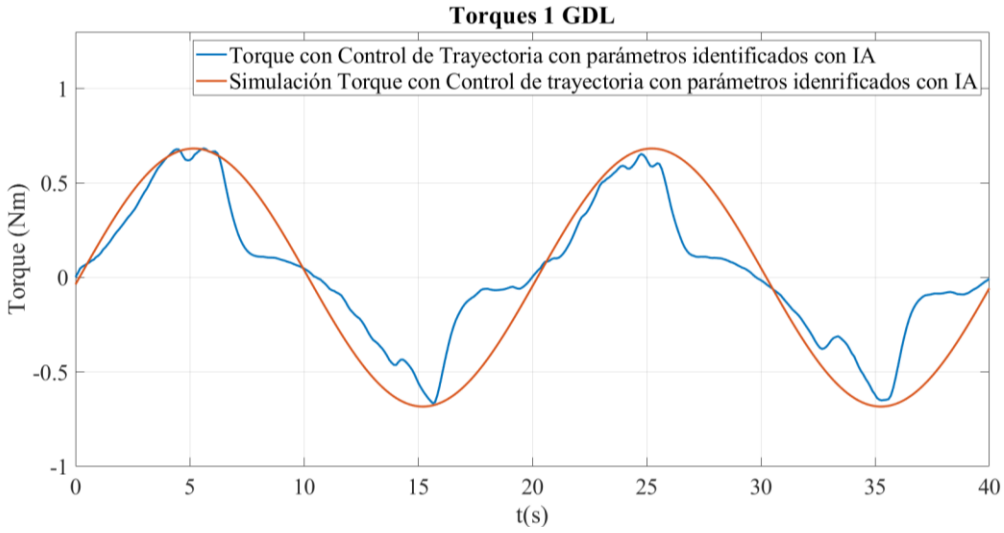


Figura 7.34: Identificación paramétrica del primer grado de libertad.

Analizando la Figura 7.34 se puede notar que en ambos casos, para la parte positiva de la señal y la negativa al se ve un efecto que no se logra aproximar con los parámetros obtenidos, esto se debe a otros efectos que no fueron modelados en el modelo propuesto, en este caso se pueden notar que pueden ser efectos de fricción, los cuales ocurren al elevar el eslabón del robot hasta los 90°, y se puede notar que en el torque disminuye, esto se debe también a la fricción y el peso del eslabón compuesto del motor y del segundo eslabón. Se observó que este efecto es más notorio en el primer grado de libertad en comparación con el segundo, ya que este, al tener menos masa, tiene diferentes propiedades.

Segundo grado de libertad

En el segundo grado de libertad se usó la misma trayectoria senoidal de 90° de amplitud y un período de 20 segundos, de igual forma se obtuvo un vector con la información de un ciclo y medio de la señal propuesta, los valores obtenidos al realizar la identificación paramétrica se muestran en la Tabla 7.6, en este caso se obtuvo una similitud de 91.54%.

Tabla 7.6: Parámetros obtenidos de la identificación paramétrica del 2 GDL

Parámetro	Valor	Unidades
$\boldsymbol{J_{m2}}$	0.0023	$Kg \cdot m^2 / rad$
$\boldsymbol{B_{m2}}$	3.8957x(10^-4)	$N \cdot m \cdot s/rad$
$\boldsymbol{B_{g2}}$	0.0019	$N \cdot m \cdot s/rad$
$\boldsymbol{K_{g2}}$	0.1593	$N \cdot m/rad$
$\boldsymbol{J_{g2}}$	0.0219	$Kg \cdot m^2/rad$
$\boldsymbol{B_{c2}}$	0.0075	$N \cdot m \cdot s/rad$
$\boldsymbol{K_{c2}}$	0.4566	$N \cdot m/rad$

Una vez que se obtuvieron los valores de la identificación paramétrica para el segundo grado se procedió a realizar la simulación con estos, en la Figura 7.35 se muestra la posición experimental y la obtenida en la simulación del control de trayectoria, se encontró una similitud de 99.47% entre la posición deseada y la experimental.

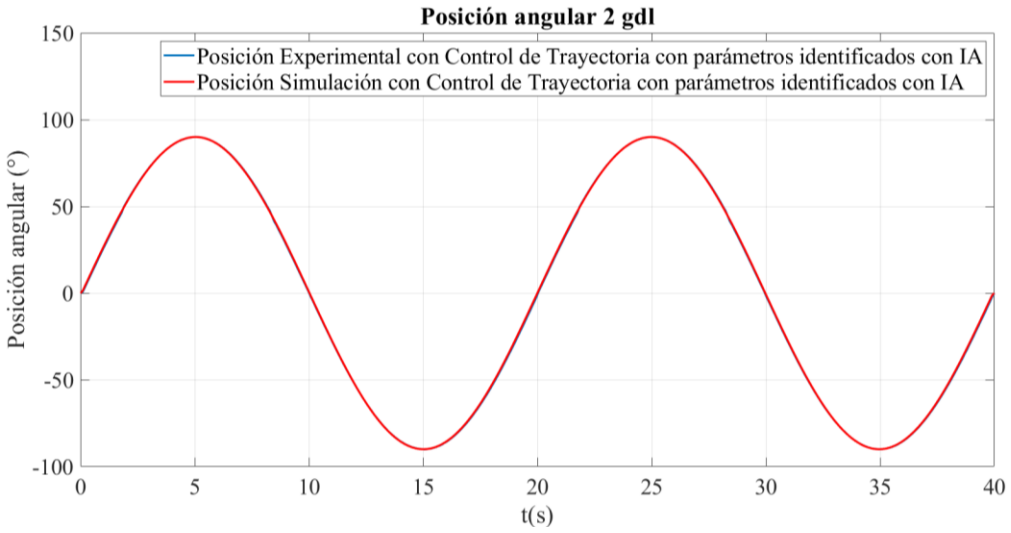


Figura 7.35: Posición Experimental vs Simulación con valores obtenidos de la identificación paramétrica para el 2 gdl.

Con la gráfica de posición se procedió a graficar el error de posición, el cual fue calculado sobre la posición deseada y la experimental, en la Figura 7.35 se muestra la gráfica del error obtenido, se puede notar que en el 2 gdl el error es menor que en el 1 gdl.

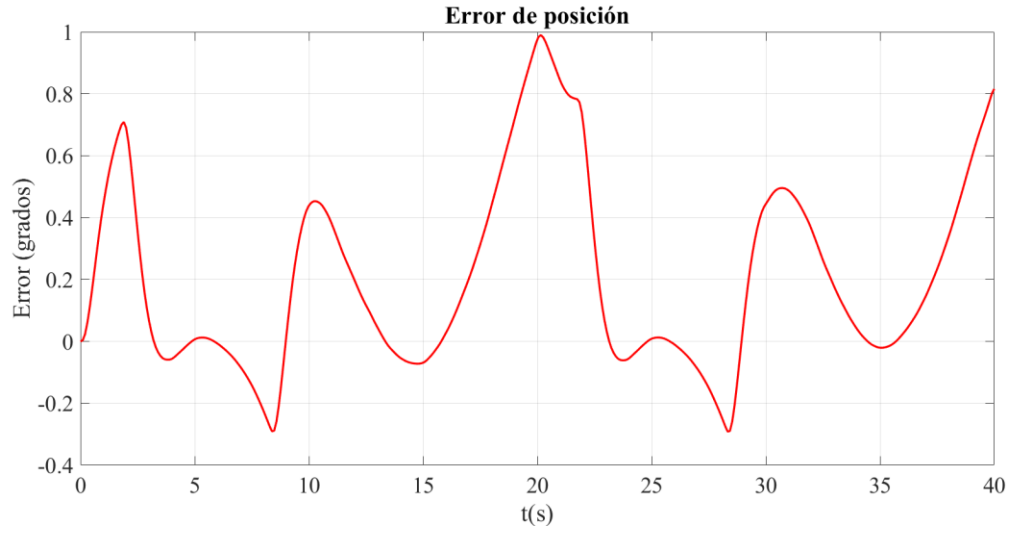


Figura 7.36: Error de posición 2 gdl.

También se graficó la velocidad obtenida en la simulación con los parámetros obtenidos de la identificación paramétrica, la cual puede verse en la Figura 7.37, en esta se puede observar la velocidad deseada, la obtenida por la simulación del control de trayectoria y la velocidad obtenida de forma experimental, se encontró una similitud del 91.66% de similitud entre la velocidad ideal y la real.

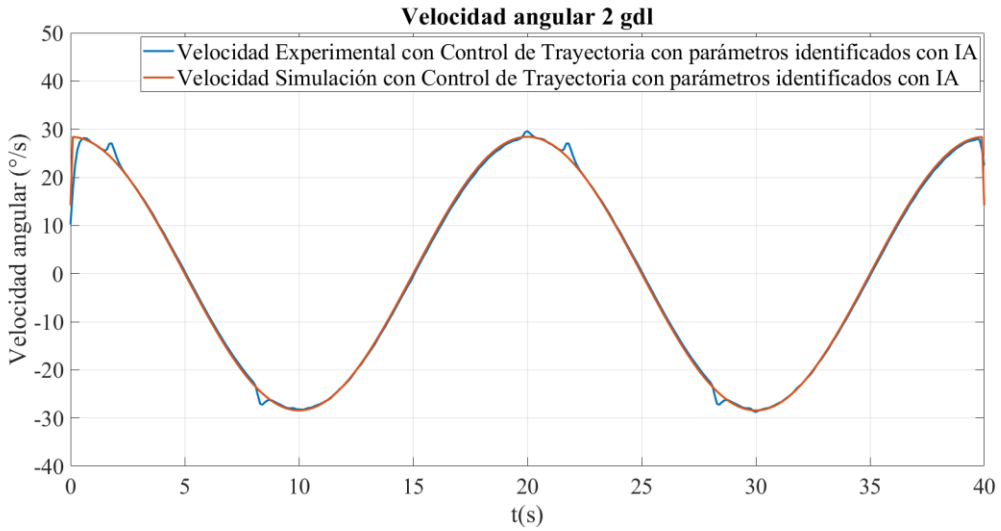


Figura 7.37: Velocidad obtenida con Control de trayectoria Simulada y Experimental con parámetros obtenidos por IA.

Una vez que se obtuvieron los parámetros al realizar la identificación paramétrica, se graficó el torque experimental y el obtenido en simulación con los parámetros hallados por la red, en la Figura 7.38 se observan ambos resultados.

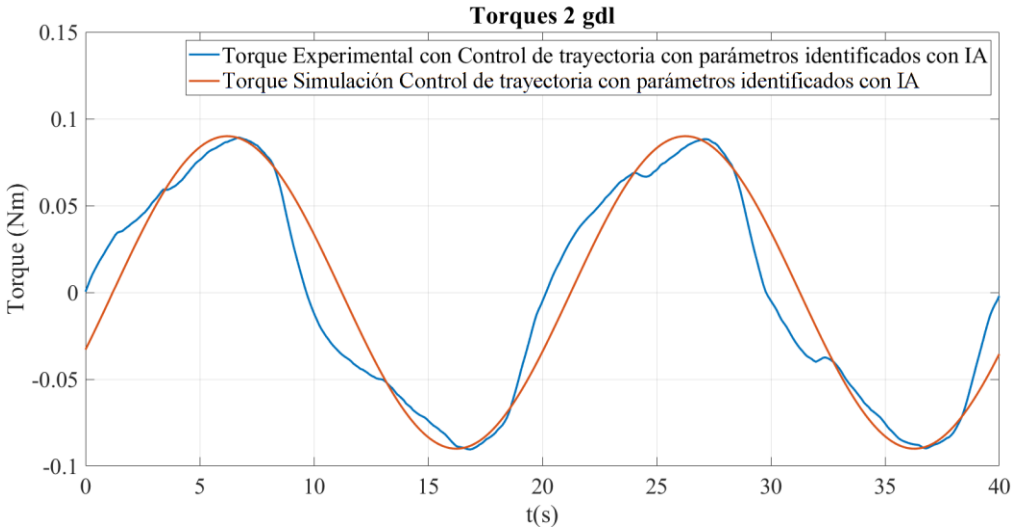


Figura 7.38: Validación de la identificación paramétrica del segundo grado de libertad.

Para obtener la similitud que existe entre el resultado obtenido por la red neuronal y la experimental se usó la métrica diseñada en [3], si bien existen otras métricas como son la de correlación de Pearson y la sumatoria de la norma euclidiana de la resta de las señales a comparar, estás métricas no consideran la similitud que existe en las señales a evaluar en frecuencia, por lo que se puede perder información al momento de señalar la similitud entre dos señales. La métrica usada en este trabajo considera la señal en tiempo y su espectro en frecuencia usando la transformada discreta del coseno, la cual está descrita en la ecuación (56), para esto es necesario evaluar los términos obtenidos usando las señales experimentales y las obtenidas en simulación, la ecuación final está dada por la expresión $\mathbf{metr2ev} = (ab)^{0.4}$ y donde $a\ y\ b$ fueron calculados para el dominio del tiempo y de la frecuencia.

En la Tabla 7.7 se muestran los valores obtenidos usando la métrica presentada para hallar la similitud entre las señales de torque experimentales y simulada, esto para poder evaluar los parámetros obtenidos usando la red neuronal convolucional propuesta.

Tabla 7.7: Similitud de los torques obtenidos experimentalmente del brazo robot

Grado	Similitud
$\boldsymbol{\tau_1}$	80.58%
$\boldsymbol{\tau_2}$	91.54 %

En la Tabla 7.7 se mostró la similitud obtenida usando la red neuronal convolucional propuesta, se puede notar que se alcanza una similitud superior al 80%, Se puede concluir que existen efectos propios del robot que no fueron considerados en el modelo presentado que afectan al comportamiento robot, como son efectos de fricción, juego mecánico, etc.

El siguiente paso es implementar el control de trayectoria directamente en el robot, esto para conseguir una similitud más cercana al 100 %, ya que una vez que se conocen los parámetros del robot, estos ya pueden ser compensados en el control, por lo que para trabajos futuros puede implementarse completamente el control de trayectoria en el brazo robot.

Conclusiones

Al realizar la identificación paramétrica se identificó que es necesario realizar variaciones en los rangos de los parámetros para la generación de las imágenes, por lo que es importante que se tenga un repositorio con determinadas características para favorecer el entrenamiento de la red neuronal.

Se puede observar que la similitud entre los torques reconstruidos y los experimentales superan el 80 %. En los resultados obtenidos se puede notar que el valor más bajo de similitud corresponde al primer grado de libertad, lo cual confirma lo esperado ya que al sostener el peso del motor y del segundo eslabón, este influye de manera más notoria

en los resultados obtenidos de torque, de igual forma se pudo notar que el segundo grado de libertad el torque es menor al solamente mover el segundo eslabón, el cual posee una menor masa en comparación con el primer eslabón.

Se pudo notar que es necesario seguir estudiando los conceptos que involucran las redes neuronales, para poder entender mejor como es que se realiza el entrenamiento de la red neuronal convolucional, así como para poder interpretar los resultados obtenidos, partiendo de las gráficas de entrenamiento y validación.

Finalmente se concluye en este apartado que la identificación paramétrica usando redes neuronales es un área que puede seguir siendo explorada, ya que, a pesar de usar la misma metodología para el desarrollo de las redes neuronales convolucionales, su comportamiento también está dado por el repositorio de imágenes y por consiguiente en el modelo dinámico del robot a estudiar.

Conclusiones generales

Con los resultados presentados en este trabajo se puede concluir que se cumplieron satisfactoriamente todos los objetivos planteados en esta tesis.

La construcción del robot propuesto aporta un elemento a la infraestructura del laboratorio de Automatización, abriendo la posibilidad de su uso en trabajos futuros.

El desarrollo de un modelo dinámico del robot basado en la técnica de parámetros agrupados ha mostrado una mejora respecto a trabajos anteriores, debido a que los resultados obtenidos de la simulación y los resultados experimentales mostraron una coincidencia del 90%, para la posición y velocidad, mientras que el torque solo logró una coincidencia del 30%, cabe señalar que estos resultado son previos a la identificación paramétrica y sin que el modelo dinámico actual considere los fenómenos de fricción. Lo anterior abre la posibilidad que en un trabajo futuro se obtenga un modelo dinámico que incluya fricción.

Se identificó que la generación del repositorio de imágenes es muy importante para poder realizar el entrenamiento de la red neuronal convolucional, ya que se observó que cuando los valores exceden los valores reales que fueron considerados del robot, la red neuronal solo memoriza y no aprende, contrario a cuando se selecciona un rango adecuado para los parámetros a identificar, ya que con este le logra el entrenamiento de la red neuronal convolucional.

La identificación paramétrica obtenida por medio del uso de una IA, dio como resultado que datos obtenidos de la simulación y los datos experimentales coincidieran en posición y velocidad en alrededor del 95%, mientras que la coincidencia del torque incremento sustancialmente ya que paso del 30% a alrededor del 80%, mientras que en trabajos anteriores con robots similares muestran una coincidencia realizada de manera cualitativa como en [28], [29].

Otro trabajo futuro puede ser en el desarrollo de una IA que realice la identificación paramétrica usando un nuevo modelo dinámico que incluya fricción.

Bibliografía

[1] P. P. Cruz, *Inteligencia artificial con aplicaciones a la ingeniería,* 1 Edición, Alfaomega, México, 2010

[2] C. L. C. -D. De León *et al*., "Parameter Identification of a Robot Arm Manipulator Based on a Convolutional Neural Network," in *IEEE Access*, vol. 10, pp. 55002-55019, 2022, doi: 10.1109/ACCESS.2022.3177209.

[3] C. L. Carreón Díaz de Léon*, "*Desarrollo de una Red Neuronal Convolucional para la identificación Paramétrica de un Robot", Tesis doctoral, Facultad de Ciencias de la Electrónica, BUAP, Puebla, Pue. 2023.

[4] Indolia, Sakshi et al. "Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach." *Procedia Computer Science* 132 (2018): 679-688.

[5] A. Zafar et al., "A Comparison of Pooling Methods for Convolutional Neural Networks," Applied Sciences, vol. 12, no. 17, p. 8643, Aug. 2022, doi: 10.3390/app12178643.

[6] M. Kachuee, S. Fazeli and M. Sarrafzadeh, "ECG Heartbeat Classification: A Deep Transferable Representation," *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, New York, NY, USA, 2018, pp. 443-444, doi: 10.1109/ICHI.2018.00092.

[7] Tao, M., Lou, J., & Wang, L. (2022). MRI Liver Image Assisted Diagnosis Based on Improved Faster R-CNN. Traitement du Signal, 39(4).

[8] Liu, Q., & Ding, H. (2022). Application of Table Tennis Ball Trajectory and Rotation-Oriented Prediction Algorithm Using Artificial Intelligence. Frontiers in Neurorobotics, 16, 46.

[9] KUKA AG 2023, El robot industrial: ¿Cómo funciona?, Disponible: https://www.kuka.com/es-es/productos-servicios/sistemas-de-robot/es-bueno-saberlo-robots-industriales

[10] Hanson Robotics Ltd, Sophia 2020, Disponible: https://www.hansonrobotics.com/sophia-2020/

[11] Engineered Arts Limited, "Ameca, The future face of robotics", [En línea], Disponible: https://www.engineeredarts.co.uk/robot/ameca/,

[12] Boston Dynamics, "Atlas and beyond: the world's most Dynamic robots", [En línea], Disponible: https://bostondynamics.com/atlas/

[13] C. L. Carreón-Díaz de León, S. Vergara-Limón, M. A. D. Vargas-Treviño, and J. M. González- Calleros, "A novel methodology of parametric identification for robots based on

a CNN," J. Intell. Fuzzy Syst., vol. 42, no. 5, pp. 4573-4586, Mar. 2022, doi: 10.3233/JIFS-219246.

[14] C. L. Carreón-Díaz de León, S. Vergara-Limón, M. A. D. Vargas-Treviño, J. López-Gómez, J. M. González-Calleros, D. M. González-Arriaga, M. Vargas-Treviño, "Parameter Identification of a Robot Arm Manipulator Based on a Convolutional Neural Network," IEEE Access, vol. 10, pp. 55002-55019, May 2022, doi: 10.1109/ACCESS.2022.3177209.

[15] B. C. Fabien, Analytical System Dynamics: Modeling and simulation", Springer, 2009.

[16] F. Reyes Cortés, *Robótica: Control de Robots Manipuladores*, 2ª ed. Barcelona, España: Marcombo, 2020.

[17] Pololu, Robotics & electronics, 131:1 Metal Gearmotor 37Dx73L mm 12V with 64 CPR Encoder (Helical Pinion),[En línea], Disponible: https://www.pololu.com/product/4756/.

[18] J. Lopez-Gomez et al., "Influence of PWM Torque Control Frequency in DC Motors by Means of an Optimum Design Method," in IEEE Access, vol. 8, pp. 80691-80706, 2020, doi: 10.1109/ACCESS.2020.2990158.

[19] Smoot, J., La inteligencia llegó al codificador de conmutación, [En línea], Disponible: https://www.digikey.com.mx/es/articles/intelligence-comes-to-the-commutation-encoder/.

[20] Alcantar Vergara L. A., Modelado y control de un robot cartesiano de dos grados de libertad realizando estudios de Jerk, Tesis de Maestría, BUAP, 2023.

[21] C. L. Carreón Diaz de León, Diseño y construcción de un cuadricóptero guiado por control inalámbrico, MCEA - FC- BUAP, Puebla, 2019.

[22] D. M. Gonzalez Arriaga, Diseño y construcción de un levitador magnético", MECA - FC - BUAP, Puebla, 2019.

[23] Referencia del Lenguaje D: D. M. Gonzalez Arriaga, M.A.D. Vargas Treviño, S. Vergara Limón "Lenguaje D" Programas de computacion, Registro Publico del Derecho de Autor, numero de registro: 03-2021-120110454200-01, 2021.

[24] Altera, "Mouser México," 2017. [En línea]. Available: https://www.mouser.mx/ProductDetail/IntelAltera/EP4CE22F17C6N?qs=jblrfmjbeiGZz%252BplBRwiDQ==

[25] Sparkfun, «Sparkfun RN-XV WiFly Module - Wire Antenna,» 2016. [En línea]. Available: https://www.sparkfun.com/products/retired/10822.

[26] F. Berzal, Redes Neuronales & Deep Learning, 1st ed. Granada, Spain: Independently published, 2018.

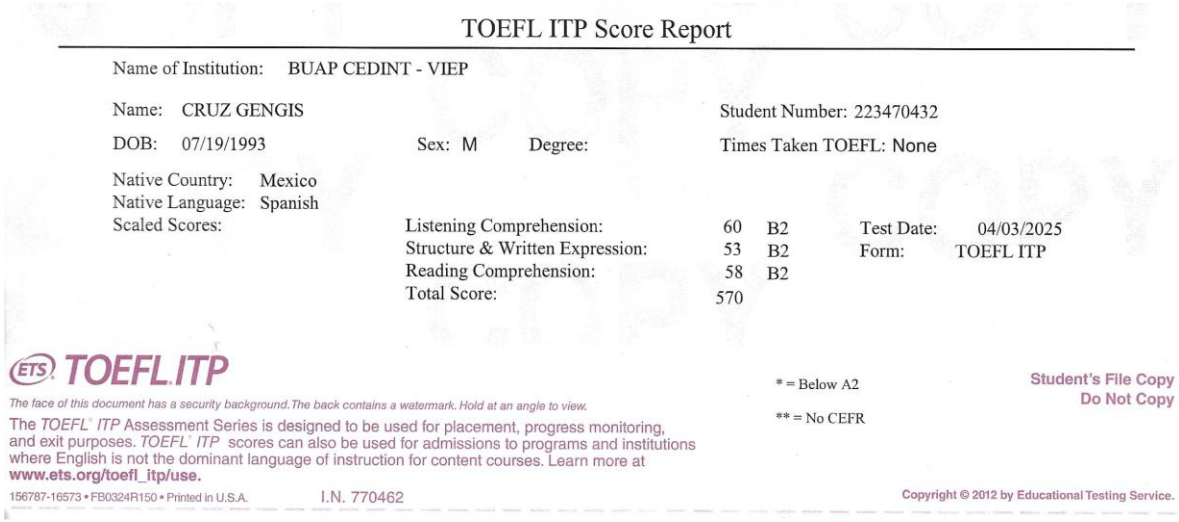
[26] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," IEEE Trans. Neural Netw. Learn. Syst., Early Access, June 2021, doi:10.1109/TNNLS.2021.3084827

[27] S. S. Arrián, Todo sobre sistemas embebidos, Lima: Universidad de ciencias Peruanas, 2017.

[28] Sánchez Pérez, J., Análisis, modelado, control y aplicaciones de Robot 2GDL de accionamiento directo. Trabajo de Fin de grado.

[29] Hüseynoğlu, M., & Abut, T. (2018). Dynamic model and control of 2-DOF robotic arm. European Journal of Technique, 8(2), 115–124.

Anexos



TOEFL ITP Score Report

Name of Institution: BUAP CEDINT - VIEP

Name: CRUZ GENGIS
Student Number: 223470432

DOB: 07/19/1993 Sex: M Degree:
Times Taken TOEFL: None

Native Country: Mexico
Native Language: Spanish
Scaled Scores:

Listening Comprehension:	60	B2	Test Date:	04/03/2025
Structure & Written Expression:	53	B2	Form:	TOEFL ITP
Reading Comprehension:	58	B2		
Total Score:	570			

TOEFL ITP

The face of this document has a security background. The back contains a watermark. Hold at an angle to view.

The *TOEFL ITP* Assessment Series is designed to be used for placement, progress monitoring, and exit purposes. *TOEFL ITP* scores can also be used for admissions to programs and institutions where English is not the dominant language of instruction for content courses. Learn more at **www.ets.org/toefl_itp/use.**

156787-16573 • FB0324R150 • Printed in U.S.A. I.N. 770462

* = Below A2

** = No CEFR

Student's File Copy
Do Not Copy

Diseño de una Red Neuronal Convolucional para Identificar 2 Parámetros de un Brazo Robótico

Ing. Gengis Cruz Coronel[1], Dra. María Aurora Diozcora Vargas Treviño[2],
Dr. Sergio Vergara Limon[3], Dr. Jesús López Gómez[4] y Dr. Carlos Leopoldo Carreón Díaz de León [5]

***Resumen*—**En este artículo se presenta el diseño de una red neuronal para la identificación de 2 parámetros de un brazo robot articular de dos grados de libertad, el cual ha sido desarrollado en la Maestría de Ciencias de la Electrónica Opción Automatización de la Benemérita Universidad Autónoma de Puebla. Se comenzará con una breve mención del estado del arte de las redes neuronales convolucionales, se expondrá el diseño del brazo robótico y la electrónica para su control usando una tarjeta de desarrollo FPGA. También se mostrará el modelo dinámico usado en este estudio y la metodología para realizar la identificación paramétrica usando redes neuronales convolucionales, así como la técnica para la generación del repositorio de imágenes para el entrenamiento de la red. Finalmente se mostrará el diseño de la red neuronal convolucional para la identificación paramétrica del brazo robótico y los resultados obtenidos para el repositorio de imágenes de entrenamiento de la red. La importancia de este trabajo yace en encontrar un modelo dinámico con una respuesta más cercana a la realidad.

***Palabras clave*—**Red neuronal, robot articular, FPGA, identificación paramétrica.

Introducción

La identificación paramétrica dentro de la robótica permite obtener los parámetros dinámicos desconocidos (inercia, fricción, rigidez, etc) de un robot manipulador. Este procedimiento es importante ya que, al conocer estos parámetros del robot y su modelo dinámico, este puede ser simulado. Por lo tanto, entre mejor se realice la identificación paramétrica más certera será su respuesta con la realidad y, se tendrá un mejor modelo dinámico del robot manipulador. Además, con este modelo dinámico y sus parámetros correctos, es posible ver su comportamiento a diferentes entradas y técnicas de control. Esto ayuda a realizar pruebas en simulación antes de aplicarlas al robot real.

El área de inteligencia artificial ha avanzado constantemente en diversas áreas, un área de estas es el estudio de las redes neuronales convolucionales, este tipo de redes se basan en la operación de convolución la cual es usada en el procesamiento digital de señales, se ha demostrado que los resultados obtenidos logran un valor muy alto de similitud entre los datos obtenidos de las simulaciones usando los parámetros y los resultados obtenidos de forma experimental tal como menciona Carreón (2022) por lo que esta técnica puede ser aplicada para la identificación paramétrica de diferentes robots.

Descripción del Método

Antecedentes

En la literatura se pueden encontrar cuatro técnicas de identificación paramétrica: mínimos cuadrados, máxima verosimilitud, redes neuronales e ingeniería inversa, sin embargo, para la técnica de mínimos cuadrados es necesario que la trayectoria a seguir cumpla con la excitación de los parámetros para poder aplicarla, por lo que la selección de esta trayectoria también debe ser probada usando el robot real.

Las redes neuronales convolucionales remontan a 1943 cuando se propuso el primer modelo computacional de una red neuronal, con la idea de usar una estructura similar al cerebro, donde se ajustan los parámetros hasta que la salida sea la deseada, de forma que pueda aprender como lo haría un cerebro biológico (Berzal, 2019).

La identificación paramétrica usando redes neuronales convolucionales usando las señales generadas de torque para generar imágenes es una nueva técnica para realizar la identificación paramétrica, demostrando hasta un

[1] Ing. Gengis Cruz Coronel es alumno de la Maestría en Ciencias de la Electrónica Opción en Automatización en la Benemérita Universidad Autónoma de Puebla, Puebla, México. cc2234070432@alm.buap.mx
[2] La Dra. María Aurora Diozcora Vargas Treviño es Profesora de la Maestría en Ciencias de la Electrónica Opción en Automatización en la Benemérita Universidad Autónoma de Puebla, Puebla, México. aurora.vargas@correo.buap.mx
[3] El Dr. Sergio Vergara Limon es Profesor de la Maestría en Ciencias de la Electrónica Opción en Automatización en la Benemérita Universidad Autónoma de Puebla, Puebla, México. sergio.vergara@correo.buap.mx
[4] El Dr. Jesús López Gómez es Profesor de la División Académica de Ingeniería y Arquitectura, en la Universidad Juárez Autónoma de Tabasco (DAIA-UJAT), Tabasco, México. jlgo6599@docente.ujat.mx
[5] El Dr. Carlos Leopoldo Carreón Díaz de León es Profesor de la Facultad de Ciencias de la Computación en la Benemérita Universidad Autónoma de Puebla, Puebla, México. carlos.carreon@correo.buap.mx

ACADEMIA JOURNALS
OPUS PRO SCIENTIA ET STUDIUM

Congreso Internacional de Investigación
Academia Journals Los Mochis 2025
Construyendo Legados, Innovando el Futuro: Excelencia en Empresas Familiares

CONGRESO INTERNACIONAL DE INVESTIGACIÓN ACADEMIA JOURNALS LOS MOCHIS 2025

Construyendo Legados, Innovando el Futuro Excelencia en Empresas Familiares

CERTIFICADO

otorgado a

Ing. Gengis Cruz Coronel
Dra. María Aurora Diozcora Vargas Treviño
Dr. Sergio Vergara Limon
Dr. Jesús López Gómez
Dr. Carlos Leopoldo Carreón Díaz de León

por su artículo titulado

Diseño de una Red Neuronal Convolucional para Identificar 2 Parámetros de un Brazo Robótico

(Artículo No. LSM030)

La ponencia correspondiente fue presentada en el Congreso Internacional de Investigación Academia Journals Los Mochis 2025, *Construyendo Legados, Innovando el Futuro: Excelencia en Empresas Familiares,* desarrollado el 27 de febrero de 2025, con sede en el Centro de Innovación y Educación Los Mochis. El evento fue presentado en colaboración con eminentes investigadores de Los Mochis. El artículo está incluido en las siguientes publicaciones: (1) en el portal de Internet AcademiaJournals.com, con ISSN 1946-5351 online, Vol. 17, No. 01, 2025 e indexación en la base de datos *Fuente Académica Plus de EBSCOHOST,* Massachusetts, Estados Unidos y (2) en libros ebook digitales compilados por área temática, con números ISBN online*. Se tiene acceso libre a todas las publicaciones del congreso en el portal de internet de Academia Journals.

Los organizadores del congreso reconocen la participación de los autores en el congreso, agradeciendo sus contribuciones.

DR. RAFAEL MORAS, P.E.
Director General
Academia Journals

CONGRESO INTERNACIONAL DE INVESTIGACION ACADEMIA JOURNALS LOS MOCHIS 2025

* Construyendo Legados Innovando el Futuro en las Ciencias Administrativas - Los Mochis 2025 979-8-89020-109-6
Construyendo Legados Innovando el Futuro en las Ciencias de la Salud - Los Mochis 2025 979-8-89020-110-2
Construyendo Legados Innovando el Futuro en las Matemáticas y Ciencias Naturales - Los Mochis 2025 979-8-89020-111-9
Construyendo Legados Innovando el Futuro en las Ciencias de la Educación - Los Mochis 2025 979-8-89020-112-6
Construyendo Legados Innovando el Futuro en las Humanidades y Ciencias Sociales - Los Mochis 2025 979-8-89020-113-3
Construyendo Legados Innovando el Futuro en la Ingeniería - Los Mochis 2025 979-8-89020-114-0