INSTITUTO TECNOLÓGICO DE CD. GUZMÁN

TÉSIS

TEMA:

DISEÑO DE UN SISTEMA HÁPTICO PARA LA CLASIFICACIÓN DE SEÑALES DE MOVIMIENTO DEL BRAZO PARA LA DETECCIÓN DEL MAL DE PARKINSON

QUE PARA OBTENER EL TÍTULO DE:

MAESTRO EN INGENIERÍA ELECTRÓNICA

PRESENTA:

ING. JOSÉ LUIS CORTÉS MENDOZA

DIRECTOR(A):

DR. JAIME JALOMO CUEVAS

CODIRECTOR(A):

DR. JORGE GUDIÑO LAU

CD. GUZMÁN JALISCO, MÉXICO, AGOSTO DE 2022

EDUCACIÓN

Instituto Tecnológico de Ciudad Guzmán
Subdirección Académica / División de Estudios de Posgrado e Investigación

Ciudad Guzmán, Jalisco, 03/agosto/2022
Asunto: Autorización de impresión de Tesis

ING. JOSÉ LUIS CORTES MENDOZA
CANDIDATO AL GRADO DE MAESTRO EN INGENIERÍA ELECTRÓNICA
PRESENTE

De acuerdo con los Lineamientos para la Operación de los Estudios de Posgrado en el Tecnológico Nacional de México y las disposiciones en este Instituto, habiendo cumplido con todas las indicaciones que la Comisión Revisora realizó con respecto a su Trabajo de Tesis titulado **"Diseño de un sistema háptico para clasificación de señales del movimiento del brazo para detección del mal de Parkinson"**, la División de Estudios de Posgrado e Investigación de este Instituto, concede la Autorización para que proceda a la impresión del mismo.

Sin otro particular, quedo de Usted.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"2022: AÑO DEL CINCUENTA ANIVERSARIO DEL INSTITUTO TECNOLÓGICO DE CIUDAD GUZMÁN"

MARÍA GUADALUPE SÁNCHEZ CERVANTES
JEFA DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

EDUCACIÓN
INSTITUTO TECNOLÓGICO DE CD. GUZMÁN
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

c.c.p. Archivo

MGSC/mxgg

2022 Ricardo Flores Magón

AGRADECIMIENTOS

Es comúnmente dicho que la grandeza existe dentro de cada uno de nosotros lo cual es cierto, se cree también que a pesar de cualquier circunstancia, teniendo un deseo suficientemente grande uno bien podría lograr por sí mismo la más grande hazaña a base de puro ímpetu, sin embargo, al finalizar esta etapa en mi vida, puedo decir que en lo que a la opinión de un humilde servidor concierne, esta última creencia es una mentira, un vago pensamiento de una mente inexperta resultado de la ingratitud que puede obrar el engañoso poder del conocimiento, en una mente tan débil como lo es la de un humano a veces.

A lo largo de la historia hemos escuchado de grandes logros y siendo completamente imparciales, indagando en cada uno de estos relatos, podemos darnos cuenta de que no hay ni un solo gran destino, que no haya sido alcanzado sino por el impulso y la sutil influencia de muchos personajes tuvieron en la vida de un solo individuo. Si bien al narrar la historia estos toman el papel de secundarios, la realidad es que lo logrado solo puede ser descrito como la suma de todos los esfuerzos y recursos invertidos en pro de la meta de ese individuo que eventualmente goza, injustamente en solitario, de los beneficios de su meta materializada.

Si redefinimos entonces la grandeza, sería obligatorio incluir en la definición lo indispensables que son todas esas personas con las que nos cruzamos en el curso de una vida, las que señalaron errores y las que no olvidaron encomiarnos por lo logrado, las que odiaron nuestro progreso y las que gustosos nos dieron de sí para seguir progresando, los que por sus actos demostraron querer ser enemigos y resultaron ser maestros, los que se acercaron como amigos y se convirtieron en animadores, todos ellos, porque de cada uno de ellos tomamos un poco para forjarnos a nosotros mismos.

Dicho en cortas cuentas, sabernos incompletos nos hace grandes al nunca truncar nuestro aprendizaje, ser grande es necesitar de los demás y entender que somos lo que somos por cada una de las personas que nos enseñaron algo. Este trabajo representa el fin de mi etapa como universitario, sin embargo, plasma mi sentir acerca de lo logrado, no solo seglarmente, sino en la vivencia completa de haber sido estudiante, hijo, amigo, enemigo o cualquier identidad que para un espectador haya podido tomar mientras vivía hasta este punto del viaje.

Agradezco primero que nadie a Dios, que pusiera las personas correctas en mi vida y me demostró que siempre estuvo al pendiente de que fuera feliz cuando acertaba y consciente de las cosas que era necesario mejorar cuando me equivocaba, todo para ser una mejor persona. Quiero agradecer a mi institución educativa el Instituto Tecnológico de Ciudad Guzmán parte del Tecnológico Nacional de México de igual forma. También mi agradecimiento al Consejo Nacional de Ciencia y Tecnología por el apoyo económico otorgado en forma de beca.

Sin embargo, quiero agradecer especialmente a mis padres José Luis Cortés Contreras y Ma. Teresa Mendoza Reyes, por todo el esfuerzo que implicó en diversas ocasiones el dar lo necesario para que pudiera cumplir lo que yo quería, esa clase de amor abnegado es de lo que están hechas las grandes historias, y en mi historia, lo que soy lo debo a ellos, cada consejo, cada vivencia, cada regaño, cada abrazo fue una confirmación de que había alguien allí que siempre quiso lo mejor para mí, sin pensar siquiera en sí lo merecía. Sería una injusticia no recordar ahora que esta etapa ha concluido, que cada día que asistí a la escuela fue el producto del trabajo y la preocupación por mí, confiando en hacer de mi vida una vida para disfrutar, sin embargo, la realidad es que todo trabajo requiere un desgaste y si bien yo trabaje en la escuela, los que se desgastaron fueron ellos, lidiando conmigo en mis malos momentos y poniendo años de su vida en la mesa con el único propósito de invertirlos en mi causa, por eso lo que me queda hacer es mostrarles con mi vida a partir de ahora que cada sacrificio fue aprovechado y forma parte de lo que soy, pues ya no se puede hablar de quien soy sin primero hablar de lo que mis padres hicieron de mí, un ser con la convicción de vivir como un buen

hombre, tomando lo que mis padres le dieron a solo un niño, este trabajo es para ellos.

Este agradecimiento abarca desde luego a mis guías en este viaje, a todos los profesores que he tenido el honor de conocer y de aprender de ellos, ahora tengo más hambre que nunca por seguir aprendiendo y eso es posible porque hubo profesores que al enseñar no creían hacer un trabajo, sino que veían una vocación, la vocación de enseñar. En cada uno de ellos veo a un mentor y un ejemplo de vida, que tengan ellos por seguro que todo lo aprendido no ha sido en vano, ahora es parte de mí y lo llevaré conmigo siempre.

Desde luego también deseo agradecer a mis asesores ene este trabajo, quienes fueron parte importante de mi desarrollo en este nuevo ámbito del conocimiento, particularmente al Dr. Jorge Gudiño Lau, quién me brindó de su tiempo y me dio la libertad y confianza de desarrollar mis ideas bajo su tutela.

Por último, quiero agradecer, a cada uno de los miembros de mi familia en quienes nunca faltó esa fe que todos necesitamos que los demás tengan en nosotros, pero más importante aún las personas que son tan cercanas a nosotros.

En conclusión, mi vida ha sido grande hasta ahora, no por lo que hice sino por las grandes personas que hubo siempre a mi lado, todos ellos han hecho más que lo suficiente para tener mi agradecimiento por siempre.

RESUMEN

En este trabajo se aborda el proceso de reingeniería de un dispositivo háptico Phantom Premium de la marca Sensable Technologies el cual salió al mercado hace más de 20 años. Este dispositivo no funcionaba adecuadamente y debido a las complicaciones que conllevaba repararlo, se optó por crear una nueva plataforma de trabajo para el mismo. Esta plataforma tiene como propósito registrar el movimiento del actuador final a través de una gráfica que registre el movimiento de este en un plano de 3 ejes (x, y, z), lo cual fue posible gracias a la obtención de la cinemática del robot. Utilizando los trazos obtenidos de diferentes trayectorias se entrena una red neuronal para hacer la clasificación de trayectorias, de manera que después sea posible detectar anormalidades en el trazo de una trayectoria definida, lo cual permitirá ayudar al diagnóstico de enfermedades como el Parkinson, que provocan rigidez y temblor.

ABSTRACT

This work takes on the reengineering process of a Phantom Premium haptic device of Sensable Technologies, this device went out to market about 20 years ago. This device wasn´t working properly when it was donated to the Colima's University and due to the complications included in trying to repair it the best option was to build and program a brand-new work platform. This platform's purpose is to register the final actuator's movement using a routine which will represent the movement into a 3 axes graph (x, y, z), all of which was possible using the robot's kinematic model. Using the movement graphs obtained a neural network was trained to classify different trajectories so it can detect abnormalities in the graph of a specific trajectory, this hopefully can be helpful to diagnose neurodegenerative diseases such as Parkinson's, that come with symptoms such as rigidity and tremor.

ÍNDICE DE CONTENIDOS

AGRADECIMIENTOS 2

RESUMEN 5

ABSTRACT 5

ÍNDICE DE CONTENIDOS 6

CAPÍTULO I. INTRODUCCIÓN Y ANTECEDENTES 13

1.1. INTRODUCCIÓN 13

1.2. ANTECEDENTES 14

1.3. JUSTIFICACIÓN 19

CAPÍTULO II. MARCO TEÓRICO 22

2.1. PYTHON 22

2.2. LABVIEW 24

2.3. COMUNICACIÓN SERIAL 26

2.4. VOLTAJE 26

2.5. CORRIENTE 27

2.6. RESISTENCIA 27

2.7. IDE 28

2.8. BAUDIO 30

2.9. PUERTO COM 31

2.10. LA NORMA RS-232 31

2.11. CINEMÁTICA 32

2.12. ARDUINO 37

2.13. CODIFICADOR ROTATORIO 45

2.14. MOTOR CD 46

2.15. PUENTE H 49

2.16. PWM 49

2.17. REDES NEURONALES 51

CAPÍTULO III. METODOLOGÍA 54

3.1. REINGENIERÍA DEL DISPOSITIVO HÁPTICO 54

3.2. DISEÑO DE RED NEURONAL 76

CAPÍTULO IV. RESULTADOS 81

4.1. ACTIVIDADES DEL PROYECTO 81

4.2. ACTIVIDADES ADICIONALES 87

4.2.1. CREACIÓN DE UNA PLATAFORMA EN LABVIEW PARA EL CONTROL DEL DISPOSITIVO HÁPTICO 87

4.2.2. CONTROL DE UN MODELO 3D DEL DISPOSITIVO HÁPTICO MEDIANTE SIMMECHANICS 91

4.2.3. CONTROL PD+I DIFUSO DE UN DISPOSITIVO HÁPTICO PHANTOM OMNI 95

4.2.4. CONTROL PD+I DIFUSO EN CASCADA PARA UN SISTEMA BALL AND BEAM 98

CAPÍTULO V. CONCLUSIONES 106

Bibliografía 109

ANEXOS 112

ÍNDICE DE FIGURAS

Figura 1 Dispositivo no invasivo para el diagnóstico del mal de Parkinson 15
Figura 2 Diagrama a bloques del dispositivo de diagnóstico de Parkinson 15
Figura 3 Sistema inalámbrico de análisis del andar para el diagnóstico de Alzheimer y Parkinson 16
Figura 4 Diagrama a bloques del hardware utilizado para el sistema.....16
Figura 5 Comparativa de tamaño del dispositivo con una moneda de 2 euros.....17
Figura 6 Gráfica para la interpretación de los datos obtenidos por el dispositivo ..17
Figura 7 Movimientos propuestos para el estudio de movimiento de la mano.....18
Figura 8 Dispositivo y su sugerencia de uso 18
Figura 9 Logo de Python.....22
Figura 10 Ejemplos de algunas librerías que contiene Python 24
Figura 11 Ejemplo del panel frontal en LabVIEW 25
Figura 12 Ejemplo de un diagrama a bloques en LabVIEW 26
Figura 13 Esquema de comunicación serial.....26
Figura 14 Representación de cuerpo sometido a un voltaje entre A y B 27
Figura 15 Representación de flujo de corriente a través de un conductor 27
Figura 16 Triángulo de Ohm 28
Figura 17 GNU Emacs, un editor normalmente utilizado como IDE en sistemas tipo UNIX.....30
Figura 18 Conexión de puerto Rs232 32
Figura 19 Diagrama para ejemplificar un sistema de coordenadas de referencia .33
Figura 20 Posición de un manipulador con respecto a un sistema de referencia ..35
Figura 21 Diagrama para cinemática inversa de un robot manipulador.....36
Figura 22 Logo de Arduino.....37
Figura 23 Diagrama a bloques de un microcontrolador.....39
Figura 24 Tarjeta Arduino Uno.....40
Figura 25 Tarjeta Arduino Mega.....42
Figura 26 Tarjeta Arduino Esplora 42
Figura 27 Tarjeta Arduino Leonardo 43
Figura 28 Tarjeta Arduino Nano.....44

Figura 29 Tarjeta Arduino Due 45
Figura 30 Encóder rotatorio 45
Figura 31 Diagrama de las fuerzas experimentadas por la espira en un motor CD 48
Figura 32 Esquemático de un puente H 49
Figura 33 Gráfica de la salida experimentada por Una carga ante un PWM 50
Figura 34 Ciclos de trabajo en un PWM 51
Figura 35 Esquema de una red neuronal con una capa oculta 53
Figura 36 Control Box 54
Figura 37 Amplificadores PWM modelo 303 de Copley Motion Systems 55
Figura 38 Dispositivo háptico Phantom Premium 56
Figura 39 Encóder HEDM-5500 57
Figura 40 Diagrama de las terminales del HEDM-5500 57
Figura 41 Señales de los canales del HEDM-5500 58
Figura 42 Diagrama abstracto de movimiento del dispositivo háptico 59
Figura 43 Posición inicial del dispositivo háptico 61
Figura 44 Notaciones para transformaciones de cuerpo rígido y cálculos cinemáticos 61
Figura 45 Diagrama a bloques del sistema para obtención de datos para el dispositivo háptico 64
Figura 46 Primera parte del programa en LabVIEW 68
Figura 47 Programa para la recepción y procesamiento de datos de la tarjeta de adquisición 69
Figura 48 VI para la verificación de la cadena de entrada recibida por el puerto serial 70
Figura 49 VI de procesamiento de cadena 70
Figura 50 Rutina para la escritura de datos en un archivo de texto 71
Figura 51 Interfaz gráfica del programa de adquisición de datos 72
Figura 52 Señales obtenidas para una trayectoria, circular, cuadrada y triangular 73
Figura 53 Trazos obtenidos después del programa de procesamiento de imágenes de Python 76

Figura 54 Trazo con una trayectoria circular y dos trayectorias circulares 76
Figura 55 imágenes giradas suministradas para el entrenamiento de la red neuronal 78
Figura 56 Clasificación de trazo de una trayectoria circular (a) 81
Figura 57 Clasificación de trazo de una trayectoria circular (b) 81
Figura 58 Clasificación de trazo de una trayectoria circular (c) 82
Figura 59 Clasificación de trazo de varias trayectorias circulares (a) 82
Figura 60 Clasificación de trazo de varias trayectorias circulares (b) 82
Figura 61 Clasificación de trazo circular (a) 83
Figura 62 Clasificación de trazo circular (b) 83
Figura 63 Clasificación de trazo circular (c) 84
Figura 64 Clasificación de trazo cuadrado (a) 84
Figura 65 Clasificación de trazo cuadrado (b) 84
Figura 66 Clasificación de trazo cuadrado (c) 85
Figura 67 Clasificación de trayectoria circular sin oscilaciones (a) 85
Figura 68 Clasificación de trayectoria circular sin oscilaciones (b) 85
Figura 69 Clasificación de trayectoria circular sin oscilaciones (c) 86
Figura 70 Clasificación de trayectoria circular con oscilaciones (a) 86
Figura 71 Clasificación de trayectoria circular con oscilaciones (b) 86
Figura 72 Clasificación de trayectoria circular con oscilaciones (c) 86
Figura 73 Interfaz gráfica del programa para el control de dispositivo háptico 88
Figura 74 Diagrama a bloques del programa de adquisición de datos y control del dispositivo háptico.....89
Figura 75 Diagrama a bloques de la rutina de composición del mensaje con las señales de control.....90
Figura 76 Área en el diagrama a bloques del programa para posicionar el controlador.....90
Figura 77 Modelo 3D del dispositivo háptico hecho en SolidWorks.....91
Figura 78 Diagrama a bloques general del sistema 92
Figura 79 Diagrama a bloques del controlador y planta para los motores de las articulaciones.....93

Figura 80 Respuesta de la cintura ante un desplazamiento de 0.5 radianes.....94
Figura 81 Respuesta del antebrazo ante un desplazamiento de 0.3 radianes 94
Figura 82 Respuesta del brazo ante un desplazamiento de 0.2 radianes 95
Figura 83 Posición inicial y final del dispositivo háptico.....95
Figura 84 Dispositivo háptico Phantom Omni.....96
Figura 85 Diagrama a bloques en Simulink del sistema de control para el Phantom Omni.....96
Figura 86 Diagrama a bloques del controlador PD+I difuso 97
Figura 87 Gráficas del seguimiento de una trayectoria senoidal de las articulaciones con respecto a la referencia.....98
Figura 88 Diagrama a bloques del sistema de control en cascada con planta para un sistema Ball and Beam 99
Figura 89 Funciones de membresía para la entrada de error de posición de la bola 100
Figura 90 Funciones de membresía para la entrada de la derivada del error de posición de la bola.....100
Figura 91 Funciones de membresía de la salida de control (en radianes).....101
Figura 92 Diagrama a bloques del controlador PD difuso + I 102
Figura 93 Respuesta del controlador de posición de la bola a una señal cuadrada 103
Figura 94 Respuesta del controlador de posición del motor.....104
Figura 95 Respuesta del controlador de posición de bola a una señal senoidal .104
Figura 96 Respuesta del controlador de posición del motor.....105

ÍNDICE DE TABLAS

Tabla 1 Parámetros del motor.....92
Tabla 2 Tarjeta de reglas del controlador PD difuso 93
Tabla 3 Reglas del controlador PD difuso 101
Tabla 4 Reglas del controlador PD difuso para un motor CD propuesto por (Cortés, 2021) 102

CAPÍTULO I. INTRODUCCIÓN Y ANTECEDENTES

1.1. INTRODUCCIÓN

Bajo la premisa de que el avance tecnológico debe estar dirigido al mejoramiento de la calidad de vida del ser humano se han hecho grandes esfuerzos en diferentes áreas del conocimiento para adherirse a este precepto. Una de las formas en que la tecnología puede ser de verdad práctica es en el área de la salud, un ámbito que, si bien recibe a muchos investigadores para solucionar sus males, en ocasiones puede ser un área caprichosa por lo compleja que resulta, tanto que algunas tecnologías simplemente han alcanzado su límite en las contribuciones que pueden hacer, y son las tecnologías emergentes las que deben ponerse al servicio de la salud para causar un impacto de verdad. La inteligencia artificial es desde hace tiempo el estandarte del avance tecnológico contemporáneo, y en este trabajo se explora un área de oportunidad para esta rama del conocimiento, particularmente en la asistencia para el diagnóstico de un mal que, si bien no es muy común, es implacable con el grupo de personas que lo padecen, a saber, el mal de Parkinson. Lo que se propone es el uso de un robot de tres grados de libertad para, mediante la detección de variaciones en el seguimiento de una trayectoria de movimiento definida, diagnosticar el degradamiento de las funciones motoras de personas con males neurodegenerativos. Ahora bien, debido a que el robot que se tenía disponible no está especializado para dicha labor y además estaba averiado, la primera parte del trabajo se centra en hacer un proceso de diagnóstico y posteriormente, reingeniería para ponerlo en funcionamiento, para después hacer el cálculo de su cinemática y hacer posible la obtención de señales de movimiento en tres ejes. La información recabada se procesa mediante programas que transforman la información a imágenes y después procesan dichas imágenes para dejarlas listas para ser usadas en el entrenamiento de una red neuronal. La red neuronal se entrena para hacer la diferenciación primero en tres señales que representan el trazado de una sola trayectoria circular y el trazado de varias de ellas, así como también entre el trazo trayectorias que describen un círculo y un cuadrado

respectivamente y por último entre trazos de trayectorias circulares suaves y trazos con trayectorias afectadas por vibraciones emulando el temblor de un paciente con Parkinson. Estos ejercicios plantearán la base para incentivar a más investigadores a involucrarse en esta iniciativa y recopilar trazos de trayectorias de personas con mal de Parkinson, lo que permita tener una base de datos lo suficientemente grande para utilizar tecnologías más avanzada como Machine Learning, que permitan una detección temprana de estos males en las personas afectadas, algo que para el presente trabajo se presenta como una limitación, al no contar con los recursos ni la infraestructura para hacer un primer acercamiento a esta idea.

1.2. ANTECEDENTES

Revisando la literatura se encontraron algunas investigaciones relacionadas con la detección de enfermedades motriz degenerativas, las cuales analizaremos a continuación.

En su artículo “A Neural Network System for Diagnosis and Assessment of Tremor in Parkinson Disease Patients”, (Omid Bazgir, 2015) un grupo de investigadores propuso un método no invasivo para detectar y diagnosticar la enfermedad de Parkinson, el dispositivo sustituía la evaluación de la escala UPDRS, la cual los médicos obtienen mediante una evaluación sugestiva muy sujeta a su propia interpretación, por lo que no resulta del todo exacta. Esta idea se vale de una estructura de aprendizaje de reconocimiento de patrones supervisada para evaluar el temblor en pacientes con PD, se utiliza el acelerómetro de dispositivo celular, el cual se enfunda en un brazalete que usará el paciente en uno de los brazos. El material del brazalete hace posible evitar detectar vibraciones que no tienen que ver con la condición del paciente.

Figura 1 Dispositivo no invasivo para el diagnóstico del mal de Parkinson

El esquema de funcionamiento a nivel de bloques se puede visualizar en la Figura 2.

Figura 2 Diagrama a bloques del dispositivo de diagnóstico de Parkinson

El sistema que se desarrolló resultó ser muy apropiado para el diagnóstico y control remoto de enfermedades neurodegenerativas con desordenes de movimiento, especialmente en etapas iniciales.

Otra iniciativa más es usar la tecnología de dispositivos móviles como relojes inteligentes para el análisis de la forma de andar o “el paso” de las personas, así se muestra en la investigación de (W. Tao, 2012), donde se buscó establecer e identificar características en el paso de las personas que son potencialmente marcadores tempranos de diagnóstico de Parkinson.

En el mismo renglón del análisis de el paso de una persona tenemos la investigación de Margiotta, la cual se centró en el desarrollo de un sistema inalámbrico de análisis del andar para el diagnóstico de Alzheimer y Parkinson. El dispositivo consta de un clúster el cual se conecta vía bluetooth a un PC, el clúster contiene varios sensores,

así como otros dispositivos básicos para su operación, como muestra la Figura 3, uno de estos dispositivos toma datos posicionándose arriba de la rodilla y un segundo hace lo mismo, pero por debajo de la rodilla.

Figura 3 Sistema inalámbrico de análisis del andar para el diagnóstico de Alzheimer y Parkinson

Un diagrama del hardware utilizado se muestra en la Figura 4.

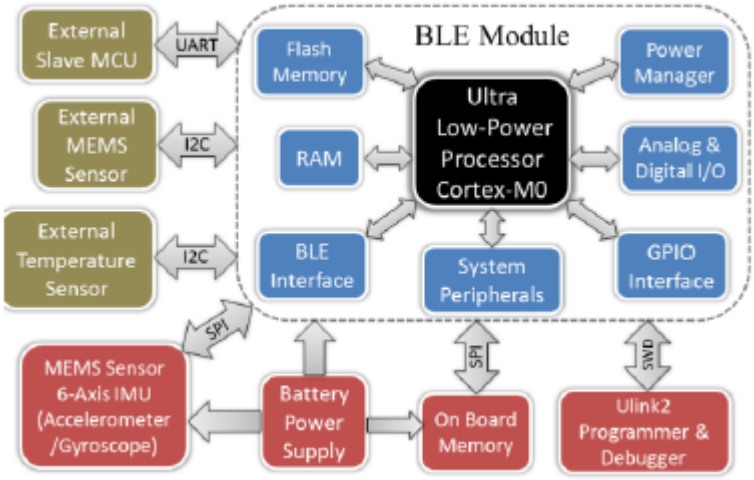


Figura 4 Diagrama a bloques del hardware utilizado para el sistema

Los acelerómetros son los encargados de obtener información acerca de la orientación del dispositivo con respecto al plano horizontal cuando la única

aceleración presente es la causada por la gravedad. Se usa un giroscopio para ayudar a la obtención de datos, así como una metodología de filtrado que permite evaluar anormalidades en el andar. El prototipo construido se muestra en la Figura 5.

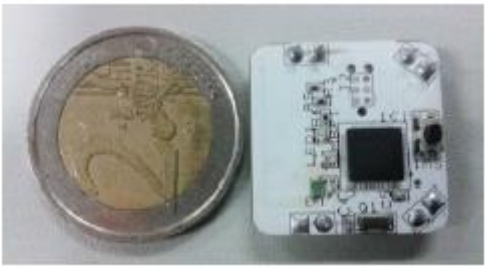


Figura 5 Comparativa de tamaño del dispositivo con una moneda de 2 euros

La gráfica que describe los datos que se obtienen y su relación con el sistema de adquisición se muestra en la Figura 6.

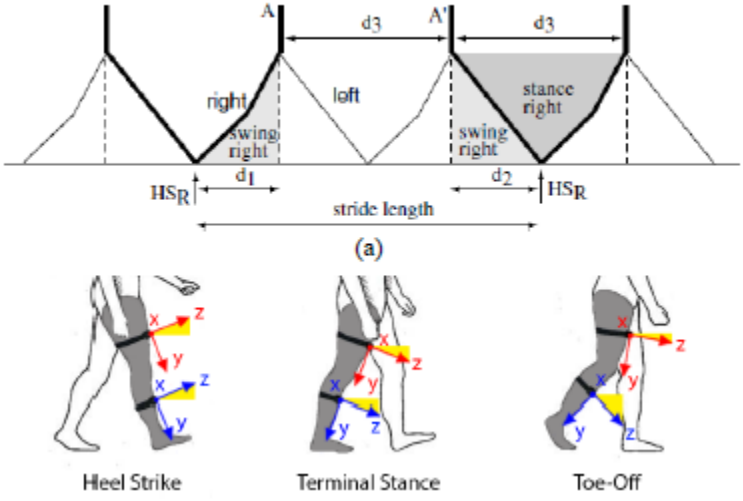


Figura 6 Gráfica para la interpretación de los datos obtenidos por el dispositivo

La validación experimental del dispositivo propuesto demostró su efectividad como una herramienta para medir los parámetros espaciales y temporales del andar en

grupos de control sanos durante diferentes perfiles de caminata (N. Margiotta, 2016).

En otra investigación, se propone un acercamiento diferente para evaluar cuantitativamente los movimientos de brazo y mano de pacientes con Parkinson, usando los sensores de una banda para brazo inalámbrica. El estudio incluyó 17 pacientes con Parkinson 16 controles. Se definió un conjunto de movimientos representativos bajo la supervisión de un especialista en desordenes de movimiento (Spasojević, 2017).

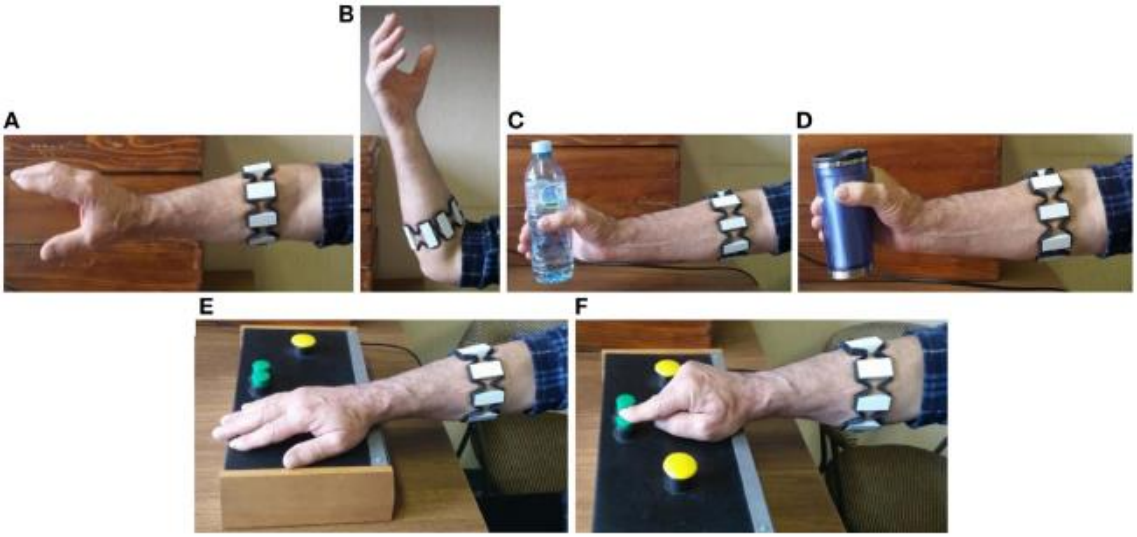


Figura 7 Movimientos propuestos para el estudio de movimiento de la mano

En la Figura 7 se muestran los movimientos propuestos para el estudio.

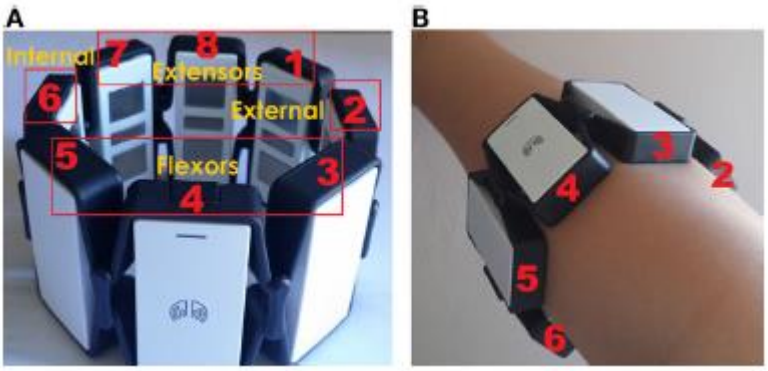


Figura 8 Dispositivo y su sugerencia de uso

Usando la tecnología de un reloj inteligente también es posible analizar y diagnosticar el temblor de cualquier otro padecimiento neurodegenerativo, así de plasma en la investigación de (Wile, 2014), en esta se explica que distinguir temblor re-emergente postural de la enfermedad de Parkinson del temblor esencial, puede ser difícil clínicamente y el uso de acelerómetros para ayudar al diagnóstico está restringido a laboratorios, por eso el uso de un reloj inteligente que es un dispositivo básico para cualquier persona es de gran practicidad. Para este método se usaron 41 pacientes. La investigación reveló que el reloj inteligente proveyó de una estimación concordante de la frecuencia del temblor, así como otros parámetros importantes que también fueron análogos a los métodos utilizados en laboratorios. La ventaja de este sistema es que no es invasivo y es fácil de usar, lo que se puede traducir en ser utilizado de forma rutinaria en clínicas o incluso en instalaciones en comunidades. La adquisición de datos puede ser completada en menos de 5 minutos.

1.3. JUSTIFICACIÓN

No es fácil para nadie enfrentar las enfermedades, sean o no graves, todas ellas nos quitan tranquilidad y tienen efectos sobre nuestro estado de ánimo, sin embargo, algunas enfermedades van más allá y afectan permanentemente nuestra calidad de vida. Tal es el caso del mal de Parkinson, una enfermedad neurodegenerativa que priva de una vida de calidad a todo aquel que la padece. Quienes sufren esta enfermedad normalmente experimentan un amplio margen de síntomas motores durante el curso de la enfermedad, los cuales afectan a cada persona en diferentes escalas. Claro que la forma en que estas personas perciben la enfermedad difiere contra la opinión de quienes los tratan, lo cual en ocasiones causa que no se pueda hacer un buen manejo de este padecimiento (Marios Politis, 2010). En un estudio llevado a cabo en 2010, 265 pacientes participaron clasificando sus tres padecimientos más problemáticos que afectan su calidad de vida, las 5 quejas más recurrentes fueron lentitud, temblor, rigidez, dolor y pérdida del olfato o

el gusto. Esta investigación revelo que conforme la enfermedad y la edad del paciente avanzan, cada vez los medicamentos resultan ser menos efectivos. Sería bueno poder asumir que la enfermedad afecta a un pequeño número de personas, sin embargo, los datos revelan lo contrario, en países industrializados se estima que la prevalencia del Parkinson es del 0.3% en la población en general, 1.0% en personas de más de 60 años y 3.0% en personas mayores a los 80 años. La edad es el mayor factor de riesgo de la enfermedad y ser hombre conforma un riesgo moderado. Algunos factores ambientales se han relacionado también con el riesgo de Parkinson, tales como lo son algunos pesticidas, el vivir en una zona rural, la exposición a manganeso, monóxido de carbón y otros gases. Datos como estos nos revelan algunos puntos importantes de la enfermedad, según las cifras vemos que no hay un gran porcentaje de personas que tengan el padecimiento siendo ellos ya ancianos, esto porque el Parkinson es muy a menudo mortal en una etapa temprana de la enfermedad, también podemos decir que todos estamos en algún grado en riesgo de contraer esta enfermedad, todo esto nos deja claro que la detección de esta enfermedad debería ser una actividad importante en el ámbito de la investigación (Roberta Balestrino, 2020). Detectar el Parkinson en una edad temprana puede hacer la diferencia, actualmente las modalidades para diagnosticar enfermedad de Parkinson están limitadas por el hecho de que se basan en la detección de problemas motrices notorios, cuando en ese punto de la enfermedad, más del 60% de todas las neuronas en partes específicas del ganglio basal ya se han perdido. Solo en EUA se cree que hay aproximadamente 849,000 personas que tienen este padecimiento, los costos anuales totales se han estimado en 11 billones, incluyendo 6.2 billones en costos directos. La mayor parte de este dinero es gastada en las etapas avanzadas de la enfermedad, cuando los síntomas son más severos, por lo que, desde un punto de vista meramente económico, cualquier técnica que pueda mantener la enfermedad en sus etapas más tempranas será sustancialmente beneficiosa a la hora de las limitantes que ponen normalmente los gastos. Ahora bien, desde la perspectiva de la calidad de vida del paciente, diagnosticar la enfermedad tempranamente es aún más importante, considerando que el Parkinson es una de las enfermedades crónicas que más dañan la calidad de vida de los

pacientes, siendo las etapas avanzadas las más severas (Pagán, 2012). Si bien no hay tratamientos que puedan curar el Parkinson si hay alternativas para su tratamiento, las cuales pueden mejorar bastante la calidad de vida de quienes lo padecen, un ejemplo de esto es el uso de la Nicotina, esta sustancia tiene el potencial de proteger contra el Parkinson. La nicotina puede ser utilizada para mejorar la cognición y el estado de alerta, un meta-estudio de 41 estudios controlados por placebo concluyó que la nicotina tuvo un buen efecto en las habilidades motoras, orientación, atención y memoria de trabajo. Todo esto es prometedor para todos los enfermos de Parkinson, sin embargo, una investigación más a fondo de los receptores de nicotina propone este tratamiento como una posibilidad de tratamiento neuro-protector en etapas tempranas de la enfermedad (Jan Aaseth, 2018)

CAPÍTULO II. MARCO TEÓRICO

2.1. PYTHON

Es un lenguaje de programación de alto nivel y de propósito general. Su diseño se enfatiza el uso de una estructura que sea fácil de leer para el ser humano, valiéndose de darle un valor a las sangrías (Kuhlman, 2012). Python es dinámicamente escrito, soporta múltiples paradigmas de programación, incluyendo programación, estructural, orientada a objeto y funcional. Normalmente se le describe como un lenguaje de programación “con pilas incluidas” debido a su comprensible librería estándar. Python está entre los lenguajes de programación más populares. (The State of Developer Ecosystem in 2020 Infographic, 2021) (PYPL PopularitY of Programming Language index, 2017)

Figura 9 Logo de Python

Su filosofía se centra en el documento “The zen of python”, que incluye aforismos como:

- Bello es mejor que feo
- Explicito es mejor que implícito
- Simple es mejor que complejo
- Complejo es mejor que complicado
- La legibilidad cuenta

En vez de construir toda su funcionalidad en su núcleo Python fue diseñado para ser ampliamente extensible a través de módulos. Esta modularidad lo ha hecho

particularmente popular como una forma de añadir interfaces programables a aplicaciones existentes. (Peters, 2018)

Python usa sangría de espacio en blanco, en vez de corchetes o keywords, para delimitar bloques. Un aumento en la sangría viene después de ciertas sentencias; un decremento en la sangría significa el fin del bloque actual. Por esto, la estructura visual del programa representa de manera precisa la estructura semántica. (Myths about indentation in Python, 2018) (Guttag, 2016)

Las sentencias de Python incluyen:

- La declaración de asignación usa un simple signo de igual “=”.
- La sentencia “if”, la cual ejecuta un bloque de código condicionalmente, junto con un “else” o un “elif”.
- La sentencia “for”, la cual itera sobre un objeto iterable.
- La sentencia “while”, la cual ejecuta un bloque de código mientras cierta condición predefinida sea verdad.
- La sentencia “try”, la cual permite que atrapar las excepciones que sean lanzadas en el bloque relacionado a la sentencia, para que de esta manera puedan se manejadas por las cláusulas de excepción.
- La sentencia “class”, la cual ejecuta un bloque de código y anexa su nombre local a una clase, para ser usada en programación orientada a objetos.
- La sentencia “def”, la cual define una función o un método.
- La sentencia “return”, usada para devolver un valor de una función
- La sentencia “import”, usada para importar módulos cuyas funciones o variables puedan ser usadas en el programa actual.

La gran librería estándar de Python provee herramientas para diversas tareas, y es comúnmente una de sus grandes fortalezas. Para aplicaciones en internet se soportan muchos formatos y protocolos estándar como MIME y HTTP. La librería estándar incluye módulos para crear interfaces de usuario gráficas, conectar con bases de datos relacionales, generar números seudo-aleatorios, etc. (Batista, 2020)

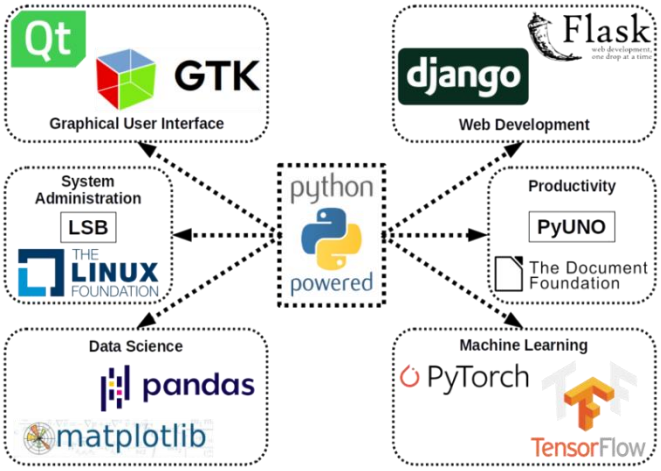


Figura 10 Ejemplos de algunas librerías que contiene Python

2.2. LABVIEW

LabVIEW es el acrónimo de Laboratory Virtual Instrument Engineering Workbench. Es un lenguaje y, a la vez, un entorno de programación gráfica en el que se pueden crear aplicaciones de una forma rápida y sencilla.

National Instruments, la empresa desarrolladora y propietaria de LabVIEW, se fundó en 1976 en Austin, Texas. Sus primeros productos eran dispositivos para el bus de instrumentación GPIB. En abril de 1983 comenzó el desarrollo de lo que sería su producto estrella: LabVIEW, que vería la luz en octubre de 1986 con el lanzamiento de LabVIEW 1.0 para Macintosh y en 1990 la versión 2. Para Windows habría que esperar hasta septiembre de 1992. (Vizcaíno, 2011)

LabVIEW puede ser usado para desempeñar actividades como:

- Adquirir datos desde instrumentos
- Procesar datos
- Analizar datos
- Controlar instrumentos y equipo

Esta herramienta hace posible traer información del mundo exterior dentro de una computadora, tomar decisiones basadas en los datos adquiridos y enviar los

resultados de nuevo al mundo exterior para controlar gran parte de equipos. (Larsen, 2011)

2.2.1. VI´s

Los programas de LabVIEW son llamados Vi's, que originalmente viene de virtual instrument.

Un VI consta de dos partes:

Panel frontal- Muestra los controles y representa la interfaz gráfica del VI. Un ejemplo de un VI lo vemos en la Figura 11.

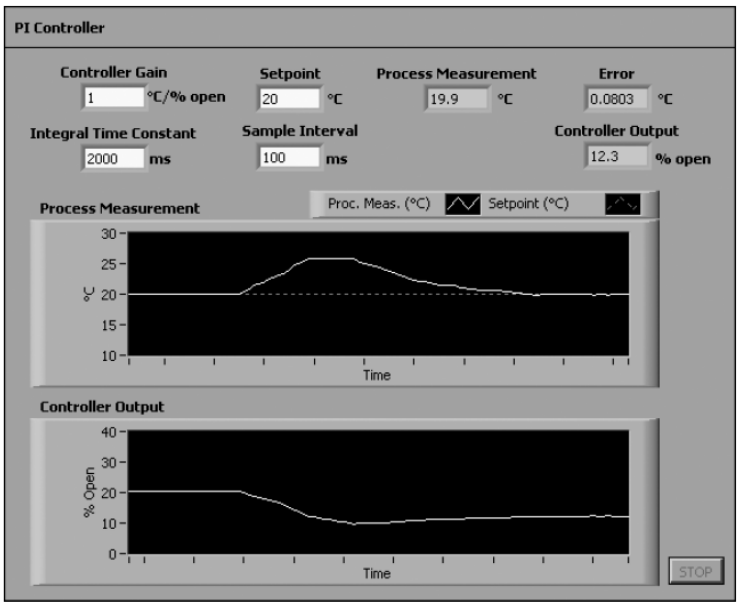


Figura 11 Ejemplo del panel frontal en LabVIEW

Diagrama de bloques- Contiene los elementos del programa que son conectados entre ellos para construir el programa gráfico. Un ejemplo de un diagrama a bloques lo vemos en la Figura 12.

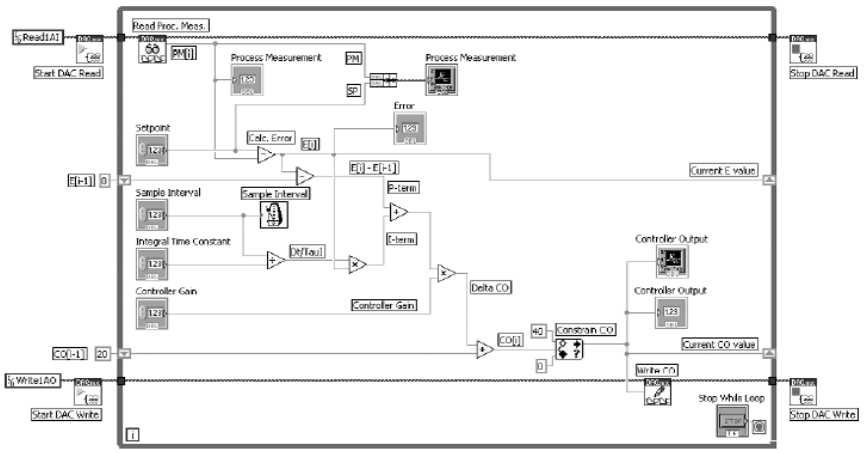


Figura 12 Ejemplo de un diagrama a bloques en LabVIEW

2.3. COMUNICACIÓN SERIAL

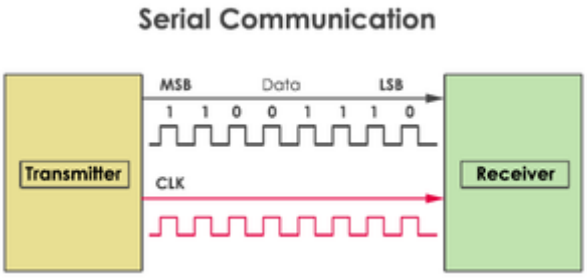


Figura 13 Esquema de comunicación serial

2.4. VOLTAJE

Partiendo de la idea de que en todo elemento eléctrico existen dos caminos por los cuales la corriente puede entrar o dejar el elemento. En base a la figura 14 supongamos que una corriente de tipo directo es enviada a la terminal A, a través del elemento y de vuelta por la terminal B. Asumamos también que empujar una carga a través del elemento supone requiere un gasto de energía. Entonces decimos que existe una diferencia de potencial (voltaje) entre las dos terminales. Por lo tanto, el voltaje a través de un par de terminales es una medida del trabajo requerido para mover carga a través del elemento. La unidad de voltaje es el Voltio, dónde 1 V es lo mismo que 1 J/C. (Hayt, 2007)

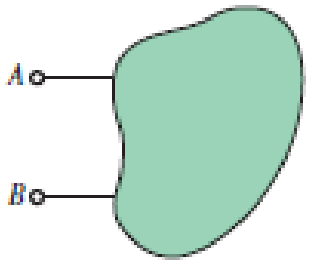


Figura 14 Representación de cuerpo sometido a un voltaje entre A y B

2.5. CORRIENTE

La rapidez con la que fluye la carga través de un conductor es llamada corriente eléctrica (Figura 15). La corriente eléctrica es causada debido al efecto que tiene el proporcionar una vía por la que los electrones puedan moverse entre dos elementos con cargas opuestas. La unidad de medida de la corriente eléctrica es el ampere. Un ampere (A) representa un flujo de carga con la rapidez de un Coulomb por segundo, al pasar por cualquier punto (Tippens, 2007).

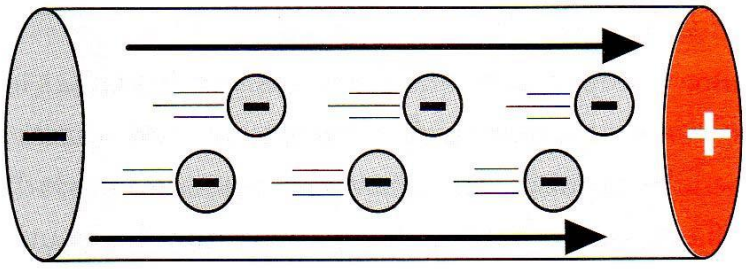


Figura 15 Representación de flujo de corriente a través de un conductor

2.6. RESISTENCIA

La resistencia se define como la oposición a que fluya la carga eléctrica. Todo material ofrece cierta oposición a que el flujo de carga eléctrica pase a través de ellos. Este parámetro es independiente del voltaje aplicado y de la corriente que pasa a través de ellos. El primero en estudiar cuantitativamente los efectos de la resistencia para limitar el flujo de carga fue Georg Simon Ohm, en 1826, el descubrió

que, para un resistor dado, a una temperatura particular, la corriente es directamente proporcional al voltaje aplicado. Así como la rapidez de flujo de agua entre dos puntos depende de la diferencia de altura que haya entre ambos, la rapidez de flujo de la carga eléctrica entre dos puntos depende de la diferencia de potencial que existe entre ellos. Proporcionalidad definida en la ley de Ohm (Figura 16). (Tippens, 2007)

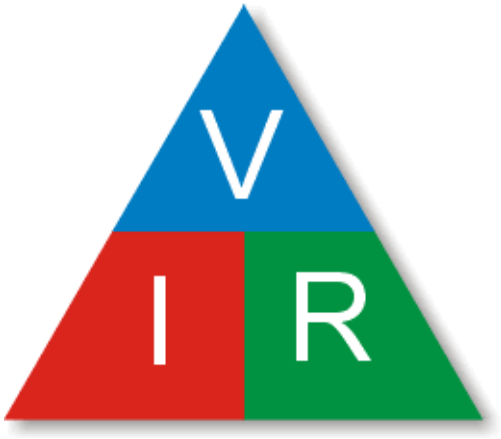


Figura 16 Triángulo de Ohm

2.7. IDE

Un entorno de desarrollo integrado o entorno de desarrollo interactivo, en inglés Integrated Development Environment (IDE), es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software. (Isidro Ramos Salavert, 2000) (José M. Troya)

Normalmente, un IDE consiste en un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDE tienen auto-completado inteligente de código (IntelliSense). Algunos IDE contienen un compilador, un intérprete, o ambos, tales como NetBeans y Eclipse; otros no, tales como SharpDevelop y Lazarus.

El límite entre un IDE y otras partes del entorno de desarrollo de software más amplio no está bien definido. Muchas veces, a los efectos de simplificar la construcción de la interfaz gráfica de usuario (GUI, por sus siglas en inglés), se integran un sistema controlador de versión y varias herramientas. Muchos IDE modernos también cuentan con un navegador de clases, un buscador de objetos y un diagrama de jerarquía de clases para su uso con el desarrollo de software orientado a objetos.

Los IDE están diseñados para maximizar la productividad del programador proporcionando componentes muy unidos con interfaces de usuario similares. Los IDE presentan un único programa en el que se lleva a cabo todo el desarrollo. Generalmente, este programa suele ofrecer muchas características para la creación, modificación, compilación, implementación y depuración de software. Esto contrasta con el desarrollo de software utilizando herramientas no relacionadas, como Vi, GNU Compiler Collection (GCC) o Make.

Uno de los propósitos de los IDE es reducir la configuración necesaria para reconstruir múltiples utilidades de desarrollo, en vez de proveer el mismo set de servicios como una unidad cohesiva. Reduciendo ese tiempo de ajustes se puede incrementar la productividad de desarrollo, en casos donde aprender a usar un IDE es más rápido que integrar manualmente todas las herramientas por separado.

Una mejor integración de todos los procesos de desarrollo hace posible mejorar la productividad en general, más que únicamente ayudando con los ajustes de configuración. Por ejemplo, el código puede ser continuamente armado mientras es editado, previendo retroalimentación instantánea, como cuando hay errores de sintaxis. Esto puede ayudar a aprender un nuevo lenguaje de programación de manera más rápida, así como sus librerías asociadas.

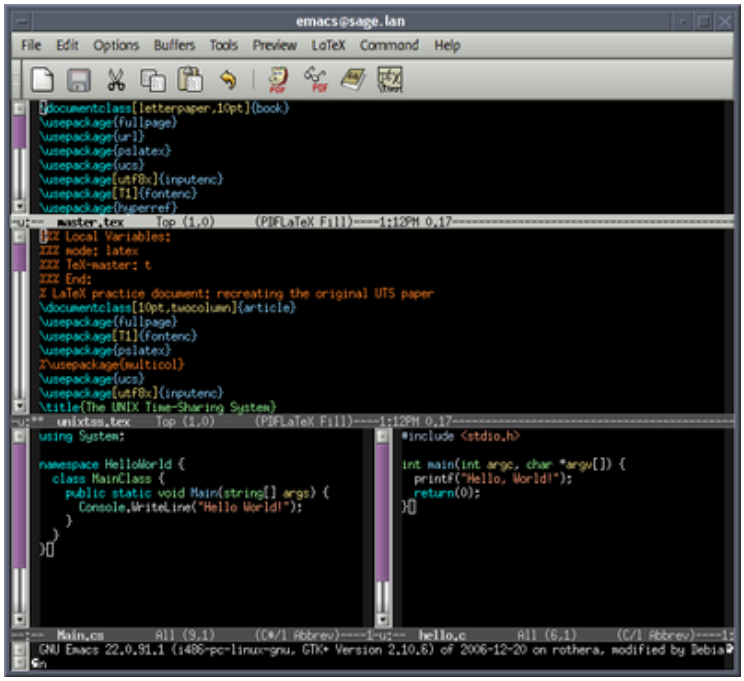


Figura 17 GNU Emacs, un editor normalmente utilizado como IDE en sistemas tipo UNIX

Algunos IDE están dedicados específicamente a un lenguaje de programación, permitiendo que las características sean lo más cercanas al paradigma de programación de dicho lenguaje. Por otro lado, existen muchos IDE de múltiples lenguajes tales como Eclipse, ActiveState Komodo, IntelliJ IDEA, MyEclipse, Oracle JDeveloper, NetBeans, Codenvy y Microsoft Visual Studio. Xcode, Xojo y Delphi están dedicados a un lenguaje cerrado o a un tipo de ajustes de tipos de lenguajes de programación.

2.8. BAUDIO

Es el número de cambios de estado por segundo de una señal discreta de valores enviada por el dispositivo UART, considerando que una señal discreta es aquella que solo puede adoptar valores finitos. Este concepto también puede tomarse como una unidad de velocidad de transmisión de pulsos, lo que en el caso de que fuesen baudios binarios, un baudio equivaldría a un bit. Esta unidad debe su nombre al

ingeniero francés Jean Maurice Baudot, quién fue el primero en realizar mediciones de la velocidad de las transmisiones telegráficas. (Collazo, 1991)

2.9. PUERTO COM

Un puerto serial es la interfaz de una computadora que transmite datos un bit a la vez. Comúnmente se utiliza el término puerto serial para referirse a los puertos que utilizan el protocolo asíncrono de comunicación. Estos puertos incluyen los puertos RS-232 en una PC y muchos puertos en sistemas embebidos. La mayoría de los puertos seriales son bidireccionales porque envían y reciben datos. (Axelson, 2007)

2.10. LA NORMA RS-232

RS-232 es una interfaz apropiada para muchas tareas básicas de comunicación entre dos computadoras. Este protocolo de comunicación está diseñado para manejar las comunicaciones entre dos dispositivos con un límite de distancia de 80 a 130 pies, dependiendo de la cantidad de bits y el tipo de cable. Típicamente, RS 232 es una interfaz de comunicación que cumple con mucho del estándar TIA-232-F.

Las líneas esenciales para la comunicación bidireccional RS 232 son las siguientes tres:

TX Esta lleva información del DTE (Data Terminal Equipment) al DCE (Data Circuit terminating Equipment), algunas veces recibe el nombre de TD o TDX

RX Esta lleva información del DCE al DTE, algunas veces recibe el nombre de RD o RDX.

SG Señal de tierra

Los niveles lógicos de esta interfaz están definidos como voltajes positivos y negativos en vez de solo voltajes positivos de la lógica TTL y CMOS. En la salida de datos, el cero lógico está definido como igual o más positivo que +5v, y el uno lógico está definido como igual o más negativo que -5v. (Axelson, 2007)

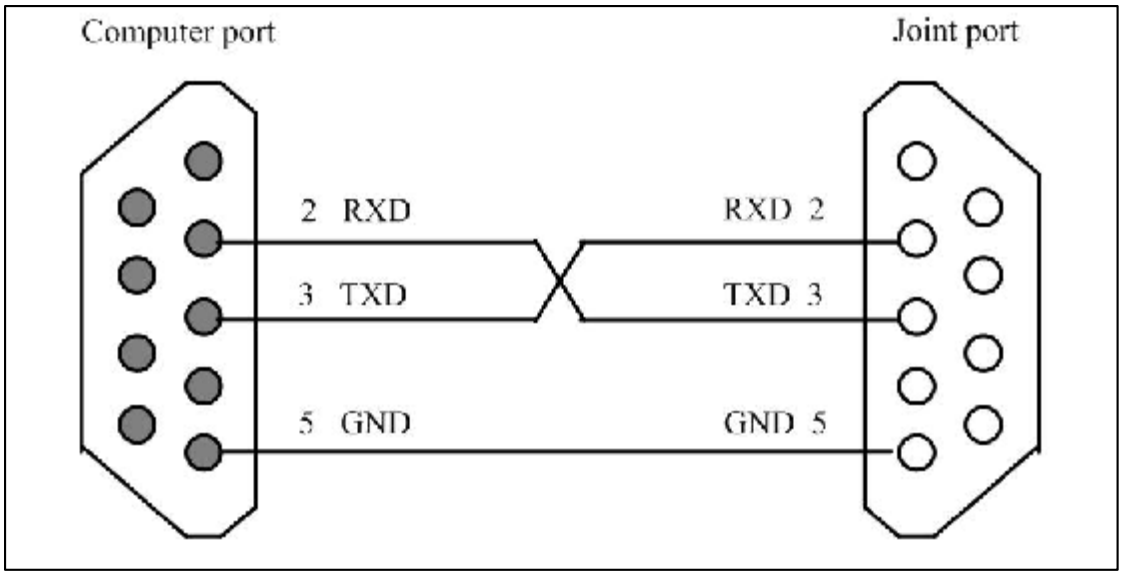


Figura 18 Conexión de puerto Rs232

2.11. CINEMÁTICA

La Cinemática es la parte de la Física que estudia el movimiento de los cuerpos, denominados en sentido general, como partículas. Así, se puede definir la 'partícula' como todo cuerpo que posee una posición, sin considerar sus dimensiones. En otras palabras, el desplazamiento o movimiento del cuerpo tiene mucha más importancia que sus dimensiones. (Olmedo, 2012)

En el estudio de la robótica nos preocupamos constantemente por la ubicación de los objetos en el espacio tridimensional. Estos objetos son los vínculos del manipulador, las piezas y herramientas con las que trabaja y los demás objetos en el entorno del manipulador. A un nivel básico pero importante, estos objetos se describen mediante sólo dos atributos: posición y orientación. Naturalmente, un tema de interés inmediato es la manera en la que representamos estas cantidades y las manipulamos matemáticamente. Para poder describir la posición y orientación de un cuerpo en el espacio, siempre adjuntamos rígidamente un sistema de coordenadas, o trama, al objeto. Después describimos la posición y orientación de esta trama con respecto a algún sistema de coordenadas de referencia. (*Vea la fi**gur**a 19*). Cualquier trama puede servir como sistema de referencia dentro del cual

se pueda expresar la posición y orientación de un cuerpo, por lo que a menudo pensamos en transformar o cambiar la descripción de estos atributos de un cuerpo de una trama a otra. Es muy útil desarrollar buenas habilidades respecto a la descripción de la posición y la rotación de cuerpos rígidos, incluso en campos externos a la robótica.

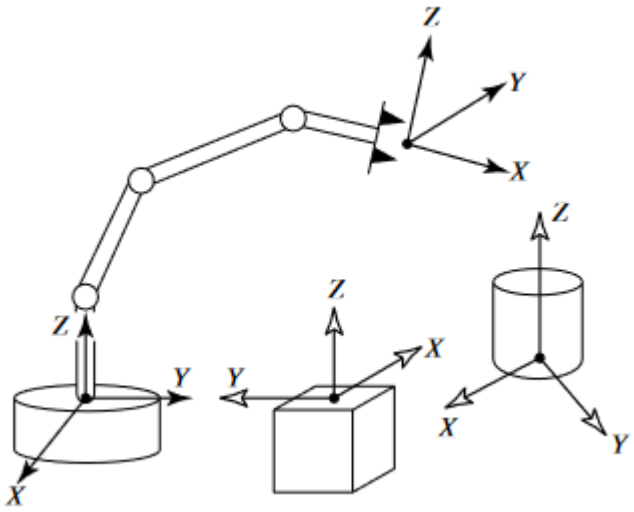


Figura 19 Diagrama para ejemplificar un sistema de coordenadas de referencia

2.11.1. CINEMÁTICA DIRECTA

El estudio de la cinemática de los manipuladores se refiere a todas las propiedades del movimiento, las geométricas y las basadas en tiempo. Los manipuladores consisten de vínculos casi rígidos, los cuales están conectados por articulaciones que permiten el movimiento relativo de los vínculos adyacentes. Estas articulaciones generalmente se instrumentan con sensores de posición, los cuales permiten medir la posición relativa de los vínculos adyacentes. En el caso de las articulaciones giratorias o angulares, estos desplazamientos se conocen como ángulos articulados. Algunos manipuladores contienen articulaciones deslizantes (o prismáticas), en las que el desplazamiento relativo entre los vínculos es una translación, algunas veces llamada desplazamiento de articulación. El número de grados de libertad que posee un manipulador es el número de variables de posición

independientes que tendrían que especificarse para poder localizar todas las piezas del mecanismo. Éste es un término general que se utiliza para cualquier mecanismo. Por ejemplo, un vínculo de cuatro barras sólo tiene un grado de libertad (incluso aunque haya tres miembros móviles). En el caso de los robots industriales comunes, como un manipulador es generalmente una cadena cinemática abierta y como la posición de cada articulación se define generalmente con una sola variable, el número de articulaciones es igual al número de grados de libertad. En el extremo libre de la cadena de vínculos que conforman el manipulador se encuentra el efector final. Dependiendo de la aplicación que se va a dar al robot, el efector final podría ser una pinza, un soplete de soldadura, un electroimán o cualquier otro dispositivo. Generalmente presentamos la posición del manipulador proporcionando una descripción de la trama de la herramienta, la cual está unida al efector final, relativa a la trama base, que está unida a la base fija del manipulador. (Vea **la figura** 1.6). Un problema muy básico en el estudio de la manipulación mecánica se conoce como cinemática directa, que es el problema geométrico estático de calcular la posición y orientación del efector final del manipulador. Específicamente, dado un conjunto de ángulos articulares, el problema de la cinemática directa es calcular la posición y orientación de la trama de la herramienta relativa a la trama base. Imaginemos que es como cambiar la representación de la posición del manipulador: de una descripción en el espacio de la articulación a una descripción en el espacio cartesiano. (Craig, 2006)

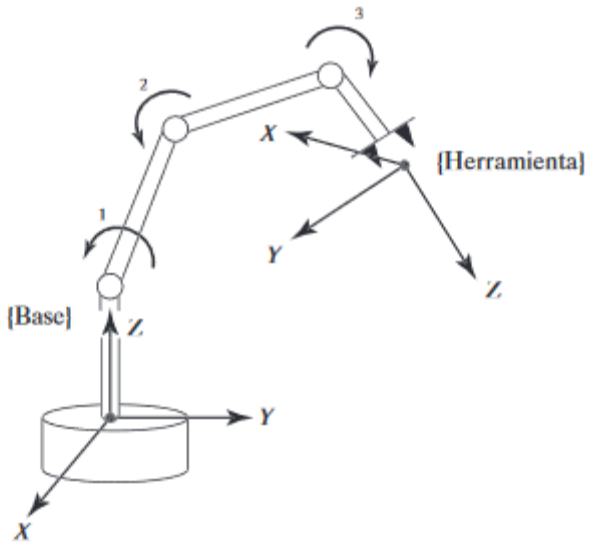


Figura 20 Posición de un manipulador con respecto a un sistema de referencia

2.11.2. CINEMÁTICA INVERSA

Este concepto se plantea de la siguiente manera: es la obtención de todos los conjuntos posibles de ángulos articulares que podrían utilizarse para obtener la posición y orientación del efector final del manipulador, los cuales son parámetros ya dados. (Vea la figura 21). Éste es un problema fundamental en el uso práctico de los manipuladores. Éste es un problema geométrico algo complicado, que se resuelve de manera rutinaria miles de veces diariamente en el sistema humano y en otros sistemas biológicos. En el caso de un sistema artificial como un robot, necesitamos crear un algoritmo en la computadora de control que pueda realizar este cálculo. En ciertos casos, la solución de este problema es el elemento más importante en un sistema manipulador. Podemos pensar en este problema como en una asignación de “ubicaciones” en el espacio cartesiano 3D, a “ubicaciones” en el espacio de articulaciones internas del robot.

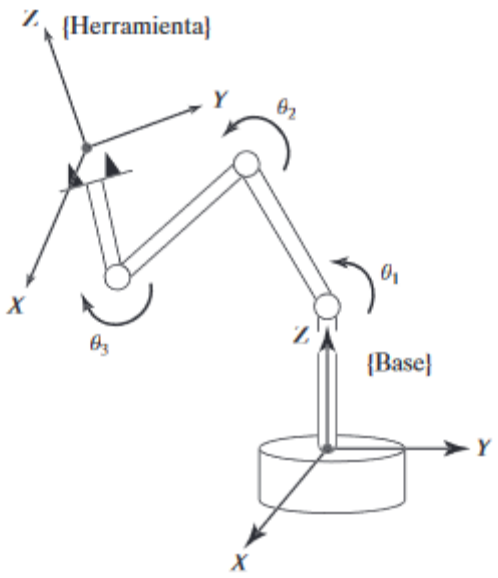


Figura 21 Diagrama para cinemática inversa de un robot manipulador

Esta necesidad surge naturalmente siempre que se especifica un objetivo en coordenadas de espacio 3D externas. Algunos de los primeros robots carecían de este algoritmo; simplemente se desplazaban (algunas veces manualmente) hacia las ubicaciones deseadas, que después se registraban como un conjunto de valores de articulación (es decir, como una ubicación en el espacio de la articulación) para su posterior reproducción. Obviamente, si el robot se utiliza solamente en el modo de registrar y reproducir las ubicaciones y los movimientos de las articulaciones, no es necesario ningún algoritmo que relacione el espacio de la articulación con el espacio cartesiano. No obstante, actualmente es raro encontrar un robot industrial que carezca de este algoritmo básico de cinemática inversa. El problema de la cinemática inversa no es tan simple como el de la cinemática directa. Debido a que las ecuaciones cinemáticas son no lineales, su solución no es siempre sencilla (o incluso posible) en una forma cerrada. Además, surgen preguntas sobre si existe una solución o existen múltiples soluciones. El estudio de estas cuestiones hace que apreciemos lo que la mente humana y el sistema nervioso logran realizar cuando, al parecer inconscientemente, movemos y manipulamos objetos con nuestros brazos y manos. La existencia o inexistencia de una solución cinemática

define el espacio de trabajo de un manipulador dado. La falta de una solución significa que el manipulador no puede obtener la posición y orientación deseadas, ya que se encuentran fuera del espacio de trabajo del manipulador. (Craig, 2006)

2.12. ARDUINO

Arduino es un sistema de programación de los lenguajes C/C++, con plataformas electrónicas que utilizan microcontroladores para formar sistemas empotrados especializados en la automatización de procesos, con un amplio espectro de aplicaciones en ciencias exactas y en ingeniería. Este tipo de plataformas se ha popularizado en todo el mundo, gracias al desempeño, costo y arquitectura abierta de los modelos; esto último, es la característica principal que lo hace atractivo para desarrollar aplicaciones de automatización, ya que admite adaptabilidad para automatizar cualquier proceso físico. Otra ventaja es que tiene software libre, así como el acceso y adecuación del código fuente y de la electrónica de sus tarjetas.

Figura 22 Logo de Arduino

El sistema Arduino tiene un ambiente de programación (IDE) con herramientas integradas para editar, compilar y descargar sketches a las tarjetas electrónicas. Además, debido a su lenguaje de programación y las características tecnológicas de la plataforma electrónica es muy fácil expandirlo a otros paquetes de programación como MATLAB y LabVIEW.

Arduino gira en torno de la familia de controladores ATMEL y por medio de un entorno de programación, con las herramientas necesarias integradas para diseñar

y desarrollar aplicaciones de manera sencilla en los lenguajes C y C++, representa un sistema empotrado de propósito específico para automatizar procesos físicos. Puede ser usado para monitorear una enorme variedad de sensores analógicos y digitales, así como para controlar servomotores de corriente directa y alterna, a pasos, además de llevar a cabo aplicaciones con robots móviles e industriales, comunicación Wi-Fi, Bluetooth y Ethernet. (Fernando Reyes Cortés, 2015)

2.12.1. INTERRUPCIONES

Las interrupciones son recursos o mecanismos del microcontrolador para responder a eventos, permitiendo suspender temporalmente el programa principal, para ejecutar una subrutina de servicio de interrupción (ISR por sus siglas en inglés Interrupt Service Routines); una vez terminada dicha subrutina, se reanuda la ejecución del programa principal.

Las interrupciones se generan cuando dispositivos periféricos conectados a la tarjeta electrónica solicitan enviar información al microcontrolador, esto puede ser de manera asíncrona. También el proceso de interrupción se puede generar de manera periódica, es decir por medio de una señal digital (por ejemplo, de un milisegundo de periodo) conectada a un pin específico del microcontrolador (INT0 o INT1) se puede atender tareas determinadas como adquisición de datos, monitoreo de sensores cálculos numéricos, envío de comandos al robot, etc.

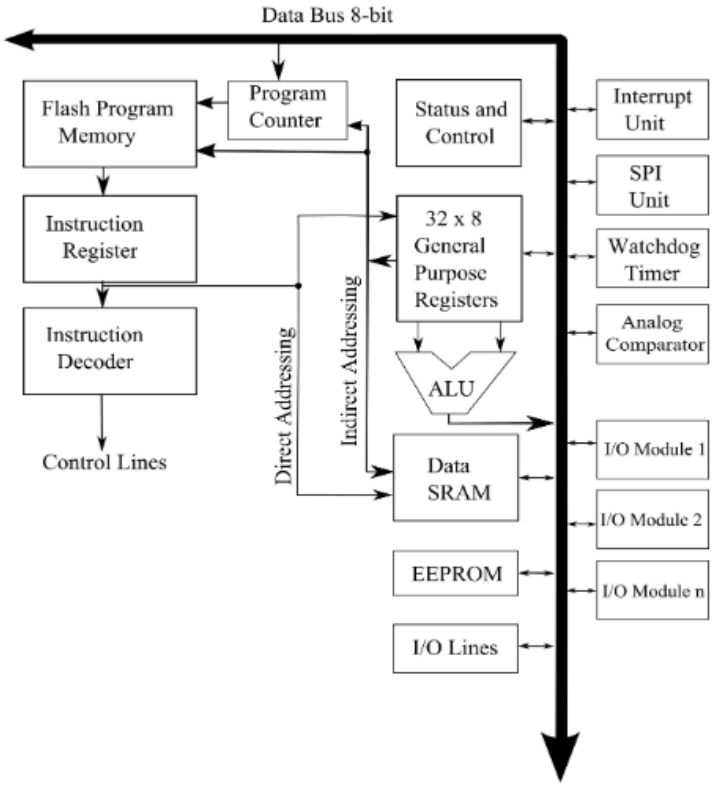


Figura 23 Diagrama a bloques de un microcontrolador

En los microcontroladores Atmega hay dos tipos de interrupciones, el primer tipo corresponde a eventos externos que generan un estado lógico (cambio de voltaje) y también por transición como un pulso electrónico de disparo (triggered), a transición se detecta por flanco de subida en la señal periódica (de bajo hacia alto, es decir: LOW → HIGH) o por flanco de caída (HIGH → LOW); esto habilita una bandera de interrupción (interrupt flag), dependiendo de la prioridad de interrupción, el contador del programa (program counter) toma la dirección de memoria de la tabla de vectores y salta a la localidad de memoria correspondiente donde se encuentra la rutina de la interrupción solicitada (interrupt handlingroutine).

De manera automática, por hardware se limpia la bandera de interrupción (interrupt flag); también se puede limpiar las banderas de interrupción por software, ya que tienen asociados sus respectivos bits de habilitación en el registro de estado (status register).

Similarmente, si más solicitudes de interrupción ocurren mientras se encuentra en proceso alguna interrupción, permanecerán en espera por orden de prioridad. Cuando el contador de programa sale de una interrupción, retornará al programa principal y ejecutará una o más instrucciones antes de atender alguna interrupción pendiente.

El segundo tipo de interrupciones corresponden a las interrupciones que pueden ser cambiadas o reasignadas por software en los pines del microcontrolador, estas interrupciones no necesariamente tienen banderas de interrupción.

2.12.2. TARJETAS

Arduino uno

La placa tiene 7 pines de alimentación, 14 pines digitales y 6 pines analógicos programables con el Arduino IDE (Entorno de desarrollo integrado) a través de un cable USB. Puede ser alimentado por el cable USB o por una batería externa de 9 voltios, aunque acepta voltajes entre 7 y 20 voltios. Es el buque insignia de Arduino ya que es la placa más popular, la que todo el mundo utiliza para iniciarse y la más sencilla de utilizar. Es el punto de partida de muchos entusiastas de la programación electrónica.

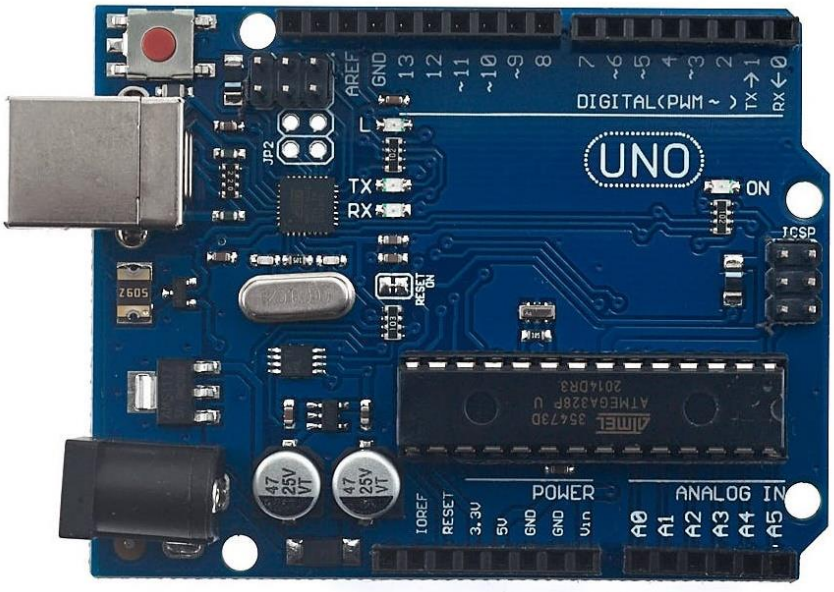


Figura 24 Tarjeta Arduino Uno

Arduino Mega

Arduino Mega es una tarjeta de desarrollo open-source construida con un microcontrolador modelo Atmega2560 que posee pines de entradas y salidas (E/S), analógicas y digitales. Esta tarjeta es programada en un entorno de desarrollo que implementa el lenguaje Processing/Wiring. Arduino puede utilizarse en el desarrollo de objetos interactivos autónomos o puede comunicarse a un PC a través del puerto serial (conversión con USB) utilizando lenguajes como Flash, Processing, MaxMSP, etc. Las posibilidades de realizar desarrollos basados en Arduino tienen como límite la imaginación.

El Arduino Mega tiene 54 pines de entradas/salidas digitales (14 de las cuales pueden ser utilizadas como salidas PWM), 16 entradas análogas, 4 UARTs (puertos serial por hardware), cristal oscilador de 16MHz, conexión USB, jack de alimentación, conector ICSP y botón de reset. Arduino Mega incorpora todo lo necesario para que el microcontrolador trabaje; simplemente conéctalo a tu PC por medio de un cable USB o con una fuente de alimentación externa (9 hasta 12VDC). El Arduino Mega es compatible con la mayoría de los shields diseñados para Arduino Duemilanove, diecimila o UNO.

Esta nueva versión de Arduino Mega 2560 adicionalmente a todas las características de su sucesor utiliza un microcontrolador ATMega8U2 en vez del circuito integrado FTDI. Esto permite mayores velocidades de transmisión por su puerto USB y no requiere drivers para Linux o MAC (archivo inf es necesario para Windows) además ahora cuenta con la capacidad de ser reconocido por el PC como un teclado, mouse, joystick, etc. (Arduino Official Page, 2022)

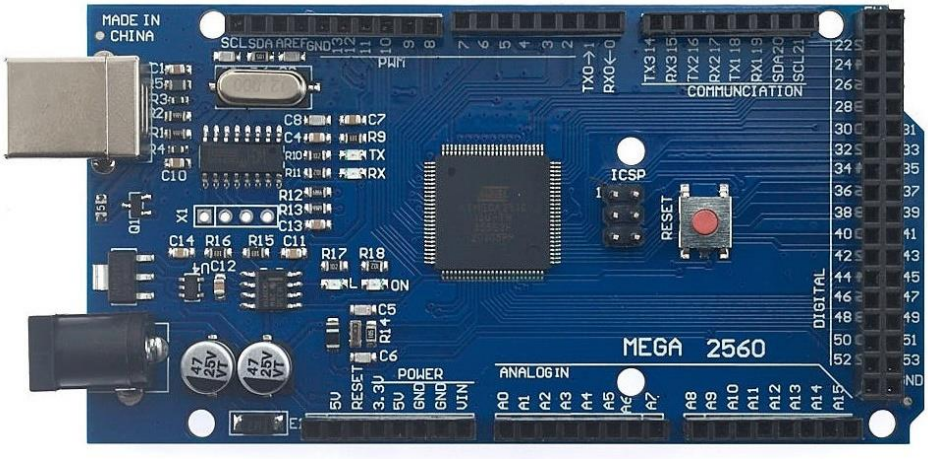


Figura 25 Tarjeta Arduino Mega

Arduino Esplora

La placa utiliza un microcontrolador Atmega32U4 AVR con oscilador de cristal de 16 MHz y una conexión micro USB capaz de actuar como un dispositivo cliente USB, como un mouse o un teclado.

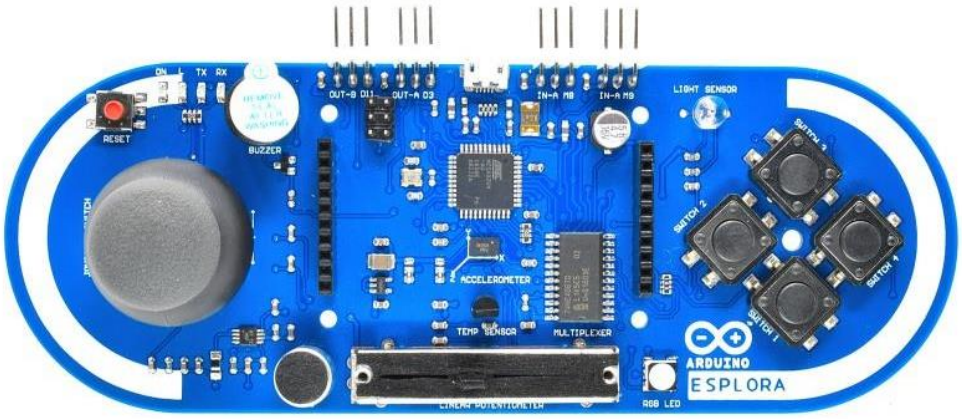


Figura 26 Tarjeta Arduino Esplora

Arduino Leonardo

La placa tiene 20 pines de entrada/salida digital (de los cuales 7 se pueden usar como salidas PWM y 12 como entradas analógicas), un oscilador de cristal de 16

MHz, una conexión micro USB, un conector de alimentación, un encabezado ICSP y un botón de reinicio.

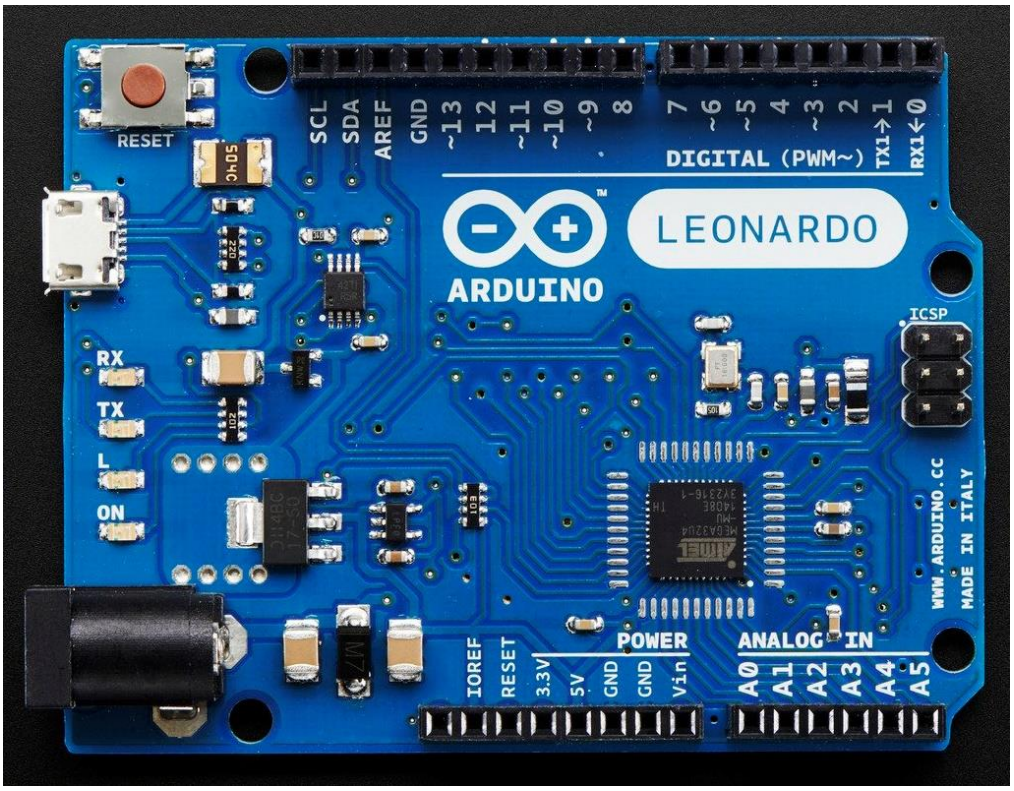


Figura 27 Tarjeta Arduino Leonardo

Arduino nano

Es una placa de desarrollo de tamaño compacto, completa y compatible con protoboards, basada en el microcontrolador ATmega328P. Tiene 14 pines de entrada/salida digital (de los cuales 6 pueden ser usando con PWM), 6 entradas analógicas, un cristal de 16Mhz, conexión Mini-USB, terminales para conexión ICSP y un botón de reseteo.

Posee las mismas capacidades que un Arduino UNO, tanto en potencia del microcontrolador como en conectividad, solo se ve recortado en su conector USB, conector jack de alimentación y los pines cambia un formato de pines header.

Figura 28 Tarjeta Arduino Nano

Arduino Due.

Arduino Due es la primera tarjeta de desarrollo construida con un poderoso microcontrolador de 32 bit CortexM3 ARM el cual puede ser programado mediante el IDE de Arduino. Incrementa el poder de procesamiento de tus proyectos Arduino manteniendo una compatibilidad en el lenguaje de programación y otorgándote una migración de tarjeta en cuestión de minutos.

El Arduino Due posee 54 pines digitales de entrada y salida (de los cuales 12 pueden ser usados como salidas PWM), 12 entradas análogas, 2 salidas análogas, 4 UART (puertas seriales por hardware), cristal oscilador de 84MHz, una conexión compatible con USB-OTG, 2 TWI, Jack de poder, conexión JTAG tambien un botón reset y un botón borrar. Además, tiene otras geniales funcionalidades como audio, DMA, una librería experimental para multitareas y más.

Para realizar una compilación de código de programación y que sea compatible con esta tarjeta, necesitaras utilizar una versión superior al IDE de Arduino 1.5.0.

De acuerdo a las limitaciones de voltaje de sistema impuestas por el Atmel SAM3X8E, la mayoría de los Shield de Arduino que funcionan con 5V no funcionarán correctamente con la tarjeta Arduino Due. Ten cuidado cuando conectes dispositivos a esta tarjeta.

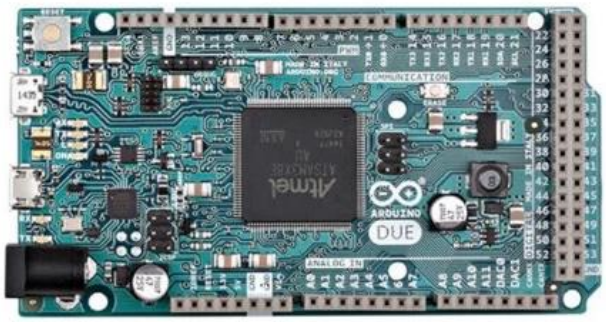


Figura 29 Tarjeta Arduino Due

2.13. CODIFICADOR ROTATORIO

Un codificador rotatorio, también llamado codificador del eje o generador de pulsos, suele ser un dispositivo electromecánico usado para convertir la posición angular de un eje a un código digital, lo que lo convierte en una clase de transductor. Estos dispositivos se utilizan en robótica, en lentes fotográficos de última generación, en dispositivos de entrada de ordenador (tales como el ratón y el trackball), y en plataformas de radar rotatorias. Hay dos tipos principales: absoluto e incremental (relativo).

Figura 30 Encóder rotatorio

Absoluto

Un codificador absoluto mantiene información de la posición incluso cuando la alimentación del mismo se desconecta. La posición estará disponible inmediatamente después de energizarlo de nuevo. La relación entre el valor del

codificador y la posición física de la maquinaria controlada se define al ensamblarse, el sistema no requiere de un punto de calibración al cual volver para mantener la precisión en la posición.

Incremental

Un codificador incremental reportara inmediatamente de cambios en la posición, pero no guarda un registro de la posición absoluta, por lo cual el sistema mecánico monitoreado por el codificador debe ser puesto en una posición de calibración al iniciar su funcionamiento (McMillan, 1999).

2.14. MOTOR CD

El motor de corriente continua, denominado también motor de corriente directa, motor CC o motor DC (por las iniciales en inglés: direct current), es una máquina que convierte energía eléctrica en energía mecánica, provocando un movimiento rotatorio, gracias a la acción de un campo magnético.

Los componentes de un motor de corriente continua se dividen en dos partes:

- Estator: parte que da soporte mecánico al aparato y contiene los polos de la máquina, que pueden ser devanados de hilo de cobre sobre un núcleo de hierro o imanes permanentes.

- Rotor: es un componente generalmente de forma cilíndrica, también devanado y con núcleo, alimentado con corriente continua a través del colector formado por delgas. Las delgas se fabrican generalmente de cobre y están en contacto alternante con las escobillas fijas.

El principal inconveniente que tienen estos motores es el mantenimiento costoso y laborioso, debido principalmente al desgaste que sufren las escobillas al entrar en contacto con las delgas. Las escobillas de los motores de baja potencia se fabrican de grafito. Por otro lado, los que requieren corrientes elevadas como los motores de arranque de los vehículos, se fabrican con una aleación de grafito y metal.

Algunas aplicaciones especiales de estos motores son: los motores lineales, servomotores, motores paso a paso o cuando ejercen tracción sobre un riel. Además, existen motores de CC sin escobillas (brushless en inglés) utilizados en el aeromodelismo por su bajo par motor y su gran velocidad.

Principio de funcionamiento

El principio de funcionamiento básico de un motor de CC se explica a partir del caso de una espira de material conductor inmersa en un campo magnético, a la cual se le aplica una diferencia de potencial (o voltaje) entre sus extremos, de forma que a través de la misma circula una corriente I.

Para este caso, la espira constituye el rotor del motor y los imanes que producen el campo magnético constituyen el estator.

Entonces, dado que cuando un conductor, por el que pasa una corriente eléctrica, se encuentra inmerso en un campo magnético, este experimenta una fuerza según la Ley de Lorentz. Dicha fuerza, denominada Fuerza de Lorentz, es perpendicular al plano formado por el campo magnético y la corriente, y su magnitud está dada por:

$$F = B * L * I * sen(\phi) \qquad (1)$$

F: $Fuerza\ en\ newtons$

I: $Intensidad\ que\ recorre\ el\ conductor\ en\ amperios$

L: $Longitud\ del\ conductor\ en\ metros$

B: $Densidad\ de\ campo\ magnético\ o\ densidad\ de\ flujo\ tesla$

Φ: $Ángulo\ que\ forma\ I\ con\ B$

La espira experimenta fuerzas en sus dos segmentos laterales, como puede observarse en la Figura 31. Sin embargo, en el resto de los segmentos que

interactúan con el campo magnético, las fuerzas experimentadas se cancelan entre sí.

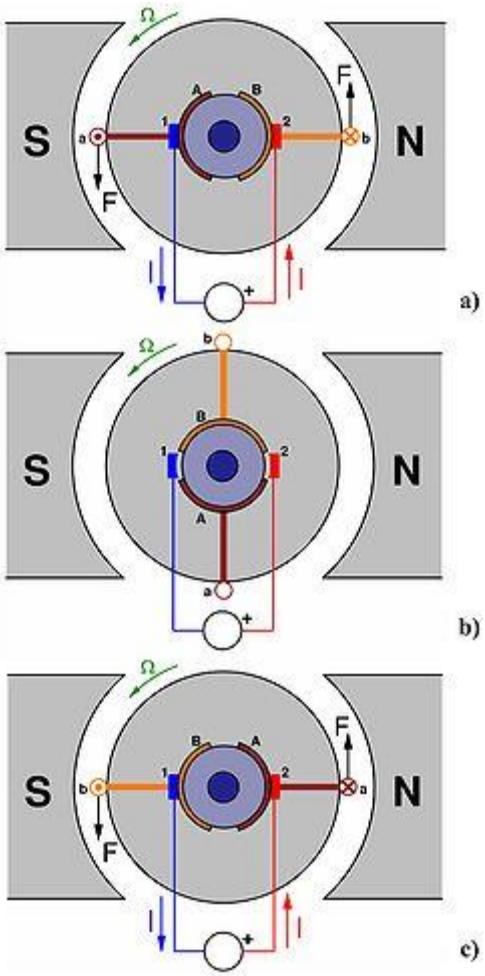


Figura 31 Diagrama de las fuerzas experimentadas por la espira en un motor CD

Como puede observarse, las fuerzas en los segmentos laterales tienen una dirección y sentido tales que producen el giro del rotor. Sin embargo, existe un punto en el que el cambio de la posición de las espiras produce que la misma fuerza se oponga a continuar con el movimiento. Es en este punto en el que se cambia o alterna la polaridad del voltaje aplicado, lo cual cambia el sentido de la fuerza y, por ende, vuelve a impulsar el giro del rotor.

Es posible que el rotor supere este punto, en el que no se aplican fuerzas, debido a su inercia. (Young, 2009) (Guru Bhag, 2001) (Gottlieb, 1994)

2.15. PUENTE H

Un puente H es un circuito electrónico el cual cambia la polaridad del voltaje aplicado a una carga. Estos circuitos son utilizados a menudo en robótica y otras aplicaciones que requieran mover un motor DC hacia ambos sentidos. El nombre se deriva de su la representación común de su diagrama esquemático, el cual tiene cuatro elementos conmutadores configurados en cada una de las ramas de una figura con la forma de la letra H, con la carga conectada en la barra vertical de en medio de ella. (Williams, 2002)

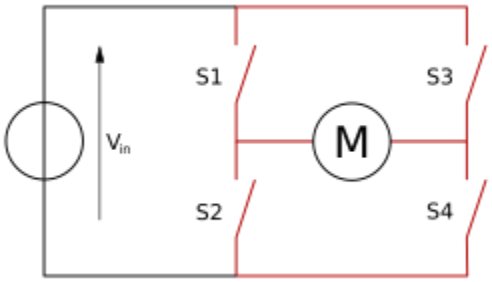


Figura 32 Esquemático de un puente H

2.16. PWM

La modulación por ancho de pulsos es un método para reducir la potencia promedio entregada por una señal eléctrica, al cortarla en partes discretas. El valor promedio de voltaje entregado a la carga es controlado al conmutar un interruptor entre el suministro de voltaje y la carga rápidamente. Entre más tiempo pase el switch encendido en comparación con el tiempo que pasa apagado, más potencia se estará entregando a la carga. PWM es particularmente adecuado para controlar cargas como motores, los cuales no son afectados por la conmutación discreta de su suministro, debido a que su inercia causa que el rotor responda lentamente. La frecuencia del PWM debe ser lo suficientemente grande como para no afectar la

carga, lo que es igual que decir que la forma de onda resultante debe ser percibida por la carga como una onda lo más suave posible

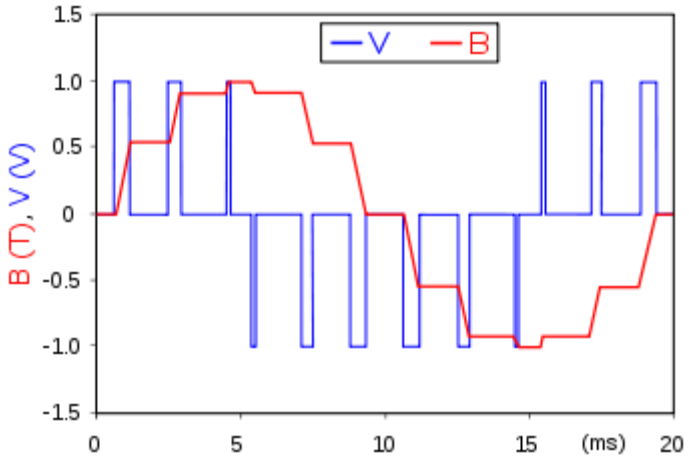


Figura 33 Gráfica de la salida experimentada por Una carga ante un PWM

Ciclo de trabajo

El término ciclo de trabajo describe la proporción del tiempo de “encendido” en comparación con el periodo de la señal. Un ciclo de trabajo bajo resultará en poca potencia, debido a que la alimentación está cortada la mayor parte del tiempo. El ciclo de trabajo se expresa en porcentajes, siendo 100% un PWM permanentemente en “encendido”.

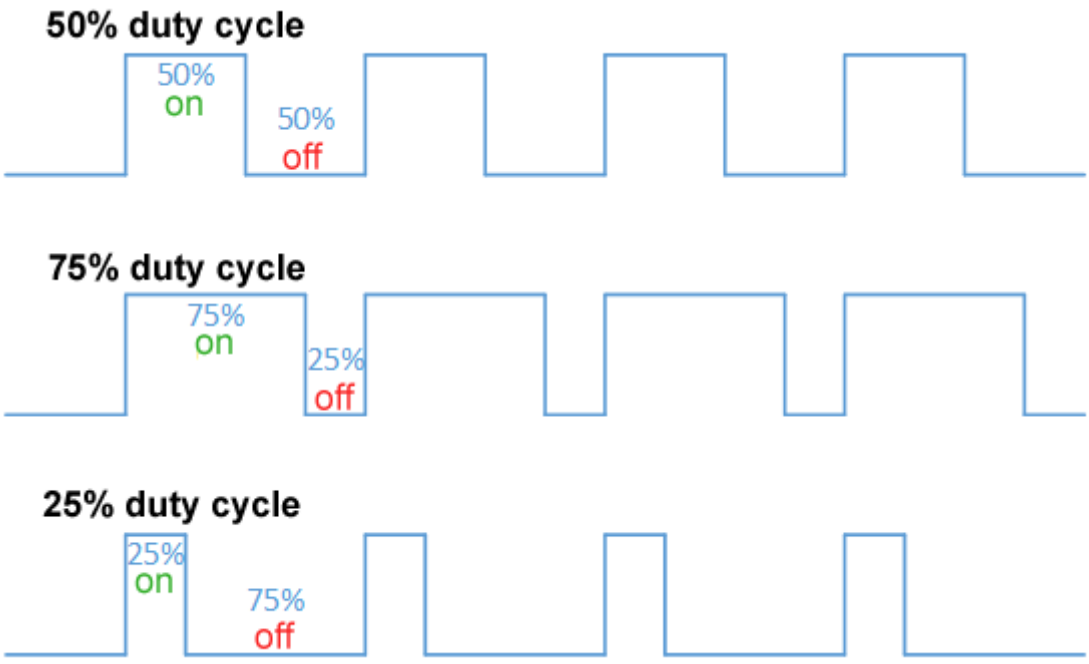


Figura 34 Ciclos de trabajo en un PWM

2.17. REDES NEURONALES

Existen numerosas formas de definir a las redes neuronales; desde las definiciones cortas y genéricas hasta las que intentan explicar más detalladamente qué son las redes neuronales. Por ejemplo:

1) Una nueva forma de computación, inspirada en modelos biológicos.
2) Un modelo matemático compuesto por un gran número de elementos procesales organizados en niveles.
3) Un sistema de computación compuesto por un gran número de elementos simples, elementos de procesos muy interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas.
4) Redes neuronales artificiales son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico.

Debido a su constitución y a sus fundamentos, las redes neuronales artificiales presentan un gran número de características semejantes a las del cerebro. Por ejemplo, son capaces de aprender de la experiencia, de generalizar de casos anteriores a nuevos casos, de abstraer características esenciales a partir de entradas que representan información irrelevante, etc. Esto hace que ofrezcan numerosas ventajas y que este tipo de tecnología se esté aplicando en múltiples áreas. Entre las ventajas se incluyen:

Aprendizaje Adaptativo. Capacidad de aprender a realizar tareas basadas en un entrenamiento o en una experiencia inicial.

Auto-organización. Una red neuronal puede crear su propia organización o representación de la información que recibe mediante una etapa de aprendizaje.

Tolerancia a fallos. La destrucción parcial de una red conduce a una degradación de su estructura; sin embargo, algunas capacidades de la red se pueden retener, incluso sufriendo un gran daño.

Operación en tiempo real. Los cómputos neuronales pueden ser realizados en paralelo; para esto se diseñan y fabrican máquinas con hardware especial para obtener esta capacidad.

Fácil inserción dentro de la tecnología existente. Se pueden obtener chips especializados para redes neuronales que mejoran su capacidad en ciertas tareas. Ello facilitará la integración modular en los sistemas existentes (Matich, 2001).

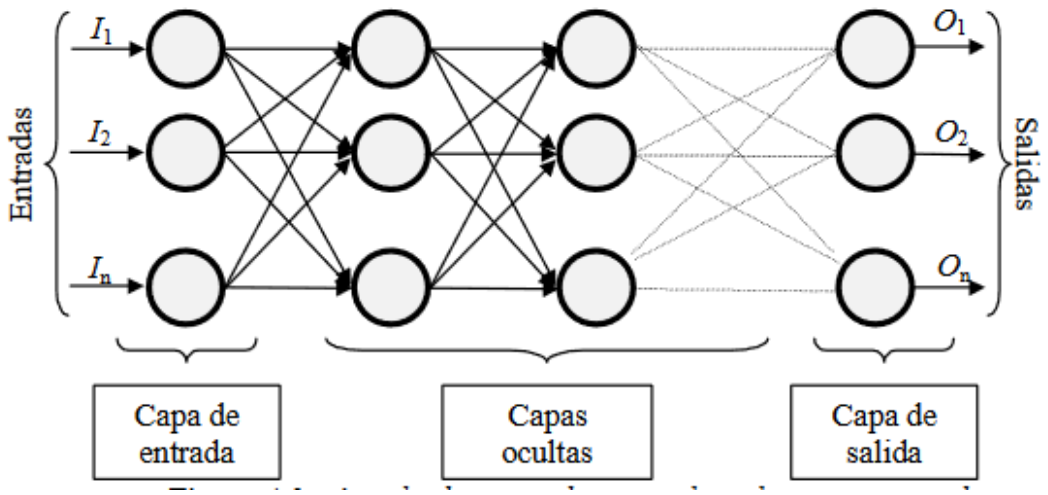


Figura 35 Esquema de una red neuronal con una capa oculta

Está constituida por neuronas interconectadas y arregladas en tres capas (esto último puede variar). Los datos ingresan por medio de la “capa de entrada”, pasan a través de la “capa oculta” y salen por la “capa de salida”. Cabe mencionar que la capa oculta puede estar constituida por varias capas.

CAPÍTULO III. METODOLOGÍA

3.1. REINGENIERÍA DEL DISPOSITIVO HÁPTICO

DIAGNÓSTICO DEL DISPOSITIVO

El dispositivo háptico marca Phantom con el que se trabajó se compone de:

- Robot háptico
- Control box
- Tarjeta de comunicación con PC

La primera parte del trabajo fue deliberar el estado general del equipo completo, pues se había reportado que el dispositivo había dejado de funcionar, sin embargo, no se sabía la razón, pues se compró en ese estado.

Figura 36 Control Box

El robot tiene dos cables de comunicación que van directo al control box, los cuales había que inspeccionar en busca de falsos contactos o daños físicos, para esto se hizo una prueba de continuidad en cada una de las líneas que atravesaban los cables, misma prueba que no reveló ninguna clase de daño.

A partir de esto fue necesario hacer una inspección más a fondo, abriendo el control box.

El módulo parece funcionar con una interfaz de comunicación con la computadora, una fuente dual de 54V y tres amplificadores para controlar los motores. Lo primero que había que inspeccionar entonces era el estado en que estaba la fuente de poder, la cual fue descartada como falla debido a que proporcionaba un voltaje estable dentro de las especificaciones de trabajo del dispositivo.

Por último, había que inspeccionar el funcionamiento de los amplificadores, para esto se tuvo que investigar su tipo y especificaciones, ya que no había nada de información sobre ellos en la carcasa u otra parte del control box. Después de hacer la búsqueda de esta información se encontró que eran amplificadores PWM modelo 303 de Copley Motion Systems, y como era de esperar ya estaban descontinuados.

Figura 37 Amplificadores PWM modelo 303 de Copley Motion Systems

Los amplificadores tienen un LED que informa de la correcta operación de los mismos, LED que no encendía en ninguno de ellos, lo que dejaba claro que no funcionaban apropiadamente.

A partir de aquí fue fácil deliberar que no era práctico hacer una reparación conservando el viejo sistema del dispositivo, ya que no solo el software lo complicaba, sino también el cable de comunicación del control box con la computadora que estaba perdido y tendría que hacerse de cero sin un diagrama confiable del cableado interno del mismo. Por si fuera poco, la plataforma corre sobre un programa basado en tecnología de hace más de 25 años.

IDENTIFICACIÓN DEL ROBOT

El primer paso para poder trabajar con el robot a fin de diseñar una plataforma que sea capaz de recolectar la información de la posición de su actuador final que permita hacer control cobre sus motores a partir de la posición de los mismos, es saber cómo es que podemos obtener una lectura de la posición de cada una de sus articulaciones.

Figura 38 Dispositivo háptico Phantom Premium

Como se aprecia en la Figura 38, el robot cuenta con 3 articulaciones, a las que llamaremos, brazo (motor superior), antebrazo (motor inferior) y cintura (motor base).

La posición de cada motor es obtenida por un encoder HEDM-5500#B02 (Figura 39).

Figura 39 Encóder HEDM-5500

El diagrama de terminales se muestra en la Figura 40.

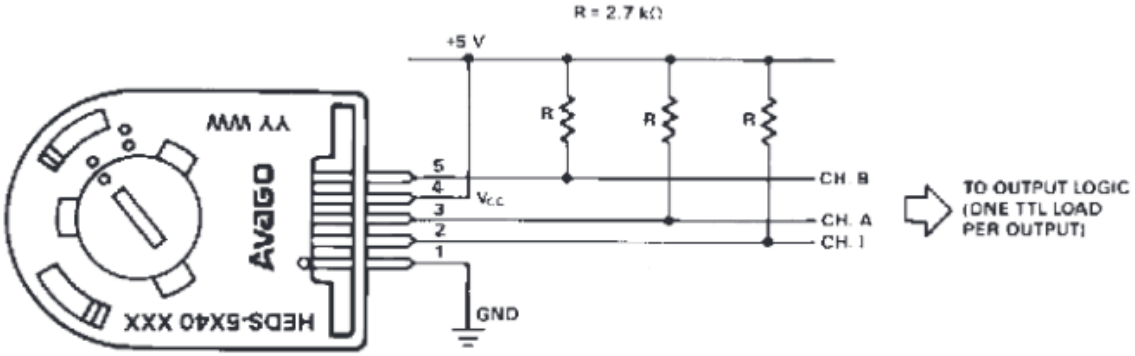


Figura 40 Diagrama de las terminales del HEDM-5500

La hoja de datos (Anexo A) nos revela que la resolución del mismo es de 1024 cuentas por giro de la flecha, una resolución bastante elevada, sin embargo, al no ser un encoder absoluto, implicará hacer un proceso de calibración cada vez que se use el robot.

Además de registrar el movimiento de los motores otra cosa muy importante es saber la dirección con la cual se están moviendo, esto se logra mediante el uso del segundo canal con que cuenta este encoder, como vemos en la Figura 41, las señales están desfasadas 90 grados, lo cual hace posible que en un sentido el canal A adelante al canal B y en sentido contrario B adelante a A.

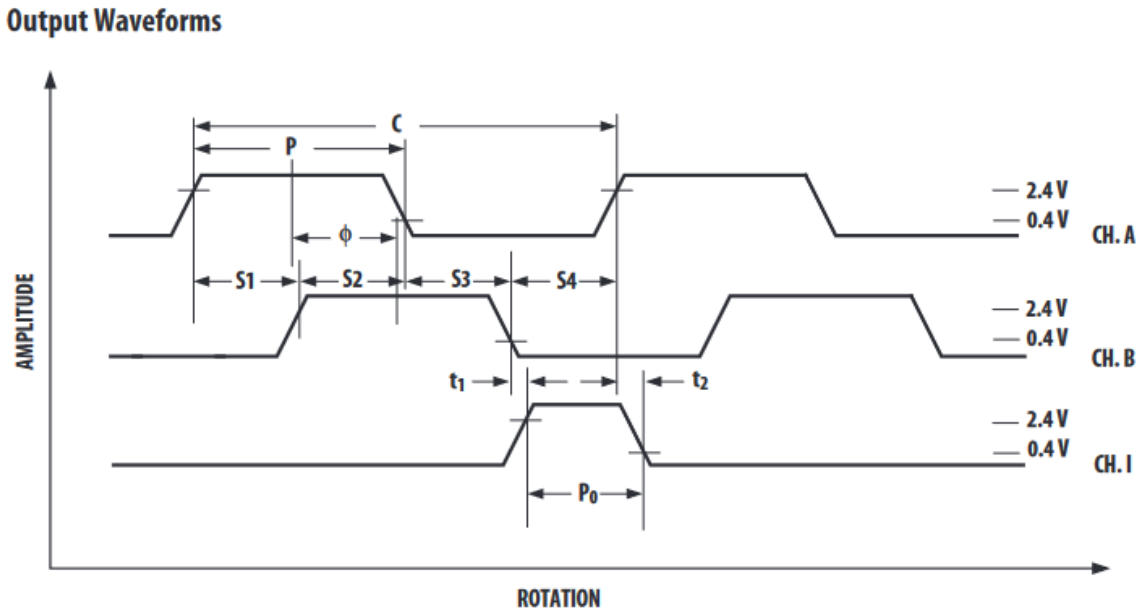


Figura 41 Señales de los canales del HEDM-5500

Teniendo en cuenta estos datos se hizo una rutina de obtención de posición que se explicará más adelante.

Considerando dos líneas para la alimentación (5V) que son terminales comunes para los 3 encoders, y 2 terminales para el canal A y el canal B de cada uno de los encoders, se consideró rastrear estas 8 señales en el conector rj15 que reúne todas las señales de obtención de datos del robot.

Una vez se rastrearon y etiquetaron dichas señales, pudimos ver el conector restante, el cual simplemente contenía las terminales de control de los motores.

Ahora bien, por la disposición de los motores nos dimos cuenta de que la posición que nos importa no es como tal la de la flecha del motor, ya que esta flecha tiene

un engrane que se traslada sobre un círculo mucho más grande, el cual define la posición final de cada una de las articulaciones con respecto a su eje principal, como se observa en el esquema de la Figura 42.

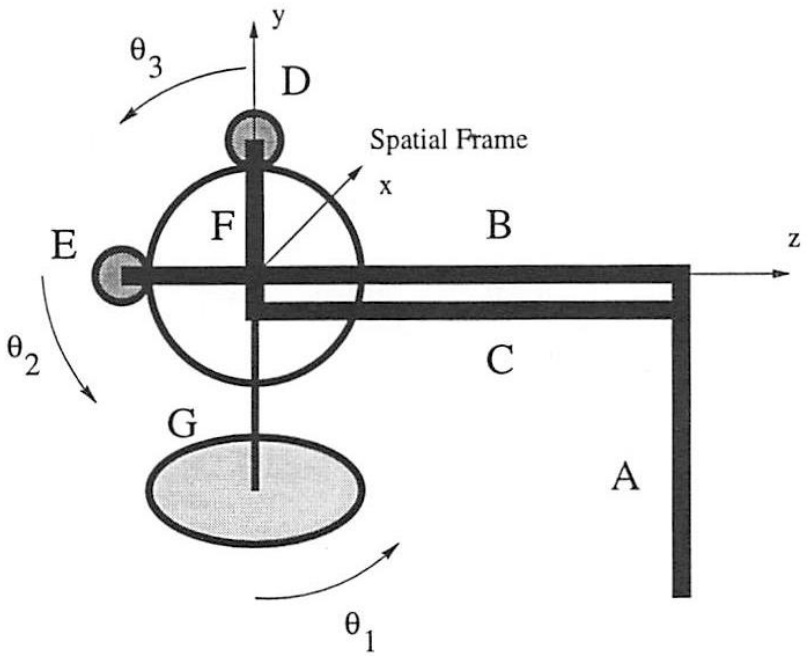


Figura 42 Diagrama abstracto de movimiento del dispositivo háptico

Para saber la posición de la articulación a partir de la posición de los motores tenemos que conocer el avance de la articulación por cada grado que avanza el motor, por lo que necesitamos primero obtener la magnitud del arco que recorre el motor y cuantas vueltas da el motor para hacer el recorrido de ese arco, lo cual está descrito por la siguiente fórmula.

$$\delta = \frac{\theta_{articulación}}{\theta_{motor}} \tag{2}$$

$$\theta_{articulación} = Magnitud\ del\ arco\ máximo\ que\ recorre\ la\ articulación$$

$$\theta_{motor} = Cantidad\ de\ vueltas\ dadas\ al\ recorrer\ el\ arco\ máximo\ de\ la\ articulación$$

Esta relación será la misma para los motores que describen Θ_2 y Θ_3 ya que el engrane del motor y el engrane sobre el que están acoplados son idénticos, si embargo, en el caso del motor de la cintura, si fue necesario calcular la relación mediante el mismo método.

Al final se obtuvieron los siguientes valores:

$$\delta 1 = 0.04804805 \tag{3}$$

$$\delta 2 = 0.04285722 \tag{4}$$

Estos valores dan la distancia en grados, por lo que la fórmula de conversión para obtener la posición en radianes al final queda de la siguiente forma:

$$\theta_{articulación} = \frac{\theta_{motor}\pi\delta}{180} \tag{5}$$

Cambiando la razón de giro entre motor y articulación en base a si se está calculando la posición del antebrazo y brazo o la cintura.

CINEMÁTICA

Una vez que tenemos la posición de cada una de las articulaciones, la forma para conocer la posición del actuador final es hacer la cinemática directa.

El problema cinemático de este manipulador se puede dividir en dos problemas independientes, los de obtener el mapa directo e inverso entre:

-Los ángulos $\theta_1, \theta_2, \theta_3$ y la posición del elemento terminal, medida con respecto al sistema base.

Los ángulos $\theta_4, \theta_5, \theta_6$ y la orientación del elemento terminal, medida con respecto al sistema base, sin embargo, por ahora nos centraremos en hacer la cinemática limitada a θ_1, θ_2 y θ_3. La configuración inicial del manipulador y la nomenclatura utilizada se muestran en la Figura 43.

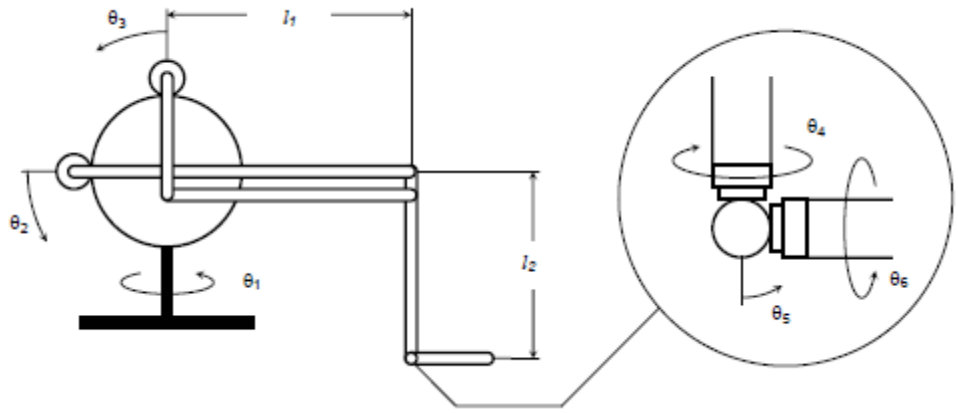


Figura 43 Posición inicial del dispositivo háptico

Las transformaciones de cuerpo rígido y cálculos cinemáticos siguen la notación de la Figura 44.

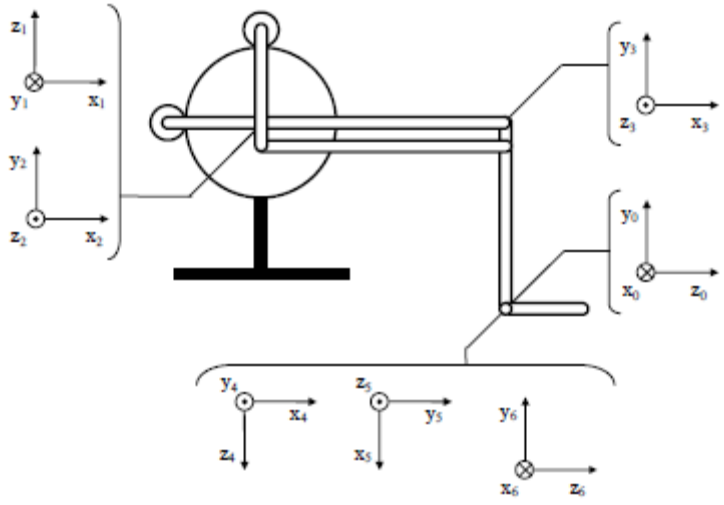


Figura 44 Notaciones para transformaciones de cuerpo rígido y cálculos cinemáticos

Las transformaciones homogéneas entre el sistema {i} y el sistema {i-1}, para iϵ{1,…,6} son:

$$ {}^{0}_{1}T = \begin{bmatrix} \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & l_2 \\ \cos(\theta_1) & -\sin(\theta_1) & 0 & -l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6) $$

$$ {}^{1}_{2}T = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin(\theta_2) & \cos(\theta_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7) $$

$$ {}^{2}_{3}T = \begin{bmatrix} \cos(\theta_3 - \theta_2) & -\sin(\theta_3 - \theta_2) & 0 & l_1 \\ \sin(\theta_3 - \theta_2) & \cos(\theta_3 - \theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8) $$

$$ {}^{3}_{4}T = \begin{bmatrix} \cos(\theta_4) & -\sin(\theta_4) & 0 & l_1 \\ 0 & 0 & -1 & -l_2 \\ \sin(\theta_4) & \cos(\theta_4) & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9) $$

$$ {}^{4}_{5}T = \begin{bmatrix} \sin(\theta_5) & \cos(\theta_5) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \cos(\theta_5) & -\sin(\theta_5) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10) $$

$$ {}^{5}_{6}T = \begin{bmatrix} -\sin(\theta_6) & -\cos(\theta_6) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\cos(\theta_6) & \sin(\theta_6) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11) $$

La transformación homogénea entre el sistema {6} y el sistema {0} se obtiene a partir de la composición:

$$ {}^{0}_{6}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{6}T \tag{12} $$

$$ {}^{0}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & {}^{0}_{6}R & & {}^{0}_{6}d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13} $$

Por razones de simplicidad y brevedad, se adoptará en lo que resta de documento la notación $\sin(\theta) = s_\theta$ y $\cos(\theta) = c_\theta$, para las posiciones de las articulaciones y $\sin(\alpha) = s_\alpha$ y $\cos(\alpha) = c_\alpha$ para los demás ángulos.

$$ r_{11} = -s_1c_3c_4s_5s_6 - s_1s_3c_5c_6 + s_1c_3s_4c_6 + c_1s_4s_5s_6 + c_1c_4c_6 \tag{14} $$

$$ r_{12} = -s_1c_3c_4s_5c_6 - s_1s_3c_5c_6 - s_1c_3s_4c_6 + c_1s_4s_5s_6 - c_1c_4c_6 \tag{15} $$

$$ r_{13} = s_1c_3c_4c_5 - s_1s_3s_5 + c_1s_4c_5 \tag{16} $$

$$ r_{21} = -s_3c_4s_5c_6 + s_3s_4c_6 + c_3c_5s_6 \tag{17} $$

$$ r_{22} = -s_3c_4s_5c_6 - s_3s_4s_6 + c_3c_5c_6 \tag{18} $$

$$ r_{23} = s_3c_4c_5 + c_3s_5 \tag{19} $$

$$ r_{31} = -c_1c_3c_4s_5s_6 - c_1s_3c_5c_6 + c_1c_3s_4c_6 - s_1s_4s_5s_6 - s_1c_4c_6 \tag{20} $$

$$ r_{32} = -c_1c_3c_4s_5s_6 - c_1s_3c_5c_6 - c_1c_3s_4s_6 - s_1s_4s_5s_6 + s_1c_4s_6 \tag{21} $$

$$ r_{33} = c_1c_3c_4c_5 - c_1s_3s_5 + s_1s_4c_5 \tag{22} $$

$$ r_{11} = -s_1c_3c_4s_5s_6 - s_1s_3c_5c_6 + s_1c_3s_4c_6 \tag{23} $$

$$ p_x = s_1(l_1c_2 + l_2s_3) \tag{24} $$

$$ p_y = l_1s_2 + l_2(1 - c_3) \tag{25} $$

$$ p_z = l_1(c_1c_2 - 1) + l_2(c_1s_3) \tag{26} $$

Las ecuaciones, son las que nos ayudan a saber la posición en metros del actuador final, sin embargo, aún es necesario obtener el valor de l_1 y l_2, lo cual haremos mediante una medición simple, lo cual como resultado nos da 5.65 in, lo que convertido a metro es 0.014351 m.

FUNCIONAMIENTO GENERAL DEL SISTEMA

El sistema propuesto consta de 4 programas separados que tienen una función en particular, el primero de ellos es el programa que se cargará a la tarjeta de adquisición de datos (Arduino mega), el programa de recepción y procesamiento de datos basado en LabVIEW, un programa en MATLAB para la graficación de la posición del actuador final ante una trayectoria y un programa de Python para el procesamiento de las gráficas obtenidas por el programa de MATLAB.

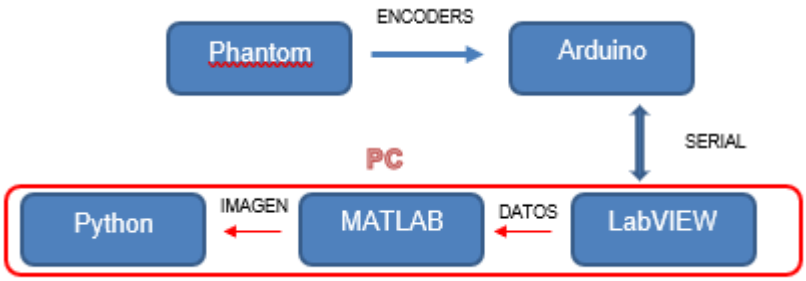


Figura 45 Diagrama a bloques del sistema para obtención de datos para el dispositivo háptico

PROGRAMA DE ARDUINO

El programa de Arduino tiene como tarea obtener las señales de los 3 encoders del sistema y convertirlas en la posición cartesiana del actuador final mediante las fórmulas de cinemática obtenidas previamente. Esta información habrá de ser enviada vía comunicación serial al programa de procesamiento de datos de LabVIEW que será tratado posteriormente.

La primera parte del programa que se abordó fue la obtención y manejo de las señales de los encoders. Como se dijo antes, los encoders cuentan con dos canales, uno de los cuales adelanta al otro o viceversa en base a la dirección de

giro, de esta forma si la señal de A aparece primero y después la de B entonces diremos que es un giro en sentido positivo, de lo contrario será un sentido negativo. Para obtener la posición en grados de la articulación utilizamos la Fórmula 5 la cual describe la relación entre la cantidad de pulsos que obtenemos del encoder y la posición en grados. Ahora bien, es importante considerar que la cantidad de pulsos recibidos por el Arduino deben ser restados o sumados a un contador asignado a cada articulación en base a si el giro es en una dirección o la otra, y todo esto debe ser procesado para cada una de las articulaciones, lo cual no podría hacerse con precisión en un programa secuencial, el cual podría perder el sensado de pulsos mientras atiende otras partes del programa. Debido a lo anterior se optó por utilizar el uso de interrupciones, las cuales están detonadas por el un flanco de subida detectado en el pin de la interrupción, el código que se ejecuta con cada evento se muestra a continuación:

```
void tetamotorinf()
{
  // Disminuimos el valor establecido
  flag=digitalRead(9);
  if(flag==HIGH){
    cuenta=cuenta-1;
  }
  if(flag==LOW){
    cuenta=cuenta+1;
  }
}
```

En este caso tomamos de ejemplo la rutina de la interrupción asignada a la articulación controlada por el motor inferior. Como es de observarse, dentro de la interrupción se obtiene la señal del puerto digital 9, esto debido al comportamiento del encoder, si consideramos que el canal A adelanta al B, entonces al momento de detectar el flanco de subida, B todavía no estará en nivel alto, sino bajo, caso contrario de si B adelanta a A, ya que en este caso al detectar el flanco de subida de A, B ya estará en un nivel alto, de esta forma podemos saber si el giro es en dirección positiva o negativa, en base a lo cual incrementamos o decrementamos el contador “cuenta”. Cada articulación tiene una rutina idéntica, pero con variables

dedicadas a cada uno de ellos, las cuales se declaran al inicio del programa, así como un puerto de interrupción y un puerto de lectura del canal B propio.

Una vez expuesta la rutina de interrupción podemos pasar al programa principal que se ejecuta de manera cíclica durante todo el tiempo que dure la ejecución.

```
void loop() {
delay(50);
teta2=(cuenta*.04804805*p)/180;
teta3=(cuental*.04804805*p)/180;
tetal=(cuenta2*.04285722*p)/180;
x=sin(tetal)*((l*cos(teta2))+(l*sin(teta3)));
y=l*sin(teta2)+(l*(l-cos(teta3)));
z=(l*((cos(tetal)*cos(teta2))-l))+(l*(cos(tetal)*sin(teta3)));
xs = String(x,2);
ys = String(y,2);
zs = String(z,2);
Serial.println("xs "+xs+" ys "+ys+" zs "+zs);
}
```

El programa se ejecuta en su totalidad una vez cada 50 milisegundos, aunque podría asignarse una ejecución más rápida, cambiando también a una velocidad mayor en la configuración de la comunicación serial.

Las variables teta1, teta2 y teta3 representan la posición en grados de cada una de las articulaciones, y se obtienen con la Fórmula 5, donde p representa a π redondeado a 11 posiciones decimales.

Ya que se han obtenido los valores de los ángulos ahora se pueden obtener las coordenadas cartesianas usando las fórmulas 24, 25 y 26, lo cual se calcula con las fórmulas que dan valor a las variables “x”, “y” y “z”. Por simplificación se decidió truncar las coordenadas de posición a 2 posiciones decimales, por lo cual se utilizó un método de conversión de variable tipo “double” a “String”, truncada a 2 posiciones, lo cual dio origen a las variables de posición finales “xs”, “ys” y “zs”.

Por último, se compone el mensaje serial final que será enviado, poniendo el identificador de la variable (xs, ys o zs), un espacio y después su respectivo valor, todo en una misma cadena que será enviada mediante el puerto serial.

PROGRAMA DE LABVIEW

Este programa tiene como propósito recibir los datos que envía el Arduino a través del puerto serial, procesar la cadena y registrar en una tabla de datos las cada una de las coordenadas con respecto al tiempo de una trayectoria hecha con el actuador final.

La primera parte del programa tiene por tarea inicializar la comunicación serial y crear los archivos para el guardado de las trayectorias, en este caso, un archivo de texto para cada eje y uno más para el tiempo, esta inicialización podemos verla en la Figura 46.

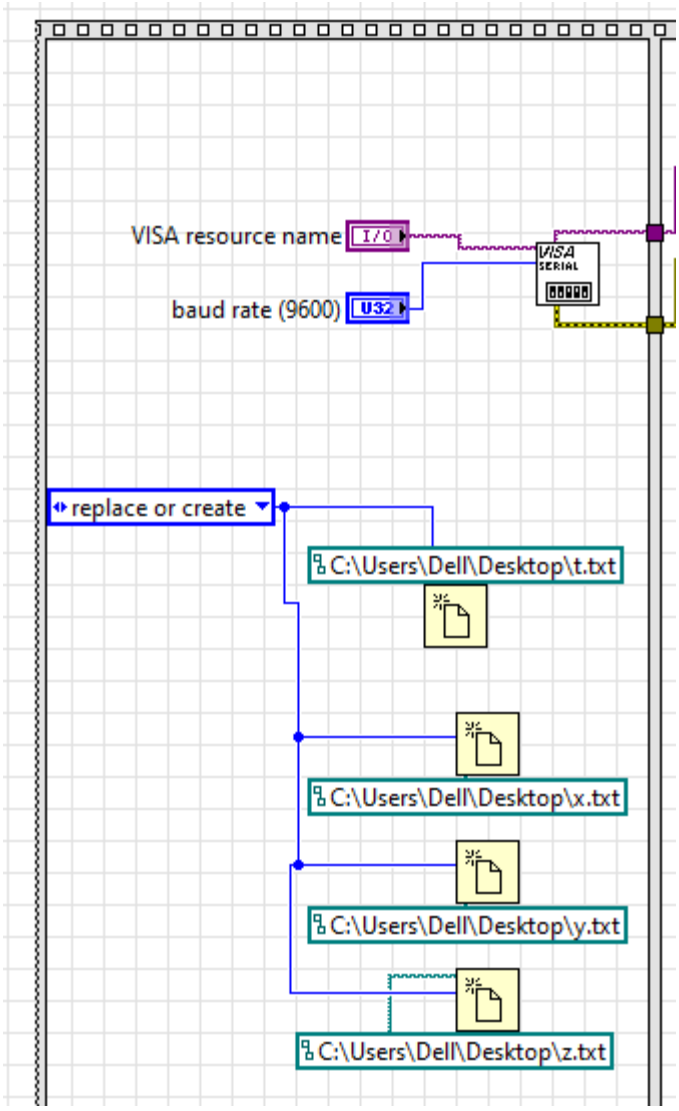


Figura 46 Primera parte del programa en LabVIEW

Estas acciones están todas en la primera etapa de una secuencia plana, asegurándonos así de que sean lo primero que se ejecute cada vez que se corra el programa.

Una vez que las variables se inicializan, se comienza a hacer la recepción de datos y simultáneamente se hace el guardado de las posiciones en los archivos de almacenamiento, rutinas que están todas dentro de una rutina cíclica que se repite durante todo el tiempo que el programa esté en funcionamiento cada 50ms.

La lectura es detonada por eventos, en este caso, por la detección de bytes en el puerto, lo cual activa una la rutina de procesamiento de datos, que podemos ver en la Figura 47.

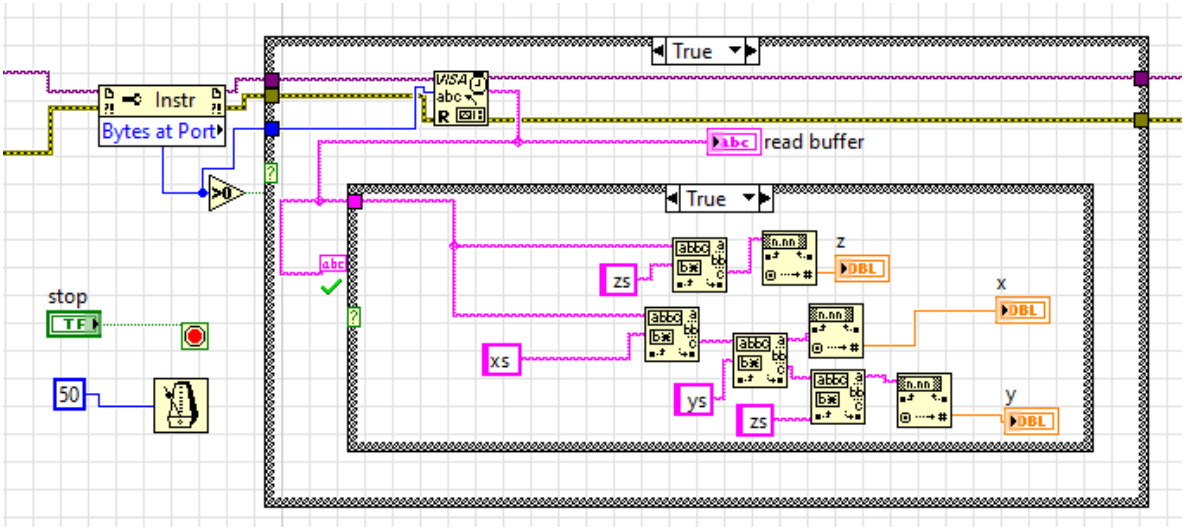


Figura 47 Programa para la recepción y procesamiento de datos de la tarjeta de adquisición

Una vez se reciben datos es importante verificar que los datos recibidos sean una cadena de texto sana, sin errores y con toda la información necesaria, para lo cual se diseñó un VI especial para la verificación de cadenas, el cual da como salida un valor booleano que indica si la cadena es funcional o no para ser procesada, lo cual permite que los datos recibidos se registren en el archivo de texto de cada plano.

El VI de verificación de cadenas considera que una cadena sana debe tener los 3 identificadores en posiciones específicas de la cadena, que van dentro de un rango especifico dependiendo del tamaño del número que acompaña a cada identificador. Este VI puede observarse en la Figura 48.

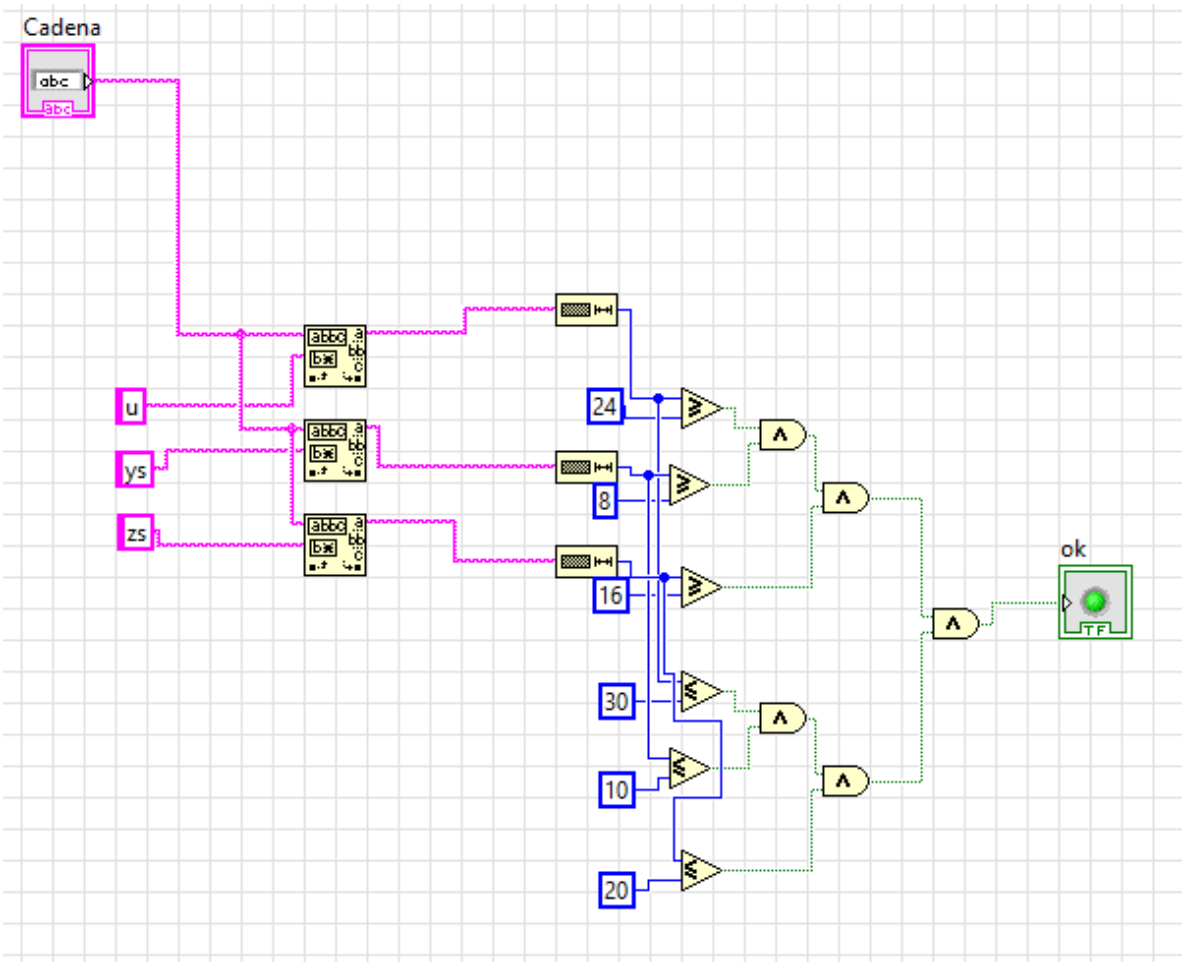


Figura 48 VI para la verificación de la cadena de entrada recibida por el puerto serial

Si la cadena se detecta como sana, entonces se secciona la misma y se direcciona cada dato a su plano correspondiente, todo mediante el siguiente diagrama a bloques mostrado en la Figura 49.

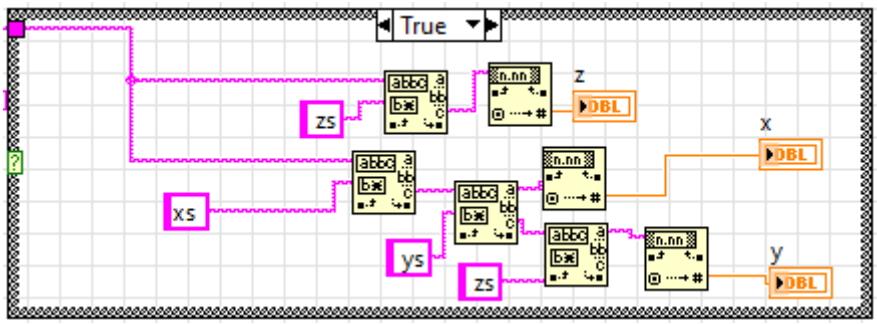


Figura 49 VI de procesamiento de cadena

En paralelo, se toman los valores de las variables relacionadas con cada plano y se guardan en el archivo creado para cada una de ellas, como ejemplo tomaremos el diagrama a bloques del guardado de datos para el tiempo.

La obtención del tiempo es diferente al de los valores de las coordenadas, ya que lo tomamos a partir de la cantidad de ciclos en el ciclo general y su tiempo de ejecución cíclico, lo cual guardamos en una variable asignada al mismo, la cual es la que se guarda en su respectivo archivo de texto.

El guardado de los valores de tiempo se muestra en la Figura 50.

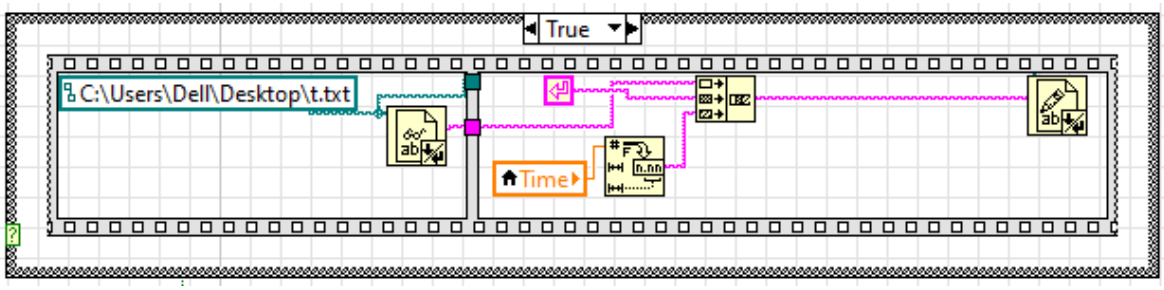


Figura 50 Rutina para la escritura de datos en un archivo de texto

Si el botón de guardado de datos está activado entonces se entra en la rutina de escritura, la cual depende de una secuencia plana, la cual primero lee el archivo de la ruta que definimos al principio del programa y después a lo ya escrito le añade primero un salto de párrafo y después el valor de tiempo leído en el momento, de esta forma no sobrescribimos sobre la misma línea, sino al final tenemos el valor de tiempo de cada ciclo en cada renglón, lo cual sucede de la misma forma para las variables de las coordenadas.

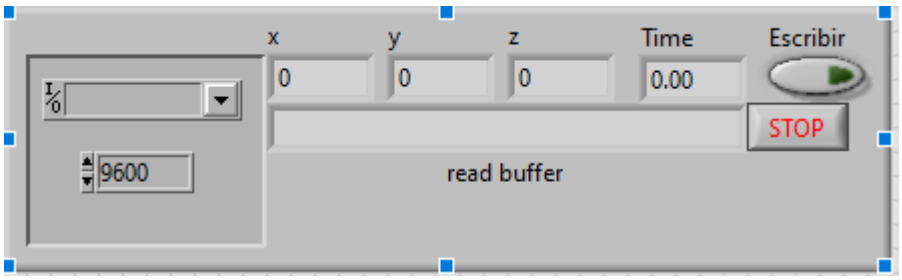


Figura 51 Interfaz gráfica del programa de adquisición de datos

PROGAMA DE MATLAB

El siguiente paso es tomar los 4 archivos de texto plano con los datos de la trayectoria registrada por LabVIEW y convertirlos a una gráfica que muestre los datos registrados durante la trayectoria como señales a través del tiempo, esta tarea se realizó mediante un código de MATLAB, el cual se muestra a continuación:

```
clear
clc
fileID = fopen('x.txt','r');
formatSpec = '%f';
x = fscanf(fileID,formatSpec);
fileID2 = fopen('t.txt','r');
t = fscanf(fileID2,formatSpec);
fileID3 = fopen('y.txt','r');
y = fscanf(fileID3,formatSpec);
fileID4 = fopen('z.txt','r');
z = fscanf(fileID4,formatSpec);
h(1) = figure;
plot(t,x,t,y,t,z)
set(gca,'visible','off')
saveas(h,sprintf('figure.jpg'));
close(h)
fclose('all');
```

Cada vez que se termina de ejecutar el programa de LabVIEW se ejecuta este código y lo que se obtiene es algo parecido a lo que se ve en las figuras.

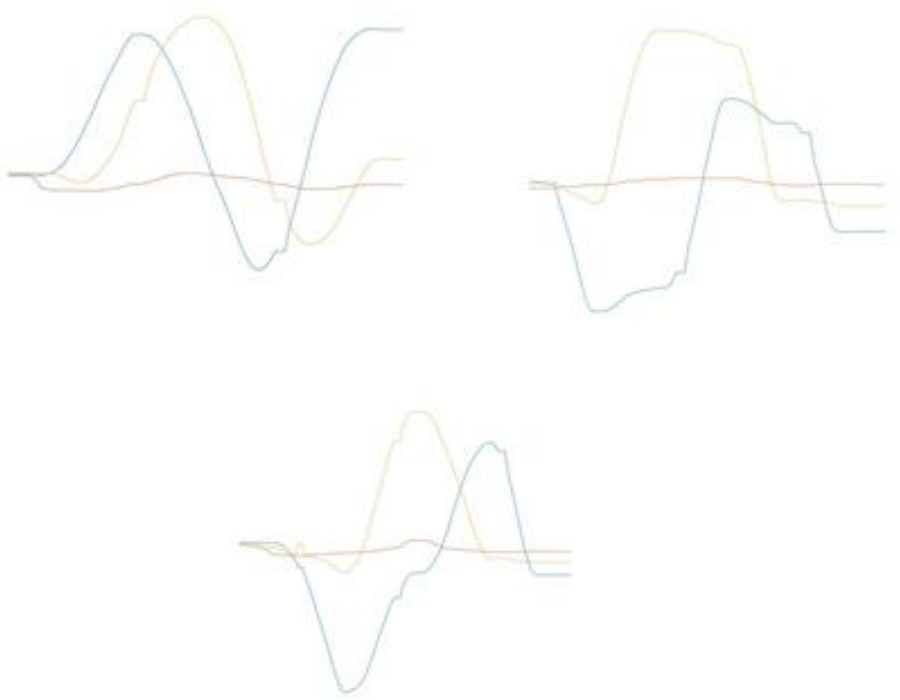


Figura 52 Señales obtenidas para una trayectoria, circular, cuadrada y triangular

Ahora bien, las imágenes obtenidas no son lo suficientemente buenas para ser procesadas por una red neuronal simple, ya que debido a su alta resolución y características requerirían de demasiados recursos y una red demasiado compleja, debido a esto se tomó la decisión de hacer un tratamiento especial a cada una de las imágenes mediante un programa más.

PROGRAMA DE PROCESAMIENTO EN PYTHON

La función de este programa es reducir la galería de imágenes obtenidas a un tamaño que sea fácil de procesar, así como hacer un tratamiento a las imágenes obtenidas, de manera que sea posible resaltar las características más relevantes, las cuales permitan a la red neuronal tener un mejor entrenamiento y realizar mejor su tarea.

El programa principal consta de dos ciclos, el primero de ellos lee la cantidad de imágenes dentro de un fichero definido en la ruta guardada en una variable, el

segundo de los ciclos utiliza la cuenta del primer ciclo para determinar la cantidad de imágenes que procesará.

Dentro del segundo ciclo se encuentra la rutina que procesa, modifica y guarda las nuevas imágenes en un nuevo fichero, el código podemos observarlo a continuación:

```
ejemplo_dir='C:/Users/uif10179/Downloads/Imagenes'
directorio = pathlib.Path(ejemplo_dir)
n=0
#Se obtiene la cantidad de imágenes dentro de la carpeta
for fichero in directorio.iterdir():
    n=n+1
n=n-1
for i in range(n):
    #Se compone la ruta en base a la imagen que se va a leer
    ruta='C:/Users/uif10179/Downloads/Imagenes/figure'+str(i+1)+'.jpg'
    #Se compone la ruta en base a la imagen que se va a escribir
    ruta2='C:/Users/uif10179/Downloads/Resize/figure'+str(i+1)+'_resize.jpg'
    src = cv2.imread(ruta, cv2.IMREAD_GRAYSCALE) #Lectura de imagen en
escala de grises
    dsize=datos_img(src.shape[0], src.shape[1]) #reduccion 25%
    src2=tratamiento(cv2.resize(src,dsize)) # Convolucion y negativo de
imagen
    #Se llama al metodo de celculo de las nuevas dimensiones
    dsize2=datos_img(src2.shape[0], src2.shape[1]) #Reduccion 25%
    output=cv2.resize(tratamiento(src2), dsize2) #Convolucion y negativo de
imagen
    cv2.imwrite(ruta2,output) #Imagen resultante
```

Como se puede observar dentro del código, la lectura de las imágenes mediante el método imred de la librería cv2, está condicionada para que sea en una escala de grises, posteriormente, esta imagen es enviada a un método que obtiene las dimensiones finales de la imagen para una reducción del 75%, datos que después se utilizan para reducir la imagen, para después enviarla a la función de tratamiento, que incluye una convolución y un negativo de la imagen, este proceso se repite una vez más antes de llegar al resultado final.

El método de obtención de las dimensiones para la reducción de la imagen se muestra a continuación:

```
scale_percent = 25
def datos_img(x,y):
    #calcular el 50 por ciento de las dimensiones originales
    width = int(y * scale_percent / 100)
    height = int(x * scale_percent / 100)
    h=[width, height]
    return h
```

Como se puede observar, una variable es la que controla el factor de reducción con que la función trabajará.

La función que se encarga del tratamiento de la imagen se muestra a continuación:

```
def tratamiento(src):
    dist = 255 - src
    dsize=[dist.shape[1], dist.shape[0]]
    #dsize=[src.shape[1], src.shape[0]]
    a = [ [ 1.0, 1.0, 1.0 ],
      [ 1.0, 1.0, 1.0 ],
      [ 1.0, 1.0, 1.0 ] ]
    kernel = np.asarray(a)
    dst = cv2.filter2D(dist, -1, kernel)
    dist2 = 255 - dst #negativo de la imagen obtenida
    return dist2
```

La función necesita que se le envíe una imagen como parámetro, esta imagen es utilizada primero para obtener sus dimensiones. Ahora bien, para hacer una convolución es importante primero tener la matriz de convolución, la cual tendrá un efecto diferente sobre la imagen en base a los valores que en ella estén, en nuestro caso decidimos utilizar una simple con unos en todas las casillas. La convolución se guarda en una variable a la que después se le hace el negativo, restando 255 a cada bit de color dentro de la imagen, para así tener el resultado final.

Las imágenes obtenidas después del tratamiento completo tienen un tamaño de 75 x 56 pixeles, cuando la imagen original era de 875 x 656 pixeles, en la Figura 53 se muestran algunos ejemplos de los resultados.

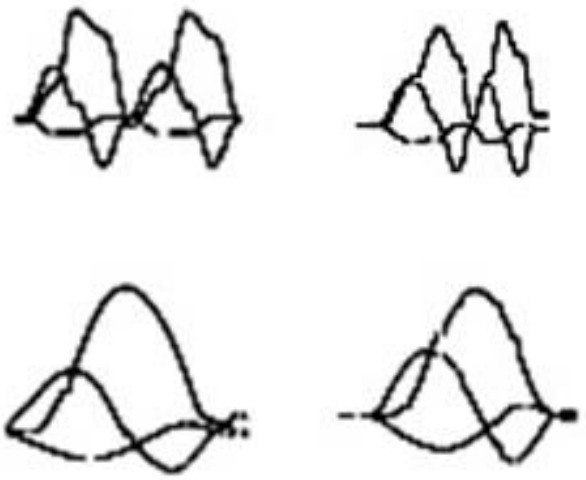


Figura 53 Trazos obtenidos después del programa de procesamiento de imágenes de Python

3.2. DISEÑO DE RED NEURONAL

Para el diseño de la red neuronal se utilizó el entorno de ejecución en Python que ofrece Google mediante Google COLAB, en este pudimos hacer el entrenamiento de una forma sencilla y sin requerir de un ordenador con altas capacidades.

Cabe señalar que la primera prueba de entrenamiento para la red neuronal fue que notará la diferencia entre un trazo que contiene una sola trayectoria circular y un trazo que tuviera más de una sola trayectoria circular, podemos ver dos muestras de estos trazos en la Figura 54.

Figura 54 Trazo con una trayectoria circular y dos trayectorias circulares

A partir de esto el primer paso es programar una rutina para generar el set de datos para el entrenamiento y para la validación, usando el código mostrado a continuación:

```
import tensorflow as tf
image_size = (56, 75)
batch_size = 2

train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "/content/Señales",
    validation_split=0.2,
    subset="training",
    seed=1337,
    image_size=image_size,
    batch_size=batch_size,
)
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "/content/Señales",
    validation_split=0.2,
    subset="validation",
    seed=1337,
    image_size=image_size,
    batch_size=batch_size,
)
```

Una vez se tienen los sets de datos es importante agregar algo de ruido al sistema utilizando rotaciones sobre las imágenes, de manera que lo que entra a la red neuronal la haga más robusta.

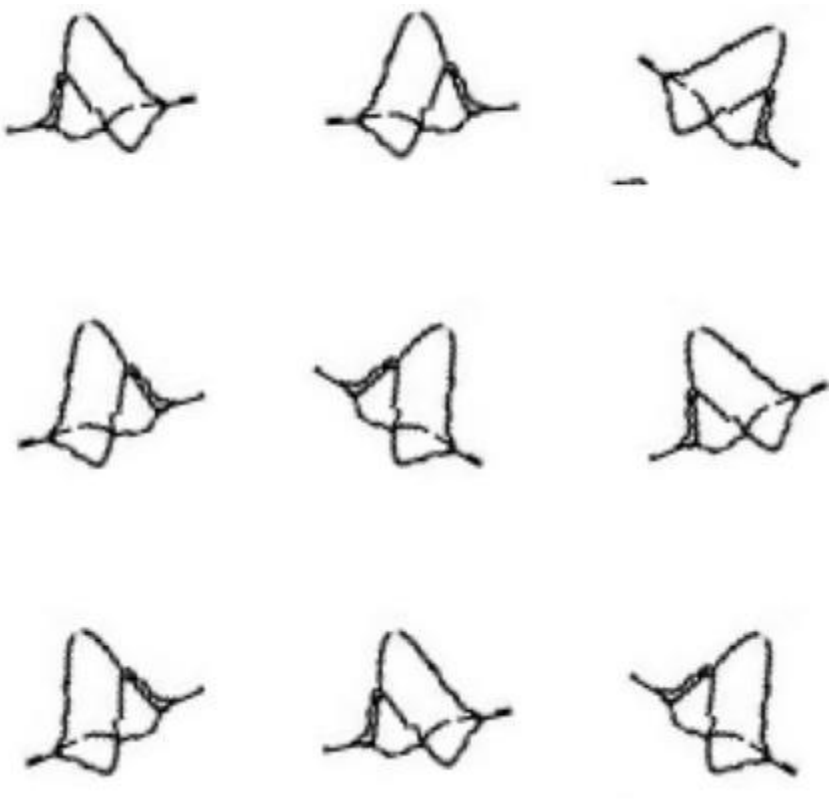


Figura 55 imágenes giradas suministradas para el entrenamiento de la red neuronal

La red neuronal diseñada, se define mediante el código mostrado a continuación:

```
def make_model(input_shape, num_classes):
    inputs = tf.keras.Input(shape=input_shape)
    # Image augmentation block
    x = data_augmentation(inputs)

    # Entry block
    x = tf.keras.layers.Rescaling(1.0 / 255)(x)
    x = tf.keras.layers.Conv2D(32, 3, strides=2, padding="same")(x)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.Activation("relu")(x)

    x = tf.keras.layers.Conv2D(64, 3, padding="same")(x)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.Activation("relu")(x)

    previous_block_activation = x  # Set aside residual

    for size in [128, 256, 512, 728]:
        x = tf.keras.layers.Activation("relu")(x)
```

```
        x = tf.keras.layers.Activation("relu")(x)
        x = tf.keras.layers.SeparableConv2D(size, 3, padding="same")(x
)
        x = tf.keras.layers.BatchNormalization()(x)

        x = tf.keras.layers.Activation("relu")(x)
        x = tf.keras.layers.SeparableConv2D(size, 3, padding="same")(x
)
        x = tf.keras.layers.BatchNormalization()(x)

        x = tf.keras.layers.MaxPooling2D(3, strides=2, padding="same")
(x)

        # Project residual
        residual = tf.keras.layers.Conv2D(size, 1, strides=2, padding=
"same")(
            previous_block_activation
        )
        x = tf.keras.layers.add([x, residual])  # Add back residual
        previous_block_activation = x  # Set aside next residual

    x = tf.keras.layers.SeparableConv2D(1024, 3, padding="same")(x)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.Activation("relu")(x)

    x = tf.keras.layers.GlobalAveragePooling2D()(x)
    if num_classes == 2:
        activation = "sigmoid"
        units = 1
    else:
        activation = "softmax"
        units = num_classes

    x = tf.keras.layers.Dropout(0.5)(x)
    outputs = tf.keras.layers.Dense(units, activation=activation)(x)
    return tf.keras.Model(inputs, outputs)


model = make_model(input_shape=image_size + (3,), num_classes=2)
tf.keras.utils.plot_model(model, show_shapes=True)
```

Con esta red de varias capas intercaladas con capas convolucionales procedimos a hacer el entrenamiento de esta, utilizando el código abajo mostrado.

```
epochs = 20

callbacks = [
    tf.keras.callbacks.ModelCheckpoint("save_at_{epoch}.h5"),
]
model.compile(
    optimizer=tf.keras.optimizers.Adam(1e-3),
    loss="binary_crossentropy",
    metrics=["accuracy"],
)
model.fit(
    train_ds, epochs=epochs, callbacks=callbacks, validation_data=val_
ds,
)
```

Utilizamos un optimizador tipo Adam, el cual se mantuvo debido a que la efectividad de la red fue de casi el 100%.

Los resultados del entrenamiento a 20 épocas arrojaron una precisión de cerca del 99%.

CAPÍTULO IV. RESULTADOS

4.1. ACTIVIDADES DEL PROYECTO

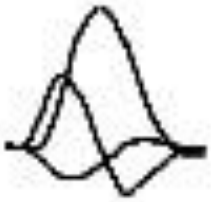
La red neuronal entrenada fue probada mediante la rutina de clasificación expuesta al final de la sección de metodología del capítulo 3, en las figuras 56 a 60, podemos observar 5 pruebas de la red neuronal, donde se muestra la figura evaluada y la clasificación arrojada por la red neuronal.

```
1circulo/figure112_resize.jpg es 99.93 porciento un circulo y 0.07 porciento más de 2 circulos.
```

Figura 56 Clasificación de trazo de una trayectoria circular (a)

```
1circulo/figure120_resize.jpg es 99.90 porciento un circulo y 0.10 porciento más de 2 circulos.
```

Figura 57 Clasificación de trazo de una trayectoria circular (b)



```
1circulo/figure180_resize.jpg es 99.98 porciento un circulo y 0.02 porciento más de 2 circulos.
```

Figura 58 Clasificación de trazo de una trayectoria circular (c)

```
2circulo/figure334_resize.jpg es 0.01 porciento un circulo y 99.99 porciento más de 2 circulos.
```

Figura 59 Clasificación de trazo de varias trayectorias circulares (a)

```
2circulo/figure419_resize.jpg es 0.06 porciento un circulo y 99.94 porciento más de 2 circulos.
```

Figura 60 Clasificación de trazo de varias trayectorias circulares (b)

La misma red neuronal después fue entrenada para hacer la clasificación entre trazos que describen trayectorias circulares y trayectorias cuadradas, los resultados junto con las imágenes se muestran en las figuras.

```
Circulo/figure368_resize.jpg es 98.59 porciento circulo y 1.41 porciento cuadrado.
```

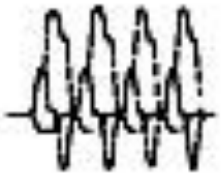
Figura 61 Clasificación de trazo circular (a)

```
Circulo/figure439_resize.jpg es 99.56 porciento circulo y 0.44 porciento cuadrado.
```

Figura 62 Clasificación de trazo circular (b)

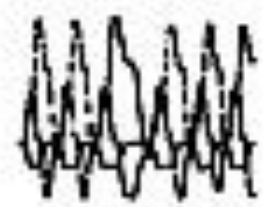
```
Circulo/figure524_resize.jpg es 80.54 porciento circulo y 19.46 porciento cuadrado.
```

Figura 63 Clasificación de trazo circular (c)



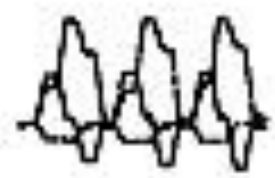
```
Cuadrado/figure3_resize.jpg es 0.01 porciento circulo y 99.99 porciento cuadrado.
```

Figura 64 Clasificación de trazo cuadrado (a)



```
Cuadrado/figure61_resize.jpg es 0.02 porciento circulo y 99.98 porciento cuadrado.
```

Figura 65 Clasificación de trazo cuadrado (b)



```
Cuadrado/figure147_resize.jpg es 0.00 porciento circulo y 100.00 porciento cuadrado.
```

Figura 66 Clasificación de trazo cuadrado (c)

Una prueba más, y la última fue entrenar a la red usando trazos de trayectorias circulares que contuvieran un factor de temblor, de modo que la red fuera capaz de distinguir entre un trazo suave y un trazo con oscilaciones.

El resultado del entrenamiento tuvo una precisión del 84%, lo cual da lugar a malas interpretaciones de las imágenes, en las figuras 67 a 72 vemos algunas clasificaciones hechas con la red.

```
El trazo de la figura Circulo/figure327_resize.jpges  42.14 porciento suave y 57.86 porciento un trazo con temblor.
```

Figura 67 Clasificación de trayectoria circular sin oscilaciones (a)

```
El trazo de la figura Circulo/figure359_resize.jpges  24.15 porciento suave y 75.85 porciento un trazo con temblor.
```

Figura 68 Clasificación de trayectoria circular sin oscilaciones (b)

```
El trazo de la figura Circulo/figure572_resize.jpges  33.34 porciento suave y 66.66 porciento un trazo con temblor.
```

Figura 69 Clasificación de trayectoria circular sin oscilaciones (c)

```
El trazo de la figura Circulo_mal/figure2_resize.jpges  0.01 porciento suave y 99.99 porciento un trazo con temblor.
```

Figura 70 Clasificación de trayectoria circular con oscilaciones (a)

```
El trazo de la figura Circulo_mal/figure118_resize.jpges  0.00 porciento suave y 100.00 porciento un trazo con temblor.
```

Figura 71 Clasificación de trayectoria circular con oscilaciones (b)

```
El trazo de la figura Circulo_mal/figure166_resize.jpges  0.00 porciento suave y 100.00 porciento un trazo con temblor.
```

Figura 72 Clasificación de trayectoria circular con oscilaciones (c)

Como se puede observar, no es posible hacer una predicción completamente certera de si un trazo es o no de una persona sin rigidez, sin embargo, es posible ver que grado de rigidez tiene el trazo, o dicho de otra forma, que tanto se parece al trazo que haría un personas con la enfermedad.

4.2. ACTIVIDADES ADICIONALES

4.2.1. CREACIÓN DE UNA PLATAFORMA EN LABVIEW PARA EL CONTROL DEL DISPOSITIVO HÁPTICO

Uno de los objetivos secundarios del trabajo de reingeniería hecho sobre el robot Phantom fue el habilitar dicho dispositivo como un potencial recurso didáctico para la enseñanza de teoría de control a estudiantes de ingeniería. Debido a esto, se creó un programa de Arduino y una interfaz en LabVIEW para permitir el control sobre cada una de las articulaciones del robot.

El programa de Arduino funciona igual que el programa de adquisición de datos utilizado en el proyecto principal, sin embargo, tiene algunas diferencias específicas, la primera de ellas es que lo que envía al ordenador no son las coordenadas cartesianas obtenidas a partir del modelo cinemático, sino más bien, la posición de cada una de las articulaciones en radianes, para que así se pueda hacer el control sobre cada una de estas, otra diferencia es que el programa de Arduino, ahora está preparado para recibir datos, a saber, 3 valores con un rango de 0 a 255, acompañados de un identificador que define la dirección del movimiento, todo en una cadena de texto que se recibe a través el puerto serial, en base a esto, y mediante el uso de un puente H basado en el chip L298N del cual se puede ver la hoja de datos en el Anexo B, se controla el movimiento y posición de los motores.

Para el manejo de dicho puente H se hace uso de dos puertos para definir el sentido del giro y uno más para, mediante un PWM suministrar un valor de voltaje de control que maneja la salida hacia el motor. La alimentación del puente H se estableció en 7V. El código fuente del Arduino puede ser visto en el Anexo C.

La interfaz creada en LabVIEW luce tal como se puede ver en la Figura 73.

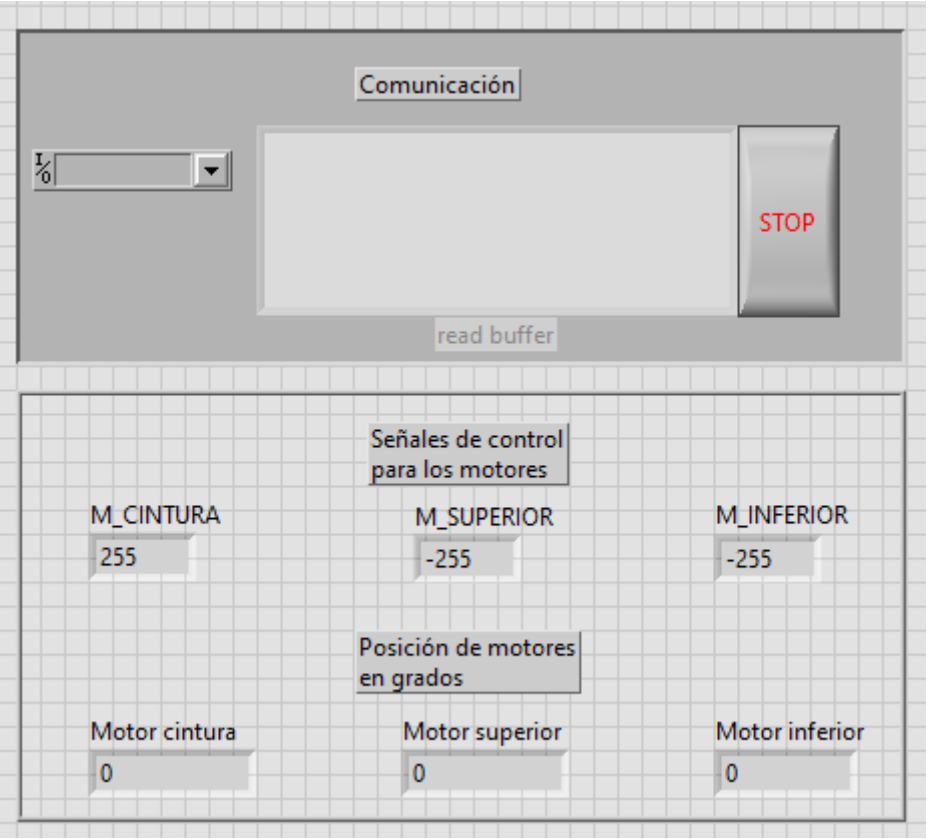


Figura 73 Interfaz gráfica del programa para el control de dispositivo háptico

La interfaz requiere que se le configure primeramente el puerto en el cual está la tarjeta de adquisición de datos, fuera de eso, la interfaz cuenta solo con indicadores, uno que muestra el buffer de datos que se está recibiendo, y el resto son indicadores que muestran las señales de control aplicadas a los motores y la posición actual de cada una de las articulaciones, todo lo cual funcionará en base al controlador que se le configure.

El diagrama a bloques se basa en dos estructuras anidadas dentro del ciclo principal, una de ellas está asignada al controlador, mientras que la otra de ellas está diseñada para activarse al recibir datos de la tarjeta de adquisición de datos, los cuales procesa para obtener las posiciones de las articulaciones y después enviar las señales de control que la tarjeta de adquisición de datos debe suministrar

a los motores, en la Figura 74, podemos ver el diagrama a bloques completo del programa.

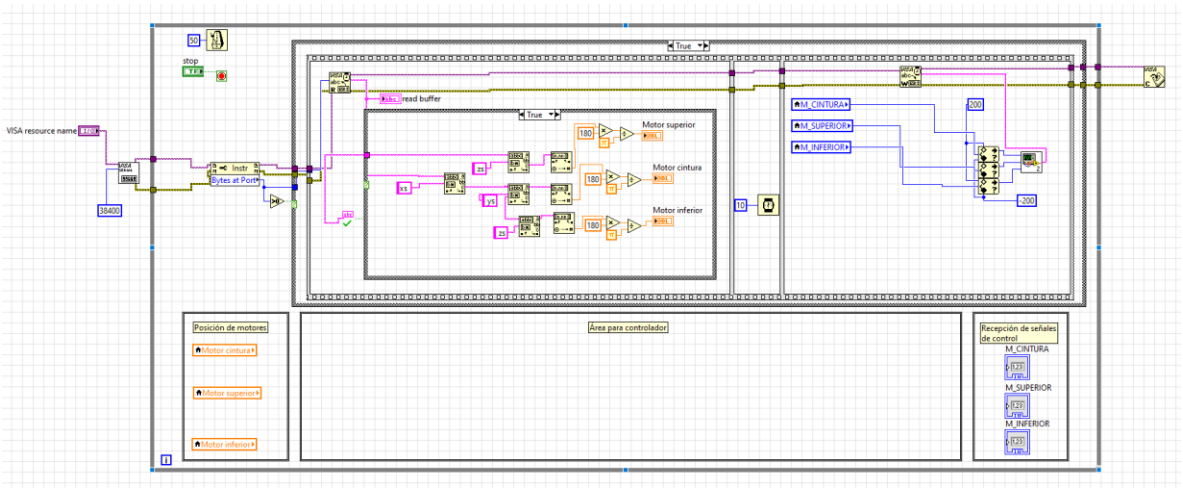


Figura 74 Diagrama a bloques del programa de adquisición de datos y control del dispositivo háptico

Dentro del programa encontramos dos sub VI's uno de los cuales se expuso antes, el cual es el encargado de verificar que la cadena recibida es una cadena sana, pero el segundo de ellos es un VI, que se encarga más bien de transformar tres números que van de -255 a 255, a una cadena con valores absolutos, donde el signo se sustituye con un identificador de dirección, "l" para izquierda y "r" par derecha. En la Figura 75 podemos ver el diagrama a bloques de este subVI.

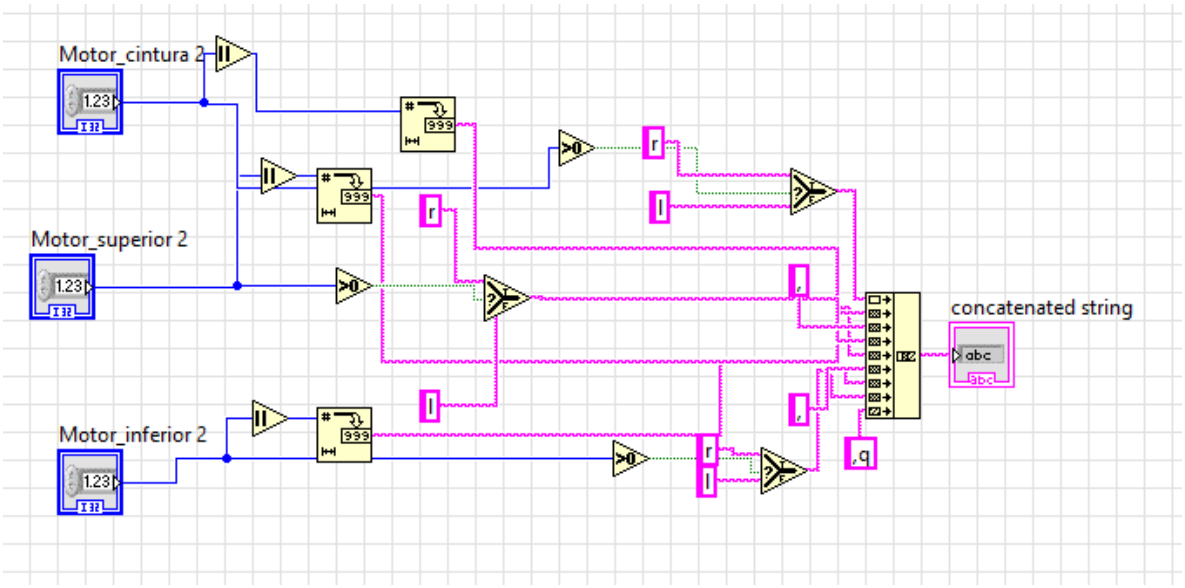


Figura 75 Diagrama a bloques de la rutina de composición del mensaje con las señales de control

El programa principal se actualiza cada 50 milisegundos y puede acelerarse en caso de ser necesario, el área para el controlador, con sus entradas y salidas se muestra en la Figura 76.

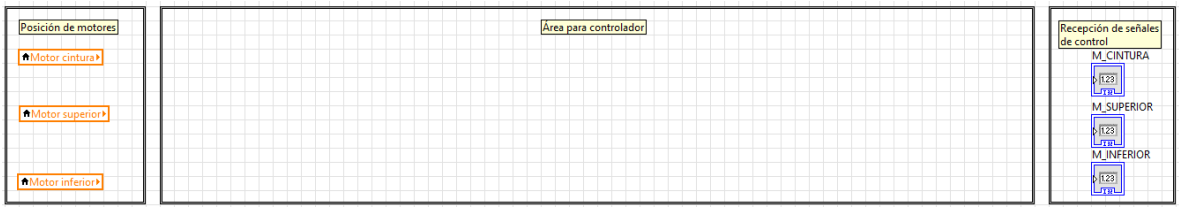


Figura 76 Área en el diagrama a bloques del programa para posicionar el controlador

Para asegurarnos de no tener que crear un nuevo módulo de control, los puentes H se pusieron junto con el Arduino de control dentro de un módulo con todos los puertos conectados, para que en caso de querer que funcione en modo control, solo sea necesario cambiar el programa de Arduino y LabVIEW usados.

Se hizo una prueba de control y el dispositivo siguió la referencia adecuadamente, incluso cuando no se hizo un proceso formal de sintonización sobre el controlador utilizado.

4.2.2. CONTROL DE UN MODELO 3D DEL DISPOSITIVO HÁPTICO MEDIANTE SIMMECHANICS

Como parte del proyecto, primero se hicieron algunas pruebas de control sobre un modelo en 3D del dispositivo háptico, el cual fue importado a Simulink usando Simmechanics, para después diseñar una rutina de control. Sobre este trabajo se publicó un artículo en la revista Pistas Educativas, del tecnológico de Celaya.

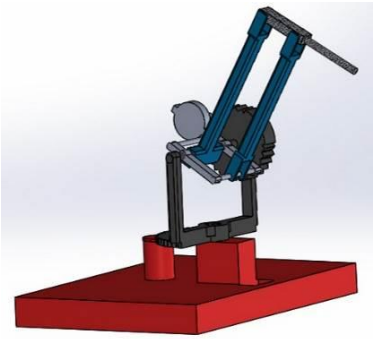


Figura 77 Modelo 3D del dispositivo háptico hecho en SolidWorks

Puesto que el modelo 3D ya estaba diseñado ya estaba hecho, pero tenía problemas importantes a la hora de hacer una rutina de control como tal, lo primero que se trabajó fue el añadir bloques que representaran la dinámica de funcionamiento de los motores, para que de esta forma se pudiera hacer una simulación del comportamiento de estos ante un controlador.

La función de transferencia se obtuvo en base a lo siguiente:

$$\frac{\Theta(s)}{E(s)} = \frac{\frac{K_T}{R_a}}{Js^2 + \left[\frac{BR_a + K_T K_b}{R_a}\right]s} \tag{27}$$

Donde:

Tabla 1 Parámetros del motor

Descripción del parámetro	Valor
$\Theta(s)$ *es la posición en grados*	--
$E(s)$ *es el voltaje aplicado*	--
K_T *es la constante de torque*	0.052
K_b *es la constante de fuerza contraelectromotriz*	0.052
J *es el momento de inercia del rotor*	11.6×10^{-5}
B *es el coeficiente de frincción viscosa*	2×10^{-6}
R_a *es la resistencia terminal*	10.61

El modelo final e Simulink quedó tal como lo muestra la figura 78, donde se puede apreciar que cada bloque de control corresponde al motor de cada articulación, este bloque recibe un escalón de diferente magnitud y es labor de controlador el hacer que el modelo 3D alcance esa referencia eficientemente.

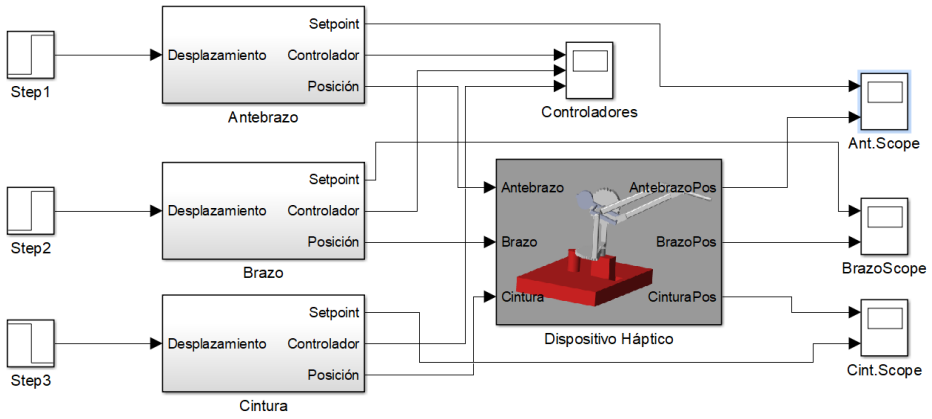


Figura 78 Diagrama a bloques general del sistema

Cada bloque está conformado como lo muestra la Figura 79.

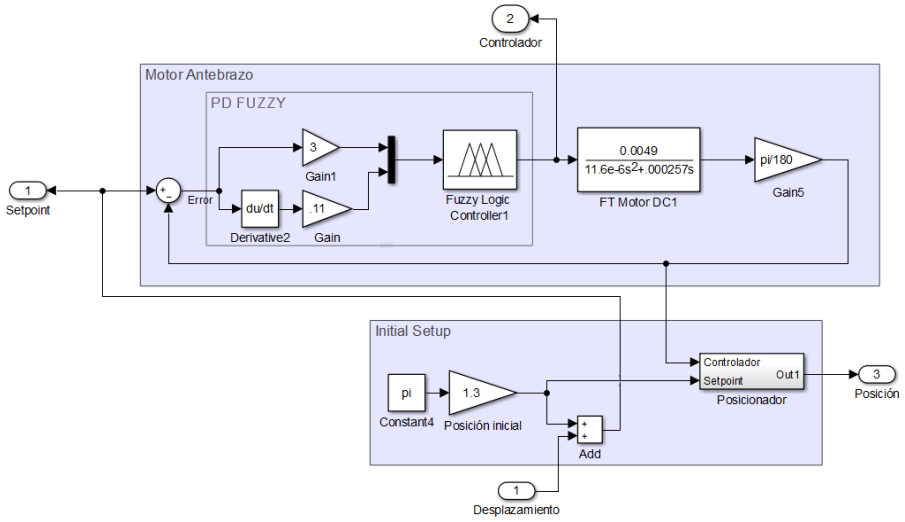


Figura 79 Diagrama a bloques del controlador y planta para los motores de las articulaciones

Cada articulación es controlada por un controlador PD difuso de 25 reglas, y tiene un diagrama a bloques especial para setear la articulación a una posición de inicial al arrancar cada simulación.

Tabla 2 Tarjeta de reglas del controlador PD difuso

SALIDA		ERROR				
		NB	NS	Z	PS	PB
DERROR	NB	NB	NM	Z	PM	PB
	NS	-	NS	Z	PS	-
	Z	NB	NM	Z	PM	PB
	PS	-	NS	Z	PS	-
	PB	NB	NM	Z	PM	PB

La tabla 2 muestra las reglas lógicas del controlador difuso y en el Anexo D se muestra las funciones de membresía de entrada y salida del mismo controlador.

El controlador PD difuso se comparó contra uno PID clásico sintonizado en MATLAB con ganancias de K_p=38.6, K_d=1.49y K_i=86.3, la comparativa se muestra en las figuras 61, 62 Y 63.

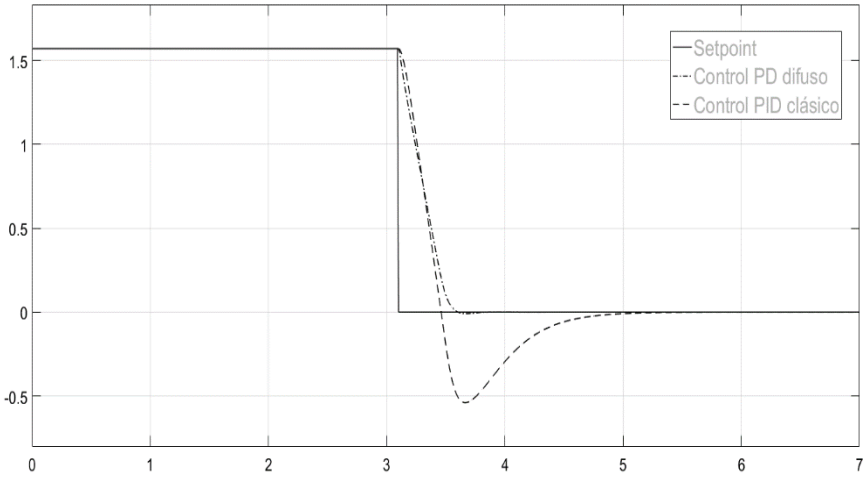


Figura 80 Respuesta de la cintura ante un desplazamiento de 0.5 radianes

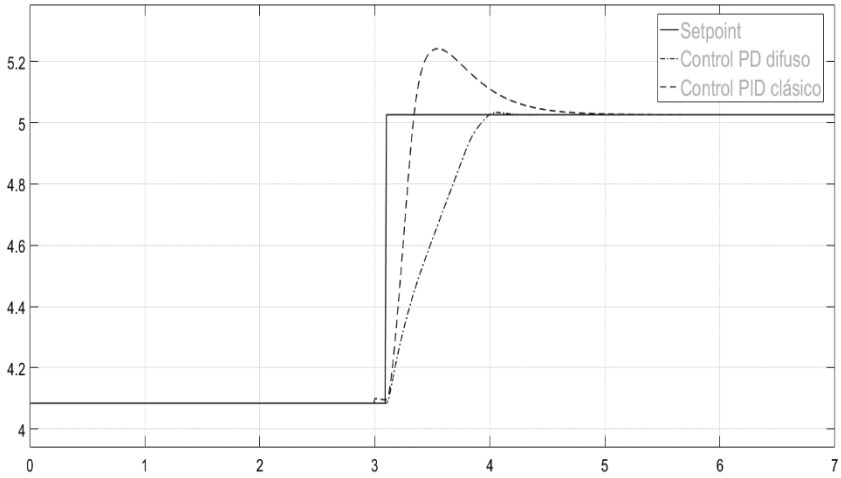


Figura 81 Respuesta del antebrazo ante un desplazamiento de 0.3 radianes

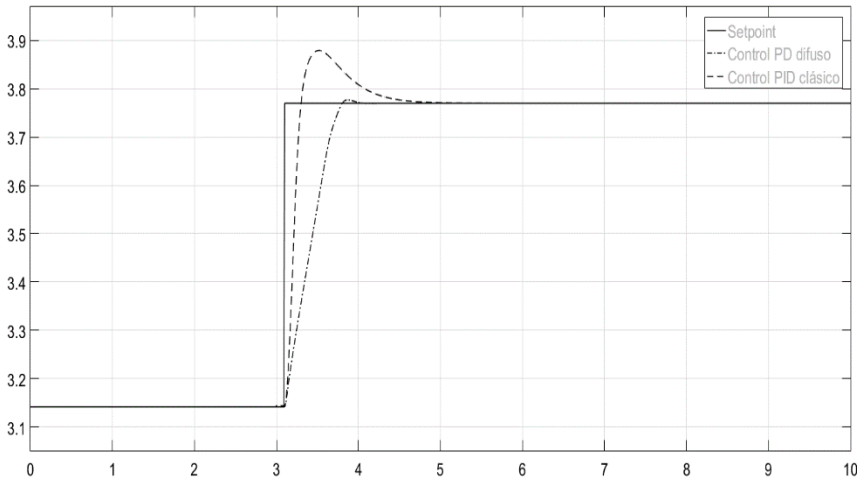


Figura 82 Respuesta del brazo ante un desplazamiento de 0.2 radianes

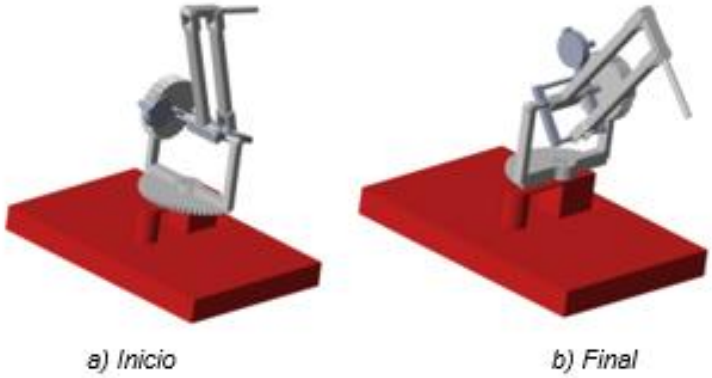


Figura 83 Posición inicial y final del dispositivo háptico

4.2.3. CONTROL PD+I DIFUSO DE UN DISPOSITIVO HÁPTICO PHANTOM OMNI

Durante el curso de la maestría tuve oportunidad de visitar la Universidad de Colima, plantel Manzanillo, donde hay un par de dispositivos háptico Phantom versión Omni (Figura 84), los cuales son la versión renovada del que se tiene en el ITCG.

Figura 84 Dispositivo háptico Phantom Omni

Con este dispositivo fue con el primero que se hizo un programa que permitiera el control de sus articulaciones, todo usando un paquete disponible dentro de Simulink de MATLAB, y el cual funciona únicamente con nuevas versiones del dispositivo háptico. El diagrama a bloques en Simulink se muestra en la Figura 85.

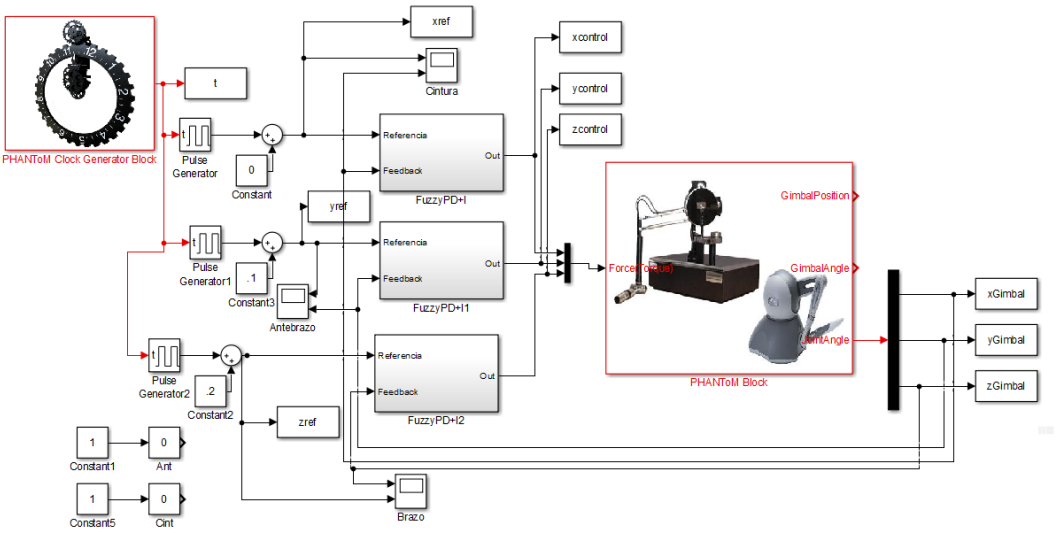


Figura 85 Diagrama a bloques en Simulink del sistema de control para el Phantom Omni

Como se observa, cada una de las articulaciones se maneja con un controlador diferente, en este caso, se propuso el uso de un controlador PD + I difuso de 25 reglas, del cual se muestra el diagrama a bloques en la Figura 86.

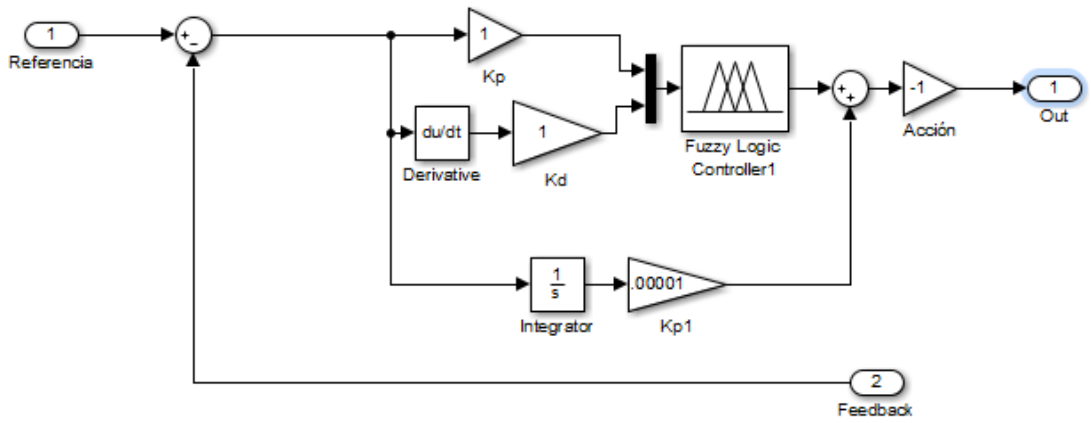


Figura 86 Diagrama a bloques del controlador PD+I difuso

Se hicieron pruebas para este controlador con señales de referencia como senos y señales cuadradas, los resultados de los trazos sinodales se muestran en la Figura 87.

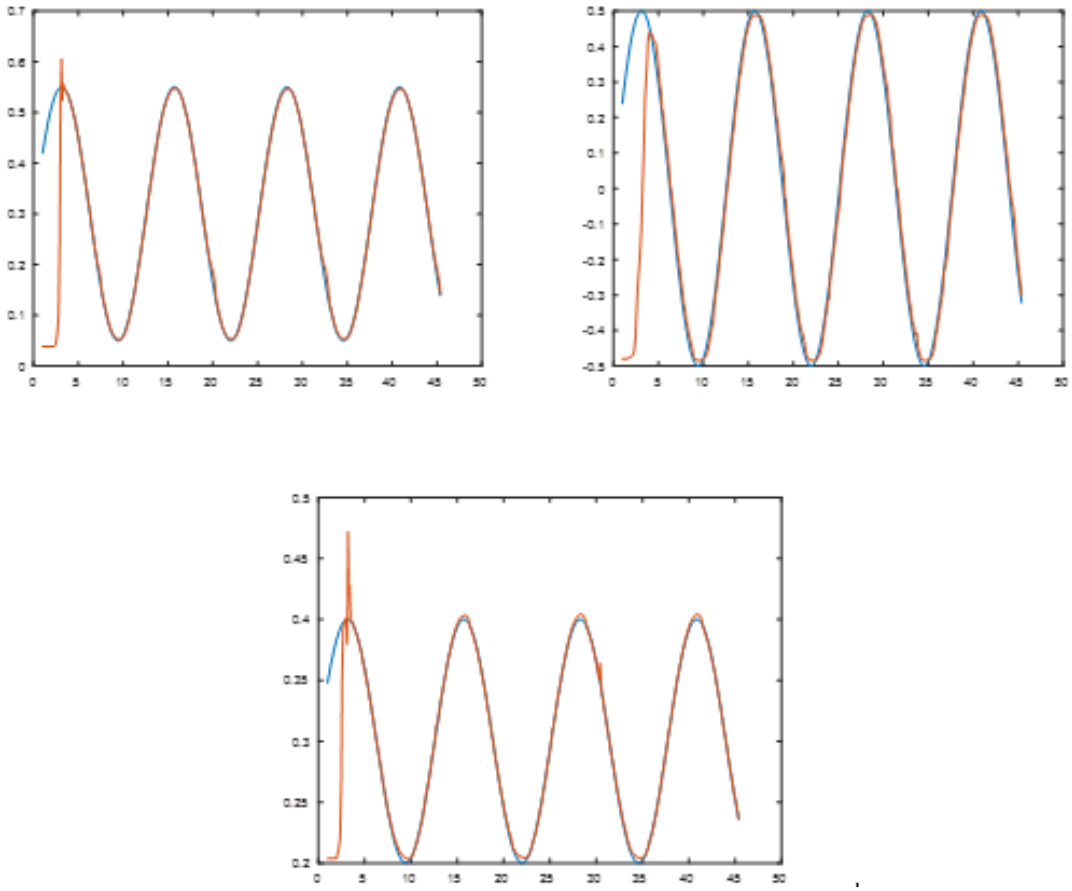


Figura 87 Gráficas del seguimiento de una trayectoria senoidal de las articulaciones con respecto a la referencia

Como se aprecia en las figuras, el seguimiento de la referencia en cada una de las articulaciones es bastante preciso, con pocos sobre impulsos de baja magnitud y un error de estado estable casi nulo.

4.2.4. CONTROL PD+I DIFUSO EN CASCADA PARA UN SISTEMA BALL AND BEAM

Mientras estuve haciendo estancia en la Universidad de Colima, una de las líneas de investigación en las que me involucré fue el diseño de un controlador PD difuso + I para una planta básica en temas de control, como lo es un sistema Ball and Beam. Toda la instrumentación fue de la marca Quanser, la planta funciona con una bola metálica que corre entre dos rieles de material conductor, lo cual permite que

mediante el movimiento de la pelota a través de las barras haya un cambio entre la resistencia medida entre ellas, así se obtiene una medición de distancia de la pelota.

El motor empleado tiene un arreglo de engranes que le permite subir uno de los extremos de la barra o bajarlo, de esta forma tiene la tarea de recibir la acción de control. Puesto que el motor es un sistema que requiere de control para poder ponerlo en posiciones específicas para subir o bajar las barras, es necesario tener un controlador de posición de motor y un controlador de posición de la pelota, los cuales funcionan en cascada. En la Figura 88, se observa el diagrama a bloques del sistema completo en cascada, donde se nota que la salida del controlador de la bola es recibida como entrada por el controlador del motor.

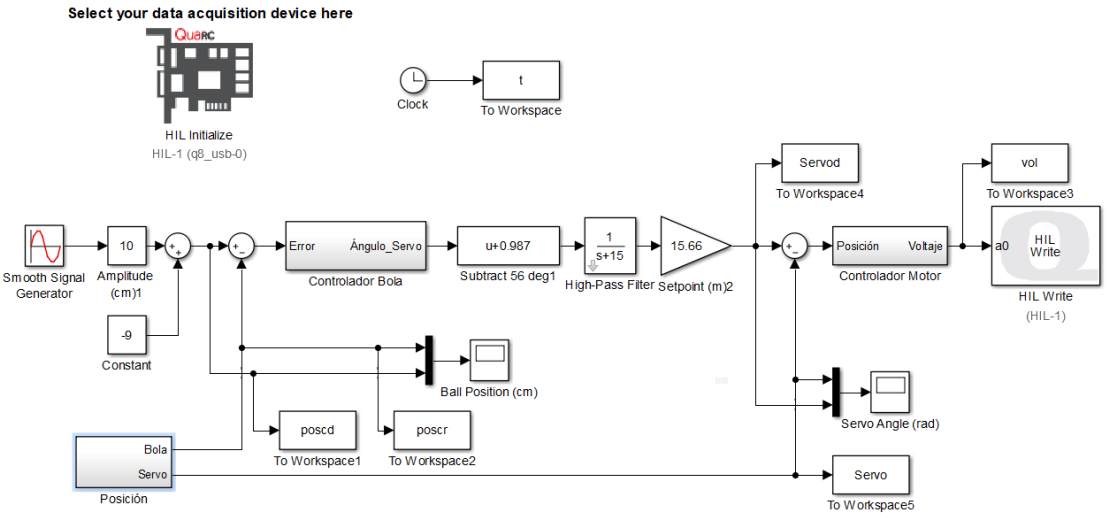


Figura 88 Diagrama a bloques del sistema de control en cascada con planta para un sistema Ball and Beam

Para el diseño del controlador de la bola se definió 40 cm de longitud de la barra por conveniencia, por lo que los errores máximos tanto positivos como negativos no habrían de rebasar esa magnitud, por lo que ±40 es el rango dentro del cual se encuentran todas las funciones de membresía del error. Las funciones de membresía consideran 5 posibles casos, un error negativo grande (NB), un error negativo pequeño (NS), un error positivo grande (PB), un error positivo pequeño (NS) y un error de cero (Z). Los rangos individuales de cada función de membresía

se definieron considerando la dinámica del sistema, especialmente en base a la velocidad de la bola y el efecto de la inercia sobre ella, ya que, por ejemplo, al pasar de un valor de error positivo grande a uno pequeño se espera que la bola se desacelere lo suficiente para no permitir que se pase de la referencia, lo cual será imposible si comenzamos a desacelerar muy cerca de ella. Mediante un poco de experimentación se llegó a las funciones de membresía mostradas en la figura 89.

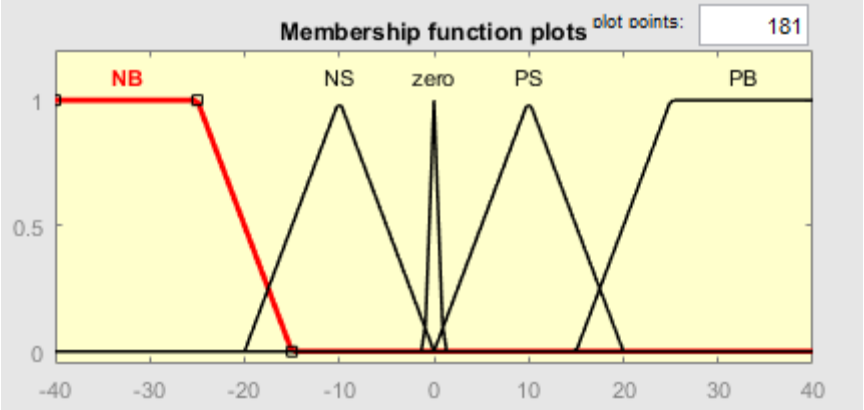


Figura 89 Funciones de membresía para la entrada de error de posición de la bola

El controlador funciona considerando también la derivada del error, la cual describe en sí la velocidad que la bola adquiere, por lo que en este apartado podemos decir que podemos controlar la velocidad máxima que se le permite a la bola, las funciones de membresía consideran 5 casos, todos dentro de un rango de ±1 como muestra la figura 90.

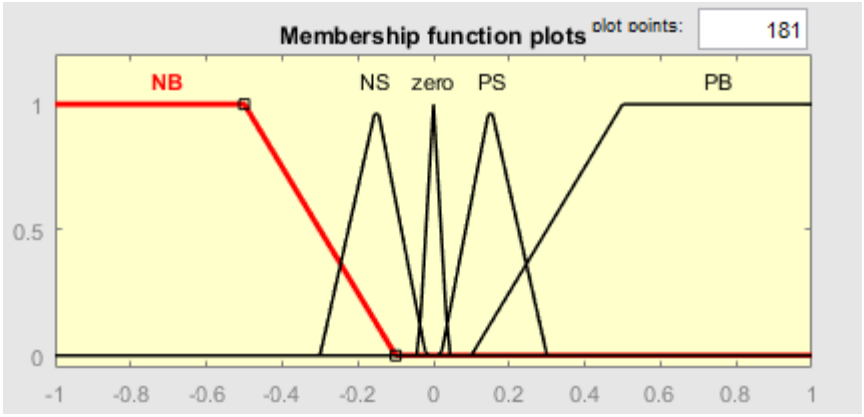


Figura 90 Funciones de membresía para la entrada de la derivada del error de posición de la bola

La salida del controlador se diseñó para trabajar en un rango de ±45°, rango que se dividió entre 7 casos posibles, que podemos ver en la figura 91.

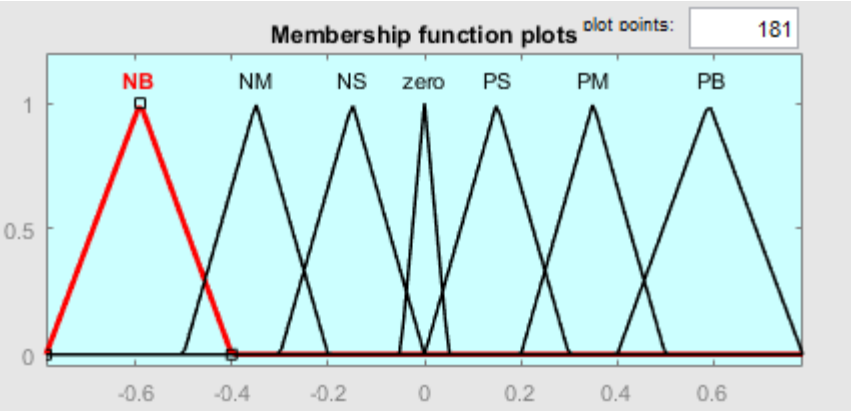


Figura 91 Funciones de membresía de la salida de control (en radianes)

La tabla 3 muestra las reglas que relacionan las funciones de membresía de entrada con las de salida.

Tabla 3 Reglas del controlador PD difuso

SALIDA		ERROR				
		NB	NS	Z	PS	PB
DERROR	NB	NB	NM	NM	NS	PS
	NS	NB	NM	Z	PS	PB
	Z	NB	NM	Z	PM	PM
	PS	NB	NS	PS	PM	PB
	PB	Z	PS	PM	PM	PB

Por último, se agregó la parte integrativa del controlador, multiplicando la integral de error por una ganancia y sumándola a la acción de control proporcionada por el controlador difuso. La figura 92 muestra la estructura final del controlador PD+I.

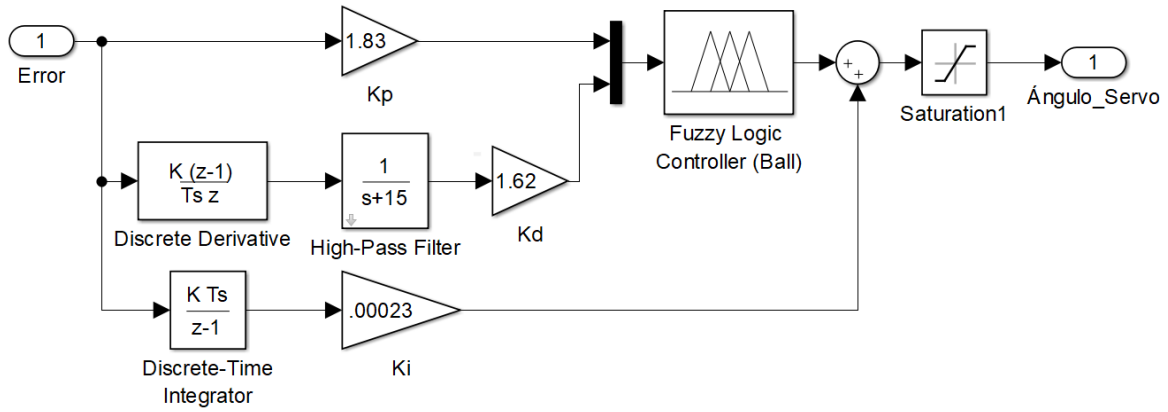


Figura 92 Diagrama a bloques del controlador PD difuso + I

Para el controlador de motor se utilizó como base el controlador PD difuso para un motor CD propuesto en el trabajo de (Cortés, 2021), a este controlador fue necesario modificarle los rangos de las funciones de membresía de entrada y salida para trabajar con los valores de ángulo de entrada y voltaje de salida requeridos de ±45° y ±12 V respectivamente. También se usó una ganancia de valor -1 a la salida de la saturación para invertir la acción de control, además de la suma de la integral del error a la salida del controlador difuso para convertirlo en un controlador tipo PD+I difuso, el cual sigue la estructura mostrada en la figura 7, pero cambia en los valores de las ganancias. Las reglas de este controlador se muestran en la tabla 4.

Tabla 4 Reglas del controlador PD difuso para un motor CD propuesto por (Cortés, 2021)

Tabla 4		ERROR				
SALIDA		NB	NS	Z	PS	PB
DERROR	NB	NB	NM	Z	PM	PB
	NS	x	NS	Z	PS	x
	Z	NB	NM	Z	PM	PB
	PS	x	NS	Z	PS	x
	PB	NB	NM	Z	PM	PB

Las pruebas se realizaron con una señal de onda cuadrada de 20 cm de amplitud con un periodo de 30 segundos, lo que permitiría ver la capacidad del sistema para mantenerse estable, también se hicieron pruebas con una señal senoidal de 20 cm de amplitud con un periodo de 20 segundos.

Señal Cuadrada

Los resultados del experimento con la señal cuadrada se muestran en la figura 93. En este experimento no se observó un sobre impulso perceptible, el tiempo de estabilización fue de 4 a 5 segundos y el error de estado estable mayor fue no rebasó nunca 1 mm lo cual representa el 0.5% de la señal. La curva obtenida como resultado de la acción de control es una curva suave y sin oscilaciones.

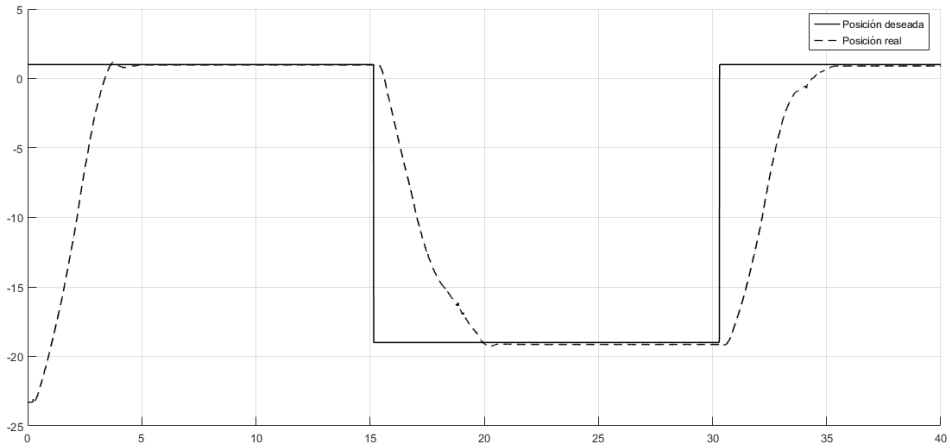


Figura 93 Respuesta del controlador de posición de la bola a una señal cuadrada

La figura 94 muestra la respuesta del controlador encargado del funcionamiento del motor durante la ejecución del programa, como es visto, la referencia se alcanza con una alta precisión a pesar de los bruscos cambios que existen en la señal. Con un tiempo de estabilización que va de los 100 a los 200 milisegundos dependiendo del cambio en la referencia, podemos decir que es un controlador rápido, pero además también es bastante preciso, pues lleva el error de estado estable a un valor máximo de .05 y no presenta sobre-impulsos ni oscilaciones.

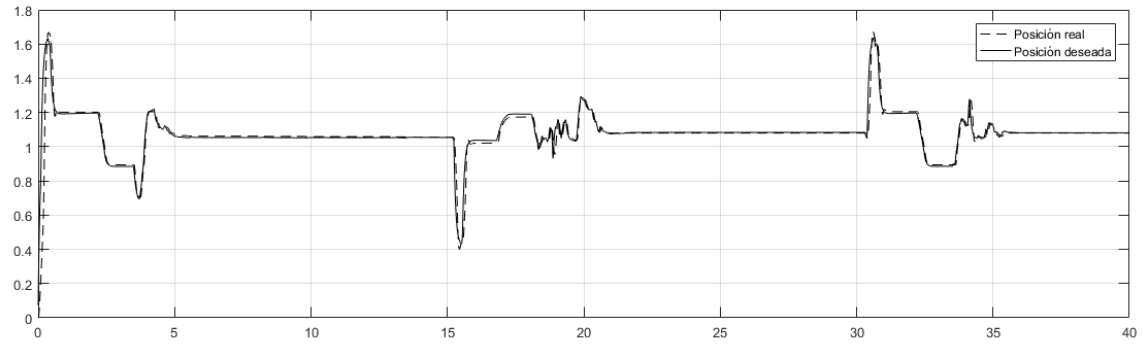


Figura 94 Respuesta del controlador de posición del motor

Señal Senoidal

Los resultados del seguimiento de una trayectoria senoidal por el sistema se muestran en la figura 95, como se puede apreciar, el seguimiento de la referencia es bastante preciso, sin embargo, en el punto inferior de la onda senoidal se percibe una singularidad en el error, la cual a pesar de ser notoria no va más allá de 2 milímetros. La señal se mantiene estable, no muestra oscilaciones y el error promedio no excede los 0.7mm.

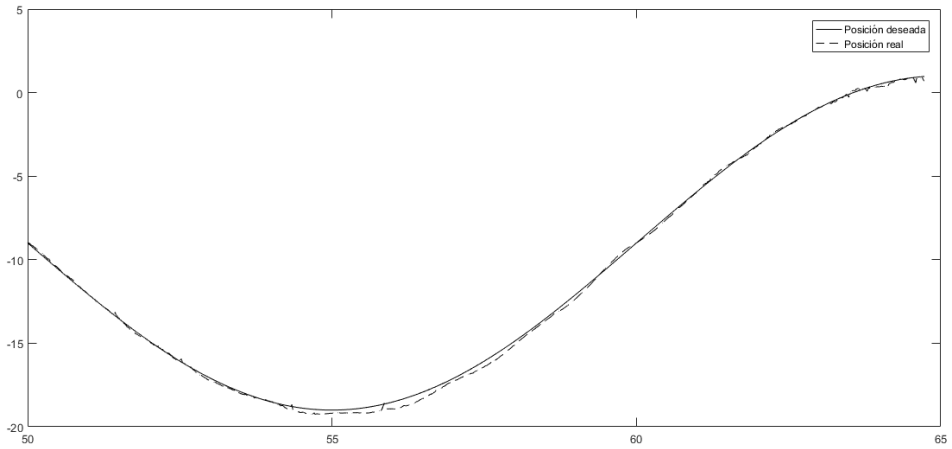


Figura 95 Respuesta del controlador de posición de bola a una señal senoidal

En la figura 96 tenemos la gráfica que describe el comportamiento del segundo controlador, el cual transforma un ángulo en niveles de voltaje para mantenerse en

la referencia. Esta gráfica revela lo robusto que es el controlador, además de lo rápido que consigue compensar las variaciones en la señal de referencia y la precisión que mantiene durante todo el tiempo del experimento. A diferencia de la señal cuadrada, la señal senoidal requiere de ajustes constantes durante casi todo el tiempo de ejecución, por lo que la carga sobre el controlador es mayor, aun así, el controlador se comportó a la altura de lo esperado.

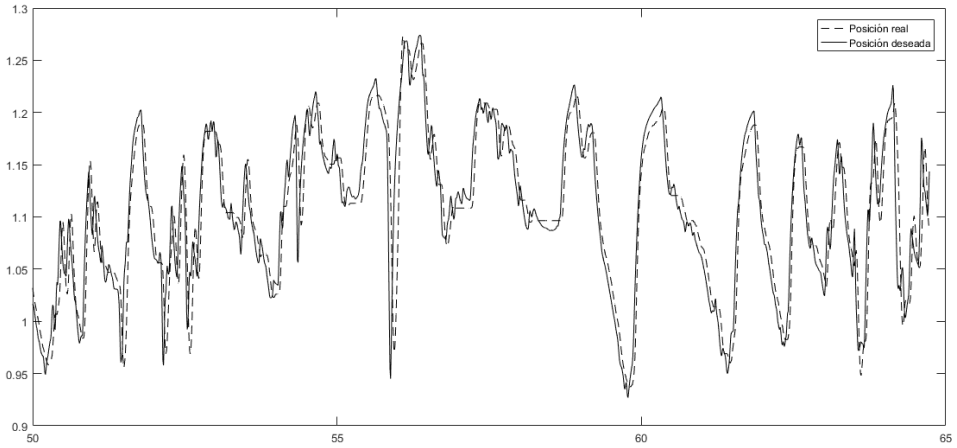


Figura 96 Respuesta del controlador de posición del motor

CAPÍTULO V. CONCLUSIONES

El proceso de reingeniería expuesto en el presente trabajo resultó estar dentro de lo esperado a pesar de ser tecnología antigua, si bien podría haber sido una posibilidad el repararlo, darle más capacidad con tecnología contemporánea fue la mejor opción. El sistema de adquisición demostró ser capaz de entregar datos precisos a una velocidad que si bien ya es bastante rápida aún no alcanza el límite al que se puede llegar, en un trabajo posterior sería importante explorar la posibilidad de hacer la adquisición de datos mediante el uso de rutinas más especializadas y usando hardware de mejores características, como una tarjeta de Raspberry o de National Instruments. Su puede decir que el sistema en general demostró tener un comportamiento estable y robusto, sin ser demasiado costoso, lo cual sin duda alguna es muy importante cuando se propone generar tecnología que pueda ser accesible al mayor número de personas y en la mayoría del mundo.

En cuanto al diseño de la red neuronal con que se trabajó, su eficiencia llegó a cerca de un 98% siendo capaz de diferenciar entre trazos de una sola trayectoria y trazos de varias trayectorias en un primer acercamiento, y entre trazos de una o más trayectorias circulares y una o más trayectorias cuadrangulares. Si bien esto aún no llega al grado de diagnosticar el mal de Parkinson nos da la seguridad que la red es funcional para detectar variaciones mínimas entre los trazos, sin embargo, la eficiencia va a depender de la cantidad de muestras que se puedan obtener para el entrenamiento, y también de que las trayectorias sean bien definidas y estandarizadas, de manera que no haya demasiadas variables involucradas, lo cual a su vez permitirá entrenar una red neuronal especializada. Cabe señalar que como optimizador se utilizó uno de Adam, el cual dio resultados lo suficientemente buenos, sin embargo, a la hora de detectar variaciones más pequeñas es de gran importancia utilizar otro tipo de optimizadores con mayores grados de eficiencia, tal como lo es el optimizador tipo SGD.

Se debe señalar que los trazos que se obtengan deben ser propiamente procesados entes de ser suministrados como datos de entrenamiento a la red neuronal, hacer procesos de convolución previos al procesamiento dentro de la red nos permite aprovechar los recursos del ordenador para generar una red neuronal de mejores proporciones, ya que en entornos de ejecución gratuitos como lo es COLAB, no disponemos de recursos ilimitados, y cuando intentamos hacer entrenamientos con imágenes de alta resolución y de varios colores, nos encontramos con que los recursos no son suficientes. Utilizar código de diferentes entornos es muy práctico pues tomamos lo mejor de cada una de estas plataformas, sin embargo, sería muy práctico hacer una consolidación de todas las funciones en un solo lenguaje de programación cuando se busque tener una plataforma universal completa para este fin.

Python se presenta como la mejor opción para hacer la actividad, debido a la facilidad de usos y sus muchos recursos. En un trabajo posterior será una buena idea involucrar otro tipo de matrices de convolución, las cuales permitan incrementar la definición de los trazos o resaltar las partes que hacen la diferencia entre trazos y trazos.

Algo que se notó en base al último de los ejercicios donde se emuló la rigidez, es que para detectar las variaciones, es necesario hacer otro tipo de tratamiento a las señales, uno que no reduzca demasiado su figura, ya que esto causa que la resolución de las oscilaciones sea mínima y se pierdan, lo cual confunde a la red neuronal, que si bien detecta estas oscilaciones, no puede decir con un grado elevado de certeza si es un trazo suave o no.

Lo más importante para seguir con esta iniciativa es el comenzar con la adquisición de trayectorias hechas por personas con diferentes etapas del mal de Parkinson, esto para crear una base de datos para hacer el entrenamiento de una red neuronal lo suficientemente poderosa para percibir la más mínima variación en los trazos. Utilizar tecnologías emergentes y de mayores capacidades como Machine Leargning sin duda alguna serán los estandartes del avance de esta iniciativa. Sin embargo, antes de todo se debe abordar a especialistas para deliberar y definir una

trayectoria estándar que sea lo suficientemente representativa para identificar variaciones en la rigidez y temblor de los brazos de los pacientes.

En base a lo anteriormente expuesto recomiendo seguir con este trabajo posteriormente pues puede ser de mucha ayuda y la tecnología en la que se basa está en constante avance, lo cual hace que las posibilidades de tener éxito sean altas.

Bibliografía

PYPL PopularitY of Programming Language index. (14 de Marzo de 2017). Obtenido de https://pypl.github.io/PYPL.html

Arduino Official Page. (10 de Junio de 2022). Obtenido de https://arduino.cl/producto/arduino-mega-2560/

Axelson, J. (2007). *Serial Port Complete COM Ports, USB Virtual COM Ports, and Ports for Embedded Systems.* Chinook Ln., Madison: Lakeview Research LLC.

Batista, F. (Junio de 2020). *Python Enhancement Proposals. Python Software Foundation.* Obtenido de https://www.python.org/dev/peps/pep-0327/

Collazo, N. M. (1991). *Manual del Sistema Internacional de Unidades.* La Habana, Cuba: Editorial Pueblo y Educación.

Craig, J. J. (2006). *Robótica.* México: Pearson Prentice Hall.

Fernando Reyes Cortés, J. C. (2015). *Arduino. Aplicaciones en robótica, mecatrónica e ingenierias.* México D.F.: Alfaomega.

Gottlieb, I. M. (1994). *Electric Motors & Control Techniques.* TAB Books.

Guru Bhag, H. H. (2001). Electric Machinery and Transformers. *The Oxford series un electrical and computer engineering*.

Guttag, J. V. (2016). Introduction to Computation and Programming Using Python: With Application to Understanding Data. *MIT Press*.

Hayt, W. (2007). *Análisis de Circuitos en Ingeniería.* México: McGraw-Hill.

Isidro Ramos Salavert, M. D. (2000). Ingeniería del software y bases de datos: tendencias actuales. *Entornos de Desarrollo Integrados*, 78.

Jan Aaseth, P. D. (2018). Prevention of progression in Parkinson's disease. *Biometals*, 737-747.

José M. Troya, A. V. (s.f.). Desarrollo de Software Basado en Componentes. *Lenguajes y Ciencias de la Computación. Universidad de Málaga*.

Kuhlman, D. (2012). *A Python Book: Beginning Python, Advanced Python, and Python Exercises.*

Larsen, R. W. (2011). *LabVIEW for Engineers.* New Jersey: Prentice Hall.

Marios Politis, K. W. (2010). Parkinson's Disease Symptoms: The Patient's Perspective. *Movement Disorders*, 1646-1651.

Matich, D. J. (2001). *Redes Neuronales: Conceptos Básicos y.* Universidad Tecnológica Nacional – Facultad Regional Rosario.

McMillan, G. K. (1999). *Process Instruments and Controls Handbook Fifth Edition.* McGraw Hill.

Myths about indentation in Python. (18 de Febrero de 2018). Obtenido de https://web.archive.org/web/20180218162410/http://www.secnetix.de/~olli/Python/block_indentation.hawk

N. Margiotta, G. A. (2016). A Wearable Wireless System for Gait Analysis for Early Diagnosis of Alzheimer and Parkinson Disease. *5th International Conference on Electronic Devices, Systems and Applications*, 1-4.

Olmedo, S. (2012). *MANUAL DE CINEMÁTICA Y DINÁMICA.* Ecuador: Editorial Universitaria Abya-Yala.

Omid Bazgir, S. A. (2015). A Neural Network System for Diagnosis and Assessment of Tremor in Parkinson Disease Patients. *Iranian Conference on Biomedical Engineering*.

Pagán, F. L. (2012). Improving Outcomes Through Early Diagnosis of Parkinson's Disease. *American Journal of Managed Care*.

Peters, T. (26 de Diciembre de 2018). *PEP 20 – The Zen of Python.* Obtenido de https://www.python.org/dev/peps/pep-0020/

Roberta Balestrino, A. H. (2020). PARKINSON DISEASE. *European journal of neurology*, 27-42.

Spasojević, S. I.-V. (2017). Quantitative Assessment of the ArmHand Movements in Parkinson. *Frontiers in neurology*, 388.

The State of Developer Ecosystem in 2020 Infographic. (1 de Marzo de 2021). Obtenido de JetBrains: https://www.jetbrains.com/lp/devecosystem-2020/

Tippens, P. E. (2007). *Fisica Conceptos y aplicaciones.* México, DF: McGraw Hill.

Vizcaíno, J. R. (2011). *LabView: entorno gráfico de programación.* Marcombo.

W. Tao, T. L. (2012). Gait analysis using wearable sensors. *Sensors*, 2255-2283.

Wile, D. J. (2014). Smart watch accelerometry for analysis and diagnosis of tremor. *Journal of neuroscience methods*, 1-4.

Williams, A. (2002). *Microcontroller projects using the Basic Stamp.* Focal Press.

Young, F. (2009). *Física Universitaria.* México: Pearson Educación.

ANEXOS

Anexo A

HEDM-55xx/560x & HEDS-55xx/56xx

Quick Assembly Two and Three Channel Optical Encoders

AVAGO TECHNOLOGIES

Data Sheet

HEDM-55xx/560x HEDS-550x/554x, HEDS-560x/564x

Description

The HEDS-5500/5540, HEDS-5600/5640, HEDM-5500/5540 and HEDM-5600 are high performance, low cost, two and three channel optical incremental encoders. These encoders emphasize high reliability, high resolution, and easy assembly.

Each encoder contains a lensed LED source, an integrated circuit with detectors and output circuitry, and a codewheel which rotates between the emitter and detector IC. The outputs of the HEDS-5500/5600 and HEDM-5500/ 5600 are two square waves in quadrature. The HEDS-5540/5640 and HEDM-5540 also have a third channel index output in addition to the two channel quadrature. This index output is a 90 electrical degree, high true index pulse which is generated once for each full rotation of the codewheel.

The HEDS series utilizes metal codewheels, while the HEDM series utilizes a film codewheel allowing for resolutions to 1024 CPR.

These encoders may be quickly and easily mounted to a motor. For larger diameter motors, the HEDM-5600, and HEDS-5600/5640 feature external mounting ears.

The quadrature signals and the index pulse are accessed through five 0.025 inch square pins located on 0.1 inch centers.

Standard resolutions between 96 and 1024 counts per revolution are presently available. Consult local Avago sales representatives for other resolutions.

Features

- Two channel quadrature output with optional index pulse
- Quick and easy assembly
- No signal adjustment required
- External mounting ears available
- Low cost
- Resolutions up to 1024 counts per revolution
- Small size –40°C to 100°C operating temperature
- TTL compatible
- Single 5 V supply

Applications

The HEDS-5500, 5540, 5600, 5640, and the HEDM-5500, 5540,5600 provide motion detection at a low cost, making them ideal for high volume applications. Typical applications include printers, plotters, tape drives, positioning tables, and automatic handlers.

Note: Avago Technologies encoders are not recommended for use in safety critical applications. Eg. ABS braking systems, power steering, life support systems and critical care medical equipment. Please contact sales representative if more clarification is needed.

L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)

Multiwatt15

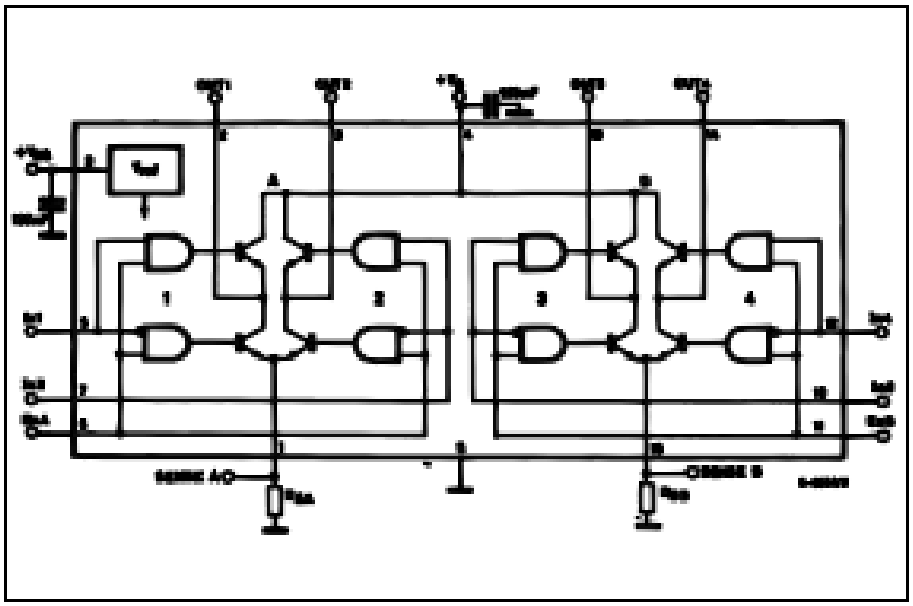
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

BLOCK DIAGRAM



Anexo C

```
// Variable global de velocidad
volatile int cuenta = 0, cuenta1 = 0, cuenta2 = 0;
volatile int flag = 0, flag1 = 0, flag2 = 0;
double teta1 = 0, teta2 = 0, teta3 = 0;
String t1 = "", t2 = "", t3 = "";
String xs = "", ys = "", zs = "";
double x = 0, y = 0, z = 0, l = 14.351, p = 3.14159265359;
bool finCadena = false;
String cadenaCharEntrada = "";
char d1 = 'n', d2 = 'n', d3 = 'n';
int m1 = 0, m2 = 0, m3 = 0;
//Puertos de control de sentido de giro de los motores
int a1 = 22, b1 = 23, a2 = 24, b2 = 25, a3 = 26, b3 = 27;
//Puertos de control de velocidad de los motores
int enA1 = 7;
int enA2 = 12;
int enA3 = 45;

void setup() {
  attachInterrupt(digitalPinToInterrupt(2), tetamotorinf, RISING);
  attachInterrupt(digitalPinToInterrupt(20), tetamotorsuperior, RISING);
  attachInterrupt(digitalPinToInterrupt(21), tetacintura, RISING);
  Serial.begin(38400);
  //Declaracion de tipo de puertos
  pinMode(enA1, OUTPUT);
  pinMode(enA2, OUTPUT);
  pinMode(enA3, OUTPUT);
  pinMode(a1, OUTPUT);
  pinMode(b1, OUTPUT);
  pinMode(a2, OUTPUT);
  pinMode(b2, OUTPUT);
  pinMode(a3, OUTPUT);
  pinMode(b3, OUTPUT);
  //Asignació de PWM en puertos 45,12 y 7 a 31,333Hz
  TCCR5B = TCCR5B & 0b000 | 0x01;
  TCCR4B = TCCR4B & 0b000 | 0x01;
  TCCR1B = TCCR1B & 0b000 | 0x01;
  //Estado inicial- Motores apagados
  digitalWrite(enA1, LOW);
  digitalWrite(enA2, LOW);
  digitalWrite(enA3, LOW);
}
```

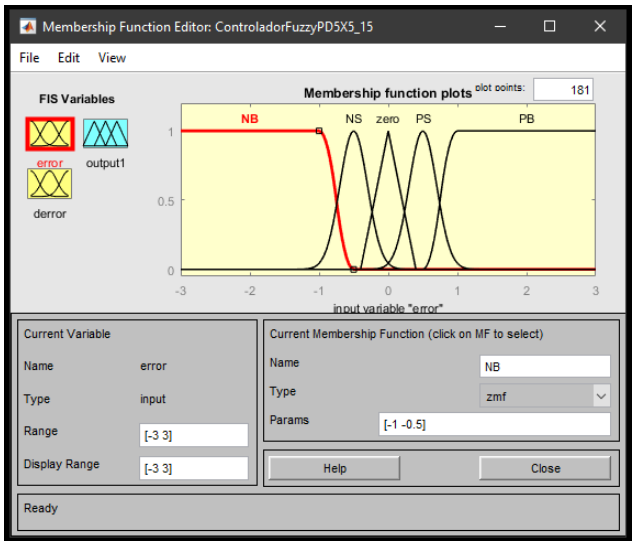
```
void loop() {
  delay(50);
  teta2 = (cuenta * .04804805 * p) / 180;
  teta3 = (cuenta1 * .04804805 * p) / 180;
  teta1 = (cuenta2 * .04285722 * p) / 180;
  t1 = String(teta1, 2);
  t2 = String(teta2, 2);
  t3 = String(teta3, 2);
  x = sin(teta1) * ((l * cos(teta2)) + (l * sin(teta3)));
  y = l * sin(teta2) + (l * (1 - cos(teta3)));
  z = (l * ((cos(teta1) * cos(teta2)) - 1)) + (l * (cos(teta1) *
sin(teta3)));
  xs = String(x, 2);
  ys = String(y, 2);
  zs = String(z, 2);
  /*analogWrite(11, m1);
  analogWrite(12, m2);
  analogWrite(13, m3);*/
  Serial.println("xs " + t1 + " ys " + t2 + " zs " + t3 + " u " + d1 +
String(m1) + ',' + d2 + String(m2) + ',' + d3 + String(m3) + ',');
}
void serialEvent() {
  cadenaCharEntrada = "";
  while (Serial.available()) {              //Si existen datos seriales,
leer a todos
    char CaracterEntrada = Serial.read();   //Leer 1 byte serial recibido

    cadenaCharEntrada += CaracterEntrada;   //Agregar el nuevo char a una
cadena String
    if (CaracterEntrada == 'q') {          //Si el char o byte recibido es
un fin de linea, activa la bandera
      finCadena = true;                        //Si la bandera finCadena =
1, entonces la transmision esta completa
    }

  }
  while (Serial.available()) {
    Serial.read();
  }
  verificacion(cadenaCharEntrada);
}
void output(){
  analogWrite(enA1, m1);
  if(d1 == 'l'){
    digitalWrite(a1,HIGH);
```

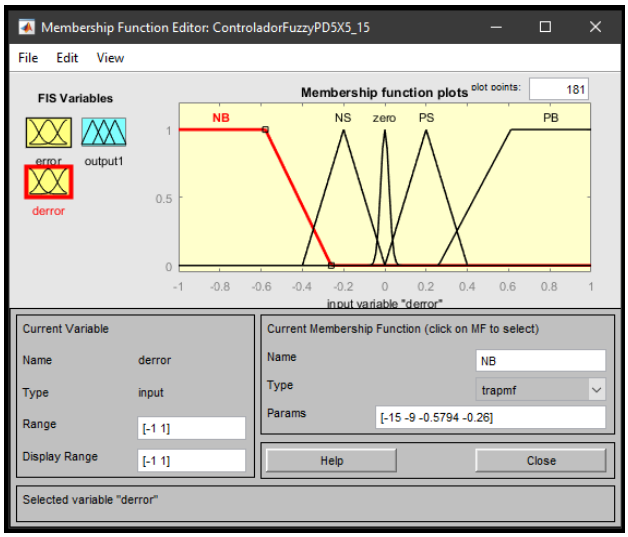
```
    digitalWrite(b1,LOW);
  }
  if(d1=='r'){
    digitalWrite(a1,LOW);
    digitalWrite(b1,HIGH);
  }
  analogWrite(enA2, m2);
  if(d2 == 'l'){
    digitalWrite(a2,HIGH);
    digitalWrite(b2,LOW);
  }
  if(d2=='r'){
    digitalWrite(a2,LOW);
    digitalWrite(b2,HIGH);
  }
  analogWrite(enA3, m3);
  if(d3 == 'l'){
    digitalWrite(a3,HIGH);
    digitalWrite(b3,LOW);
  }
  if(d3=='r'){
    digitalWrite(a3,LOW);
    digitalWrite(b3,HIGH);
  }
}
void verificacion(String cadenaEntrada) {
  int str_len = cadenaEntrada.length() + 1;
  int contlr = 0, contcomas = 0;
  char ch[str_len];
  cadenaEntrada.toCharArray(ch, str_len);
  for (int y = 0; y < str_len; y++) {
    if (ch[y] == 'l' || ch[y] == 'r') {
      contlr = contlr + 1;
    }
    if (ch[y] == ',') {
      contcomas = contcomas + 1;
    }
  }
  if (str_len > 8 && contlr == 3 && contcomas == 3) {
    tratamiento_string(cadenaEntrada);
  }
}
void tratamiento_string(String g) {
  String motor1 = "", motor2 = "", motor3 = "";
  int f1 = 0, conteo_motor = 0;
```

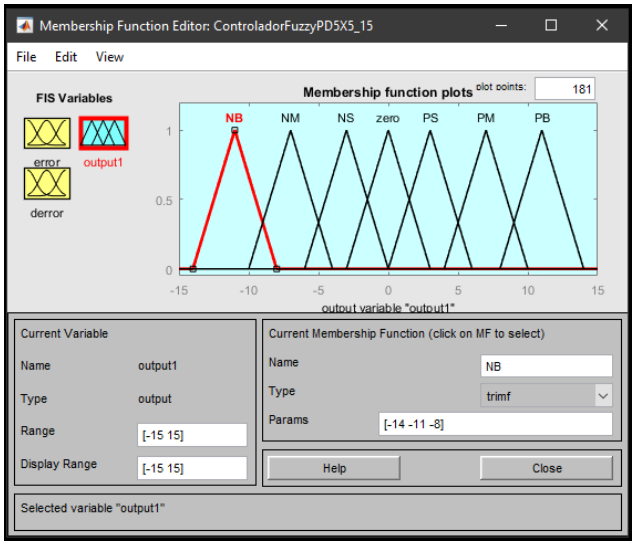
```
  int str_len = g.length() + 1;
  char char_array[str_len];
  g.toCharArray(char_array, str_len);
  for (int n = 0; n < str_len; n++) {
    if (char_array[n] == ',') {
      for (int j = f1; j < n; j++) {
        if (conteo_motor == 0) {
          motor1 = String(motor1 + String(char_array[j]));
        }
        if (conteo_motor == 1) {
          motor2 = String(motor2 + String(char_array[j]));
        }
        if (conteo_motor == 2) {
          motor3 = String(motor3 + String(char_array[j]));
        }
      }
      conteo_motor = conteo_motor + 1;
      f1 = n + 1;
    }
  }
  char chm1[motor1.length() + 1];
  motor1.toCharArray(chm1, motor1.length() + 1);
  char chm2[motor2.length() + 1];
  motor2.toCharArray(chm2, motor2.length() + 1);
  char chm3[motor3.length() + 1];
  motor3.toCharArray(chm3, motor3.length() + 1);
  d1 = chm1[0];
  d2 = chm2[0];
  d3 = chm3[0];
  int len1 = motor1.length(), len2 = motor2.length(), len3 =
motor3.length();
  motor1 = "";
  motor2 = "";
  motor3 = "";
  for (int n = 1; n < 4; n++) {
    if (n <= len1) {
      motor1 = String(motor1 + String(chm1[n]));
    }
    if (n <= len2) {
      motor2 = String(motor2 + String(chm2[n]));
    }
    if (n <= len3) {
      motor3 = String(motor3 + String(chm3[n]));
    }
  }
```

```
  m1 = motor1.toInt();
  m2 = motor2.toInt();
  m3 = motor3.toInt();
  output();
}
void tetamotorinf()
{
  // Disminuimos el valor establecido
  flag = digitalRead(9);
  if (flag == HIGH) {
    cuenta = cuenta - 1;
  }
  if (flag == LOW) {
    cuenta = cuenta + 1;
  }
}
void tetamotorsuperior()
{
  // Disminuimos el valor establecido
  flag1 = digitalRead(8);
  if (flag1 == HIGH) {
    cuenta1 = cuenta1 - 1;
  }
  if (flag1 == LOW) {
    cuenta1 = cuenta1 + 1;
  }
}
void tetacintura()
{
  // Disminuimos el valor establecido
  flag2 = digitalRead(4);
  if (flag2 == HIGH) {
    cuenta2 = cuenta2 - 1;
  }
  if (flag2 == LOW) {
    cuenta2 = cuenta2 + 1;
  }
}
```

Anexo D


Membership Function Editor: ControladorFuzzyPD5X5_15
File Edit View
FIS Variables
error
output1
derror
Membership function plots
plot points: 181
NB NS zero PS PB
input variable "error"
Current Variable
Name error
Type input
Range [-3 3]
Display Range [-3 3]
Current Membership Function (click on MF to select)
Name NB
Type zmf
Params [-1 -0.5]
Help Close
Ready


Membership Function Editor: ControladorFuzzyPD5X5_15
File Edit View
FIS Variables
error
output1
derror
Membership function plots
plot points: 181
NB NS zero PS PB
input variable "derror"
Current Variable
Name derror
Type input
Range [-1 1]
Display Range [-1 1]
Current Membership Function (click on MF to select)
Name NB
Type trapmf
Params [-15 -9 -0.5794 -0.26]
Help Close
Selected variable "derror"


Membership Function Editor: ControladorFuzzyPD5X5_15
File Edit View
FIS Variables
error
output1
derror
Membership function plots
plot points: 181
NB NM NS zero PS PM PB
output variable "output1"
Current Variable
Name output1
Type output
Range [-15 15]
Display Range [-15 15]
Current Membership Function (click on MF to select)
Name NB
Type trimf
Params [-14 -11 -8]
Help Close
Selected variable "output1"

Anexo E

Publicación en revista Pistas educativas Vol. 43, Núm. 139 (2021): Número semestral (julio-diciembre 2021)

DISEÑO DE UN CONTROLADOR PD DIFUSO Y ESTUDIO COMPARATIVO CONTRA UN CONTROL PID CLÁSICO PARA EL MODELO 3D DE UN ROBOT HÁPTICO

FUZZY PD CONTROLLER DESING AND COMPARATIVE STUDY WITH A CLASSIC PID CONTROLLER FOR A 3D MODEL OF AN HAPTIC ROBOT

Jaime Jalomo Cuevas
Tecnológico Nacional de México / IT de Cd. Guzmán, México
jaime.jc@cdguzman.tecnm.mx

José Luis Cortés Mendoza
Tecnológico Nacional de México / IT de Cd. Guzmán, México
josem20291055@cdguzman.tecnm.mx

Jorge Gudiño Lau
Universidad de Colima, México.
jgalu@ucol.mx

Sergio Sandoval Pérez
Tecnológico Nacional de México / IT de Cd. Guzmán, México
ssandoval@cdguzman.tecnm.mx

José de Jesús García Cortes
Tecnológico Nacional de México / IT de Cd. Guzmán, México
jesusgc@cdguzman.tecnm.mx

Recepción: 29/agosto/2021 **Aceptación:** 24/noviembre/2021

Resumen

Este trabajo presenta el rediseño y optimización de un dispositivo háptico mediante las herramientas de SolidWorks, para su exportación a Simulink, mediante la herramienta Simmechanics segunda generación. Se crea un diagrama a bloques para la simulación del comportamiento de los motores de cada articulación, mediante el uso de una función de transferencia obtenida a partir de los parámetros de un motor de características similares. Se crea un controlador PD (proporcional-derivativo) difuso, sintonizado heurísticamente a partir de las características del sistema y la respuesta deseada. El controlador diseñado prueba ser eficiente ante los movimientos programados en las articulaciones. La presente metodología

CERTIFICADO

Registro Público del Derecho de Autor

Para los efectos de los artículos 13, 162, 163 fracción I, 164 fracción I, y demás relativos de la Ley Federal del Derecho de Autor, se hace constar que la **OBRA** cuyas especificaciones aparecen a continuación, ha quedado inscrita en el Registro Público del Derecho de Autor, con los siguientes datos:

AUTORES:	CORTES MENDOZA JOSE LUIS JALOMO CUEVAS JAIME
TÍTULO:	FUZZY PHANTOM
RAMA:	PROGRAMAS DE COMPUTACION
TITULARES:	CORTES MENDOZA JOSE LUIS JALOMO CUEVAS JAIME

Con fundamento en lo establecido por el artículo 3° de la Ley Federal del Derecho de Autor, el presente certificado ampara única y exclusivamente la obra original de programa de cómputo.

Con fundamento en lo establecido por el artículo 168 de la Ley Federal del Derecho de Autor, las inscripciones en el registro establecen la presunción de ser ciertos los hechos y actos que en ellas consten, salvo prueba en contrario. Toda inscripción deja a salvo los derechos de terceros. Si surge controversia, los efectos de la inscripción quedarán suspendidos en tanto se pronuncie resolución firme por autoridad competente.

Con fundamento en los artículos 2, 208, 209 fracción III y 211 de la Ley Federal del Derecho de Autor; artículos 64, 103 fracción IV y 104 del Reglamento de la Ley Federal del Derecho de Autor, y artículos 1, 3 fracción I, 4, 8 fracción I y 9 del Reglamento Interior de Instituto Nacional del Derecho de Autor, se expide el presente certificado.

Número de Registro: 03-2021-111811561500-01

Ciudad de México, a 26 de noviembre de 2021

EL DIRECTOR DEL REGISTRO PÚBLICO DEL DERECHO DE AUTOR

JESUS PARETS GOMEZ

SECRETARÍA DE CULTURA
INSTITUTO NACIONAL DEL DERECHO DE AUTOR
DIRECCIÓN DE REGISTRO PÚBLICO DEL DERECHO DE AUTOR

CULTURA
SECRETARÍA DE CULTURA

INDAUTOR
INSTITUTO NACIONAL DEL DERECHO DE AUTOR