



TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ
Depto. De ingeniería Eléctrica y Electrónica



Sistema de ubicación y reconocimiento de objetos para robot autónomo.

ALUMNO:

Manuel de Jesús Cancino Escobar

No. Control:

16270264

Ing. Electrónica

ASESOR:

ING. ÁLVARO HERNÁNDEZ SOL

Tuxtla Gutiérrez Chiapas al 23 de enero de 2021

DEDICATORIA

Este proyecto está dedicado a las personas que más me han influenciado en mi vida, a mi madre por el esfuerzo y apoyo brindado para la formación de mi persona, a Dey por su amistad sincera y apoyo incondicional en mis peores momentos, a mis profesores que me orientaron a ejercer mi vocación como profesionista y encaminarme a la carrera que me apasiona.

A ti Wen, por ser la chica que festeja mis triunfos y me anima en mis fracasos, a ti por ser uno de los impulsores para la finalización de mis objetivos, a ti por creer siempre en mí, un paso más juntos.

Manuel Cancino

AGRADECIMIENTOS

Agradecido con Dios por permitirme seguir culminando mis objetivos, permitirme seguir adelante dándome resiliencia, amor y bondad, a pesar de haber dudado muchas veces de su ayuda fue imposible no notar su guía en cada decisión tomada.

A mi tutor el ingeniero Álvaro Hernández Sol quien me brindo la oportunidad de continuar con este proyecto junto a su apoyo y asesoramiento para la culminación del mismo, de igual forma por brindar su apoyo al club de Robotica del cual fui parte durante toda la carrera.

A mi universidad, la cual me formo profesionalmente y me brindo la oportunidad de ser parte de la comunidad estudiantil.

Manuel Cancino

ÍNDICE DE CONTENIDOS

DEDICATORIA	ii
AGRADECIMIENTOS	iii
ÍNDICE DE CONTENIDOS	iv
LISTADO DE TABLAS	viii
LISTADO DE FIGURAS	ix
ABREVIATURAS	xv
RESUMEN	xvi
ABSTRACT	xvii
Capítulo 1. ESTADO DEL ARTE.	
1.1. Antecedentes	1
1.1.1 “Reconstrucción de mapas para el recorrido de un vehículo sobre ruedas mediante un dispositivo de cómputo móvil.”	1
1.1.2 “Robot Oruga detector y esquivador de obstáculos con Fuzzy Logic IA.”	2
1.1.3 “Localización activa de robots móviles.”	4
1.1.4 “PMITO-Plataforma móvil para investigación de técnicas de edometría.”	5
1.1.5 “Movimiento restrictivo de un robot móvil.”	7
1.1.6 “detection of movable ground areas of a robots environment using atransducer array”. 8	
1.2. Formulación del problema	10
1.3. Justificación	11
1.4. Objetivos	11
1.4.1. Objetivo general	11
1.4.2. Objetivos específicos	12
Capítulo 2. MARCO TEORICO	
2.1 Diseño asistido por computadora	13
SolidWorks	14
2.2 Robótica	15

2.2.1 Estructura Mecánica de un Robot	16
2.2.2 Robots móviles	16
2.2.4 Esquema de un robot	17
2.2.5 locomoción	18
2.3 Sensores	28
2.3.1 Ultrasónicos SFR08	28
2.3.2 Giroscopio	35
2.4 Motores	38
2.4.1 Motor DC	38
2.4.2 Motor paso a paso	39
2.4.3 Motor VEX EDR 393	42
2.5 Arduino	43
2.5.1 Arduino Mega	43
2.6 Raspberry Pi	44
2.6.1 raspberry Pi 3b+	44
2.7 Matlab	47
2.7.1 Redes Neuronales Artificiales	47
2.7.2 Deep Learning	49
2.7.3 Machine Learning	51
2.7.4 Rasberry pi & Matlab	52
2.7.5 Matlab Coder	54
2.7.6 Matlab App Designer	55
2.8 Python	55
2.9 Proteus	56
Capítulo 3. DISEÑO Y MODELADO 3D	
3.1 Diseño del chasis para el prototipo robótico	57
3.1.1 Piezas que componen el chasis del prototipo robótico	57
3.1.2 Diseño completo del chasis	78
3.2 Diseño de la base giratoria para la captura de datos	78
3.2.1 Piezas que componen la plataforma giratoria	79
3.2.2 Diseño completo de la plataforma giratoria	92
3.3 Diseño completo del prototipo robótico	93

3.4 mejoras al diseño inicial.	94
3.5 diseño Final con mejoras implementadas al diseño inicial.	97
Capítulo 4. DISEÑO DE PLACA PCB	
4.1 Diseño de la primera shield para Arduino mega	100
ESQUEMATICO	100
PCB	104
4.2 Diseño de la shield final profesional para Arduino mega	106
Capítulo 5. METODOLOGÍA	
5.1 Construcción del prototipo Robótico	111
5.1.1 Construcción del chasis	111
5.1.2 Construcción de la base giratoria para el prototipo robótico	113
5.2 Programación en la placa Arduino mega 2560	116
5.2.1 Recolección de datos mediante Arduino y los sensores ultrasónicos SFR08	116
5.2.2 movimientos del prototipo Robótico	120
5.2.3 Comunicación entre Arduino y Raspberry	126
5.2.4 Comunicación Arduino y Matlab	135
5.3 Comunicación Matlab y Raspberry	136
5.4 Entrenamiento de una red neuronal artificial con Matlab	147
Prueba 1	151
Prueba 2	155
Prueba 2.3	156
Prueba 2.4	157
Prueba 3.1	158
Prueba 3.5	161
Prueba 3.6	163
Prueba 3.7	165
Prueba 4.6	167
Prueba 5.2	170
Prueba 5.7	173
5.5 interfaz gráfica en Matlab	176
Capítulo 6. RESULTADOS Y PRUEBAS FINALES	178
CONCLUSIONES	185

RECOMENDACIONES	186
REFERENCIAS	187
LISTADO DE ANEXOS	192
ANEXO A- CODIGOS	192
A.1.- función getRange	192
A.2.-Funcion girar.	193
A.3.- Función start_1	193
A.4.- Funcion leftmotorcontrol.	194
A.5.- Funcion rightmotorcontrol	195
A.6.- código para el entrenamiento de las redes neuronales con 3 salidas	195
A.7.- código para la comprobación con nuevos datos de las redes neuronales entrenadas con 3 salidas	197
A.8.- Función para la graficación de las distancias enviadas a Matlab	201
A.8.- Función para recolección de datos de la raspberry y Matlab	210
ANEXO B.-Tablas.	213
B.1.- Tabla de los entrenamientos para la etapa uno	213
B.2.- Tabla de los entrenamientos para la etapa dos	213
B.3.- Tabla de los entrenamientos para la etapa tres	213
B.4.- Tabla de los entrenamientos para la etapa cuatro	214
B.5.- Tabla de los entrenamientos para la etapa cinco	214

LISTADO DE TABLAS

Tabla 2.1 registros del SFR08	31
Tabla 2.2 Corriente utilizada por el sensor SFR08	35
Tabla 5.1 direcciones para el protocolo de comunicación i2c de los sfr08	119

LISTADO DE FIGURAS

figura 1.1 Resultados obtenidos de la reconstrucción de mapas.....	2
figura 1.2 Robot oruga perfil frontal lateral.....	3
figura 1.3 Montaje final del prototipo.....	4
figura 1.4 Foto superior del chasis.....	6
figura 1.7 diagrama de flujo del sistema “ pmito ” en general.	6
figura 1.8 Vista en perspectiva de un robot	8
figura 1.9 Operación de ejemplo de un robot por primera vez, según una realización de ejemplo.....	9
figura 1.10 Vista en perspectiva de un robot cuadrúpedo, según un ejemplo de realización.	9
figura 2.1 Interfaz CAD SolidWorks	15
figura 2.2 Robótica educativa.	16
figura 2.3 robot y su interacción con el entorno.....	17
figura 2.4 Sistema Ackerman.....	19
figura 2.5 locomoción de triciclo clásico.....	20
figura 2.6 Configuración para transmisión síncrona.....	20
figura 2.7 Robot móvil en configuración diferencial.	21
figura 2.8 Robot Terragator con locomoción tipo "Skid Steer"	22
figura 2.9 Rueda convencional fija.....	23
figura 2.10 Rueda convencional de centro orientable.....	24
figura 2.11 Rueda convencional de centro orientable desplazado.....	25
figura 2.12 Rueda omnidireccional o sueca.	26
figura 2.13 Batería VEX 7.2V.	27
figura 2.14 Sensor Ultrasónico SFR08	29
figura 2.15 Conexiones del sensor ultrasónico SFR08	30
figura 2.16 Dimensiones del sensor ultrasónico SFR08	34
figura 2.17 El mecanismo de la fuerza Coriolis.....	36
figura 2.18 La estructura interna del giroscopio: 1 - masa de sujeción, 2 - peso de trabajo, 3 - fijación del marco interior, 4 - sensores que mueven el marco interior, 5 - marco interior, 6 – sustrato.....	36
figura 2.19 Vista de sección de un motor paso a paso de reluctancia variable.....	40
figura 2.20 Vista en sección de un magneto permanente.....	41
figura 2.21 Vista expandida ilustrativa del desplazamiento de dientes.....	42
figura 2.22 Arduino Mega 2560 REV 3.....	44
figura 2.23 Raspberry pi 3B+.....	46
figura 2.24 Dimensiones de la Raspberry pi 3B+	46
figura 2.25 Arquitectura de una red neuronal típica.....	49
figura 3.1 Dimensiones del Subensamble de la rueda de tracción	58
figura 3.2 Subensamble de la rueda de tracción	58
figura 3.3 Dimensiones del CAD carril 2 x 1 x 25 agujeros.....	59
figura 3.4 Diseño CAD carril 2 x 1 x 25 agujeros.....	60

figura 3.5 Dimensiones del diseño CAD canal 1 x 2 x 1 x 25 agujeros.	61
figura 3.6 Diseño CAD canal 1 x 2 x 1 x 25 agujeros.	62
figura 3.7 Dimensiones del subensamble del motor VEX EDR 269	63
figura 3.8 Subensamble del motor VEX EDR 269	63
figura 3.9 Dimensiones del Diseño CAD del Eje de 3”	64
figura 3.10 Diseño CAD del Eje de 3”	64
figura 3.11 Dimensiones del diseño CAD de los collarines	65
figura 3.12 Dimensiones del diseño CAD de los collarines	66
figura 3.13 Dimensiones del diseño CAD de las chumaceras para ejes	66
figura 3.14 Diseño CAD de las chumaceras para ejes	67
figura 3.15 Ensamble de los componentes para el chasis	67
figura 3.16 dimensiones del diseño CAD de la base del motor paso a paso (nema17).	68
figura 3.17 Diseño CAD de la base del motor paso a paso (nema17).	69
figura 3.18 Dimensiones del diseño CAD del Nema 17	69
figura 3.19 Diseño CAD del Nema 17	70
figura 3.20 Dimensiones del diseño CAD de base NEMA-COJINETE	71
figura 3.21 Diseño CAD de base NEMA-COJINETE	71
figura 3.22 Dimensiones del diseño CAD del cojinete de giro integral inferior	72
figura 3.23 Diseño CAD del cojinete de giro integral inferior	73
figura 3.24 Dimensiones del diseño CAD del engrane central del Nema 17	74
figura 3.25 Diseño CAD del engrane central del Nema 17	74
figura 3.26 Dimensiones del diseño CAD del engrane planetario	75
figura 3.27 Diseño CAD del engrane planetario	76
figura 3.28 Dimensiones del diseño CAD del cojinete de giro integral	77
figura 3.29 Diseño CAD del cojinete de giro integral	77
figura 3.30 Ensamble del chasis del autómata	78
figura 3.31 Dimensiones del diseño CAD de la base negra de la parte superior.	79
figura 3.32 Diseño CAD de la base negra de la parte superior.	80
figura 3.33 Dimensiones del diseño CAD de la base para los sensores ultrasónicos SFR08. 81	81
figura 3.34 Diseño CAD de la base para los sensores ultrasónicos SFR08.	81
figura 3.35 Medidas provistas del sensor ultrasónico SFR08.	82
figura 3.36 Diseño CAD del sensor ultrasónico SFR08.	82
figura 3.37 Dimensiones del diseño CAD del acrílico inferior	83
figura 3.38 Diseño CAD del acrílico inferior	84
figura 3.39 Dimensiones del diseño CAD de la base que alojaría la rueda loca.	85
figura 3.40 Diseño CAD de la base que alojaría la rueda loca.	85
figura 3.41 Dimensiones del diseño CAD del balín o rueda loca	86
figura 3.42 Diseño CAD del balín o rueda loca	86
figura 3.43 Dimensiones de los diseños CAD de la tornillería	87
figura 3.44 Diseños CAD de la tornillería	88
figura 3.45 Dimensiones de los diseños CAD de la tornillería	88
figura 3.46 Diseños CAD de la tornillería	89

figura 3.47 Dimensiones del diseño CAD pantalla LCD 240*128 pixeles	89
figura 3.48 Diseño CAD pantalla LCD 240*128 pixeles	90
figura 3.49 Modelo CAD Raspberry pi 3b+.....	91
figura 3.50 Modelo CAD Arduino Mega 2560.....	91
figura 3.51 Modelo CAD Shield Para Arduino Mega 2560.....	91
figura 3.52 Ensamble final de la parte superior del prototipo final (A).....	92
figura 3.53 Ensamble final de la parte superior del prototipo final (B).....	92
figura 3.54 Ensamble final del prototipo robótico (A).....	93
figura 3.55 Ensamble final del prototipo robótico (B).....	93
figura 3.56 Ensamble final del prototipo robótico (C).....	94
figura 3.57 Dimensiones del diseño CAD de la llanta de fricción de 4 pulgadas.....	95
figura 3.58 Diseño CAD de la llanta de fricción de 4 pulgadas.....	95
figura 3.59 Dimensiones del diseño CAD del motor VEX EDR 393.....	96
figura 3.60 Diseño CAD del motor VEX EDR 393.....	96
figura 3.60 Ensamble final del prototipo robótico con las mejoras implementadas (A)	97
figura 3.61 Ensamble final del prototipo robótico con las mejoras implementadas (B).....	97
figura 3.62 Ensamble final del prototipo robótico con las mejoras implementadas (C)	98
figura 4.1 Creación y etiquetado de los pines de la placa Arduino mega en ISIS proteus..	100
figura 4.2 Diseño de la etapa reguladora de voltaje de entrada.....	101
figura 4.3 Diseño de la base para el driver a4988.	102
figura 4.4 Diseño de la base para los sensores SFR08, modulo bluetooth y modulo SD.....	103
figura 4.5 Diseño de la base para la LCD 240*128 Píxeles.....	104
figura 4.6 Diseño de la PCB en proteus.....	105
figura 4.7 Diseño CAD generado por ISIS proteus.....	105
figura 4.8 Pines para control de motores y activador Infrarrojo	106
figura 4.9 Gestor de Reglas del diseño	107
figura 4.10 Diseño final de la PCB	108
figura 4.11 Diseño CAD de la PCB final generado por ISIS proteus (A)	108
figura 4.12 Diseño CAD de la PCB final generado por ISIS proteus (B).....	109
figura 4.13 Placa PCB finalizada (A)	110
figura 4.14 Placa PCB finalizada (B).....	110
figura 5.1 Montaje del chasis vista inferior	112
figura 5.2 Montaje del chasis vista superior	112
figura 5.3 Montaje de la caja reductora de engranes helicoidales (A)	112
figura 5.4 Montaje de la caja reductora de engranes helicoidales (B)	113
figura 5.5 Montaje de la base giratoria con los sensores ultrasónicos SFR08.....	113
figura 5.6 Montaje del shield de control junto a la placa Arduino mega 2560.....	114
figura 5.7 Montaje completo de la base giratoria	114
figura 5.8 empalmado de cables planos para el protocolo de comunicación I2C de los sensores ultrasónicos SFR08	115
figura 5.9 Montaje del prototipo robótico	115
figura 5.10 Diagrama de bloques de la rutina getRange().....	118

figura 5.11 Distribución de los sensores SFR08 en el prototipo robótico	119
figura 5.12 Diagrama de flujo para la recolección de datos.....	121
figura 5.13 Diagrama de flujo para la configuración y obtención de los ángulos mediante el MPU6050 y su DMP.....	122
figura 5.14 Tren de pulsos a 50hz para el control de velocidad.....	123
figura 5.15 Sistema de control a implementar.....	124
figura 5.16 Diagrama de flujo para la configuración del DMP y control PID	125
figura 5.17 Software para formateo de tarjetas SD	128
figura 5.18 Software baleanaEtcher	128
figura 5.19 configurando la IP estática de nuestra raspberry (a).....	129
figura 5.20 VNC Connect	130
figura 5.21 Raspi-config – Interfacing options	131
figura 5.22 Raspi-config – Interfacing options- serial (a).....	131
figura 5.23 Raspi-config – Interfacing options- serial (b).....	132
figura 5.24 Raspi-config – Interfacing options- serial (c)	132
figura 5.25 Raspi-config – Interfacing options- reboot.....	133
figura 5.26 diagrama de flujo comunicando raspberry con Arduino mediante el puerto serial	134
figura 5.27 instalación del soporte para raspberry de Matlab (a).....	137
figura 5.28 instalación del soporte para raspberry de Matlab (b).....	137
figura 5.29 Instalación del soporte para raspberry de Matlab - Manage.....	138
figura 5.30 Configuración del paquete de raspberry pi - selección de la placa raspberry pi.	138
figura 5.31 Configuración del paquete de raspberry pi - selección del sistema operativo para la tarjeta.....	139
figura 5.32 Configuración del paquete de raspberry pi - selección del sistema operativo que nos ofrece Matlab.....	140
figura 5.33 Formateando SD	141
figura 5.34 Configuración del paquete de raspberry pi - validando el sistema operativo descargado	142
figura 5.35 Configuración del paquete de raspberry pi - selección la configuración de red.	142
figura 5.36 Configuración del paquete de raspberry pi – configuración de nuestra red privada.....	143
figura 5.37 Configuración del paquete de raspberry pi – insertar nuestra tarjeta SD.....	144
figura 5.38 Configuración del paquete de raspberry pi – instalando el sistema operativo.	144
figura 5.39 Configuración del paquete de raspberry pi – pasos finales de instalación.....	145
figura 5.40 Configuración del paquete de raspberry pi – raspberry pi detectada por Matlab.	146
figura 5.41 Configuración del paquete de raspberry pi – configuración completa.....	146
figura 5.42 propiedades de la conexión con la raspberry pi.....	147
figura 5.43 Programación tradicional vs machine learning	148
figura 5.44 inteligencia artificial.....	149

figura 5.45 modelo de una red neuronal artificial.....	150
figura 5.46 Capturando datos nuevos de un balon	153
figura 5.47 Capturando datos nuevos de un cilindro.....	153
figura 5.48 Capturando datos nuevos de una caja	154
figura 5.49 Parámetros utilizados para el primer entrenamiento de la tercera etapa.....	158
figura 5.50 Entrenamiento de la red 3.1 y matrices de confusión.....	159
figura 5.51 Evaluación de la figura “balon”	160
figura 5.52 Evaluación de la figura “caja”	160
figura 5.53 Evaluación de la figura “cilindro”	160
figura 5.54 Parámetros utilizados para el quinto entrenamiento de la tercera etapa	161
figura 5.55 Entrenamiento de la red 3.5 y matrices de confusión.....	162
figura 5.56 Parámetros utilizados para el sexto entrenamiento de la tercera etapa	163
figura 5.57 Resultados del entrenamiento del sexto entrenamiento de la tercera etapa (A).....	164
figura 5.58 Resultados del entrenamiento del sexto entrenamiento de la tercera etapa (B).....	164
figura 5.59 Parámetros utilizados para el séptimo entrenamiento de la tercera etapa	165
figura 5.60 Resultados del entrenamiento del séptimo entrenamiento de la tercera etapa (A)	166
figura 5.61 Resultados del entrenamiento del séptimo entrenamiento de la tercera etapa (B)	166
figura 5.62 Parámetros utilizados para el sexto entrenamiento de la cuarta etapa	168
figura 5.63 Resultados del entrenamiento del sexto entrenamiento de la cuarta etapa (A)	168
figura 5.64 Resultados del entrenamiento del sexto entrenamiento de la cuarta etapa (B)	169
figura 5.65 Resultados del entrenamiento del sexto entrenamiento de la cuarta etapa (C)	169
figura 5.66 Parámetros utilizados para el segundo entrenamiento de la quinta etapa.....	171
figura 5.67 Resultados del entrenamiento del segundo entrenamiento de la quinta etapa (A)	171
figura 5.68 Resultados del entrenamiento del segundo entrenamiento de la quinta etapa (B)	172
figura 5.69 Resultados del entrenamiento del segundo entrenamiento de la quinta etapa (C)	172
figura 5.70 Parámetros utilizados para el séptima entrenamiento de la quinta etapa	173
figura 5.71 Resultados del entrenamiento del séptimo entrenamiento de la quinta etapa (A)	174
figura 5.72 Resultados del entrenamiento del séptimo entrenamiento de la quinta etapa (B)	174
figura 5.73 Resultados del entrenamiento del séptimo entrenamiento de la quinta etapa (C)	175
figura 5.74 Interfaz gráfica desarrollada en App Designer de Matlab para el prototipo robótico.	176
figura 6.1 Modificaciones al chasis del prototipo robótico.....	178
figura 6.2 primeras pruebas con la red neuronal artificial implementada en el prototipo robótico	179

figura 6.2 Primer objeto detectado con la red neuronal artificial implementada en el prototipo robótico	180
figura 6.2 Segundo objeto detectado con la red neuronal artificial implementada en el prototipo robótico	181
figura 6.3 Tercer objeto detectado con la red neuronal artificial implementada en el prototipo robótico	182
figura 6.4 Cálculo de la distancia mediante pulsos ultrasónicos.....	183
figura 6.5 Cuarto objeto detectado con la red neuronal artificial implementada en el prototipo robótico	184

ABREVIATURAS

CAD (Computer Aided Design): Diseño asistido por computador.

CADD (Computer Aided Design and Drafting): Diseño y dibujo asistido por computadora.

CAE (Computer Aided Engineering): Ingeniería asistida por computador.

CAM (Computer Aided Manufacturing): Manufactura asistida por computador.

CIM (Computer Integrated Manufacturing) Manufactura integrada por computador.

FEA (Finite Element Analysis): Análisis por elementos finitos.

VR (Variable reluctance) : Reluctancia variable

PM (PERMANENT MAGNET) : motor de magneto permanente

PoE (Power over Ethernet) : Alimentación a través de Ethernet

IMU (inertial motion unit) : unidad de movimiento inercial

DMP (digital motion processor) : procesador digital de movimiento

ESC (Electronic Speed Controller) : Controlador de velocidad electrónico

RPM (revolutions per minute) : revoluciones por minuto

RESUMEN

El principal objetivo de este trabajo de fin de grado es el Diseñar un sistema de reconocimiento de objetos usando sensores ultrasónicos, para lo que se diseñó un prototipo robótico donde se implementaron nueve sensores posicionados en una base circular con una apertura de 22.5° entre sensores y mediante ellos hacer la captura del entorno junto con estos datos poder hacer un reconocimiento de ciertas características de los objetos capturados.

Del mismo modo para realizar un reconocimiento acertado de los objetos se plantea el desarrollo de una red neuronal artificial desde cero mediante un sistema de cómputo numérico por lo que se utilizó Matlab y mediante este mismo emplear placas de software libre como lo es Arduino y Raspberry para la implementación en el autómata.

Por otra parte, también se pretendía explorar las herramientas que Matlab nos ofrece para el desarrollo de las redes neuronales artificiales y así determinar el tipo de red neuronal necesaria para que nuestro objetivo se cumpliera y nuestro sistema fuera eficiente, dentro de Matlab encontramos distintos métodos numéricos, probabilidad y estadística , redes neuronales convulsionales , machine learning , Deep learning entre otros.

Al implementar todo lo anterior en nuestro prototipo robótico este se debería desarrollar en un ambiente controlado en el cual pueda identificar qué tipo de objeto tiene cerca y así evitarlo trazando nuevas rutas de navegación.

PALABRAS CLAVE: Robot , Redes Neuronales Artificiales, Reconocimiento de patrones, microcontroladores, remoto.

ABSTRACT

The main objective of this final degree project is to Design an object recognition system using ultrasonic sensors, for which a robotic prototype was designed where nine sensors were implemented positioned on a circular base with an opening of 22.5° between sensors and by means of them to capture the environment together with this data, to be able to make a recognition of certain characteristics of the captured objects.

In the same way, to carry out a successful recognition of objects, the development of an artificial neural network from scratch is proposed by means of a numerical computing system, which is why Matlab was chosen and through this same use of free software boards such as Arduino and Raspberry for implementation in the automaton.

On the other hand, it was also intended to explore the tools that Matlab offers us for the development of artificial neural networks and thus determine the type of neural network necessary for our objective to be fulfilled and our system to be efficient, within Matlab we find different numerical methods , probability and statistics, convulsive neural networks, machine learning, Deep learning among others.

By implementing all of the above in our robotic prototype, it should be developed in a controlled environment in which it can identify what type of object is nearby and thus avoid it by tracing new navigation route

KEY WORDS: Robot, Artificial Neural Networks, Pattern recognition, microcontrollers, remote.

Capítulo 1. ESTADO DEL ARTE.

1.1. Antecedentes

El desplazamiento de autómatas o prototipos robóticos en ambientes desconocidos es uno de los aspectos fundamentales a tener en consideración al momento del desarrollo del mismo, durante la evolución de la robótica e inteligencia artificial se han implementado nuevos métodos para hacer más eficiente el desplazamiento en distintos ambientes, algunos de ellos ha sido la visión artificial o dispositivos LIDAR.

A nivel nacional encontraremos interesantes artículos sobre la robótica móvil y el reconocimiento de entornos.

1.1.1 “Reconstrucción de mapas para el recorrido de un vehículo sobre ruedas mediante un dispositivo de cómputo móvil.”

El objetivo de este trabajo fue desarrollar un sistema completo que captura datos a través del uso de sensores ultrasónicos de distancia y una brújula digital que forma parte de un sistema embebido con base en un microcontrolador, en este caso, fue una tarjeta Arduino, montado en un vehículo móvil circular, que tiene en sus laterales dos motorreductores de corriente continua trabajando de manera aislada. En la parte baja cuenta con dos ruedas locas colocadas una al frente y otra atrás. Como etapa de potencia que alimenta a los dos motores, se utiliza el circuito integrado L293D Se adaptaron también en el vehículo tres sensores de distancia ultrasónicos HC-SR 04, uno en la parte frontal y los otros en los laterales. Para el desarrollo del control de movimiento y procesamiento de datos se elaboró una aplicación que se ejecuta en un sistema de cómputo móvil con sistema operativo Android para obtener de manera gráfica el entorno que ha recorrido el vehículo. Por último, se construyó un sistema de comunicación inalámbrica wifi con arquitectura

cliente-servidor entre las dos plataformas. Como resultados obtuvieron que, de acuerdo con los valores obtenidos por el sensor ultrasónico frontal, se pudo apreciar que las mediciones se realizaron cada 3 cm de avance del vehículo, aproximadamente, lo que proporciona de manera gráfica un mapa con buena resolución. Por la manera de obtener datos, solo en posiciones longitudinales y transversales a la posición inicial se pudo apreciar huecos en las esquinas internas por donde el vehículo circula, porque son lugares donde no se toma información, a diferencia de las esquinas del lado externo, donde no existe algún problema para registrar la información pertinente.[1]



figura 1.1 Resultados obtenidos de la reconstrucción de mapas.

1.1.2 “Robot Oruga detector y esquivador de obstáculos con Fuzzy Logic IA.”

De igual manera, pero a nivel internacional también nos encontramos el siguiente proyecto el cual fue realizado para poner en práctica los conocimientos de robótica buscando la autonomía del prototipo, con el propósito de facilitar las labores de ejecución del robot basado en una guía lógica de recepción, toma de decisiones y respuesta de las mismas. La construcción se basó en un robot tipo oruga que detecta y esquia los obstáculos en un entorno plano. Se usó un método de inteligencia artificial llamado Fuzzy Logic para el control de los motores y de los sensores de proximidad. Este prototipo

robótico tiene una etapa de Potencia con un L298, alimentado por 2 Baterías de 3.7 Vdc a 3000 mA, una placa Arduino mega con un microcontrolador atmega2560 que es el encargado de recibir los datos de los sensores Sharp e implementar la inteligencia artificial. El Rover 5 no detecta obstáculos con diámetros menores a los 7 cm, también tiene puntos ciegos en su diseño, en cualquier oportunidad el objeto puede estar ubicado preciso donde el robot no lo detecte y este choque contra él. Como conclusiones el rover 5 cumple con los parámetros del Fuzzy logic, este atiende o es capaz de visualizar los objetos y dependiendo de la posición de los objetos los intenta esquivar. debido a sus puntos ciegos el rover 5 no es capaz de esquivar todos los objetos que se encuentra en el camino. [2]

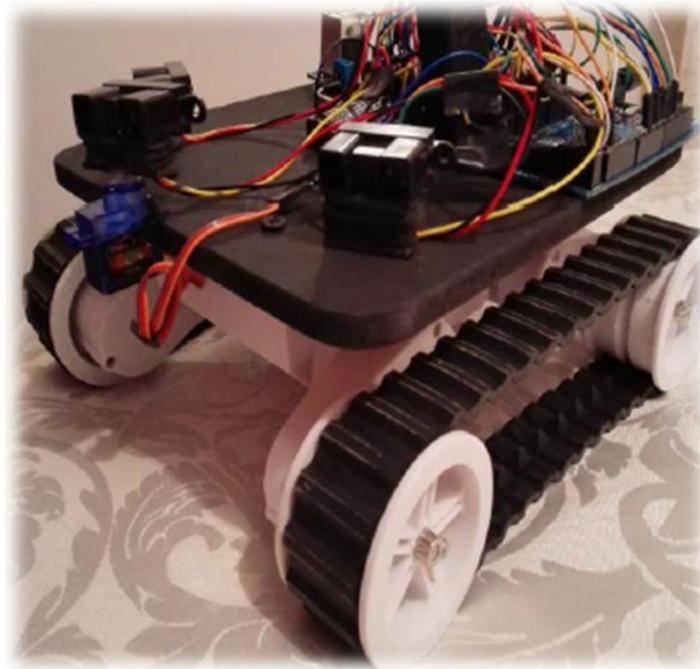


figura 1.2 Robot oruga perfil frontal lateral

1.1.3 “Localización activa de robots móviles.”

Nos explica que la estimación de la localización de robots móviles es un requerimiento esencial hoy en día como ayuda en la navegación, por ello su objetivo es dotar de dispositivos de localización a un robot móvil, carente de dispositivo GPS. Este estudio decidió utilizar sensores de posición, concretamente sensores ultrasónicos, y como cerebro de operación microcontroladores Arduino, un módulo de radiofrecuencias para la comunicación entre placas, y un actuador para su movimiento, en este caso un servomotor. El prototipo final consistía en que el actuador enviara pulsos mediante un par de ultrasónicos y otros ultrasónicos colocados en distintas áreas recibieran el pulso y así se determinara la distancia en la que se encuentra. Como conclusiones obtuvieron que Las medidas hechas por los sensores son aproximadas a las reales. Todo esto, resultado de los cambios tanto en el software como en el hardware que se llevaron a cabo, lo que permitió darles un uso diferente a los sensores ultrasónicos. [3]



figura 1.3 Montaje final del prototipo

1.1.4 “PMITO-Plataforma móvil para investigación de técnicas de edometría.”

El trabajo describe el diseño y construcción de una plataforma móvil con anillos de sensores infrarrojos y sensores ultrasónicos como dispositivos para la medición de distancia, los cuales serán posteriormente utilizados para la navegación de robots móviles en ambientes dinámicos, también muestran una breve reseña sobre la arquitectura de robots móviles y la importancia de determinar una estructura de control para formular alternativas de navegación y construcción de mapas de entorno con la implementación de sensores que permitan al sistema conocer la distancia que ha recorrido, su ubicación, y la distancia existente entre el entorno y el sistema mismo, utilizando sistemas embebidos, en este caso tarjetas de la serie ARDUINO.

El robot incluye un anillo sensorial que se compone de ocho sensores de ultrasonido ubicados en la parte superior de la plataforma, tres de ellos son colocados en la parte delantera, uno en la parte trasera y los otros cuatro son colocados de a dos en las partes laterales, este dispositivo sirve para detectar los objetos que se encuentren alrededor del anillo midiendo la distancia y así poder tomar decisiones para esquivar el objeto sin dificultades.

También incluye un anillo sensorial que está ubicado en la parte inferior de la plataforma, la cual se compone de ocho sensores infrarrojos, tres de ellos son colocados en la parte delantera, uno en la parte trasera y los otros cuatro son colocados de a dos en la parte lateral, este dispositivo sirve para detectar los objetos que se encuentren alrededor del anillo para un rango de distancia entre 0 y 20 cm. Cuenta con una brújula digital (hmc5883l) la cual es un dispositivo capaz de detectar los campos magnéticos terrestres, su funcionamiento se lleva a cabo mediante unos sensores magneto resistivos que aplicando una diferencia de potencial los sensores convierten cualquier campo magnético incidente en la dirección del eje sensible con una salida de voltaje diferencial

Para la movilidad del prototipo robótico se eligió la configuración de disposición de ruedas con guiado diferencial, esta tenía dos motores DC con encoders stemann, para el microcontrolador se eligieron dos Arduinos megas conectados por el protocolo i2c , con lo

que uno funcionaría como maestro y recibiría los datos de los sensores y la brújula y el otro como esclavo el cual le indicaría la movilidad al prototipo y decodificaría los encoders. [4]

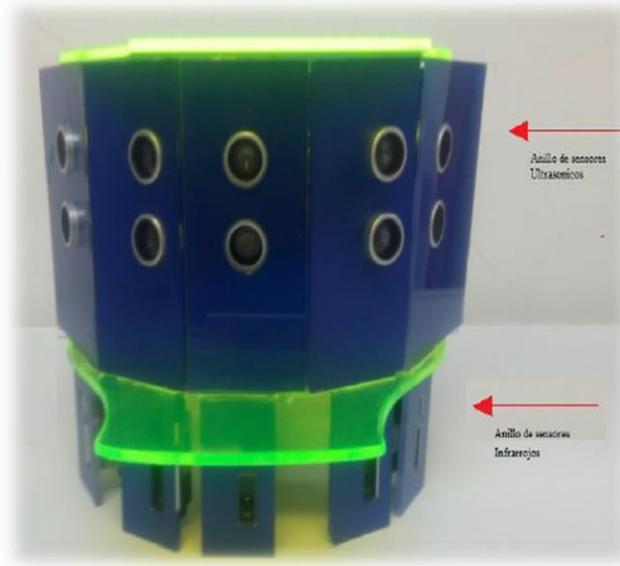


figura 1.4 Foto superior del chasis

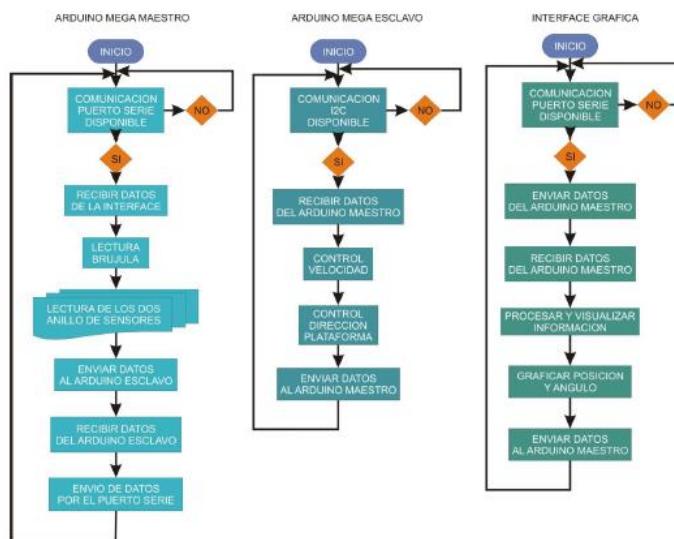


figura 1.5 diagrama de flujo del sistema “ pmito ” en general.

1.1.5 “Movimiento restrictivo de un robot móvil.”

Hablando sobre patentes, encontraremos muchas referentes a prototipos robóticos evasores de obstáculos o de algún sistema mapeador de entornos enfocados a la limpieza doméstica de superficies, entre ellos encontramos un robot que puede identificar áreas de un entorno que no se pueden atravesar, aunque no exista un límite estructural, tal como una pared, obstáculo u otra superficie, para impedir la entrada a esas áreas. El robot puede generar una barrera virtual para impedir el movimiento hacia esas áreas.

La patente estimula que el robot debe incluir algunas de las características provistas en ella como: “La barrera puede extenderse a través de una entrada, y la posición inicial del robot puede estar dentro de la entrada. El cuerpo puede incluir una parte frontal y una parte posterior. La barrera puede extenderse a lo largo de una línea que es paralela a la parte posterior del robot. La línea puede ser tangencial a la parte posterior del robot. La línea puede intersecar el cuerpo del robot en una ubicación indicada por un indicador visual en el robot. La barrera puede incluir una primera línea que se extiende paralela a la parte posterior del robot y una segunda línea que se extiende perpendicular a la parte posterior del robot. La ubicación inicial del robot puede colocar la parte posterior del cuerpo adyacente a la primera línea y un lado del cuerpo adyacente a la segunda línea”[5]. Del mismo modo menciona que el controlador puede ser programado para restringir el movimiento del cuerpo realizando operaciones que incluyen generar un mapa que represente un área que ha de ser limpiada y designar una barrera virtual en el mapa que pueda indicar una ubicación que el robot tiene prohibido cruzar. La barrera puede ser designada designando coordenadas correspondientes a la barrera como que no se pueden atravesar. De igual manera menciona que el usuario puede controlar el robot y las áreas a través de las cuales navega. El robot puede estar restringido a áreas donde puede moverse libremente mientras reduce el riesgo de daños a los objetos en el área. En algunas implementaciones, el robot funciona de manera autónoma y el usuario no necesita vigilar el robot cuando cubre una habitación con el fin de mantener el robot fuera de áreas particulares de la habitación. [5]

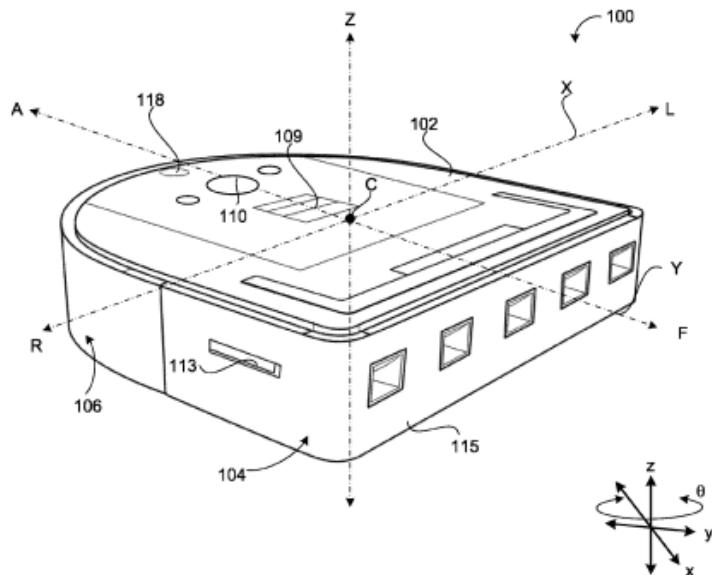


figura 1.6 Vista en perspectiva de un robot

1.1.6 “detection of movable ground areas of a robots environment using atransducer array”.

Ahora bien, ubicándonos en el continente americano tenemos otra patente de estados unidos, la solicitud describe implementaciones que se relacionan con la detección de superficies inestables dentro de un entorno. El robot incluye un sensor de profundidad, al menos un transductor y un dispositivo informático. El método implica recibir, desde el sensor de profundidad acoplado al robot móvil, una primera medición de profundidad entre él y una superficie del suelo. También implica hacer que al menos un transductor acoplado al robot móvil emita una onda de presión direccional hacia la superficie del suelo. De igual manera, el sensor de profundidad permite una segunda medición de profundidad hacia la superficie del suelo después de emitir la onda de presión direccional. Así mismo implica proporcionar instrucciones de navegación al robot móvil en base a una o más diferencias identificadas entre la primera medición de profundidad y la segunda medición de profundidad. Además, las operaciones incluyen identificar una o más diferencias entre la

primera medición de profundidad y la segunda medición de profundidad que indica que la superficie del suelo incluye un elemento móvil. [6]

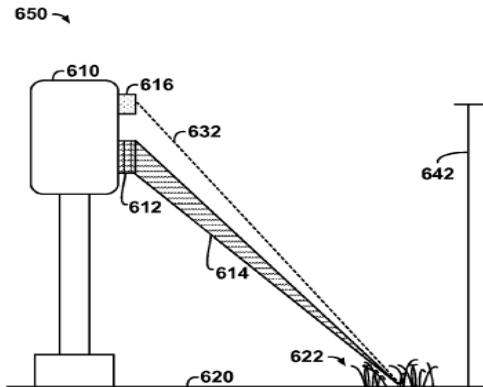


figura 1.7 Operación de ejemplo de un robot por primera vez, según una realización de ejemplo.

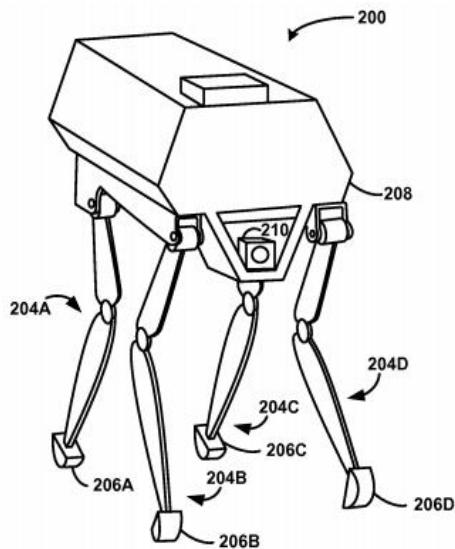


figura 1.8 Vista en perspectiva de un robot cuadrúpedo, según un ejemplo de realización.

1.2. Formulación del problema

La autonomía de un robot se ve definida por su capacidad de reaccionar ante los acontecimientos que surgen a su alrededor, para ello se debe tener un sistema estable y robusto el cual resuelva de la mejor manera tales circunstancias.

En el departamento de ingeniería de eléctrica y electrónica del TecNM instituto tecnológico de tuxtla Gutiérrez, se plantea el desarrollar prototipos robóticos lo cuales puedan navegar en ambientes desconocidos, sin embargo, el enfoque que se da al programar estos prototipos robóticos se basa en indicarle cómo reaccionar a distintas condiciones o reglas que debe seguir dependiendo de los datos de entrada.

Por otro lado, lo que se pretende desarrollar es un sistema el cual mediante redes neuronales artificiales detecten características específicas de figuras geométricas como cuadrados y cilindros con esto abrir paso hacia sistemas inteligentes embebidos en próximos prototipos. A diferencia del enfoque anterior, una red neuronal artificial como la que se plantea es entrenada con datos de entrada y sus etiquetas o respuestas por lo que al finalizar el entrenamiento esta red entrega las reglas que se utilizará para reaccionar a nuevos datos de entrada y con esto la red nos da un porcentaje de acierto.

Para poder diseñar este sistema se implementan algoritmos que involucran distintos tipos de hardware para su procesamiento, del mismo modo necesitamos dotar al robot de “sentidos” que lo ayuden a percibir el ambiente que lo rodea, estos sensores funcionan como datos de entrada para el prototipo robótico, sin estos sensores sería imposible interactuar con el medio donde se desarrolla el prototipo.

1.3. Justificación

En la actualidad la robótica y la inteligencia artificial están tomando un papel cada vez más importante, la mayoría de los trabajos humanos requieren que realicemos una combinación de los siguientes cuatro tipos básicos de tareas: manual repetitivo, manual no repetitivo, cognitivo repetitivo y cognitivo no-repetitivo. Por lo que empresas dedicadas a la robótica se han enfocado en desarrollar robot dotados de tecnología cognitivas e inteligencia artificial, estas tecnologías son las que pueden “desempeñar y/o aumentar tareas, ayudar a informar mejor las decisiones, y lograr objetivos que tradicionalmente han requerido inteligencia humana, tales como planeación, razonamiento a partir de información parcial o incierta, y aprendizaje.”[7]

Con esta tesis se desarrollará un prototipo robótico el cual pueda desplazarse en un ambiente controlado, pero con distintos obstáculos de formas geométricas, los cuales nuestro robot será capaz de identificar mediante sensores ultrasónicos y por medio de una Red Neuronal Artificial determinar qué forma geométrica tiene el objeto y así trazar una nueva ruta para esquivar el objeto, también mediante una interfaz gráfica visualizar la forma del objeto en cuestión.

Ahora bien, el desarrollo de este prototipo robótico lograra generar un impacto tecnológico a la comunidad estudiantil del TecNM instituto tecnológico de tuxtla Gutiérrez, el cual motivara a involucrarse en áreas de investigación científica aplicando la ingeniería de una manera práctica.

1.4. Objetivos

1.4.1. Objetivo general

“Diseñar un sistema de reconocimiento de objetos usando sensores ultrasónicos.”

1.4.2. Objetivos específicos

- Diseñar un sistema giratorio de captura mediante sensores ultrasónicos SFR08.
 - Diseñar un prototipo robótico capaz de navegar con precisión en entornos controlados e implementar el sistema giratorio de captura de datos.
 - Desarrollar una programación de captura y almacenamiento de datos para su postproceso.
 - Desarrollar una red neuronal artificial en el sistema de cómputo MATLAB con los datos recolectados.
 - Implementar la red neuronal artificial en el prototipo robótico.
 - Diseñar e implementar un sistema de navegación autónomo al prototipo robótico.
- .

Capítulo 2. MARCO TEORICO

2.1 Diseño asistido por computadora

El diseño es una actividad que se proyecta hacia la solución de problemas planteados por el ser humano en su adaptación al medio que lo rodea, para la satisfacción de sus necesidades, para lo cual utiliza recursos como la tecnología CAD/CAE/CAM. Estas tecnologías se vienen aplicando a través de los métodos de la ingeniería concurrente. La técnica más desarrollada en la ingeniería asistida por computador (CAE), es la aplicación de los análisis por elementos finitos (FEA), que con la mejora de los equipos de cómputo se ha convertido en técnicas accesibles para todos los usuarios. Estas técnicas son usadas industrialmente desde el diseño hasta la fabricación consiguiendo optimizar costos, calidad, tiempo, seguridad, etc.

El avance vertiginoso del software y hardware, en estos últimos años ha modificado la forma de entender el concepto de CAD. El CAD es una técnica de análisis, una manera de crear un modelo del comportamiento de un producto aun antes de que se haya construido.

Las características generales que deben tener el software CAD/CAE son:

- Simulaciones dinámicas con características especiales de visualización de procesos y resultados (representaciones foto realistas, tabulaciones, diagramas, giros, sonido, etc.).
- Capacidad del software de generar soluciones óptimas según los tipos de aplicación. Desarrollo de sistemas virtuales dentro de un entorno, permitiendo en muchos casos eliminar los prototipos físicos.
- Ingeniería concurrente on-line (trabajo multidisciplinario vía red, con niveles de acceso y con geoprocесamiento referenciado).
- Arquitectura abierta del software (posibilidad de personalizar y generar programas complementarios - “glue functions”).

- Ingeniería inversa (obtener un modelo CAD a partir del escaneado tridimensional de una pieza real).
- Intercambio estandarizado de formatos de archivos para el trabajo multiplataforma (run anywhere).
- Pantalla de trabajo (workspace) compartidos con diferentes aplicaciones y programas adicionales (plug-ins).[8]

SolidWorks

SolidWorks es un software de diseño asistido por computadora que nos permite modelar piezas y ensamblajes en 3D y planos en 2D. El software nos ofrece la posibilidad de crear, diseñar, simular, fabricar, publicar y gestionar los datos del proceso de diseño usando sólidos paramétricos, diseño de forma que va dejando un historial de operaciones para que puedas hacer referencia a ellas en cualquier momento. De igual forma contiene funciones de simulación complejas para ayudar a probar el rendimiento del producto en circunstancias reales[9].

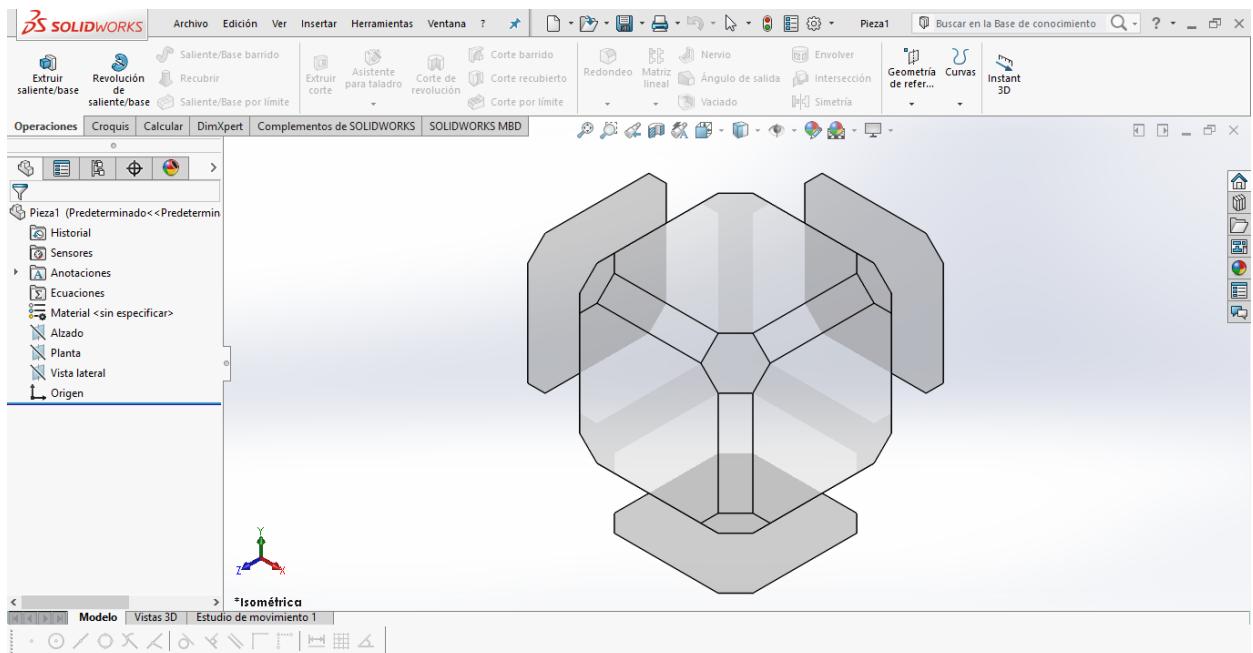


figura 9 Interfaz CAD SolidWorks .

2.2 Robótica

La robótica es una de las expresiones de la tecnología cuya aplicación se ha extendido a diversos contextos de la vida del hombre. A mediados de los noventas, se inicia la utilización de diversos tipos de plataformas de aprendizaje apoyadas por robots, se diversifica la oferta de cursos en las universidades y colegios sobre robótica, en paralelo a esta actividad, se inicia un nuevo campo de investigación y desarrollo que ha tomado el nombre de Robótica educativa[10]. De igual manera las empresas asumen el desarrollo de materiales de apoyo a las actividades en el aula; ejemplo de esto son Lego (Lego MindStorms [en línea]), VexRobotics (VEX Robotics [en línea]) y los Ataos (Ata Epe [en línea]), quienes promueven una propuesta pedagógica para ciencia y tecnología del grupo de investigación “El aprendizaje y la enseñanza” de la Escuela Pedagógica Experimental en Bogotá, Colombia [10].

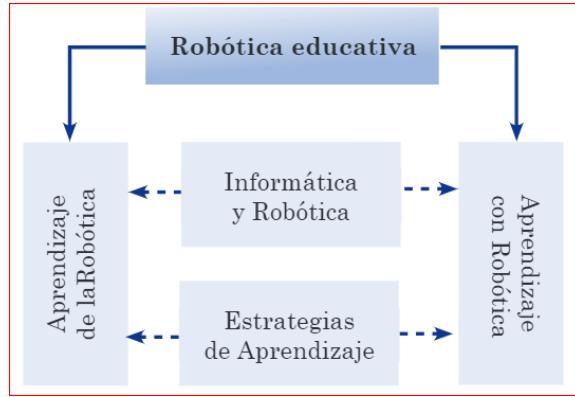


figura 10 Robótica educativa.

Otro camino para el aprendizaje sobre robots es diseñando, construyendo y controlando un robot a medida que se estudian los conceptos necesarios para hacerlo; en este proceso se integran conocimientos, se materializa una idea y se aplica el conocimiento adquirido para justificar la utilización de cada elemento en el robot y explicar su funcionamiento [11].

2.2.1 Estructura Mecánica de un Robot

2.2.2 Robots móviles

La robótica móvil autónoma es un tema de investigación fascinante, por muchas razones. Primero, cambiar un robot móvil de una computadora con ruedas que simplemente es capaz de sentir algunas propiedades físicas del medio ambiente a través de sus sensores en un agente inteligente, capaz de identificar características, detectar patrones y regularidades, aprender de la experiencia, localizar, construir mapas y navegar requiere la aplicación simultánea de muchas disciplinas de investigación. En este sentido, la robótica

móvil invierte la tendencia de la ciencia hacia una especialización cada vez mayor, y exige pensamiento lateral y la combinación de muchas disciplinas. La ingeniería y la informática son elementos centrales de la robótica móvil, obviamente, pero cuando surgen cuestiones de comportamiento inteligente, la inteligencia artificial, la ciencia cognitiva, la psicología y la filosofía ofrecen hipótesis y respuestas [12].

2.2.4 Esquema de un robot

La principal característica de funcionamiento de un robot se seccionada en diferentes áreas, las cuales trabajan de forma conjunta y en las cuales se destaca:

- Sistema Mecánico
- Sensores
- Sistema de Control
- Actuadores

Cada uno de ellos se encarga de una condición específica de funcionamiento dentro del robot [13].

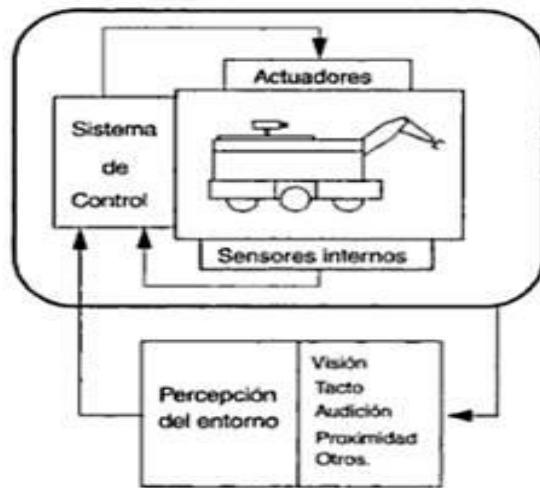


figura 11robot y su interacción con el entorno

2.2.5 locomoción

Los vehículos con ruedas son la solución más simple y eficiente para conseguir la movilidad en terrenos suficientemente duros y libres de obstáculos, permitiendo conseguir velocidades relativamente altas.

Como limitación más significativa cabe mencionar el deslizamiento en la impulsión. Dependiendo de las características del terreno pueden representar también desplazamientos y vibraciones. La locomoción mediante ruedas es poco eficiente en terrenos blandos.

Los robots móviles emplean diferentes tipos de locomoción mediante ruedas que le confieren características y propiedades diferentes respecto a la eficiencia energética, dimensiones, cargas útiles y maniobrabilidad. La mayor maniobrabilidad se consigue en vehículos omnidireccionales. Un vehículo omnidireccional en el plano es capaz de trasladarse simultáneamente e independientemente en cada eje del sistema de coordenadas, rotar según el eje perpendicular[14].

Ackerman

Es el utilizado en vehículos de cuatro ruedas convencionales. De hecho, los vehículos robóticos para exteriores resultan normalmente de la modificación de vehículos convencionales tales como automóviles o incluso vehículos más pesados. Este sistema de locomoción se ilustra en la Figura 2.4. la rueda delantera interior gira un ángulo ligeramente superior a la exterior($\theta_1 > \theta_0$) para eliminar el deslizamiento. Las prolongaciones de los ejes de las dos ruedas delanteras interceptan en un punto sobre la prolongación del eje de las ruedas trasera. El lugar de los puntos trazados sobre el suelo por los centros de los neumáticos son circunferencias concéntricas con centro el eje de rotación P1 en la figura 2.4. Si no se tienen en cuenta las fuerzas centrifugas, los vectores de velocidad instantánea son tangentes a estas curvas[14].

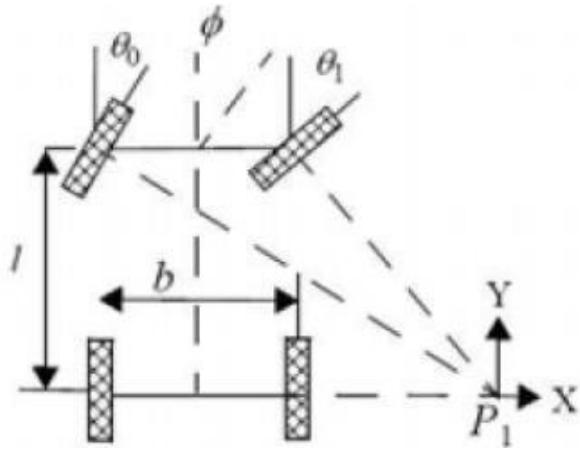


figura12 Sistema Ackerman.

Triciclo clásico

Este sistema de locomoción se ilustra en la Figura 2.5. La rueda delantera sirve tanto para la tracción como para el direccionamiento. El Eje trasero con dos ruedas laterales, es pasivo y sus ruedas se mueven libremente, esta configuración se conoce como delta. La maniobrabilidad es mayor que en la configuración anterior, pero puede presentar problemas de estabilidad en terrenos difíciles. El centro de gravedad tiende a desplazarse cuando el vehículo se mueve por una pendiente, causando la pérdida de tracción. La premisa principal de un triciclo es proporcionar una plataforma estable. El efecto de la base en un triciclo influye en la maniobrabilidad y distribución de su peso. La base de un triciclo es la longitud entre el eje de las ruedas traseras y la rueda delantera. Una base corta genera un radio de giro del triciclo muy pequeño, mientras que una base larga hace al radio de giro más grande. Adicionalmente, un triciclo con base corta exhibe mayor maniobrabilidad que un triciclo con una base larga. Debido a su simplicidad, es bastante frecuente en vehículos robóticos para interiores y exteriores pavimentados[14].

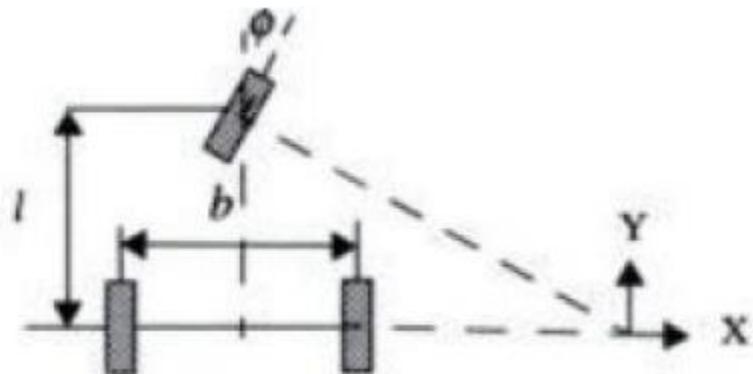


figura 13 locomoción de triciclo clásico.

Configuración síncrona

En este diseño todas las ruedas (tres o más) son tanto de dirección como de tracción; las ruedas se encuentran ubicadas de tal forma que siempre apuntan en la misma dirección. Para cambiar la dirección, el robot gira simultáneamente todas sus ruedas alrededor de un eje vertical de modo que la dirección del robot cambia, pero su chasis sigue apuntando en la misma dirección que tenía antes del giro [15]. La transmisión se consigue mediante coronas de engranes o con correas concéntricas [14]. Esta configuración se muestra en la Figura 2.6 .

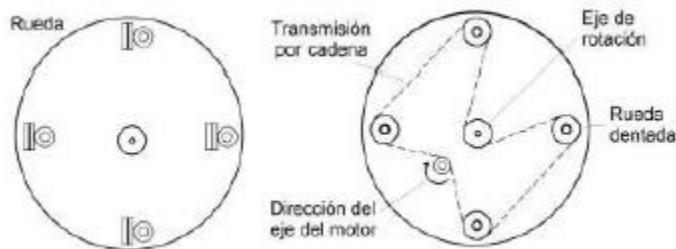


figura 14 Configuración para transmisión síncrona

Direccionamiento diferencial

El punto IRC (Centro instantáneo de rotación, por sus siglas en inglés) sobre el cual pivotea esta sobre una línea perpendicular que atraviesa el centro de las ruedas ver Figura 2.7 . El radio llega a ser mínimo cuando el punto del pivote se localiza en el punto medio entre las dos ruedas. El espacio mínimo para que el robot gire es determinado por la distancia máxima de ese punto a cualquier otro punto en el robot móvil, normalmente la esquina delantera. El robot puede moverse en línea recta, girar sobre su mismo eje y seguir trayectorias. El equilibrio del robot se obtiene mediante una o dos ruedas adicionales [15]. El direccionamiento principal viene dado por la diferencia de las velocidades (V_L y V_D) de las ruedas laterales. La tracción se consigue también con estas mismas ruedas. Adicionalmente, existe una o más ruedas para soporte [14, 15]. En la Figura 2.7 se ilustra el sistema de locomoción mencionado.

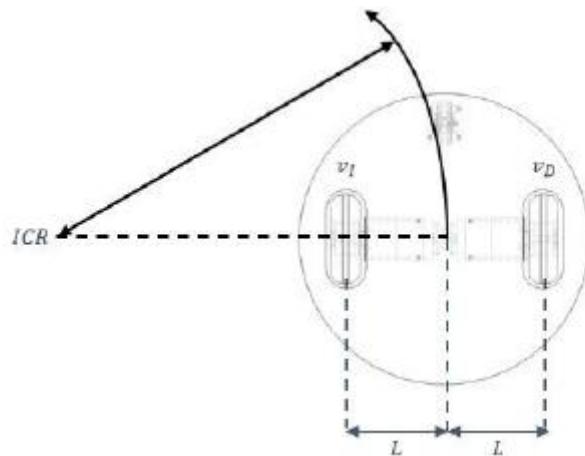


figura 15 Robot móvil en configuración diferencial.

Direccionamiento Skid Steer.

En este sistema se disponen de varias ruedas en cada lado del vehículo, las cuales actúan simultáneamente. El desplazamiento es el resultado de combinar la velocidad tanto

de la rueda de la izquierda como de la derecha. En la Figura 2.8 se muestra el “Terregator” un vehículo desarrollado en el Robotic Institute de la Carnegie Mellon University, para aplicaciones en exteriores tales como la minería. Este robot se ha aplicado también para inspección y obtención de mapas de tuberías enterradas, empleando para ello el sistema radar (Ground Penetrating Radar). [14]



figura 16 Robot Terragator con locomoción tipo "Skid Steer"

Ruedas

Existen dos tipos básicos de ruedas, las ruedas convencionales y las ruedas suecas. Con base en la clasificación presentada por Campion et al., las ecuaciones de restricción para cada uno de los tipos de ruedas están dadas por las siguientes relaciones cinemáticas [15].

Rueda fija

El centro de la rueda es el punto fijo A , cuya posición respecto al marco de referencia móvil puede ser obtenida en coordenadas polares, es decir, la distancia l de $A-P$ y el ángulo α . La orientación del plano de la rueda con respecto a l está representada por el ángulo constante β (figura 2.9). Las ecuaciones de restricción para este tipo de rueda están dadas por :

*A lo largo del plano de la rueda

$$[-\sin(\alpha + \beta) \cos(\alpha + \beta) l \cos(\beta)]R(\theta)\xi^+ + r\phi^+ = 0$$

*Ortogonal al plano de la rueda

$$[\cos(\alpha + \beta) \sin(\alpha + \beta) l \sin(\beta)]R(\theta)\xi^- = 0.$$

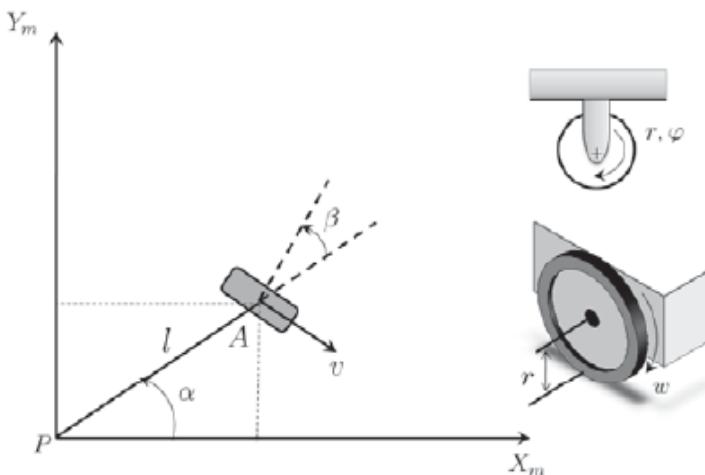


figura 17 Rueda convencional fija.

Rueda de centro orientable

Una rueda con centro orientable es aquella en la que el movimiento del plano de la rueda, con respecto a su marco de referencia, es una rotación $\beta(t)$ alrededor del eje vertical (figura 2.10), pasando a través del centro de la rueda. Sus ecuaciones de restricción son :

*A lo largo del plano de la rueda

$$[-\sin(\alpha + \beta) \cos(\alpha + \beta) l \cos(\beta)]R(\theta)\xi^+ + r\phi^+ = 0.$$

*Ortogonal al plano de la rueda

$$[\cos(\alpha + \beta) \sin(\alpha + \beta) l \sin(\beta)]R(\theta)\xi^- = 0$$

Es importante hacer notar que la descripción es la misma que para una rueda fija, excepto porque ahora el ángulo de orientación β no es constante[15].

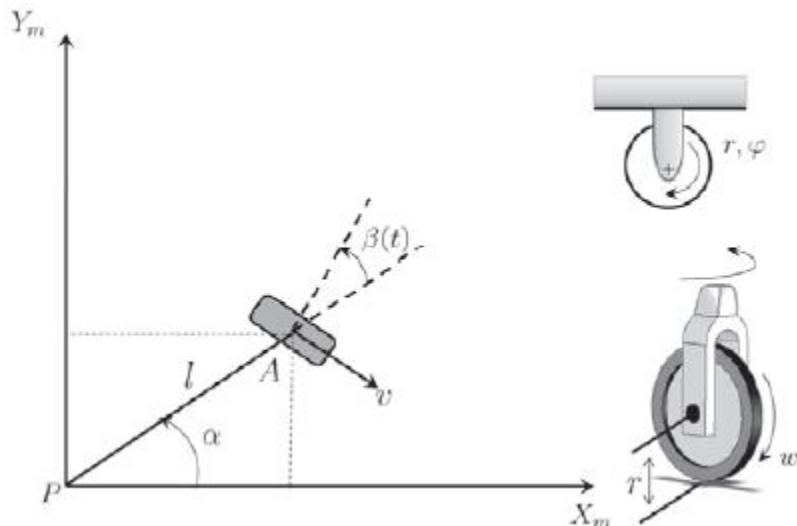


figura 18 Rueda convencional de centro orientable.

Rueda de centro orientable desplazado (giratoria)

Una rueda de centro orientable desplazado es también una rueda orientable con respecto al marco de referencia, sin embargo, la rotación del plano de la rueda es alrededor de un eje vertical que no pasa a través del centro de la rueda. El centro de la rueda es el punto B y está conectada al punto A mediante de una varilla rígida, de longitud constante d , la cual puede rotar alrededor del eje vertical que pasa por el punto A (figura 2.11). Sus ecuaciones de restricción son :

*A lo largo del plano de la rueda

$$[-\sin(\alpha + \beta) \cos(\alpha + \beta) l \cos(\beta)]R(\theta)\xi^+ + r\phi^+ = 0.$$

*Ortogonal al plano de la rueda

$$[\cos(\alpha + \beta) \sin(\alpha + \beta) d + l \sin(\beta)]R(\theta) \cdot \xi^+ + d\beta^+ = 0$$

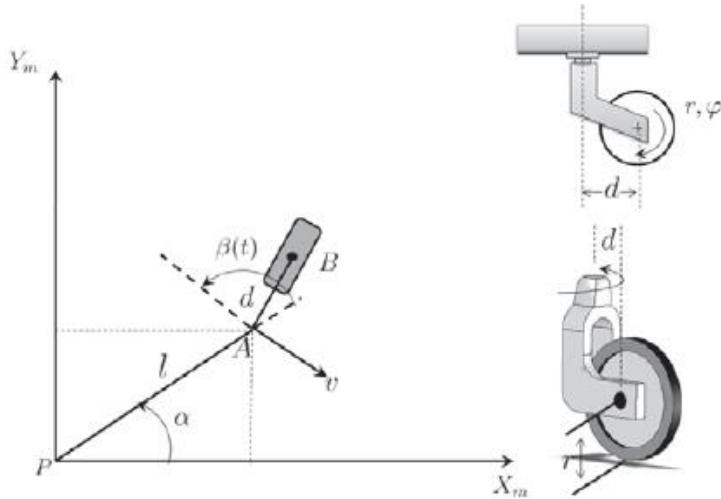


figura 19 Rueda convencional de centro orientable desplazado.

Rueda omnidireccional o sueca

La rueda omnidireccional es similar a una rueda convencional, pero posee muchas ruedas direccionales pequeñas que se mueven en un ángulo de 90 grados por su superficie de rodado. Mientras la rueda completa se mueve hacia adelante y hacia atrás, las ruedas pequeñas que cruzan su superficie se mueven indirectamente, por lo tanto, la rueda puede desplazarse en cuatro direcciones (figura 2.12) .

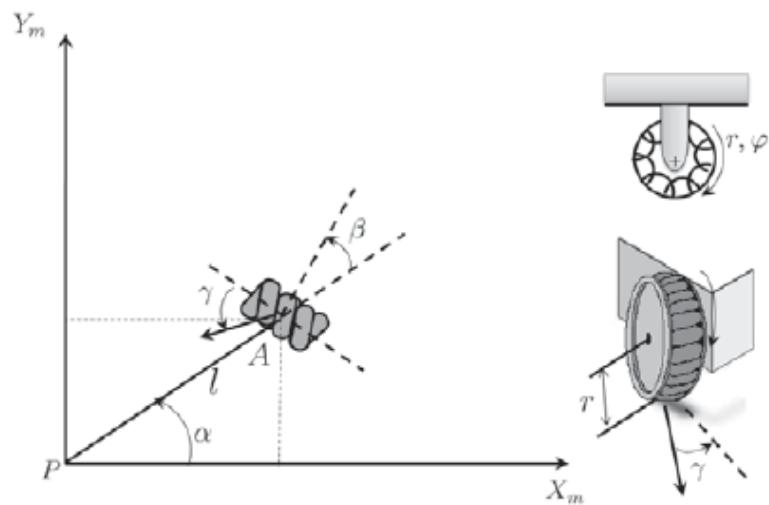


figura 20 Rueda omnidireccional o sueca.

La posición de la rueda con respecto al marco de referencia es descrita como en el caso de la rueda convencional fija. Sin embargo, se requiere un parámetro adicional γ para caracterizar la dirección con respecto al plano de la rueda. Esta configuración solo tiene una restricción [15].

*A lo largo del plano de la rueda

$$[-\sin(\alpha + \beta + \gamma) \cos(\alpha + \beta + \gamma) l \cos(\beta + \gamma)] R(\theta) \xi + r \cos(\gamma) \cdot \phi = 0.$$

Baterías

La batería VEX de 7.2 volts ofrece una gran capacidad de hasta 3000mAh, con esta batería podemos alimentar nuestros componentes ofreciendo una autonomía hasta por una hora mientras nuestra carga no sobre pase los 3 Amperios y 7.2 Voltios de lo contrario el tiempo de autonomía bajaría y dependería del consumo generado, esta batería al ser de NiMH tiene con un cargador de alimentación estándar el cual nos ofrece dos métodos de carga una rápida y otra segura, este cargador VEX robotics lo exhibe en su catálogo de productos.[16]



figura 21 Batería VEX 7.2V.

2.3 Sensores

Los sensores se encargan de proporcionar la información necesaria para realizar el control del robot. Los sensores se los puede emplear para la medición de diversos principios físicos y químicos.

En la actualidad la mayoría de sensores emplean un procesamiento electrónico, y para poder ser empleadas es necesario que estas sean señales eléctricas, teniendo en cuenta que en muchas o en casi todas las aplicaciones de estos dispositivos son varios datos que se requieren conocer, tales como, la magnitud y sus respectivas derivadas [17].

2.3.1 Ultrasónicos SRF08

SRF08 es un medidor ultrasónico de distancias para robots que representa la última generación en sistemas de medidas de distancias por sonar, consiguiendo niveles de precisión y alcance únicos e impensables hasta ahora con esta tecnología. El sensor es capaz de detectar objetos a una distancia de 11 m con facilidad además de conectarse al microcontrolador mediante un bus I2C, por lo que se pueden conectar hasta 16 sensores. Con una alimentación única de 5V, solo requiere 15 mA, para funcionar y 3mA mientras estado de reposo, lo que representa una gran ventaja para robots alimentados por pilas. El sensor SRF08 Incluye además un sensor de luz que permite conocer el nivel de luminosidad usando igualmente el bus I2C y sin necesidad de recursos adicionales [17].



figura 22 Sensor Ultrasónico SRF08

Controlando el sensor de distancias ultrasónico SRF08

La comunicación con el sensor ultrasónico SRF08 se realiza a través del bus I2C. Este está disponible en la mayoría de los controladores del mercado como BasicX-24, OOPic y Basic Stamp 2P, así como en una amplia gama de microcontroladores. Para el programador, el sensor SRF08 se comporta de la misma manera que las EEPROM de las series 24xx, con la excepción de que la dirección I2C es diferente. La dirección por defecto de fábrica del SRF08 es 0xE0. El usuario puede cambiar esta dirección con 16 direcciones diferentes: E0, E2, E4, E6, E8, EA, EC, EE, F0, F2, F4, F6, F8, FA, FC o FE, por lo que es posible utilizar hasta 16 sensores sobre un mismo bus I2C. Además de las direcciones anteriores, todos los sonares conectados al bus I2C responderán a la dirección 0 -al ser la dirección de atención general. Esto significa que escribir un comando de medición de la distancia para la dirección 0 de I2C (0x00) iniciará las mediciones en todos los sensores al mismo tiempo. Esto debería ser útil en el modo ANN. Los resultados deben leerse de manera individual de cada uno de las direcciones reales de los sensores. Disponemos de

ejemplos del uso de un módulo SRF08 con una amplia gama de controladores del mercado[17].

Conexiones

El pin señalado como "Do Not Connect" (No conectar) debería permanecer sin conexión. En realidad, se trata de la línea MCLR de la CPU y se utiliza solamente en la fábrica para programar el PIC16F872 después del montaje, dispone de una resistencia interna de tipo pull-up. Las líneas SCL y SDA deberían tener cada una de ellas una resistencia pull-up de +5v en el bus I2C. Sólo necesita un par de resistencias en todo el bus, no un par por cada módulo o circuito conectado al bus I2C. Normalmente se ubican en el bus maestro en vez de en los buses esclavos. El sensor SRF08 es siempre un bus esclavo - y nunca un bus maestro. Un valor apropiado sería el de 1,8 K en caso de que las necesitase. Algunos módulos como el OOPic ya disponen de resistencias pull-up por lo que no es necesario añadir ninguna más[17].

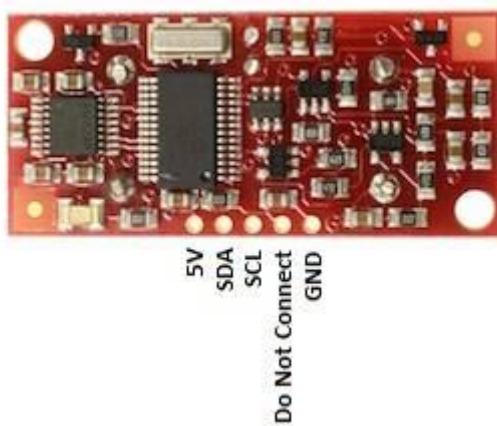


figura 23 Conexiones del sensor ultrasónico SFR08

Registros

El sensor SRF08 tiene un conjunto de 36 registros.

UBICACIÓN	LECTURA	ESCRITURA
0	Revisión de Software	Registro de comando
1	Sensor de luz	Registro de ganancia máx. (por defecto 31)
2	Byte alto de 1º eco	Registro de alcance de distancia (por defecto 255)
3	Byte alto de 2º eco	No disponible
...
34	Byte alto de 17º eco	No disponible
35	Byte alto de 17º eco	No disponible

Tabla 1 registros del SFR08

Solamente se puede escribir en las ubicaciones 0, 1 y 2. La ubicación 0 es el registro de comandos y se utiliza para iniciar la sesión de cálculo de la distancia. No puede leerse. La lectura de la ubicación da como resultado la revisión del software de SRF08. Por defecto, la medición dura 65mS, aunque puede cambiarse modificando el registro de alcance de la ubicación 2. Si lo hace, tendrá que cambiar la ganancia analógica en la

ubicación 1. Consulte las secciones siguientes relacionadas con el cambio de medición y ganancia analógica.

La ubicación 1 es el sensor de luz en placa. Este dato se actualiza cada vez que se ejecuta un comando de medición de distancia y se puede leer cuando se leen los datos de la medición. Las dos ubicaciones siguientes, 2 y 3, son resultados sin signo de 16 bits de la última medición - el nivel lógico alto en primer lugar. El significado de este valor depende del comando utilizado, y puede estar expresado en pulgadas, o en centímetros, o bien el tiempo de vuelo del ping expresado en uS. Un valor cero indica que no se ha detectado objeto alguno. Hay hasta 16 resultados adicionales que indican los ecos de objetos más lejanos[17].

Rango de alcance del sensor ultrasónico SFR08

El alcance máximo del sensor SRF08 está controlado por el temporizador interno. Por defecto, este es 65mS o el equivalente a 11 metros de alcance. Esto supera los 6 metros de los que el SRF04 es realmente capaz de ofrecer. Es posible reducir el tiempo que espera el sensor SRF08 a escuchar un eco, y por lo tanto el alcance, modificando el registro range en la ubicación 2. El alcance puede regularse en pasos de aproximadamente 43mm (0,043 metros o 1,68 pulgadas) hasta llegar a los 11 metros. El alcance es ((Range Register x 43mm) + 43mm) por lo que fijar este registro (Range Register) en el valor 0 (0x00) ofrece un alcance máximo de 43mm. Fijar el registro Range Register en el valor 1 (0x01) ofrece un alcance máximo de 86mm. En un ejemplo más útil, el valor 24 (0x18) ofrece un alcance de 1 metro mientras que el valor 140 (0x8C) da 6 metros. El valor 255 (0xFF) ofrece los 11 metros originales(255 x 43 + 43 es 11008mm).

Existen dos razones por las que es positivo reducir el tiempo de medición.

1. Para obtener la información sobre el alcance en menos tiempo

2. Para poder realizar mediciones con el sensor SRF08 a una tasa más rápida.

Si lo único que desee en recibir en menos tiempo, la información sobre el alcance y pretende realizar las mediciones a una tasa de 65ms o más lento, todo funcionará de manera correcta. Sin embargo, si desea lanzar el sensor SRF08 a una tasa ligeramente más alta de 65mS, deberá reducir la ganancia - consulte la siguiente sección.

El alcance está fijado en el valor máximo cada vez que se pone en marcha el sensor SRF08. Si necesita un alcance diferente, cámbielo al principio como parte del código de iniciación del sistema[17].

Sensor de luz

El medidor ultrasónico SRF08 dispone de un sensor fotoeléctrico en la propia placa. Este medidor realiza una lectura de la intensidad de la luz cada vez que se calcula la distancia en los modos Ranging o ANN (La conversión analógica/digital se realiza realmente justo antes de que se lance el "ping" mientras el generador de 10v +/- se encuentra en fase de estabilización). El valor de la lectura va aumentando a medida que aumenta la intensidad de la luz, por lo que el valor máximo lo obtendrá con una luz brillante y el valor mínimo en total oscuridad. La lectura debería acercarse a 2-3 en total oscuridad y aproximadamente a 248 (0xF8) en luz diurna. La intensidad de la luz puede leerse en el registro del sensor de luz en la ubicación 1 al mismo tiempo que puede leer los datos del alcance[17].

LED

EL indicador LED rojo se utiliza para indicar el código de la dirección I2C del sensor en el encendido (ver abajo). Así mismo, también emite un breve destello durante el "ping" en el cálculo de la distancia[17].

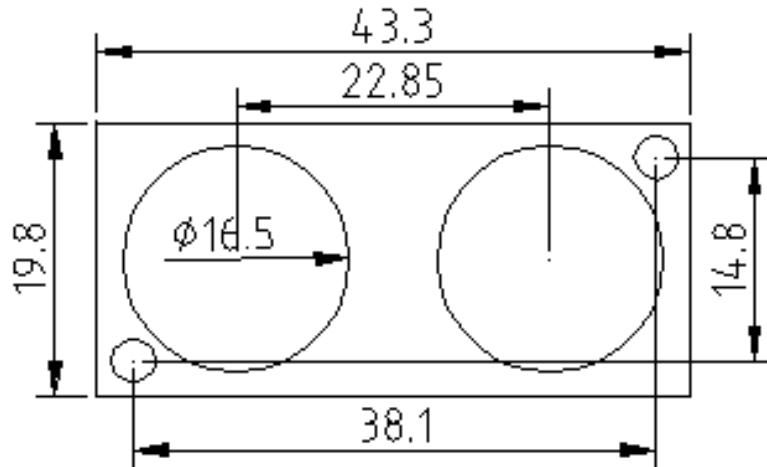


figura 24 Dimensiones del sensor ultrasónico SFR08

Consumo de corriente

El consumo medio de corriente se calcula que es aproximadamente 12mA durante el cálculo de la distancia, y 3mA en modo de espera. El módulo entrará automáticamente en modo de espera después de terminar la medición, mientras espera al siguiente comando del bus I2C. El perfil real de consumo de corriente del srf08 es el siguiente[17]:

Tipo de operación realizada	Corriente	Duración
Comando de medición de la distancia recibido -Encendido	275mA	3uS
Estabilización del generador de +/- 10v	25mA	600uS
8 ciclos de "ping" 40kHz	40mA	200uS
Medición	1mA	65mS máx.
Modo de espera (Stand-by)	3mA	Indefinido

Tabla 2 Corriente utilizada por el sensor SFR08

2.3.2 Giroscopio

Los dispositivos MEMS se fabrican sobre un sustrato de silicio de la misma forma que los circuitos integrados de un solo chip, por lo que sus tamaños varían desde unas pocas decenas de micrones hasta varios milímetros. Existen varios tipos de giroscopios MEMS que se diferencian en su estructura interna, pero todos ellos están unidos en el hecho de que su trabajo se basa en el uso de la fuerza de Coriolis. En cada uno de ellos hay un cuerpo de trabajo, realizando movimientos alternativos. Si gira el sustrato sobre el que se encuentra este cuerpo, entonces la fuerza de los Coriolis, que se dirige perpendicularmente al eje de rotación y la dirección de movimiento del cuerpo, comenzará a actuar sobre él. La Figura 2.17 muestra una ilustración para comprender el principio de esta fuerza.

Conociendo la velocidad lineal y la fuerza Coriolis , se puede determinar la velocidad angular. Una de las posibles implementaciones de un giroscopio tiene la siguiente estructura: un marco fijado sobre perchas flexibles, dentro del cual una masa

realiza movimientos oscilatorios de traslación [18]. La estructura de dicho sensor se muestra en la figura 2.18.

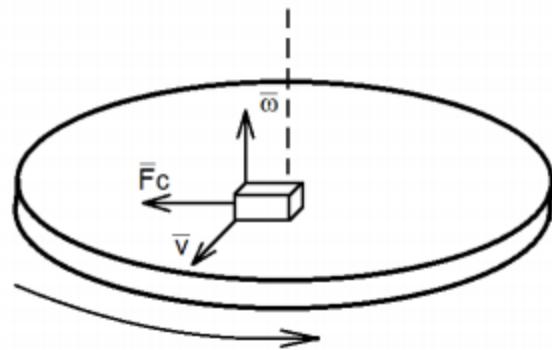


figura 25 El mecanismo de la fuerza Coriolis

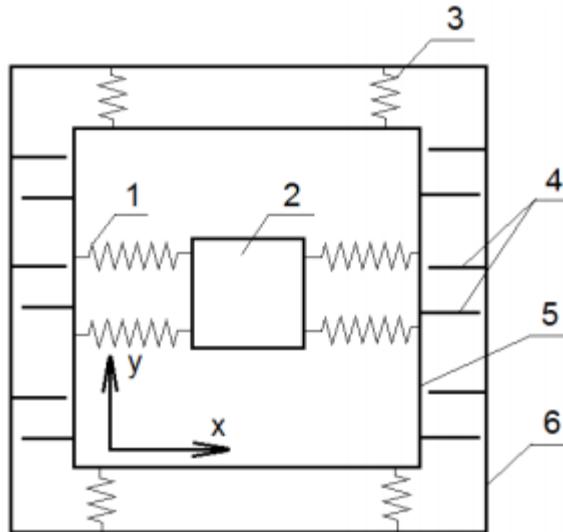


figura 26 La estructura interna del giroscopio: 1 - masa de sujeción, 2 - peso de trabajo, 3 - fijación del marco interior, 4 - sensores que mueven el marco interior, 5 - marco interior, 6 – sustrato.

MPU6050

El MPU-60X0 es el primer dispositivo MotionTracking integrado de 6 ejes del mundo que combina un dispositivo de 3 ejes giroscopio, acelerómetro de 3 ejes y un procesador de movimiento digital (DMP) todo en un pequeño paquete 4x4x0.9mm . Con su bus de sensores I2C dedicado, acepta directamente entradas de una brújula externa de 3 ejes para proporcionar una salida completa de 9 ejes MotionFusion. El dispositivo MotionTracking MPU-60X0, con sus 6 ejes integración, a bordo MotionFusion, y firmware de calibración de tiempo de ejecución, permite a los fabricantes eliminar la costosa y compleja selección, cualificación e integración de nivel de sistema de dispositivos discretos, garantizando rendimiento de movimiento óptimo para los consumidores. El MPU-60X0 también está diseñado para interactuar con múltiples sensores digitales no iniciales, tales como sensores de presión, en su Puerto I2C.

El MPU-60X0 cuenta con tres convertidores analógico-digitales (ADC) de 16 bits para digitalizar las salidas del giroscopio y tres ADCs de 16 bits para digitalizar las salidas del acelerómetro. Para un seguimiento preciso tanto rápido como lento las piezas presentan un rango de escala completa programable por el usuario de 250, 500, 1000, y 2000 ° / seg (dps) y un acelerómetro programable por el usuario con un rango de escala completa de 2g , 4g, 8g y 16g.

Un búfer FIFO de 1024 bytes en el chip ayuda a reducir el consumo de energía del sistema al permitir que el procesador del sistema lea los datos del sensor en ráfagas y luego ingrese en un modo de bajo consumo a medida que la MPU recopila más datos. Con todos los componentes de sensor y procesamiento en chip necesarios para admitir muchos casos de uso, el MPU-60X0 permite de manera única aplicaciones MotionInterface de bajo consumo en aplicaciones portátiles con requisitos de procesamiento reducidos para el procesador del sistema. Al proporcionar una salida MotionFusion integrada, el DMP en el MPU-60X0 descarga los requisitos de cálculo intensivo de MotionProcessing del procesador del sistema, minimizando la necesidad de sondeo frecuente de la salida del sensor de movimiento [19].

Características técnicas:

- giroscopio de 3 ejes.
- Acelerómetro de 3 ejes.
- Sensor termal.
- Fuente de alimentación 2.375V-3.46V.
- Búfer FIFO de 1024 bytes.
- Filtros digitales programables por el usuario para el giroscopio, acelerómetro y sensor térmico.
- Interfaz I2C para escritura y lectura de registros de dispositivos
- Rango de medición programable por el usuario: 250, 500, 1000, y 2000 ° / s.
- ADC incorporado de 16 bits.
- Filtro de paso bajo programable digital.
- Corriente en modo de operación - 3.6 mA.
- Corriente de espera 5 µA[19].

2.4 Motores

2.4.1 Motor DC

Los motores de DC están constituidos por dos devanados internos, inductor e inducido, que se alimentan con corriente continua: el inductor, también denominado devanado de excitación, está situado en el estator y crea un campo magnético de dirección fija, denominada excitación. El inducido, situado en el rotor, hace girar al mismo debido a la fuerza de lorentz que aparece como combinación de la corriente circulante por él y del campo magnético de excitación. Recibe la corriente del exterior a través del colector de delgas, en el que se apoyan unas escobillas de grafito.

Para que se pueda dar la conversión de energía eléctrica en energía mecánica de forma continua es necesario que los campos magnéticos del estator y del rotor permanezcan estáticos. Esta transformación es máxima cuando ambos campos se encuentran en cuadratura. El colector de delgas es un conmutador sincronizado con el rotor encargado de que se mantenga el ángulo relativo entre el campo del estator y el creado por las corrientes rotoricas. De esta forma se consigue transformar automáticamente, en función de la velocidad de la máquina, la corriente continua que alimenta al motor en corriente alterna de frecuencia variable en el inducido este tipo de funcionamiento se conoce con el nombre de auto pilotado.

Al aumentar la tensión del inducido aumenta la velocidad de la máquina. Si el motor esta alimentado a tensión constante, se puede aumentar la velocidad disminuyendo el flujo de excitación. Pero cuanto más débil sea el flujo, menor será el par motor que se puede desarrollar para una intensidad de inducido constante, mientras que la tensión del inducido se utiliza para controlar la velocidad de giro [20].

2.4.2 Motor paso a paso

Los motores paso a paso son dispositivos electromagnéticos, rotativos, incrementales que convierten pulsos digitales en rotación mecánica. La cantidad de rotación es directamente proporcional al número de pulsos y la velocidad de rotación es relativa a la frecuencia de los pulsos. Estos motores son simples de operar en una configuración de lazo cerrado y debido a su tamaño proporcionan un excelente torque a baja velocidad. Entre los beneficios de estos motores se incluyen:

- Un diseño efectivo y un bajo costo
- Alta confiabilidad
- Libres de mantenimiento (no disponen de escobillas)
- Lazo abierto (no requieren dispositivos de realimentación)

- Límite conocido al "error de posición dinámica"

A pesar de que varios tipos de motores paso a paso han sido desarrollados, todos se enmarcan dentro de tres categorías básicas.

- De reluctancia variable (V.R)
- De magneto permanente
- Híbridos

El motor de Reluctancia variable o V.R (figura 2.19) consiste en un motor y un estator cada uno con un numero diferente de dientes. Como el Rotor no dispone de un magneto permanente, la misma gira libremente, ósea que no tiene torque de detección. A pesar de que la relación del torque a la inercia es buena, el torque dado para un tamaño de armazón específico es restringido, por lo tanto, tamaños pequeños de armazones son por lo general usados y raramente varían para industriales.



figura 27 Vista de sección de un motor paso a paso de reluctancia variable

El motor de magneto permanente o tipo enlatado (figura 2.20) es quizá el motor paso a paso más usado para aplicaciones no industriales. En su forma más simple, el motor consiste en un rotor de magneto permanente con magnetizado radial y en un estator similar

al motor V.R. debido a las técnicas de manufacturación usadas en la construcción del estator, ellos se conocen a veces como motores de “polo de uñas” o “claw pole” en inglés.

El tipo híbrido es muy probable que sea el más usado de todos los motores paso a paso.

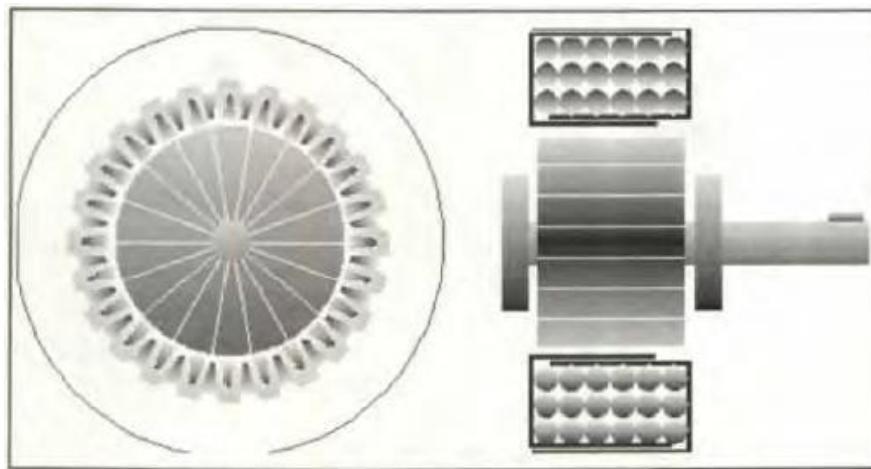


figura 28 Vista en sección de un magneto permanente

Originalmente desarrollado como un motor PM síncrono de baja velocidad, su construcción es una combinación de los diseños V.R y P.M. El motor híbrido consiste de un estator dentado y un rotor de tres partes(apilado simple). El rotor de apilado simple contiene dos piezas de polos separados por un magneto permanente magnetizado con dos dientes opuestos desplazados en una mitad de un salto de diente (figura 2.21) para permitir una alta resolución de pasos[21].

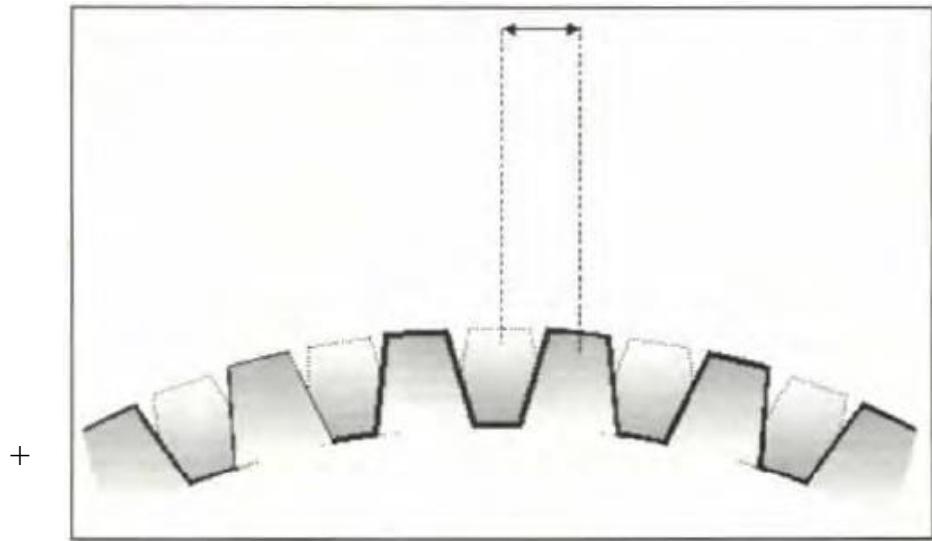


figura 29 Vista expandida ilustrativa del desplazamiento de dientes.

2.4.3 Motor VEX EDR 393

Los motores VEX 393 son los motores principales que se utilizan para los mecanismos de los robots. También ofrecen engranajes de repuesto para aumentar la velocidad de los motores.

El motor de 2 cables 393 es el actuador principal utilizado en el VEX EDR. Utilizado para mecanismos de rotación, bases de accionamiento, juntas de rotación, cintas transportadoras; cualquier cosa que gire se puede construir con el Motor 393 de 2 cables.

- Engranajes internos de acero resistentes.

Puede configurarse en una versión de "alta velocidad". El motor de 2 cables 393 no tiene un controlador de motor interno[22].

Características del motor de 2 cables (Motor VEX EDR 393)

- 0.192 lbs (87.1 gramos)
- Tornillo # 6-32 x 1/4 "0.0014 lbs (0.617 gramos)
- Tornillo # 6-32 x 1/2" 0.00209 lbs (0.948 gramos)
- Acoplador del motor 0.004 lbs (1.814 gramos))
- Poste del motor 0,002 libras (0,907 gramos)
- Velocidad libre: 100 rpm (como se envía) / 160 rpm (opción de alta velocidad)
- Par de bloqueo: 1,67 Nm (14,76 pulg-lbs) (como se envía) / 1,04 Nm (9,2 pulglbs) (opción de alta velocidad)
- Corriente de bloqueo: 4.8A
- Corriente libre: 0.37A

Nota: Todas las especificaciones del motor están a 7,2 voltios.

2.5 Arduino

2.5.1 Arduino Mega

El Arduino Mega 2560 es una placa de microcontrolador basada en el ATmega2560 . Tiene 54 pines de entrada / salida digital (de los cuales 15 se pueden usar como salidas PWM), 16 entradas analógicas, 4 UART (puertos serie de hardware), un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, un encabezado ICSP, y un botón de reinicio. Contiene todo lo necesario para soportar el microcontrolador; simplemente conéctelo a una computadora con un cable USB o enciéndalo con un adaptador de CA a CC o una batería para comenzar. La placa Mega

2560 es compatible con la mayoría de los escudos diseñados para la Uno y las antiguas placas Duemilanove o Diecimila [23].



figura 30 Arduino Mega 2560 REV 3

2.6 Raspberry Pi

La Raspberry Pi Foundation es una organización benéfica con sede en el Reino Unido que trabaja para poner el poder de la informática y la creación digital en manos de personas de todo el mundo. Hacemos esto para que más personas puedan aprovechar el poder de la computación y las tecnologías digitales para trabajar, para resolver problemas que les importan y para expresarse de manera creativa [24].

2.6.1 raspberry Pi 3b+

La Raspberry Pi 3 Modelo B + es el último producto de la gama Raspberry Pi 3, con un procesador de cuatro núcleos de 64 bits que funciona a 1.4GHz, LAN inalámbrica de banda dual de 2.4GHz y 5GHz, Bluetooth 4.2 / BLE, Ethernet más rápido y PoE

capacidad a través de un PoE HAT separado la LAN inalámbrica de doble banda viene con certificación de cumplimiento modular, lo que permite que la placa se diseñe en productos finales con pruebas de cumplimiento de LAN inalámbrica significativamente reducidas , lo que mejora tanto el costo como el tiempo de comercialización. La Raspberry Pi 3 Modelo B + mantiene la misma huella mecánica que la Raspberry Pi 2 Modelo B y la Raspberry Pi 3 Modelo B [25].

Especificación

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) SoC de 64 bits a 1,4 GHz
- SDRAM LPDDR2 de 1 GB
- LAN inalámbrica IEEE 802.11.b / g / n / ac de 2,4 GHz y 5 GHz, Bluetooth 4.2, BLE
- Gigabit Ethernet sobre USB 2.0 (rendimiento máximo 300 Mbps)
- Cabecera GPIO extendida de 40 pines
- HDMI de tamaño completo
- 4 puertos USB 2.0
- Puerto de cámara CSI para conectar una cámara Raspberry Pi
- Puerto de pantalla DSI para conectar una pantalla táctil Raspberry Pi
- Salida estéreo de 4 polos y puerto de video compuesto
- Puerto micro SD para cargar su sistema operativo y almacenar datos
- Entrada de alimentación de 5 V / 2,5 A CC
- Soporte Power-over-Ethernet (PoE) (requiere PoE HAT separado) [25].



figura 31 Raspberry pi 3B+

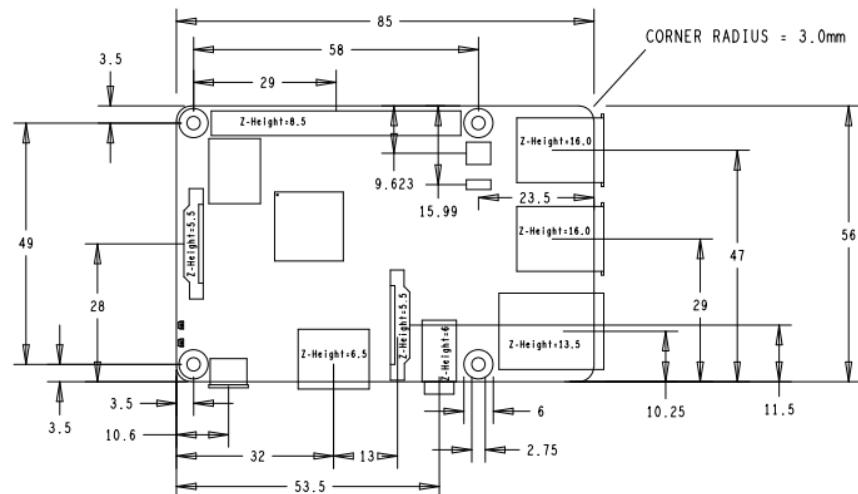


figura 32 Dimensiones de la Raspberry pi 3B+

2.7 Matlab

MATLAB combina un entorno de escritorio en sintonía para los procesos de análisis y diseño iterativos con un lenguaje de programación que expresa la matriz y las matemáticas de matriz directamente. Incluye el Live Editor para crear scripts que combinan código, salida y texto formateado en un cuaderno ejecutable.

Construido profesionalmente

Las cajas de herramientas de MATLAB se desarrollan profesionalmente, se prueban rigurosamente y se documentan completamente.

Con aplicaciones interactivas

Las aplicaciones de MATLAB permiten ver cómo funcionan los diferentes algoritmos con nuestros datos. Repetir hasta obtener los resultados deseados, luego generar automáticamente un programa MATLAB para reproducir o automatizar nuestro trabajo.

Capacidad de escalar

Escalar análisis para que se ejecuten en clústeres, GPU y nubes con solo pequeños cambios de código. No es necesario reescribir nuestro código o aprender técnicas de programación de big data y de memoria insuficiente[26].

2.7.1 Redes Neuronales Artificiales

Una red neuronal (también llamada red neuronal artificial) es un sistema adaptativo que aprende mediante el uso de nodos o neuronas interconectados en una estructura en capas que se asemeja a un cerebro humano. Una red neuronal puede aprender de los datos, por lo que puede entrenarse para reconocer patrones, clasificar datos y pronosticar eventos futuros.

Una red neuronal divide la entrada en capas de abstracción. Se puede entrenar usando muchos ejemplos para reconocer patrones en el habla o imágenes, por ejemplo, tal como lo hace el cerebro humano. Su comportamiento se define por la forma en que sus elementos individuales están conectados y por la fuerza, o pesos, de esas conexiones. Estos pesos se ajustan automáticamente durante el entrenamiento de acuerdo con una regla de aprendizaje específica hasta que la red neuronal artificial realiza correctamente la tarea deseada[27].

¿Por qué son importantes las redes neuronales?

Las redes neuronales son especialmente adecuadas para realizar el reconocimiento de patrones para identificar y clasificar objetos o señales en los sistemas de control, visión y habla. También se pueden utilizar para realizar predicciones y modelos de series de tiempo.

A continuación, se muestran algunos ejemplos de cómo se utilizan las redes neuronales artificiales:

- Detectar la presencia de comandos de voz en el audio entrenando un modelo de aprendizaje profundo.
- Aplicar la apariencia estilística de una imagen al contenido de la escena de una segunda imagen mediante la transferencia de estilo neuronal .
- Conversión de caracteres japoneses escritos a mano en texto digital .
- Detectar el cáncer guiando a los patólogos en la clasificación de tumores como benignos o malignos, según la uniformidad del tamaño celular, el grosor del grupo, la mitosis y otros factores[27].

¿Cómo funcionan las redes neuronales?

Una red neuronal combina varias capas de procesamiento, utilizando elementos simples que operan en paralelo e inspirados en sistemas nerviosos biológicos. Consiste en

una capa de entrada, una o más capas ocultas y una capa de salida. En cada capa hay varios nodos, o neuronas, y cada capa usa la salida de la capa anterior como entrada, por lo que las neuronas interconectan las diferentes capas. Cada neurona normalmente tiene pesos que se ajustan durante el proceso de aprendizaje y, a medida que el peso aumenta o disminuye, cambia la intensidad de la señal de esa neurona[27].

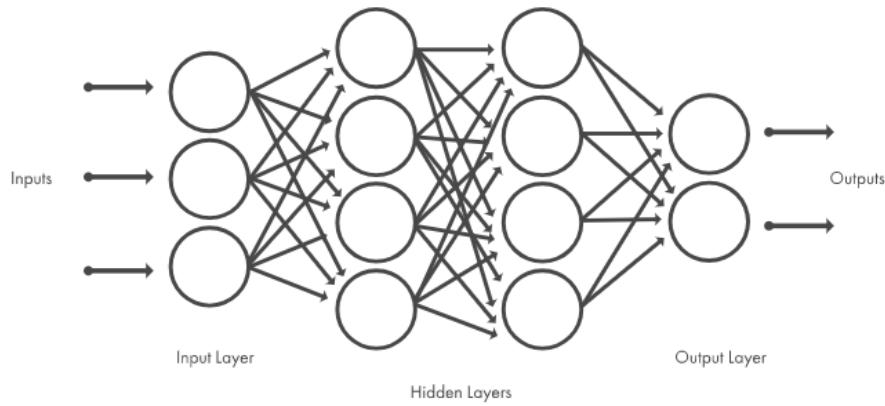


figura 33 Arquitectura de una red neuronal típica

2.7.2 Deep Learning

El aprendizaje profundo es una técnica de aprendizaje automático que enseña a las computadoras a hacer lo que es natural para los humanos: aprender con el ejemplo. El aprendizaje profundo es una tecnología clave detrás de los automóviles sin conductor, que les permite reconocer una señal de alto o distinguir a un peatón de una farola. Es la clave para el control por voz en dispositivos de consumo como teléfonos, tabletas, televisores y altavoces de manos libres. El aprendizaje profundo está recibiendo mucha atención últimamente y por una buena razón. Está logrando resultados que antes no eran posibles.

En el aprendizaje profundo, un modelo de computadora aprende a realizar tareas de clasificación directamente a partir de imágenes, texto o sonido. Los modelos de aprendizaje

profundo pueden lograr una precisión de vanguardia, a veces superando el rendimiento a nivel humano. Los modelos se entran mediante el uso de un gran conjunto de datos etiquetados y arquitecturas de redes neuronales que contienen muchas capas[28].

¿Cómo logra el aprendizaje profundo resultados tan impresionantes?

En una palabra, precisión. El aprendizaje profundo logra una precisión de reconocimiento a niveles más altos que nunca. Esto ayuda a que la electrónica de consumo cumpla con las expectativas de los usuarios y es crucial para aplicaciones críticas para la seguridad, como los automóviles sin conductor. Los avances recientes en el aprendizaje profundo han mejorado hasta el punto en que el aprendizaje profundo supera a los humanos en algunas tareas como clasificar objetos en imágenes.

Si bien el aprendizaje profundo se teorizó por primera vez en la década de 1980, hay dos razones principales por las que solo recientemente se ha vuelto útil:

1. El aprendizaje profundo requiere grandes cantidades de datos etiquetados . Por ejemplo, el desarrollo de automóviles sin conductor requiere millones de imágenes y miles de horas de video.
2. El aprendizaje profundo requiere una potencia informática sustancial . Las GPU de alto rendimiento tienen una arquitectura paralela que es eficiente para el aprendizaje profundo. Cuando se combina con clústeres o computación en la nube, esto permite a los equipos de desarrollo reducir el tiempo de capacitación para una red de aprendizaje profundo de semanas a horas o menos[28].

Ejemplos de aprendizaje profundo en el trabajo

Las aplicaciones de aprendizaje profundo se utilizan en industrias que van desde la conducción automatizada hasta los dispositivos médicos.

Conducción automatizada: los investigadores automotrices están utilizando el aprendizaje profundo para detectar automáticamente objetos como señales de alto y

semáforos. Además, el aprendizaje profundo se utiliza para detectar peatones, lo que ayuda a disminuir los accidentes.

Aeroespacial y defensa: el aprendizaje profundo se utiliza para identificar objetos de satélites que ubican áreas de interés e identificar zonas seguras o inseguras para las tropas.

Investigación médica: los investigadores del cáncer están utilizando el aprendizaje profundo para detectar automáticamente las células cancerosas. Los equipos de UCLA construyeron un microscopio avanzado que produce un conjunto de datos de alta dimensión que se utiliza para entrenar una aplicación de aprendizaje profundo para identificar con precisión las células cancerosas.

Automatización industrial: el aprendizaje profundo está ayudando a mejorar la seguridad de los trabajadores alrededor de maquinaria pesada al detectar automáticamente cuando hay personas u objetos a una distancia insegura de las máquinas.

Electrónica: el aprendizaje profundo se está utilizando en la traducción automática del habla y la audición. Por ejemplo, los dispositivos de asistencia domiciliaria que responden a su voz y conocen sus preferencias funcionan con aplicaciones de aprendizaje profundo[28].

2.7.3 Machine Learning

El aprendizaje automático es una técnica de análisis de datos que enseña a las computadoras a hacer lo que es natural para los humanos y los animales: aprender de la experiencia. Los algoritmos de aprendizaje automático utilizan métodos computacionales para "aprender" información directamente de los datos sin depender de una ecuación predeterminada como modelo. Los algoritmos mejoran su rendimiento de forma adaptativa a medida que aumenta el número de muestras disponibles para el aprendizaje. El aprendizaje profundo es una forma especializada de aprendizaje automático[29].

¿Por qué es importante el aprendizaje automático?

Con el aumento del big data , el aprendizaje automático se ha convertido en una técnica clave para resolver problemas en áreas como:

- Finanzas computacionales , para puntuación crediticia y negociación algorítmica
- Procesamiento de imágenes y visión por computadora , para reconocimiento facial, detección de movimiento y detección de objetos
- Biología computacional , para detección de tumores, descubrimiento de fármacos y secuenciación de ADN
- Producción de energía , para previsión de precios y carga
- Automotriz, aeroespacial y de fabricación , para mantenimiento predictivo
- Procesamiento de lenguaje natural , para aplicaciones de reconocimiento de voz [29].

2.7.4 Rasberry pi & Matlab

La programación de Raspberry Pi generalmente implica trabajar con imágenes, videos, audio y otros datos de sensores.

MATLAB y Simulink ayudan a los usuarios analizar rápidamente y visualizar estos datos en el contexto de su programación Frambuesa Pi. Los productos admiten dos flujos de trabajo principales:

- Leer, escribir y analizar datos de sensores y cámaras Raspberry Pi
- Desarrollo de algoritmos que se ejecutan de forma independiente en Raspberry Pi[30].

Leer, escribir y analizar datos de sensores y cámaras Raspberry Pi

Con el paquete de soporte de MATLAB para Raspberry Pi , puede escribir programas MATLAB que se comuniquen con su Raspberry Pi y adquirir datos de los pines GPIO de la placa, cámaras y otros dispositivos conectados. Debido a que MATLAB es un lenguaje interpretado de alto nivel, es fácil crear prototipos y refinar algoritmos para sus proyectos Raspberry Pi. MATLAB incluye miles de funciones matemáticas y de trazado integradas que puede utilizar para la programación de Raspberry Pi en áreas como aprendizaje profundo , procesamiento de imágenes y video, optimización, estadísticas y procesamiento de señales.

Con MATLAB para la programación de Raspberry Pi, puede:

- Analizar los datos del sensor Raspberry Pi y desarrollar algoritmos utilizando miles de funciones prediseñadas para procesamiento de imágenes , procesamiento de señales , modelado matemático y más
- Visualizar rápidamente nuestros datos utilizando la amplia gama de tipos de gráficos de MATLAB
- Utilizar el mismo software para programar otros dispositivos de hardware como Arduino .
- Desarrollar algoritmos que se ejecuten de forma independiente en la Raspberry Pi

Hay dos opciones para desarrollar algoritmos que se ejecutan de forma independiente en la Raspberry Pi. El primero es compatible con Raspberry Pi de MATLAB Coder . Puede generar código C legible y portátil a partir de algoritmos de MATLAB e implementarlo como una aplicación independiente en Raspberry Pi.

El segundo es con el paquete de soporte Simulink para Raspberry Pi . Puede desarrollar algoritmos en Simulink, un entorno de diagrama de bloques para modelar

sistemas dinámicos y desarrollar algoritmos, y ejecutarlos de forma independiente en su Raspberry Pi. El paquete de soporte extiende Simulink con bloques para configurar su Raspberry Pi, enviar y recibir paquetes UDP y leer y escribir datos de sensores. Esto incluye escribir datos en el servicio gratuito de agregación de datos ThingSpeak para aplicaciones de Internet de las cosas. Después de crear su modelo de Simulink, puede simularlo, ajustar los parámetros del algoritmo hasta que lo haga correctamente y descargar el algoritmo completo para su ejecución independiente en el dispositivo. Con el bloque de funciones MATLAB, puede incorporar código MATLAB en su modelo de Simulink[30].

Con Simulink para la programación de Raspberry Pi, puede:

- Desarrollar y simular nuestros algoritmos en Simulink y utilice la generación automática de código para ejecutarlos en el dispositivo
- Incorporar procesamiento de señales, diseño de control , lógica de estado y otras rutinas avanzadas de matemáticas e ingeniería en sus proyectos de programación de Raspberry Pi
- Sintonizar y optimizar los parámetros de forma interactiva en Simulink mientras nuestro algoritmo se ejecuta en la Raspberry Pi[30].

2.7.5 Matlab Coder

MATLAB Coder genera código C y C ++ a partir de MATLAB código para una variedad de plataformas de hardware, de los sistemas de escritorio a hardware incorporado. Es compatible con la mayor parte del lenguaje MATLAB y una amplia gama de cajas de herramientas. Se puede integrar el código generado de nuestros proyectos como código fuente, bibliotecas estáticas o bibliotecas dinámicas. El código generado es legible y portátil. Se puede combinar con partes clave del código y bibliotecas C y C ++

existentes. También el poder empaquetar el código generado como una función MEX para usar en MATLAB.

Cuando se usa con Embedded Coder , MATLAB Coder ofrece personalizaciones de código, optimizaciones objetivo específicos, trazabilidad código, y software-in-the-loop (SIL) y verificación procesador-in-the-loop (PIL)[31].

2.7.6 Matlab App Designer

App Designer permite crear aplicaciones profesionales sin tener que ser un desarrollador de software profesional. Arrastrar y soltar componentes visuales para diseñar el diseño de una interfaz gráfica de usuario (GUI) y utilizar el editor integrado para programar rápidamente su comportamiento[32].

2.8 Python

Python es un lenguaje de programación interpretado, interactivo y orientado a objetos. Incorpora módulos, excepciones, tipificación dinámica, tipos de datos dinámicos de muy alto nivel y clases. Admite múltiples paradigmas de programación más allá de la programación orientada a objetos, como la programación de procedimientos y funcional. Python combina una potencia notable con una sintaxis muy clara. Tiene interfaces para muchas llamadas del sistema y bibliotecas, así como para varios sistemas de ventanas, y es extensible en C o C ++. También se puede utilizar como lenguaje de extensión para aplicaciones que necesitan una interfaz programable. Finalmente, Python es portátil: se ejecuta en muchas variantes de Unix, incluidos Linux y macOS, y en Windows[33].

2.9 Proteus

Proteus Design Suite combina la facilidad de uso con un poderoso conjunto de funciones para permitir el diseño, prueba y disposición rápidos de placas de circuito impreso profesionales[34].

Software de diseño de PCB

El software de diseño de PCB Proteus combina los módulos Schematic Capture y PCB Layout para proporcionar un conjunto de herramientas asequible, potente y fácil de usar para el diseño profesional de PCB.

El diseño de PCB Proteus es una elección profesional que presenta un diseño basado en restricciones, un potente enrutador automático y soporte dedicado para señales de alta velocidad.

- Sistema de reglas de diseño flexible que incluye soporte para salas de reglas de diseño.
- Enrutador automático basado en formas integrado incluido de serie.
- Sintonización de red de señal, modo de enrutamiento de par diferencial y corrección automática de sesgo[35].

Capítulo 3. DISEÑO Y MODELADO 3D

3.1 Diseño del chasis para el prototipo robótico

Al iniciar con las ideas para diseñar el prototipo robótico, se planteó elegir un sistema de desplazamiento que fuese eficiente y robusto para poder navegar en interiores con un ambiente controlado y pudiese soportar el peso de la base giratoria para la recolección de datos, entre las configuraciones para el sistema de locomoción figuraban la Ackermann, triciclo, síncrona, Skid Steer y diferencial como se explica en el apartado 2.2, con esto en mente se optó el utilizar una configuración Skid Steer la cual es muy parecida a la configuración diferencial, pero al utilizar cuatro ruedas de tracción del tipo fijas se produce deslizamiento transversal, cosa que no ocurre con la diferencial.

Con la idea clara de lo que se necesitaba para el chasis del prototipo, se procedió hacer el análisis de materiales con los que se contaba y así adaptar el diseño al presupuesto.

3.1.1 Piezas que componen el chasis del prototipo robótico

La mayor parte de materiales con los que se contaba eran partes de kits robóticos de la compañía VEX robotics Inc., por lo que se optó el utilizar esas piezas y formar un diseño en el Software CAD SolidWorks , para ello la empresa Vex Robotics ofrece archivos STEP para formato educativo en su página oficial, por lo que facilitaron algunas piezas hasta cierto punto.

Las piezas que se utilizarían para la primera versión del chasis serían :

Rueda de tracción

Las ruedas de tracción están diseñadas para un agarre óptimo en superficies blandas como las baldosas de espuma utilizadas en la competición de Robótica VEX.

- Peso de las ruedas de 2.75": 0.110 libras (50 gramos)[36]

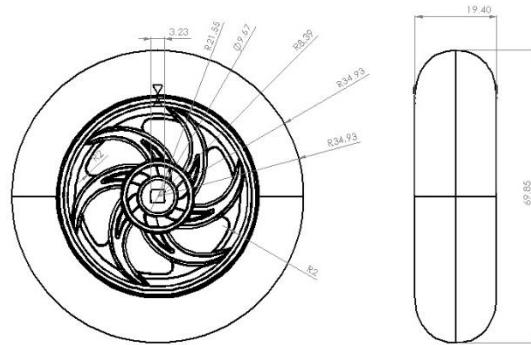


figura 34 Dimensiones del Subensamble de la rueda de tracción



figura 35 Subensamble de la rueda de tracción

Carriles

Los rieles de chasis son parte de los kits de chasis VEX más grandes. Este carril tiene agujeros en incrementos de 0.500 pulgadas y otra fila en medio, Esta parte es un ángulo versátil con un lado estrecho y un lado más alto.

Peso de los carriles:

- Carril del chasis 2 x 1 x 25: 0.187 libras
- Carril del chasis 2 x 1 x 35: 0.258 libras

Material: Acero laminado en frío según ASTM A-1008 CS Tipo B (resistencia a la tracción de aproximadamente 40 ks), 0.046 pulgadas de grueso, cinc plateado[37].

Para el diseño del chasis se utilizaron seis piezas del carril 2 x 1 x 25 agujeros.

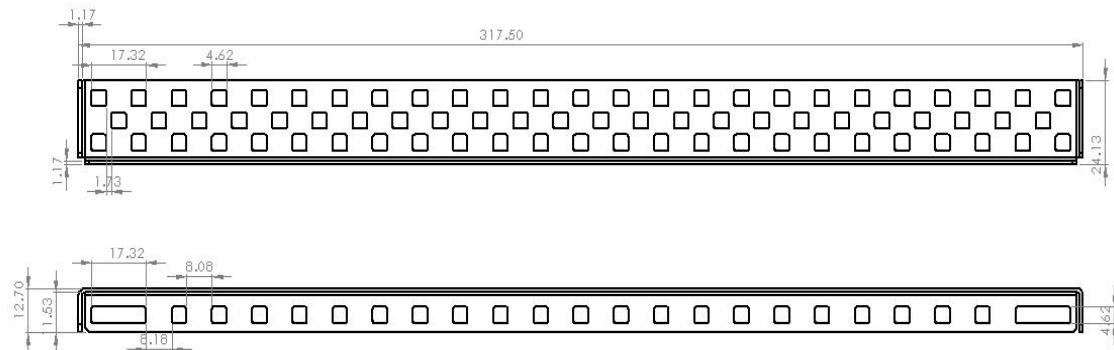


figura 36 Dimensiones del CAD carril 2 x 1 x 25 agujeros.

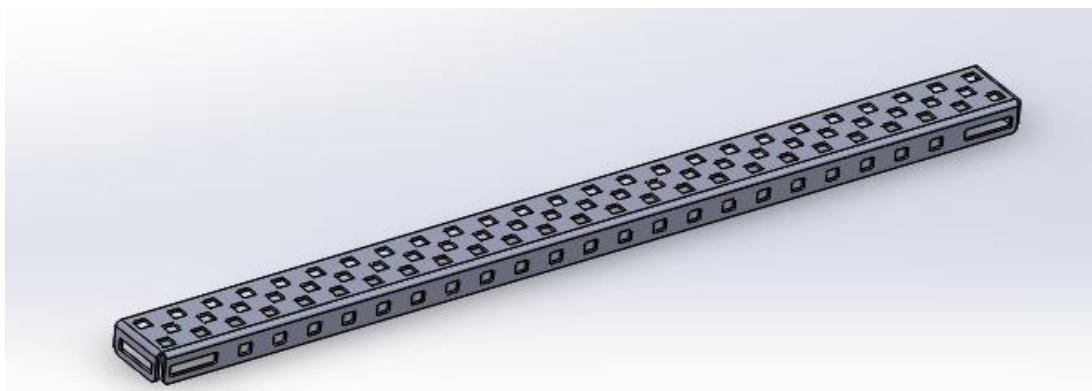


figura 37 Diseño CAD carril 2 x 1 x 25 agujeros.

Canales (8-tamaños)

El canal de VEX tiene agujeros en incrementos de 0.500 pulgadas. Esta excelente resistencia y a la flexión de este miembro estructural es perfecta para construir robots robustos.

- Múltiples tamaños disponibles, en dos tipos de materiales diferentes.
- Hecho del acero laminado o aluminio 5052-H32
- Cada canal se segmenta en trozos cortantes de 2.5 pulgadas

Tipos de materiales:

Acero: Acero laminado en frío según ASTM A-1008 CS Tipo B (resistencia a la tracción de aproximadamente 40 ks).

Aluminio: Aluminio, 5052-H32.

Dimensiones de los tamaños:

Acero Canal:

- Canal de acero 1 x 2 x 1 x 35
- Canal de acero 1 x 5 x 1 x 25
- Canal de acero 1 x 5 x 1 x 35

Aluminio Canal:

- Canal de aluminio 1 x 2 x 1 x 25
- Canal de aluminio 1 x 2 x 1 x 35
- Canal de aluminio 1 x 3 x 1 x 35
- Canal de aluminio 1 x 5 x 1 x 25
- Canal de aluminio 1 x 5 x 1 x 35

Tamaño de los hoyos cuadrados 0.182 pulgadas (estándar VEX).

Para chasis se utilizaron 2 canales de aluminio de las dimensiones 1 x 2 x 1 x 25 agujeros[38].

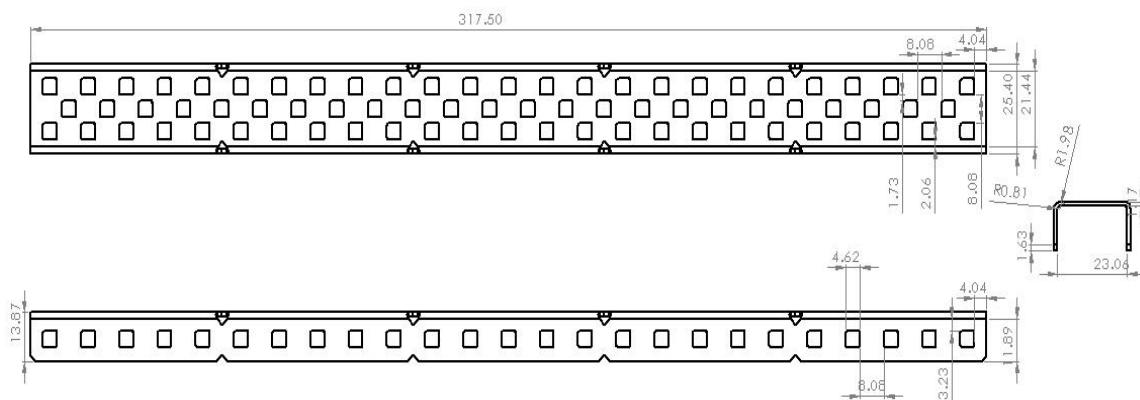


figura 38 Dimensiones del diseño CAD canal 1 x 2 x 1 x 25 agujeros.

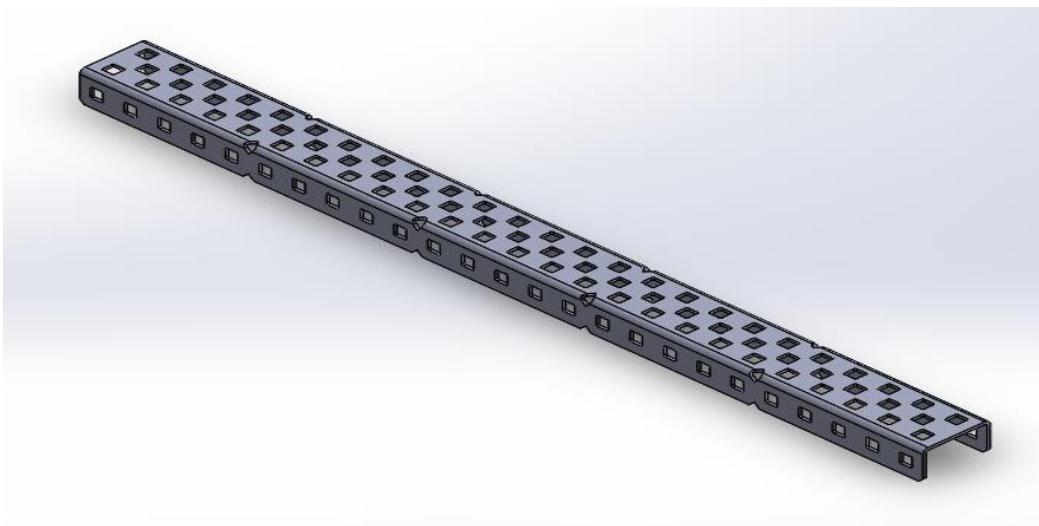


figura 39 Diseño CAD canal 1 x 2 x 1 x 25 agujeros.

Motores

Inicialmente se contempló utilizar motores VEX EDR 269, el motor de 2 cables 269 reemplaza al motor de 3 cables como motor VEX estándar . Todos los engranajes internos están hechos de una aleación de acero, lo que significa que los embragues y engranajes de repuesto ya no son necesarios. El motor de 2 hilos se puede conectar directamente a los ESC los cuales se conectaron a la shield que administra la etapa de potencia y las señales mandadas por la placa controladora Arduino mega 2560.

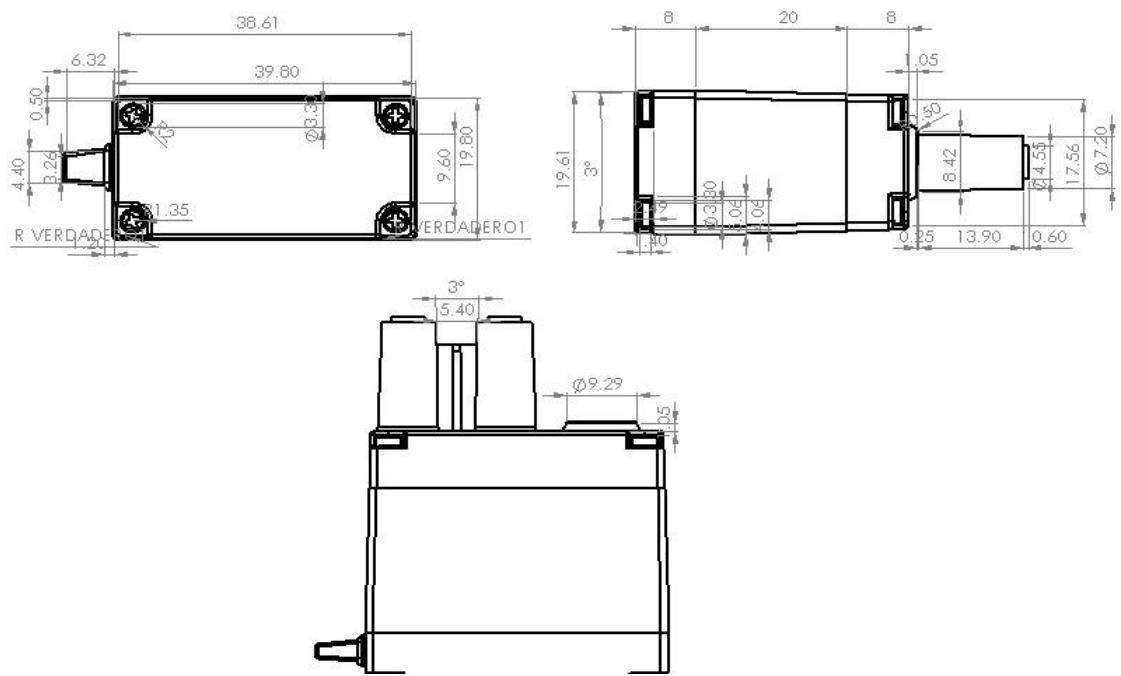


figura 40 Dimensiones del subensamble del motor VEX EDR 269

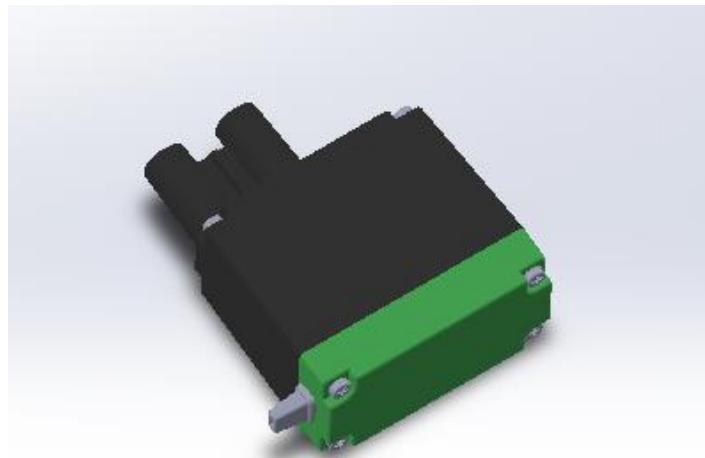


figura 41 Subensamble del motor VEX EDR 269

Ejes VEX

Para transmitir la fuerza del motor hacia las ruedas se utilizaron ejes impulsores cuadrados que tienen esquinas redondeadas que permiten girar fácilmente en un orificio redondo, mientras que en el agujero cuadrado se bloquean.

- Longitudes: 2 ", 3" y 12 "
- Funciona con motores, ruedas, engranajes y cojinetes[39]

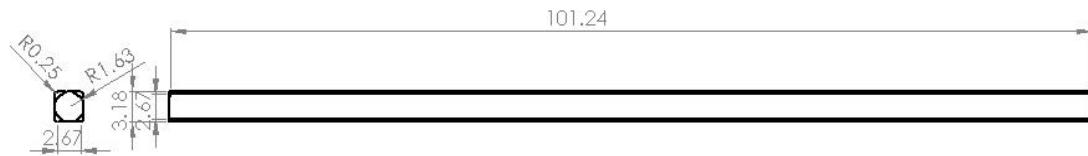


figura 42 Dimensiones del Diseño CAD del Eje de 3"

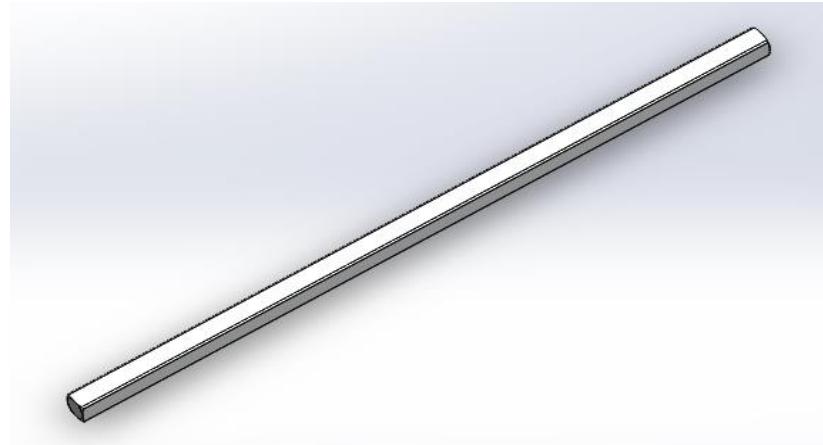


figura 43 Diseño CAD del Eje de 3"

Collarines y collares de sujeción

También se necesitaron de collarines que fijaran las piezas y de esta forma evitar el desplazamiento de los ejes y que con el movimiento estos de salieran de su sitio. Los collarines se bloquean en los ejes de transmisión, evitando que las ruedas y los engranajes se deslicen a lo largo del eje. Los collarines se ajustan alrededor de los ejes de transmisión y se pueden fijar en su lugar con un tornillo de fijación .

De igual forma para que el eje trasmita la fuerza del motor a la rueda, se necesitó una chumacera que pudiera utilizar el eje, por lo que VEX ofrece algo parecido que son los collares de sujeción para ejes.

Las ventajas de usar esto son :

- La baja fricción permite que los ejes giren suavemente
- Montaje con tornillos y tuercas o remaches adjuntos.[39]

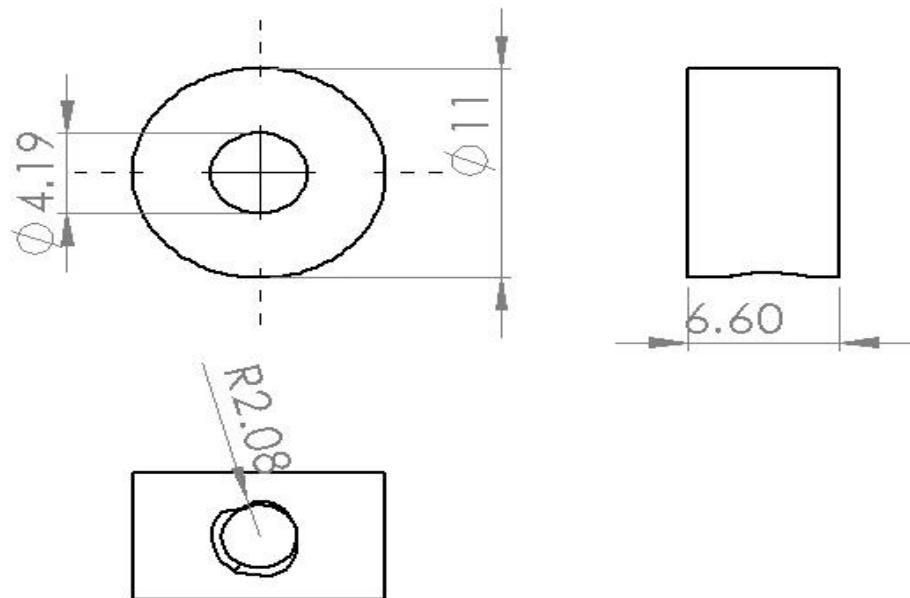


figura 44 Dimensiones del diseño CAD de los collarines

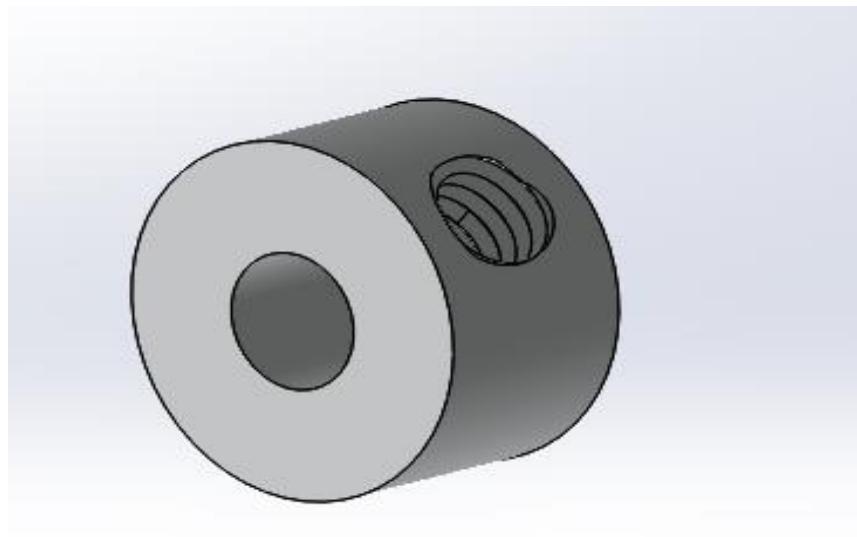


figura 45 Dimensiones del diseño CAD de los collarines

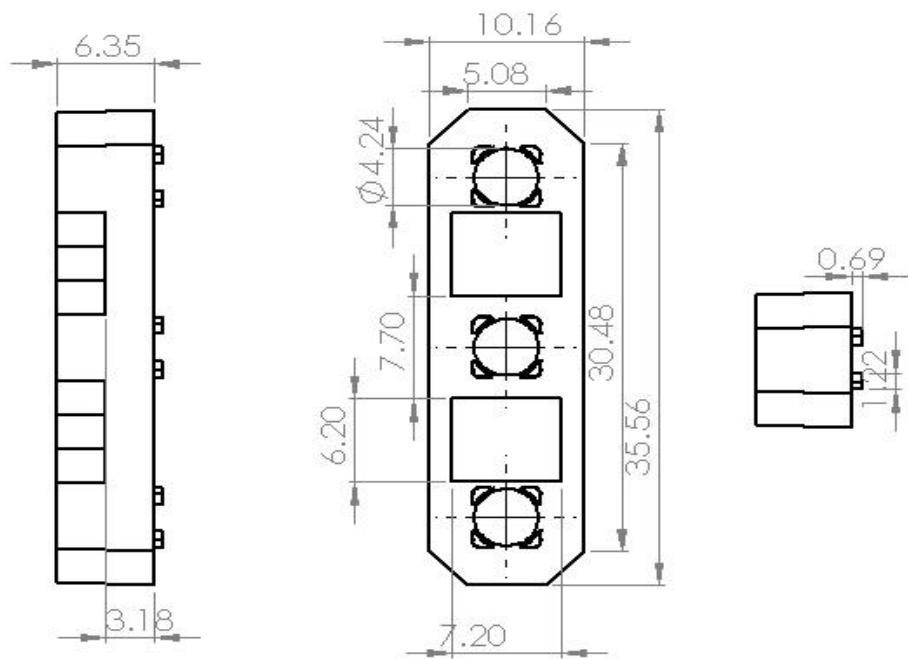


figura 46 Dimensiones del diseño CAD de las chumaceras para ejes

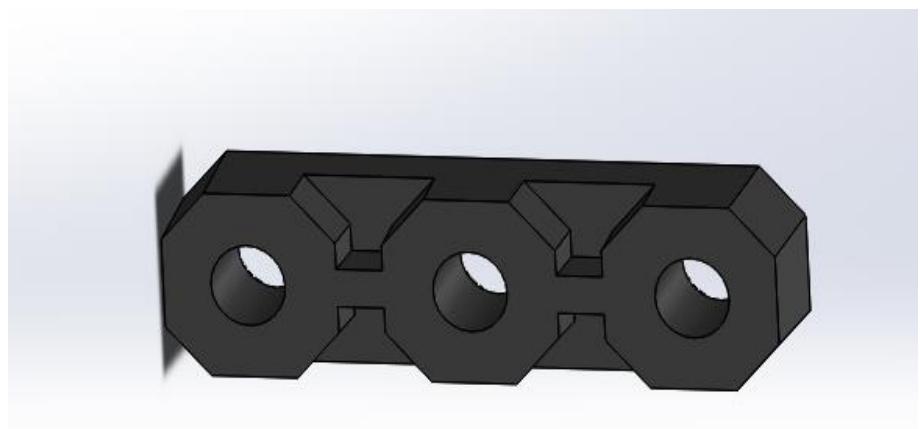


figura 47 Diseño CAD de las chumaceras para ejes

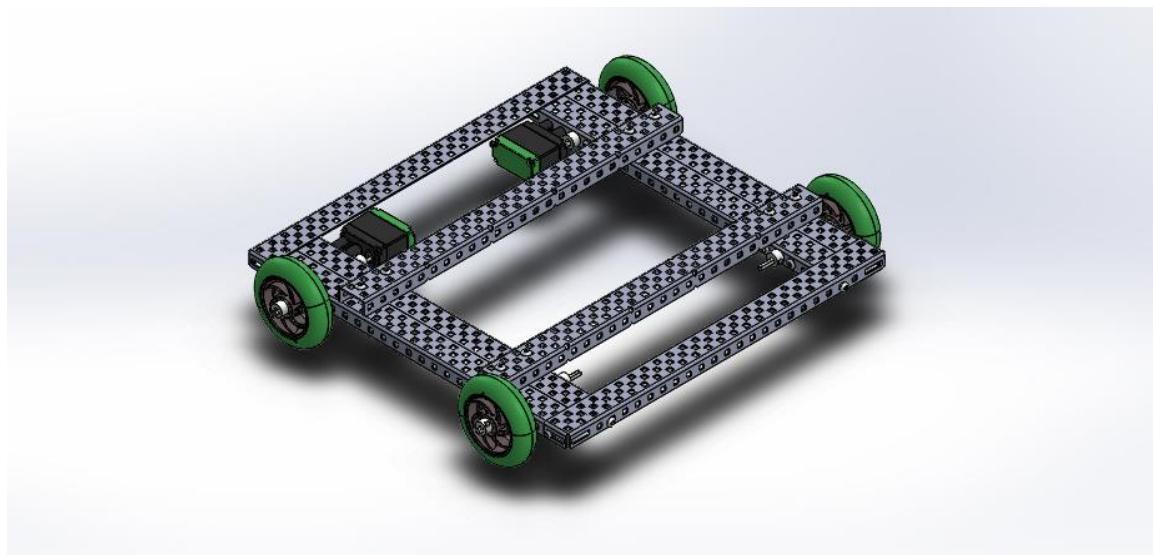


figura 48 Ensamble de los componentes para el chasis

Acoplamiento de la superficie giratoria

Se diseño una estructura superior que fungiera como base para la etapa giratoria, por lo que se utilizó un motor paso a paso como los descritos en la sección 2.4.2, el cual moviera la estructura donde se colocarían los sensores ultrasónicos y la electrónica de control.

El motor paso a paso (nema17) que se utilizó nos proporciona por paso ángulos de 1.8 grados en total por revolución son de 200 pasos , con un consumo de corriente de 1.2 A, sin embargo, por las dimensiones el torque que proporciona no era suficiente para hacer rotar la estructura para la captura de datos, así que se optó por diseñar una caja de engranajes planetarios que aumentaran la fuerza de giro que haría el motor.

Primero se diseñó la base donde se acoplaría el motor a pasos(nema17) y de igual manera se diseñó el motor a pasos(nema 17) con las medidas originales.

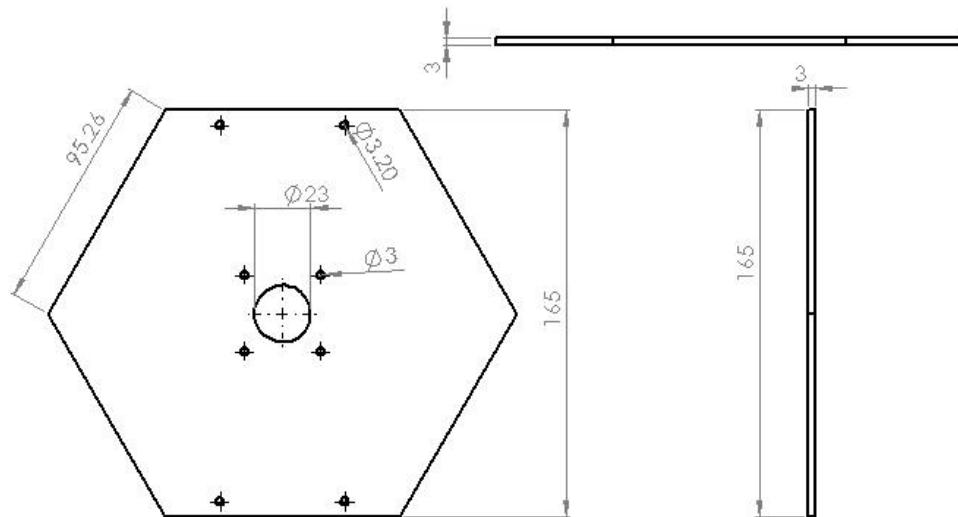


figura 49 dimensiones del diseño CAD de la base del motor paso a paso (nema17).

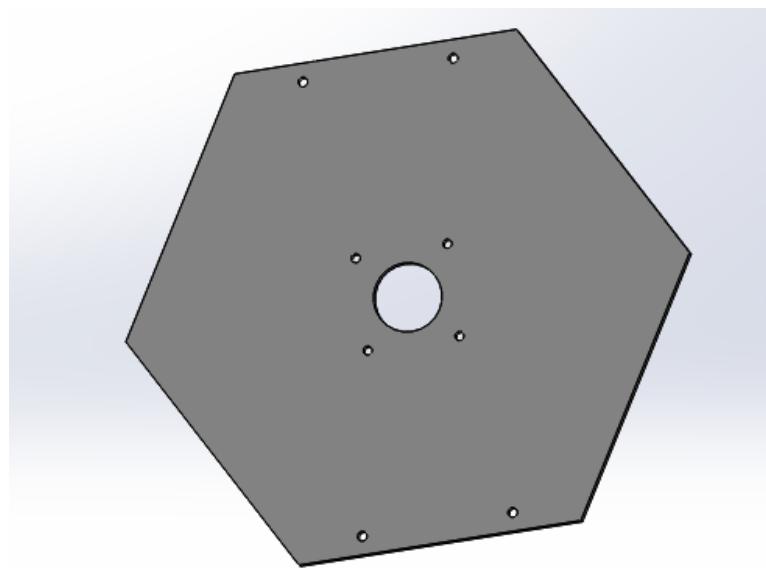


figura 50 Diseño CAD de la base del motor paso a paso (nema17).

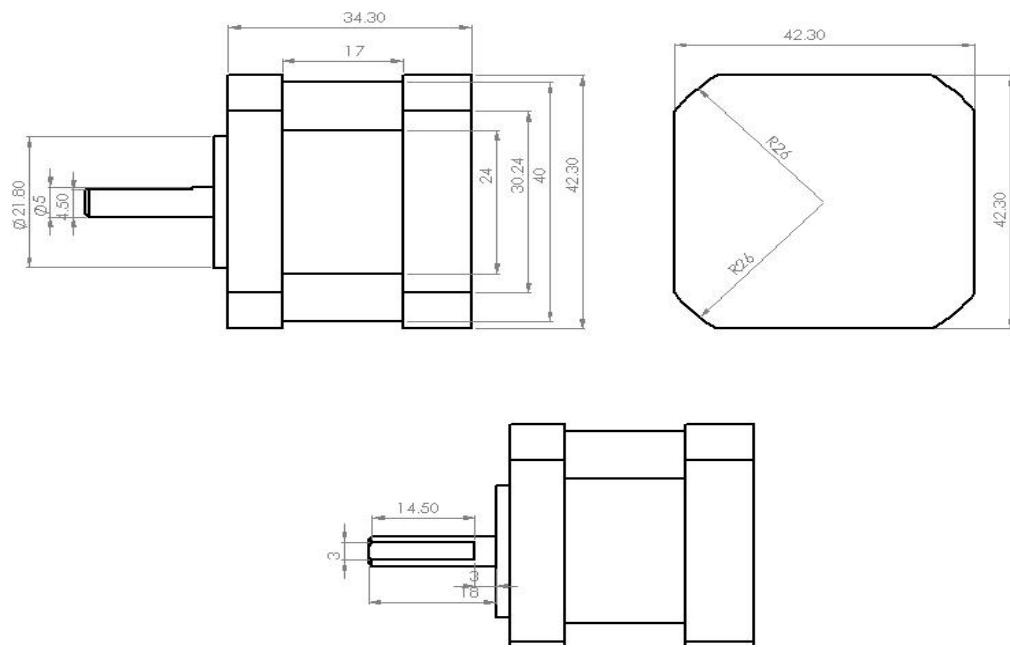


figura 51 Dimensiones del diseño CAD del Nema 17

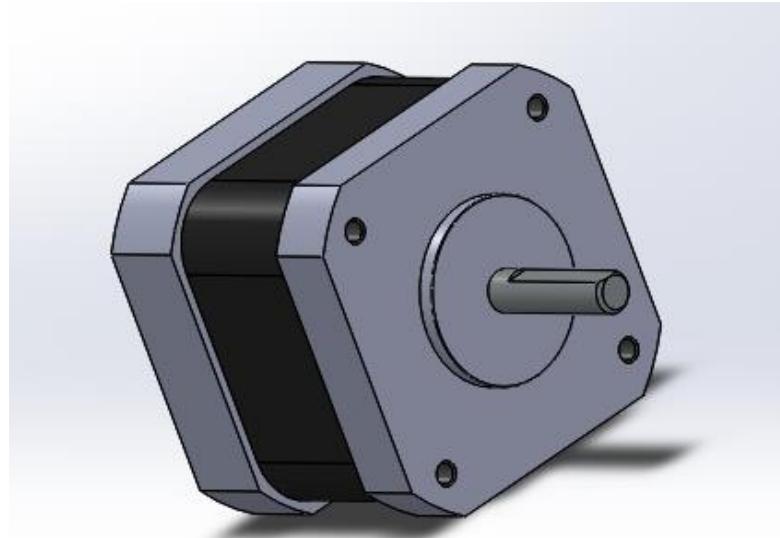


figura 52 Diseño CAD del Nema 17

Para diseñar la caja de engranajes planetarios se utilizó un toolbox de SolidWorks, la relación final de la caja de engranes fue de 38,4: 1 , con un cojinete de giro integral.

Primero se diseñó una base que se acoplaría entre el Nema 17 y la base de la caja de engranajes, para ello se diseñó una base de forma circular de 60mm de diámetro con cuatro agujeros de 3.3mm donde se atornillaría el nema17, y a las orillas se dejaron ocho agujeros más los que se utilizarían para asegurar esta base a la base de nuestra caja de engranajes, de los cuales cuatro fueron de 4mm y cuatro más de 3mm.

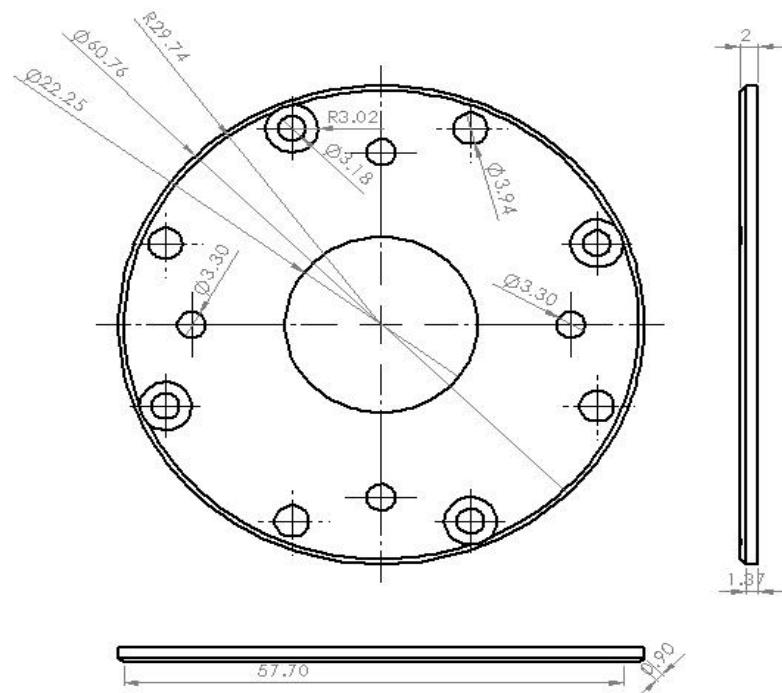


figura 53 Dimensiones del diseño CAD de base NEMA-COJINETE

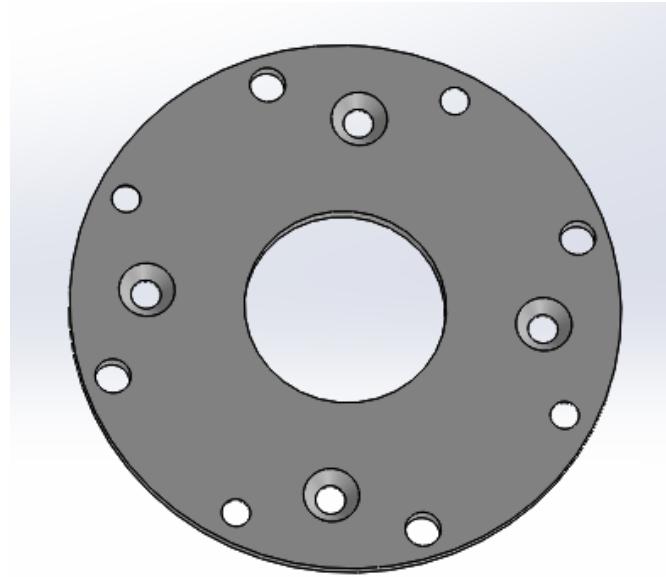


figura 54 Diseño CAD de base NEMA-COJINETE

Posterior a ello utilizando el toolbox de SolidWorks se diseñó la base de la caja de engranajes con 30 dientes helicoidales, para mayor velocidad se utilizó un Angulo de hélice de 30° grados. El diámetro exterior de la pieza es de 60mm, el diámetro interior de la pieza es de 26mm, el espesor del diente es de 1.36mm y con una anchura de hueco de 1.16mm.

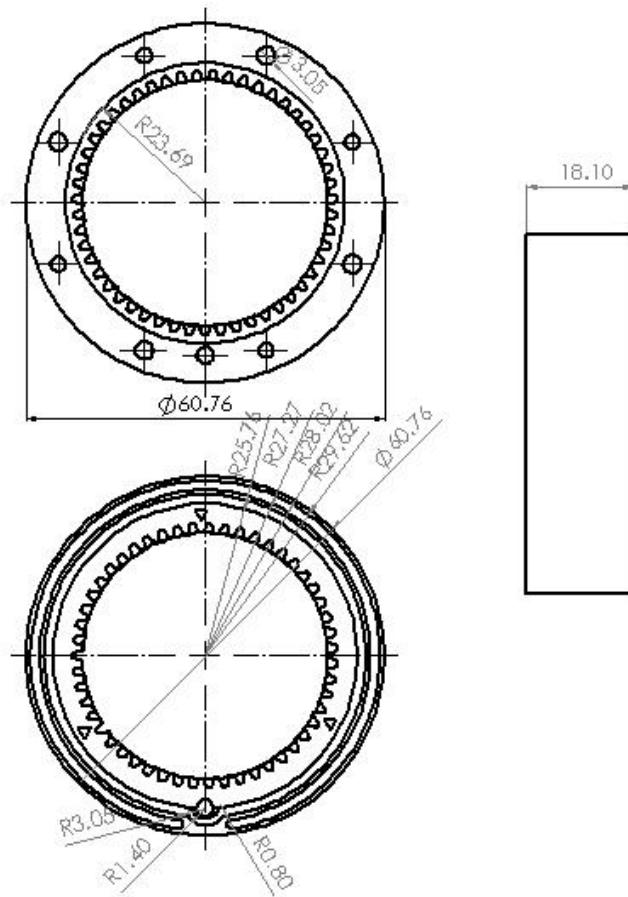


figura 55 Dimensiones del diseño CAD del cojinete de giro integral inferior



figura 56 Diseño CAD del cojinete de giro integral inferior

Posteriormente se diseñó el engrane central que se acoplaría a la nema17, este engrane cuenta con 15 dientes helicoidales y para mayor velocidad se utilizó un Angulo de hélice de 30° grados, la circunferencia superior es de 14 mm de diámetro, la circunferencia interior es de 11mm, la circunferencia primitiva es de 12 mm, el espesor del diente es de 1.36mm y con una anchura de hueco de 1.16mm. el diámetro interno donde se colocara el eje del motor tiene un diámetro de 5.26 mm en el mismo tiene un recorte el cual servirá para ser acoplada con la guía que tiene por defecto el eje del motor nema17.

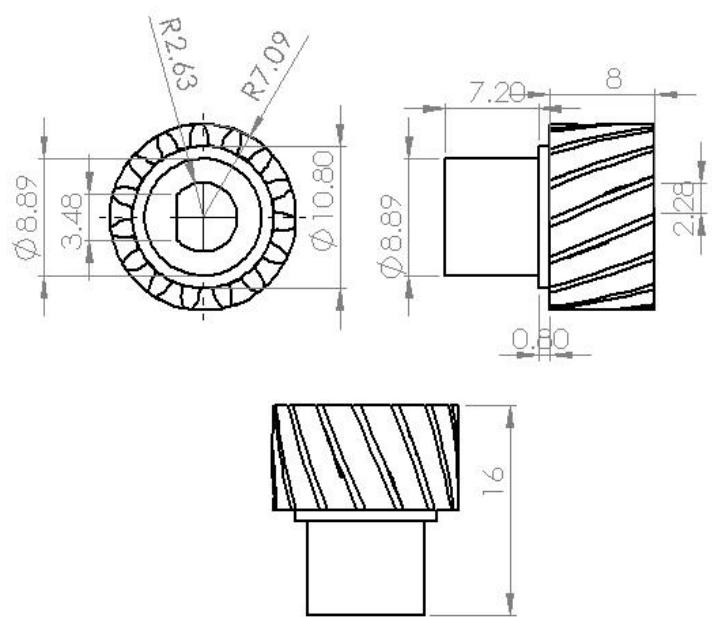


figura 57 Dimensiones del diseño CAD del engrane central del Nema 17

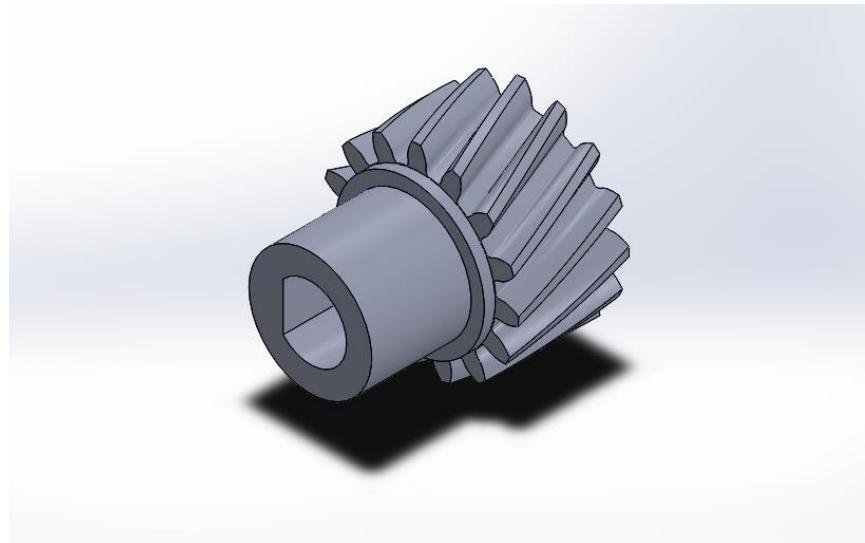


figura 58 Diseño CAD del engrane central del Nema 17

Se diseñó tres engranes dobles que formarían parte del planetario, los tres engranes tienen las mismas medidas, el engrane inferior contiene 18 dientes, la circunferencia superior es de 16.70 mm de diámetro la circunferencia interior es de 13.40 mm, la circunferencia primitiva es de 14.50 mm el espesor del diente es de 1.36 mm y con una anchura de hueco de 1.16mm.

Para el engrane superior se invierte la dirección de giro, este engrane cuenta con 15 dientes helicoidales, la circunferencia superior es de 14 mm de diámetro la circunferencia interior es de 11mm, la circunferencia primitiva es de 12 mm el espesor del diente es de 1.36mm y con una anchura de hueco de 1.16mm al igual que el primero.

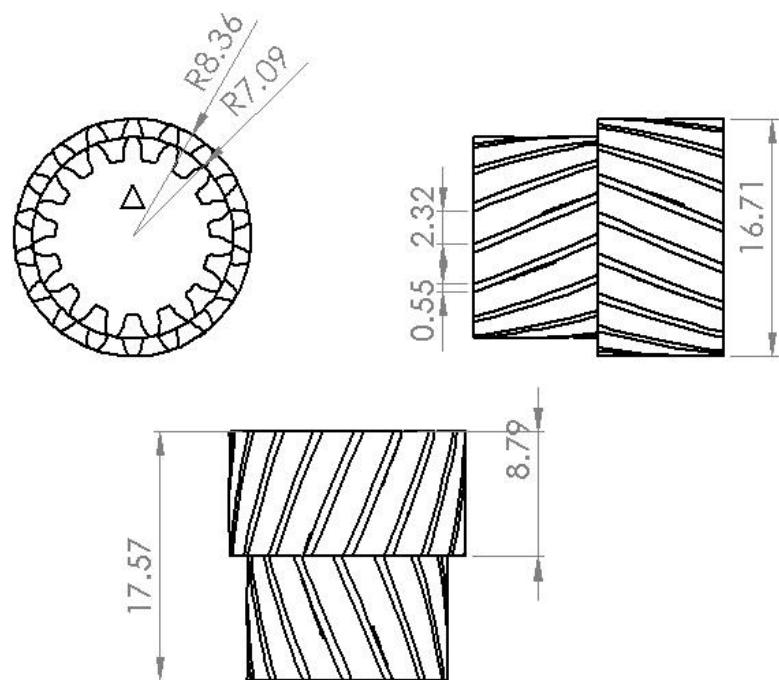


figura 59 Dimensiones del diseño CAD del engrane planetario

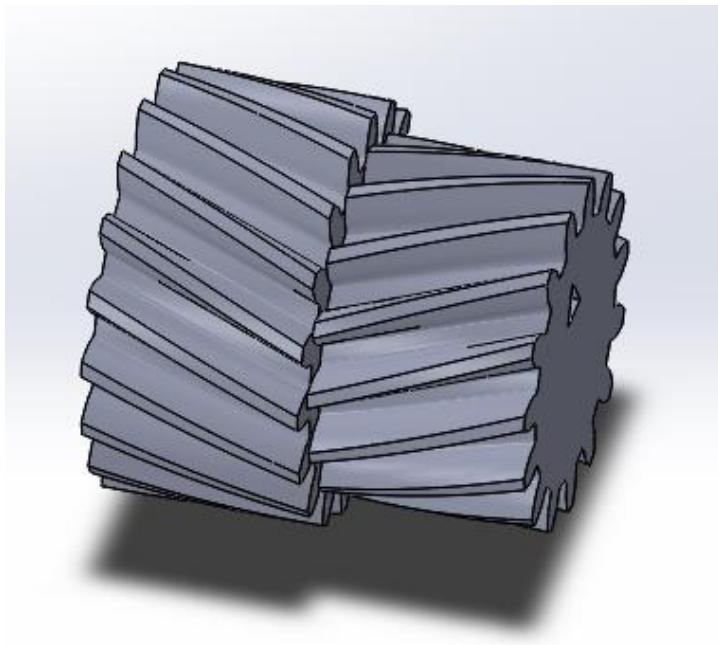


figura 60 Diseño CAD del engrane planetario

Finalmente se implementó un cojinete de giro integral , al igual se dejaron orificios con los cuales podríamos atornillar la base donde se encontrarían nuestro sensor. Cuenta con 48 dientes helicoidales con un Angulo de hélice de 30° grados. El diámetro exterior de la pieza es de 49mm el diámetro interior de la pieza es de 39mm, el espesor del diente es de 1.36mm y con una anchura de hueco de 1.16mm.

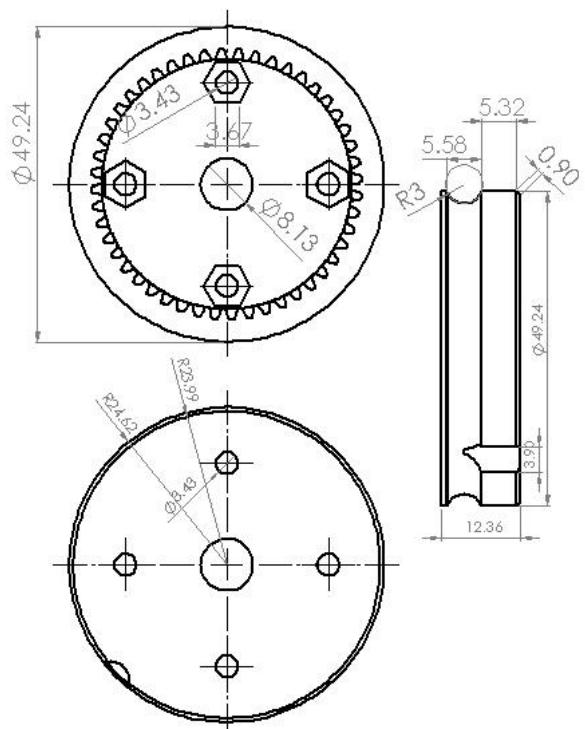


figura 61 Dimensiones del diseño CAD del cojinete de giro integral

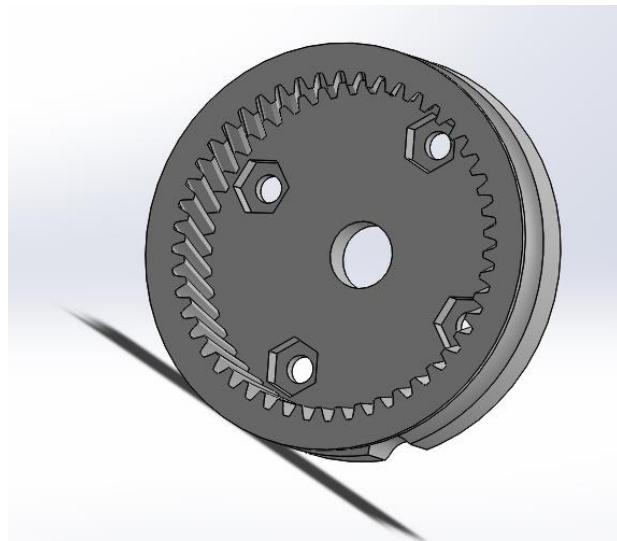


figura 62 Diseño CAD del cojinete de giro integral

3.1.2 Diseño completo del chasis

Al tener todas las piezas en formato CAD se hizo el ensamble, de esa forma se tendría la estructura del chasis para el autómata.

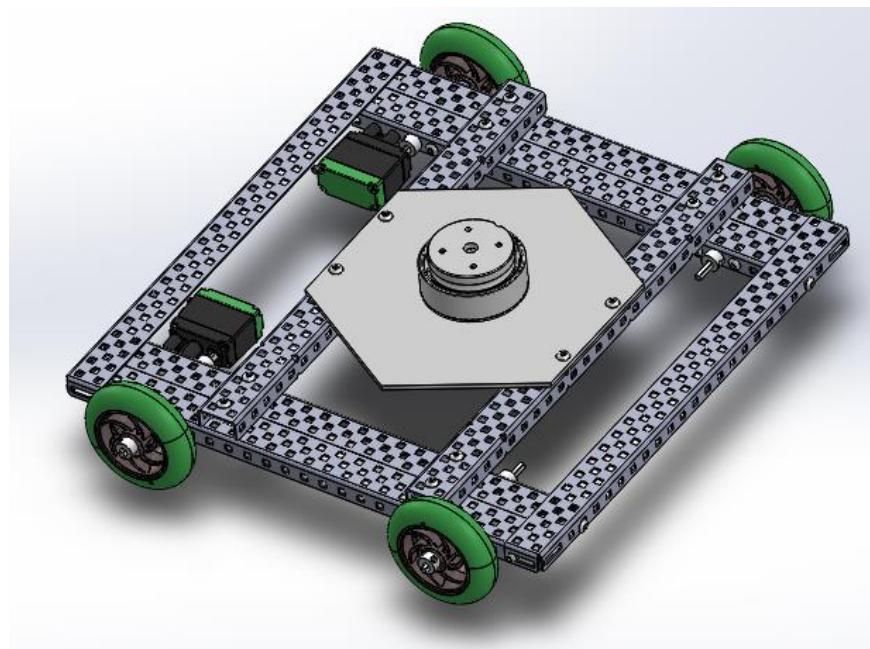


figura 63 Ensamble del chasis del autómata.

3.2 Diseño de la base giratoria para la captura de datos

Para el diseño de la plataforma donde se apoyaría toda la electrónica se optó por una base circular de ABS Plastic , en la orilla de tal superficie se colocaron los sensores ultrasónicos SFR08 a una distancia o separación de 22.5 grados.

3.2.1 Piezas que componen la plataforma giratoria

Base negra de la parte superior

Se diseño la base circular de 31 cm de diámetro y un grosor de 6mm, en esta base se hicieron distintas perforaciones las cuales servirán para colocar las bases donde irán colocados los sensores Ultrasónicos, ruedas locas metálicas, tornillos para la base donde irán los microcontroladores y la pantalla LCD.

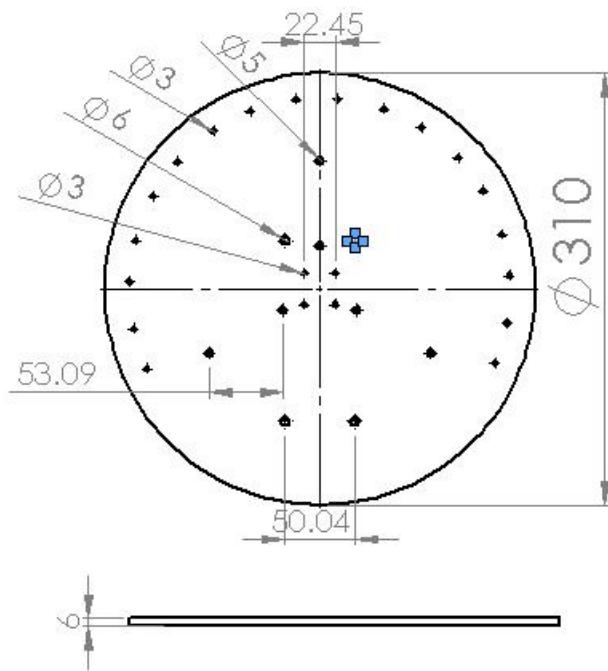


figura 64 Dimensiones del diseño CAD de la base negra de la parte superior.

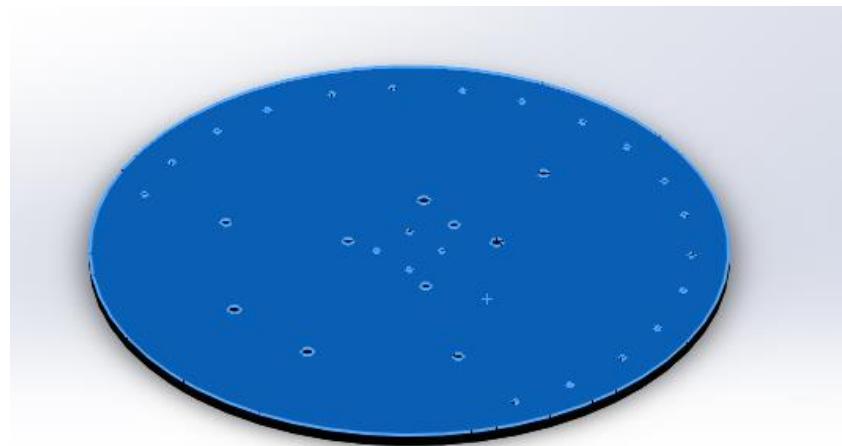


figura 65 Diseño CAD de la base negra de la parte superior.

Base para los Sensores Ultrasónicos SFR08

Para la base donde se colocarían los sensores ultrasónicos, se diseñó una en forma de L , en la parte donde se colocaría el sensor tendría las dimensiones de 50 mm de ancho y 48 mm de largo, la parte que serviría para atornillarla a la base negra de la parte superior tendría 50 mm de ancho y 11mm de largo.

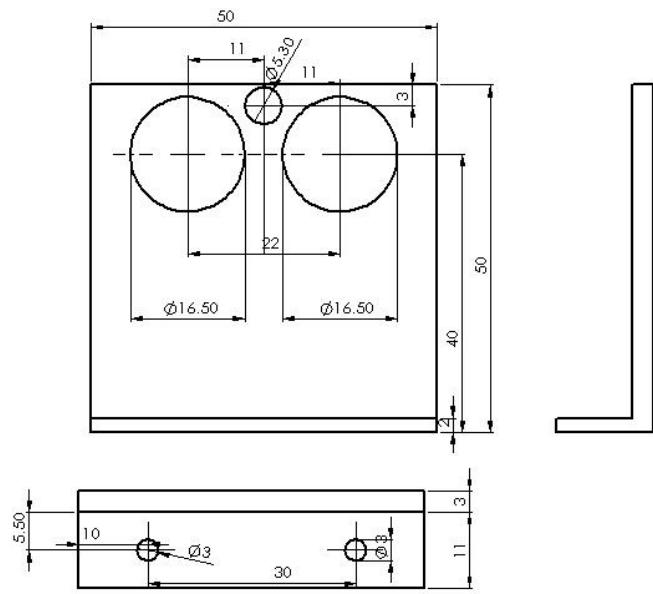


figura 66 Dimensiones del diseño CAD de la base para los sensores ultrasónicos SFR08.

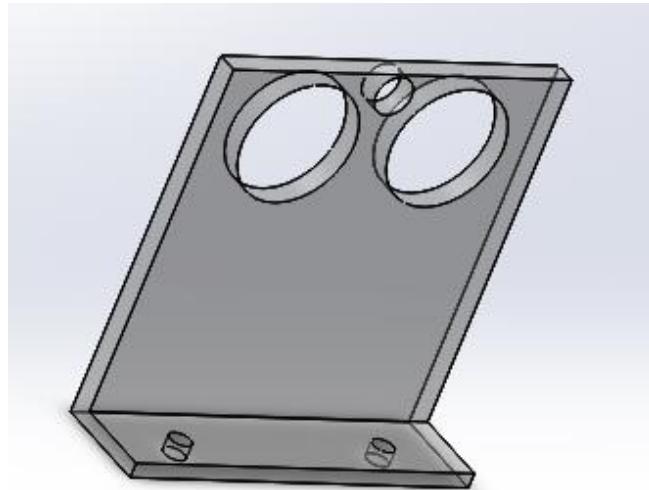


figura 67 Diseño CAD de la base para los sensores ultrasónicos SFR08.

Sensor ultrasónico SFR08

Para el diseño del sensor ultrasónico se utilizaron las medidas mencionadas en el capítulo dos y se implementaron en el sensor y en la base que sería utilizada para alojar al sensor ultrasónico.

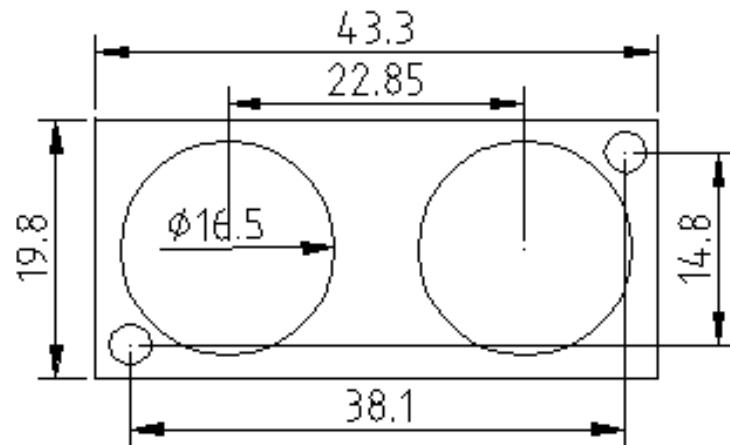


figura 68 Medidas provistas del sensor ultrasónico SFR08.

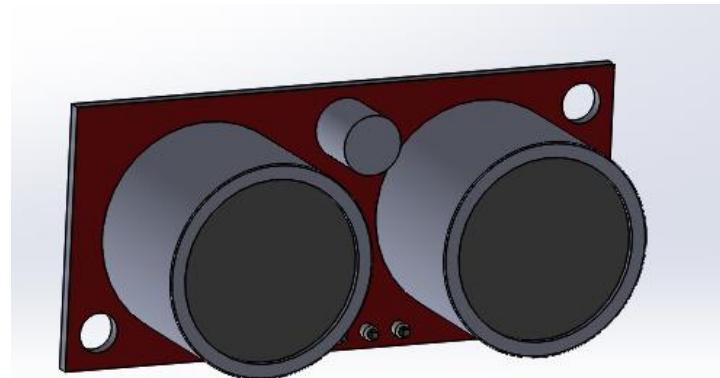


figura 69 Diseño CAD del sensor ultrasónico SFR08.

Acrílico inferior.

Para la parte donde se colocarían los microcontroladores y electrónica de potencia, se eligió el utilizar una superficie de acrílico la cual sería de 150 mm de largo y de 150 mm de ancho, con sus respectivos agujeros por los cuales pasarían tornillos de 3mm y de 6mm, los cuales asegurarían el acrílico a la base negra y los más pequeños asegurarían la placa microcontrolador Arduino mega 2560.

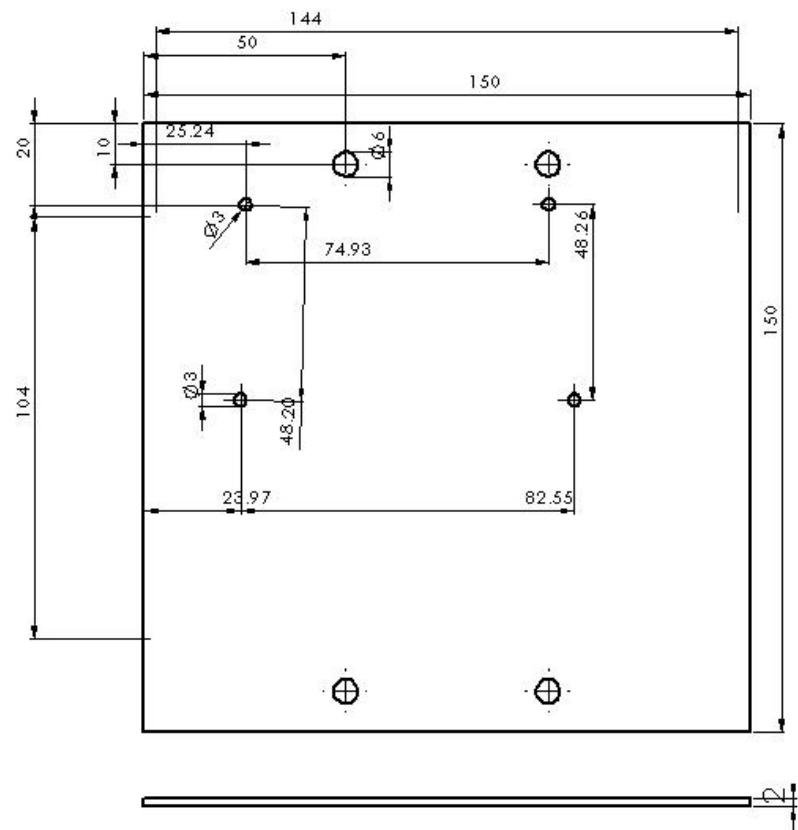


figura 70 Dimensiones del diseño CAD del acrílico inferior

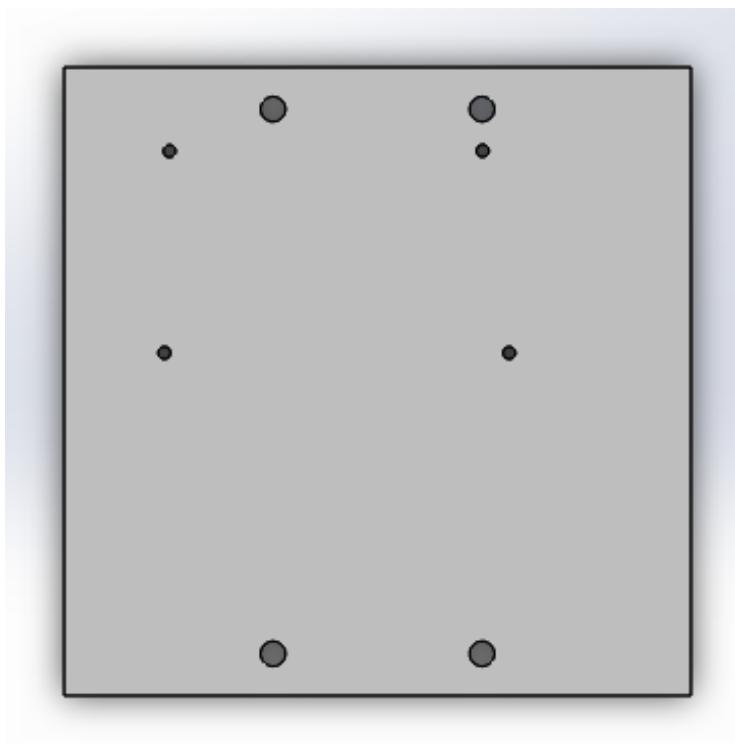
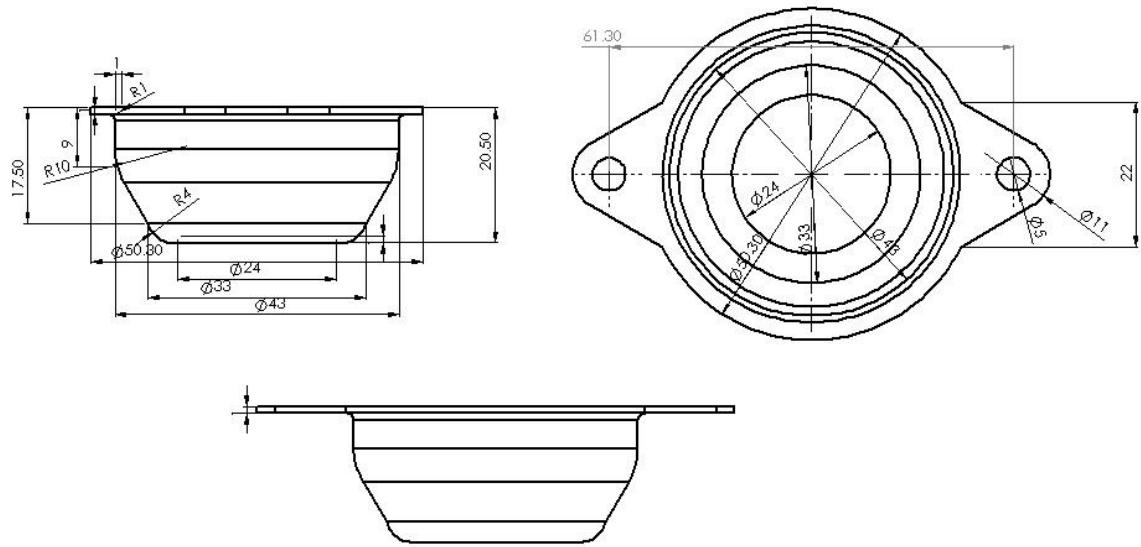


figura 71 Diseño CAD del acrílico inferior

Rueda loca metálica

Para el diseño de la rueda loca se diseñó una esfera que sería el balín o rueda loca con un radio de 12.25 mm , posteriormente se procedió a diseñar la base donde este balín la cual tendría un diámetro de 38mm y una altura de 20.50mm.



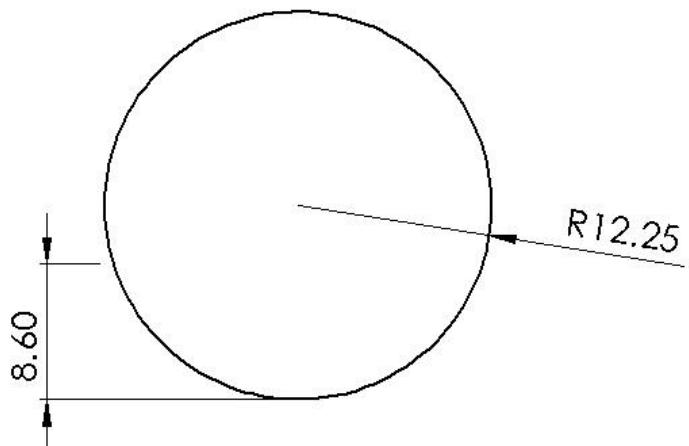


figura 74 Dimensiones del diseño CAD del balín o rueda loca



figura 75 Diseño CAD del balín o rueda loca

Tornillos y tuercas

Para el diseño de los tornillos se utilizaron distintas medidas 3mm ,5mm, y 6mm de grosor y alturas variables dependiendo el uso del mismo, de igual forma se diseñaron tuercas para las medidas de los distintos grosores de los tornillos.

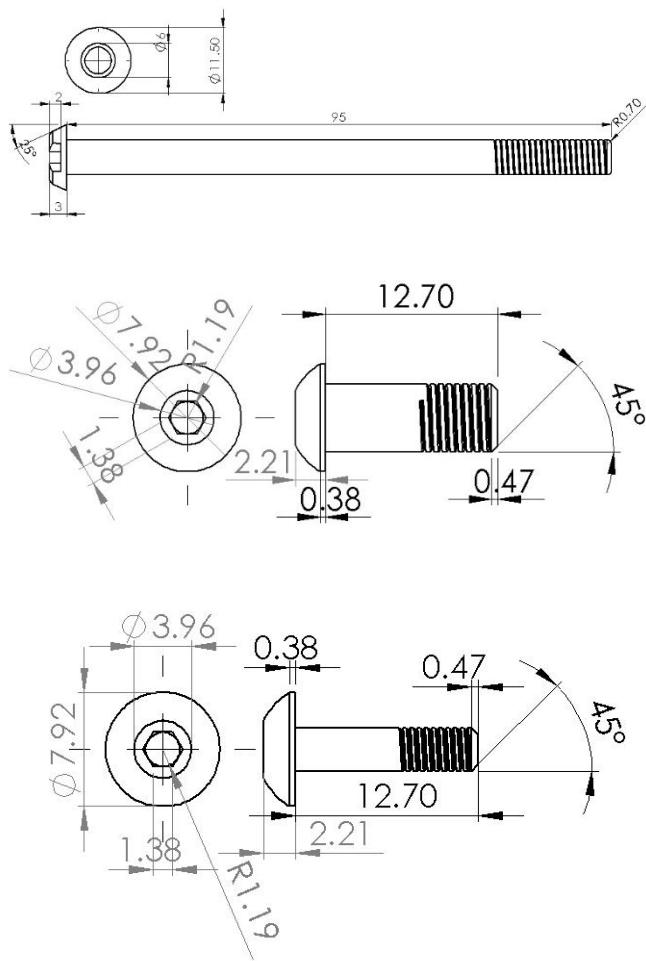


figura 76 Dimensiones de los diseños CAD de la tornillería

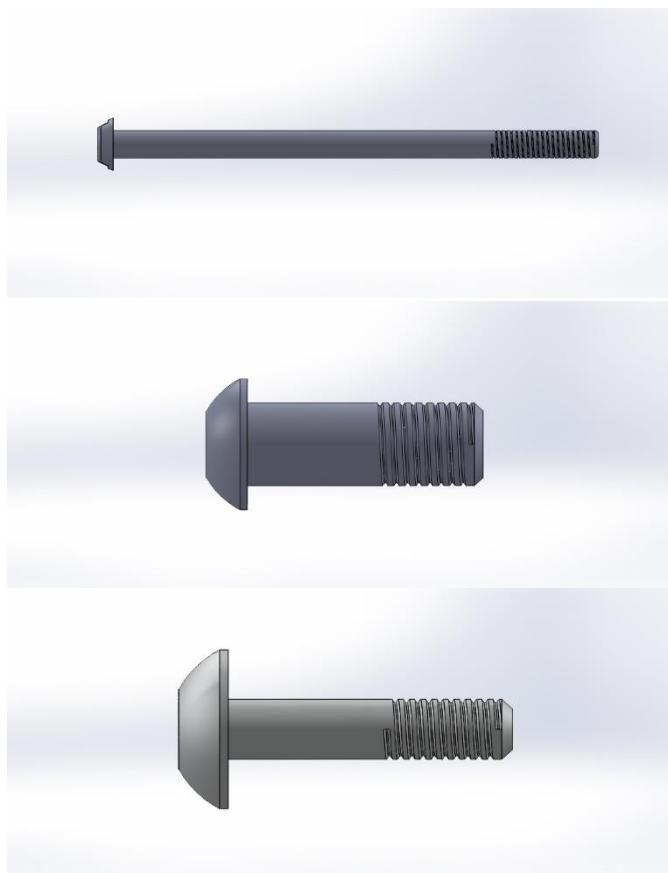


figura 77 Diseños CAD de la tornillería

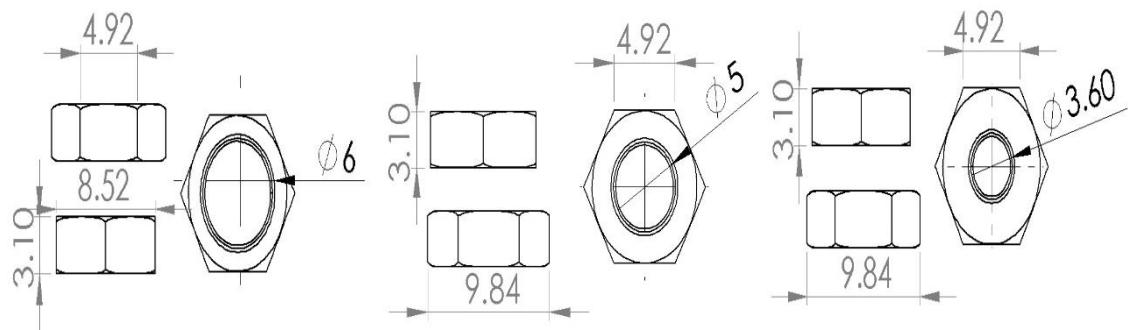


figura 78 Dimensiones de los diseños CAD de la tornillería

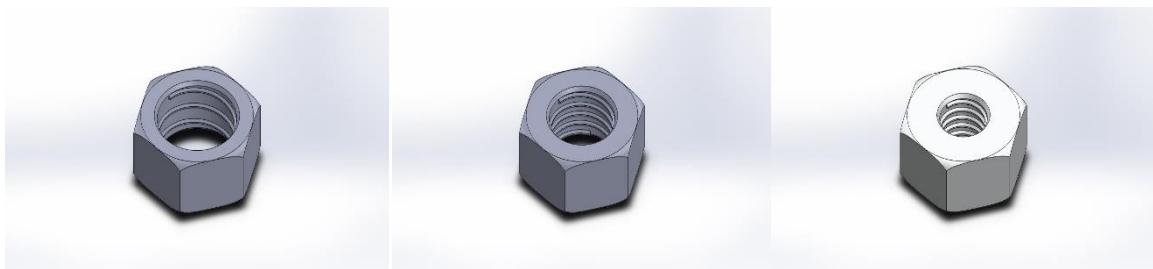


figura 79 Diseños CAD de la tornillería

Pantalla LCD

Al diseñar la pantalla LCD la placa base donde se encuentra se tomó las medidas de 144 largo y 104 ancho y el display o monitor de 240*128 pixeles con las dimensiones 133mm de largo 80mm de ancho.

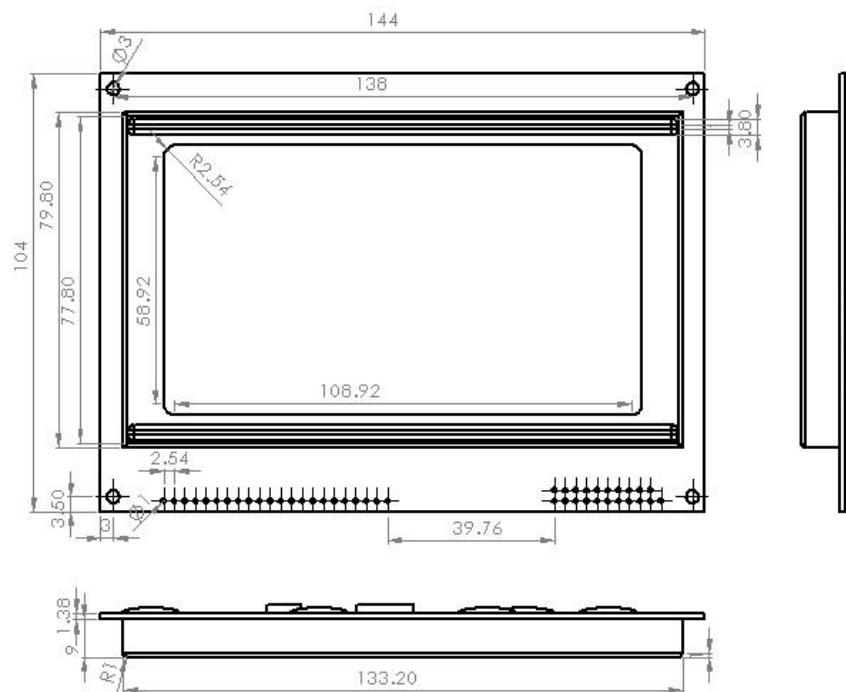


figura 80 Dimensiones del diseño CAD pantalla LCD 240*128 pixeles

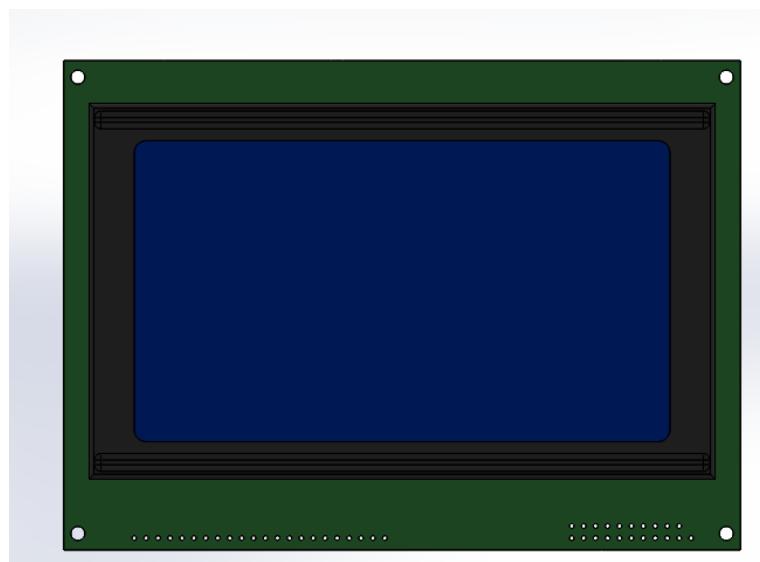


figura 81 Diseño CAD pantalla LCD 240*128 pixeles

Microcontroladores y shield

Para el diseño de las placas Arduino mega 2560, raspberry pi 3 b+ y el shield de Arduino el cual contiene una etapa de potencia, adquisición de datos y control de actuadores se utilizó el software de ISIS PROTEUS el cual ofrece diseños CAD de estas placas y de la shield que diseñamos dentro del software, por ello podemos exportar los archivos CAD de proteus a SolidWorks y así hacer el ensamblaje en conjunto.

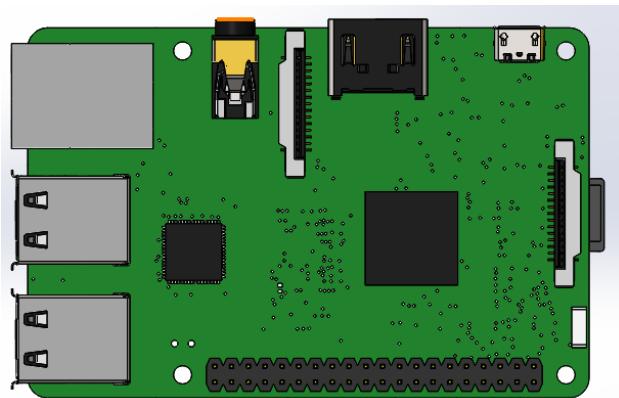


figura 82 Modelo CAD Raspberry pi 3b+



figura 83 Modelo CAD Arduino Mega 2560

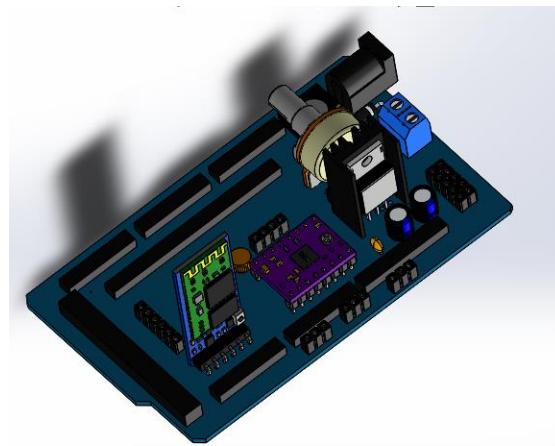


figura 84 Modelo CAD Shield Para Arduino Mega 2560

3.2.2 Diseño completo de la plataforma giratoria

Al final se hizo un ensamble con todas las piezas de la parte superior del prototipo robótico.

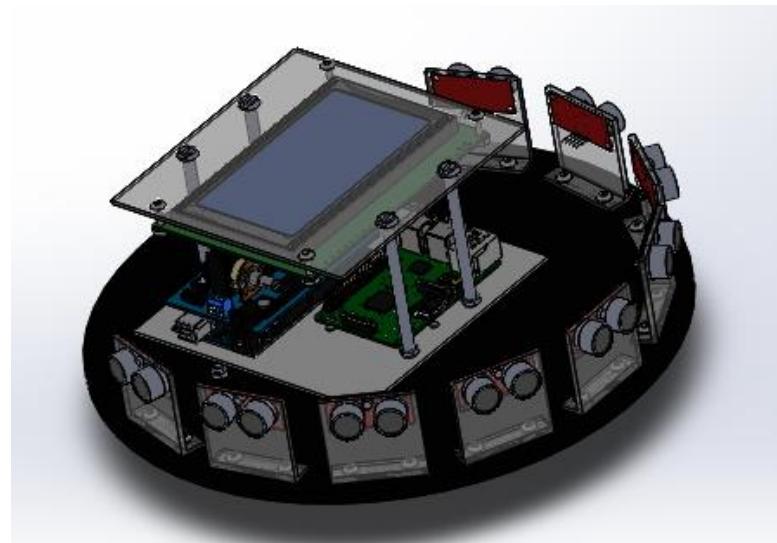


figura 85 Ensamble final de la parte superior del prototipo final (A)

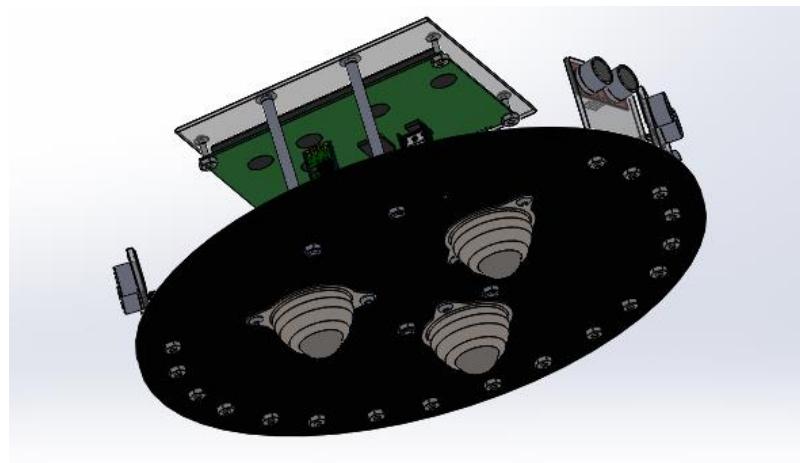


figura 86 Ensamble final de la parte superior del prototipo final (B)

3.3 Diseño completo del prototipo robótico

Al tener diseñado tanto la parte inferior como la superior del prototipo robótico, se procedió a hacer un ensamblaje final juntando ambas partes.

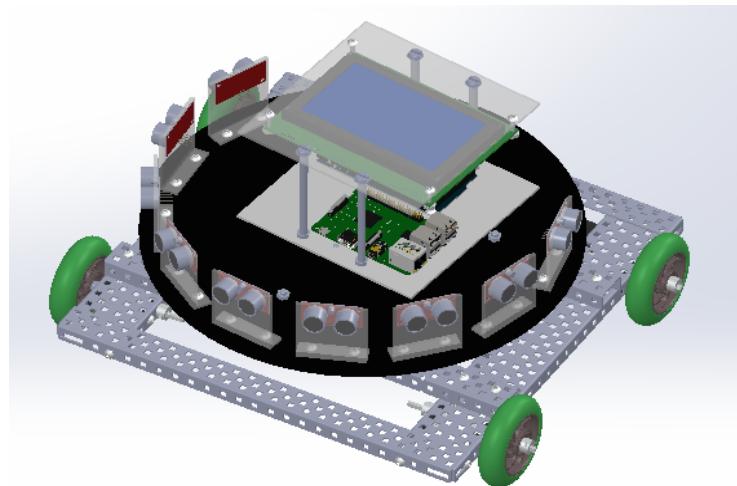


figura 87 Ensamble final del prototipo robótico (A)

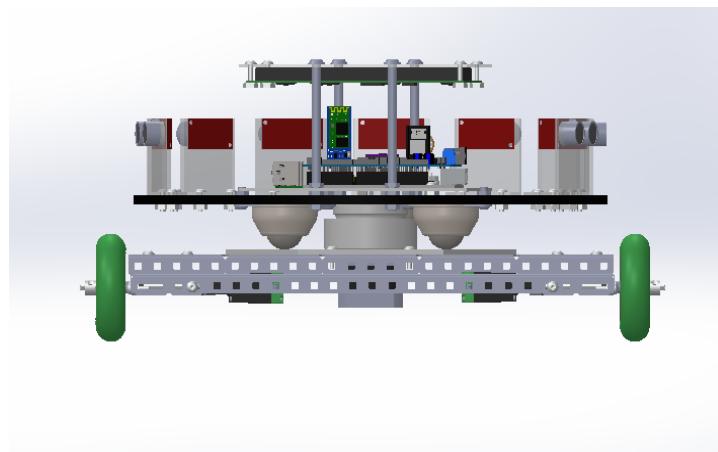


figura 88 Ensamble final del prototipo robótico (B)

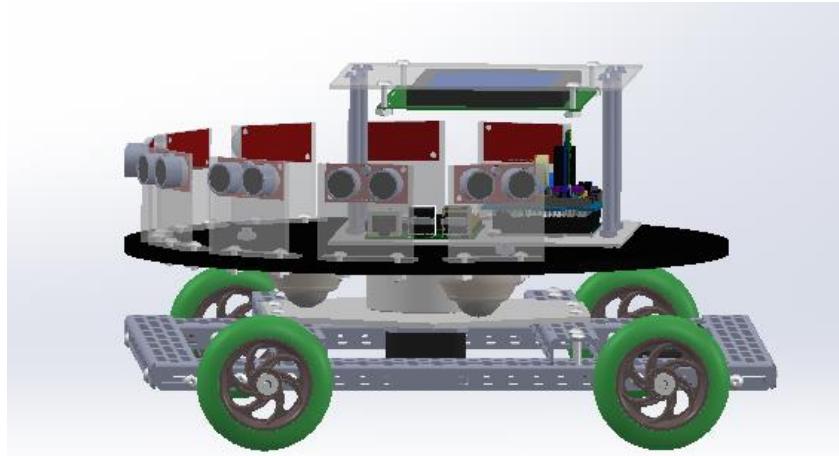


figura 89 Ensamble final del prototipo robótico (C)

3.4 mejoras al diseño inicial.

Al armar el prototipo físicamente como describiremos en capítulos posteriores, la fricción generada por las llantas de 2.5" era muy poca por lo que surgían deslices en superficies de azulejo, el toque del motor VEX EDR 296 también era muy pobre para mover la estructura completa, al hacer pruebas físicas de la base giratoria notamos un funcionamiento correcto, sin embargo, al aumentarle dicho peso a la estructura surgieron estos problemas, por lo que se optó el utilizar motores VEX EDR 393 y utilizar llantas de fricción de 4".

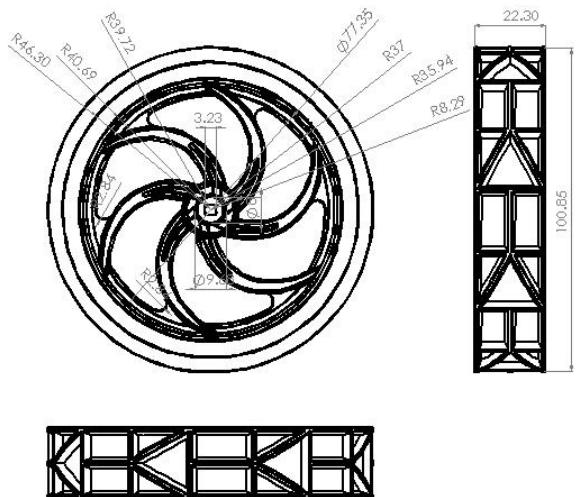


figura 90 Dimensiones del diseño CAD de la llanta de fricción de 4 pulgadas.



figura 91 Diseño CAD de la llanta de fricción de 4 pulgadas.

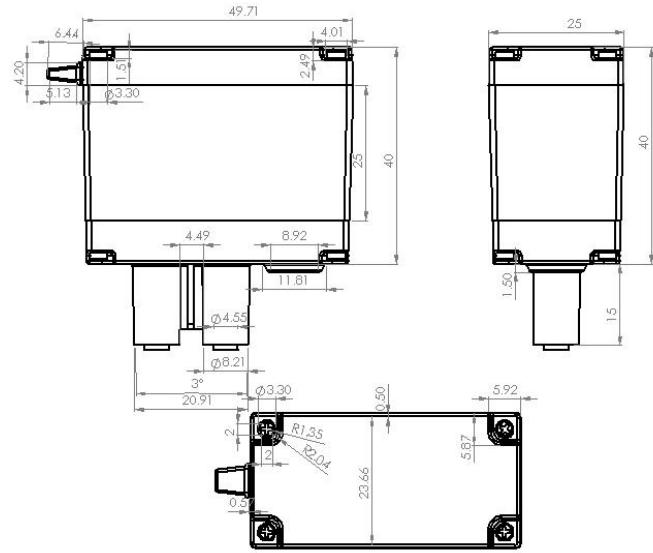


figura 92 Dimensiones del diseño CAD del motor VEX EDR 393.

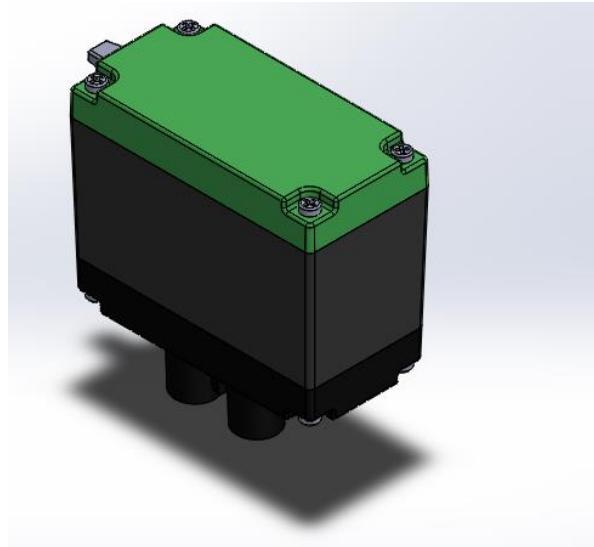


figura 93 Diseño CAD del motor VEX EDR 393.

3.5 diseño Final con mejoras implementadas al diseño inicial.

Al implementar nuevos cambios al diseño inicial se encontraron ciertos problemas, por ejemplo el tamaño de la llanta era mayor, y al utilizar 4 motores VEX EDR 393 más grandes en lugar de los dos 2 motores pequeños VEX EDR 296 como se tenía inicialmente, se modificó levemente la estructura para poder acoplar dichas piezas al diseño final.

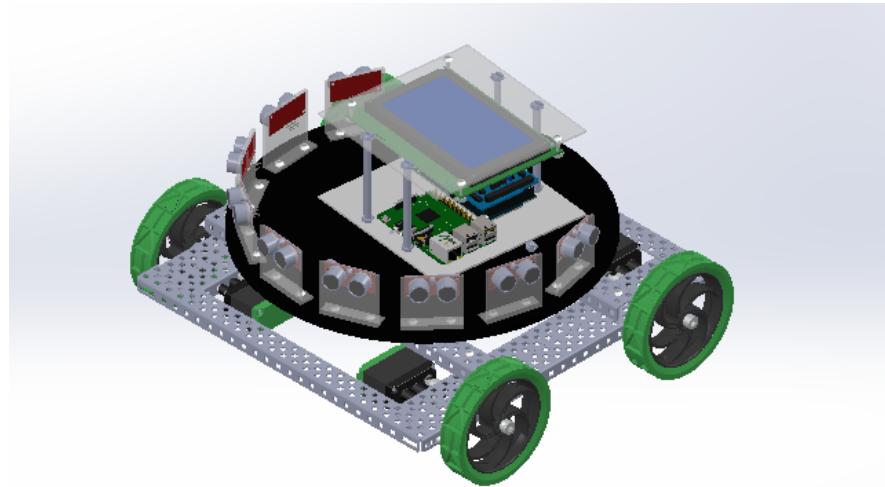


figura 94 Ensamble final del prototipo robótico con las mejoras implementadas (A)

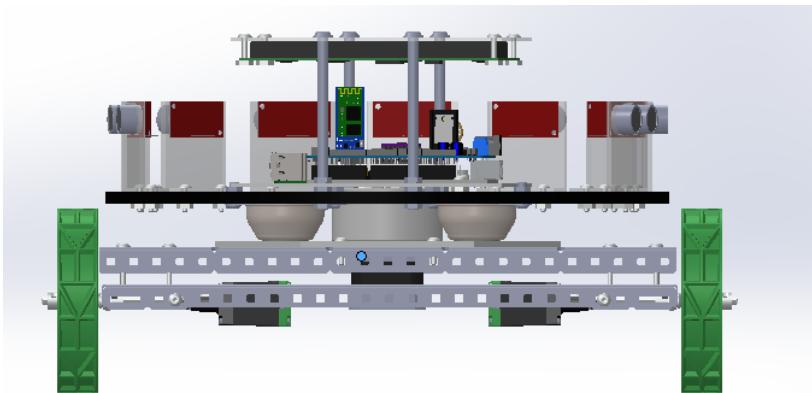


figura 95 Ensamble final del prototipo robótico con las mejoras implementadas (B)

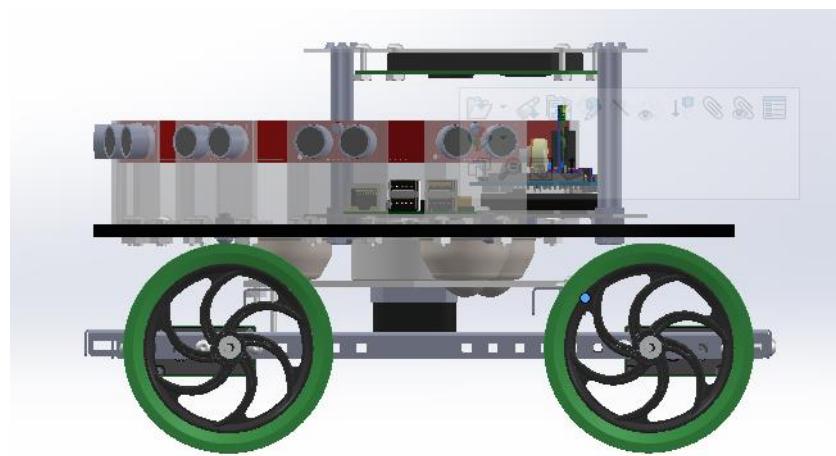


figura 96 Ensamble final del prototipo robótico con las mejoras implementadas (C)

Capítulo 4. DISEÑO DE PLACA PCB

Se ha mencionado anteriormente el uso de microcontroladores para este proyecto, en este caso el uso de la placa Arduino mega 2560 el cual contiene el microcontrolador ATmega 2560, sin embargo, para la comunicación con los sensores y actuadores no se puede conectar directamente al microcontrolador, la placa Arduino mega tiene un regulador de voltaje en el que permite mediante un Jack de alimentación una entrada de 7 volts mínimos y 12 volts máximos de entrada como voltajes recomendables, el regulador baja ese voltaje a 5 voltios y a una corriente de 1 Amperio, sin embargo, la salida de 5V que proporciona Arduino para alimentación de sensores o actuadores es de máximo 150 mA y cada sensor puede llegar a utilizar hasta 275 mA por lo que al usar 9 sensores se necesita más corriente para el correcto funcionamiento de los sensores.

De igual forma, se utiliza una pantalla LCD de 240*128 pixeles la cual contiene 20 pines para conectarlos con el microcontrolador y con ello poder usarla correctamente, a esto le sumamos el uso de dos motores VEX EDR 269 el cual consume una corriente nominal de 0.18 A y una de bloqueo o corriente máxima de hasta 2.6 A. para controlarlos se utilizó un puente H por lo que se utilizarían dos pines de la placa para controlar la dirección del giro y la velocidad de los motores, a esto añadimos el motor paso a paso NEMA17 que permitiría girar el sistema de captura de distancias mediante los sensores, para controlarlo se utilizaría un driver a4988 el cual también necesitaría un voltaje de 12v – 36v con una corriente de 1.5 A.

Con todo esto es necesario una fuente de alimentación que cumpla con las necesidades del proyecto, por lo cual se utilizó una batería vex de 7.2v con 3000mAh niMH. El motor nema funcionaría bien simplemente que al utilizar un voltaje inferior lo compensaría consumiendo mayor amperaje.

Es imperativo el diseñar una placa la cual tenga una etapa de potencia que alimente a los motores VEX EDR 269 y al motor NEMA17, de igual forma una etapa reguladora de voltaje para alimentar la placa Arduino y los sensores que trabajen a 5v. y con todo ello se

pueda conectar todos los sensores y actuadores a la placa para así tenerlo de la forma más compacta, funcional y estéticamente posible.

4.1 Diseño de la primera shield para Arduino mega

ESQUEMÁTICO

El diseño de la placa PCB se hizo en el software de diseño ISIS Proteus, la forma correcta de diseñarla es creando la parte esquemática primero y con ella proceder a hacer la parte del diseño PCB.

Inicialmente se creó los pines de la placa de Arduino mega y etiquetándolos para tener una forma visual de identificarlos y así poder hacer las conexiones correctas a la placa física.

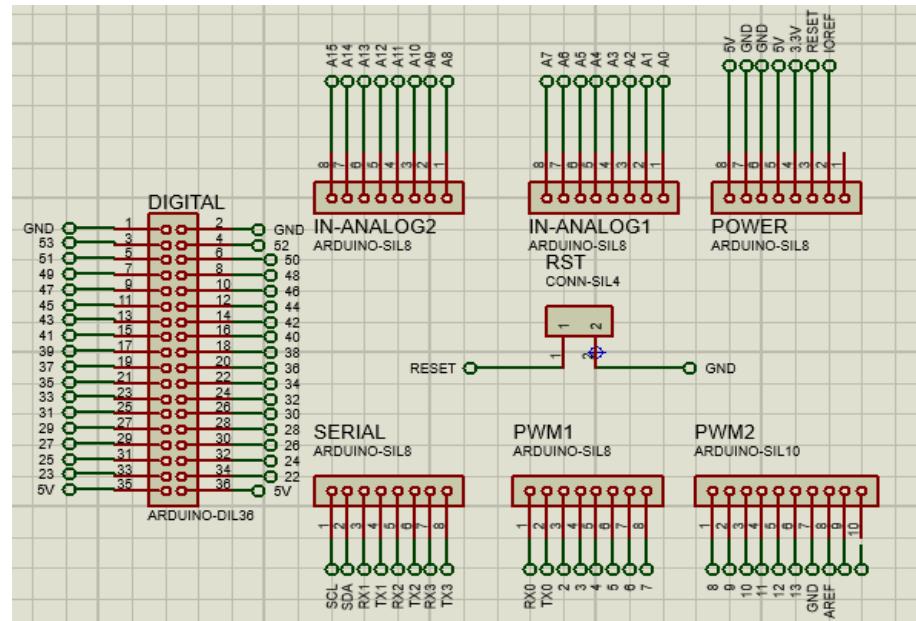


figura 97 Creación y etiquetado de los pines de la placa Arduino mega en ISIS proteus.

Teniendo lo anterior, se procedió a diseñar la etapa reguladora de voltaje para lo cual se utilizó un Jack de alimentación y una bornera de 2 terminales, de esta forma se conecta la batería a la bornera, para hacer pruebas utilizando fuentes externas como cargadores de 12V o fuentes reguladas de voltaje se conectaría al Jack de alimentación.

Usando un diodo 1n4001 se protege el circuito, dirigiendo la corriente en un solo sentido. Se utilizó el regulador de voltaje lm7809 el cual tiene tres pines, en VI entra en voltaje del Jack de alimentación en esa línea se colocó un capacitor electrolítico de 47uF para estabilizar el voltaje de entrada, GND es tierra y todas las tierra se conectan entre si, finalmente VO es el voltaje de salida regulado en esta línea también se colocó un capacitor electrolítico de 47uF para mantener estable el voltaje y la corriente de salida, la salida del regulador lm7809 pasa a la entrada del regulador lm7805 con la misma configuración de capacitores electrolíticos la salida de este último regulador ira a los circuitos alimentados con 5V , como vemos la entrada del lm7805 también va conectado a la bornera de alimentación para poder colocar la batería de 7.4V de Vex y así regularla a 5V directamente.

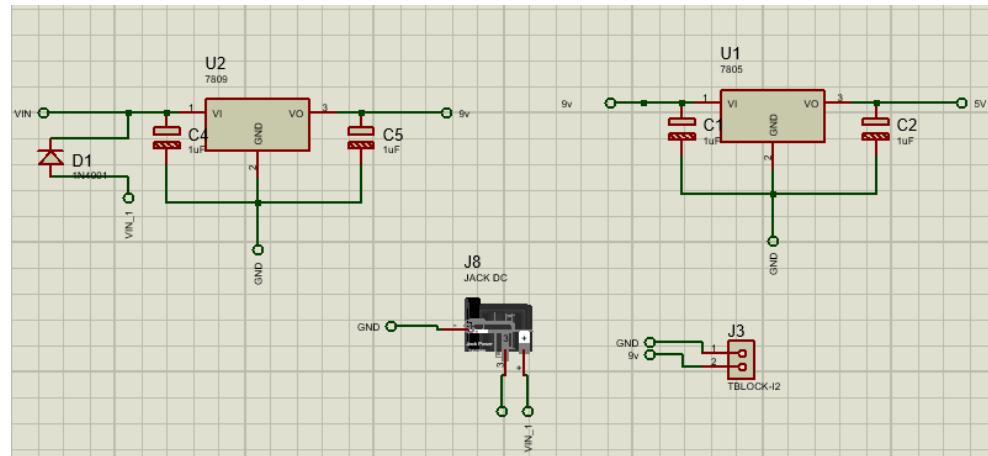


figura 98 Diseño de la etapa reguladora de voltaje de entrada.

Continuando con el diseño, se procedió a elaborar la etapa de potencia la cual controlaría el motor NEMA17 el cual haría girar la base superior, para ello el uso del driver A4988 era necesario así que se crearon los pines donde se colocaría el driver y se procedió con el etiquetado de cada pin sin antes olvidar el agregar un capacitor electrolítico en la entrada del voltaje de alimentación que iría directo al motor.

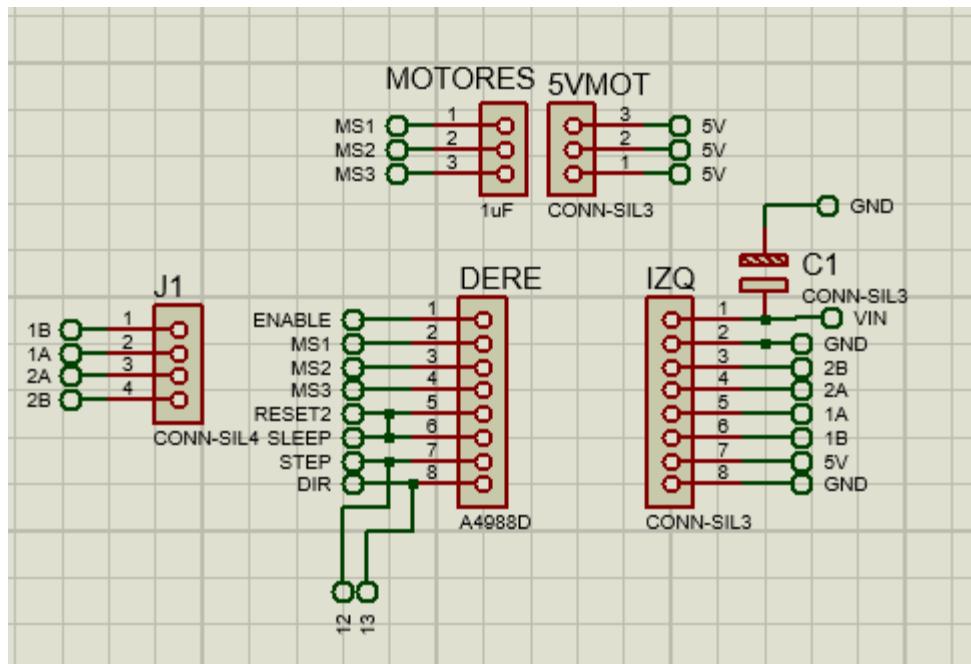


figura 99 Diseño de la base para el driver a4988.

La ventaja del sensor SFR08 es la comunicación mediante el protocolo I2C que ofrece, de esta forma se obtiene la información mediante dos pines y usar dos más para la alimentación de 5V y GND. También agregaremos entradas para conectar un módulo bluetooth y un módulo SD para poder guardar las capturas de las distancias provistas por los sensores, sin embargo , este módulo no se utilizó.

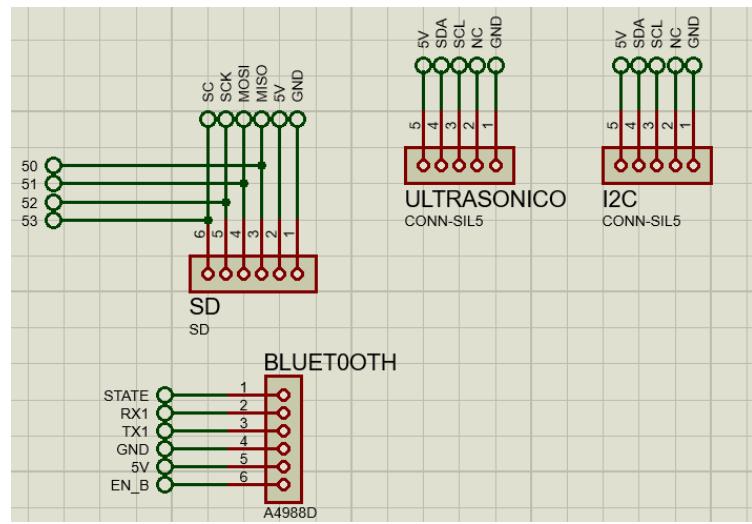


figura 100 Diseño de la base para los sensores SFR08, modulo bluetooth y modulo SD

Finalmente se diseñó y etiquetó los pines que se utilizaron para colocar la LCD de 240*128 pixeles.

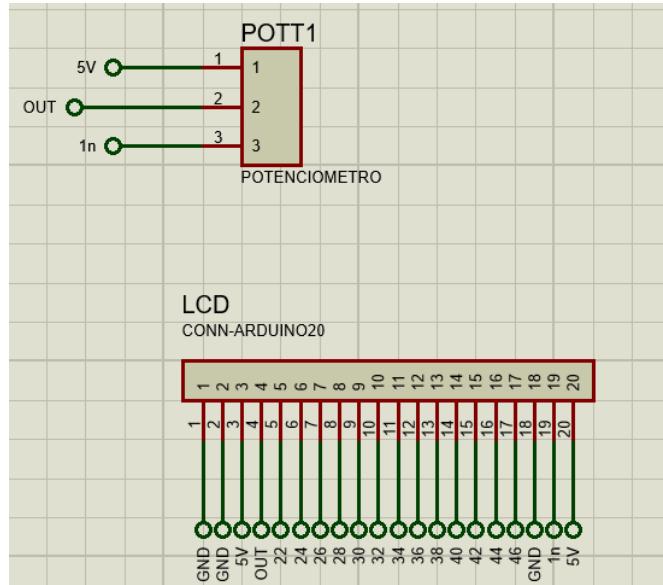


figura 101 Diseño de la base para la LCD 240*128 Píxeles.

PCB

Con un esquemático ya hecho, el diseño de la PCB se vuelve más sencillo, primero las medidas de la placa serían de 128mm de largo y de 70.5mm de ancho.

Agregamos los componentes dentro de la PCB dándole un orden adecuado, el sistema de ISIS proteus indica el conexionado de los componentes tomando la referencia del esquemático previamente hecho, luego de colocar todos los componentes se unieron con pistas de un grosor de 30Th (milésima de pulgada).

También se agregó una superficie de disipación la cual será aterrizada a tierra para así reducir el ‘ruido’ de las señales, y ya que usaremos el método de transferencia por calor para hacer la placa, el tiempo que estará en el cloruro férrico es corto.

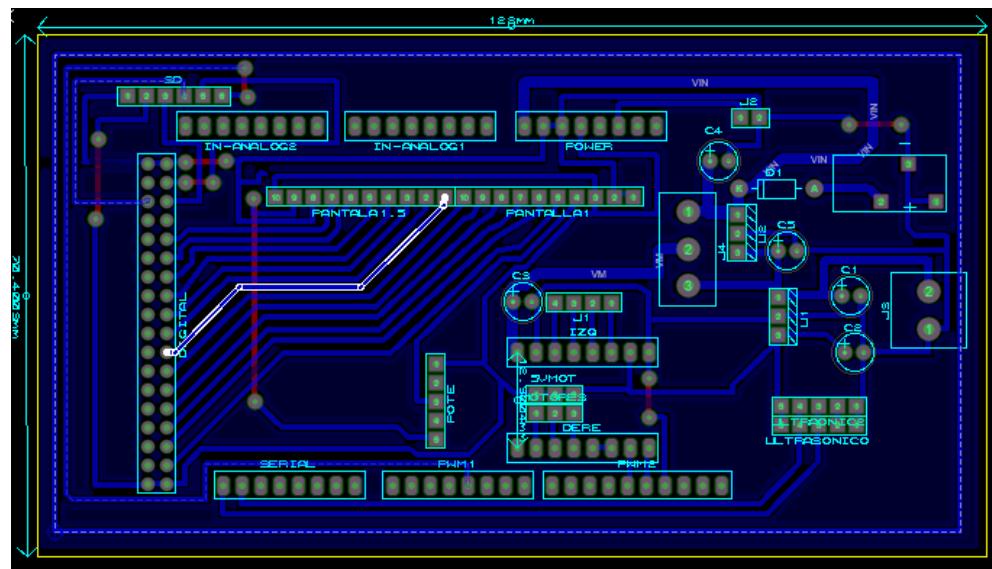


figura 102 Diseño de la PCB en proteus.

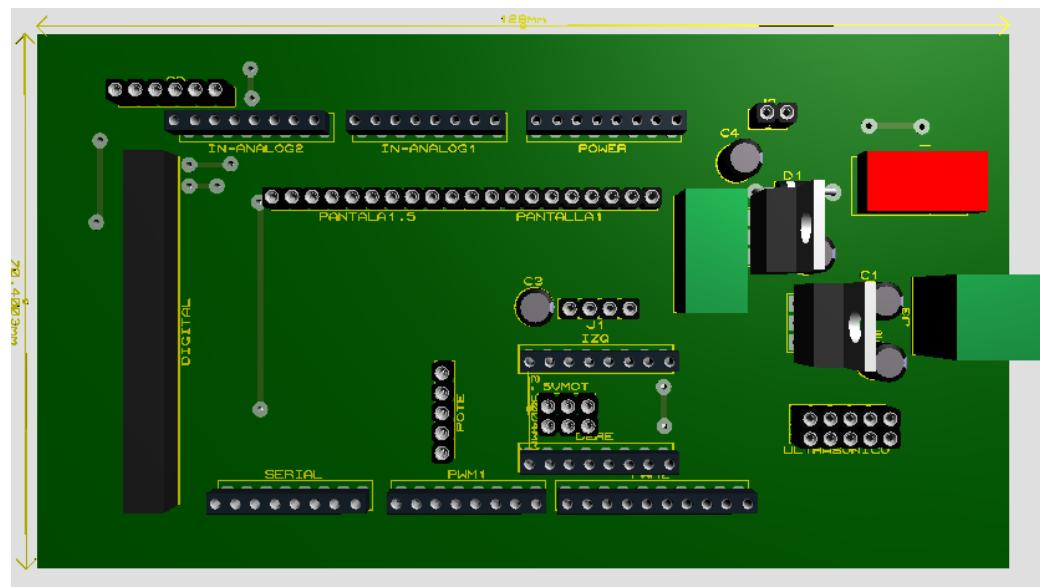


figura 103 Diseño CAD generado por ISIS proteus

4.2 Diseño de la shield final profesional para Arduino mega

El diseño anterior funcionaba perfectamente y cumplía con su propósito, sin embargo, no contaba con una etapa de potencia para los motores VEX EDR por lo que se agregó las salidas que controlarían los motores, se decidió utilizar los puentes H que VEX recomienda, por lo que para controlarlo solo necesitaremos una salida para cada motor también alimentación y GND.

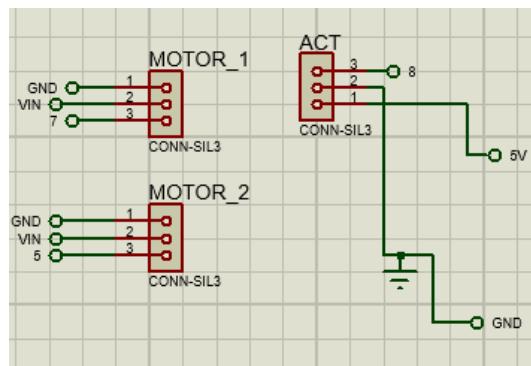


figura 104 Pines para control de motores y activador Infrarrojo

Como se aprecia en la figura 4.8 también se agregó un conector de tres pines para implementarle un sensor infrarrojo y de esa forma poder activarlo remotamente si así se requiere.

Para diseñar la PCB se planteó utilizar las herramientas que el editor de diseño que proporciona proteus, primero se dibujan las dimensiones de la placa de 100mm de largo por 61mm de ancho, también se configura el “gestor de reglas de diseño” para elegir el grosor de pista, estrangulado, distancia entre pads entre otras.

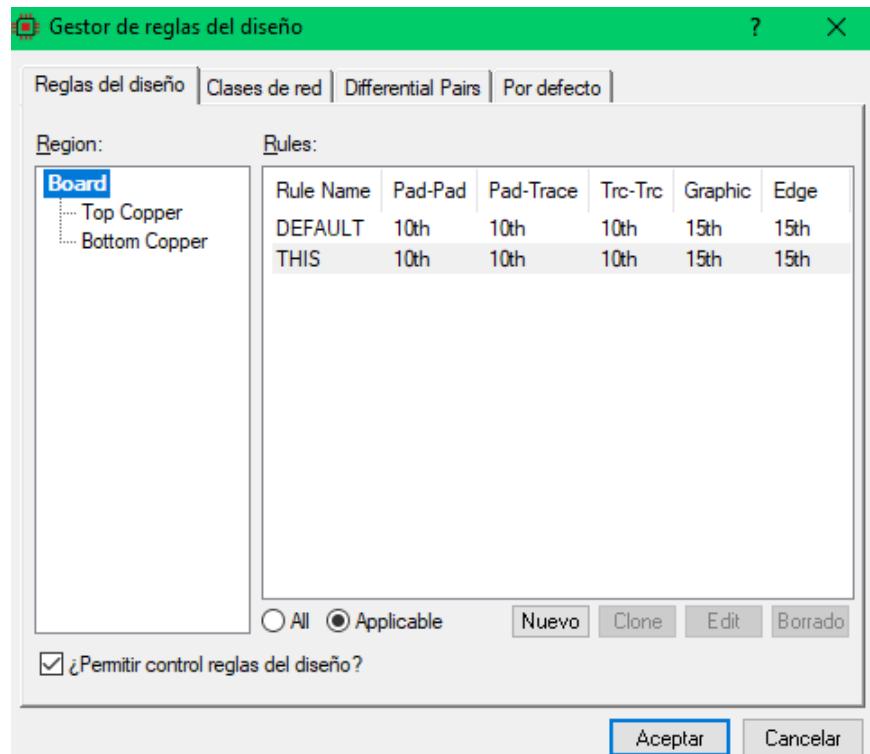


figura 105 Gestor de Reglas del diseño

Después de colocar los componentes que llevaría la placa, se unen las pistas de alimentación de los motores con un grosor de pista de 50th ya que ahí es donde conducirá más corriente, para unir las pistas de señales se utilizó la herramienta “auto trazador” la cual unirá las pistas de la mejor forma posible automáticamente.

Finalmente se agrega la superficie de disipación tanto en la capa inferior como en la superior, también se añaden algunas serigrafías como logos de la institución entre otras cosas para finalmente obtener el resultado visto en la figura 4.10.

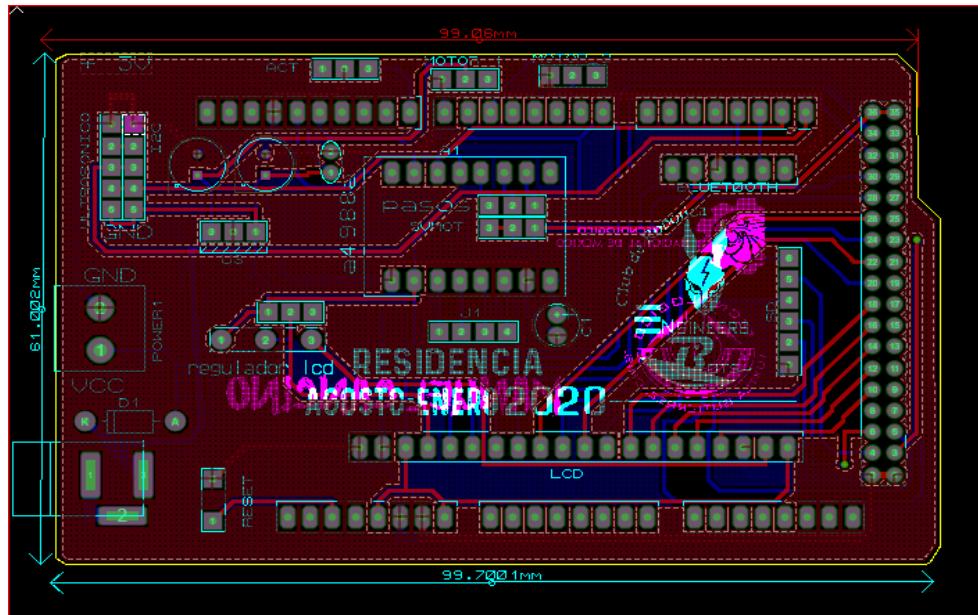


figura 106 Diseño final de la PCB



figura 107 Diseño CAD de la PCB final generado por ISIS proteus (A)

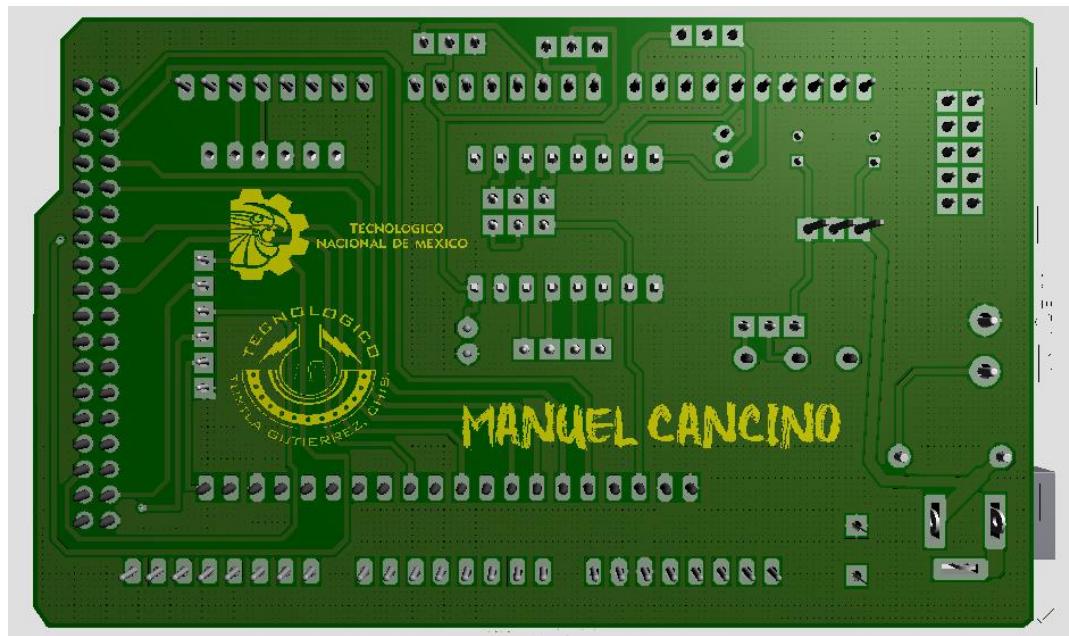


figura 108 Diseño CAD de la PCB final generado por ISIS proteus (B)

Luego de terminar el diseño se exporta el archivo Gerber y se comprime en un archivo .zip y así enviarlo a Pcb Way para que elaboren la placa y tener un aspecto profesional, al paso de unas semanas se obtiene la placa como se muestra en las figuras 4.13 y 4.14.



figura 109 Placa PCB finalizada (A)

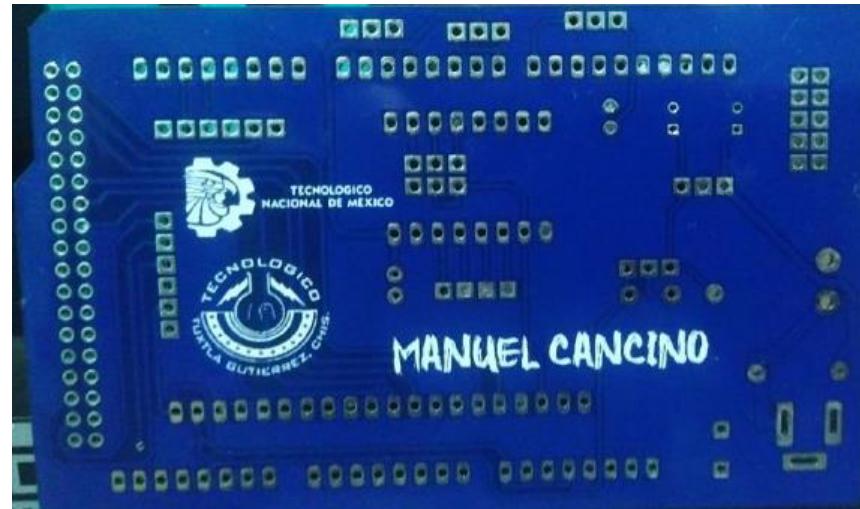


figura 110 Placa PCB finalizada (B).

Capítulo 5. METODOLOGÍA

5.1 Construcción del prototipo Robótico

En este apartado se explica la construcción del prototipo robótico el cual se dividirá en dos secciones:

- Construcción del chasis
- Construcción de la base giratoria

5.1.1 Construcción del chasis

Para elaborar el chasis se construyó un marco con los carriles de carril 2 x 1 x 25 agujeros los cuales cuentan con una medida de 32cm de largo y 3 cm de ancho. Para el diseño del chasis se utilizó seis piezas de carril, finalizando la estructura del marco con 32cm de largo y 32cm de ancho.

Inicialmente se colocó dos motores VEX EDR 269 uno en cada lateral, se utilizaron las llantas de 2” de fricción para movilizar el prototipo, de igual forma se implementó los ejes donde se sostendrían las cuatro ruedas con sus respectivas chumaceras que evitarían la fricción de los ejes.

Utilizando dos canales de aluminio de las dimensiones 1 x 2 x 1 x 25 agujeros los cuales cuentan con una medida de 32cm de largo y 3 cm de ancho. Se colocaron paralelamente entre ellos de tal forma que la base donde se colocaría el motor NEMA17 estuviera en la parte central del prototipo, luego de colocar la base junto al NEMA17, se colocó la caja de engranajes reductora la cual se había impreso en 3d.

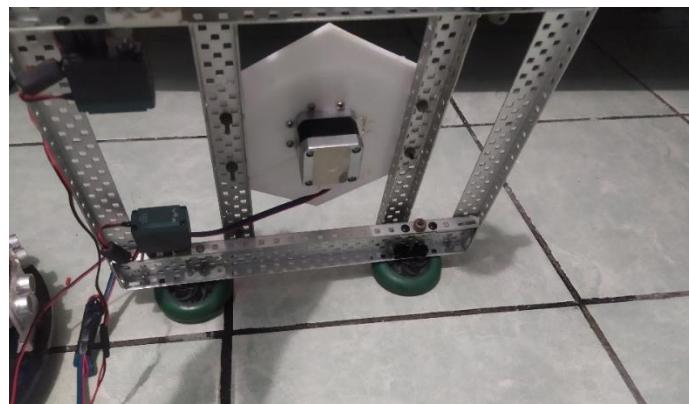


figura 111 Montaje del chasis vista inferior

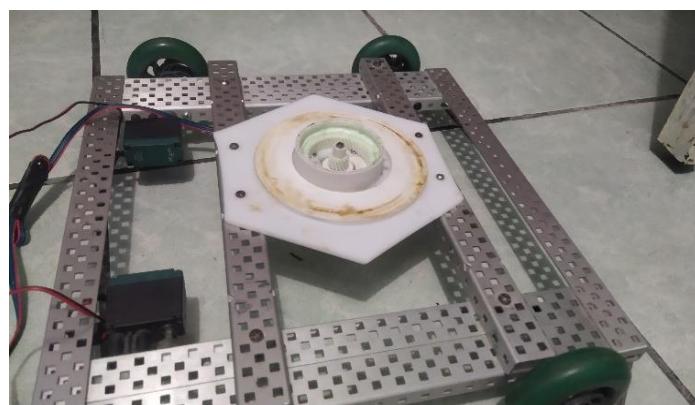


figura 112 Montaje del chasis vista superior



figura 113 Montaje de la caja reductora de engranes helicoidales (A)

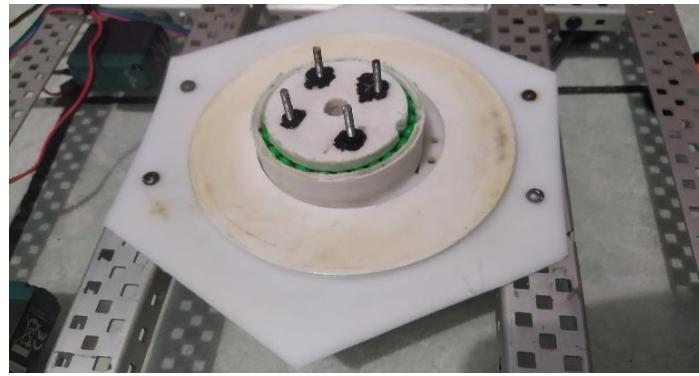


figura 114 Montaje de la caja reductora de engranes helicoidales (B)

5.1.2 Construcción de la base giratoria para el prototipo robótico

Para la construcción de la base giratoria se utilizó un círculo de 31cm de diámetro y con un grosor de 6 mm del material ABS plastic, para la base de los sensores ultrasónicos SRF08 se utilizó acrílico transparente cortado a medida para cada sensor.

Se procedió a elaborar el ensamble físico de la base giratoria como se observa en la figura 5.5.

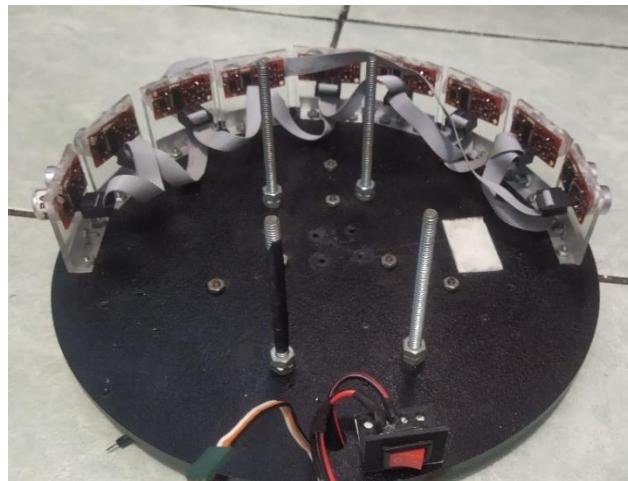


figura 115 Montaje de la base giratoria con los sensores ultrasónicos SFR08.

Para la base donde se colocaría la electrónica de control y potencia se utilizó un cuadro de acrílico blanco de 16cm de ancho por 16cm de largo, con un grosor de 3mm. Sobre la base de acrílico se colocaron la placa Arduino mega 2560 con el shield que serviría para controlar los sensores y actuadores del prototipo.

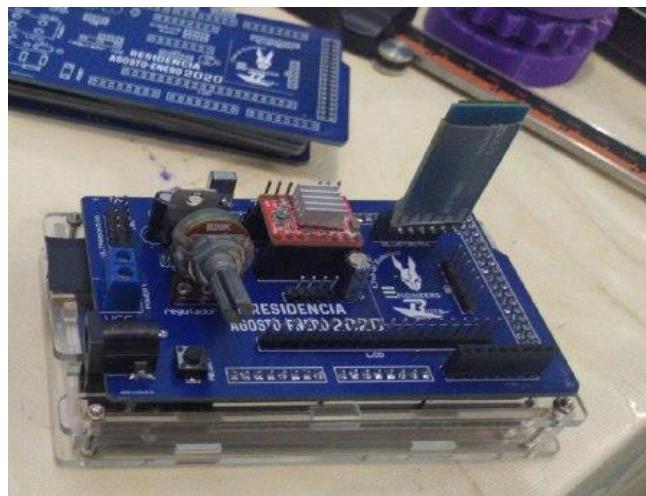


figura 116 Montaje del shield de control junto a la placa Arduino mega 2560

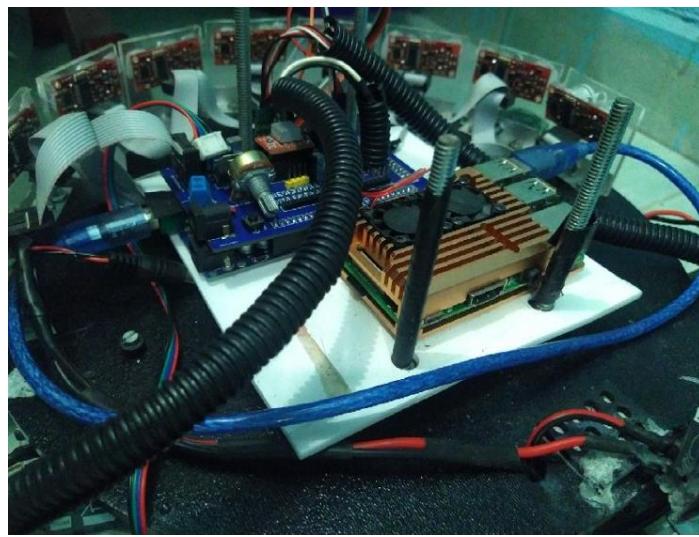


figura 117 Montaje completo de la base giratoria

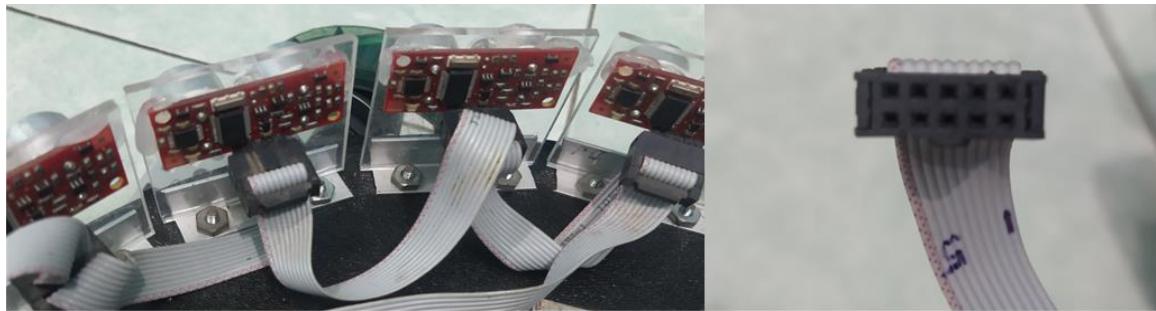


figura 118 empalmado de cables planos para el protocolo de comunicación I2C de los sensores ultrasónicos SFR08

Para el desarrollo de la red neuronal se optó el utilizar una raspberry pi 3 B+ la cual recolectaría los datos y haría la comunicación remota con el prototipo robótico y el pc. Finalmente se unió la base giratoria con el chasis ensamblado anteriormente.

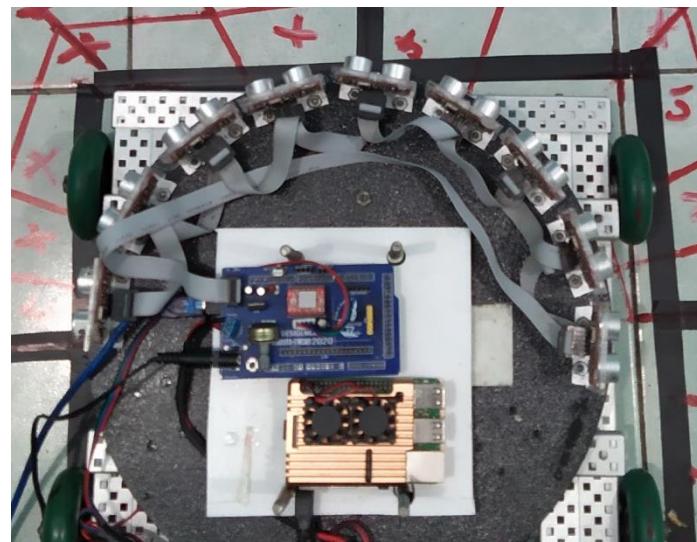


figura 119 Montaje del prototipo robótico

5.2 Programación en la placa Arduino mega 2560

En esta sección se procederá a explicar la programación con la cual se ha conseguido realizar los movimientos de giro para recolección de datos, recorridos y cambios de dirección del robot. Para ello, el software cuenta con diversos apartados diferenciados, como los siguientes:

- Adquisición de datos del exterior mediante los sensores Ultrasónicos SFR08 y el ángulo de giro mediante el MPU6050.
- Graficación de distancias obtenidas mediante los sensores en una pantalla LCD graficadora de 240*128 pixeles.

Para implementar el código se utilizaron varias herramientas de programación, como las librerías que se emplearon para las diferentes comunicaciones que se realizaron. Otras herramientas que se utilizaron son las subrutinas e interrupciones, mediante las cuales se llevaron a cabo diversas operaciones.

5.2.1 Recolección de datos mediante Arduino y los sensores ultrasónicos SFR08

Para la recolección de los datos de los sensores ultrasónicos SFR08 mediante la placa Arduino mega 2560 se utilizó el IDE de Arduino el cual se descarga de forma gratuita en la página oficial de Arduino.

Como mencionamos en el apartado 2.3.1 la ventaja del modelo SFR08 contra el modelo más utilizado SFR04 es el uso del protocolo de comunicación I2C que permite conectar múltiples sensores al bus I2C usando solo dos cables para la comunicación e identificándolos con direcciones diferentes, por ello se explica los códigos utilizados para la comunicación con cada uno de los sensores mediante este protocolo desde Arduino.

Para entender el flujo del programa se utiliza diagramas de flujo que permiten entender de forma visual la lógica del código. Para conseguir los datos deseados se emplea el bus de comunicación I2C que tiene incorporado el propio sensor, el cual se conecta con el módulo I2C de Arduino. La librería Wire.h nos sirve para establecer el protocolo de comunicación I2C y poder configurar correctamente el sensor, también se agrega una dirección la cual será la inicial para los sensores y variables que son necesarias para establecer el protocolo de comunicación, la librería Wire nos incluye las siguientes funciones:

- Wire.begin(): permite iniciar la comunicación I2C.
- Wire.beginTransmission(): inicia la comunicación con el sensor, indicando también su dirección.
- Wire.write(): permite al maestro (Arduino) escribir un byte en el bus de comunicación para que el esclavo pueda leerlo.
- Wire.read(): permite al esclavo escribir un byte en el bus de comunicación para que el maestro (Arduino) pueda leerlo.
- Wire.endTransmission(): finaliza la comunicación.[40]

La comunicación I2C es una comunicación bidireccional basada en la topología de maestro-esclavo, en la que Arduino funcionará como maestro y los sensores ultrasónicos harán la función de esclavo.

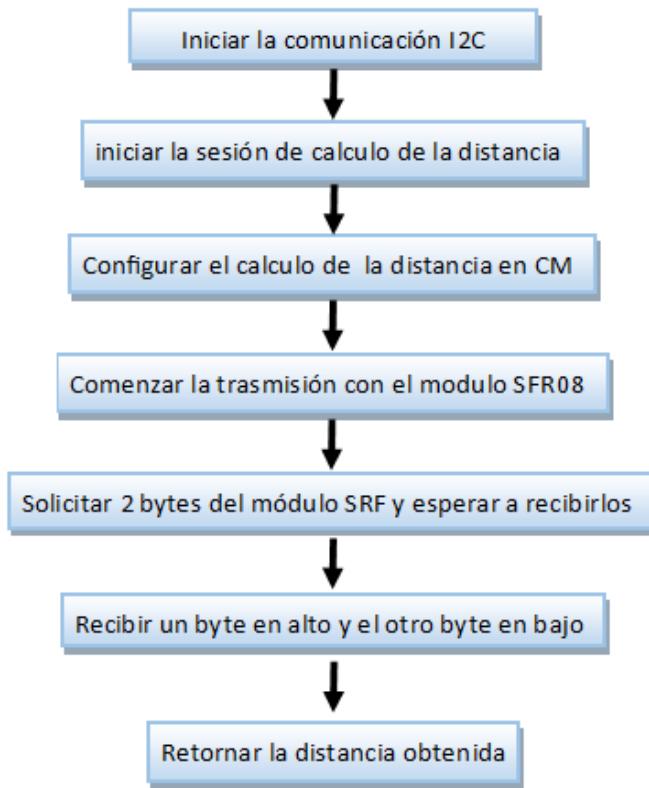


figura 120 Diagrama de bloques de la rutina `getRange()`.

Como se plantea en el apartado 2.3.1 el sensor ultrasónico sfr08 contiene un PIC16F872 el cual se encarga de tomar las distancias para posteriormente enviarlas cuando se le sean requeridas, por lo que se configura ciertos registros del microcontrolador para indicarle cuando se necesitan sus datos y en que unidades de medidas se precisan, se puede tener hasta 16 sensores conectado en el bus I2C por lo que se creó la función `getRange(srfAddress)`, la cual se puede observar en el código A.1 del Anexo Códigos, esta función se reutiliza con las diferentes direcciones de los sensores.

SFR08	Dirección I2C
sensor 1	0x70
sensor 2	0x71
sensor 3	0x72
sensor 4	0x73
sensor 5	0x74
sensor 6	0x75
sensor 7	0x76
sensor 8	0x77
sensor 9	0x78

Tabla 3 direcciones para el protocolo de comunicación i2c de los sfr08

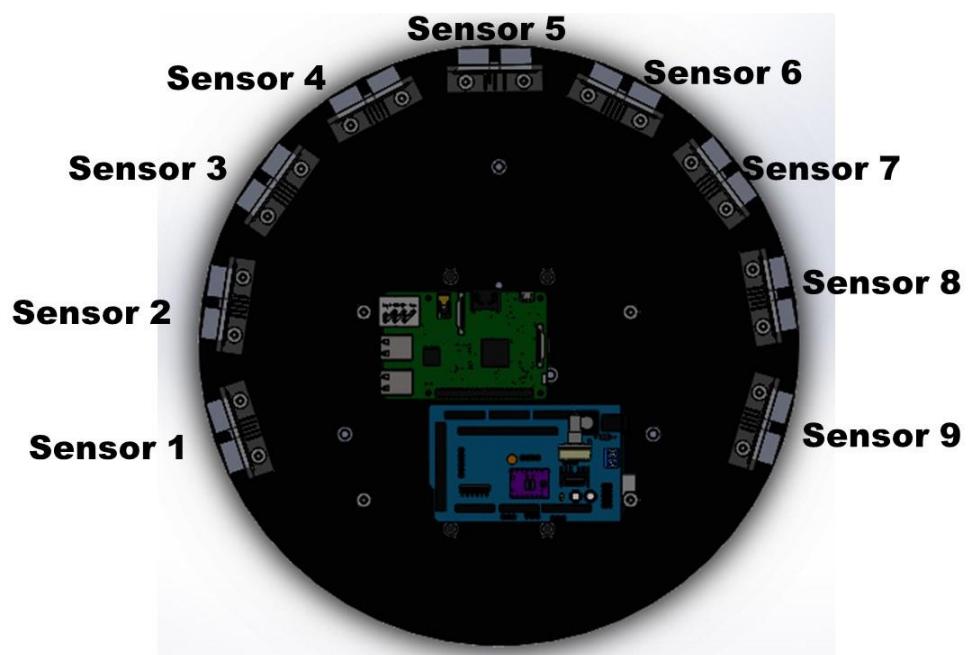


figura 121 Distribución de los sensores SFR08 en el prototipo robótico

5.2.2 movimientos del prototipo Robótico

Una vez creada la rutina de recolección de datos para sensores ultrasónicos se pasó a crear rutinas o funciones que servirán para mover el motor paso a paso (Nema17) 4.5° y así conseguir una buena resolución de captura, como se mencionó en el capítulo 3, cada sensor se ubica con una apertura 22.5° entre sensor y sensor, por lo que para obtener la resolución de 45 datos tenemos que mover la base giratoria los grados ya mencionados, de igual manera se necesita regresar a la posición inicial, inicialmente se estableció un contador de los pasos que se habían mandado al driver y con ello reiniciarlo a la posición de origen, sin embargo, al hacer varias pruebas de recolección se notó un desface a la posición original, por lo que se agregó un final de carrera y con ello regresar a la posición inicial girando en sentido contrario hasta que el final de carrera sea activado.

Como se observa en la imagen 5.12 se tiene el diagrama de flujo que sirve para la recolección de datos y giro de la base recolectora, se inicia esperando a que exista datos en el puerto serial , si hay algún dato se procede a identificarlo y con ello si el dato es “1” entrara en un bucle “for” en cual se hará 5 ciclos de recolección así se sumaran los 45 datos recolectados ya que se cuenta con 9 sensores, como se observa en el diagrama se tiene que hacer uso de rutinas para girar la base, por lo cual se describen dichas rutinas en el código A.2 y A.3 del Anexo Códigos.

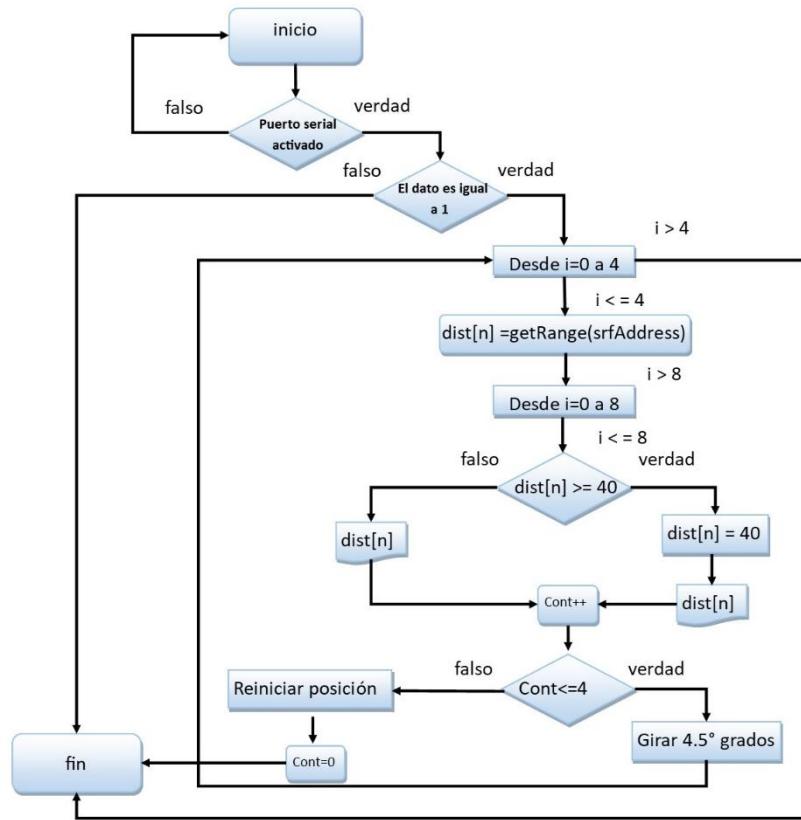


figura 122 Diagrama de flujo para la recolección de datos

Luego de obtener los resultados esperado con la base giratoria quedaría por desarrollar la parte del desplazamiento para el prototipo robótico, para esto el robot se moverá en línea recta y cuando esté detecte un obstáculo con los sensores frontales se detendrá para proceder hacer el reconocimiento del objeto que tiene frente a él, posteriormente imprimir los datos en el puerto serial y girar 90° hacia la derecha.

Para lograr que el movimiento del robot sea en línea recta y haga los giros correctamente se utilizó la IMU MPU6050 como se menciona en el apartado 2.3.2, la cual combina un giroscopio de 3 ejes junto a un acelerómetro de 3 ejes , lo interesante del mpu6050 es el uso de un DMP que se puede programar con firmware y es capaz de hacer

cálculos complejos con los valores de la IMU, al hacer uso del DMP el microcontrolador se puede encargar de otros procesos.

En el diagrama de flujo de la figura 5.13 se explica la obtención del ángulo mediante el DMP.

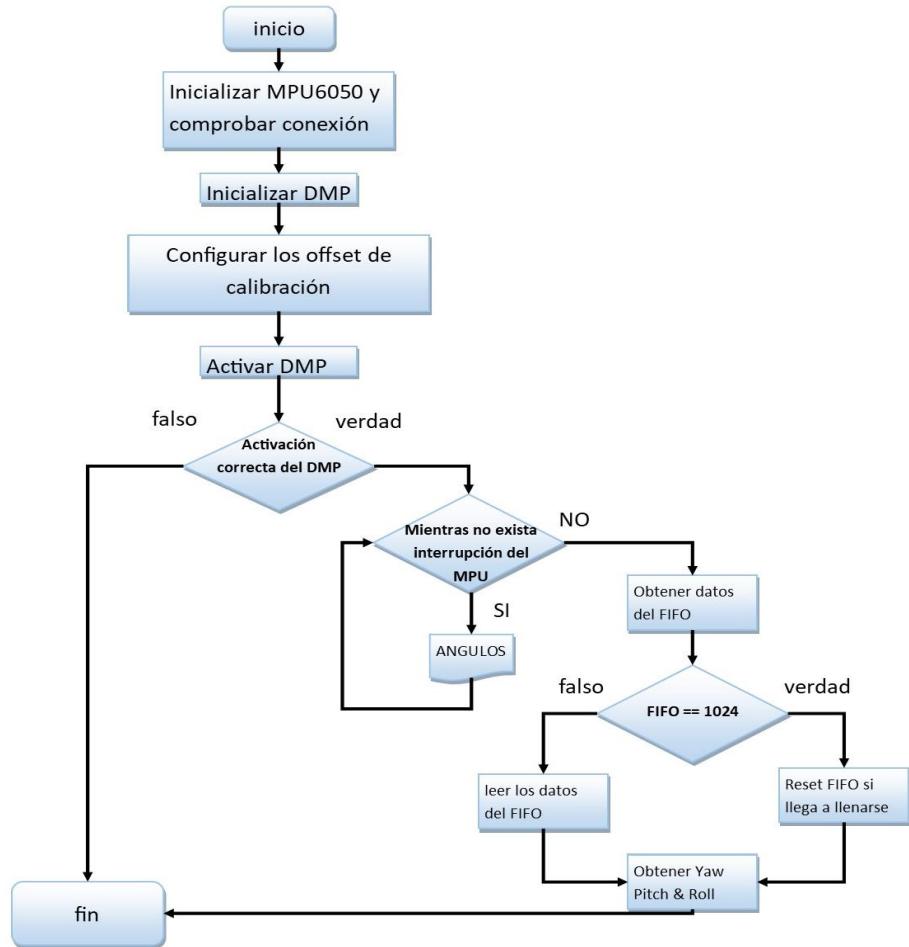


figura 123 Diagrama de flujo para la configuración y obtención de los ángulos mediante el MPU6050 y su DMP

Luego de haber inicializado correctamente el DMP y obtener los ángulos y las aceleraciones de cada eje se procede a crear funciones para mover los motores, estos motores manejan un ESC por lo que para controlarlo tenemos que enviar un tren de pulsos cada 20ms es decir a 50hz, la anchura del pulso determinara la velocidad del motor , esta anchura tiene la duración de 0.5 y 2.5 ms.

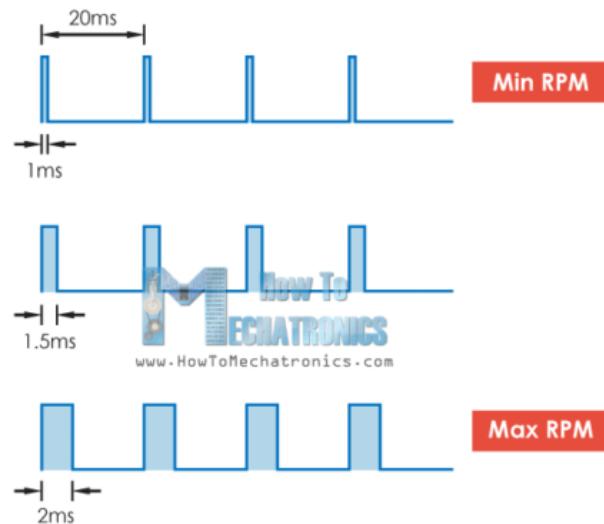


figura 124 Tren de pulsos a 50hz para el control de velocidad

Como se menciona en el apartado 2.4, los motores por motivos de fabricación, desgaste o relación de engranaje entre otras cosas hacen que las RMP no sean iguales en ambos motores por lo que se necesita aplicar un control que iguale las velocidades en ambos motores para conseguir que el motor avance en línea recta, de lo contrario giraría hacia los costados dependiendo del motor que tenga mayor velocidad.

Para conseguir que el robot se desplace en línea recta será necesario convertir la información del ángulo de inclinación proporcionada por el sensor MPU-6050 en una velocidad de giro de las ruedas tal, que corrija dicha inclinación. El sistema gracias al cual se consigue la relación entre una entrada, y una salida deseada, es un control basado en la realimentación de la salida del sistema. Para este caso, el esquema del sistema realimentado será el siguiente:



figura 125 Sistema de control a implementar

Existen una gran variedad de controles diferentes, para este caso se optó por diseñar un algoritmo de control PID digital e implementarlo en Arduino. Este cálculo se realiza mediante la implementación de la librería PID, el hacer esto hace el código más legible y eficiente al configurar los parámetros necesarios, esta implementación se encarga de combinar las tres partes del control(proporcional, integral y derivativa) en una acción de control.

Haciendo esta implementación el diagrama de bloque quedaría como se observa en la figura 5.16.

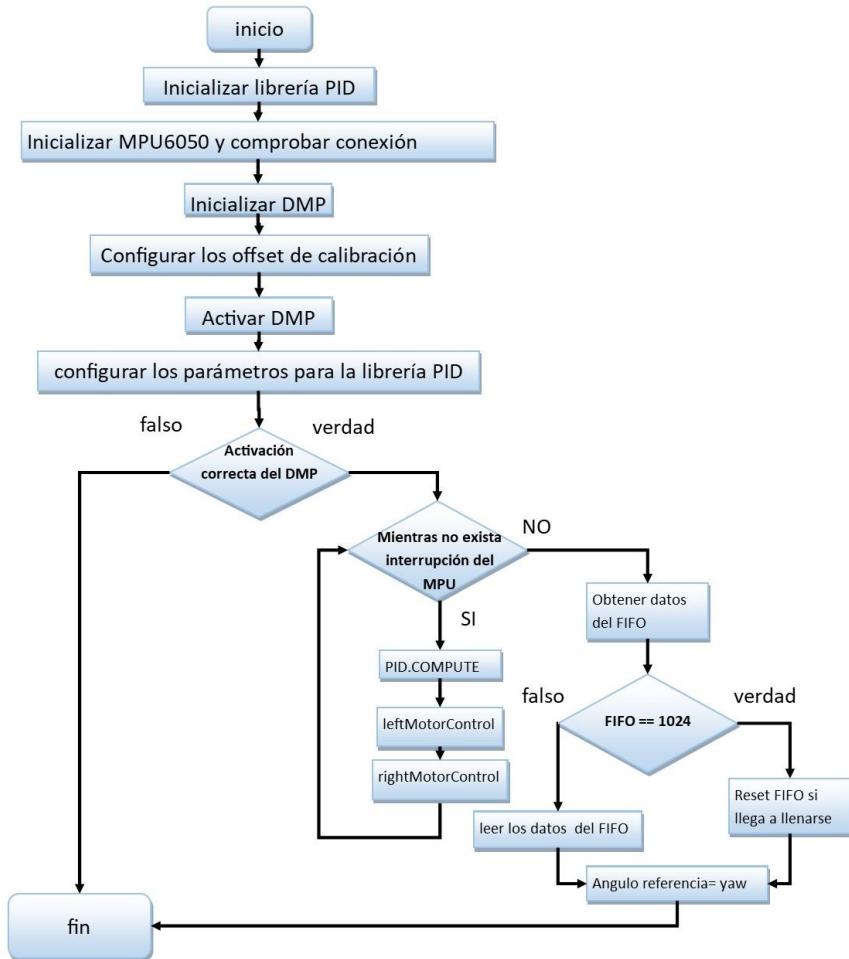


figura 126 Diagrama de flujo para la configuración del DMP y control PID

Como se observa en el diagrama de bloques se desarrollaron dos funciones para el control de los motores dichas funciones se visualizan en el código A.4 y A.5 del Anexo Códigos.

int leftMotorControl(int value).- esta función controla la velocidad del motor de la izquierda a una velocidad constante (int value) entre 0-100%, y que como control usa un control PID el cual calcula la inclinación que tiene el prototipo robótico durante el recorrido

y como resultado arroja un valor de 0-100%, el cual se suma a la velocidad constante para equilibrar la velocidad del motor.

int rigthMotorControl(int value).- esta función hace el mismo control que “leftMotorControl(int value)” la diferencia es que su control se basa en el motor derecho.

5.2.3 Comunicación entre Arduino y Raspberry

Inicialmente para almacenar los datos recolectados de los sensores, se planeaba utilizar un módulo SD el cual se comunicaría mediante el protocolo SPI a la placa Arduino, pero esto dio ciertas complicaciones como:

- Poco control sobre la cantidad de datos recolectados
- No poder identificar datos erróneos enviados por nuestros sensores
- Mantener conectado la placa Arduino al pc para iniciar la recolección por el puerto serial

Estas limitantes hicieron replantear el uso de una SD como método del almacenamiento directo con Arduino, al entender esto se decidió el utilizar una raspberry pi ya que el uso de esta como mini ordenador permite la comunicación serial directa con Arduino y al mismo tiempo un control remoto desde otro ordenador.

El modelo elegido fue una raspberry pi 3 B+ la cual dispone de muchas características como vimos en el apartado 2.6.1.

Al hacer uso de esta placa se tiene que realizar algunos pasos para poder utilizarla, a continuación, se explica el proceso que se realizó para utilizarla en este proyecto:

Materiales para el uso de la Raspberry pi 3B+

- Tarjeta raspberry pi

- Tarjeta SD de al menos 8 Gb para el caso fue de 32Gb
- Fuente de alimentación de 5V de 3A.
- Cable de red ethernet

1. Elegir el sistema operativo

Existen diferentes sistemas operativos soportados por Raspberry Pi 3, y evidentemente, cada cual tiene sus particularidades en el proceso de instalación. Se instalo el sistema operativo oficial Raspberry Pi OS que proporciona la página de la fundación raspberry.

Para ello se descarga el archivo comprimido en la sección de software con el nombre de “Raspberry Pi OS with desktop and recommended software” .

2. Formatear la tarjeta SD

Después de haber descargado la imagen del sistema operativo, se toma la tarjeta SD y se formatea con el formato FAT32 para lo cual se utilizó un software para tal acción llamado “SD Card Formatter”, en este software se selecciona la ruta de la tarjeta SD , formato y nombre de la SD, por último, se formatear como se muestra en la figura 5.17.

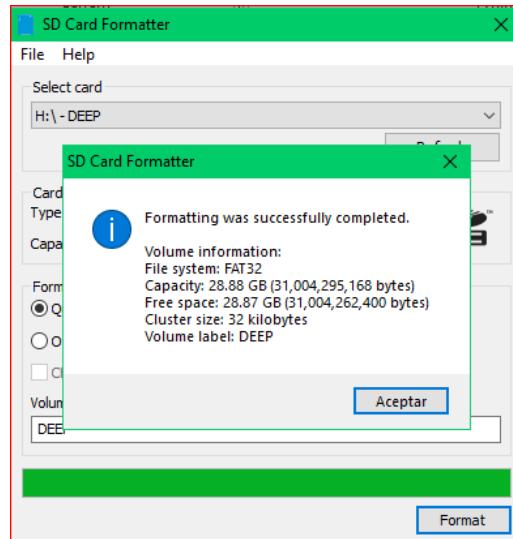


figura 127 Software para formateo de tarjetas SD

3. Instalación del sistema operativo en la tarjeta SD

Para la instalación se utilizó un software llamado “BaleanaEtcher”, el cual nos permite seleccionar el archivo comprimido que contiene el sistema operativo y la tarjeta SD en la que se instalará como se muestra en la figura 5.18, luego de hacer la selección correspondiente se inicia la instalación.

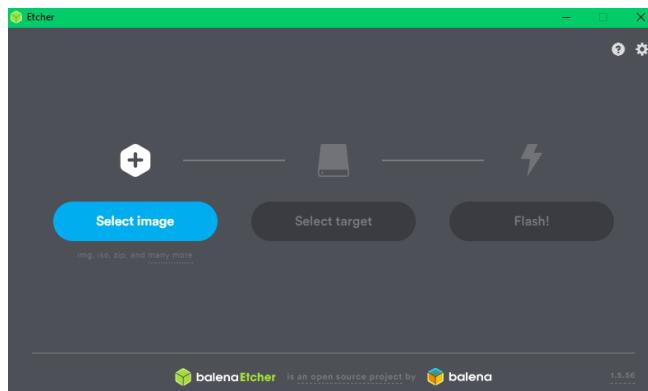


figura 128 Software baleanaEtcher

4. Accediendo de forma remota

Una vez terminado el proceso de instalación, se conecta la raspberry con el cable de red para asegurar que la raspberry esté conectada a una red privada, con esto se puede ver cuál es el número de IP con la que se conectó y así acceder a ella de forma remota, de no contar con un cable de red disponible se puede utilizar un monitor, teclado y ratón para configurar una IP estática para la raspberry.

La configuración de la IP estática se hizo de forma visual, al entrar por primera vez a la raspberry pedirá que se configure parámetros básicos como lo son:

- Nombre de usuario
- Contraseña
- País y zona horaria
- Idioma
- Tipo de teclado
- Configuración de red y contraseña

Una vez terminado de hacer esto se procede con las configuraciones de red y se selecciona “wireless and wired network settings” como se muestra en la figura 5.19.



figura 129 configurando la IP estática de nuestra raspberry (a)

En el menú desplegable que saldrá se configura tanto la IP , mascara de red, DNS y puerta de enlace. Con lo anterior resuelto ahora se descarga el software VNC connect que permite acceder de forma remota a la raspberry como se muestra en la figura 5.20.

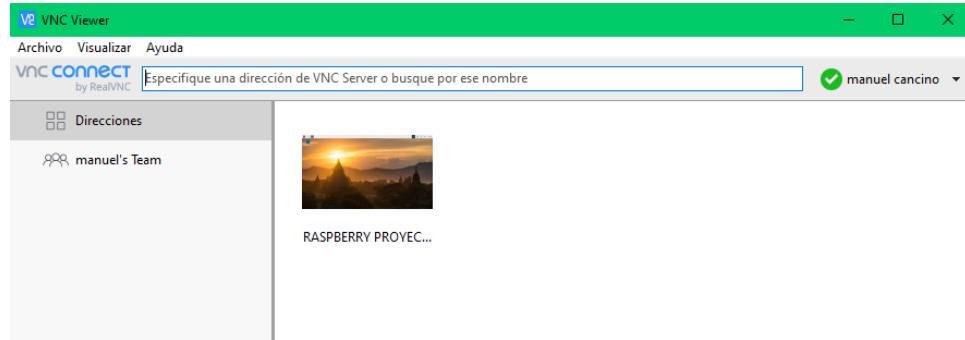


figura 130 VNC Connect

Con las configuraciones anteriores realizadas se puede enlazar a la raspberry pi desde otro ordenador de forma remota, y con ello enviar información a la placa Arduino mega 2560 para obtener los datos capturados por los sensores ultrasónicos, y con esto, tener control sobre la existencia de errores en los datos recolectados, cuantos datos se tienen y en qué momento se inicia la captura de datos.

Antes de poder utilizar el puerto serial para comunicarnos con la placa Arduino Mega 2560 se tiene que habilitar el puerto COM de la raspberry pi, este viene configurado por defecto para usarse con la consola (o terminal) de la raspberry, es necesario quitar la consola para poder emplear la UART en otros fines. Para desactivar la consola hay que hacer los siguientes pasos:

1. Desplegar la herramienta de configuración

Para poder acceder a la herramienta de configuración basta con ejecutar el siguiente comando en la terminal:

```
pi@raspberrypi:~$ sudo raspi-config
```

2. Seleccionar Interfacing Options

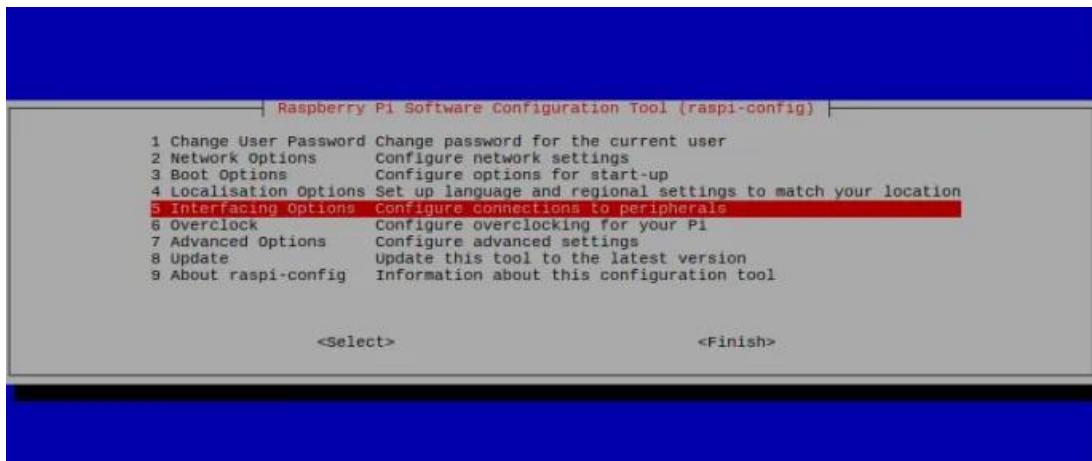


figura 131 Raspi-config – Interfacing options

3. Seleccionar P6 Serial

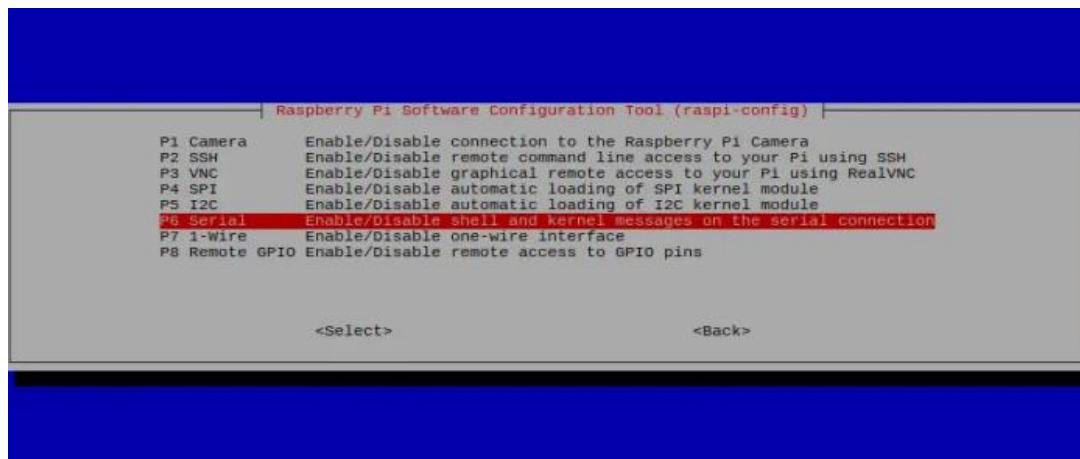


figura 132 Raspi-config – Interfacing options- serial (a)

4. Confirmar la habilitación de la interfaz Serial

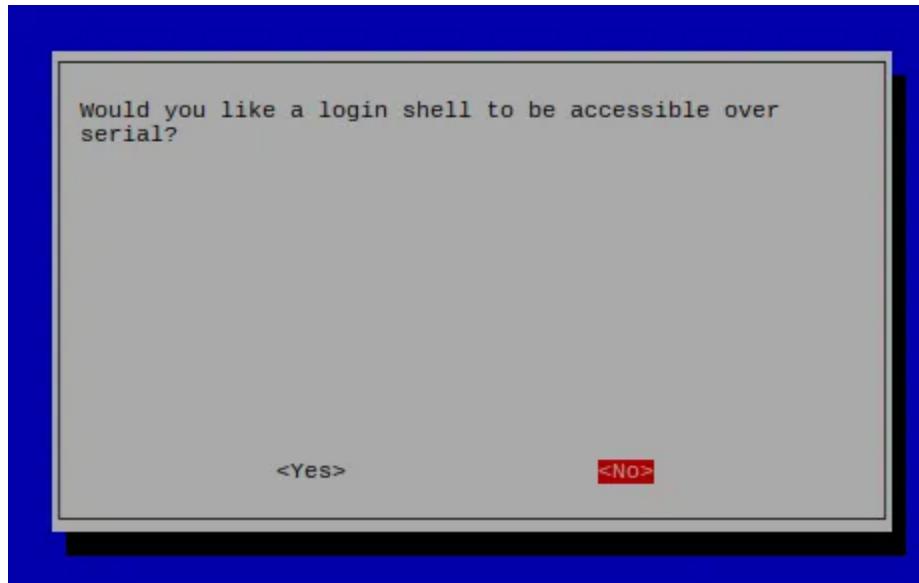


figura 133 Raspi-config – Interfacing options- serial (b)

Confirmar seleccionando <Yes>.

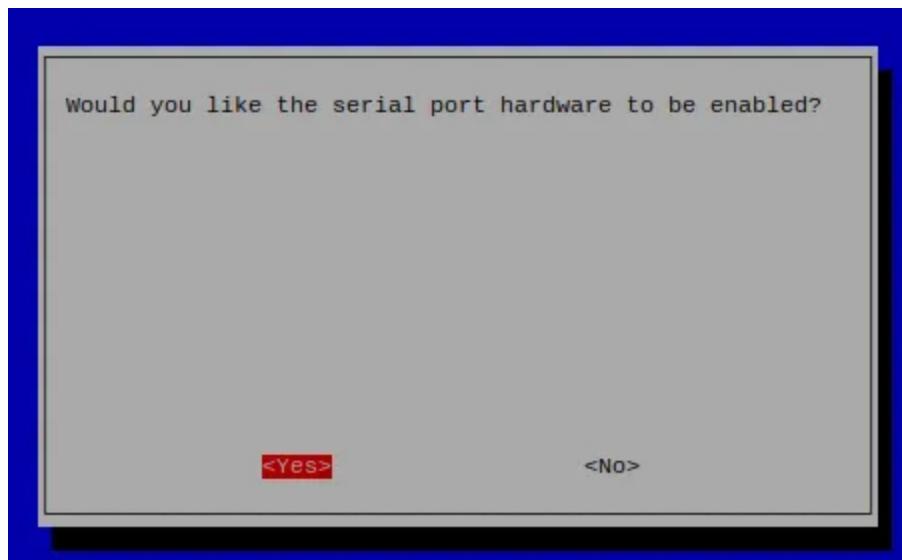


figura 134 Raspi-config – Interfacing options- serial (c)

5. Reiniciar el sistema de la Raspberry Pi

Para que los cambios a la configuración tengan efecto se reinicia el sistema seleccionando <Yes>. Una vez el sistema reinicie se puede utilizar la comunicación Serial en la Raspberry Pi.



figura 135 Raspi-config – Interfacing options- reboot

Luego de este procedimiento ya se puede tener acceso al puerto COM y así comunicarse con la placa Arduino mega enviando y recibiendo datos.

ahora bien, en el diagrama de flujo de la figura 5.26 se explica el script en Python que se utilizó para recolectar los datos enviados por Arduino.

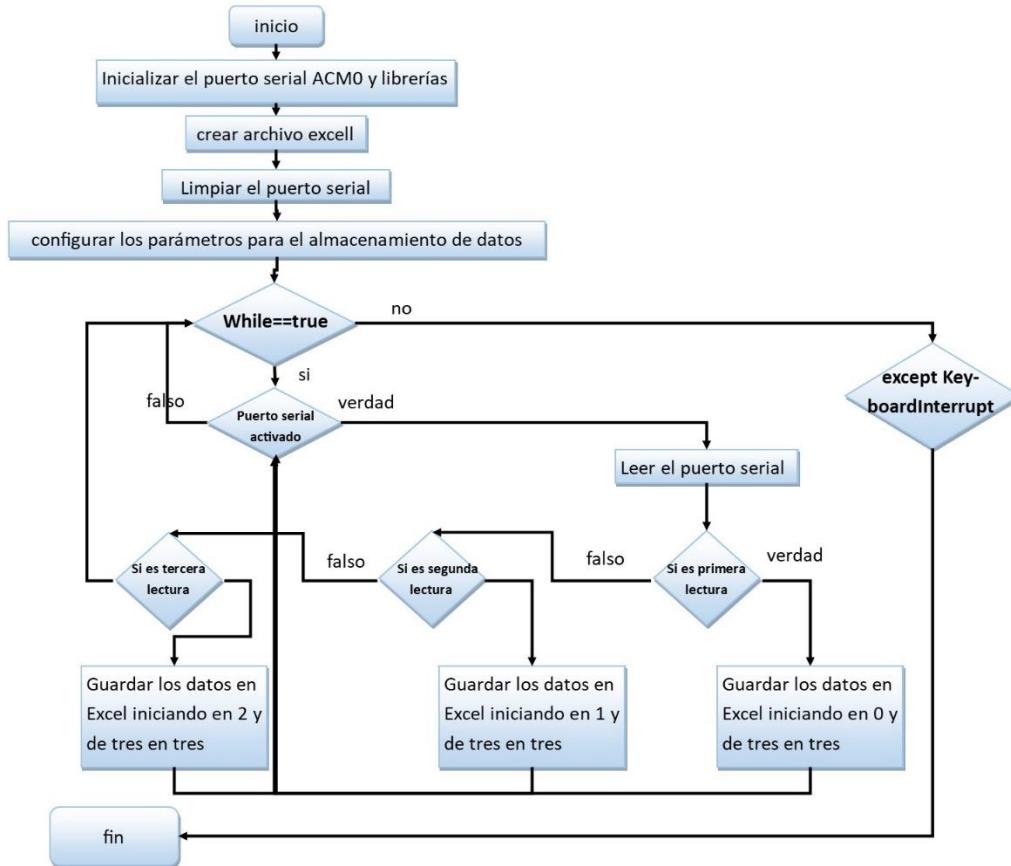


figura 136 diagrama de flujo comunicando raspberry con Arduino mediante el puerto serial

El código en Python se limita a esperar que enviemos el comando de activación y leer los datos en el puerto serial para crear un archivo .xls en el cual se ordenan los datos recaudados, como se menciona en el capítulo 3 y en la sección 5.2.2 se cuenta con 9 sensores los cuales proporcionan distancias, sin embargo, la plataforma giratoria hace 4 giros de 4.5° por lo que se obtiene 45 datos de distancias, así pues se necesitan ordenar de tal forma que los primeros 5 datos sean del sensor uno, los próximos 5 datos sean del sensor dos y así sucesivamente.

En este primer código de Python solo se toma tres lecturas y los ordena de 3 en 3, esto sucede porque en un principio se pretendía tener 27 datos con los cuales trabajar, en la sección 5.4 se describe por qué se cambió la resolución de captura, pasando a los 45 datos finales.

5.2.4 Comunicación Arduino y Matlab

El objetivo principal de esta comunicación es el poder comunicar la placa Arduino directamente con el software con el que se realizaría el entrenamiento de la red neuronal artificial, el uso de la raspberry y Python para la recolección de datos suponía más trabajo al momento de recuperar los datos finales guardados en los archivos .xls, es cierto que Python da una gran flexibilidad y control sobre la recolección de datos, pero esto es de la misma manera e incluso mejor el recolectar los datos por medio de Matlab y guardarlos en una dirección de archivo específica donde podríamos guardar los archivos correctamente y tener mayor facilidad al acceder a ellos.

Así pues, Matlab facilita la comunicación serial, por lo cual solo se tiene que conectar la placa Arduino mega 2560 y configurar el puerto COM al cual se ha conectado la placa junto a la velocidad de baudios por segundo en los que se transmitirán los datos , al terminar la configuración se usó la misma lógica de programación provista en el diagrama de flujo para el código de Python de la figura 5.26 en la sección 5.2.3.

5.3 Comunicación Matlab y Raspberry

Con relación a la comunicación serial, existían algunas deficiencias como la de tener conectada la placa directamente al ordenador para realizar la captura de datos, por otro lado, como vimos en el apartado 2.7.4 Matlab ofrece soporte para la raspberry pi en sus diferentes modelos, con esto la implementación de código sobre la raspberry desde Matlab se hace más sencillo.

Matlab permite tener acceso a la raspberry pi mediante dos formas:

Comunicación interactiva : Se puede comunicar de forma remota con una Raspberry Pi desde una instalación de escritorio de MATLAB o mediante un navegador web con MATLAB Online. Adquirir datos de sensores y dispositivos de imágenes conectados a Raspberry Pi y luego analizarlos y visualizarlos en MATLAB.

Ejecución independiente : con MATLAB Coder, se puede desarrollar aplicaciones integradas independientes para Raspberry Pi. Utilizar la comunicación interactiva para crear prototipos y desarrollar algoritmos de MATLAB, luego generar automáticamente un código C equivalente e implementarlo en Raspberry Pi para que se ejecute como una aplicación independiente.

Por lo tanto, hace factible la comunicación de Matlab y raspberry, de esta forma la raspberry se comunica con Arduino y Matlab adquiere los datos directamente desde la raspberry, para poder utilizar esta comunicación es necesario agregar el toolbox de raspberry para Matlab y configurar la raspberry pi, a continuación, se describe los pasos necesarios para llevar esto a cabo:

Para iniciar con la comunicación primero se necesitan los siguientes materiales:

- 1 raspberry pi
- 1 adaptador de corriente a 5v 3A
- 1 tarjeta SD clase 10 mayor a 8gb
- El paquete de raspberry para Matlab instalado en el ordenador.

primero se necesita el soporte de raspberry para Matlab, por lo que se instala en la pestaña HOME y a la sección Add-Ons.

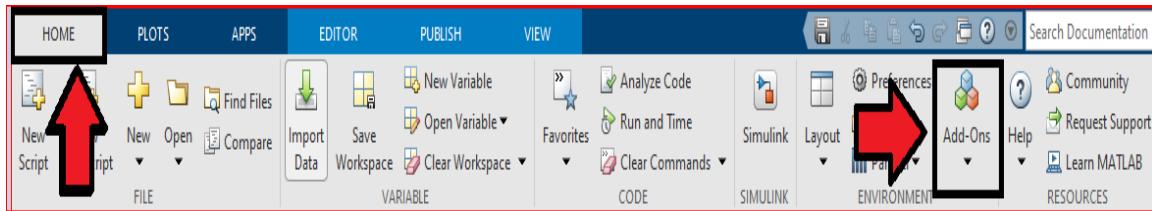


figura 137 instalación del soporte para raspberry de Matlab (a).

Se selecciona “Get hardware support package” y se redirigirá a los paquetes que se tienen instalados, se busca el paquete de raspberry en la barra del buscador y se selecciona, en caso de no tenerla instalada solo se tendría que presionar el botón de instalar y pediría una cuenta de MathWorks la cual se crea gratuitamente y no se necesita licencia para darse de alta con la cuenta.

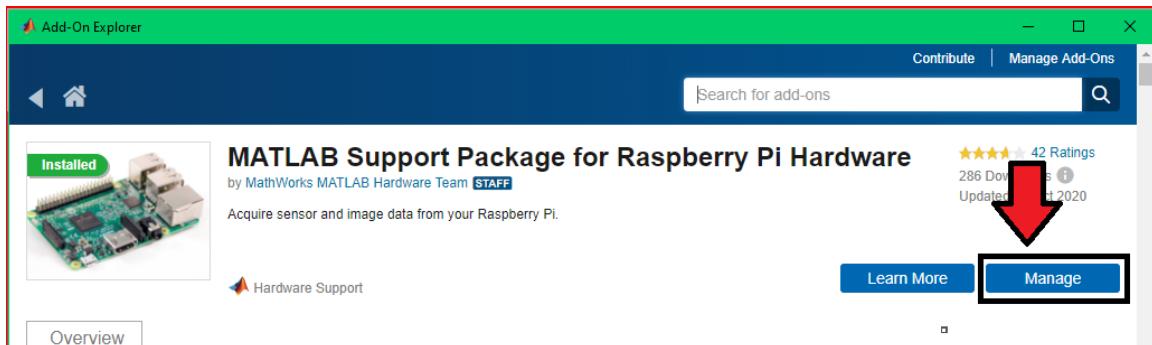


figura 138 instalación del soporte para raspberry de Matlab (b)

En el paquete de raspberry para Matlab y asegurándose de tenerlo instalado, aparecerá un botón con el nombre “manage”, el cual abrirá una ventana donde estarán todos los paquetes con los que se cuentan.

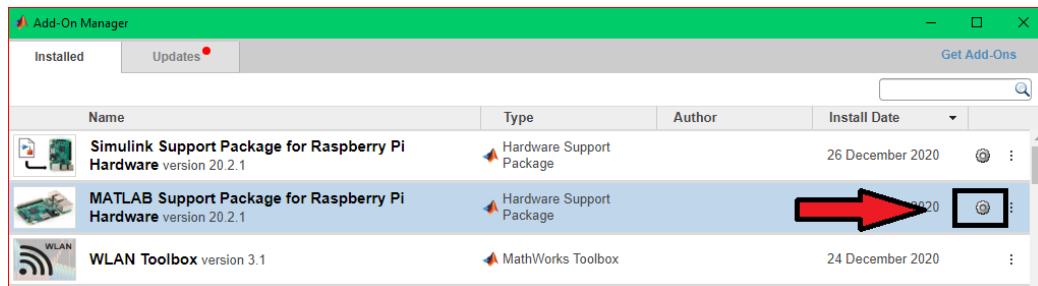


figura 139 Instalación del soporte para raspberry de Matlab - Manage.

Así pues, en el paquete de soporte para el hardware de raspberry pi se da clic en el engranaje que permitirá enlaza la raspberry con la versión de Matlab con la que se cuenta. Luego de ello aparecerá la siguiente ventana en donde seleccionaremos que modelo de raspberry se utilizará.

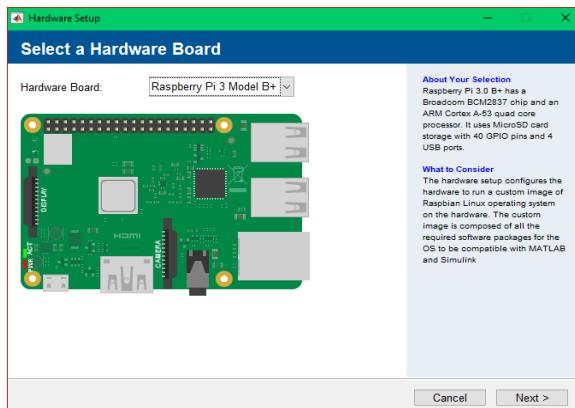


figura 140 Configuración del paquete de raspberry pi - selección de la placa raspberry pi.

posteriormente al dar clic en next aparecerá otra ventana en la que se elige si se utilizará una imagen optimizada de rasbian para Matlab o si se quiere personalizar el sistema que se tiene instalado en la raspberry. Se eligió utilizar una imagen optimizada por Matlab.

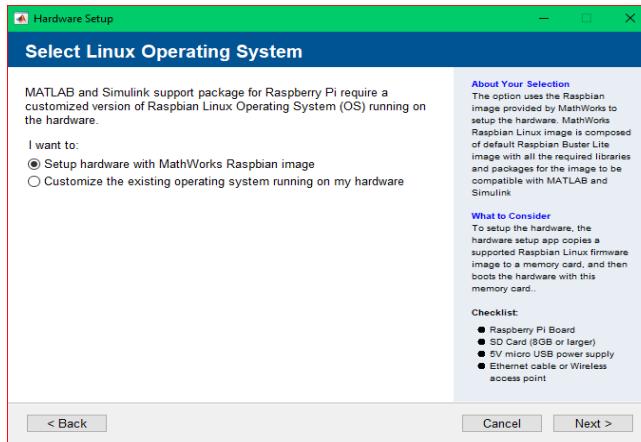


figura 141 Configuración del paquete de raspberry pi - selección del sistema operativo para la tarjeta.

Después de haber hecho la selección se elige entre dos imágenes optimizadas de rasbian, una que permite la comunicación de forma remota desde una computadora a Raspberry Pi y usarla para controlar sensores, actuadores y periféricos. La otra imagen esta optimizada enfocada para ser usada con bibliotecas para Deep learning por lo que elimina ciertas aplicaciones que no se necesitaría para ese enfoque.

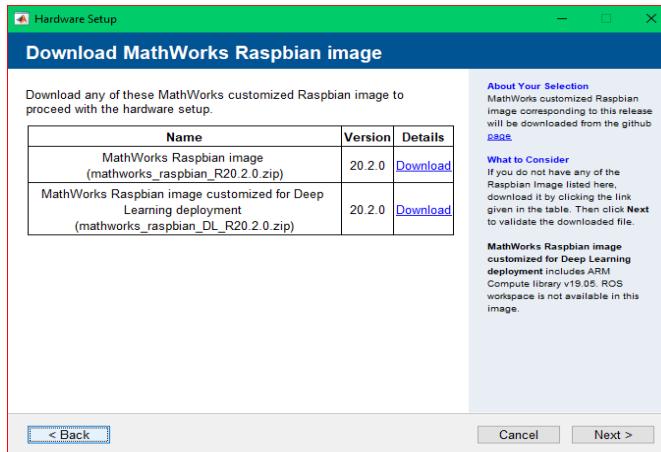


figura 142 Configuración del paquete de raspberry pi - selección del sistema operativo que nos ofrece Matlab

Se utilizo la imagen optimizada para Deep learning. Al seleccionarla empezara la descarga de aproximadamente 1.28gb en un archivo comprimido.

Después de haber descargado la imagen, se formateo la tarjeta SD con el formato FAT32 para lo cual se utilizó el software SD Card Formatter, en este software se seleccionó la ruta de la tarjeta SD junto a su formato y nombre de la SD, por último, se formateo.

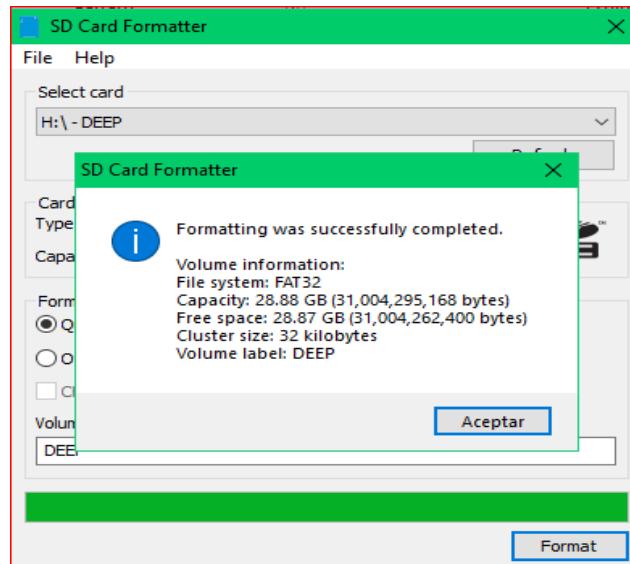


figura 143 Formateando SD

Luego de a ver concluido con el formato, se continuo con la configuración del sistema operativo para la raspberry. Se selecciono la ruta donde se descargó el sistema operativo y se presiona al botón de validar , luego de un momento confirmara que el sistema fue validado y poder seguir con la configuración.

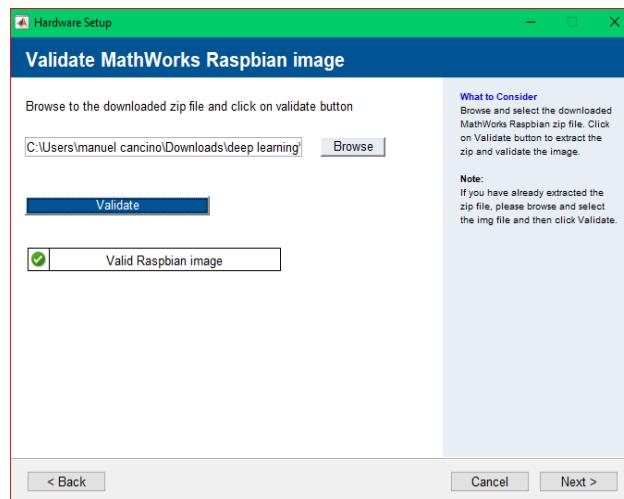


figura 144 Configuración del paquete de raspberry pi - validando el sistema operativo descargado

Para poder acceder a nuestra raspberry existen distintas formas, sin embargo, la que nos beneficiaría a nosotros sería de forma inalámbrica en nuestra red local (LAN), por lo que en la siguiente imagen seleccionaremos “connect to Wireless network” .

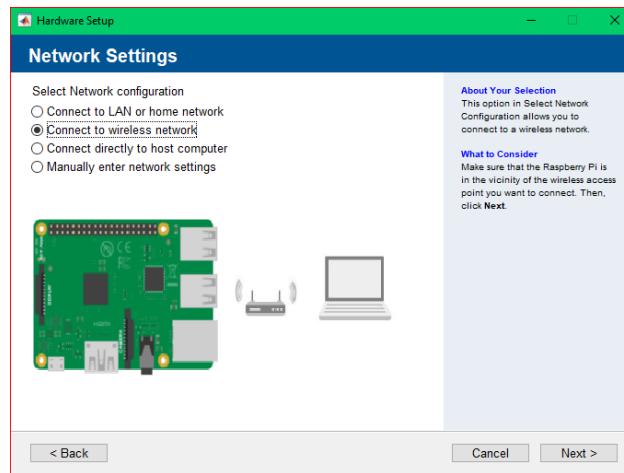


figura 145 Configuración del paquete de raspberry pi - selección la configuración de red.

Lo siguiente será configurar la red a la que se conectara la raspberry pi y la contraseña de dicha red, se accederá a la raspberry con VNC viewer por lo que se le configuro una dirección IP estática, no hay muchos dispositivos conectados a la red privada con la que se hicieron la pruebas por lo cual la última dirección que se agregan a esta red seria la 13.0.157.110 así que se dejó margen grande para que no existan problemas con direcciones IP iguales, por ejemplo si un dispositivo llega a la dirección 13.0.157.110 y la raspberry tiene esa dirección pero no está conectada, el dispositivo tiene preferencia al adquirir esa IP antes , y cuando la raspberry se conecte existirán problemas con la IP, por lo cual se utilizó la IP: 13.0.157.141 dejando un margen amplio para que otros dispositivos se puedan conectar a la red privada.

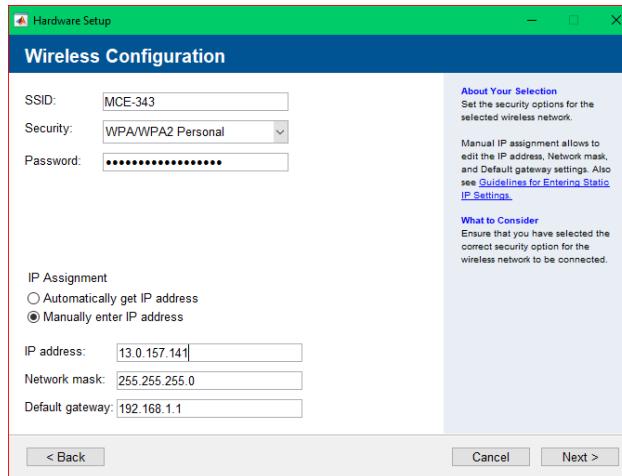


figura 146 Configuración del paquete de raspberry pi – configuración de nuestra red privada.

Luego se pide insertar la SD a la computadora para que Matlab pueda instalar el sistema operativo en la SD.



figura 147 Configuración del paquete de raspberry pi – insertar nuestra tarjeta SD

En la siguiente ventana se presiona el botón WRITE para que Matlab empiece a grabar en sistema operativo en la SD y le se otorgan permisos para que pueda acceder a ella, para esto esperamos unos minutos.

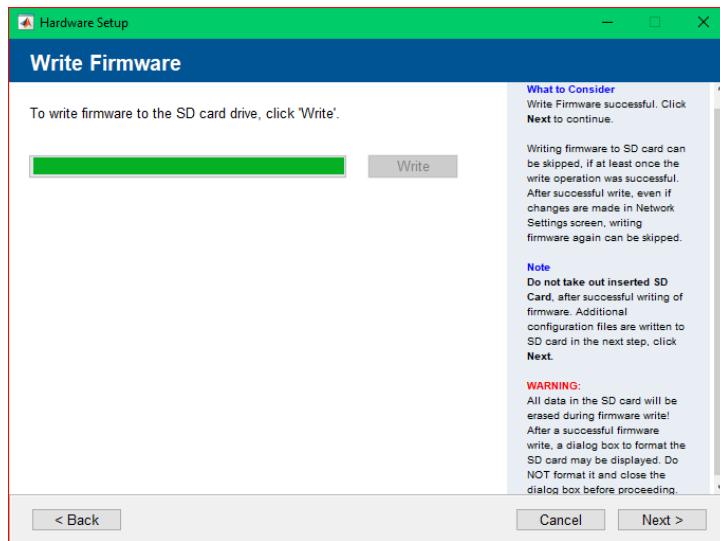


figura 148 Configuración del paquete de raspberry pi – instalando el sistema operativo.

Al terminar la instalación, se retira la SD e introduce a nuestra raspberry pi y se conecta a la fuente de alimentación.

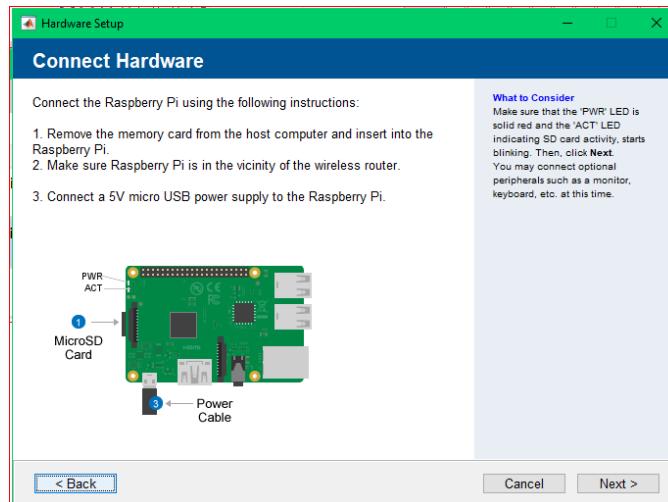


figura 149 Configuración del paquete de raspberry pi – pasos finales de instalación.

Al presionar next el sistema empezara a buscar la raspberry , es importante que esté conectada a internet , puede conectarse con el cable ethernet o conectarse a un monitor para asegurar que el wifi este encendido ya que por defecto lo trae apagado y si no está conectada a la red Matlab no la detectara.

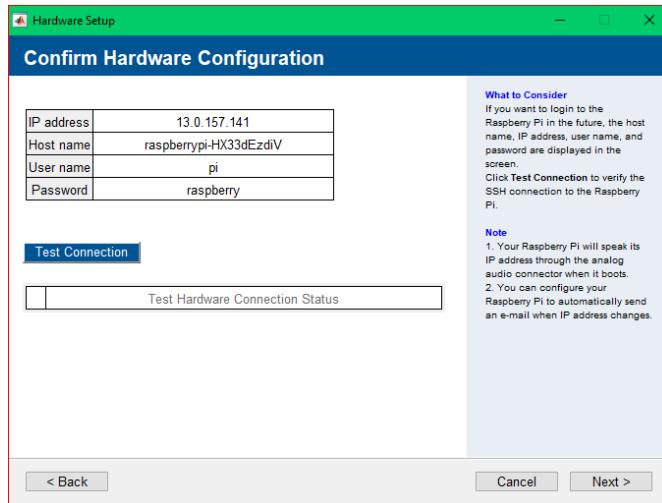


figura 150 Configuración del paquete de raspberry pi – raspberry pi detectada por Matlab.

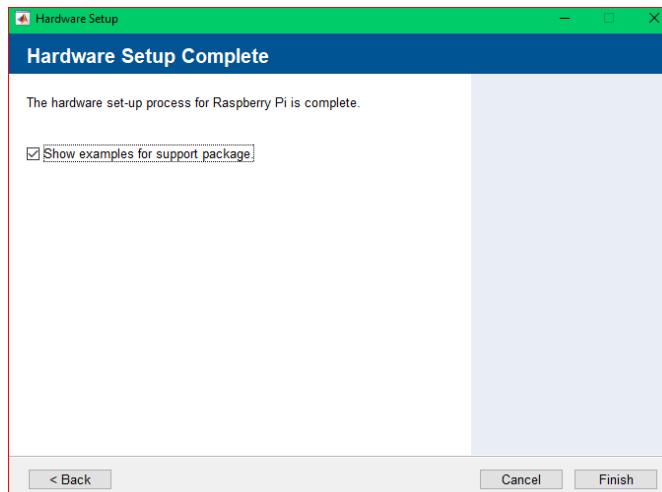
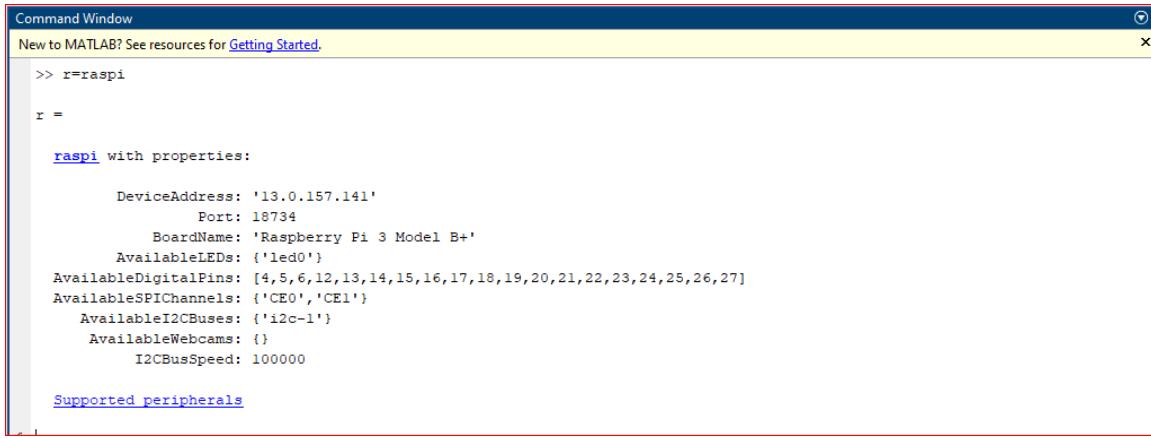


figura 151 Configuración del paquete de raspberry pi – configuración completa.

Finalmente, al presionar “Finish” la instalación junto a la configuración habrá terminado, Matlab mandará una pestaña emergente que contendrá la documentación de la raspberry, ahora para hacer una prueba de conexión hay que dirigirse a la terminal de Matlab y ejecutar el siguiente comando “r = raspi” con lo que devolverá lo siguiente:



The screenshot shows the MATLAB Command Window with the title 'Command Window' at the top. Below it, a yellow bar says 'New to MATLAB? See resources for [Getting Started](#)'. The main window displays the following code and its output:

```
>> r=raspi
r =
raspi with properties:
    DeviceAddress: '13.0.157.141'
        Port: 18734
    BoardName: 'Raspberry Pi 3 Model B+'
    AvailableLEDs: {'led0'}
    AvailableDigital Pins: [4,5,6,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27]
    AvailableSPIChannels: {'CE0','CE1'}
    AvailableI2CBuses: {'i2c-1'}
    AvailableWebcams: {}
    I2C Bus Speed: 100000
Supported peripherals
```

figura 152 propiedades de la conexión con la raspberry pi

Así pues , la conexión fue un éxito y devuelve las propiedades junto a los periféricos a los que se puede acceder de forma remota, con esto se finalizada la conexión Raspberry pi -Matlab.

5.4 Entrenamiento de una red neuronal artificial con Matlab

El tema a desarrollar es sobre el tipo de red neuronal artificial a usar en el prototipo robótico, por lo que se tiene un tema muy amplio en el cual existen distintas alternativas para el mismo fin, por ello, iniciar con una recopilación bibliográfica sobre las redes neuronales y sobre proyectos donde se implementaron redes neuronales artificiales es la opción más lógica, con esto se inicia el primer paso importante para el entrenamiento de la red neuronal artificial.

Inicialmente como se describe en los apartados 5.2.3 y 5.2.4 la lógica del código estaba diseñado para recolectar 27 datos en total por lo cual el robot solo daba 3 giros, de este modo se hacían las capturas de los datos a 7.5° entre capturas.

Ahora bien, puede surgir distintas preguntas como:

- ¿Cuál es la diferencia a la programación tradicional?
- ¿por qué usar redes neuronales artificiales para el prototipo?

En primer lugar como vemos en la figura 5.43, la programación tradicional va enfocada a que el desarrollador le indique las condiciones o reglas que debe seguir dependiendo de los datos de entrada, con esto puede generar la respuesta correspondiente a estas reglas, en cambio el machine learning el cual es un subconjunto de la inteligencia artificial va enfocado a que el desarrollador le de entrenamiento indicándole los datos de entrada y las respuestas correspondiente a esos datos y mediante formulación matemática dar una respuesta con cierto margen de aciertos.

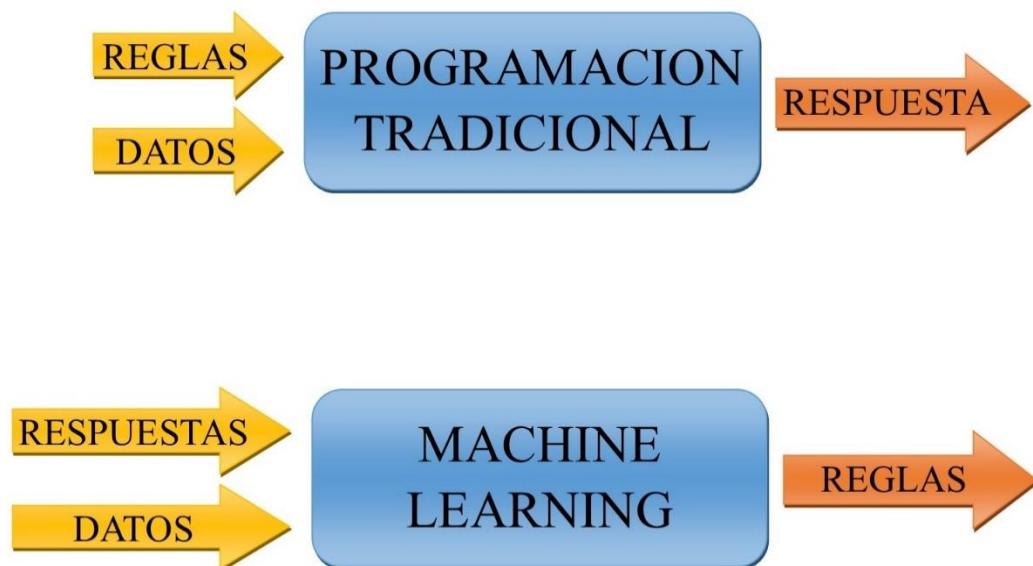


figura 153 Programación tradicional vs machine learning

Ahora bien, el ¿Por qué usarlo en el prototipo? Es una pregunta sobrante, se describen las ventajas que nos ofrece las redes neuronales en los apartados 2.7.1, 2.7.2 y 2.7.3. con esto queda claro que el utilizar redes neuronales artificiales hará que el prototipo pueda reconocer distintos objetos independientemente del tamaño teniendo en cuenta las reglas que el modelo de la red neuronal artificial entrenada retorne.

Para iniciar con el entrenamiento de la red neuronal, se necesitan los datos con los cuales se hará el entrenamiento, los objetos elegidos para entrenar la red fueron: un bote de pintura en aerosol para utilizar su figura cilíndrica como referencia, una caja por su figura cuadrada y un balón ya que su forma esférica dará otra opción con respecto al cilindro.

Al tener una base de datos con las capturas de las distancias, se continuo con la programación en Matlab, en el que se usaron funciones definidas para los entrenamientos de redes neuronales artificiales del toolbox de Deep learning para entrenar una red neuronal que solo se enfocara a resolver una tarea específica.

Como se menciona en el apartado 2.7.2 y se observa en la figura 5.44 el Deep learning es un subconjunto del machine learning por lo cual fue el enfoque elegido para el prototipo.

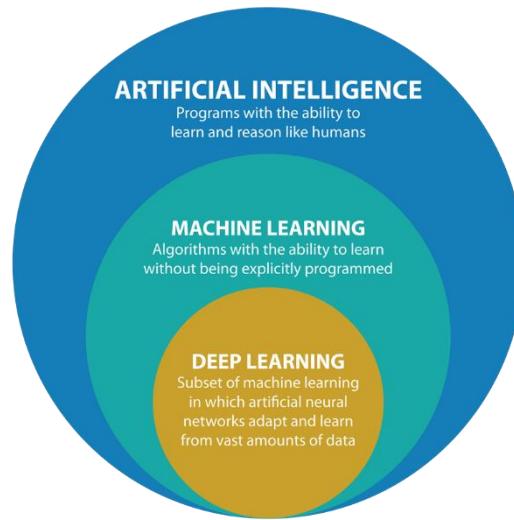


figura 154 inteligencia artificial.

Se inicio agregando los datos recogidos antes guardados en las tablas de Excel, para ello se crearon variables en Matlab con el nombre de los objetos que se evaluaron y dentro de ellas están las distancias recolectadas anteriormente definiéndolos como “los datos” de entrada. Con estos datos ahora se crearon targets de salida o “las respuestas”, los cuales sirvieron en el entrenamiento para enseñarle al sistema el tipo de salida que se necesita con respecto a los datos de entrada.

Para esta primera etapa el total de datos recogidos fueron los siguientes:

- Para el cilindro se capturaron 393 capturas en distintas posiciones.
- Para la caja fueron 222 capturas en distintas posiciones.
- Para el balón fueron 276 capturas para distintas posiciones.

Al iniciar el entrenamiento de la red neuronal artificial se introdujeron los datos recaudados en cada captura, en esta primera etapa serian 27 datos de entrada, sin embargo, los datos actuales están en una posición horizontal por lo que se tienen que posicionar de tal forma que cada una de las 27 muestras que se obtuvo por lectura sea una entrada (visualizar la figura 5.45).

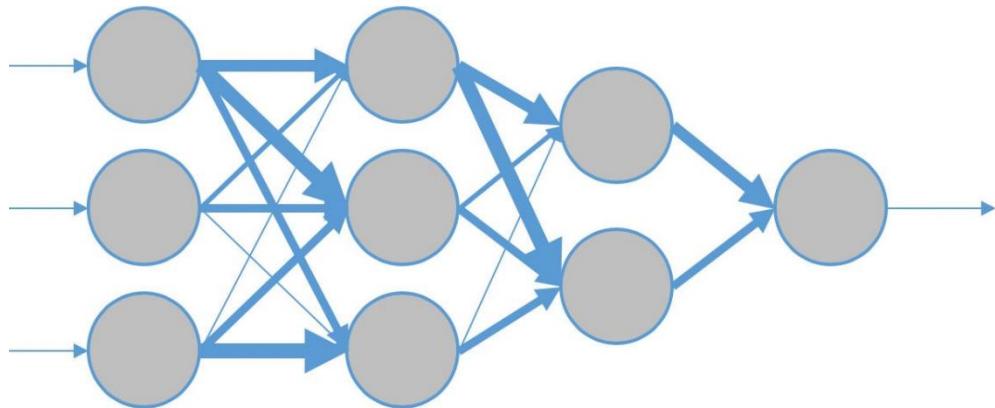


figura 155 modelo de una red neuronal artificial.

Por otro lado, se iniciaron los preparativos para la configuración del toolbox de Matlab con la que se hizo el entrenamiento de la red neuronal, se utilizó el algoritmo de Backpropagation el cual es uno de los algoritmos más famosos para entrenar una red neuronal de retroalimentación, permite actualizar los pesos moviéndose hacia adelante y hacia atrás hasta que la función de error se atasque en su mínimo local. En forma de resumen se presentarán los resultados mejor destacados en las pruebas de entrenamiento.

Para hacer la configuración de la primera red neuronal se utilizó la función :

- `patternnet(hiddenSizes,trainFcn,performFcn)`

Esta función devuelve una red neuronal de reconocimiento de patrones con un tamaño de capa oculta de “hiddenSizes”, una función de entrenamiento, especificada por “trainFcn”, y una función de rendimiento, especificada por “performFcn”. Las redes de reconocimiento de patrones son redes de retroalimentación que se pueden entrenar para clasificar las entradas de acuerdo con las clases objetivo, para el entrenamiento de las redes se utilizó el código A.6 del anexo códigos y para la corroboración con datos nuevos se utilizó el código 6.7 del mismo anexo.

Prueba 1

Para la primera prueba la configuración de la red quedo configurada con la siguiente línea de código:

- `red=patternnet(12,'trainlm');`

Especificamos que la primera red está hecha de una capa oculta con 12 neuronas y con la función de entrenamiento “trainlm la cual es una función de entrenamiento de red que actualiza los valores de peso y sesgo de acuerdo con la optimización de Levenberg-Marquardt. Trainlm es a menudo el algoritmo de retro propagación más rápido en la caja de herramientas y es muy recomendable como un algoritmo supervisado de primera elección, aunque requiere más memoria que otros algoritmos” [41].

También se configuro las épocas a realizar, los mínimos locales posibles, el error máximo permitido, el factor de aprendizaje el cual sirve para modificar los pesos iniciales, el factor de aprendizaje decreciente, y el factor de aprendizaje creciente.

- `red.trainParam.epochs=(1000);` % número de épocas máximas
- `red.trainParam.max_fail=100;` % verifica mínimos locales posibles
- `red.trainParam.min_grad=1e-29;` %error máximo permitido
- `red.trainParam.mu=0.1;` %factor de aprendizaje para modificar los pesos iniciales
- `red.trainParam.mu_dec=0.22;` % factor de aprendizaje decreciente
- `red.trainParam.mu_inc=6;` %factor de aprendizaje creciente

luego, se utilizó la función `configure()`, en la cual se agregó la configuración de la red, las entradas y las salidas.

- `configure(red,input,targets);`

ahora bien, se seleccionó que porcentaje de los datos totales se usarían para el entrenamiento, validación y test.

- `red.divideParam.trainRatio=94/100;`
- `red.divideParam.valRatio=3/100;`
- `red.divideParam.testRatio=3/100;`

Para la primera prueba se secciono 94% para entrenar, 3% para validar y 3% para testear.

Finalmente se hizo el entrenamiento de la red:

- [red,tr]=train(red,input,targets);

Como resultado de este primer entrenamiento se obtuvo una red del 98% de aciertos en Matlab , sin embargo, al hacer nuevas pruebas con datos nuevos, se obtuvieron diferencias con lo que Matlab proporcionaba y con lo que se obtenía en tiempo real.

DETECTADO EL BALON												
	1 CAPTURA	2 CAPTURA	3 CAPTURA		1 captura	2 captura	3 captura		1 captura	2 captura	3 captura	
5 cm	5	6	6	balones	1			cilindros	1	1	1	cuadrados
10 cm	5	4	4	balones	1	2	2	cilindros	1	1	1	cuadrados
15 cm	4	3	4	balones	2	3	2	cilindros	1	1	1	cuadrados
20 cm	4	4	4	balones	4	3	3	cilindros				cuadrados
25 cm	3	3	3	balones	4	4	4	cilindros				cuadrados
30 cm	4	5	4	balones	2	1	2	cilindros	1	1	1	cuadrados
totales:	25	25	25	cilindros	14	13	13	cuadrados	4	4	4	cuadrados
total balones:	75	59.05511811 %		total cilindros:	40	31.496063 %		total cuadrados	12	9.4488189 %		
total de capturas:	127	100 %										

figura 156 Capturando datos nuevos de un balon

DETECTADO CUADRADOS												
	1 CAPTURA	2 CAPTURA	3 CAPTURA		1 captura	2 captura	3 captura		1 captura	2 captura	3 captura	
5 cm	1	2	3	balones	1	1	1	cilindros	3	4	2	cuadrados
10 cm	4	4	3	balones	2	2	2	cilindros	1	2	2	cuadrados
15 cm	0	0	0	balones	1	1	2	cilindros	6	6	5	cuadrados
20 cm	1	1	1	balones	1	1	1	cilindros	5	5	5	cuadrados
25 cm	0	0	0	balones	1	1	1	cilindros	6	6	6	cuadrados
30 cm	0	0	0	balones	2	1	1	cilindros	4	5	5	cuadrados
totales:	6	7	7	cilindros	8	7	8	cuadrados	25	28	25	cuadrados
total balones:	20	16.52892562 %		total cilindros:	23	19.0082645 %		total cuadrados	78	64.4628099 %		
total de capturas:	121	100 %										

figura 157 Capturando datos nuevos de un cilindro

DETECTADO EL CILINDRO												
	1 CAPTURA	2 CAPTURA	3 CAPTURA		1 captura	2 captura	3 captura		1 captura	2 captura	3 captura	
5 cm	3	3	3	balones	5	5	5	cilindros	0	0	0	cuadrados
10 cm	1	1	1	balones	6	6	6	cilindros	0	0	0	cuadrados
15 cm	3	3	3	balones	4	4	4	cilindros	0	0	1	cuadrados
20 cm	2	2	2	balones	5	3	4	cilindros	0	0	0	cuadrados
25 cm	2	2	3	balones	4	4	4	cilindros	1	1	1	cuadrados
30 cm	0	0	0	balones	5	5	5	cilindros	2	2	1	cuadrados
totales:	11	11	12	cilindros	29	27	28	cuadrados	3	3	3	cuadrados
total balones:	34	26.77165354 %		total cilindros:	84	66.1417323 %		total cuadrados	9	7.08661417 %		
total de capturas:	127	100 %										

figura 158 Capturando datos nuevos de una caja

Como se observa en las figuras 5.46, 5.47 y 5.48 , el margen de aciertos es mucho menor al que Matlab ofrece, cuando se capturan nuevos datos para balón se obtiene un acierto del 59.05%, para el cilindro un 66.14% y para el cuadrado un 64.46% de acierto, para lo cual esta primera red no cumpliría con un porcentaje aceptable, así que se implementó mejoras al entrenamiento de la red, la primera fue el aumentar la resolución de datos pasando de 27 a 45 datos por captura, la segunda mejora fue cambiar el número de capas ocultas así como también modificar el número de neuronas en sus capas y seguir cambiando los pesos iniciales junto a los incrementos y decrementos en los factores de aprendizaje.

Con estas nuevas modificaciones lo que cambiaría en el código para el entrenamiento serían los datos de entrada, y también se hizo un cambio con la figura del balón por una pirámide y se agregó a la salida la opción de “nada”, para que cuando no se detecte nada la red tenga esa opción en su entrenamiento.

Por lo cual los objetos a evaluar o más bien las opciones de salida con respecto a las entradas de la red serían: caja, cilindro, pirámide y nada. Para la configuración del tipo de entrenamiento y capas ocultas se hicieron algunos cambios más notorios para esta primera red con 45 datos, se utilizaron 5 capas ocultas con diferentes neuronas en cada una de ellas.

- red=patternnet([25,19,16,12,8], 'trainlm');
- ed.trainParam.epochs=(1000); % numero de epochas maximas
- red.trainParam.max_fail=100; %verifica minimos locales posibles
- red.trainParam.min_grad=1e-29; %error maximo permitido
- red.trainParam.mu=0.1; %factor de aprendisaje para modificar los pesos iniciales
- red.trainParam.mu_dec=0.2;% factor de aprendizaje decreciente
- red.trainParam.mu_inc=10; %factor de aprendizaje creciente

También se cambiaron los porcentajes para entrenamiento, validación y test.

- configure(red,input,targets);
- red.divideParam.trainRatio=90/100;
- red.divideParam.valRatio=5/100;
- red.divideParam.testRatio=5/100;
- [red,tr]=train(red,input,targets);

Como resultado se obtuvo una red neuronal del 98.8% según lo arrojado por Matlab, al pasarlos con los nuevos datos recolectados se obtuvo con esa red que: PIRAMIDE = 32.2033898% de acierto, CILINDRO = 58.6206897 % de aciertos, CAJA = 55.9322034% de aciertos y NADA=100% de aciertos.

Prueba 2

Se procedió a realizar una segunda prueba , en esta prueba se dejaron los parámetros anteriores igual, solo se modificó las neuronas en las capas:

- red=patternnet([25,18,12,6,4], 'trainlm');

así se obtuvo una red de 96.6% según lo arrojado por Matlab, al pasarlo con los mismos datos con la que se evaluó la anterior red se obtuvieron los siguientes resultados: PIRAMIDE 37.2881356% de aciertos, CILINDRO= 56.8965517% de aciertos, CAJA= 64.4067797% de aciertos y NADA=100% de aciertos en total del 64.64% de acierto.

Se continuo con una segunda fase de entrenamiento, en esta segunda fase solo se destacan dos entrenamientos ya que fueron los que mejores resultados presentaron, estos fueron la prueba 3.1 y la prueba 4.1.

Prueba 2.3

Se configuran los parámetros de entrenamiento para la etapa dos de la siguiente forma:

- `red=patternnet([25,19,16,12,8], 'trainlm');` % el número de neuronas en la capa oculta y el método de entrenamiento
- `red.trainParam.epochs=(1000);` % número de épocas máximas
- `red.trainParam.max_fail=100;` % verifica mínimos locales posibles
- `red.trainParam.min_grad=1e-29;` % error máximo permitido
- `red.trainParam.mu=0.1;` % factor de aprendizaje para modificar los pesos iniciales
- `red.trainParam.mu_dec=0.2;` % factor de aprendizaje decreciente
- `red.trainParam.mu_inc=10;` % factor de aprendizaje creciente
- `configure(red,input,targets);`
- `red.divideParam.trainRatio=90/100;`
- `red.divideParam.valRatio=5/100;`
- `red.divideParam.testRatio=5/100;`
- `[red,tr]=train(red,input,targets);`

Se obtuvo una red de 99.7% según Matlab, al pasarlo con los mismos datos con la que se evaluó la anterior red se obtuvieron los siguientes resultados: PIRAMIDE 40.6779661% de aciertos, CILINDRO= 59.3220339% de aciertos, CAJA= 77.9661017% de aciertos y NADA=100% de aciertos haciendo un promedio de 69.48% de acierto.

Prueba 2.4

En esta prueba se dejaron los parámetros anteriores igual, solo modificamos las neuronas en las capas:

- `red=patternnet([29,24,19,14,11],'trainlm');`

En esta prueba se dejaron todos los parámetros anteriores igual, y se obtuvo una red de 99.3% según Matlab, al pasarlo con los mismos datos con la que se evaluó la anterior red se obtuvieron los siguientes resultados: PIRAMIDE 22.0338983% de aciertos, CILINDRO= 71.9298246% de aciertos, CAJA= 84.7457627% de aciertos y NADA=100% de acierto haciendo un promedio de 69.67% de acierto.

El desarrollo de los continuos entrenamientos para obtener una red neuronal que satisfaga los estándares planteados inicialmente iba decayendo en cuanto a porcentaje de aciertos reales, por lo que se tuvo que replantear las funciones de entrenamiento que Matlab ofrece y del mismo modo las salidas de nuestra red neuronal artificial. Anteriormente se utilizaron 4 salidas para el entrenamiento de la red neuronal artificial, las cuales eran: PIRAMIDE, CAJA, CILINDRO Y NADA. La función que se utilizó para el entrenamiento fue TRAINLM.

En modo de experimentación se cambió las salidas, al notar que la salida de PIRAMIDE no mostraba mejoras significativas en las pruebas reales y siempre se mantenía por debajo del 38% de aciertos, se cambió esta salida por la salida BALON que al igual que la salida CILINDRO se refiere a un objeto circular, contando con que el sistema solo captura el entorno 2D, se planeaba manejar esa salida como una diferente debido al tamaño superior a la de CILINDRO.

A primera instancia se decidió dejar la misma función de entrenamiento para ver los resultados reales comparados a los anteriores y ver si el promedio de acierto de las 4 salidas mejoraba con respecto a sus antecesoras redes, de igual manera solo se presentan los entrenamientos con mejor porcentaje de acierto.

Prueba 3.1

Se recopilaron datos del balón que fueron utilizados para el entrenamiento, de igual forma se capturaron otros más para las pruebas finales, para la primera red de esta tercera etapa se inicializó con los parámetros que aparecen en la figura 5.49.

```
red=patternnet([29,21,17,15,12], 'trainlm'); % el numero de neuronas en la capa oculta y el metodo de
red.trainParam.epochs=(1000); % numero de epochas maximas
red.trainParam.max_fail=100; %verifica minimos locales posibles
red.trainParam.min_grad=1e-29; %error maximo permitido
red.trainParam.mu=0.1; %factor de aprendisaje para modificar los pesos iniciales
red.trainParam.mu_dec=0.2;% factor de aprendizaje decreciente
red.trainParam.mu_inc=10; %factor de aprendizaje creciente
%red.layers(3).transferFcn='tansig';%cambiar la fision de activacion de las neuronas de la capa uno
%red.layers(2).transferFcn='tansig';%cambiar la fision de activacion de las neuronas de la capa uno
configure(red,input,targets);
red.divideParam.trainRatio=90/100;
red.divideParam.valRatio=5/100;
red.divideParam.testRatio=5/100;
[red,tr]=train(red,input,targets);
```

figura 159 Parámetros utilizados para el primer entrenamiento de la tercera etapa

Se inicio con una red de cinco capas ocultas, en su primera capa con 29 neuronas en la segunda 21 neuronas, en la tercera 17 neuronas, en la cuarta 15 neuronas y en la quinta 12 neuronas y 4 neuronas en su capa de salida, se sigue utilizando la función TRAINLM, los parámetros de épocas, mínimos locales, error máximos permitido y el factor de aprendizaje se dejaron iguales que en los otros entrenamientos , en esta ocasión el factor de aprendizaje decreciente quedo en 0.2 y el factor de aprendizaje creciente en 10, el porcentaje de datos utilizados para el entrenamiento será del 90% del total de los datos a manejar y un 5% para la validación del entrenamiento y otro 5% para el test final.

Matlab arroja un resultado completo del entrenamiento de la red neuronal, también entre esos resultados arroja una ventana con las matrices de confusión en las cuales expresa cuántos datos de cada entrada corresponde a su salida que debería de tener y también cuántos datos se fueron a otras salidas, arroja un porcentaje de acierto en cada matriz, la de entrenamiento, la de validación y la matriz de test, de todas estas arroja una matriz global con el porcentaje final y para esta red seria del 99.4% de aciertos.

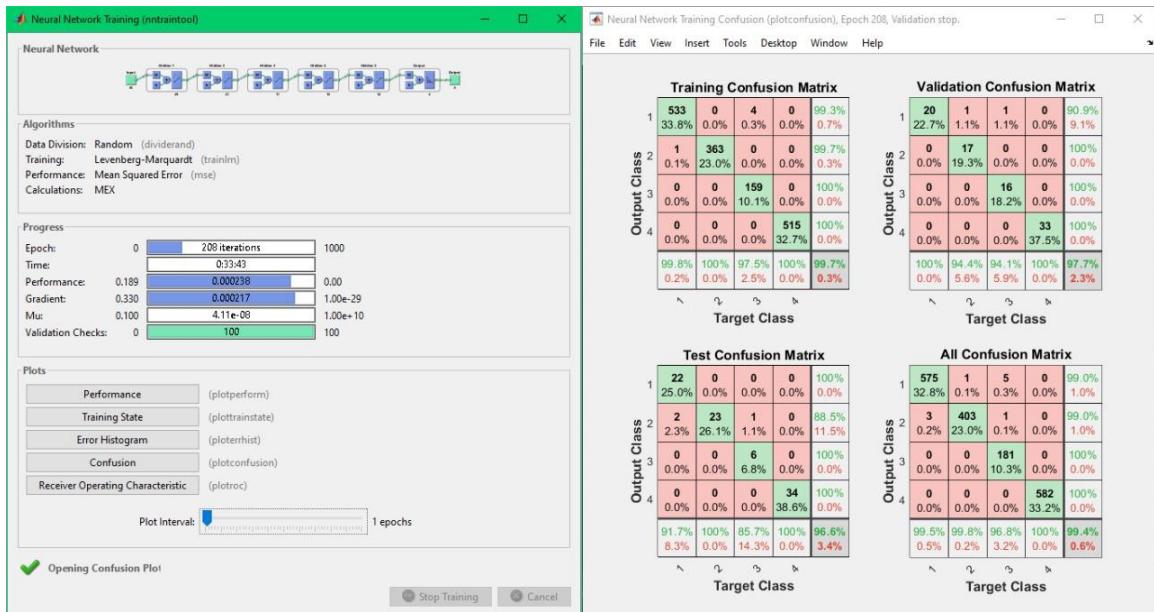


figura 160 Entrenamiento de la red 3.1 y matrices de confusión

Con estos datos, se extrajo la red entrenada y se guardó con un pseudónimo que permite identificarla para poder utilizarla, luego se procede a probar en un código con datos nuevos no utilizados en el entrenamiento de la red, y así saber exactamente qué porcentajes de aciertos tendría al momento de implementarla en el autómata.

Por lo cual se recolectaron 59 datos nuevos de cada objeto evaluado en el entrenamiento, después de cargar la red que se entrenó y ejecutar el código en el cual permite medir el grado de aciertos, se genera un archivo Excel con los datos de aciertos finales como en las siguientes figuras.



figura 161 Evaluación de la figura “balon”

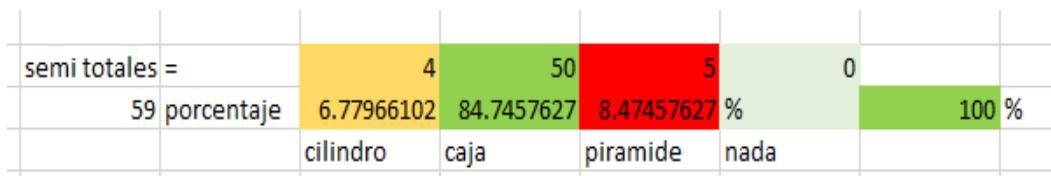


figura 162 Evaluación de la figura “caja”



figura 163 Evaluación de la figura “cilindro”

Como se observa en las figuras 5.51, 5.52 y 5.53 se tienen distintos porcentajes de aciertos en cada clasificación: para CAJA un acierto de 84.74%, para CILINDRO un acierto de 61.01% , para BALON un acierto de 33.89% y para NADA un 100% al promediar nos da una red con un 69.91% de acierto. Como vemos al agregar BALON como salida, la red sufre confusión con la salida de CILINDRO por lo que los aciertos para la salida de BALON es mucho menor o igual a la salida de PIRAMIDE la cual se eliminó para aumentar el porcentaje real de aciertos.

Teniendo lo anterior en cuenta se continuó experimentando con distintos parámetros para así determinar que mejoras sufre la red.

Prueba 3.5

Para la siguiente prueba se utilizaron los parámetros expresados en la figura 5.54.

```
red=patternnet([36,22,19,15,12], 'trainlm'); % el numero de neuronas en la capa oculta y el metodo de entrenamiento
red.trainParam.epochs=(1000); % numero de epochas maximas
red.trainParam.max_fail=100; %verifica minimos locales posibles
red.trainParam.min_grad=1e-29; %error maximo permitido
red.trainParam.mu=0.1; %factor de aprendizaje para modificar los pesos iniciales
red.trainParam.mu_dec=0.2;% factor de aprendizaje decreciente
red.trainParam.mu_inc=6; %factor de aprendizaje creciente
%red.layers(3).transferFcn='tansig';%cambiar la fision de activacion de las neuronas de la capa uno
%red.layers(2).transferFcn='tansig';%cambiar la fision de activacion de las neuronas de la capa uno
configure(red,input,targets);
red.divideParam.trainRatio=90/100;
red.divideParam.valRatio=5/100;
red.divideParam.testRatio=5/100;
[red,tr]=train(red,input,targets);
```

figura 164 Parámetros utilizados para el quinto entrenamiento de la tercera etapa

En esta red solo se modificó la cantidad de neuronas en una de las capas ocultas de la red neuronal artificial, se dejaron los mismos parámetros que la red anterior.

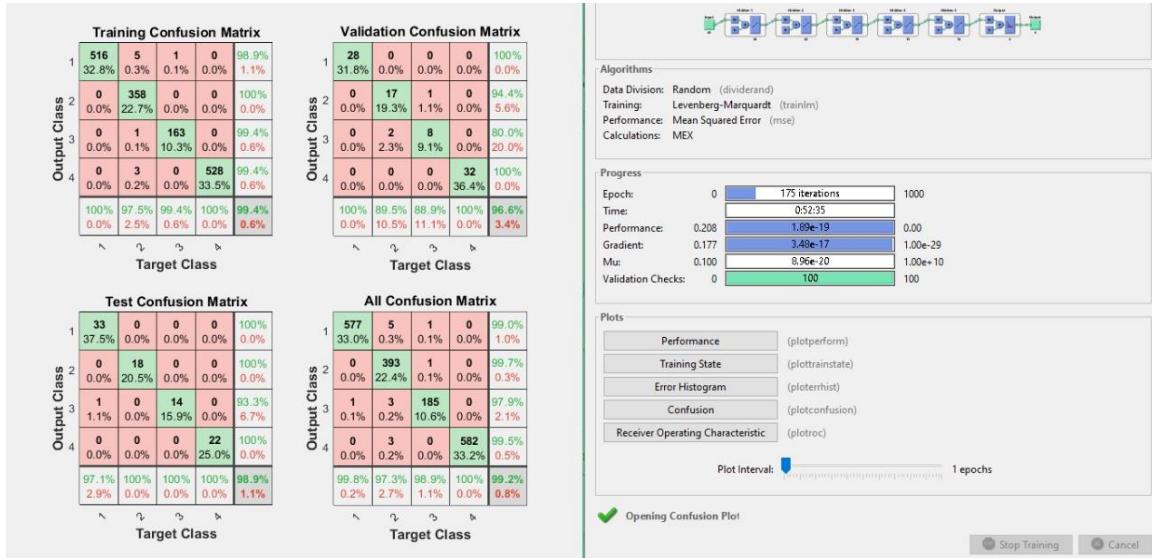


figura 165 Entrenamiento de la red 3.5 y matrices de confusión

Como se puede observar, Matlab en la matriz de confusión global da un acierto de 99.2%, se procede a guardar la red y a cargarla en el programa de prueba de aciertos reales. Al recabar los datos que arroja el código de prueba para las redes entrenadas obtenemos que para: CAJA 79.66% de acierto, CILINDRO = 67.79% de acierto, BALON=32.2% de acierto y para NADA=100% de aciertos, por lo que da un promedio final de 69.91% de acierto, lo cual es ALGO MAYOR que la red anterior y está a la par que la prueba 3.1.

Prueba 3.6

Para la siguiente prueba se utilizaron los parámetros expresados en la figura 5.56.

```
red=patternnet([45,39,32,22,19,15,12], 'trainscg'); % el numero de neuronas en la capa oculta y el metodo de aprendizaje
red.trainParam.epochs=(1000);% numero de epochas maximas
red.trainParam.max_fail=100; %verifica minimos locales posibles
red.trainParam.min_grad=1e-29; %error maximo permitido
red.trainParam.mu=0.1; %factor de aprendisaje para modificar los pesos iniciales
red.trainParam.mu_dec=0.2;% factor de aprendizaje decreciente
red.trainParam.mu_inc=6; %factor de aprendizaje creciente
%red.layers(3).transferFcn='tansig';%cambiar la fision de activacion de las neuronas de la capa uno
%red.layers(2).transferFcn='tansig';%cambiar la fision de activacion de las neuronas de la capa uno
configure(red,input,targets);
red.divideParam.trainRatio=90/100;
red.divideParam.valRatio=5/100;
red.divideParam.testRatio=5/100;
[red,tr]=train(red,input,targets);
```

figura 166 Parámetros utilizados para el sexto entrenamiento de la tercera etapa

Para el entrenamiento de esta red se modificó la cantidad de neuronas de las capas ocultas, al igual se agregaron más capas ocultas a esta red, también se cambió la función de entrenamiento a esta red, anteriormente se utilizaba la función *trainlm* y ahora se prueba la función *trainscg*. En la prueba 1 de esta sección mencionó en qué consistía la función *trainlm*, ahora la función *trainscg* “es una función de entrenamiento de red que actualiza los valores de ponderación y sesgo de acuerdo con el método de gradiente conjugado escalado”[42].

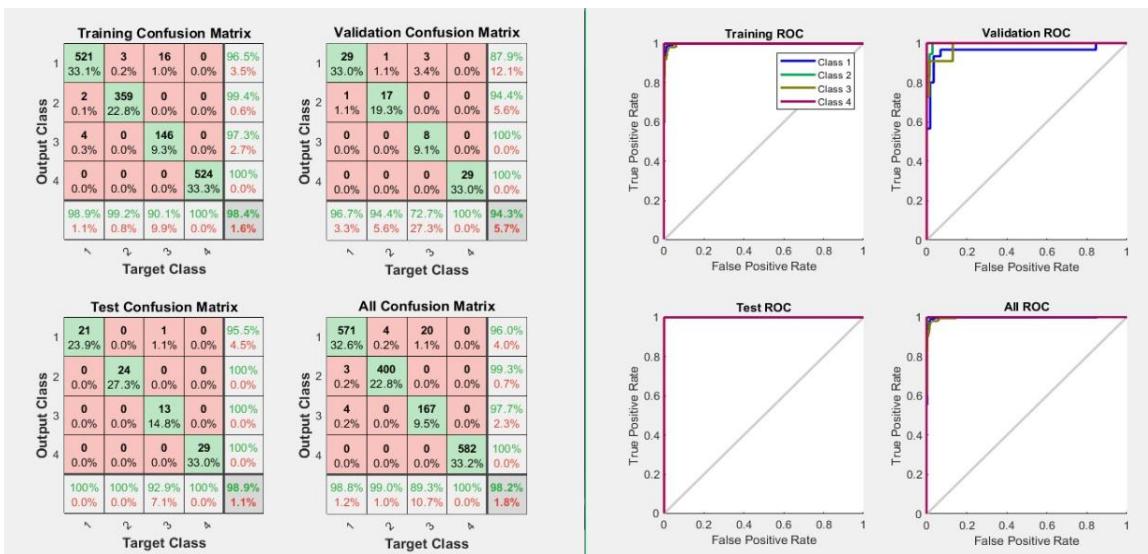


figura 167 Resultados del entrenamiento del sexto entrenamiento de la tercera etapa (A)

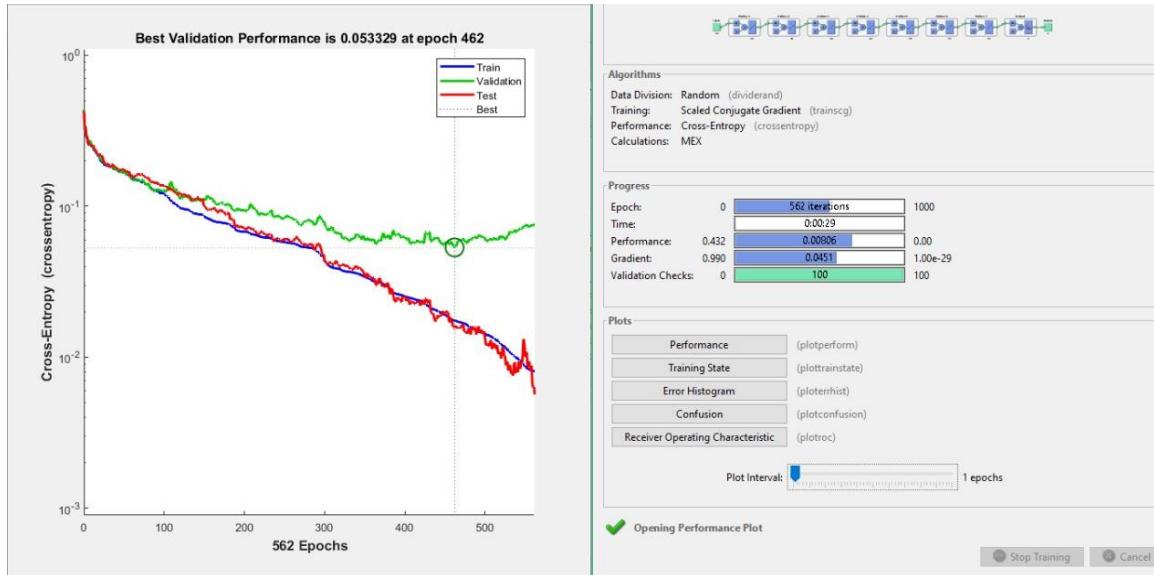


figura 168 Resultados del entrenamiento del sexto entrenamiento de la tercera etapa (B)

Como se puede observar, Matlab en la matriz de confusión global da un acierto de 98.2%, se procede a guardar la red y a cargarla en el programa de prueba de aciertos reales.

Al recabar los datos que arroja el código de prueba para las redes entrenadas obtenemos que para: CAJA 88.13% de acierto, CILINDRO = 57.62% de acierto, BALON=24.13% de acierto y para NADA=100% de aciertos, por lo que da un promedio final de 67.47% de acierto, lo cual es ALGO MENOR que la red anterior.

Prueba 3.7

Para la siguiente prueba se utilizaron los parámetros expresados en la figura 5.59.

```
red=patternnet([36,22,19,15,12], 'trainbr'); % el numero de neuronas en la capa oculta y el metodo de entrenamiento
red.trainParam.epochs=(1000); % numero de epochas maximas
red.trainParam.max_fail=100; %verifica minimos locales posibles
red.trainParam.min_grad=1e-29; %error maximo permitido
red.trainParam.mu=0.1; %factor de aprendisaje para modificar los pesos iniciales
red.trainParam.mu_dec=0.2;% factor de aprendizaje decreciente
red.trainParam.mu_inc=6; %factor de aprendizaje creciente
%red.layers(3).transferFcn='tansig';%cambiar la fision de activacion de las neuronas de la capa uno
%red.layers(2).transferFcn='tansig';%cambiar la fision de activacion de las neuronas de la capa uno
configure(red,input,targets);
red.divideParam.trainRatio=90/100;
red.divideParam.valRatio=5/100;
red.divideParam.testRatio=5/100;
[red,tr]=train(red,input,targets);
```

figura 169 Parámetros utilizados para el séptimo entrenamiento de la tercera etapa

En esta red se volvió a modificar la cantidad de neuronas de las capas ocultas, también se quitaron algunas capas ocultas a la red ya que no había cambios al agregar más capas, también se cambió la función de entrenamiento a esta red, anteriormente se utilizaba la función *trainscg* y ahora se utilizó la función *trainbr*.

En la prueba anterior se mencionó en qué consistía la función *trainscg*, ahora “la función *trainbr* es una función de entrenamiento de red que actualiza los valores de peso y sesgo de acuerdo con la optimización de Levenberg -Marquardt. Minimiza una combinación de errores al cuadrado y pesos, y luego determina la combinación correcta para producir una red que generalice bien. El proceso se llama regularización bayesiana”[43].

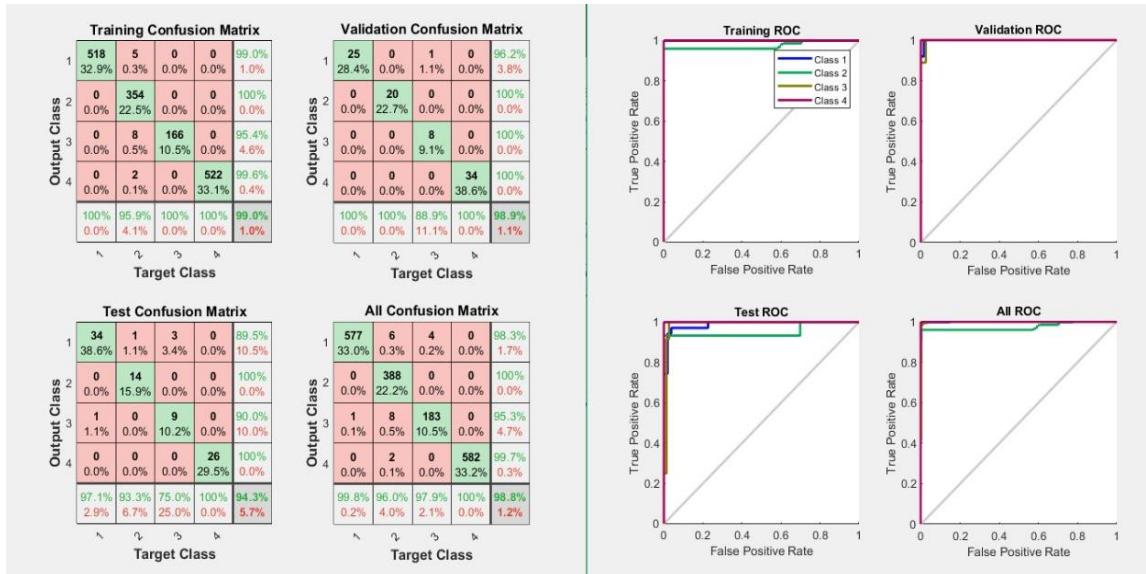


figura 170 Resultados del entrenamiento del séptimo entrenamiento de la tercera etapa (A)

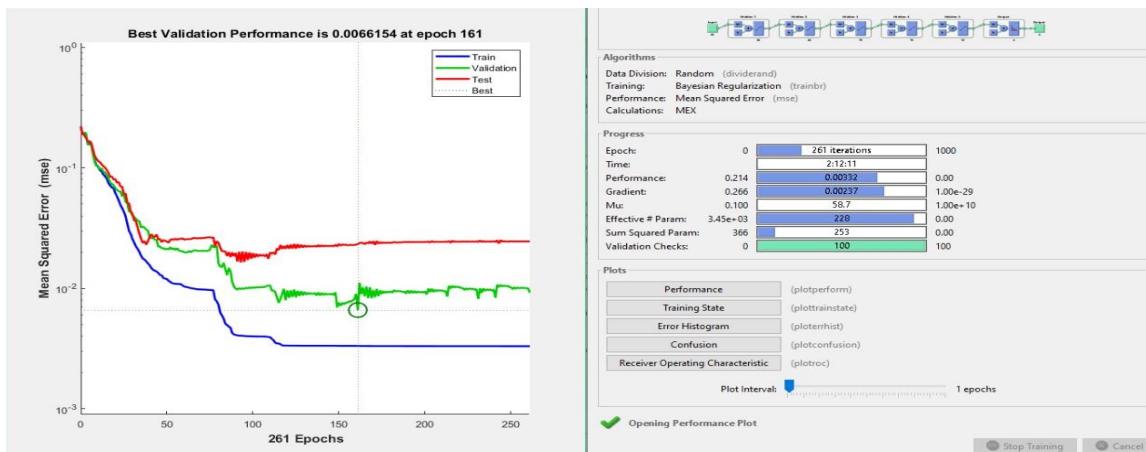


figura 171 Resultados del entrenamiento del séptimo entrenamiento de la tercera etapa (B)

Como se puede observar, Matlab en la matriz de confusión global da un acierto de 98.8%, se procede a guardar la red y a cargarla en el programa de prueba de aciertos reales. Al recabar los datos que arroja el código de prueba para las redes entrenadas obtenemos que para: CAJA 71.18% de acierto, CILINDRO = 71.18% de acierto, BALON=27.58% de acierto y para NADA=100% de aciertos, por lo que da un promedio final de 67.48% de acierto.

Después de obtener los anteriores resultados aun cuando cambio una de las salidas, se notó que la red neuronal no obtuvo mejoras significativas en cuanto a la configuración de salidas en donde se incluía la PIRAMIDE en lugar del BALON, esto se debió esencialmente a que los datos de entrada que recibía la red eran muy parecidos en cuanto rasgos de BALON y de CILINDRO por lo que hacía que los aciertos finales bajaran circunstancialmente.

Con esto en mente y al ver que aun cuando se aumentó la resolución de captura del sistema el entrenamientos no mejoraba, se decidió bajar la cantidad de salidas de la red, ya que el patrón de coincidencias de una CAJA y de un CILINDRO son diferentes , se decidió utilizar solo esas dos y por supuesto incluir una salida cuando no hubiera nada que detectar, con estos cambios se entrenó nuevas redes para un total de 24 nuevas redes entrenadas con diferentes funciones de entrenamiento, se fue iterando las tres funciones antes mencionadas : TRAINLM, TRAINBR, y TRAINSCG.

Se presentan solo los entrenamientos con mayor índice de aciertos.

Prueba 4.6

Para la siguiente prueba se utilizaron los parámetros expresados en la figura 5.62.

```

red=patternnet([64,55,55,50,48,49,39,45,38], 'trainscg'); % el numero de neuronas
red.trainParam.epochs=(1000); % numero de epochas maximas
red.trainParam.max_fail=100; %verifica minimos locales posibles
red.trainParam.min_grad=le-29; %error maximo permitido
red.trainParam.mu=0.1; %factor de aprendisaje para modificar los pesos iniciales
red.trainParam.mu_dec=0.2;% factor de aprendizaje decreciente
red.trainParam.mu_inc=6; %factor de aprendizaje creciente
configure(red,input,targets);
red.divideParam.trainRatio=90/100;
red.divideParam.valRatio=5/100;
red.divideParam.testRatio=5/100;
[red,tr]=train(red,input,targets);

```

figura 172 Parámetros utilizados para el sexto entrenamiento de la cuarta etapa

Para el entrenamiento se utilizó la función TRAINSCG y se agregaron 9 capas ocultas con distintas neuronas entre capas.

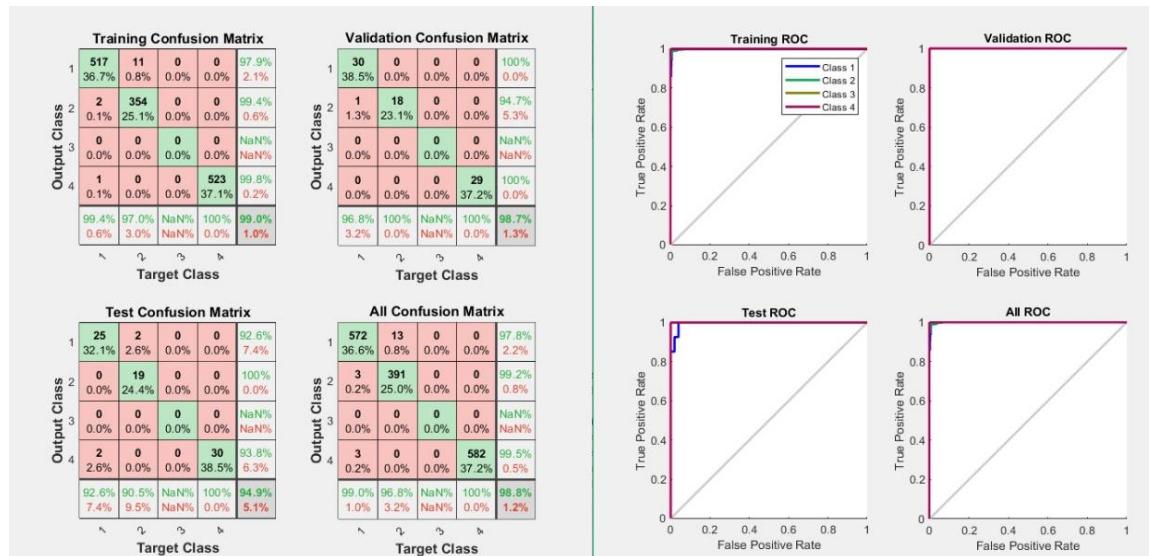


figura 173 Resultados del entrenamiento del sexto entrenamiento de la cuarta etapa (A)

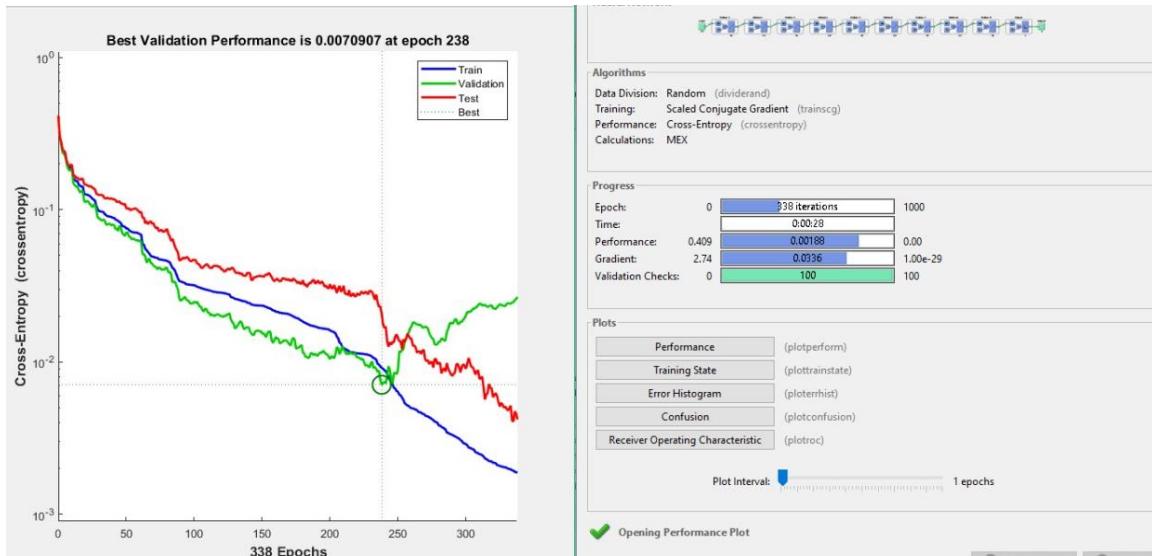


figura 174 Resultados del entrenamiento del sexto entrenamiento de la cuarta etapa (B)

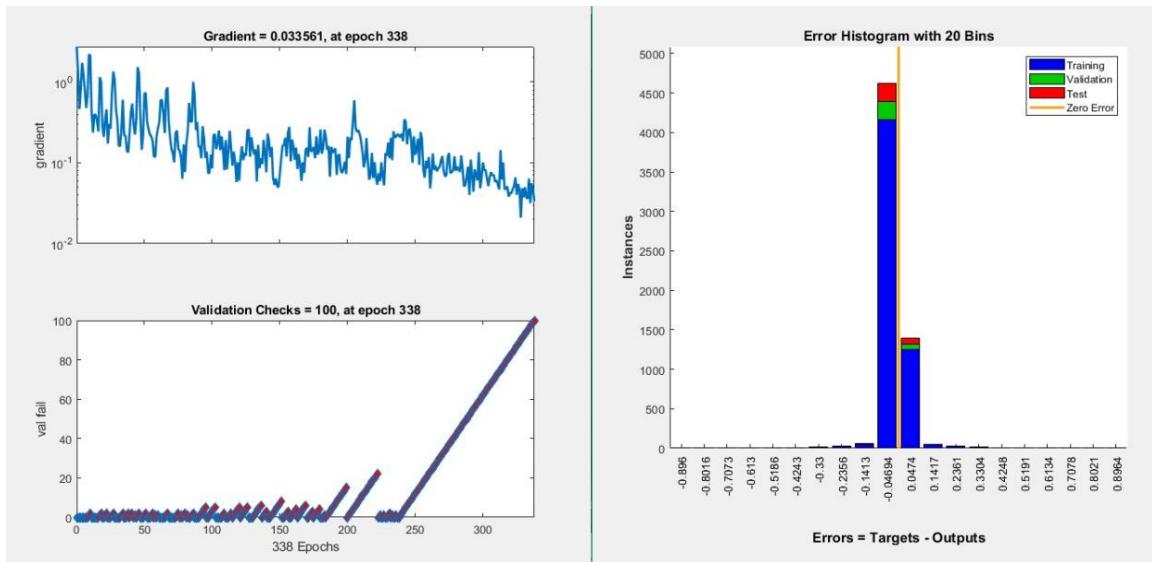


figura 175 Resultados del entrenamiento del sexto entrenamiento de la cuarta etapa (C)

Como se puede observar, Matlab en la matriz de confusión global da un acierto de 98.8%, se procede a guardar la red y a cargarla en el programa de prueba de aciertos reales. Al recabar los datos que arroja el código de prueba para las redes entrenadas obtenemos que para: CAJA 83.05% de acierto, CILINDRO = 94.91% de acierto, y para NADA=100% de aciertos, por lo que da un promedio final de 92.65% de acierto.

Se implementaron distintas configuraciones para las distintas funciones de entrenamiento, aun así, solo se obtuvo una red con un promedio arriba del 90%, sin embargo, se busca un acierto mayor, por lo que al revisar los parámetros se observó que había una diferencia circunstancial en los datos de entrada ya que había 578 vectores de CILINDRO, 403 de CAJA y 582 de NADA, por lo que se optó por nivelar los vectores de entrada, por lo que se recaudaron más datos para CILINDRO y para CAJA.

En total para el entrenamiento de esta nueva etapa se utilizaron 583 vectores de entrada con 45 datos por cada vector para la salida de CILINDRO y para CAJA se utilizaron 583 vectores de entrada con 45 datos por cada vector, para la salida de NADA se utilizaron 582 vectores de entrada con 45 datos por cada vector.

Se presentan solo los entrenamientos con mayor índice de aciertos.

Prueba 5.2

Para la siguiente prueba se utilizaron los parámetros expresados en la figura 5.66.

```

red=patternnet([29,25,39,25,18], 'trainlm'); % el numero de neuronas en la capa oculta
red.trainParam.epochs=(1000);% numero de epochas maximas
red.trainParam.max_fail=100; %verifica minimos locales posibles
red.trainParam.min_grad=le-29; %error maximo permitido
red.trainParam.mu=0.1; %factor de aprendisaje para modificar los pesos iniciales
red.trainParam.mu_dec=0.2;% factor de aprendizaje decreciente
red.trainParam.mu_inc=6; %factor de aprendizaje creciente
configure(red,input,targets);
red.divideParam.trainRatio=67/100;
red.divideParam.valRatio=18/100;
red.divideParam.testRatio=15/100;
[red,tr]=train(red,input,targets);

```

figura 176 Parámetros utilizados para el segundo entrenamiento de la quinta etapa

Para realizar este entrenamiento se utilizó la función TRAINLM

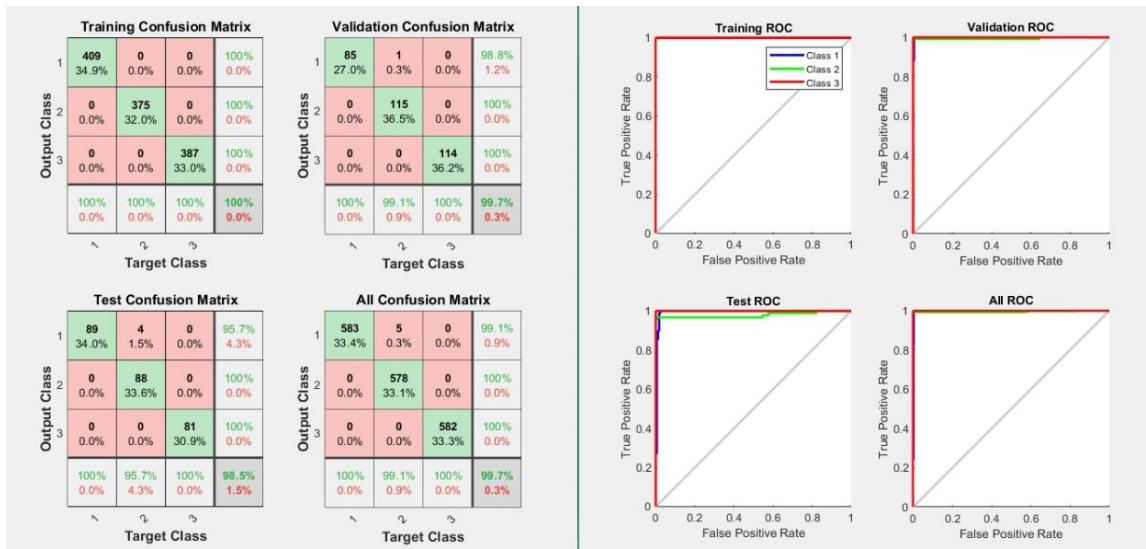


figura 177 Resultados del entrenamiento del segundo entrenamiento de la quinta etapa (A)

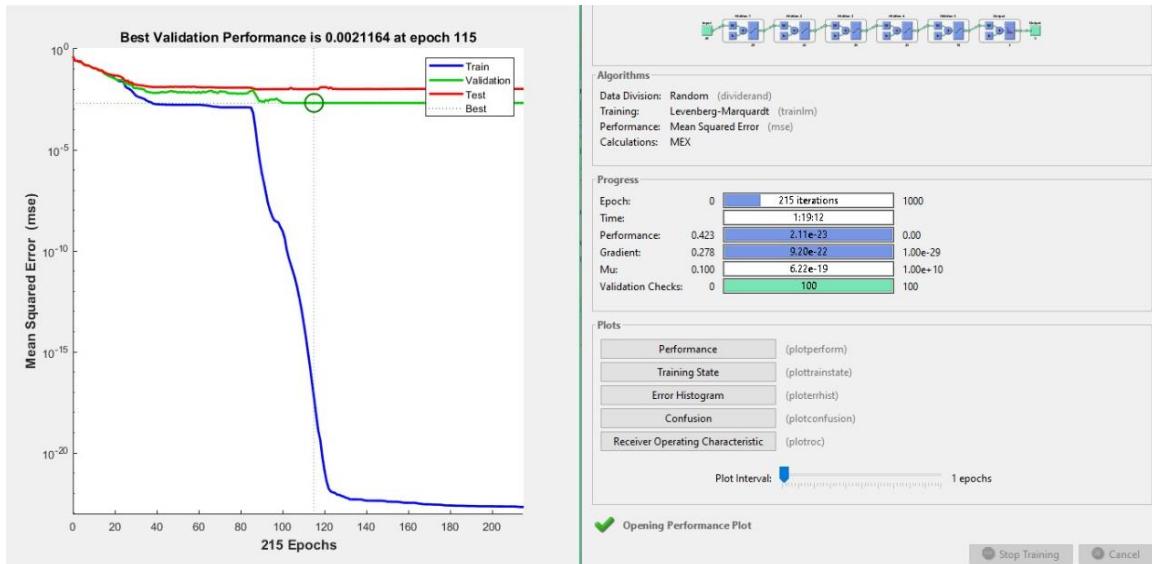


figura 178 Resultados del entrenamiento del segundo entrenamiento de la quinta etapa (B)

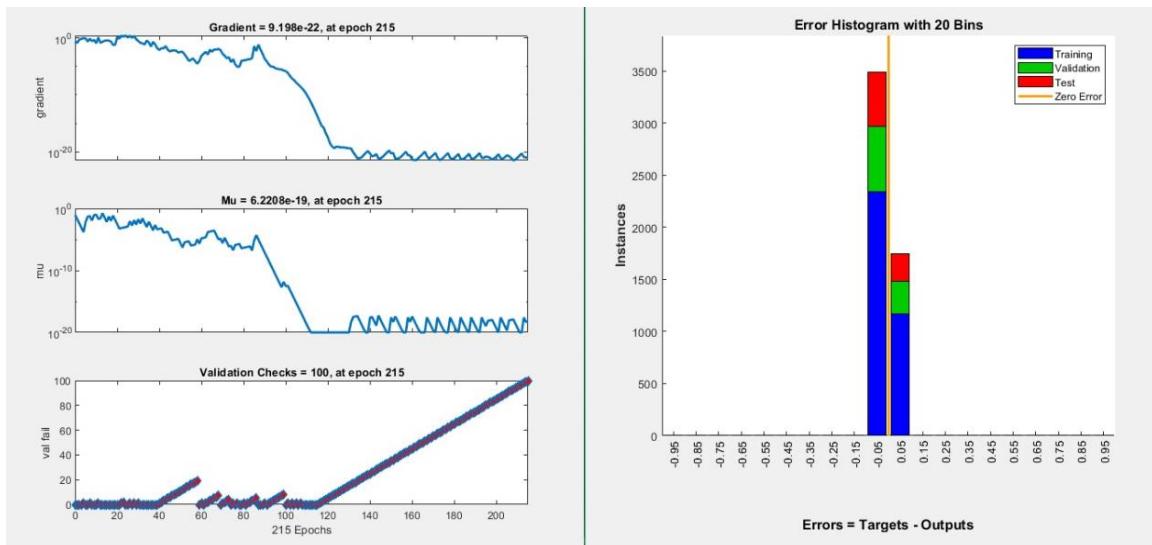


figura 179 Resultados del entrenamiento del segundo entrenamiento de la quinta etapa (C)

Como se puede observar, Matlab en la matriz de confusión global da un acierto de 99.7%, se procede a guardar la red y a cargarla en el programa de prueba de aciertos reales. Al recabar los datos que arroja el código de prueba para las redes entrenadas obtenemos que para: CAJA 93.22% de acierto, CILINDRO = 84.74% de acierto, y para NADA=100% de aciertos, por lo que da un promedio final de 92.65% de acierto.

Prueba 5.7

Para la siguiente prueba se utilizaron los parámetros expresados en la figura 5.70.

```
red=patternnet([45,29,65,25,19,12],'trainscg'); % el numero de neuronas en la capa de salida
red.trainParam.epochs=(1000);% numero de epochas maximas
red.trainParam.max_fail=100; %verifica minimos locales posibles
red.trainParam.min_grad=le-29; %error maximo permitido
red.trainParam.mu=0.1; %factor de aprendisaje para modificar los pesos iniciales
red.trainParam.mu_dec=0.22;% factor de aprendizaje decreciente
red.trainParam.mu_inc=6; %factor de aprendizaje creciente
configure(red,input,targets);
red.divideParam.trainRatio=68/100;
red.divideParam.valRatio=17/100;
red.divideParam.testRatio=15/100;
[red,tr]=train(red,input,targets);
```

figura 180 Parámetros utilizados para el séptima entrenamiento de la quinta etapa

Para realizar este entrenamiento se utilizó la función TRAINSCG y seis capas ocultas.

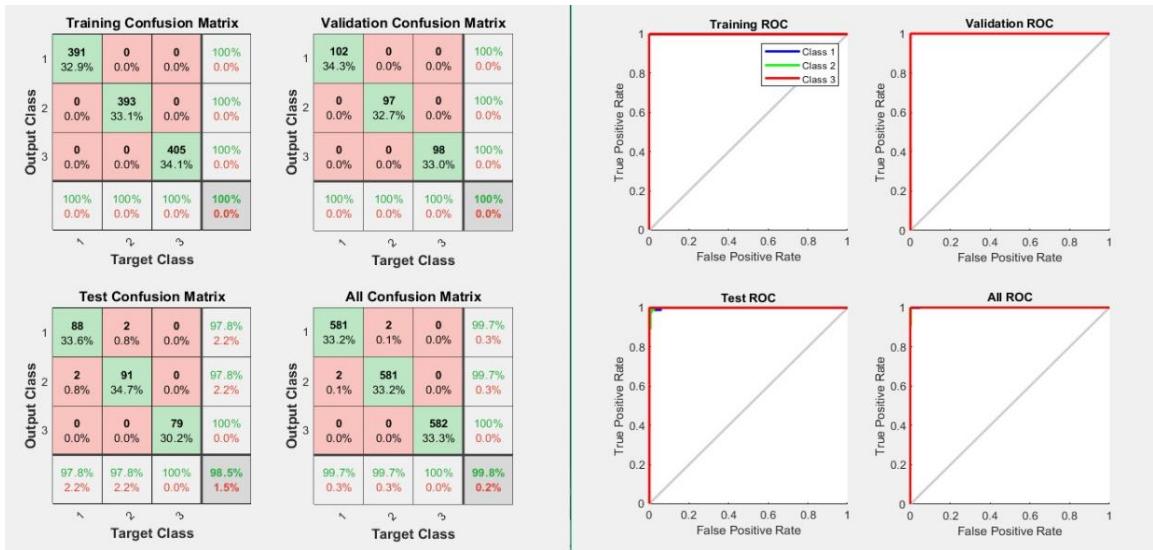


figura 181 Resultados del entrenamiento del séptimo entrenamiento de la quinta etapa (A)

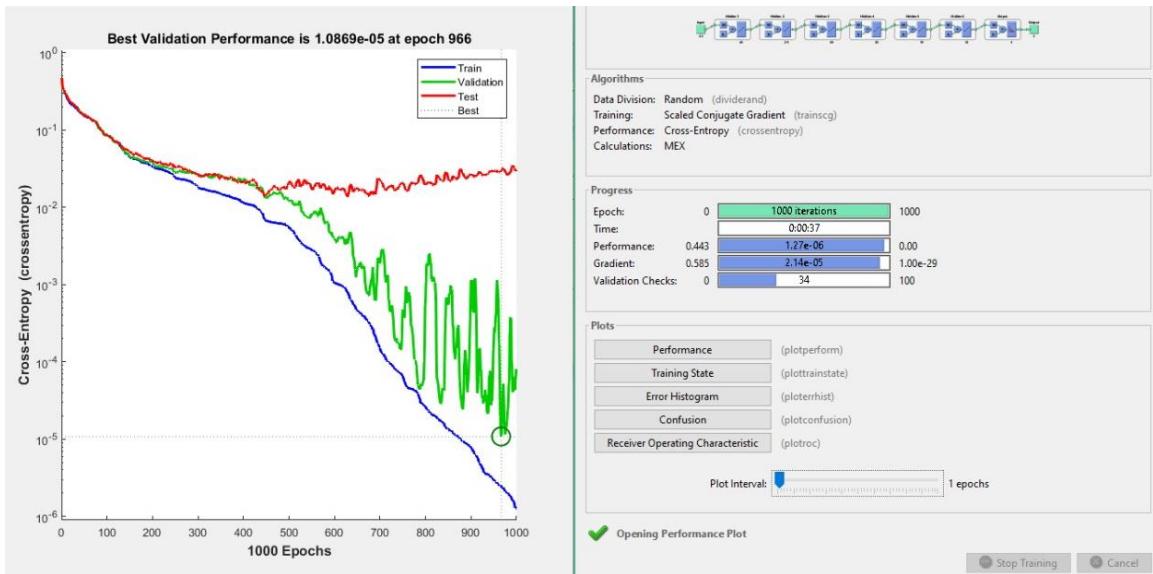


figura 182 Resultados del entrenamiento del séptimo entrenamiento de la quinta etapa (B)

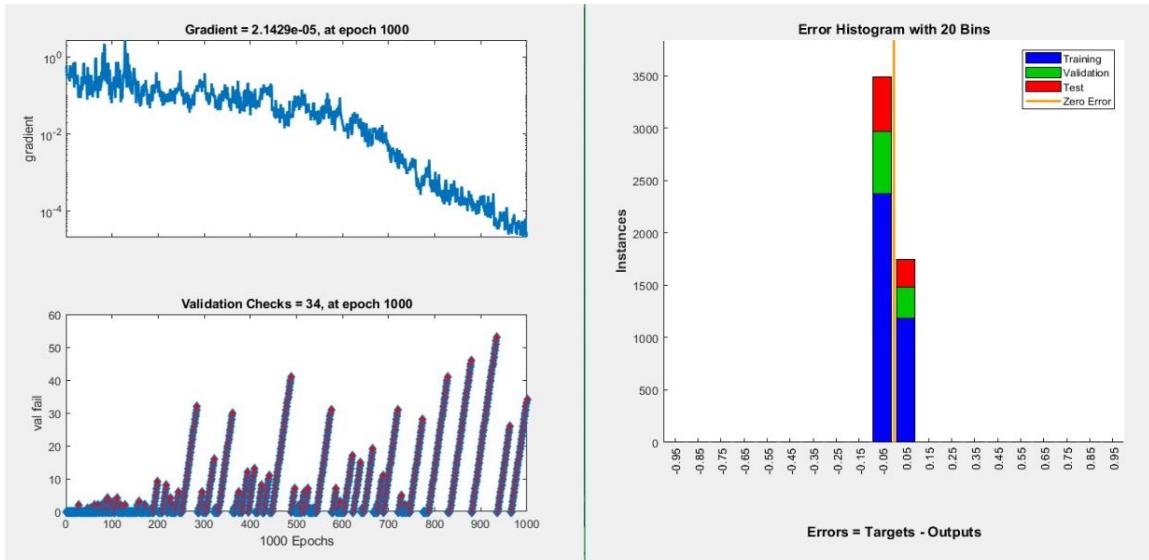


figura 183 Resultados del entrenamiento del séptimo entrenamiento de la quinta etapa (C)

Como se puede observar, Matlab en la matriz de confusión global da un acierto de 99.8%, se procede a guardar la red y a cargarla en el programa de prueba de aciertos reales. Al recabar los datos que arroja el código de prueba para las redes entrenadas obtenemos que para: CAJA 98.3% de acierto, CILINDRO = 91.52% de acierto, y para NADA=100% de aciertos, por lo que da un promedio final de 96.6% de acierto.

Al hacer un total de 41 pruebas se optó por elegir el entrenamiento de la prueba 5.7 el cual fue el que obtuvo mayores aciertos en las pruebas con nuevos datos no incluidos en el entrenamiento.

Para consultar las tablas de los porcentajes finales de todos los entrenamientos, consultar el Anexo B- tablas.

5.5 interfaz gráfica en Matlab

Para el desarrollo de la interfaz gráfica, se planteaba que el prototipo fuera un sistema plenamente embebido por lo que las primeras pruebas se enfocaron en exportar todos los códigos de Matlab a Python el cual es un lenguaje nativo para la raspberry pi, sin embargo, debido a problemas derivados de la exportación de la red neuronal artificial se tuvo que declinar a ejecutar el software de la red neurona en un pc con Matlab.

Así pues, Matlab ofrece alternativas para el desarrollo de interfaces gráficas, como se explica en la sección 2.7.6 Matlab genera un entorno interactivo para la creación de tales interfaces utilizando App Designer.

Para la interfaz se necesita establecer la comunicación con la raspberry pi 3b+, esta comunicación enviara los datos que hubiese en el puerto COM0 en el cual se ha conectado la placa Arduino mega 2560, de igual forma, se visualizaría el resultado de objeto escaneado que la red neuronal artificial arroje con los datos recaudados por los sensores y enviados a Matlab, también se graficarían las 45 distancias recolectadas para observar la forma que debería arrojar la respuesta de la red neuronal artificial entrenada, con estos objetivos en mente se desarrolló la interfaz gráfica que se observa en la figura 5.74.

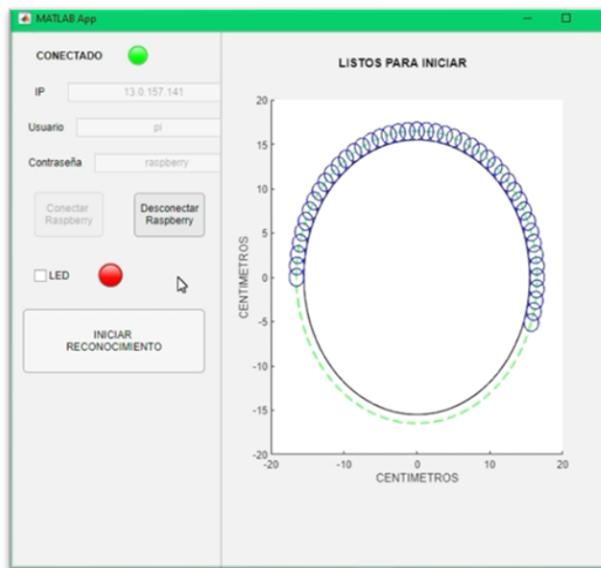


figura 184 Interfaz gráfica desarrollada en App Designer de Matlab para el prototipo robótico.

Como se observó en la imagen 5.74, inicialmente los *text box* para la *IP*, *Usuario* y *Contraseña* tienen un valor definido, esto es , porque en la sección 5.3 se explico que al comunicar la raspberry con Matlab se definió una IP estática a la raspberry y al hacer las configuraciones iniciales por defecto el usuario es *pi* y la contraseña *raspberry*, esto se puede configurar a disposición del usuario, sin embargo, para efectos del proyecto se decidió dejar tal cual ya que no afecta al desarrollo del mismo.

Para poder graficar las distancias de los objetos escaneados por los sensores, se tuvo que hacer uso de las funciones trigonométricas para poder posicionar un punto en la gráfica donde se plasmaría todas estas distancias, para ello se utilizó el código A.8 del anexo códigos.

Capítulo 6. RESULTADOS Y PRUEBAS FINALES

Al hacer las primeras pruebas de movimiento hubo percances que hicieron replantear el diseño, el peso del prototipo era mayor a lo que los motores vex 269 podían mover, y la fricción generada por las llantas era muy baja haciendo que existiera deslizamiento por parte del prototipo, de igual manera, luego de varias pruebas de giro para la recolección de datos se observó perdidas de algunos pasos, por lo que no se regresaba a la posición inicial.

Para corregir lo anterior se cambiaron las llantas de 2" por unas llantas más grande y con un grosor mayor utilizando llantas vex de 4", pasando de usar dos motores VEX EDR 269 a cuatro motores VEX EDR 393 los cuales son más grandes y con la suficiente fuerza para mover el prototipo, también se le implemento un final de carrera para que siempre se conserve la posición inicial y con ello corregir la perdida de pasos.

Para realizar los cambios de las ruedas y motores se tuvo que modificar de igual manera el chasis haciéndolo un poco más alto, ya que al utilizar motores más grandes y duplicar la cantidad de ellos, los espacios que se tenían anteriormente no eran los suficientes, al terminar el ensamblado podemos observar cómo quedó en la figura 6.1

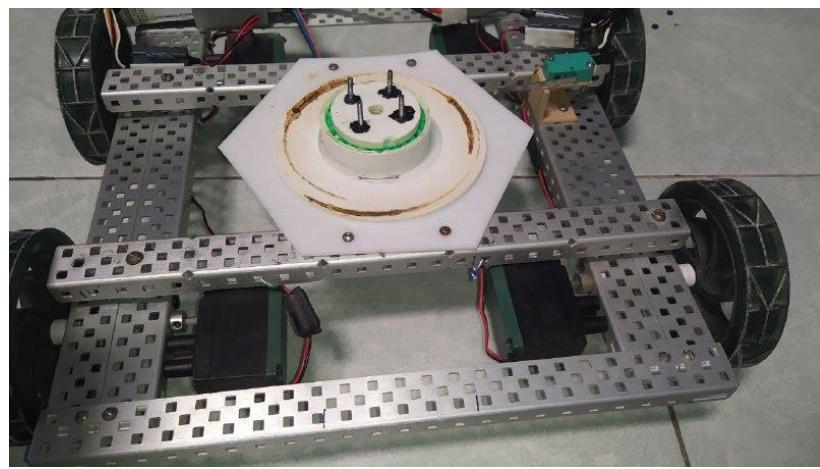


figura 185 Modificaciones al chasis del prototipo robótico.

Al tener las modificaciones listas sobre el chasis, se procedió a implementar la base giratoria sobre el chasis y se hicieron las primeras pruebas con la red neuronal ya implementada.

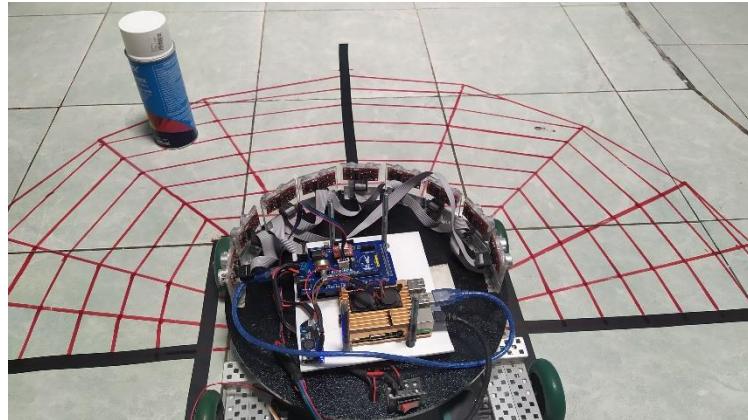


figura 186 primeras pruebas con la red neuronal artificial implementada en el prototipo robótico

Al tener resultados positivos con los primeros objetos, se decidió hacer un recorrido marcando cuatro obstáculos los cuales tendrían que ser identificados por el prototipo y graficados en la interfaz gráfica.

Para la lógica del recorrido como se explicó en la sección 5.2, la placa Arduino mega 2560 alojaría el código para la navegación, mediante la interfaz gráfica enviaremos el código de activación para el código del Arduino mega, esta leería los datos del mpu 6050 y mediante un control PID digital dirigiría al prototipo robótico, el prototipo avanzaría en línea recta hacia adelante y mediante los sensores ultrasónicos 4 , 5 y 6 vistos en la figura 5.11 de esa misma sección haría lecturas constantes de tal forma que cuando se topase con algún objeto dentro del rango de 40cm, este se detendría y procedería a hacer el escaneo del área de reconocimiento para así determinar la forma geométrica del objeto detectado, una vez hecho el escaneo las distancias serían enviadas a Matlab y pasarían por la red neuronal artificial y de igual forma se graficaría el objeto en cuestión, una vez hecho lo

anterior el prototipo gira 90° a la derecha en espera de recibir la instrucción desde la interfaz gráfica para continuar con su recorrido en línea recta.

Con esto se procede a iniciar el primer recorrido desde la interfaz gráfica y observar el recorrido del prototipo robótico, así como los datos enviados a la interfaz gráfica como se observa en la figura 6.3.

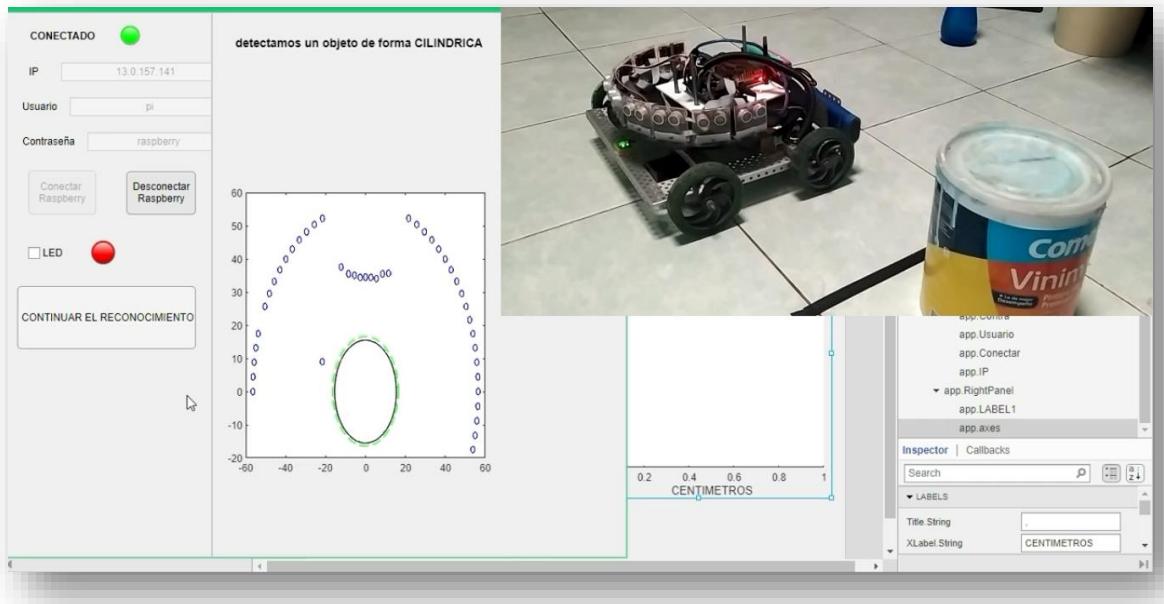


figura 187 Primer objeto detectado con la red neuronal artificial implementada en el prototipo robótico

Cómo se nota en la figura 6.2, el prototipo robótico ya ha terminado el escaneo del objeto y ha girado 90° correctamente, en este caso se ha topado con un bote de pintura de forma cilíndrica, en la interfaz gráfica el título ha cambiado , en la sección 5.5 donde se explicó la interfaz gráfica el título que aparecía era “LISTOS PARA INICIAR”, ahora ha cambiado y arroja la forma que tiene el objeto que se a escaneado, esto como resultado de la red neuronal artificial previamente entrenada, en la parte de abajo aparecen los puntos de las distancias obtenidas en el escáner, el circulo grande de la parte inferior hace

referencia a la base giratoria y los círculos pequeños son los puntos de las distancias obtenidas por cada posición de los 9 sensores obteniendo un total de 45 posiciones.

Para continuar con el recorrido, se presiona el botón “continuar el reconocimiento” y así el prototipo continuo en línea recta hasta detectar una caja y hacer el escaneo mandando los datos observados en la figura 6.3.

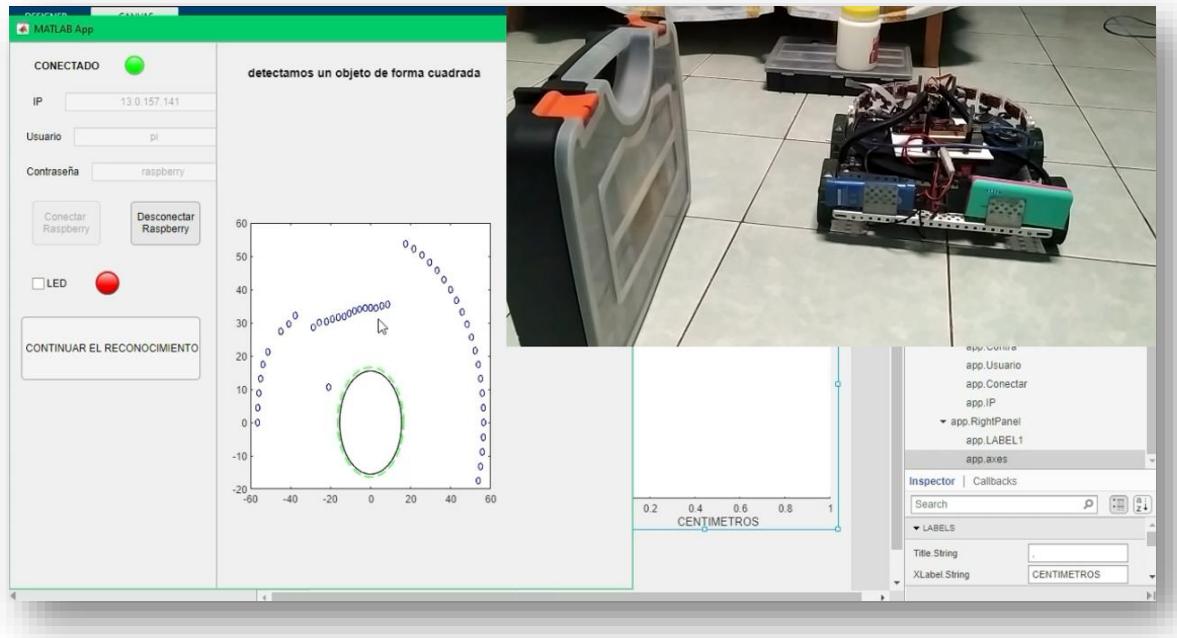


figura 188 Segundo objeto detectado con la red neuronal artificial implementada en el prototipo robótico

Los datos recibidos hacen que la red neuronal artificial nos arroje el mensaje “detectamos un objeto de forma cuadrada”, y así se grafica una línea con los puntos de las distancias recibidas, el prototipo robótico giro 90° y se volvió a presionar el botón para seguir con el reconocimiento, sin embargo, el prototipo robótico no avanzo en línea recta ya que el siguiente objeto se encontraban en su rango de escaneo el cual como se menciono es de 40cm, luego de hacer el reconocimiento envía los datos a Matlab y vuelve a girar 90°

, con los datos del escaneo obtenemos el resultado en la interfaz gráfica como se observa en la figura 6.3.

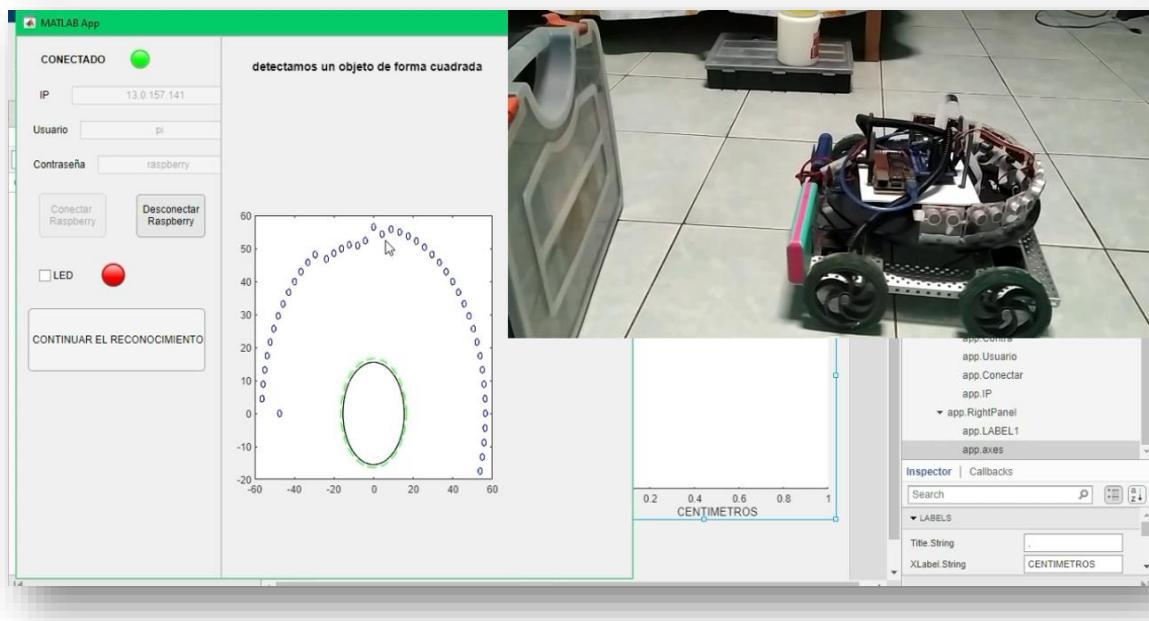


figura 189 Tercer objeto detectado con la red neuronal artificial implementada en el prototipo robótico

Los datos recibidos hacen que la red neuronal artificial arroje el mensaje “detectamos un objeto de forma cuadrada”, sin embargo, el objeto a evaluar es una frasco circular el cual esta sobre una caja cuadrada, surge una deficiencia de nuestra red neuronal esto se debe que como se explica en la sección 2.3.1, el sensor ultrasónico envía una honda en forma cónica la cual al rebotar en un objeto regresa al receptor del sensor y midiendo el tiempo de retorno de la honda se determina la distancia, esto se ve reflejado en la figura 6.4, sin embargo, como se menciona el frasco esta posicionado sobre una base cuadrada y al ver la gráfica se observa que los puntos graficados se encuentran casi en el límite de la distancia de captura, por lo cual al existir una distancia considerable hay una gran probabilidad que la honda ultrasónica enviada por el sensor rebote en algún extremo de la caja, y con esto la red neuronal artificial reconoce los patrones como el de un objeto de

forma cuadrada, así pues, podemos determinar que la deficiencia de la red neuronal artificial se debe a la calidad de precisión que nuestros sensores ultrasónico pueden obtener.

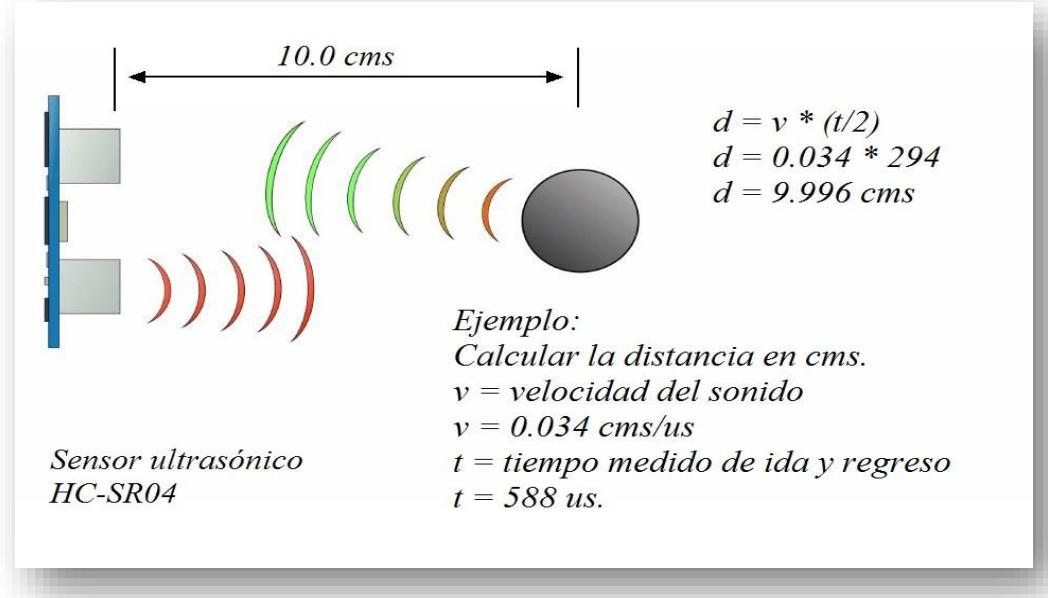


figura 190 Cálculo de la distancia mediante pulsos ultrasónicos

continuando con el recorrido del prototipo, se presiona una vez más el botón “continuar el reconocimiento” y el prototipo avanza en línea recta hasta detectar el ultimo objeto a evaluar, el cual es un bote de agua con características circulares uniformes, el prototipo procede hacer su rutina de recolección de datos y enviarlos a la interfaz grafica para el procesamiento de los mismos y así obtener los resultados vistos en la figura 6.5.

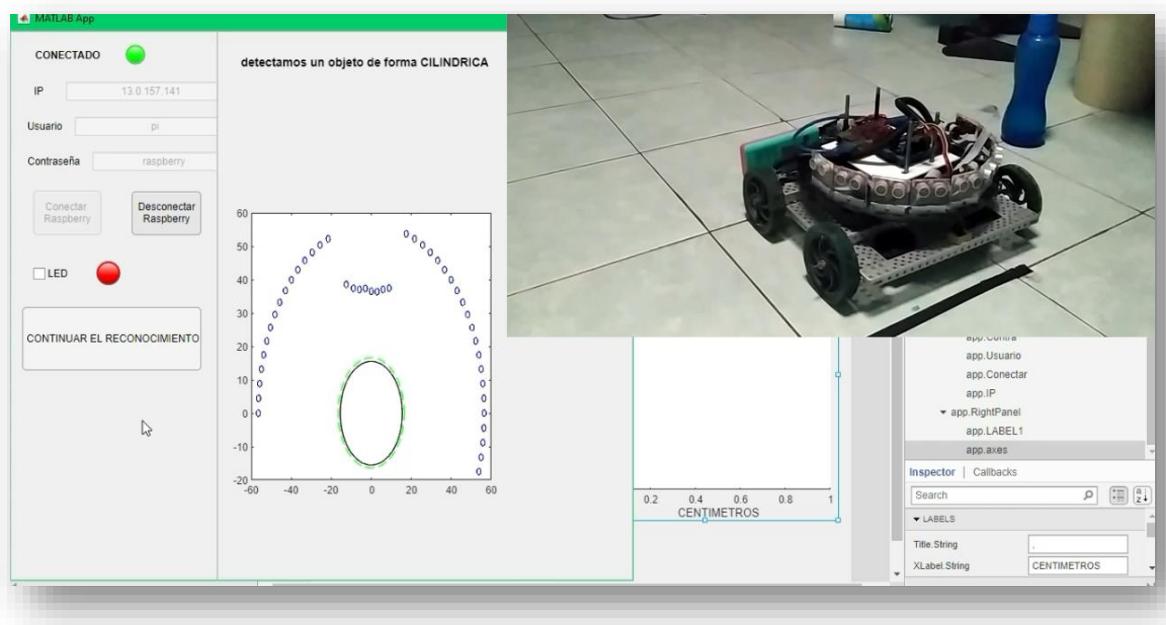


figura 191 Cuarto objeto detectado con la red neuronal artificial implementada en el prototipo robótico

Como se observa en la figura 6.5, la red neuronal artificial analiza correctamente los datos enviados por el prototipo robótico y arroja el mensaje “detectamos un objeto de forma cilíndrica”, con esto el prototipo robótico gira 90° y espera en la posición inicial.

CONCLUSIONES

Se diseño un prototipo robótico el cual mediante una base giratoria y nueve sensores ultrasónicos SFR08 pueden detectar objetos a su alrededor.

Se diseño e implemento una programación para la captura y almacenamiento de las distancias enviadas por los sensores ultrasónicos y guardados en archivos .xls para su postproceso.

Se diseño y entreno múltiples redes neuronales artificiales en el sistema de computo numérico MATLAB con los datos recaudados y guardados en los archivos .xls, con esto se elegio la red neuronal artificial con el mejor porcentaje de acierto y con ello se implemento dicha red al prototipo robótico.

Se diseño un sistema de navegación el cual mediante una IMU se obtiene la posición del prototipo robótico y mediante un controlador PID digital corregir la dirección del mismo en un ambiente controlado.

Se logro obtener una red neuronal artificial la cual Matlab le atribuye un acierto del 99.8% , sin embargo, mediante pruebas con datos nuevos no utilizados en el entrenamiento se obtuvo que tiene un porcentaje de acierto real del 96%.

El sistema a pesar de tener un excelente desempeño en la detección de obstáculos y el reconocimiento de patrones, tiene ciertos márgenes de mejora, ya que los sensores ultrasónicos al no generar una precisión excelente, se generan errores al momento de procesar los vectores de entrada para la red neuronal artificial.

RECOMENDACIONES

Se recomienda el utilizar otros tipos de sensores para el reconocimiento de los objetos, algunas de las opciones a utilizar podrían ser sensores laser, infrarrojos, ópticos o del tipo lidar.

Se recomienda implementar la red neuronal artificial junto a la interfaz grafica en el prototipo robótico de tal forma de generar un sistema embebido el cual no necesite de hardware adicional que haga el procesamiento de la red neuronal artificial, esto se puede lograr implementando el uso total de la raspberry y de una pantalla LCD la cual muestre la interfaz gráfica elaborada en Python.

Se recomienda implementas mas de una red neuronal al prototipo, o una red de mayor nivel que pueda enfocarse en diversas tareas, tanto para la clasificación de objetos como la de generación de mapas de entorno y de creación de rutas de navegación dependiendo del entorno al que sea expuesto.

REFERENCIAS

- [1] RIOS, A. J. (2018). “Reconstrucción de mapas para el recorrido de un vehículo sobre ruedas mediante un dispositivo de cómputo móvil”. Programación Matemática y Software.
- [2] Rodríguez, A., & Cortés, J. (2015). “Robot Oruga detector y esquivador de obstáculos con Fuzzy Logic IA”. Memorias.
- [3] García Saavedra, C. (2017). “Localización activa de robots móviles”.
- [4] Valencia, J. C. Y., & MESA, M. A. F. (2016). “PMITO-Plataforma móvil para investigación de técnicas de odometría” (Doctoral dissertation, Universidad Tecnológica de Pereira. Facultad de Ingenierías Eléctrica, Electrónica, Física y Ciencias de la Computación. Ingeniería Eléctrica.).
- [5] WILLIAMS, MARCUS; LU, PING-HONG y JOHNSON, JOSEPH M. (2019). “*Movimiento restrictivo de un robot móvil*” (ES2746498T3), ESPAÑA.
- [6] Junichi Urata Yoshito Ito (2017). “detection of movable ground areas of a robots environment using atransducer array” (US9701016B1). ESTADOS UNIDOS DE AMERICA.
- [7] David Schatsky, Craig Muraskin, and Ragu Gurumurthy, *Cognitive technologies: The real opportunities for business*, Deloitte University Press, January 26, 2015.
- [8] Lazo, O. R., & Rojas, L. R. (2006). Diseño asistido por computador. *Industrial Data*, 9(1), 7-15.
- [9] Dassault Systèmes SolidWorks Corp. (1995-2021). SolidWorks 3D Cad. SolidWorks.: <https://www.solidworks.com/es/product/solidworks-3d-cad>

- [10] Ramírez, P. A. L., & Sosa, H. A. (2013). Aprendizaje de y con robótica, algunas experiencias. *Revista Educación*, 37(1), 43-63.
- [11] Jacek, M. (2001). Some thoughts on robotics for education. Proceeding of American Association of Artificial Intelligence Symposium on Robotics and Education. Lund University.
- [12] Nehmzow U. (2003) Introducción. En: Robótica móvil: una introducción práctica. Springer, Londres. https://doi.org/10.1007/978-1-4471-0025-6_1
- [13] Siciliano, B., Sciavicco, L., Villani, L. y Oriolo, G. (2010). Robótica: modelado, planificación y control . Springer Science & Business Media.
- [14] Baturone, A. O. (2005). Robótica: manipuladores y robots móviles. Marcombo.
- [15] J. F. Reyes Cortés, Robótica: Control de Robots Manipuladores, 1a Ed. Alfaomega, pp 509-546, 2011.
- [16] VEX Robotics, Inc. (2002-2021). Batería y cargadores. VEX. <https://www.vexrobotics.com/batteries-and-chargers.html>
- [17]] INPLUS.(2002-2021). SRF08 sensor distancias por ultrasonidos i2c srf08 s320112. Super Robótica. Recuperado 25-01-2021: <http://www.superrobotica.com/s320112.htm>
- [18] Fedorov, D. S., Ivoilov, A. Y., Zhmud, V. A., & Trubin, V. G. (2015). Using of measuring system MPU6050 for the determination of the angular velocities and linear accelerations. *Automatics & Software Enginerry*, 11(1), 75-80.
- [19] MPU-6050 Datasheet (PDF) - List of Unclassified Manufacturers. – URL: <http://www.alldatasheet.com/datasheetpdf/pdf/517744/ETC1/MPU-6050.html>
- [20] del Cusco, A. A. (2019). Control de Motor DC.

- [21] Jennings, S. (2002). Motores paso a paso. *Informador Técnico*, 65, 47-58.
- [22] VEX Robotics, Inc. (2002-2021). Motores. VEX.
https://www.vexrobotics.com/motors.html#attr-vex_outputs
- [23] ARDUINO . (2005-2021). Arduino mega 2560 rev3. ARDUINO.
<https://store.arduino.cc/usa/mega-2560-r3>
- [24] Raspberry Pi Foundation. (2012-2021). About Us. Raspberry pi.
<https://www.raspberrypi.org/about/>
- [25] Raspberry Pi Foundation. (2012-2021). Raspberry pi 3B+. Raspberry pi.:
<https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>
- [26] The Mathworks (1970-2021). Matlab . MathWorks.:
https://www.mathworks.com/products/matlab.html?s_tid=hp_products_matlab
- [27] The Mathworks (1970-2021). Redes Neuronales . MathWorks.:
<https://www.mathworks.com/discovery/neural-network.html>
- [28] The Mathworks (1970-2021). Deep Learning . MathWorks.:
<https://www.mathworks.com/discovery/deep-learning.html>
- [29] The Mathworks (1970-2021). Machine Learning . MathWorks.:
<https://www.mathworks.com/discovery/machine-learning.html>
- [30] The Mathworks (1970-2021). Programación Raspberry Pi con Matlab y Simulink . MathWorks.: <https://www.mathworks.com/discovery/raspberry-pi-programming-matlab-simulink.html>
- [31] The Mathworks (1970-2021). MATLAB CODER. MathWorks.:
<https://www.mathworks.com/products/matlab-coder.html>

- [32] The Mathworks (1970-2021). App Designer. MathWorks.:
<https://www.mathworks.com/products/matlab/app-designer.html>
- [33] Python Software Foundation. (1991-2021). Preguntas frecuentes generales sobre Python. Python.: <https://docs.python.org/3/faq/general.html#what-is-python>
- [34] Labcenter Electronics Ltd.(1988-2021).Proteus. LabCenter.:
<https://www.labcenter.com/>
- [35] Labcenter Electronics Ltd.(1988-2021). Diseño de pcb. LabCenter.:
<https://www.labcenter.com/pcb/>
- [36] VEX Robotics, Inc. (2002-2021). Ruedas. VEX. Ruedas - VEX EDR - VEX Robotics México
- [37] VEX Robotics, Inc. (2002-2021). Carriles . VEX. [Carriles \(2 opciones\) - VEX EDR - VEX Robotics México](#)
- [38] VEX Robotics, Inc. (2002-2021). C-Canales (8-tamaños). VEX. [C-Canales \(8-tamaños\) - VEX EDR - VEX Robotics México](#)
- [39] VEX Robotics, Inc. (2002-2021). Ejes y hardware . VEX. [Ejes y Hardware - VEX EDR - VEX Robotics México](#)
- [40] ARDUINO . (2005-2021). Arduino – Wire. ARDUINO.
<https://www.arduino.cc/en/reference/wire>
- [41] The Mathworks (1970-2021). trainlm. MathWorks.:
<https://www.mathworks.com/help/deeplearning/ref/trainlm.html>
- [42] The Mathworks (1970-2021). trainscg. MathWorks.:
<https://www.mathworks.com/help/deeplearning/ref/trainscg.html;jsessionid=a6d1cc45a73f521afbb7a8919e6f>

[43] The Mathworks (1970-2021). trainbr. MathWorks.:
<https://www.mathworks.com/help/deeplearning/ref/trainbr.html>

LISTADO DE ANEXOS

ANEXO A- CODIGOS

A.1.- función getRange

```
int getRange(int srfAddress) {                                // Esta función obtiene un rango del SRF08
    int range = 0;

    Wire.beginTransmission(srfAddress);                      // Comience a comunicarse con SRF08
    Wire.write(cmdByte);                                     // Envía byte de comando
    Wire.write(0x51);                                       // Envía 0x51 para iniciar un rango
    Wire.endTransmission();

    delay(100);                                            // Espere a que se complete el rango

    Wire.beginTransmission(srfAddress);                      // comenzar a comunicarse con SRF08
    Wire.write(rangeByte);                                   // Llama al registro para iniciar los datos de rango
    Wire.endTransmission();

    Wire.requestFrom(srfAddress, 2);                          // Solicitar 2 bytes del módulo SRF
    while (Wire.available() < 2);                            // Espere a que lleguen los datos
    highByte = Wire.read();                                  // Obtener un byte alto
    lowByte = Wire.read();                                   // Obtener un byte BAJO
    range = (highByte << 8) + lowByte;                     // Ponerlos juntos
    return (range);                                         // Rango de devoluciones
}
```

A.2.-Función girar.

```
int girar(int dir, int pasos) { // función para girar el motor cada 4.5°  
    digitalWrite(dirPin, dir); // configuramos la dirección del giro del motor  
  
    for (int i = 0; i <= pasos; i++) { // giramos los pasos que necesitaremos para  
        llegar a 4.5 grados en nuestro caso serial 96 pasos teniendo en cuenta la caja de  
        engranajes reductora  
  
        digitalWrite(stepPin, HIGH); //enviamos un tren de pulsos al driver del nema17  
        delayMicroseconds(800);  
  
        digitalWrite(stepPin, LOW);  
        delayMicroseconds(800);  
  
    }  
  
    delay(500);  
}
```

A.3.- Función start_1

```
int start_1() // función para girar el motor en dirección contraria  
  
while (marca==1){ // creamos un ciclo infinito para poder leer el pin del final  
de carrera constantemente  
  
    digitalWrite(dirPin, LOW); // configuramos la dirección del giro contrario para el  
motor  
  
    int sensorVal = digitalRead(50); // leemos el pin del final de carrera  
  
  
    if (sensorVal == HIGH) { // mientras no mande un pulso bajo
```

```

digitalWrite(stepPin, HIGH); // enviamos un tren de pulsos al driver del nema17

delayMicroseconds(800);

digitalWrite(stepPin, LOW);

delayMicroseconds(800);

digitalWrite(13,LOW);

}

else {      // cuando deja de mandar un pulso alto el fin de carrera

digitalWrite(13, HIGH);

marca=0;      // salimos del ciclo While

}}}

```

A.4.- Funcion leftmotorcontrol.

```

int leftMotorControl(int value) {      // función del motor izquierdo

int vel=SPEED + value;    // agregamos una velocidad constante y le sumamos la
corrección de velocidad proveniente del control PID

if(vel<-100) vel=-100;// limitamos máximos y mínimos velocidad -100 a 100% de
velocidad

else if( vel> 100) vel=100;

leftMotor.write(map(vel, -100, 100, 1000, 2000)); // mapeamos los máximos y mínimos

}

```

A.5.- Funcion rightmotorcontrol.

```
int rightMotorControl(int value) { // función del motor derecho  
    int vel2 =-SPEED + value; // agregamos una velocidad constante y le sumamos  
    la corrección de velocidad proveniente del control PID  
  
    if(vel2<-100) vel2=-100; // limitamos máximos y mínimos velocidad -100 a 100% de  
    velocidad  
  
    else if( vel2> 100) vel2=100;  
  
    rightMotor.write(map(vel2, -100, 100, 1000, 2000)); // mapeamos los máximos y  
    mínimos  
}
```

A.6.- código para el entrenamiento de las redes neuronales con 3 salidas

```
clear all  
clc  
close all;  
v_cilindro=[40 40 40 40 40 9 8 9 9 8 11 12 4 40 40 40 40 40 40 40 40 40 40 40 40 40  
40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40  
. % 581 datos mas  
. .  
40 40 40 40 40 17 16 16 16 16 17 18 21 40 40 40 40 40 40 40 40 40 40 40 40 40 40  
40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40  
v_caja=[7 6 6 7 7 8 9 9 11 17 18 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40  
40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40  
. % 581 datos mas  
. .  
13 12 12 11 11 11 11 11 12 12 14 14 16 18 40 40 40 40 40 40 40 40 40 40 40 40 40 40  
40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40 40  
];
```


A.7.- código para la comprobación con nuevos datos de las redes neuronales entrenadas con 3 salidas

```

,"sensor_9","sensor_9.1","sensor_9.2","sensor_9.3","sensor_9.4","
","cilindro","caja","nada"];
```

c=1; % variables globales para nuestro codigo
c22=2;
filename = 'prueba de red neuronal 2_datos.xlsx';
d=1;
c1_1=1;
c2_2=2;
e=1;

%%

while(true)

for cuad=1:1:59
d=d+1;
DATA='cuadro';
xlswrite(filename,encabezado,DATA,'A1');
c222=num2str(c22);
c2=strcat('A',c222);
a=caja1(cuad,1:45);
wen=caja1(cuad,1:45);
pira2=num2str(d);
cilindro=strcat('AU',c222);
caja=strcat('AV',c222);
triangulo=strcat('AW',c222);
nada=strcat('AX',c222);
x=red(wen);% metemos los datos a la red que ya importamos al codigo
xlswrite(filename,a,DATA,c222);

if x(1,1)>=0.5
disp(' escaneamos un CILINDRO cerca -->')
disp(d)
xlswrite(filename,'1',DATA,cilindro);

elseif x(2,1)>=0.5
disp(' escaneamos UNA CAJA cerca -->')
disp(d)
xlswrite(filename,'1',DATA,caja);

elseif x(3,1)>=0.5

```

disp(' NO ESCANEAMOS NADA CERCA -->')
disp(d)
xlswrite(filename,'1',DATA,nada);

end
c22=c22+1;
end

c=1;
c22=2;
d=1;

for cili=1:1:59
d=d+1;
DATA='cilindro';
xlswrite(filename,encabezado,DATA,'A1');
c222=num2str(c22);
c2=strcat('A',c222);
a=cilindro1(cili,1:45);
wen=cilindro1(cili,1:45)';
pira2=num2str(d);
cilindro=strcat('AU',c222);
caja=strcat('AV',c222);
triangulo=strcat('AW',c222);
nada=strcat('AX',c222);
x=red(wen);% metemos los datos a la red que ya importamos al codigo
xlswrite(filename,a,DATA,c222);

if x(1,1)>=0.5
disp(' escaneamos un CILINDRO cerca -->')
disp(d)
xlswrite(filename,'1',DATA,cilindro);

elseif x(2,1)>=0.5
disp(' escaneamos UNA CAJA cerca -->')
disp(d)
xlswrite(filename,'1',DATA,caja);

elseif x(3,1)>=0.5
disp(' NO ESCANEAMOS NADA CERCA -->')
disp(d)
xlswrite(filename,'1',DATA,nada);

```

```

    end
    c22=c22+1;
end

c=1;
c22=2;
d=1;

for nadaa=1:1:25
    d=d+1;
    DATA='nada';
    xlswrite(filename,encabezado,DATA,'A1');
    c222=num2str(c22);
    c2=strcat('A',c222);
    a=nad(nadaa,1:45);
    wen=nad(nadaa,1:45)';
    pira2=num2str(d);
    cilindro=strcat('AU',c222);
    caja=strcat('AV',c222);
    triangulo=strcat('AW',c222);
    nada=strcat('AX',c222);
    x=red(wen);% metemos los datos a la red que ya importamos al codigo
    xlswrite(filename,a,DATA,c222);

if x(1,1)>=0.5
    disp('    escaneamos un CILINDRO cerca -->')
    disp(d)
    xlswrite(filename,'1',DATA,cilindro);

elseif x(2,1)>=0.5
    disp('    escaneamos UNA CAJA cerca -->')
    disp(d)
    xlswrite(filename,'1',DATA,caja);

elseif x(3,1)>=0.5
    disp('    NO ESCANEAMOS NADA CERCA -->')
    disp(d)
    xlswrite(filename,'1',DATA,nada);

end
c22=c22+1;
end

```

```
break;  
end
```

A.8.- Función para la graficación de las distancias enviadas a Matlab

```
function grafica(app)  
data_base2=app.conjunto;  
rall = 1;% radio del circulo sensores  
theta = linspace(0, 2*pi);  
n=4.5; % grados entre giros de sensores;  
%----- circulo principal -----  
r = 15.5;% radio del circulo  
xc = 0; %posición del círculo central en x  
yc = 0; %posición del círculo central en y  
x = xc+cos(theta)' *r;  
y = yc+sin(theta)' *r;  
%-----  
%----- circulo donde posicionaremos los sensores -----  
rr = 16.5;% radio del circulo  
xcr = 0; %posición del círculo central en x  
ycr = 0; %posición del círculo central en y  
xr = xcr+cos(theta)' *rr;  
yr = ycr+sin(theta)' *rr;  
%##### SENSOR 1 #####  
num=0;  
xc1 =data_base2(1)* (cosd(num));%-16.1687; %posición del círculo central en x  
yc1 = data_base2(1)* (sind(num)); %-3.3569; %posición del círculo central en y  
xc1=xc1+16.5000 ;  
yc1=yc1+0;  
x_1 = -xc1+cos(theta)' *rall;  
y_1 = -yc1+sin(theta)' *rall;  
%##### SENSOR 2 #####  
num=n;  
xc2 = (data_base2(2)* (cosd(num)));% %posición del círculo central en x  
yc2 = (data_base2(2)* (sind(num))); % %posición del círculo central en y  
xc2=xc2+16.4491 ;  
yc2=yc2+ 1.2946;  
x2 = -xc2+cos(theta)' *rall;  
y2 = yc2+sin(theta)' *rall;  
%##### SENSOR 3 #####  
num=num+n;
```

```

xc3 = data_base2(3)* cosd(num);%posición del círculo central en x
yc3 = data_base2(3)* sind(num);%posición del círculo central en
xc3= xc3+ 16.2969;
yc3= yc3 + 2.5812;
x3 = -xc3+cos(theta)' *rall;
y3 = yc3+sin(theta)' *rall;
%##### SENSOR 4 #####
num=num+n;
xc4 = (data_base2(4)* (cosd(num))); %posición del círculo central en x
yc4 = (data_base2(4)* (sind(num))); %posición del círculo central en
xc4=xc4 + 16.0441;
yc4=yc4 + 3.8518;
x4 = -xc4+cos(theta)' *rall;
y4 = yc4+sin(theta)' *rall;
%##### SENSOR 5 #####
num=num+n;
xc5 = (data_base2(5)* (cosd(num))); %posición del círculo central en x
yc5 = (data_base2(5)* (sind(num))); %posición del círculo central en y
xc5=xc5+ 15.6924;
yc5=yc5+ 5.0988;
x5 = -xc5+cos(theta)' *rall;
y5 = yc5+sin(theta)' *rall;
%##### SENSOR 6 #####
num=num+n;
xc6 = (data_base2(6)* (cosd(num))); %posición del círculo central en x
yc6 = (data_base2(6)* (sind(num))); %posición del círculo central en
xc6=xc6+ 15.2440;
yc6=yc6+ 6.3143;
x6 = -xc6+cos(theta)' *rall;
y6 = yc6+sin(theta)' *rall;
%##### SENSOR 7 #####
num=num+n;
xc7 = (data_base2(7)* (cosd(num))); %posición del círculo central en x
yc7 = (data_base2(7)* (sind(num))); %posición del círculo central en y
xc7=xc7 + 14.7016 ;
yc7=yc7 + 7.4908;
x7 = -xc7+cos(theta)' *rall;
y7 = yc7+sin(theta)' *rall;
%##### SENSOR 8 #####
num=num+n;
xc8 = (data_base2(8)* (cosd(num))); %posición del círculo central en x
yc8 = (data_base2(8)* (sind(num))); %posición del círculo central en y
xc8=xc8+ 14.0686;
yc8= yc8 + 8.6212 ;
x8 = -xc8+cos(theta)' *rall;

```

```

y8 = yc8+sin(theta)' *rall;
%##### SENSOR 9 #####
num=num+n;
xc9 = (data_base2(9)* (cosd(num))); %posición del círculo central en x)
yc9 = (data_base2(9)* (sind(num))); %posición del círculo central en y
xc9=xc9 +13.3488 ;
yc9= yc9 +9.6985 ;
x9= -xc9+cos(theta)' *rall;
y9 = yc9+sin(theta)' *rall;
%##### SENSOR 10 #####
num=num+n;
xc10 = (data_base2(10)* (cosd(num))); %posición del círculo central en x)
yc10 = (data_base2(10)* (sind(num))); %posición del círculo central en y
xc10= xc10 + 12.5467;
yc10= yc10 + 10.7159;
x10= -xc10+cos(theta)' *rall;
y10 = yc10+sin(theta)' *rall;
%##### SENSOR 11 #####
num=num+n;
xc11 = (data_base2(11)* (cosd(num))); %posición del círculo central en x)
yc11 = (data_base2(11)* (sind(num))); %posición del círculo central en y
xc11= xc11 + 11.6673 ;
yc11=yc11 + 11.6673 ;
x11 = -xc11+cos(theta)' *rall;
y11 = yc11+sin(theta)' *rall;
%##### SENSOR 12 #####
num=num+n;
xc12 = (data_base2(12)* (cosd(num))); %posición del círculo central en x)
yc12 = (data_base2(12)* (sind(num))); %posición del círculo central en y
xc12 = xc12 + 10.7159 ;
yc12 = yc12 + 12.5467;
x12 = -xc12+cos(theta)' *rall;
y12 = yc12+sin(theta)' *rall;
%##### SENSOR 13 #####
num=num+n;
xc13 = (data_base2(13)* (cosd(num))); %posición del círculo central en x)
yc13 = (data_base2(13)* (sind(num))); %posición del círculo central en y
xc13=xc13 + 9.6985;
yc13=yc13 + 13.3488;
x13 = -xc13+cos(theta)' *rall;
y13 = yc13+sin(theta)' *rall;
%##### SENSOR 14 #####
num=num+n;
xc14 = (data_base2(14)* (cosd(num))); %posición del círculo central en x)
yc14 = (data_base2(14)* (sind(num))); %posición del círculo central en y

```

```

xc14 = xc14 + 8.6212;
yc14= yc14 + 14.0686;
x14 = -xc14+cos(theta)' *rall;
y14 = yc14+sin(theta)' *rall;
%##### SENSOR 15 #####
num=num+n;
xc15 = (data_base2(15)* (cosd(num))); %posición del círculo central en x
yc15 = (data_base2(15)* (sind(num))); %posición del círculo central en y
xc15 =xc15+ 7.4908;
yc15= yc15+ 14.7016;
x15= -xc15+cos(theta)' *rall;
y15 = yc15+sin(theta)' *rall;
%##### SENSOR 16 #####
num=num+n;
xc16 = (data_base2(16)* (cosd(num))); %posición del círculo central en x
yc16 = (data_base2(16)* (sind(num))); %posición del círculo central en y
xc16=xc16+ 6.3143;
yc16= yc16+ 15.2440;
x16 = -xc16+cos(theta)' *rall;
y16 = yc16+sin(theta)' *rall;
%##### SENSOR 17 #####
num=num+n;
xc17 = (data_base2(17)* (cosd(num))); %posición del círculo central en x
yc17 = (data_base2(17)* (sind(num))); %posición del círculo central en y
xc17=xc17 + 5.0988;
yc17= yc17 + 15.6924;
x17 = -xc17+cos(theta)' *rall;
y17= yc17+sin(theta)' *rall;
%##### SENSOR 18 #####
num=num+n;
xc18 = (data_base2(18)* (cosd(num))); %posición del círculo central en x
yc18 = (data_base2(18)* (sind(num))); %posición del círculo central en y
xc18= xc18 + 3.8518;
yc18=yc18 + 16.0441;
x18 = -xc18+cos(theta)' *rall;
y18= yc18+sin(theta)' *rall;
%##### SENSOR 19 #####
num=num+n;
xc19 = (data_base2(19)* (cosd(num))); %posición del círculo central en x
yc19 = (data_base2(19)* (sind(num))); %posición del círculo central en y
xc19= xc19 +2.5812 ;
yc19= yc19 + 16.2969;
x19 = -xc19+cos(theta)' *rall;
y19= yc19+sin(theta)' *rall;
%##### SENSOR 20 #####

```

```

num=num+n;
xc20 = (data_base2(20)* (cosd(num))); %posición del círculo central en x
yc20 = (data_base2(20)* (sind(num))); %posición del círculo central en y
xc20= xc20+ 1.2946;
yc20=yc20 + 16.4491;
x20 = -xc20+cos(theta)' *rall;
y20= yc20+sin(theta)' *rall;
%##### SENSOR 21 #####
num=num+n;
xc21 = (data_base2(21)* (cosd(num))); %posición del círculo central en x
yc21 = (data_base2(21)* (sind(num))); %posición del círculo central en y
xc21=xc21 + 0;
yc21= yc21 + 16.5000;
x21= -xc21+cos(theta)' *rall;
y21= yc21+sin(theta)' *rall;
%##### SENSOR 22 #####
num=num+n;
xc22 = (data_base2(22)* (cosd(num))); %posición del círculo central en x
yc22 = (data_base2(22)* (sind(num))); %posición del círculo central en y
xc22=xc22+ -1.2946;
yc22=yc22 + 16.4491 ;
x22= -xc22+cos(theta)' *rall;
y22= yc22+sin(theta)' *rall;
%##### SENSOR 23 #####
num=num+n;
xc23 = (data_base2(23)* (cosd(num))); %posición del círculo central en x
yc23 = (data_base2(23)* (sind(num))); %posición del círculo central en y
xc23=xc23 + -2.5812;
yc23=yc23 + 16.2969;
x23= -xc23+cos(theta)' *rall;
y23= yc23+sin(theta)' *rall;
%##### SENSOR 24 #####
num=num+n;
xc24 = (data_base2(24)* (cosd(num))); %posición del círculo central en x
yc24 = (data_base2(24)* (sind(num))); %posición del círculo central en y
xc24=xc24 + -3.8518 ;
yc24=yc24 + 16.0441 ;
x24= -xc24+cos(theta)' *rall;
y24= yc24+sin(theta)' *rall;
%##### SENSOR 25 #####
num=num+n;
xc25 = (data_base2(25)* (cosd(num))); %posición del círculo central en x
yc25 = (data_base2(25)* (sind(num))); %posición del círculo central en y
xc25=xc25 + -5.0988;
yc25=yc25 + 15.6924 ;

```

```

x25= -xc25+cos(theta)' *rall;
y25= yc25+sin(theta)' *rall;
%##### SENSOR 26 #####
num=num+n;
xc26 = (data_base2(26)* (cosd(num))); %posición del círculo central en x
yc26 = (data_base2(26)* (sind(num))); %posición del círculo central en y
xc26=xc26 + -6.3143;
yc26=yc26 + 15.2440;
x26= -xc26+cos(theta)' *rall;
y26= yc26+sin(theta)' *rall;
%##### SENSOR 27 #####
num=num+n;
xc27 = (data_base2(27)* (cosd(num))); %posición del círculo central en x
yc27 = (data_base2(27)* (sind(num))); %posición del círculo central en y
xc27=xc27 + -7.4908;
yc27=yc27+ 14.7016;
x27= -xc27+cos(theta)' *rall;
y27= yc27+sin(theta)' *rall;
%##### SENSOR 28 #####
num=num+n;
xc28 = (data_base2(28)* (cosd(num))); %posición del círculo central en x
yc28 = (data_base2(28)* (sind(num))); %posición del círculo central en y
xc28=xc28 + -8.6212;
yc28=yc28 + 14.0686;
x28= -xc28+cos(theta)' *rall;
y28= yc28+sin(theta)' *rall;
%##### SENSOR 29 #####
num=num+n;
xc29 = (data_base2(29)* (cosd(num))); %posición del círculo central en x
yc29 = (data_base2(29)* (sind(num))); %posición del círculo central en y
xc29=xc29 + -9.6985;
yc29=yc29 + 13.3488;
x29= -xc29+cos(theta)' *rall;
y29= yc29+sin(theta)' *rall;
%##### SENSOR 30 #####
num=num+n;
xc30 = (data_base2(30)* (cosd(num))); %posición del círculo central en x
yc30 = (data_base2(30)* (sind(num))); %posición del círculo central en y
xc30=xc30 + -10.7159 ;
yc30=yc30 + 12.5467;
x30= -xc30+cos(theta)' *rall;
y30= yc30+sin(theta)' *rall;
%##### SENSOR 31 #####
num=num+n;
xc31 = (data_base2(31)* (cosd(num))); %posición del círculo central en x

```

```

yc31 = (data_base2(31)* (sind(num))); %posición del círculo central en y
xc31=xc31 + -11.6673;
yc31=yc31 + 11.6673 ;
x31= -xc31+cos(theta)' *rall;
y31= yc31+sin(theta)' *rall;
%##### SENSOR 32 #####
num=num+n;
xc32 = (data_base2(32)* (cosd(num))); %posición del círculo central en x)
yc32 = (data_base2(32)* (sind(num))); %posición del círculo central en y
xc32=xc32 + -12.5467;
yc32=yc32 + 10.7159;
x32= -xc32+cos(theta)' *rall;
y32= yc32+sin(theta)' *rall;
%##### SENSOR 33 #####
num=num+n;
xc33 = (data_base2(33)* (cosd(num))); %posición del círculo central en x)
yc33 = (data_base2(33)* (sind(num))); %posición del círculo central en y
xc33=xc33 + -13.3488;
yc33=yc33 + 9.6985;
x33= -xc33+cos(theta)' *rall;
y33= yc33+sin(theta)' *rall;
%##### SENSOR 34 #####
num=num+n;
xc34 = (data_base2(34)* (cosd(num))); %posición del círculo central en x)
yc34 = (data_base2(34)* (sind(num))); %posición del círculo central en y
xc34=xc34 + -14.0686 ;
yc34=yc34 + 8.6212;
x34= -xc34+cos(theta)' *rall;
y34= yc34+sin(theta)' *rall;
%##### SENSOR 35 #####
num=num+n;
xc35 = (data_base2(35)* (cosd(num))); %posición del círculo central en x)
yc35 = (data_base2(35)* (sind(num))); %posición del círculo central en y
xc35=xc35 + -14.7016;
yc35=yc35 + 7.4908;
x35= -xc35+cos(theta)' *rall;
y35= yc35+sin(theta)' *rall;
%##### SENSOR 36 #####
num=num+n;
xc36 = (data_base2(36)* (cosd(num))); %posición del círculo central en x)
yc36 = data_base2(36)* sind(num); %posición del círculo central en y
xc36 = xc36 + -15.2440;
yc36 = yc36 + 6.3143;
x36= -xc36+cos(theta)' *rall;
y36= yc36+sin(theta)' *rall;

```

```

%##### SENSOR 37 #####
num=num+n;
xc37 = (data_base2(37)* (cosd(num))); %posición del círculo central en x
yc37 = (data_base2(37)* (sind(num))); %posición del círculo central en y
xc37=xc37 + -15.6924 ;
yc37=yc37 + 5.0988 ;
x37= -xc37+cos(theta)' *rall;
y37= yc37+sin(theta)' *rall;
%##### SENSOR 38 #####
num=num+n;
xc38 = (data_base2(38)* (cosd(num))); %posición del círculo central en x
yc38 = (data_base2(38)* (sind(num))); %posición del círculo central en y
xc38=xc38 + -16.0441;
yc38=yc38 + 3.8518 ;
x38= -xc38+cos(theta)' *rall;
y38= yc38+sin(theta)' *rall;
%##### SENSOR 39 #####
num=num+n;
xc39 = (data_base2(39)* (cosd(num))); %posición del círculo central en x
yc39 = (data_base2(39)* (sind(num))); %posición del círculo central en y
xc39= xc39 + -16.2969 ;
yc39= yc39 + 2.5812;
x39= -xc39+cos(theta)' *rall;
y39= yc39+sin(theta)' *rall;
%##### SENSOR 40 #####
num=num+n;
xc40 = (data_base2(40)* (cosd(num))); %posición del círculo central en x
yc40 = (data_base2(40)* (sind(num))); %posición del círculo central en y
xc40= xc40 + -16.4491;
yc40= yc40 + 1.2946 ;
x40= -xc40+cos(theta)' *rall;
y40= yc40+sin(theta)' *rall;
%##### SENSOR 41 #####
num=0;
xc41 = (data_base2(41)* (cosd(num))); %posición del círculo central en x
yc41 = (data_base2(41)* (sind(num))); %posición del círculo central en y
xc41=xc41 + 16.5000;
yc41= yc41 + 0;
x41= xc41+cos(theta)' *rall;
y41= -yc41+sin(theta)' *rall;
%##### SENSOR 42 #####
num=num+n;
xc42 = (data_base2(42)* (cosd(num))); %posición del círculo central en x
yc42 = (data_base2(42)* (sind(num))); %posición del círculo central en y
xc42 = xc42+ 16.4491;

```

```

yc42 = yc42+ 1.2946;
x42= xc42+cos(theta)' *rall;
y42= -yc42+sin(theta)' *rall;
%##### SENSOR 43 #####
num=num+n;
xc43 = (data_base2(43)* (cosd(num))); %posición del círculo central en x
yc43 = (data_base2(43)* (sind(num))); %posición del círculo central en y
xc43 = xc43+ 16.2969;
yc43 =yc43+ 2.5812;
x43= xc43+cos(theta)' *rall;
y43= -yc43+sin(theta)' *rall;
%##### SENSOR 44 #####
num=num+n;
xc44 = (data_base2(44)* (cosd(num))); %posición del círculo central en x
yc44 = (data_base2(44)* (sind(num))); %posición del círculo central en y
xc44 = xc44+ 16.0441;
yc44 = yc44 + 3.8518;
x44= xc44+cos(theta)' *rall;
y44= -yc44+sin(theta)' *rall;
%##### SENSOR 45 #####
num=num+n;
xc45 = (data_base2(45)* (cosd(num))); %posición del círculo central en x
yc45 = (data_base2(45)* (sind(num))); %posición del círculo central en y
xc45 = xc45 +15.6924;
yc45 = yc45 + 5.0988;
x45= xc45+cos(theta)' *rall;
y45= -yc45+sin(theta)' *rall;
%-----
%-----sensores plot -----
X=[x_1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16 x17 x18 x19 x20 x21 x22
x23 x24 x25 x26 x27 x28 x29 x30 x31 x32 x33 x34 x35 x36 x37 x38 x39 x40 x41 x42
x43 x44 x45];
Y=[y_1 y2 y3 y4 y5 y6 y7 y8 y9 y10 y11 y12 y13 y14 y15 y16 y17 y18 y19 y20 y21 y22
y23 y24 y25 y26 y27 y28 y29 y30 y31 y32 y33 y34 y35 y36 y37 y38 y39 y40 y41 y42
y43 y44 y45];
plot(app.axes,x,y,'k',xr,yr,'-- g',X,Y,'b');

end

```

A.8.- Función para recolección de datos de la raspberry y Matlab

```
function raspberry(app)

    app.GRAFICAR.Enable='off';
    write(app.myserialdevice,'1') %envía un 1 al puerto serial

    while app.c==1

        ultra = read(app.myserialdevice,2,'char');%recolectamos lo que hay en el
        puerto serial de dos en dos
        x = str2num(ultra); %trasform2amos los datos de tipo str a numérico
        app.dat = [app.dat,x]; %guardamos el dato recolectado en un array

        if length(app.dat)==9 % cuando dato llegue a los 9 datos recolectados entrara en
        este if
            disp(app.dat)
            if app.cont_break==1

                for e = 1:5:45
                    app.conjunto(1,e)=app.dat(1,app.cont1);% agregamos los primeros 9 datos a
                    un conjunto , dejando 2 espacios entre dato y dato
                    app.cont1 = app.cont1 +1;
                    if app.cont1 >9 % cuando agregamos los 9 datos rompemos el ciclo y
                    reiniciamos los contadores y la variable dat
                        app.LABEL1.Text = 'INICIANDO EL RECONOCIMIENTO .';
                        app.cont_break = app.cont_break +1;
                        app.cont1 = 0;
                        %borrar luego
                        disp(app.conjunto);
                        app.dat=[];
                    end
                end
            elseif app.cont_break==2

                for f = 2:5:45
                    app.conjunto(1,f)=app.dat(1,app.cont2);% agregamos los primeros 9 datos
                    a un conjunto , dejando 2 espacios entre dato y dato
                    app.cont2 = app.cont2 +1;
                    if app.cont2>9 % cuando agregamos los 9 datos rompemos el ciclo y
                    reiniciamos los contadores y la variable dat
                        app.cont_break =app.cont_break +1;
                        app.cont2=0;
                        %borrar luego
                        disp(app.conjunto);
                    end
                end
            end
        end
    end
```

```

        app.dat=[];
    end
end
elseif app.cont_break==3

for g = 3:5:45
    app.conjunto(1,g)=app.dat(1,app.cont3);% agregamos los primeros 9
    datos a un conjunto , dejando 2 espacios entre dato y dato
    app.cont3 = app.cont3 +1;
    if app.cont3>9 % cuando agregamos los 9 datos rompemos el ciclo y
    reiniciamos los contadores y la variable dat
        app.cont_break = app.cont_break +1;
        app.cont3=0;
        %borrar luego
        disp(app.conjunto);
        app.dat=[];
    end
end
elseif app.cont_break==4
    for h = 4:5:45
        app.conjunto(1,h)=app.dat(1,app.cont4);% agregamos los primeros 9 datos
        a un conjunto , dejando 2 espacios entre dato y dato
        app.cont4 = app.cont4 +1;
        if app.cont4>9 % cuando agregamos los 9 datos rompemos el ciclo y reiniciamos
        los contadores y la variable dat
            app.cont_break = app.cont_break +1;
            app.cont4=0;
            %borrar luego
            disp(app.conjunto);
            app.dat=[];
        end
    end
elseif app.cont_break==5
    for i = 5:5:45
        app.conjunto(1,i)=app.dat(1,app.cont5);% agregamos los primeros
        9 datos a un conjunto , dejando 2 espacios entre dato y dato
        app.cont5 = app.cont5 +1;
        if app.cont5>9 % cuando agregamos los 9 datos rompemos el ciclo y
        reiniciamos los contadores y la variable dat
            app.cont_break = app.cont_break +1;
            app.cont5=0;
            %borrar luego
            disp(app.conjunto);
            app.dat=[];
    end

```

```

end
end
if app.cont_break>=6 % cuando el cont_break llega a 4 o sea que hizo 3 iteraciones ya
que lo iniciamos con el valor de 1
    A = app.conjunto';
    x=app.red.red(A);
if x(1,1)>=0.5
%cilindro
    app.LABEL1.Text ="detectamos un objeto de forma CILINDRICA";
elseif x(2,1)>=0.5
%caja
    app.LABEL1.Text ="detectamos un objeto de forma cuadrada";
elseif x(3,1)>=0.5
%nada
    app.LABEL1.Text ="NO DETECTAMOS NADA CERCANO";
end
cla(app.axes,"reset");% reseteamos la grafica limpiandola para volver a graficar con
nuevos datos
grafica(app);
app.cont_break = 1;
app.dat=[];
app.cont1 = 1;
app.cont2 = 1;
app.cont3 = 1;
app.cont4 = 1;
app.cont5 = 1;
app.c=0;
app.conjunto=[];
end
end
end
end

```

ANEXO B.-Tablas.

B.1.- Tabla de los entrenamientos para la etapa uno

NO. PRUEBA	FUNCION DE ENTRENAMIENTO	PIRAMIDE	CAJA	CILINDRO	NADA	PORCENTAJE FINAL	%
prueba 1	trainlm	32.2	55.93	58.62	100	61.6875	%
prueba 2	trainlm	37.28	64.4	56.89	100	64.6425	%
prueba 3	trainlm	30.5	71.18	55.93	100	64.4025	%
prueba 4	trainlm	29.31	67.79	55.93	100	63.2575	%

B.2.- Tabla de los entrenamientos para la etapa dos

NO. PRUEBA	FUNCION DE ENTRENAMIENTO	PIRAMIDE	CAJA	CILINDRO	NADA	PORCENTAJE FINAL	%
prueba 2.1	trainlm	22.03	74.57	77.96	100	68.64	%
prueba 2.2	trainlm	27.11	61.01	77.96	100	66.52	%
prueba 2.3	trainlm	40.67	77.96	59.32	100	69.4875	%
prueba 2.4	trainlm	22.03	84.74	71.92	100	69.6725	%
prueba 2.5	trainlm	20.68	74.13	69.49	100	66.075	%
prueba 2.6	trainlm	7.4	74.57	69.49	100	62.865	%

B.3.- Tabla de los entrenamientos para la etapa tres

NO. PRUEBA	FUNCION DE ENTRENAMIENTO	CAJA	BALON	CILINDRO	NADA	PORCENTAJE FINAL	%
prueba 3.1	trainlm	84.74	33.89	61.01	100	69.91	%
prueba 3.2	trainlm	74.57	32.2	64.4	100	67.7925	%
prueba 3.3	trainlm	74.57	25.42	66.1	100	66.5225	%
prueba 3.4	trainlm	74.57	30.5	67.24	100	68.0775	%
prueba 3.5	trainlm	79.66	32.2	67.79	100	69.9125	%
prueba 3.6	trainscg	88.13	24.13	57.62	100	67.47	%
prueba 3.7	trainbr	71.18	27.58	71.18	100	67.485	%
prueba 3.8	trainscg	76.27	37.93	61.01	100	68.8025	%

B.4.- Tabla de los entrenamientos para la etapa cuatro

NO. PRUEBA	FUNCION DE ENTRENAMIENTO	CAJA	CILINDRO	nada	PORCENTAJE FINAL	%
prueba 4.1	trainlm	74.57	81.35	100	85.306667	%
prueba 4.2	trainbr	88.13	74.57	100	87.566667	%
prueba 4.3	trainscg	88.13	72.88	100	87.003333	%
prueba 4.4	trainscg	93.22	67.79	100	87.003333	%
prueba 4.5	trainscg	83.05	81.35	100	88.133333	%
prueba 4.6	trainscg	83.05	94.91	100	92.653333	%
prueba 4.7	trainbr	86.44	64.4	100	83.613333	%
prueba 4.8	trainbr	94.91	72.88	100	89.263333	%
prueba 4.9	trainlm	72.88	86.44	100	86.44	%
prueba 4.10	trainlm	89.83	76.27	100	88.7	%
prueba 4.11	trainlm	88.13	84.74	100	90.956667	%
prueba 4.12	trainlm	77.96	72.88	100	83.613333	%

B.5.- Tabla de los entrenamientos para la etapa cinco

NO. PRUEBA	FUNCION DE ENTRENAMIENTO	CAJA	CILINDRO	nada	PORCENTAJE FINAL	%
PRUEBA 5.1	TRAINLM	77.96	69.49	100	82.483333	%
PRUEBA 5.2	TRAINLM	93.22	84.74	100	92.653333	%
PRUEBA 5.3	TRAINLM	79.66	81.35	100	87.003333	%
PRUEBA 5.4	TRAINLM	84.74	74.57	100	86.436667	%
PRUEBA 5.5	TRAINS CG	89.83	83.05	100	90.96	%
PRUEBA 5.6	TRAINS CG	88.13	72.88	100	87.003333	%
PRUEBA 5.7	TRAINS CG	98.3	91.52	100	96.606667	%
PRUEBA 5.8	TRAINS CG	74.57	86.44	100	87.003333	%
PRUEBA 5.9	TRAINS CG	93.22	81.35	100	91.523333	%
PRUEBA 5.10	TRAINS CG	83.05	84.74	100	89.263333	%
PRUEBA 5.11	TRAINS CG	91.52	83.05	100	91.523333	%
PRUEBA 5.12	TRAINS CG	88.13	67.79	100	85.306667	%