



SEP

SECRETARÍA DE
EDUCACIÓN PÚBLICA



UNIVERSIDAD TECNOLÓGICA DE TIJUANA

TÍTULO DEL PROYECTO

PROTOTIPO DE MANO ROBOTICA CONTROLADA POR
DIFERENTES INTERFACES

TRABAJO RECEPCIONAL

**TÍTULO A OTORGAR
INGENIERÍA EN MECATRÓNICA**

PRESENTA

ROJAS HIGUERA MORIANCUMER

TIJUANA, B.C.

ABRIL, 2021

UNIVERSIDAD TECNOLÓGICA DE TIJUANA
MECATRÓNICA



TÍTULO A OTORGAR
INGENIERÍA EN MECATRÓNICA
TRABAJO RECEPCIONAL

Realizada por

Rojas Higuera Moriancumer

En la empresa

Universidad Tecnológica de Tijuana

Director de Trabajo Receptacional

M.C. Julio César Castro Bojórquez

Tijuana, B.C. Abril, 2021

RESUMEN

En este documento se presentarán diferentes interfaces para el control manual de una mano robótica. Se mostrará una interfaz de HMI elaborada en LabVIEW, un control por medio de un exoesqueleto que estará vinculado al HMI, un control por medio de gesturas en visión a través de una cámara web y una interfaz por medio de una aplicación de celular, la cual tendrá control y lectura de variables sobre la mano robótica.

ÍNDICE

	Página
ÍNDICE DE TABLAS	7
ÍNDICE DE ILUSTRACIONES	8
CAPITULO I. INTRODUCCION.....	16
Antecedentes.....	16
De la institución.....	16
Objetivos.....	17
Objetivo General	17
Objetivos Específicos	17
Factibilidad	18
Plan de Trabajo	20
CAPÍTULO II MARCO TEÓRICO.....	22
Interfaz.....	22
Mano Robótica.....	23
Exoesqueleto.....	27
Impresora 3D	27
PLA.....	28
Microcontrolador	28
Aduino DUE	29
HMI	31
LabVIEW	32
MySQL.....	33

phpMyAdmin	34
XAMPP	34
IDE	35
OpenCV	35
ApplInventor	36
Potenciómetro 3382G-1-252G	37
Fuente de poder	38
Mediapipe	39
CAPÍTULO III. ESTRATEGIA METODOLÓGICA Y RESULTADOS	41
Elaboración del HMI	41
Diseño del panel frontal.....	48
Diagrama de bloques	52
Elaboración del exoesqueleto.....	73
Elaboración código Arduino	89
Filtro para potenciómetros.....	89
Control desde LabVIEW y aplicación de celular.....	91
Comunicación entre botones físicos y virtuales	92
Elaboración de aplicación de celular	96
Elaboración control por gesturas	119
CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES	124
REFERENCIAS.....	127
ANEXOS	130
Elaboración de la mano robótica	130
Elaboración caja de control	142
Elaboración fuente de poder.....	144

ÍNDICE DE TABLAS

Tabla 1. Tabla de costos (Hardware)	19
Tabla 2. Tabla de costos (Software).....	20
Tabla 3. Cronograma de Actividades	21

ÍNDICE DE ILUSTRACIONES

Ilustración 1. Mano mecánica Ambroise (1564)	24
Ilustración 2. Brazo industrial con agarre tipo mano (1962)	25
Ilustración 3. Mano Belgrado/Usc (1969)	25
Ilustración 4. Mano Utah/Mit (1982)	26
Ilustración 5. Mano Shadow (2003).....	26
Ilustración 6. Exoesqueleto	27
Ilustración 7. Impresora 3D	27
Ilustración 8. Arduino DUE	30
Ilustración 9. HMI	32
Ilustración 10. MySQL logo	33
Ilustración 11. phpMyAdmin	34
Ilustración 12. XAMPP logo.....	35
Ilustración 13. OpenCV logo.....	36
Ilustración 14. AppInventor logo	36
Ilustración 15. Potenciómetro 252G	38
Ilustración 16. Cargador K3 Kingboard	38
Ilustración 17. Detección y rastreo de posición	39
Ilustración 18. Malla facial	39
Ilustración 19. Detección y rastreo de objetos.....	40
Ilustración 20. Rastreo de manos.....	40
Ilustración 21. Pantalla usuario	41
Ilustración 22. Usuario incorrecto	42
Ilustración 23. Pantalla contraseña	42
Ilustración 24. Contraseña incorrecta.....	42
Ilustración 25. HMI pestaña principal	43
Ilustración 26. HMI pestaña principal modo lectura	43
Ilustración 27. HMI pestaña prinipal modo control.....	44
Ilustración 28. HMI pestaña grafica combinada.....	44
Ilustración 29. HMI pestaña grafica horizontal modo lectura	45

Ilustración 30. HMI señales específicas	45
Ilustración 31. HMI pestaña grafica horizontal modo control	46
Ilustración 32. HMI ubicación de botón para las alarmas	46
Ilustración 33. HMI pestaña de alarmas	47
Ilustración 34. HMI cambio a búsqueda por fechas.....	47
Ilustración 35. HMI tabla de alertas por sección.....	47
Ilustración 36. HMI botones de control para tabla SQL	48
Ilustración 37. Tab control	48
Ilustración 38. Arquitectura pestaña usuario/contraseña.....	49
Ilustración 39. Modo personalizado LED	49
Ilustración 40. Paleta de herramientas	50
Ilustración 41, Arquitectura de pestaña no.2 HMI.....	50
Ilustración 42. Decoraciones pestaña no.2 HMI	51
Ilustración 43. Ubicación para herramienta grafica	51
Ilustración 44. Medidor plateado	51
Ilustración 45. Lista y control de fechas	52
Ilustración 46. Conversión de control a indicador.....	53
Ilustración 47. Máquina de estados	53
Ilustración 48. Pre-usuario - máquina de estados	54
Ilustración 49. Menú de property node	54
Ilustración 50. Variable local para "tab control"	55
Ilustración 51. Configuración estado usuario.....	56
Ilustración 52. Configuración estado contraseña.....	56
Ilustración 53. Estado apagado	57
Ilustración 54. Estado encendido	57
Ilustración 55. Lógica de control HMI	58
Ilustración 56. "Is value change" y "boolean to (0,1)"	58
Ilustración 57. Ventana de configuración serial.....	59
Ilustración 58. Configuración puerto serial	60
Ilustración 59. Modo lectura VISA read	60

Ilustración 60. "Match pattern" y "Decimal string to number"	61
Ilustración 61. "VISA write" en pre estado	61
Ilustración 62. "Formula node" animaciones mano robótica.....	62
Ilustración 63. Caso no.4 animación mano robótica.....	62
Ilustración 64. Lógica para las alarmas	63
Ilustración 65. Archivo "ODBC driver for MySQL (connector/ODBC)"	64
Ilustración 66. Creación base de datos	65
Ilustración 67. Creación de tabla	65
Ilustración 68. Configuración de la tabla	65
Ilustración 69. Tabla configurada para inserción de datos	66
Ilustración 70. Buscador ODBC source data	66
Ilustración 71. Administrador de origen de datos ODBC	66
Ilustración 72. MySQL connector/ODBC	67
Ilustración 73. Vinculo de datos LabVIEW	67
Ilustración 74. Configuración vinculo de datos	68
Ilustración 75. Programación entrada y salida DB tools	68
Ilustración 76. Configuración DB tools insert data	69
Ilustración 77. Programación para consulta de datos.....	70
Ilustración 78. Database funciones	70
Ilustración 79. Editor en línea de phpMyAdmin	71
Ilustración 80. Programación LabVIEW para insertar en editor en línea	71
Ilustración 81. Comando para limpiar tabla	72
Ilustración 82. Programación graficas	72
Ilustración 83. Propiedades para grafica específica	73
Ilustración 84. Exoesqueleto BioMakers Industries	73
Ilustración 85. Prueba exoesqueleto con cartón.....	74
Ilustración 86. Mediciones para piezas de cartón.....	74
Ilustración 87. Dorso y muñeca de cartón	74
Ilustración 88. Diseño CAD base potenciómetro	75
Ilustración 89. Base nudillos con vista de selección.....	75

Ilustración 90. Corte base de potenciómetro	76
Ilustración 91. Conexión con giro potenciómetro.....	76
Ilustración 92. Pieza de entrada para base giratoria del potenciómetro	76
Ilustración 93. Piezas de flexión para los dedos.....	77
Ilustración 94. Diseño CAD dorso	77
Ilustración 95. Diseño CAD base nudillos	78
Ilustración 96. Diseño CAD curvatura del pulgar.....	79
Ilustración 97. Diseño CAD base para el pulgar.....	79
Ilustración 98. Diseño CAD configuración muñeca	80
Ilustración 99. Diseño CAD corte en muñeca.....	80
Ilustración 100. Diseño CAD muñeca.....	81
Ilustración 101. Diseño CAD unión antebrazo y muñeca	81
Ilustración 102. Diseño CAD pieza unión antebrazo	81
Ilustración 103. Diseño CAD distancias para pieza de unión	82
Ilustración 104. Diseña CAD unión esférica	82
Ilustración 105. Diseño CAD redondeo cubo.....	83
Ilustración 106. Diseño CAD copia sólido	83
Ilustración 107. Agregando configuraciones a una pieza	83
Ilustración 108. Diseño CAD configuración de pieza.....	84
Ilustración 109. Diseño CAD anillo dedo índice.....	84
Ilustración 110. Diseño CAD exoesqueleto	85
Ilustración 111. Impresiones 3D de muñeca y antebrazo.....	85
Ilustración 112. Impresiones 3D de eslabones para los dedos	85
Ilustración 113. Dorso y dedos armados	86
Ilustración 114. Exoesqueleto armado	86
Ilustración 115. Diseño CAD base potenciómetro 252G	87
Ilustración 116. Base horizontal para potenciómetro 252G	87
Ilustración 117. Eje para giro de potenciómetros	88
Ilustración 118. Seguro para eje de potenciómetro	88
Ilustración 119. Diseño CAD nuevo potenciómetro	88

Ilustración 120. Exoesqueleto final.....	89
Ilustración 121. Programación filtro potenciómetros.....	89
Ilustración 122. Declaración de variables para filtro	90
Ilustración 123. Operación de array filtro.....	90
Ilustración 124. Redondeo de array filtro.....	90
Ilustración 125. Mapeo para servomotor	91
Ilustración 126. Variable tipo carácter	91
Ilustración 127. Programación de control	92
Ilustración 128. Declaración de variables para botones	92
Ilustración 129. Entradas, salidas y valores iniciales.....	93
Ilustración 130. Programación arduino apagado	93
Ilustración 131. Encendido de sistema Arduino.....	94
Ilustración 132. Encendido de sistema Arduino 2.....	95
Ilustración 133. Modo lectura Arduino	96
Ilustración 134. Modo lectura Arduino 2	96
Ilustración 135. Aplicación de celular menú	97
Ilustración 136. Aplicación de celular control.....	97
Ilustración 137. Aplicación de celular gráficas.....	98
Ilustración 138. Aplicación de celular comando de voz	99
Ilustración 139. Aplicación de celular configuración	99
Ilustración 140. Creación nuevo proyecto	100
Ilustración 141. Herramientas para elaborar nuevo proyecto	100
Ilustración 142. Renombrar componentes.....	101
Ilustración 143. Propiedades de componente	101
Ilustración 144. Subir archivo	102
Ilustración 145. Inserción de "layout"	102
Ilustración 146. Layout height.....	103
Ilustración 147. Propiedades de "label"	104
Ilustración 148. Propiedades de botones	104
Ilustración 149. App Inventor Designer	105

Ilustración 150. Función para botón control	105
Ilustración 151. Función para abrir pestaña	106
Ilustración 152. Dato string.....	106
Ilustración 153. Código de bloques pestaña menú.....	106
Ilustración 154. Selección de "ListPicker".....	107
Ilustración 155. Propiedades de "Slider"	107
Ilustración 156. Ubicación de "BluetoothClient".....	108
Ilustración 157. Propiedades de "BluetoothClient"	108
Ilustración 158. Programación en bloques pestaña “Control”.....	108
Ilustración 159. Comandos de “ListPicker”	109
Ilustración 160. Programación nombres y direcciones bluetooth	109
Ilustración 161. Programación "AfterPicking"	110
Ilustración 162. Conexión "call BluetoothClient Connectaddress"	110
Ilustración 163. Conexión "set ListPicker Text to"	110
Ilustración 164. Programación control slider.....	111
Ilustración 165. Ubicación bloque "if then"	111
Ilustración 166. Programación botón de vuelta	112
Ilustración 167. Ubicación se "Canvas" y "Clock"	112
Ilustración 168. Propiedades de "Canvas" y "Clock"	113
Ilustración 169. Programación para gráfica	113
Ilustración 170. Ubicación mayor igual a	114
Ilustración 171. Ubicación de variables	115
Ilustración 172. Primer parte programación gráfica	115
Ilustración 173. Segunda parte de programación gráfica	116
Ilustración 174. Ubicación "SpeechRecognizer" y "TinyDB".....	117
Ilustración 175. Programación 1ra parte comando por voz	118
Ilustración 176. Programación 1ra parte comando de voz	118
Ilustración 177. Puntos de referencia para la mano	119
Ilustración 178. Librerías para Python.....	119
Ilustración 179. Programa Firmata en Arduino	120

Ilustración 180. Declarando salidas Arduino en Python	120
Ilustración 181. Posición inicial de servos en Python	120
Ilustración 182. Variables en Python	121
Ilustración 183. Inicio de bucle para detección de mano	121
Ilustración 184. Variables para eje X y eje Y	121
Ilustración 185. Conversión para rango de servos	122
Ilustración 186. Dibujos en la pantalla e instrucciones de cierre	122
Ilustración 187. Movimientos en la mano con puntos de referencia	123
Ilustración 188. Mano robótica doble vista	130
Ilustración 189. Diseño CAD dedo robótico de prueba.....	130
Ilustración 190. Diseño CAD dedo robótico de prueba contraído.....	131
Ilustración 191. Diseño CAD dedor robotico de prueba vista trasnparente	131
Ilustración 192. Diseño CAD dedo robótico de prueba contraído vista transparente	131
Ilustración 193. Diseño CAD dedo robótico de prueba contraído con vista de sección	131
Ilustración 194. Diseño CAD dedo robótico de prueba con vista de sección	132
Ilustración 195.Impresiones 3D dedo robótico de prueba	132
Ilustración 196. Dedo robótico de prueba impreso y armado	132
Ilustración 197. Mano robótica del canal de Will Cogley	133
Ilustración 198. Bosquejo dedo robótico nuevo prototipo.....	133
Ilustración 199. Diseño dedo robótico	133
Ilustración 200. Impresiones 3D eslabones dedo robótico	134
Ilustración 201. Impresiones 3D dedo robótico	134
Ilustración 202. Diseño CAD mano robótica vista lateral.....	134
Ilustración 203. Diseño CAD mano robótica.....	135
Ilustración 204. Diseño CAD dorso	135
Ilustración 205. Diseño CAD muñeca.....	135
Ilustración 206. Diseño CAD antebrazo con motores	136
Ilustración 207. Diseño CAD mano robótica vista acostada	136

Ilustración 208. Impresiones 3D mano robótica	136
Ilustración 209. Maquinado pieza de metal	137
Ilustración 210. Pieza con cortes en taller de maquinado	137
Ilustración 211. Piezas maquinadas terminadas	137
Ilustración 212. Pieza de conexión para muñeca y antebrazo	138
Ilustración 213. Armado antebrazo con motores	138
Ilustración 214. Ensamble motores y antebrazo.....	138
Ilustración 215. Ensamble motores terminado	139
Ilustración 216. Palma con bases para dedos.....	139
Ilustración 217. Muñeca armada	139
Ilustración 218. Palma con muñeca	140
Ilustración 219. Palma con muñeca y antebrazo.....	140
Ilustración 220. Mano robótica armada	140
Ilustración 221. Mano robótica terminada vista frontal	141
Ilustración 222. Mano robótica terminada vista trasera	141
Ilustración 223. Mano finalizada	141
Ilustración 224. Mano sosteniendo objeto delicado.....	142
Ilustración 225. Caja de control	142
Ilustración 226. Marcas para realizar orificios	143
Ilustración 227. Maquinado caja de control	143
Ilustración 228. Maquinado caja de control 2	143
Ilustración 229. Maquinado caja de control 3	144
Ilustración 230. Cableado caja de control	144
Ilustración 231. Fuente de poder.....	144
Ilustración 232. Cargador Kingsman	145
Ilustración 233. Protoboard de cargador Kingsman	145
Ilustración 234. Fuente de poder vista interior.....	146

CAPITULO I. INTRODUCCION

Antecedentes

De la institución

La Universidad Tecnológica de Tijuana fue fundada el 14 de agosto de 1998 por el Gobierno del Estado de Baja California. Esta institución se fundó con la intención de impulsar la educación superior en el estado de Baja California, así también para el preparamiento de profesionistas a nivel Técnico Superior Universitario, mediante la aplicación de un modelo educativo vinculado con el sector productivo y organismos públicos y sociales en la región. Al inicio de la institución se contaba con tan solo 4 carreras: Electrónica y Automatización, Informática, Mantenimiento Industrial y Procesos de Producción. En el año 1999 se añade la carrera de Tecnología Ambiental.

La Universidad Tecnológica de Tijuana inicialmente empezó sus actividades curriculares en las instalaciones de la Universidad Autónoma de Baja California, localizadas en la colonia Juárez en la ciudad de Tijuana, en la que estuvo en funciones desde septiembre de 1998 al mes de agosto de 1999, en estas fechas se trasladó a sus propias instalaciones ubicadas en el fraccionamiento el Refugio Quintas Campestre en el km. 10 de la carretera libre Tijuana-Tecate en la ciudad de Tijuana. El gobierno construyó un edificio de dos pisos para aulas y un segundo edificio para laboratorios y talleres.

En el año 2009 se iniciaron operaciones en el municipio de Ensenada con dos programas educativos. En el 2010 se abrió el programa educativo de Tecnologías de la información y Comunicación.

Actualmente se cuentan con áreas de rectoría, dirección de normatividad y transparencia, secretaría de administración y finanzas, secretaría de vinculación, dirección de extensión universitaria, secretaría académica, subdirección de recursos humanos, subdirección de admisión y servicios al estudiante, subdirección de promoción y difusión, y una coordinación de la unidad académica.

Estas áreas con la finalidad de brindar un mejor servicio para el desarrollo de las actividades académicas de esta institución.

Hoy en día se cuentan con 15 carreras en total en las instalaciones del Refugio y las extensiones de Ensenada y San Quintín. Se cuenta con las carreras de Ingeniería electromecánica industrial, Ingeniería en energías renovables, Licenciatura en innovación de negocios y mercadotecnia, Licenciatura en contaduría, Ingeniería en logística comercial global, Ingeniería en mecatrónica, Ingeniería en procesos y operaciones industriales, Ingeniería en tecnología ambiental, Ingeniería en tecnologías de la información, Técnico superior universitario en manufactura aeronáutica, Técnico superior universitario en gastronomía, Ingeniería en procesos alimenticios, Ingeniería en biotecnología, Técnico superior universitario en lengua inglesa, Licenciatura en diseño y gestión de redes logísticas. Todas estas al alcance de ciudadanos egresados con nivel medio-superior.

Objetivos

Objetivo General

Desarrollar investigación para la selección de una interfaz adecuada para el proyecto “Prototipo de mano robótica controlada por diferentes interfaces”, integrando diferentes tecnologías de programación y software, para obtener datos y manipular dispositivos de accionamiento.

Objetivos Específicos

- Analizar las necesidades de los estudiantes por medio de entrevistas para que la selección de las interfaces sean las adecuadas.
- Diseñar interfaces como muestra de facilidad y uso de la mano robótica.
- Desarrollar los diseños del exoesqueleto, la aplicación de celular y programación para el control y monitoreo de la mano robótica.

Factibilidad

Viendo las condiciones de trabajar con interfaces sencillas de elaborar y poner en uso, se tomarán en cuenta los recursos obtenidos y faltantes para la realización del proyecto, los medios disponibles, el modo de organizar, el tiempo requerido y los gastos directos.

Teniendo como objetivo general el seleccionar la mejor interfaz para la realización de este proyecto, se utiliza como medio disponible el internet. Gracias a esta herramienta se obtiene la información necesaria y vital para realizar investigaciones y proyectos hoy en día, se le da el uso de comunicación y compras en línea, los cuales son indispensables para la elaboración de este proyecto.

Como recurso general se utilizará una computadora, para el acceso al internet y softwares capaces de realizar la programación y comunicación con la mano robótica. Se requiere el uso de software para diseño en 3D, softwares para programar la pantalla HMI y microprocesador, se utilizará una licencia de visión gratuita para la comunicación por medio de gesturas, de igual manera se necesita el acceso a una impresora 3D.

Primeramente, se hicieron compras en línea del material para la impresora 3D, para la elaboración del exoesqueleto y la mano robótica, se consiguieron materiales para la caja de control y su cableado necesario. Se ordenaron algunos potenciómetros, sensores, servomotores y algunas fuentes independientes para estos. Se reciclaron algunos componentes ya utilizados anteriormente en proyectos y se pidieron algunas donaciones para ahorrar una cantidad de dinero considerable.

El tiempo para realizar este proyecto se planificó por medio de un cronograma de actividades y otras herramientas para utilizar el tiempo de manera efectiva. Junto con esto, los gastos se consideraron accesibles y sencillos de

obtener, ya que se utilizan componentes de gama baja, recursos fáciles de obtener y accesibles para una persona que se está introduciendo en este ámbito.

Tabla 1. Tabla de costos (Hardware)

Hardware					
Materiales	Descripcion	Precio p/u	Cantidad	Total	Referencia
PLA	1kg Hatchbox, color negro	\$591.99	1	\$591.99	Amazon
PETG	1 kg Hatchbox, color rojo	\$528.42	1	\$528.42	Amazon
Tornillo M2 5mm	2mmd1 X 5mmL	\$1.16	68	\$78.88	Grupo Torbacal
Tornillo M2 8mm	2mmd1 X 8mmL	\$0.97	36	\$34.92	Grupo Torbacal
Modulo bluetooth	Hc05 arduino	\$92.00	1	\$92.00	Electronica Madrigal
Jumpers	macho-macho, 20cm 40 pz	\$84.62	1	\$84.62	Electronica Madrigal
Indicadores	LED piloto 22mm, 24v, color rojo, verde, amarillo y azul	\$51.60	4	\$206.40	Mercado libre
Boton	boton pulsador N/A	\$35.75	2	\$71.50	Donado
Boton E.S.	boton paro de emergencia N/C	\$124.90	1	\$124.90	Donado
Breaker	breaker con montaje de riel, 6A	\$145.84	1	\$145.84	PET Express
Riel	real para montaje 1m	\$78.48	1	\$78.48	PET Express
Arduino DUE	tarjeta de programacion tipo DUE	\$736.12	1	\$736.12	Steren
Impresora 3D	Ender 3v2	\$7,790.00	1	\$7,790.00	Donada
Cargador	cargador universal 3.1A	\$35.00	1	\$35.00	Steren
Thermofit	color negro 3.2mm	\$5.56	10	\$55.60	Steren
Cable de alimentacion	cable universal para computadora	\$91.76	2	\$183.52	Steren
Transformador 24v	transformador 110V AC a 24v 2A AC	\$209.00	1	\$209.00	Donado
Transformador 3.3v	transformador 110V AC 3.3v 1A CD	\$125.00	1	\$125.00	Mercado libre
Modulo Reles	4 canales, 5v	\$75.90	1	\$75.90	Donado
Capacitor	4700 uF	\$35.32	1	\$35.32	Electronica Madrigal
Cable para bocina	cable duplex bicolor calibre 22	\$4.64	20	\$92.80	Steren
Potenciómetro	3382G-1-252G	\$17.70	12	\$212.40	Electronica Arrow
Caja de control	220 x 250 x100mm metal	\$982.35	1	\$982.35	Donado
Laptop	HP ProbOOK 450 G2	\$3,500.00	1	\$3,500.00	Donado
			Total	\$16,070.96	\$3,393.21

Fuente: Propia

Tabla 2. Tabla de costos (Software)

Software			
Materiales	Descripción	Precio p/u	Referencia
Arduino IDE	IDE para arduino	Codigo abierto	www.arduino.cc
servo.h	librería servo para arduino	Codigo abierto	www.arduino.cc
Firmata	librería StandarFirmata para arduino	Codigo abierto	www.arduino.cc
LabVIEW	Software suite de LabVIEW para estudiantes	Codigo abierto	www.NI.com
NI-VISA	controlador de instrumentos seriales	Codigo abierto	www.NI.com
MySQL	MySQL Community server	Codigo abierto	www.mysql.com
ODBC driver	8.0.23 windows version para conexiones ODBC	Codigo abierto	www.mysql.com
ASO.NET driver	8.0.23 windows version para conexiones con interfaces .NET	Codigo abierto	www.mysql.com
XAMPP	8.0.3 sistema de gestión de base de datos	Codigo abierto	www.apachefriends.org
phpMyAdmin	5.1.0 PHP software para administrar MySQL	Codigo abierto	www.phpmyadmin.net
Solid Works	SOLIDWORKS Student Edition 2020-2021	Codigo abierto	www.solidworks.com
Ultimaker Cura	4.8 windows version impresión 3D	Codigo abierto	www.ultimaker.com
Python	3.9.4 windows version	Codigo abierto	www.python.org
PyCharm	IDE Pycharm community 2021.1	Codigo abierto	www.jetbrains.com
OpenCV	4.5.2 version open source computer vision library	Codigo abierto	www.opencv.org
mediapipe	mediapipe python package	Codigo abierto	www.mediapipe.dev
pyFirmata	interface de python para protocolo Firmata	Codigo abierto	www.pypi.org
App Inventor	Interface en linea para crear aplicaciones	Codigo abierto	www.appinventor.mit.edu
Emulador AI2	MIT appinventor tools 2.3.0	Codigo abierto	www.appinventor.mit.edu

Fuente: Propia

Plan de Trabajo

Para recabar los datos que dará a conocer el contexto preciso y las necesidades específicas de estudiantes en carreras tecnológicas, se realizaron entrevistas a un grupo selecto de estudiantes, tanto de la Universidad Tecnológica de Tijuana como de las demás instituciones cercanas en la región. Para ello se siguieron los siguientes pasos:

- Agendar reuniones con los estudiantes de diferentes instituciones.
- Enfocar las entrevistas a las necesidades de los estudiantes para lograr desempeñar exitosamente la realización de proyectos tecnológicos relacionados a brazos robóticos.
- Presentar la propuesta de interfaces sencillas de manejar y realizar para el control de una mano robótica.
- Definir fechas y acciones posteriores para la realización del proyecto.

Al finalizado de la elaboración de la solución, se organizaron de nuevo reuniones con los estudiantes y se verificó el cumplimiento de los parámetros y expectativas definidas en las entrevistas. Se compararon las solicitudes de parte de los estudiantes con la investigación para determinar si las necesidades se cubrieron de manera correcta.

Tabla 3. Cronograma de Actividades

Actividad	Enero				Febrero				Marzo				Abril		
	Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 1	Sem. 2	Sem. 3
Diseño	Entrevistas														
	Lista de materiales														
	Costo de fabricacion														
	Diseño CAD														
	Programacion														
	Circuito electrico														
Desarrollo	Compra de materiales														
	Materiales donados														
	Instalacion electrica														
	Instalacion exoesqueleto														
	Instalacion de la mano														
	Aplicación														
Cierre	HMI														
	Pruebas														
	Analisis														
	Ajustes														
	Conclusiones														
	Entrega de proyecto														

Fuente: Propia

CAPÍTULO II MARCO TEÓRICO

El marco teórico que se desarrolla a continuación se permite conocer los conceptos necesarios para el entendimiento del desarrollo de este proyecto.

Se empezará con el tema de una interfaz con el fin de comprender la importancia de la comunicación y control de una mano robótica. Se repasarán sus tipos más comunes, posteriormente se describirán los antecedentes de la mano robótica para entender su término y aplicación, a continuación de estos, se definirán los conceptos claves para comprender los elementos utilizados en la investigación de este proyecto.

Interfaz

La interfaz es una demarcación compartida a través de la cual dos o más dispositivos individuales de una computadora intercambian información. Por tanto, es la conexión e interacción entre hardware, software y el usuario. (Sy Corvo, 2021) Los usuarios se comunican con el software, el software se comunica con el hardware o con otro software, el hardware se comunica con otro hardware, todo esto es una interfaz.

La función de una interfaz en el hardware es que las señales electrónicas activan diferentes situaciones. Los datos se escriben, se leen, se envían, se reciben, se comprueban, etc. La función en el software son las instrucciones que activan el hardware a través de protocolos de enlace de datos, métodos de acceso, etc. (Sy Corvo, 2021)

Las interfaces si carecen de usabilidad nadie la deseara. La facilidad con la que alguien utiliza el producto es lo que lograra el objetivo deseado. Se debe considerar la usabilidad inherente de las interfaces para poder comprender y usar el sistema subyacente. La usabilidad debe ser sencilla si se desea que las personas lo usen ampliamente. La interfaz se debe diseñar para que sea intuitiva y familiar, ya que los usuarios después de usar un producto no recordaran

realmente todas las funciones. Para reducir la complejidad, la interfaz debe tener consistencia, además de ser previsible. (Sy Corvo, 2021)

La retroalimentación es clave para el diseño de la interfaz. El producto debe comunicarse con los usuarios brindando una retroalimentación cuando se realice la tarea deseada y sobre qué se debe hacer a continuación. El tiempo de respuesta en la retroalimentación también es un factor clave. Debe ser en tiempo real y de respuesta inmediata, dentro del rango entre 0,1 segundos y 5 segundos. (Sy Corvo, 2021)

Viendo estas características y recomendaciones de una interfaz, se implementarán los 3 tipos de interfaces en el proyecto, interfaz de hardware, interfaz de software e interfaz de usuario. Se utilizarán estos diferentes tipos de interfaces, tanto individualmente como una combinación entre ellos para la realización del más adecuado para cumplir los requisitos y necesidades de los estudiantes.

Mano Robótica

En esencia la definición corta de una mano robótica es “robot articulado”, pues una mano mecánica es construida y elaborada con estructuras de carácter flexible y conformadas por articulaciones adaptables que permiten ejecutar un amplio rango de movimientos y funciones simulando a una mano humana. (RIPIPSA, 2019)

Tiene como principal característica el cumplir las funciones de agilizar actividades, tareas y funciones que requieren una actividad de repetividad y precisión con entornos humanos seguros; durante los procesos de automatización de líneas de producción y manipulación de máquinas en diferentes industrias. (RIPIPSA, 2019)

Para el funcionamiento de una mano robótica generalmente es mediante la ejecución de un conjunto de lenguajes de programación que establecen las principales funciones, además de la incorporación de sensores que consolidan los

objetivos de automatización para poder ejecutar las referencias por: fuerzas, aceleración, temperatura, ubicación, etc., durante la realización de sus actividades. (RIPIPSA, 2019)

Para su funcionamiento las partes o componentes que conforman una mano robótica según sus principales actividades de funcionamiento son:

- Controlador: Se constituye por un microcontrolador con unidad central encargado de calcular los procesos, movimientos y comandos a ejecutar.
 - Actuadores: Corresponde a los motores encargados de generar fuerza para los movimientos.
 - Manipulador: Parte mecánica que cumple la función de realizar los movimientos de la mano mecánica.
 - Muñeca: Es una parte de los robots industriales encargada de realizar los movimientos de elevación, desviación y giro, según el tipo de funciones.
- (RIPIPSA, 2019)

En los antecedentes de las manos robóticas podemos observar que se han desarrollado desde los años 1564 D.C. en París, donde Ambroise publicó su primer diseño de una mano mecánica. Que imitaba a una mano real con músculos mecánicos; este avance avizoraba fronteras de posibilidades inimaginables, solo limitadas por la superstición propia del tiempo en cuestión. (Andrade Zeas & Zuñiga Tenesaca, 2011)

Ilustración 1. Mano mecánica Ambroise (1564)



Fuente: <http://sites.google.com/site/anonymousspynet/cronologia>

Después en el año de 1962 se desarrolla el primer brazo industrial robótico controlado por ordenador con sensores táctiles por HA Ernst. (Andrade Zeas & Zuñiga Tenesaca, 2011)

Ilustración 2. Brazo industrial con agarre tipo mano (1962)



Fuente: <http://www.mundotech.net/historia-del-robot-cronologia-de-hechos-fundamentales/>

En la misma década en el año 1969, con el trabajo conjunto de la Universidad del Sur de California y la Universidad de Novi-Sad en Belgradetiene, crearon el prototipo Belgrade/Usc. Con 4 dedos, cada uno con tres ejes y un grado de libertad que permite la flexión de todas las articulaciones de la unión. La mano es controlada mediante un controlador proporcional derivativo (PD) simple como un microcontrolador. (Andrade Zeas & Zuñiga Tenesaca, 2011)

Ilustración 3. Mano Belgrado/Usc (1969)

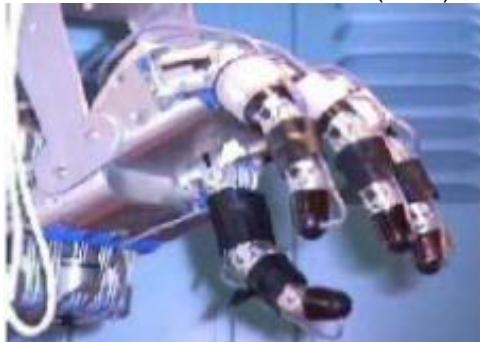


Fuente: www.cenidet.edu.mx/subaca/web-elec/tesis_mc/208MC_cmm.pdf

Después en el año 1982, en el Instituto de Tecnología de Massachussets se desarrolló la mano Utah/Mit con 16 grados de libertad, distribuidos en cuatro

dedos con cuatro grados de libertad cada uno. Cuenta con un antebrazo donde se colocan los actuadores neumáticos que mueven cada una de las articulaciones mediante un arreglo de tendones conducidos por poleas. (Andrade Zeas & Zuñiga Tenesaca, 2011)

Ilustración 4. Mano Utah/Mit (1982)



Fuente: <https://dspace.ups.edu.ec/bitstream/123456789/1068/13/UPS-CT002117.pdf>

Gradualmente se han ido aportando demasiados proyectos de manos robóticas a lo largo de la historia hasta utilizar manos con 24 ejes de movimiento buscando reproducir los movimientos de una mano real lo más cerca posible. Tal ejemplo es la mano Shadow elaborada en el año 2003 en el Reino Unido. (Andrade Zeas & Zuñiga Tenesaca, 2011)

Ilustración 5. Mano Shadow (2003)



Fuente: <https://adquisiciondedatos.wordpress.com/2013/11/13/shadowhand/>

Exoesqueleto

Un exoesqueleto mecánico, robótico o una servo-armadura es una máquina que consiste en un armazón externo que lleva puesto una persona. (Muñoz de frutos, 2017) Con la finalidad de ayudar al portador a realizar movimientos que por el mismo no podría.



Fuente: <https://www.ticbeat.com/salud/exoesqueletos-medicos-un-mercado-de-572-millones-de-dolares-en-2023/>

Impresora 3D

Una impresora 3D es una maquina capaz de imprimir figuras con volumen a partir de un diseño hecho por ordenador. Con volumen quiere decir que tiene ancho, largo y alto. Una impresora 3D lo que realmente hace es producir un diseño 3D creado con el ordenador en un modelo 3D físico.
(www.areatecnologia.com, s.f.)



Fuente: https://m.media-amazon.com/images/I/61yMMy+sCL._AC_SS350_.jpg

PLA

El ácido poliláctico (PLA) es uno de los materiales más populares en la impresión 3D. El PLA es un material eco-amigable y, a diferencia del ABS, es biodegradable. Otra ventaja es su bajo costo y amplia gama de colores, aunque su fragilidad del material no lo hace apropiado para prototipos funcionales. (TRESDE, 2020)

Microcontrolador

Un microcontrolador es un circuito integrado que es el componente principal de una aplicación embebida. Es como una pequeña computadora que incluye sistemas para controlar elementos de entrada/salida. También incluye a un procesador y por supuesto memoria que puede guardar el programa y sus variables (flash y RAM). Funciona como una mini PC. Su función es la de automatizar procesos y procesar información. (E-Marmolejo, s.f.)

Los elementos internos de un microcontrolador son el procesador o microprocesador, periféricos y tipos de memoria. Un procesador incluye al menos tres elementos, ALU, unidad de control y registros. (E-Marmolejo, s.f.)

- ALU. También conocida como Unidad Aritmética y Lógica. Esta unidad está compuesta por los circuitos electrónicos digitales del tipo combinatorios (compuertas, sumadores, multiplicadores), cuya principal función es el realizar operaciones. Estas operaciones están divididas en tres tipos: lógicas, aritméticas y misceláneas.
- Unidad de control. La unidad de control es el conjunto de sistemas digitales secuenciales (aquellos que tienen memoria) que permiten distribuir la lógica de las señales.
- Registros. Los registros son las memorias principales de los procesadores, ya que funcionan a la misma velocidad que el procesador a diferencia de otras memorias un tanto más lentas (como la RAM, FLASH o la CACHE).

Los registros están construidos por Flip-Flops. Los Flip-Flops son circuitos digitales secuenciales.

Los periféricos son los circuitos digitales que nos permiten una interacción con el mundo «exterior» al microcontrolador. Su función es la de poder habilitar o deshabilitar las salidas digitales, leer sensores analógicos, comunicación con terminales digitales o sacar señales analógicas de una conversión digital. Se divide en puertos de entrada/salida, puertos seriales y periféricos analógicos. (E-Marmolejo, s.f.)

Los tipos de memoria se dividen en tres. La memoria para el programa (FLASH), la memoria para los datos o variables del programa (RAM) y la memoria para configuraciones o no volátil (EEPROM). (E-Marmolejo, s.f.)

Aduino DUE

Esta placa Arduino Due guarda grandes similitudes con otras placas de desarrollo Arduino, y su utilidad es exactamente la misma. Es decir, poder crear multitud de proyectos electrónicos y programar diversos sketchs para controlarlos. Pero, al igual que otras versiones de Arduino, tiene sus diferencias notables. (Isaac, s.f.)

Un Arduino Due se basa en chips microcontroladores o MCU como el Atmel SAM3X8E. La primera placa de Arduino en estar basada en ARM, concretamente en el núcleo de procesamiento Cortex-M3 de 32-bit. Un plus de rendimiento sobre las unidades MCU de 8-bit que tienen otras placas similares. Este chip Atmel (actualmente adquirida por la empresa Microchip) inició su serie en 2009 para competir con sus propios AVR. Unos RISC muchos más interesantes y potentes que los anteriores. (Isaac, s.f.)

Además de eso, a grandes rasgos, también tienes más pines, ya que incluye 54 pines digitales de E/S, de los cuales 12 son salidas PWM. También incluye 12 entradas analógicas, 4 UARTs (puertos serie de hardware), etc. Así mismo, a

diferencia de otras placas Arduino, la Arduino Due funciona a 3.3v en vez de los 5v de otras placas. (Isaac, s.f.)

Ilustración 8. Arduino DUE



Fuente: <https://www.dynamoelectronics.com/tienda/arduino-due/>

Las características técnicas de este microprocesador son las siguientes:
(Isaac, s.f.)

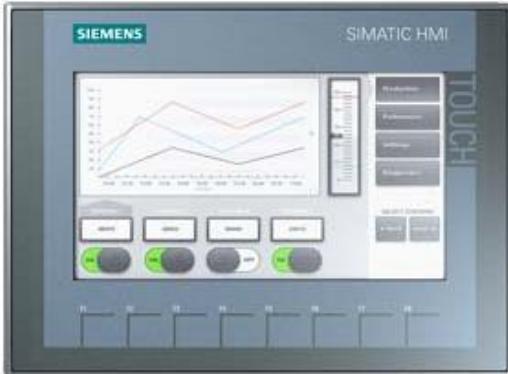
- Microcontrolador: Atmel SAM3X8E ARM Cortex-M3 de 32-bit a 84 MHz
- Memoria SRAM: 96 KB (distribuidos en 2 bancos de 64KB + 1 banco de 32 KB)
- EEPROM: no tiene este tipo de memoria, a diferencia de las otras placas. ARM tiene la capacidad de hacer IAP (In Application Programming) escrito en la flash. Así puede ser usado para almacenamiento de datos no volátiles y código.
- microUSB: tiene 2.
 - Uno de programación (el más cercano al jack de alimentación) para el que tendrás que elegir Arduino Due (ProgrammingPort) en Arduino IDE. Este está conectado directamente al chip 16U2.
 - Otro nativo (el más alejado del jack de alimentación) que se puede usar seleccionando Arduino Due (NativeUSBPort) en Arduino IDE. En este caso está conectado directamente al microcontrolador SAM3X.

- Flash: 512 KB, todos disponibles para programar, ya que no le resta nada el bootloader como en otras placas Arduino.
- Voltaje de operación: 3.3v (aunque posee pin de 5v para tus proyectos, así como GND o tierra).
- Voltaje de entrada (recomendado): 7-12v.
- Voltaje de entrada (límite máximo): 6-16v.
- Pines E/S digitales: 54, de los cuales 12 son PWM.
- Pines de entradas analógicas: 12 canales.
- Pines de salidas analógicas: 2 (DAC).
- Intensidad de corriente por pin E/S: 130mA.
- Intensidad de corriente para pin 3.3v: 800mA.
- Intensidad de corriente para pin 5v: 800mA.
- Peso y dimensiones: 101.52×53.3mm y 36 gramos.

HMI

Se trata de una Interfaz Humano-Máquina, la abreviación se debe por su nombre en inglés: Human-Machine Interface. En sí, es un panel de instrumentos que el operario puede manipular para controlar un proceso. Es la principal herramienta que utilizan los operarios y los supervisores de línea para coordinar y controlar procesos industriales y de fabricación. El HMI traduce variables de un proceso complejo en información útil y procesable. La función principal de los HMI es mostrar información en tiempo real, proporcionar gráficos visuales y digeribles que aporten significado y contexto sobre el estado del motor, la válvula, niveles y demás parámetros de un determinado proceso. Es decir, suministran información operativa al proceso y permiten controlar y optimizar los objetivos de productos y del proceso en sí. Si tuviéramos que mencionar palabras clave que definen el sistema HMI es: operar y observar. (AUTYCOM, s.f.)

Ilustración 9. HMI



Fuente: <https://www.autycom.com/producto/pantalla-hmi-simatic-6av2123-2gb03-0ax0-siemens/>

LabVIEW

Es un software con un entorno de desarrollo integrado especializado en informática industrial y científica. Su particularidad es que se basa en el lenguaje G (G por gráfico), creada por National Instruments que es enteramente gráfica. Permite el desarrollo de programas informáticos complejos facilitando al mismo tiempo la programación y en consecuencia disminuir los plazos de desarrollo. Gracias a sus librerías de funciones dedicadas a la adquisición de datos, la instrumentación, al análisis matemático de las mediciones y la visualización, LabVIEW se dedica especialmente a los bancos de pruebas y mediciones. (Jolly, s.f.)

LabVIEW es especialmente conveniente a la informática industrial y científica, pues puede ser utilizado para el desarrollo de: (Jolly, s.f.)

- Software para Windows, UNIX/Linux o Mac, Windows Mobile o Palm OS,
- Librerías (DLL, Active X, .NET),
- Controles de instrumentos,
- componentes embarcados,
- componentes tiempo real,
- tarjetas FPGA.

Gracias a sus numerosas librerías, LabVIEW puede intercomunicarse con las siguientes tarjetas y protocolos: (Jolly, s.f.)

- VXI, PXI, Compact PCI,
- PCI express, PXI express,
- PCI,
- USB, FireWire,
- serie RS 232, 422, 485...
- TCP/IP, UDP
- Modbus RTU, Modbus TCP, Profibus, otros protocolos industriales...
- Bluetooth, WIFI.

Con LabVIEW no se programa el software a escribir líneas de códigos con una sintaxis compleja. La programación se hace con iconos los cuales representan funcionalidades, ligados entre ellos por cables quienes representan los flujos de datos (un poco a la manera de una tarjeta electrónica con sus componentes y circuitos integrados). (Jolly, s.f.)

MySQL

Es un sistema de gestión de bases de datos de código abierto. Una de sus principales características es que trabaja con bases de datos relacionales, es decir, utiliza tablas múltiples que se interconectan entre sí para almacenar la información y organizarla correctamente. MySQL puede ejecutarse en múltiples plataformas como Linux, Windows, Unix y en un esquema de información para definir y administrar sus metadatos. Se puede instalar en un sistema local o incluso en el servidor. (neoattack, s.f.)

Ilustración 10. MySQL logo



Fuente: <https://www.mysql.com/>

phpMyAdmin

Es un software de código abierto, diseñado para manejar la administración y gestión de bases de datos MySQL a través de una interfaz gráfica de usuario. Escrito en PHP, se ha convertido en una de las herramientas más populares basadas en web de gestión de base de datos. (Cahuana, 2020)

Con esta herramienta se pueden crear y eliminar bases de datos; crear, eliminar y alterar tablas; borrar, editar y añadir campos; ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios y exportar datos en varios formatos. (Hernandez Latorre, s.f.)

Ilustración 11. phpMyAdmin



Fuente: <https://www.phpmyadmin.net/>

XAMPP

Es una distribución de Apache, un servidor independiente de plataforma de código abierto que incluye varios softwares libres. El nombre es un acrónimo compuesto por las iniciales de los programas que lo constituyen: el servidor web Apache, los sistemas relacionales de administración de bases de datos MySQL y MariaDB, así como los lenguajes de programación Perl y PHP. La inicial X se usa para representar a los sistemas operativos Linux, Windows y Mac OS X. (www.ionos.mx, 2019)

Esta herramienta de desarrollo permite probar tu trabajo (páginas web o programación, por ejemplo) en tu propio ordenador sin necesidad de tener acceso a internet.

Ilustración 12. XAMPP logo



Fuente: <https://www.apachefriends.org/es/index.html>

IDE

Un entorno de desarrollo integrado (IDE) es un sistema de software para el diseño de aplicaciones que combina herramientas del desarrollador comunes en una sola interfaz gráfica de usuario (GUI). Generalmente, un IDE cuenta con las siguientes características: (Middleware, s.f.)

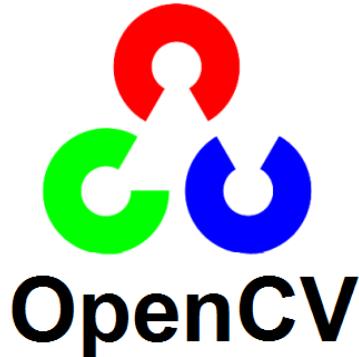
- Editor de código fuente: editor de texto que ayuda a escribir el código de software con funciones como el resaltado de la sintaxis con indicaciones visuales, el relleno automático específico del lenguaje y la comprobación de errores a medida que se escribe el código.
- Automatización de compilación local: herramientas que automatizan tareas sencillas e iterativas como parte de la creación de una compilación local del software para su uso por parte del desarrollador, como la compilación del código fuente de la computadora en un código binario, el empaquetado del código binario y la ejecución de pruebas automatizadas.
- Depurador: programa que sirve para probar otros programas y mostrar la ubicación de un error en el código original de forma gráfica.

OpenCV

OpenCV (Open Source Computer Vision) es una librería open source de visión por computador, análisis de imagen y aprendizaje automático. Para ello dispone de infinitud de algoritmos que permiten, con sólo escribir unas pocas líneas de código, identificar rostros, reconocer objetos, clasificarlos, detectar

movimientos de manos. OpenCV es una librería multiplataforma disponible para Windows, Mac, Linux y Android distribuida bajo licencia BSD. Puede programarse con C, C++, Python, Java y Matlab. (robologs.net, s.f.)

Ilustración 13. OpenCV logo



Fuente: http://robologs.net/wp-content/uploads/2014/04/opencv_logo.png

ApplInventor

ApplInventor es un entorno de desarrollo de software creado por Google para la elaboración de aplicaciones destinadas al sistema operativo de Android. El lenguaje es gratuito y se puede acceder fácilmente de la web. Las aplicaciones creadas con ApplInventor están limitadas por su simplicidad, aunque permiten cubrir un gran número de necesidades básicas en un dispositivo móvil, cualquier usuario puede distribuir sus creaciones libremente. (PROGRAMO, s.f.)

Ilustración 14. ApplInventor logo



Fuente: <https://inventordeaplicaciones.es/app-inventor-desde-0/que-es-app-inventor/>

Como ventajas a la hora de programar con ApplInventor, encontramos las siguientes:

- Se pueden crear aplicaciones por medio de bloques de manera intuitiva y gráfica, sin necesidad de saber código de programación.
- Se puede acceder en cualquier momento y cualquier lugar siempre que estemos conectados a internet.
- Nos ofrece varias formas de conectividad: directa, o wifi o por medio del emulador.
- Nos permite descargar la aplicación mediante el formato apk a nuestro pc.

Potenciómetro 3382G-1-252G

Un resistor eléctrico con un valor de resistencia variable y generalmente ajustable manualmente. Utiliza 4 terminales y se recomienda utilizarse en circuito de poca corriente. El valor del potenciómetro viene expresado en ohmios (símbolo Ω) como las resistencias, y su valor es el máximo que puede representar. Las características de este potenciómetro se muestran a continuación: (Ingenieria Mecafenix, 2017)

- Resistencia: 2.5 KOhm
- Tolerancia: $\pm 30\%$
- Rango de potencia: 0.05W
- Numero de vuelta: 1
- Angulo de rotación: 330°
- Tipo de taper: lineal
- Dirección: horizontal
- temperatura mínima de operación: -40 °C
- Temperatura máxima de operación: 120 °C
- Tipo de shaft: Hueco
- Diámetro del shaft: 4 mm
- Tipo de montaje: Superficial
- Altura: 3.1 mm (DISTRELEC, s.f.)

Ilustración 15. Potenciómetro 252G



Fuente: <https://us-es.alliedelec.com/product/bourns/3382g-1-252g/70408586/>

Fuente de poder

Como fuente de poder se seleccionó un cargador de viaje K3 Kingboard, por su bajo precio y para cumplir los requisitos de los motores. Las características de este cargador se muestran a continuación:

- Voltaje de Entrada: 100-240V AC 50-60 Hz 150mA
- Voltaje de Salida: 5V
- Tamaño del conector: V8 y USB
- Amperaje de salida: 3.1A

Ilustración 16. Cargador K3 Kingboard

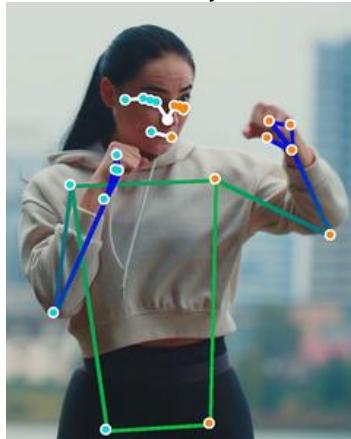


Fuente: <https://www.clasf.mx/cargador-de-viaje-2-puertos-usb-31a-kincboard-v8-en-m%C3%A9xico-9103083/>

Mediapipe

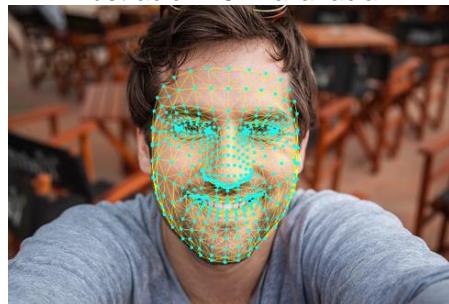
Es un código abierto de google multiplataforma para aplicar multimodal con machine learning. Es capaz de analizar con mayor exactitud los componentes de archivos de video, ya sean en vivo o grabados, audio y datos de serie temporal. Trabaja con Android, iOS, Python y JavaScript. Hasta la fecha, tiene algunos modelos capaces de reconocimiento facial, detección y rastreo de movimiento de manos, detección y rastreo de objetos, detección y estimación profunda del iris, segmentación de cabello y detección y rastreo de objetos 3D. (Google Open Source, s.f.)

Ilustración 17. Detección y rastreo de posición



Fuente: <https://www.mediapipe.dev/>

Ilustración 18. Malla facial



Fuente: <https://www.mediapipe.dev/>

Ilustración 19. Detección y rastreo de objetos



Fuente: <https://www.mediapipe.dev/>

Ilustración 20. Rastreo de manos



Fuente: <https://www.mediapipe.dev/>

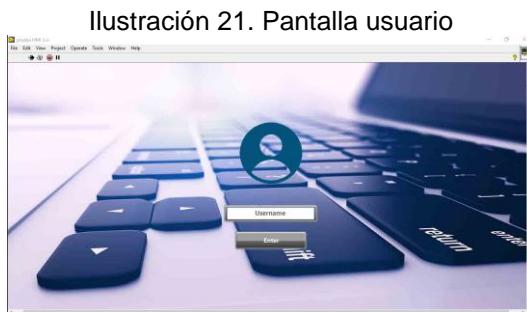
CAPÍTULO III. ESTRATEGIA METODOLÓGICA Y RESULTADOS

Empezando con la elaboración del HMI como principal interfaz para la mano robótica, se explican sus funciones e instrucciones en LabVIEW, tanto el diseño del panel frontal como su programación en el diagrama de bloques, junto con ellos, también se explican la programación e instrucciones para la comunicación serial con arduino y la comunicación con MySQL.

Elaboración del HMI

Se realizó el HMI con la capacidad para lecturas de variables, para recibir valores de los voltajes en los potenciómetros y los grados de las posiciones de los servomotores; de igual manera se le añadió control sobre la mano robótica, indicadores del estado del sistema, indicadores de alertas, gráficas para observar el comportamiento de las señales, animaciones de la mano robótica y una tabla en tiempo real para las alarmas. En la estructura del HMI se utilizan varias pestañas para obtener una mejor observación del comportamiento del sistema y así poder enfocarse en un área específica que se desee.

Inicialmente aparece una pantalla de usuario y contraseña como se observa en la Ilustración 21, limitando el acceso exclusivamente a personal registrado para la operación del HMI.



Fuente: Propia

En el caso de que se intente ingresar con un usuario incorrecto, el programa restringirá el acceso y notificara que el usuario ingresado no está permitido.

Ilustración 22. Usuario incorrecto



Fuente: Propia

Una vez el usuario sea correcto, el sistema pasara a solicitar una contraseña.

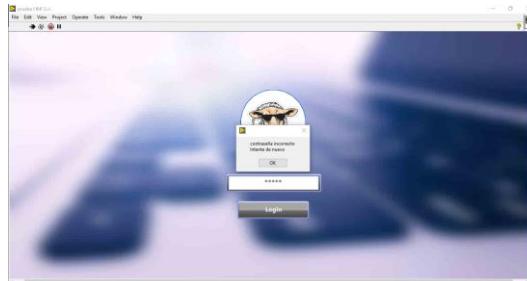
Ilustración 23. Pantalla contraseña



Fuente: Propia

Si la contraseña ingresada es incorrecta, el sistema restringirá el acceso anunciándolo con un mensaje.

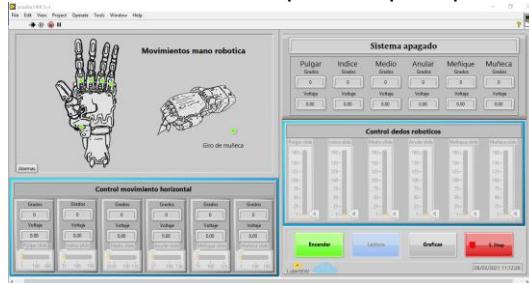
Ilustración 24. Contraseña incorrecta



Fuente: Propia

Cuando se haya ingresado el usuario y contraseña correctos, el HMI pasara a la pestaña principal del sistema.

Ilustración 25. HMI pestaña principal

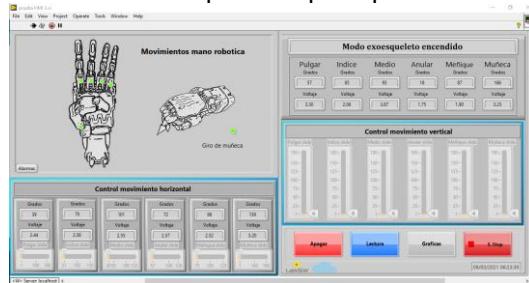


Fuente: Propia

En esta pestaña se pueden observar el estado del sistema, los valores de los voltajes y los grados, sliders para el control de la mano robótica, indicadores de color para reconocer el modo de operación, animaciones del movimiento de la mano robótica, indicadores de alarmas, y botones para el manejo del HMI.

Una vez iniciado el programa, automáticamente se empieza en modo lectura, en el cual solo se están recibiendo datos a través de un puerto serial conectado a Arduino y se muestran las animaciones de la mano robótica para obtener una idea visual de la posición de los dedos en la mano robótica. Los sliders en esta etapa se encuentran en un color gris, para así mostrar al usuario que se encuentran bloqueados en ese instante.

Ilustración 26. HMI pestaña principal modo lectura



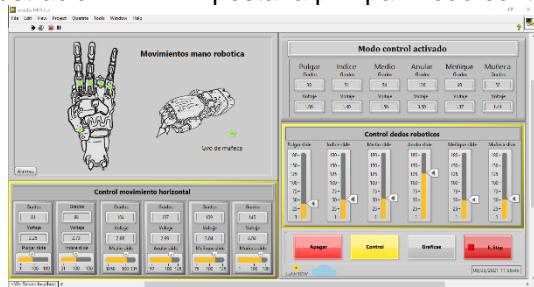
Fuente: Propia

Los datos que se reciben a través del puerto serial se distribuyen en los indicadores del lado izquierdo y derecho de la pantalla. En el lado izquierdo inferior se observan los indicadores para los voltajes y grados de los movimientos horizontales de la mano robótica. En el lado derecho superior se observan los valores para los movimientos verticales de la mano robótica. Los indicadores de

color permanecen en azul facilitando a la vista del usuario que el sistema se encuentra en modo lectura.

Si se presiona el botón de cambio del sistema, el HMI pasa a modo control. En el cual los indicadores de color cambian a amarillo, y en esta etapa, los sliders toman color y se activan, permitiendo al usuario manipular a la mano robótica desde del HMI.

Ilustración 27. HMI pestaña principal modo control

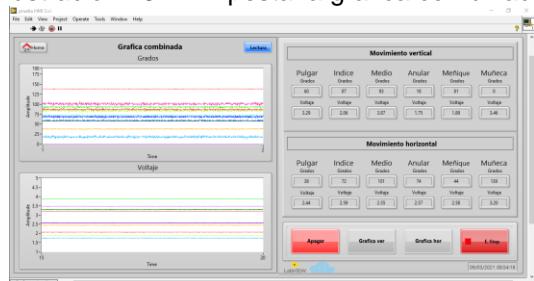


Fuente: Propia

En la parte inferior derecha se observan botones para operar diferentes comandos en el HMI, el primer botón sirve para apagar el sistema y encenderlo; el segundo botón cambia al modo lectura o al modo control dependiendo el modo en el que se encuentre; el tercer botón cambia a las siguientes pestañas, en las que se muestran graficas; y, por último, el 4to botón detiene el sistema por completo como un paro de emergencia.

Presionando el tercer botón se pasa a la siguiente pestaña del HMI, donde se puede observar una gráfica con todos los valores de voltaje y grados que se encuentran en el sistema.

Ilustración 28. HMI pestaña grafica combinada

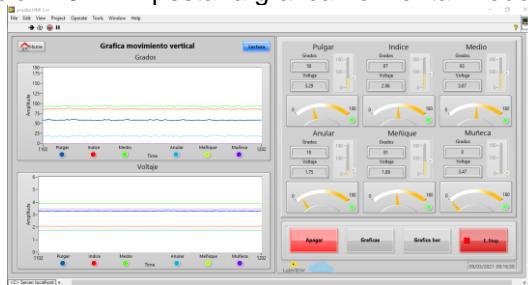


Fuente: Propia

En esta pestaña se pueden observar los comportamientos de las 24 señales del sistema. Pero se tiene la desventaja de ser demasiadas señales en una sola gráfica, por lo que las siguientes pestañas del HMI son graficas específicas de los movimientos vertical y horizontal de la mano robótica. En caso de que se deseé observar señales particulares de la mano robótica, se pasa a la pestaña deseada del movimiento que se seleccione.

Si presionamos el botón para la gráfica de los movimientos verticales, por ejemplo, pasara a la pestaña que se observa en la Ilustración 29.

Ilustración 29. HMI pestaña grafica horizontal modo lectura



Fuente: Propia

En esta pestaña se observan graficas con 6 valores en cada una, desde las señales del pulgar hasta la muñeca. Esta pestaña tiene indicadores para los voltajes y grados de los movimientos verticales. También cuenta con los botones para detener y cambiar de pestañas en el HMI, indicadores de alarmas y un botón para cambiar a modo control si se desea. Y en especial, si se desea ser más específico, se pueden seleccionar la cantidad de señales que se deseen observar.

Ilustración 30. HMI señales específicas



Fuente: Propia

En la pestaña grafica de movimiento horizontal, se observan los mismos indicadores y controles que en la gráfica de movimiento vertical, tan solo cambian los valores que están en operación en el sistema.

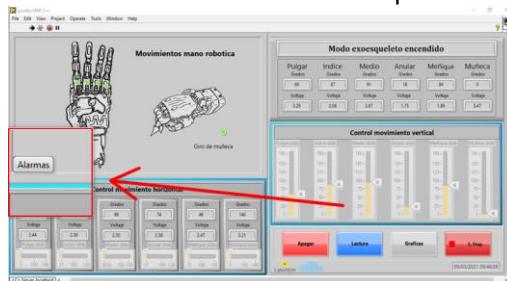
Ilustración 31. HMI pestaña grafica horizontal modo control



Fuente: Propia

Si se llegan a activar alarmas, estas se pueden observar gracias a los indicadores que se encuentran en cada pestaña del HMI. Si se desea observar la tabla de registros, se tiene que regresar a la pestaña principal, en donde se encuentra el botón para direccionar a la pestaña correcta.

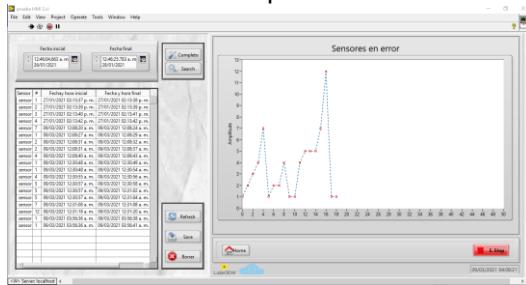
Ilustración 32. HMI ubicación de botón para las alarmas



Fuente: Propia

En esta pestaña se puede observar una tabla con todas las alarmas que se hayan registrado. Estas alarmas se están registrando en tiempo real, y se pueden consultar en cualquier momento. También se cuenta con una gráfica para observar el comportamiento de estas alarmas.

Ilustración 33. HMI pestaña de alarmas



Fuente: Propia

Para la observación de estas alarmas se utiliza el slider a la orilla de la tabla para recorrer todos los datos ingresados. Pero si se desea ser más preciso, se tiene la opción de seleccionar por fechas y hora.

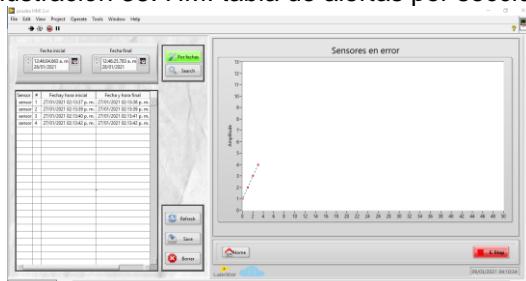
Ilustración 34. HMI cambio a búsqueda por fechas



Fuente: Propia

Se selecciona una fecha inicial y hora en específico, después se selecciona la fecha final y hora del rango que se desea observar, y finalmente se presiona el botón de buscar. Limpiando así la tabla de registros para solo contemplar las alarmas de las fechas o horas requeridas.

Ilustración 35. HMI tabla de alertas por sección

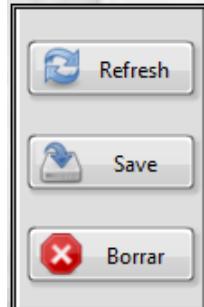


Fuente: Propia

Otra de las opciones que se pueden utilizar en esta pestaña, es salvar los datos en una hoja de Excel, este proceso se realiza por medio de macros y un

vínculo entre Excel y MySQL. También se tiene la opción de borrar la tabla de registros si es que se desea, esto para evitar saturación de datos.

Ilustración 36. HMI botones de control para tabla SQL



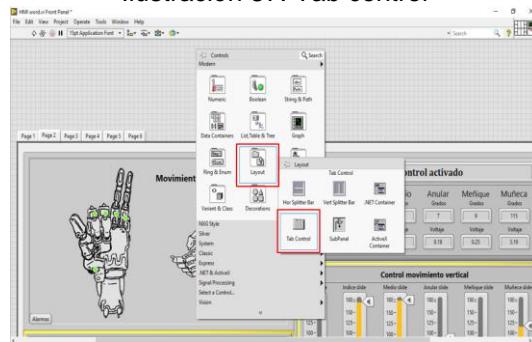
Fuente: Propia

Diseño del panel frontal

El diseño y programación en LabVIEW se divide en dos ventanas: el panel frontal y el diagrama de bloques. En el panel frontal se diseña la interfaz gráfica con la que interactúa el usuario. En el diagrama de bloques se conectan entre sí los elementos añadidos al panel frontal y se hacen operaciones con ellos. Iniciando con el panel frontal, se explica el diseño y los pasos para llevarse a cabo.

Una vez ubicado el mouse en el panel frontal, se dió click derecho para que apareciera la paleta de controles, luego se seleccionó “layout” y en la pequeña ventana que se abrió se encontró el “tab control”.

Ilustración 37. Tab control



Fuente: Propia

Esta herramienta ayuda a insertar varias pestañas para facilitar el cambio de ventanas internas en el HMI con diferentes arquitecturas en cada una de ellas.

En la primera pestaña, se colocó la arquitectura para el funcionamiento de ingreso usuario/contraseña. En la que simplemente se utilizaron LEDs personalizados, controladores tipo string y botones.

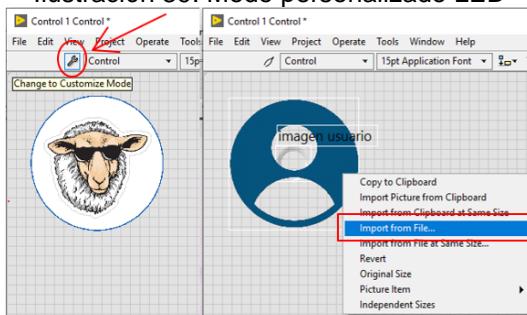
Ilustración 38. Arquitectura pestaña usuario/contraseña



Fuente: Propia

Los LEDs se personalizaron dando click derecho sobre ellos y arrastrando el mouse sobre “advanced” y sobre la ventana subsiguiente se seleccionó “customize...”. Esta abrió una ventana de LabVIEW exclusiva para la modificación del led. Dando click en la herramienta “change to customize mode”, se pudo añadir imágenes desde una ubicación en específico.

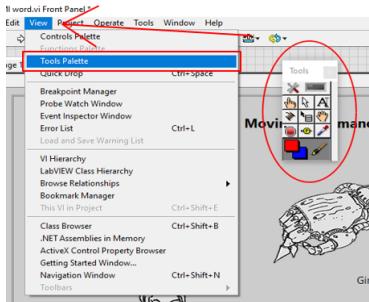
Ilustración 39. Modo personalizado LED



Fuente: Propia

La configuración para el botón y el controlador string que se le puede dar en el diagrama de bloques, es el cambio de tamaño y cambio de color. Los cuales se pudieron modificar desde la paleta de herramientas. Esta se pudo visualizar dando click en “view” en la barra de herramientas, y luego se seleccionó “tools palette”.

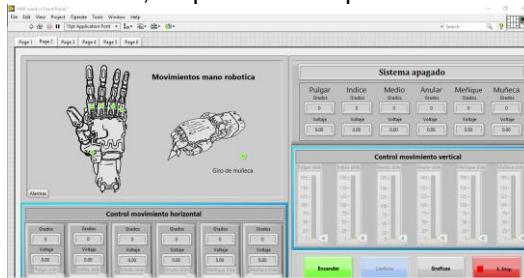
Ilustración 40. Paleta de herramientas



Fuente: Propia

En la segunda pestaña tenemos la arquitectura principal, en la que se añadieron LEDs rectangulares, LEDs personalizados para las animaciones de la mano robótica, indicadores string, indicadores numéricos, sliders y botones. Primeramente, asignando espacios específicos para cada uno de ellos por medio de decoraciones.

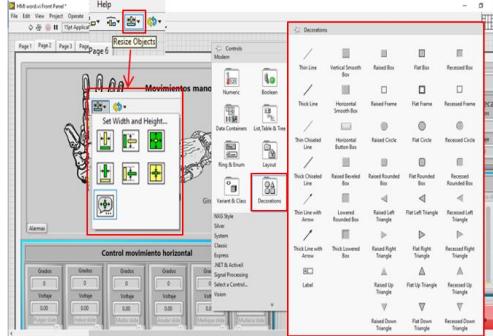
Ilustración 41, Arquitectura de pestaña no.2 HMI



Fuente: Propia

Ubicado el mouse en el panel frontal, se dió click derecho y se seleccionó “decorations” en la paleta de funciones. De esta ventana se logró escoger de una gran variedad de funciones la que se consideró más conveniente. Después se le dió el tamaño deseado con la herramienta “resize objects” para ajustar al espacio asignado.

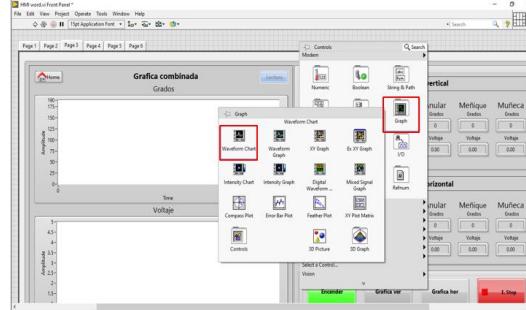
Ilustración 42. Decoraciones pestaña no.2 HMI



Fuente: Propia

Pasando a la siguiente pestaña, se repitieron la mayoría de los pasos para adornar y acomodar las decoraciones deseadas. Se insertaron indicadores, botones, graficas, etc.

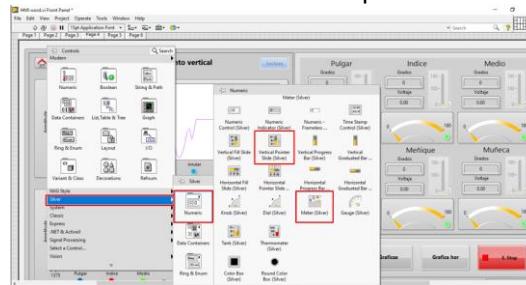
Ilustración 43. Ubicación para herramienta grafica



Fuente: Propia

En la cuarta pestaña, se utilizaron los mismos pasos con la diferencia que en esta pestaña se tienen la mitad de indicadores que en la pestaña anterior, pero se añadieron medidores plateados para la ayuda visual y sliders para el modo control.

Ilustración 44. Medidor plateado

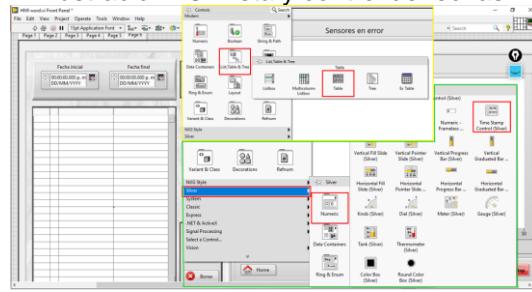


Fuente: Propia

En la quinta pestaña se utilizaron exactamente las mismas decoraciones que en la cuarta pestaña, se añadió la misma cantidad de botones, sliders, medidores, gráficas y LEDs. Tan solo el nombre que aparece en ella y la programación en el diagrama de bloques es la que marca la diferencia.

En la sexta pestaña se siguió el mismo procedimiento, solo añadiendo una lista para las alarmas y los controles para insertar las fechas y hora. La lista se encuentra en la paleta de controles, en el menú despegable dió click en “list, table & tree”. Los controles para insertar fechas se encontraron en la paleta de controles, sobre “silver” » “numeric”.

Ilustración 45. Lista y control de fechas



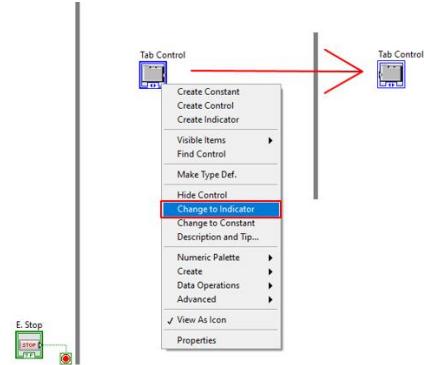
Fuente: Propia

Diagrama de bloques

Cuando se terminó el diseño del panel frontal se continuó la programación en el diagrama de bloques. En el que se asignaron instrucciones y operaciones a realizar en todos los elementos insertados en el panel frontal.

Una vez ubicado el mouse en el diagrama de bloques se insertó un while loop, dentro de esta estructura se colocó la programación que se buscaba repetir hasta que se detuviera por parte del usuario. Después se ubicó el ícono del tab control fuera del while loop, ya que solo se le llama por medio de variables locales. A este ícono se le asignó un cambio de control a un indicador. Ya que se deseaba que este recibiera señales para su funcionamiento, no que este tuviera control sobre otros.

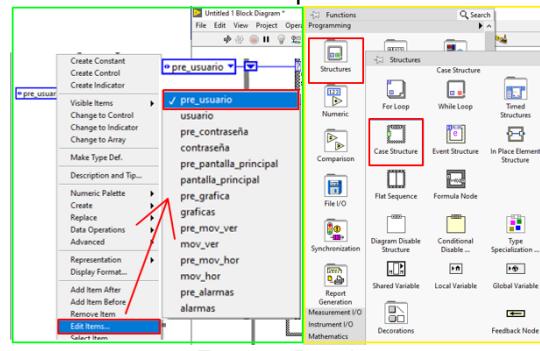
Ilustración 46. Conversión de control a indicador



Fuente: Propia

Una vez hecho esto, se usó una estructura case; la cual se utilizó para convertirla en una máquina de estados. Esta máquina de estados ayudó a cambiar de pestañas conforme se le solicitaba desde el panel de control por medio de botones. Para la elaboración de la máquina de estados se hizo una lista de los casos que habría disponibles para esta función. En el HMI se utilizaron 6 pestañas, pero se usaron 14 estados en total. La lógica de esta programación lleva a utilizar un estado previo antes de pasar al caso que estará siendo utilizado por la pestaña, solamente la pestaña de usuario/contraseña utiliza 4 estados.

Ilustración 47. Máquina de estados



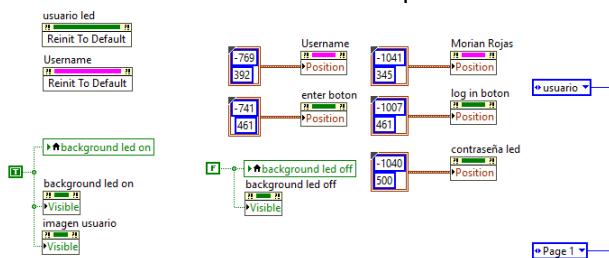
Fuente: Propia

La máquina de estados se creó insertando una estructura case, a continuación, se insertó un “shift register”, y después un “enum constant”, con el cual se creó la lista de estados. Dando click derecho sobre el “enum constant” » “edit ítems...” se abrió una ventana en la que se añadieron la cantidad de casos y

sus respectivos nombres. Una vez terminada la lista, se conectó un “enum constant” al “shift register” y luego este se unió a la estructura case.

Terminada la máquina de estados se empezó la programación de la primera pestaña. Utilizando el primer caso de la máquina de estados, se asignaron las instrucciones que solamente se querían ejecutar una sola vez. Aquí se programaron, por ejemplo, las dimensiones y posiciones de los LEDs, controles y botones que aparecen cuando se enciende el programa, como se observa en la Ilustración 48.

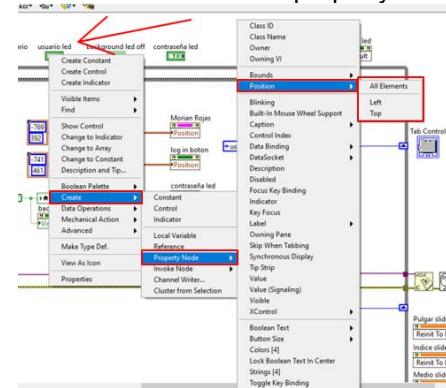
Ilustración 48. Pre-usuario - máquina de estados



Fuente: Propia

Para la inserción de este tipo de propiedades, se dió click sobre el ícono deseado y sobre el menú se seleccionó “create”»“property node” y a continuación se desplegó un menú con todas las propiedades para ese ícono, en donde se seleccionaron las propiedades convenientes.

Ilustración 49. Menú de property node

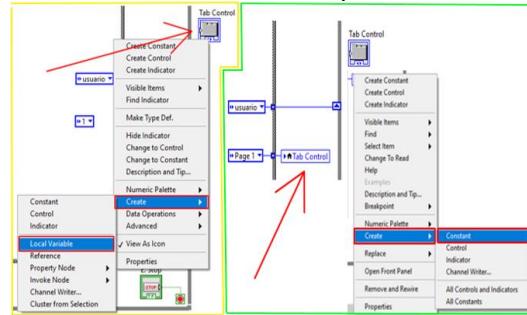


Fuente: Propia

Una vez terminada la configuración de las propiedades de estos iconos, se creó una variable local del “tab control”, esta se generó dando click derecho sobre

el icono para que apareciera el menú, subsiguiente se seleccionó “create” » “local variable”. Esta se colocó fuera de la máquina de estados, para que permaneciera aun cuando se cambiara de estado constantemente. Después se le dió click derecho sobre ella para crear una constante, se seleccionó “create” » “constant”. Con esta acción apareció un “enum constant” ya editado por defecto, este icono se agregó adentro de la máquina de estados y tan solo se escogió el estado que se deseaba en el momento, finalmente se conectó a la variable local del “tab control”.

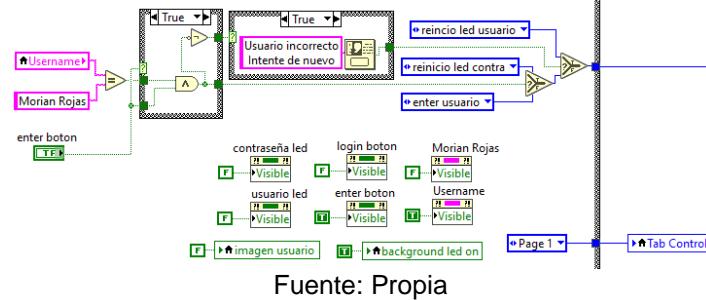
Ilustración 50. Variable local para "tab control"



Fuente: Propia

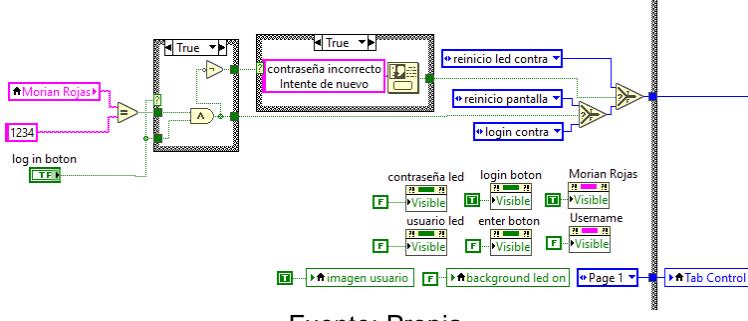
El programa se corre tan solo una vez y la configuración aplicada se lleva a cabo, después pasará al segundo estado. En el segundo estado se ejecutan algunas propiedades de los controles y LEDs, al igual que las operaciones para el ingreso del usuario. Se utilizaron dos selectores, uno para permanecer en el segundo estado hasta que se presionara el botón de enter, y el otro selector se encargaba de enviar de vuelta al primer estado. Si el usuario ingresado era igual al establecido en la programación del diagrama de bloques, se activaba una estructura case que por medio de una compuerta AND enviaba la señal al selector de avanzar al siguiente estado. Si la señal de la compuerta AND era falsa, se activaba la segunda estructura case, en esta se mandaba un mensaje al usuario de que los datos ingresados eran erróneos. Después restauraba los datos por medio del segundo selector para nuevamente ingresar un usuario nuevo, tal como se observa en la Ilustración 51.

Ilustración 51. Configuración estado usuario



Avanzando al tercer caso de la máquina de estados, en la que se aplican los valores predeterminados para el reinicio de la contraseña, esta se ejecutó una sola vez y avanzó al cuarto estado. En este estado se utilizó la misma programación del ingreso de usuario, pero con los datos para la contraseña.

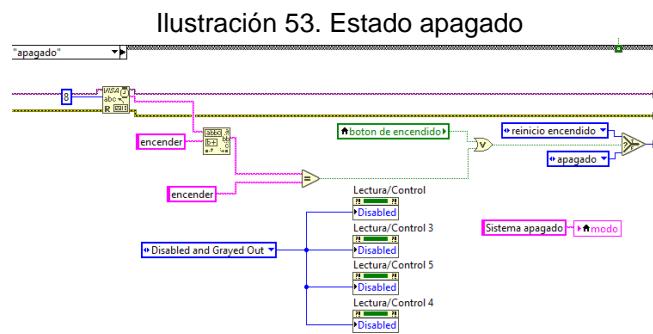
Ilustración 52. Configuración estado contraseña



Estos primeros cuatro estados permanecen en la primera pestaña, solo se están cambiando los casos de la máquina de estados. Pasando al quinto estado, se ingresaron las instrucciones para el reinicio de la pantalla principal. Igual que en todos los estados preparatorios, se usó programación para volver valores por default y dar instrucciones que se deseaban cuando apareciera dicha pestaña. Pasando a la pestaña número seis, se utilizó la programación más compleja, siendo esta la pantalla principal del HMI, donde corren la mayoría de los datos.

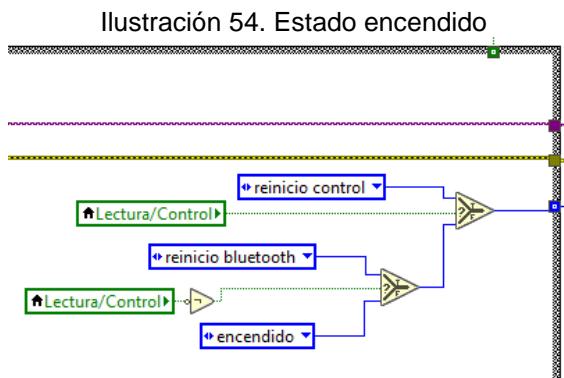
Llegada a esta sección se elaboró una nueva máquina de estados para realizar cambios de encendido, apagado, lectura de variables y control sobre la mano robótica. Siendo 4 estados se multiplicaron por 2 para obtener un estado preparatorio en cada uno de ellos, siendo una lista con un total de 8 estados.

Para el cambio de estos estados se utilizaron selectores, los cuales se activaban con los botones de encendido/apagado y lectura/control. Dependiendo lo que el usuario desee observar en su momento se presiona el botón correspondiente y así el sistema utiliza el estado solicitado. Los estados preparatorios ayudan a activar las propiedades que se deseen antes de pasar al estado que se estará repitiendo continuamente. El estado de apagado se encuentra casi vacío, tan solo está esperando ser llamado para cambiar por el botón de encendido o el string que llega desde arduino, así como se observa en la Ilustración 53.



Fuente: Propia

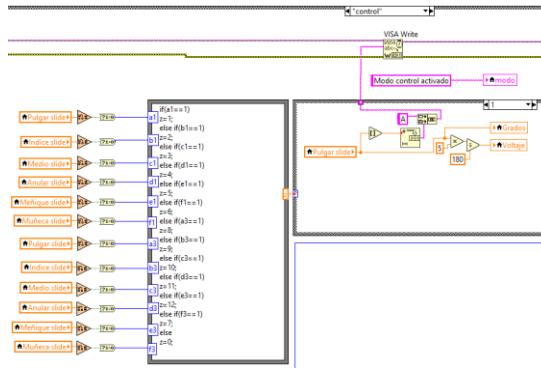
Una vez se pasa al estado de encendido, por defecto el botón de lectura/control se encuentra encendido para la lectura; en el estado de encendido solo se detecta que instrucción es la que se dio, lectura o control. Y sin repetirse el código pasa al estado seleccionado.



Fuente: Propia

La lógica en este estado utiliza un solo botón, al cual se le cambian sus propiedades para que muestre diferente color y texto cuando está apagado y encendido. Si el botón está apagado enviará al sistema al estado lectura, y si está encendido, enviará al sistema al estado de control como se puede observar en la Ilustración 54.

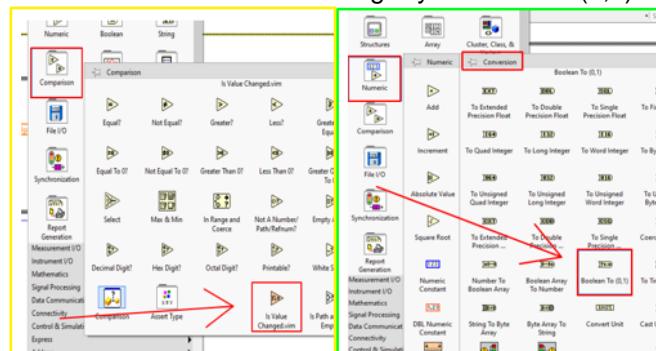
Ilustración 55. Lógica de control HMI



Fuente: Propia

Si el estado se encuentra en modo control, se utilizan variables locales de los sliders para manejar la señal de control, esta variable local se insertó en un “is value changed.vim”; la función de esta herramienta sirvió para detectar el cambio de un valor, en este caso el valor del slider, enviaba en su salida un valor TRUE. Esta señal booleana se conectó a una estructura “formula node”, pero primero necesitaba transformarse a una señal digital (0,1), ya que la estructura “formula node” no acepta entradas booleanas. La estructura “formula node” funciona por medio de programación en C++, tiene una de tantas ventajas el utilizar varias entradas y una sola salida.

Ilustración 56. "Is value change" y "boolean to (0,1)"

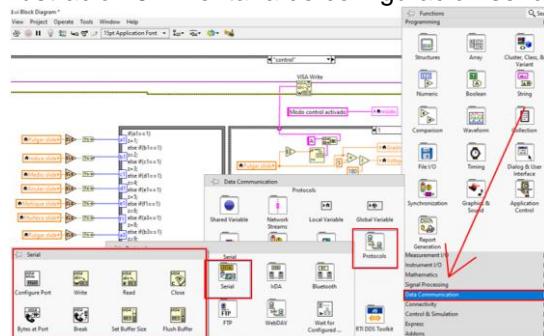


Fuente: Propia

La salida se conectó a una estructura case, la cual se adapta a la programación del “formula node”. Generando 13 casos en total, los cuales se llaman dependiendo la programación en C++. Por ejemplo, si el slider del pulgar se mueve, cambiando su valor, el “formula node” recibe la señal y efectúa una operación por medio de un IF, enviando a la salida un numero 1, este cambia la estructura case a la no.1 y se efectúa la programación interna del case. En este caso, el valor del slider se manda a los indicadores de “grados”, “voltajes” y al puerto serial como se muestra en la Ilustración 55.

A diferencia del estado de control, el estado de lectura no manda caracteres por el puerto serial sino los recibe. Para este tipo de comunicación con arduino se necesitó instalar el archivo de “NI-VISA”, que se puede encontrar en la página oficial. Una vez hecho esto, se agregó a la paleta de funciones los comandos para generar una lectura y envío de datos por el puerto serial.

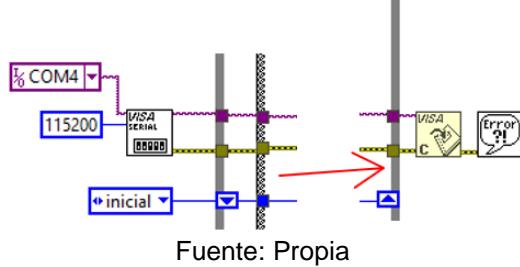
Ilustración 57. Ventana de configuración serial



Fuente: Propia

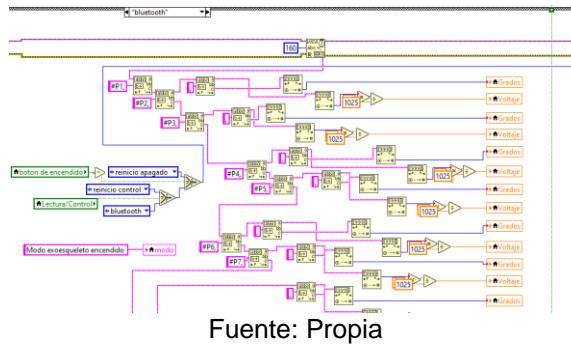
Se inició colocando un “VISA configure serial port” afuera del while loop y configurándolo para la conexión con el arduino. Se seleccionó la velocidad del baud rate y el número de puerto en el que se encontraba conectado el arduino. Del otro extremo del while loop se colocó un “VISA close”, a este solamente se le conectaron los cables de “error in” y “visa resource name”.

Ilustración 58. Configuración puerto serial



Dependiendo si el sistema se encuentra en modo lectura o modo control se utilizan las funciones “VISA read” y “VISA write”. En el modo lectura, utilizando un “VISA read” se reciben los datos de tipo string. Estos se necesitan separar, ya que vienen en una sola cadena de caracteres. Usualmente se utiliza un símbolo entre los datos que se separan, por costumbre se usa la coma (“,”). Pero no es necesario utilizar ese símbolo, el usuario puede escoger cualquier símbolo para la separación de datos en una cadena de caracteres.

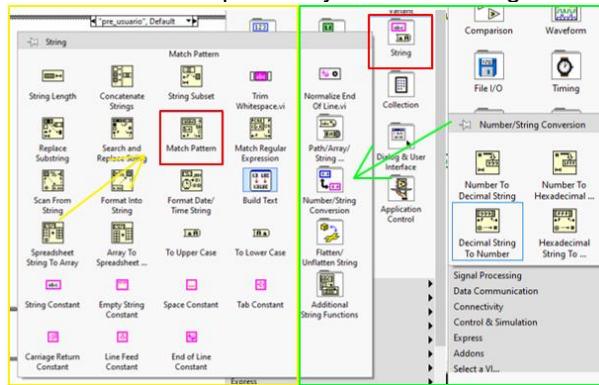
Ilustración 59. Modo lectura VISA read



En la Ilustración 59 se muestra el uso de los caracteres “#P1” para separar por primera vez la cadena de caracteres, consecuente se utilizó “#P2...” y así consecutivamente para seguir separando la cadena. Entre estas cadenas separadas se vuelve a utilizar un símbolo para separar la cadena de nuevo (“,”). Y el resultado de esta división son los datos que ya se deseaban insertar en los indicadores de voltaje y grados, tal como se observa en la Ilustración 59. Pero, primeramente, se necesitaban convertir a dato tipo double. Esta herramienta string “match pattern” se localizó en la paleta de funciones y abriendo la ventana string.

El convertidor de string a decimal se encontró en esta misma ventana en la sección de “number/string conversion”, como “decimal string to number”.

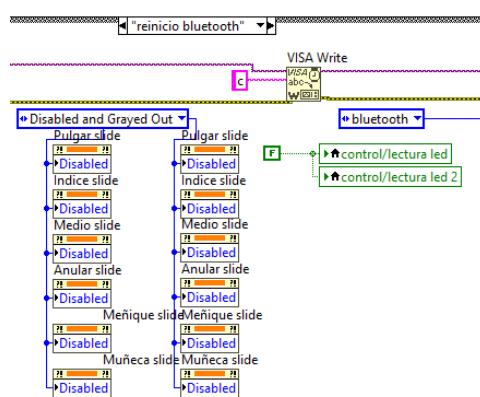
Ilustración 60. "Match pattern" y "Decimal string to number"



Fuente: Propia

En el modo control utilizando un “VISA write” se envían caracteres al arduino. Tan simple como enviar un carácter conectándolo al VISA write o concatenar varios strings para el envío. Se decidió utilizar el envío de caracteres en los pre estados, donde se establecieron las propiedades iniciales de la pestaña o estado que se estaba utilizando. Algunos ejemplos de esta programación se pueden observar en la Ilustración 61 e Ilustración 55.

Ilustración 61. “VISA write” en pre estado

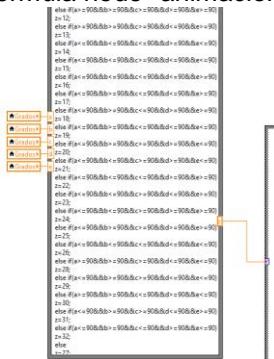


Fuente: Propia

Terminada la comunicación serial, se continuó la programación para las animaciones de la mano robótica. Estas animaciones son LEDs personalizados, esto quiere decir que son imágenes de todas las posiciones de la mano robótica

insertadas en LEDs, sumando un total de 34 imágenes. Para animar estos LEDs se volvió a utilizar la estructura “formula node”. En las entradas del “formula node” se crearon 5 conexiones, estas eran las variables locales de los indicadores pertenecientes a los grados del movimiento vertical. Tal como se observa en la Ilustración 62.

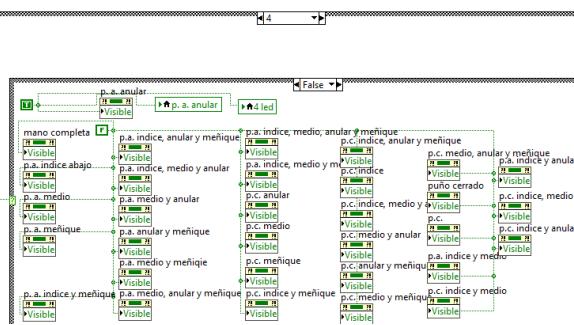
Ilustración 62. "Formula node" animaciones mano robótica



Fuente: Propia

En este “formula node” se utilizaron IF, ELSE IF y ELSE junto con comparaciones de mayor y menor a. Los valores de las entradas tenían un rango de $1^\circ - 180^\circ$ grados por lo que solamente se comparaba menor a 90° y mayor a 90° , ya que las animaciones solo tienen dos movimientos por dedo, tal como se observa en la Ilustración 62. Conectando la salida a una estructura case, se generaron automáticamente los 34 casos.

Ilustración 63. Caso no.4 animación mano robótica



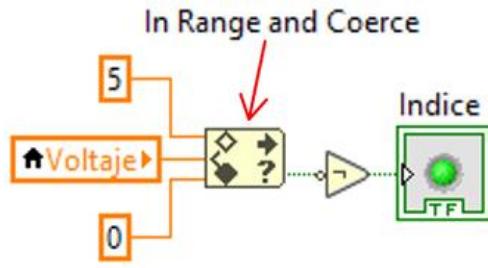
Fuente: Propia

Dentro de estos casos tan solo se enciende un led y los demás se programaron para que no sean visibles. Esta propiedad se encontró dando click

derecho en el mouse en el ícono deseado, y después se seleccionó “create” » “property node” » “visible”.

Pasando a la programación de las alarmas, se utilizaron LEDs de color rojo para indicar que se activaban. La activación de este led se llevó a cabo por medio de la herramienta “in range and coerce”. Esta herramienta compara los valores ingresados en su entrada “x” con los límites especificados en las entradas “upper limit” y “lower limit”. Un ejemplo se observa en la Ilustración 64, en el que el valor “5” es el límite superior y el “0” es el límite inferior, si el valor del indicador de voltaje excede estos límites, se enciende el led. Este led tiene un NOT conectado antes de él, porque la herramienta “in range and coerce” envía una señal TRUE si los valores del indicador se encuentran dentro de los límites. Pero se desea activar el LED cuando estos valores estén fuera de los límites.

Ilustración 64. Lógica para las alarmas



Fuente: Propia

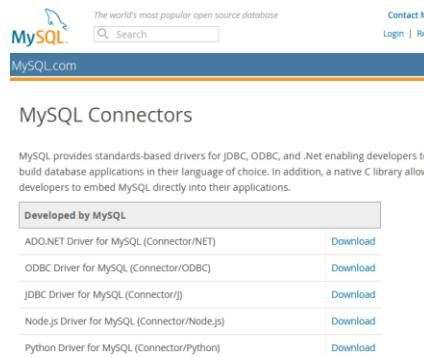
Esta misma lógica se utilizó en todos los indicadores de voltaje y grados. Si alguno excedía los límites programados, el usuario podría observarlo a través de los indicadores. En el caso de que el usuario no observara en su momento estas alarmas, las podía consultar en el registro de alarmas. Las cuales se ingresan en tiempo real a una base de datos a través de MySQL.

Para la programación de la base de datos con LabVIEW se necesitaron descargar los archivos de “ODBC driver for MySQL (connector/ODBC)”. Estos archivos se encuentran en la página oficial de MySQL como se observa en la Ilustración 65. Este driver ayuda a crear un vínculo con el servicio de base de datos de MySQL y LabVIEW. Otro de los requisitos para esta comunicación, es la

instalación del servidor XAMPP. Este servidor local se encargará de varias aplicaciones web si es que se llegaran a necesitar. Si no se llegara a tener acceso internet este servidor ayudaría a probar la programación sin problemas.

Para la instalación de XAMPP se dirigió a la página oficial y se descargó el paquete para el sistema operativo compatible. Este servidor incluye la instalación de phpMyAdmin, que ayuda a la programación de la base de datos. Una vez instalado el servidor, se recomienda revisar el puerto para phpMyAdmin, ya que por defecto este puerto tiene el no. “3306” que es utilizado por otras aplicaciones y puede generar un error al operarse.

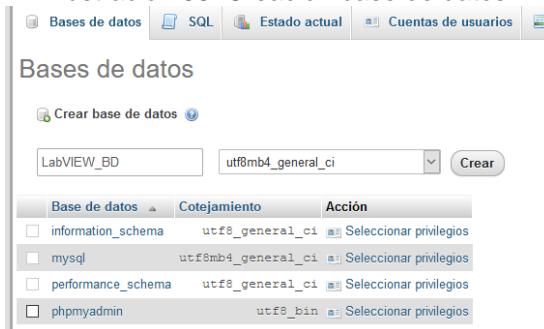
Ilustración 65. Archivo "ODBC driver for MySQL (connector/ODBC)"



Fuente: <https://www.mysql.com/products/connector/>

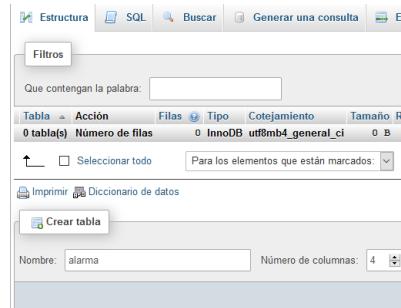
Después de esta configuración, se revisó que phpMyAdmin se pudiera abrir dando click en “admin” en la ventana de XAMPP. Si la fila de phpMyAdmin obtenía un color verde, debería abrir una página en el buscador especificado en la configuración de XAMPP. Una vez ubicado en phpMyAdmin, se creó una base de datos y tablas para el almacenamiento de las alarmas. Primeramente, se creó una base de datos en phpMyAdmin para la inserción de tablas y consecuentemente se creó una tabla con la que se estaría trabajando desde LabVIEW. Ejemplos de esta configuración se pueden observar en la Ilustración 66 y la Ilustración 67.

Ilustración 66. Creación base de datos



Fuente: Propia

Ilustración 67. Creación de tabla



Fuente: Propia

Una vez creada la tabla, se configuraron las columnas que esta tendría. Se le asignaron los tipos de valores que irían en cada una de las columnas. Tal como se observa en la Ilustración 68.

Ilustración 68. Configuración de la tabla

Nombre	Tipo	Longitud/Valores	Predeterminado	Cotejamiento
sensor	VARCHAR	6	Ninguno	
#	INT	5	Ninguno	
fecha_inicial	DATETIME	6	Ninguno	
fecha_final	DATETIME	6	Ninguno	

Fuente: Propia

Ilustración 69. Tabla configurada para inserción de datos

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	sensor	varchar(6)	utf8mb4_general_ci		No	Ninguna		Cambiar Elim.	
2	#	int(5)			No	Ninguna		Cambiar Elim.	
3	fecha_inicial	datetime(6)			No	Ninguna		Cambiar Elim.	
4	fecha_final	datetime(6)			No	Ninguna		Cambiar Elim.	

↑ □ Seleccionar todo Para los elementos que están marcados: Examinar Cambiar Eliminar Pŕ

Texto completo Agregar a columnas centrales Eliminar de las columnas centrales

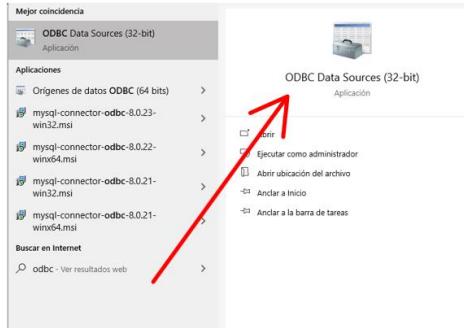
Imprimir Planamiento de la estructura de tabla Hacer seguimiento a la tabla Mover columnas Normalizar

Agregar 1 columna(s) después de fecha_final Continuar

Fuente: Propia

Ya creada la tabla con sus respectivas columnas, se continuó a la configuración de ODBC data source. Donde se configuró la conexión para vincular con LabVIEW.

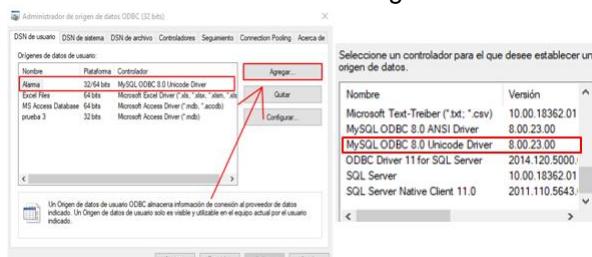
Ilustración 70. Buscador ODBC source data



Fuente: Propia

Se abrió ODBC data source y se le dió click en agregar. Después se escogió la opción “MySQL ODBC 8.0 Unicode Driver”.

Ilustración 71. Administrador de origen de datos ODBC

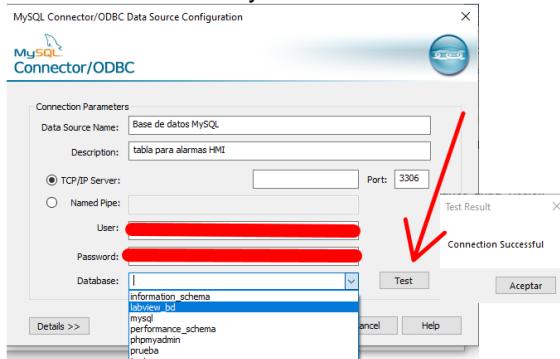


Fuente: Propia

Seguido a esto apareció una ventana para configurar la conexión con la base de datos, se escribió un nombre que se deseaba dar a la conexión, una descripción de esta conexión, el puerto que estaba utilizando XAMPP, el usuario y

contraseña, y se seleccionó la base de datos de phpMyAdmin. Al final se presionó la tecla “test” para verificar que la conexión fuera correcta, si tal era el caso, aparecía una pequeña ventana que informaba que la conexión había sido exitosa. Se le dió ok para terminar, una vez creada, se añadió al administrador de origen de datos de ODBC, tal como se observa en la Ilustración 73.

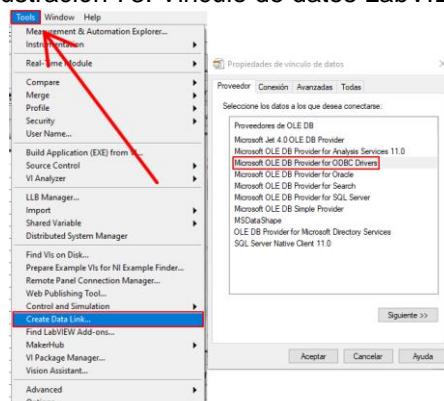
Ilustración 72. MySQL connector/ODBC



Fuente: Propia

Ya que se obtuvo la configuración de ODBC, se pasó a configurar en LabVIEW para generar el vínculo con ODBC. Se dió click en “tools” » “create data link” y esta abrió una ventana en la que se seleccionó la opción “Microsoft OLE DB provider for ODBC drivers”.

Ilustración 73. Vinculo de datos LabVIEW

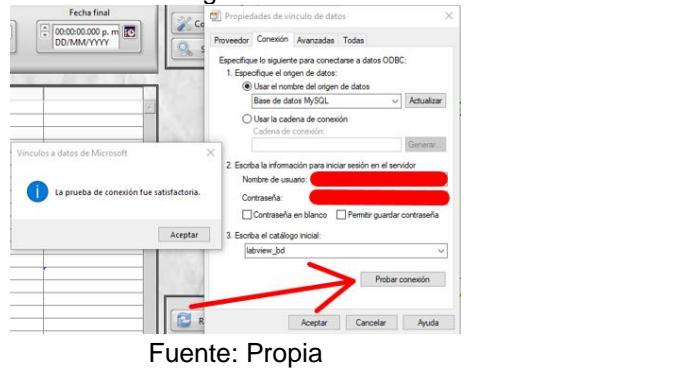


Fuente: Propia

Se llenaron los datos necesarios al igual que en la configuración de ODBC drivers, se hizo la prueba de conexión y al momento de ser exitosa se dio click en

aceptar. Esta conexión creó un archivo de tipo “UDL” al que se le asignó un nombre.

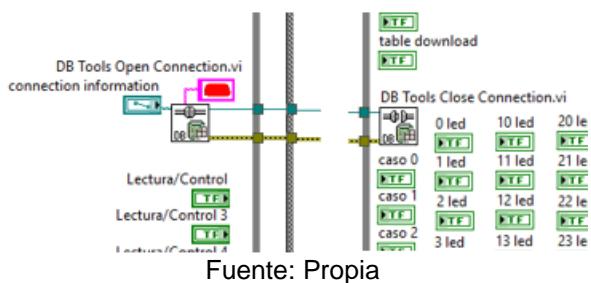
Ilustración 74. Configuración vínculo de datos



Fuente: Propia

Una vez creado el vínculo de LabVIEW con phpMyAdmin, se pasó a la programación de LabVIEW. Se creó la entrada y salida de conexión por medio de “DB tools open connection.vi” y “DB tools close connection.vi”. En la función de entrada se creó un control para la herramienta “connector information” en el que se insertó el archivo “UDL” que se creó por medio del vínculo de datos de LabVIEW. En la entrada de “user ID” se creó una constante en la que se ingresó el nombre del usuario para phpMyAdmin. En el otro extremo se creó un “DB tools close connection.vi”. tal como se observa en la Ilustración 75.

Ilustración 75. Programación entrada y salida DB tools



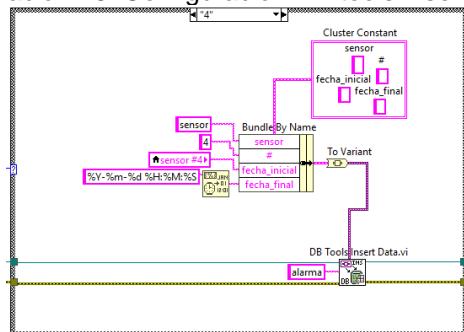
Fuente: Propia

Después se creó la inserción de alarmas en la tabla. Ya que se estaba ubicado en el case donde se registraba la activación de alarmas, se creó un “DB tools insert data.vi”. A esta herramienta se le conectó en la entrada “table” una constante de tipo string que contenía el nombre de la tabla creada en phpMyAdmin. Para la entrada que dice “cluster” se creó un “bundle by name”, al

que se le insertaron los valores que se deseaban ingresar en la fila siguiente de la tabla, con el orden de las columnas creadas en la tabla en phpMyAdmin. Para la fecha inicial y fecha final se utilizó un “format date/time string” que sirve para insertar la fecha y/u hora con el formato deseado. Una vez insertados los valores al “bundle by name”, a esta herramienta se le insertó una constante de tipo cluster.

Adentro de esta constante se insertaron constantes de tipo string en el orden correspondiente y con el nombre de las columnas de la tabla creada en phpMyAdmin. Tal como se observa en la Ilustración 76.

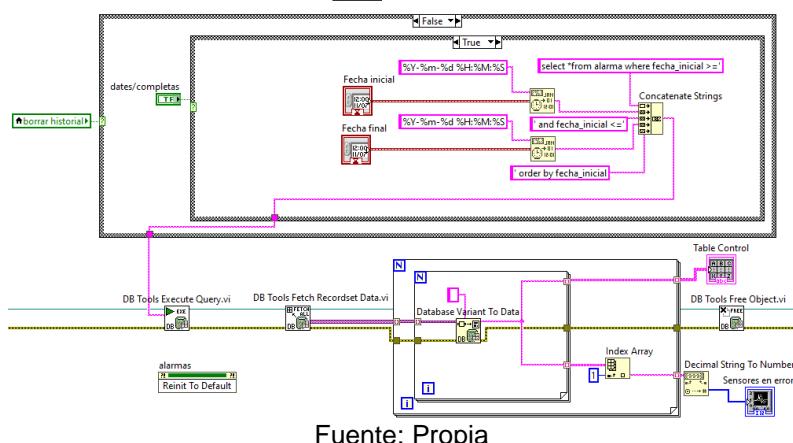
Ilustración 76. Configuración DB tools insert data



Fuente: Propia

Esta inserción se hace cada vez que el programa corra este caso, se buscó la manera que solo se efectuara una vez, en la configuración anterior se guardaba el valor de la fecha inicial mucho antes de llegar al caso que se muestra en la Ilustración 76 , se insertó una vez que la alarma se apagaba, que es el tiempo en el que se utilizaba el caso de ejemplo de la Ilustración 76. Esta configuración solo se utilizó para insertar datos en la tabla de phpMyAdmin. Si se deseaba consultar estos datos, el usuario debía dirigirse a la pestaña de alarmas donde se efectuaba la siguiente configuración.

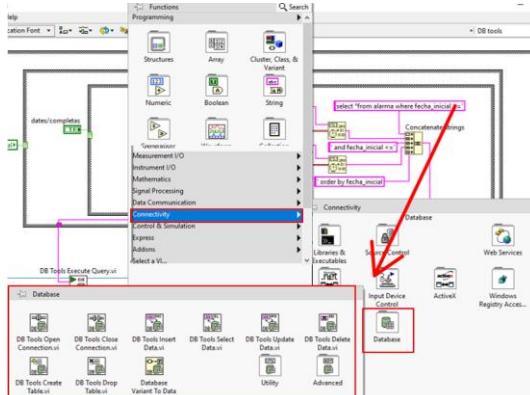
Ilustración 77. Programación para consulta de datos



Fuente: Propia

En esta parte de la programación se utilizaron 4 herramientas de la paleta “DB tools”. Las cuales se encuentran en la paleta de funciones dando click en “connectivity” » “database”, como se observa en la Ilustración 78.

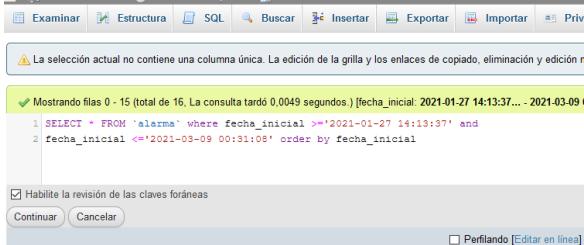
Ilustración 78. Database funciones



Fuente: Propia

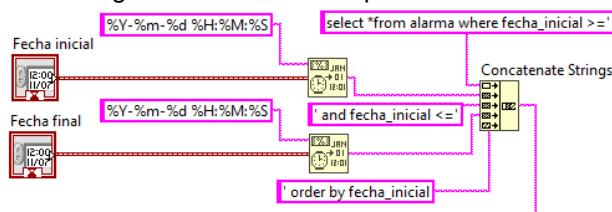
Para la herramienta “DB tools execute query.vi”, se insertó en su entrada, por medio de dato string, la programación necesaria SQL. En esta sección de la programación se apoyó en el editor en línea de phpMyAdmin. Donde lo mismo que se escribía en el editor en línea era lo mismo que se necesitaba escribir en la entrada de esta herramienta. Pues lo que se estaba efectuando, era la trasferencia de la programación escrita en LabVIEW al editor en línea de phpMyAdmin. Tal como se observa en la Ilustración 79 y en la Ilustración 80.

Ilustración 79. Editor en línea de phpMyAdmin



Fuente: Propia

Ilustración 80. Programación LabVIEW para insertar en editor en línea



Fuente: Propia

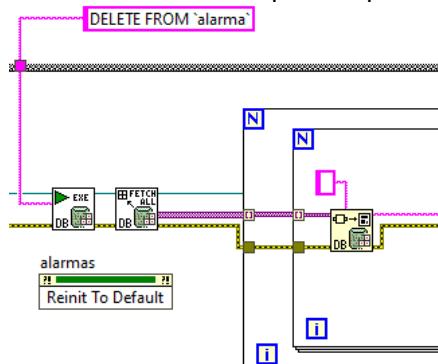
Para la programación SQL se puede apoyar dirigiéndose a la página oficial de w3schools, donde vienen varios lenguajes de programación de manera gratuita.

Siguiendo con la herramienta “DB tools fetch recordset data.vi”, esta se conecta a una estructura “For Loop”. En la cual se utilizó otra herramienta llamada “Database variant to data”, esta tiene una entrada a la que se le conectó una constante de tipo string. La salida de esta herramienta se conectó a la tabla creada en el panel de control y a un “index array” para poder insertarse en la gráfica. Por último, se utilizó la herramienta “DB tools free object.vi” dentro del while loop, a la que solamente se le conectó el cable de error y el cable de “recordset reference”, esta conexión esta en paralelo con la herramienta “Database variant to data”, no pasaba dentro del “For loop”, tal como se observa en la Ilustración 77.

Pasando a la programación para borrar la tabla de registros en Excel. En la pestaña de alarmas se encuentra un botón de “delete”, que ejecuta un cambio de caso. El cual envía al editor en línea de phpMyAdmin el siguiente comando:

“DELETE FROM ‘alarma’”, este se encarga de limpiar los datos. Tal como se observa en la Ilustración 81.

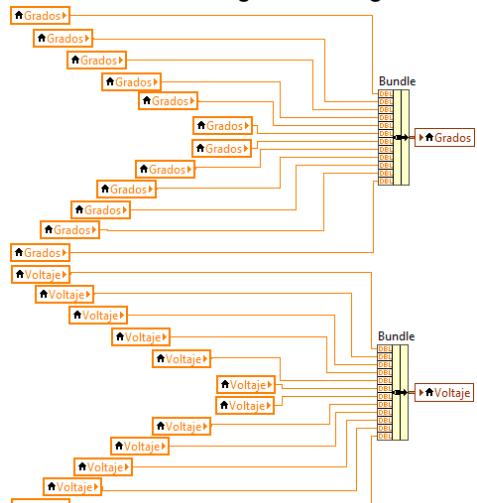
Ilustración 81. Comando para limpiar tabla



Fuente: Propia

Terminada la sección de SQL, se continua a las pestañas de gráficas, en las que simplemente se insertaron los valores de los indicadores en la gráfica colocada en el panel frontal, por medio de la herramienta “bundle” como se observa en la Ilustración 82.

Ilustración 82. Programación graficas

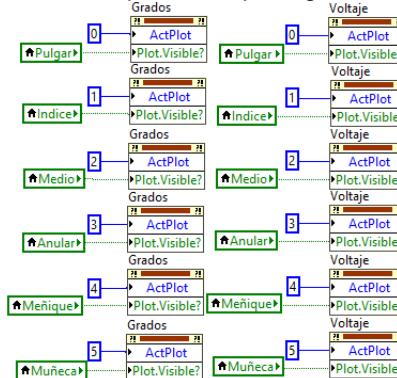


Fuente: Propia

En las siguientes pestañas donde se tienen las gráficas específicas, se utilizó una propiedad para desaparecer señales en la gráfica, en caso que solo deseara observar cierta señal. Tal como se observa en la Ilustración 83, se utilizaron propiedades de la gráfica en la que se deseaban controlar sus señales.

Estas propiedades se encuentran dando click derecho del mouse sobre la gráfica y se selecciona “create” » “property node”. Y se seleccionan las propiedades “ActPlot” y “Plot.Visible?”, donde “ActPlot” sirve para indicar que numero de señal es la que se esté configurando, y “Plot.Visible?” para que no sea visible.

Ilustración 83. Propiedades para grafica especifica



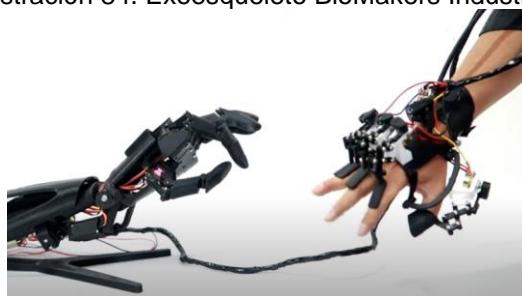
Fuente: Propia

Para los medidores plateados, se conectaron los valores de los indicadores en paralelo. La programación de los sliders es la misma que en la pestaña principal repitiéndose los mismo pasos.

Elaboración del exoesqueleto

Investigando en internet, se encontró con el canal de BioMakers Industries en YouTube. Este canal tiene varios proyectos de manos robóticas con diferentes tipos de control. Se tomó la idea del proyecto “Mano robótica controlada con exoesqueleto”, el cual se usó como apoyo para realizar el exoesqueleto a continuación.

Ilustración 84. Exoesqueleto BioMakers Industries



Fuente: Propia

Una vez obtenida la idea de cómo realizar las partes del exoesqueleto, se empezaron hacer piezas de cartón para obtener una mejor idea del movimiento mecánico que este tendría.

Ilustración 85. Prueba exoesqueleto con cartón



Fuente: Propia

Ilustración 86. Mediciones para piezas de cartón



Fuente: Propia

Ilustración 87. Dorso y muñeca de cartón

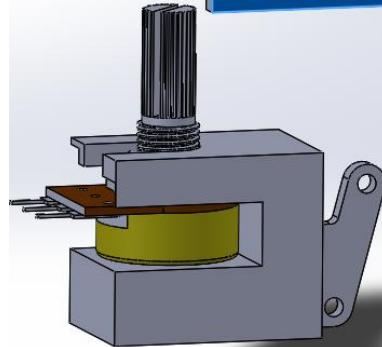


Fuente: Propia

Una vez realizadas las piezas de cartón con las medidas para que estas ajustaran bien en el brazo, se empezó a dibujar en SolidWorks.

Iniciando por la base de los potenciómetros logarítmicos, se descargó de GrabCAD.com un archivo de potenciómetro logarítmico, para utilizar en el diseño de SolidWorks y acercarse a las medidas deseadas.

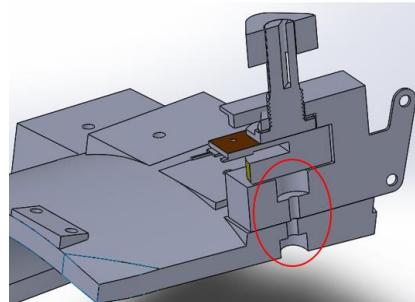
Ilustración 88. Diseño CAD base potenciómetro



Fuente: Propia

Sobre un sketch se hizo un rectángulo y se utilizó la herramienta “extruir” para generar una pieza. Después se hizo un corte en medio para que pudiera acomodarse el potenciómetro. Por debajo del potenciómetro se utilizó la herramienta “asistente de taladro”, que permite utilizar un tornillo para conectar la base del potenciómetro con la base de los nudillos. Tal como se observa en la Ilustración 89.

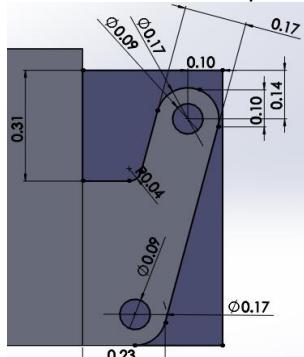
Ilustración 89. Base nudillos con vista de selección



Fuente: Propia

Sobre la cara trasera de la base se hizo un sketch para crear la base de conexiones que se utilizan para flexionar los dedos. Después se hizo un corte para darle la forma requerida.

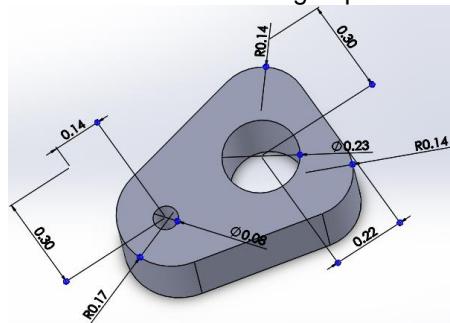
Ilustración 90. Corte base de potenciómetro



Fuente: Propia

Siguiendo con la siguiente pieza, solo se hizo un sketch en el que se le dio la forma parecida a un triángulo con las orillas circulares, y en el centro dos círculos y utilizando la herramienta “extruir” se terminó la pieza.

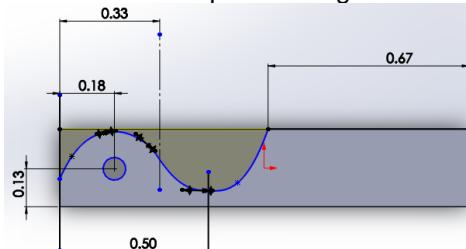
Ilustración 91. Conexión con giro potenciómetro



Fuente: Propia

La siguiente pieza se realizó con la figura de un rectángulo, pero con el corte de media luna para permitir la entrada de la base giratoria del potenciómetro.

Ilustración 92. Pieza de entrada para base giratoria del potenciómetro

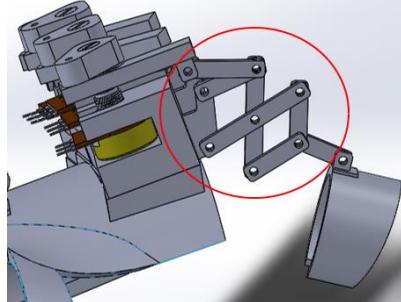


Fuente: Propia

Después de esto, se realizaron los dibujos 3D de las piezas que ayudaron a flexionar los dedos. Para las medidas de este sketch, se utilizaron las mismas de

las piezas de cartón, pues una vez terminadas las pruebas, estas demostraron tener el movimiento adecuado.

Ilustración 93. Piezas de flexión para los dedos

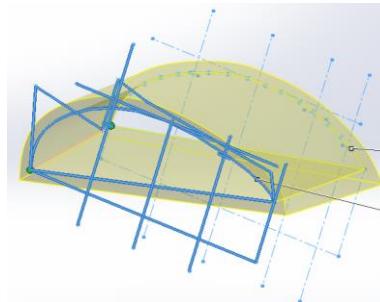


Fuente: Propia

Todas estas piezas, al igual que la base del potenciómetro y sus conexiones, se copian para completar todos los dedos. La única diferencia es la última pieza de conexión que esta junto con el anillo que se insertara en los dedos. Esta pieza se hace más larga dependiendo el dedo de la mano.

Siguiendo con la base de los nudillos, se creó un sketch y se dibujó una tipo medio luna. Dándole la forma necesaria para acomodarse sobre el dorso de la mano. Enseguida se creó un plano en paralelo a este sketch, y se hizo el mismo dibujo. Una vez hecho esto se utilizó la herramienta “saliente/base por límite” junto con la función “operación lámina”. Después solo se hizo el corte necesario para borrar el exceso inferior.

Ilustración 94. Diseño CAD dorso

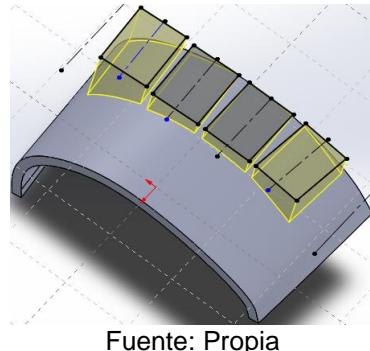


Fuente: Propia

Una vez hecho el dorso, se añadieron las bases para conectar con las piezas de los potenciómetros. Creando un plano superior solo se hicieron

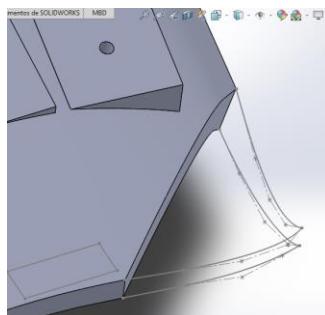
rectángulos a las medidas de las bases de los potenciómetros y se extruyó con la función “hasta la superficie”, uniendo así a la cara del dorso, tal como se observa en la Ilustración 95.

Ilustración 95. Diseño CAD base nudillos



Fuente: Propia

A la orilla derecha se hizo un corte semi triangular para el movimiento del pulgar, para que la superficie no le estorbara al momento de flexionar el pulgar. Después se hicieron planos de forma perpendicular para crear una nueva superficie que permitiera el movimiento del pulgar sin que este se viera limitado.

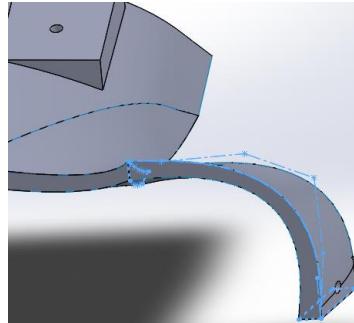


Fuente: Propia

Utilizando las herramientas “rellenar superficies” y luego la herramienta “recubrir superficie” se definió la parte faltante dándole una forma con curvatura para la flexión del pulgar.

Siguiendo esta curvatura, se hizo otro plano transversal a su extremo, con la intención de hacer un dibujo y utilizar la herramienta “recubrir superficie”. Para el extremo de la curvatura se creó un croquis3D ya que la curvatura no permite utilizar un croquis normal.

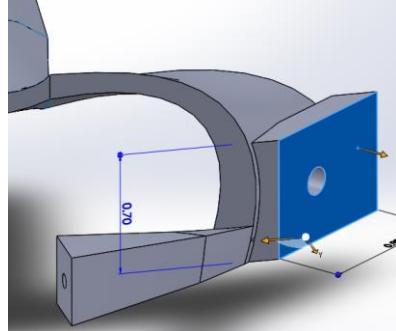
Ilustración 96. Diseño CAD curvatura del pulgar



Fuente: Propia

Después se crearon planos para la elaboración de la base para el potenciómetro en el pulgar. Y con la herramienta “recubrir superficie” se crea una extensión lateral para la parte del dorso que se conecta con la muñeca, como se muestra en la Ilustración 97. Este tiene la función de servir de unión con la muñeca.

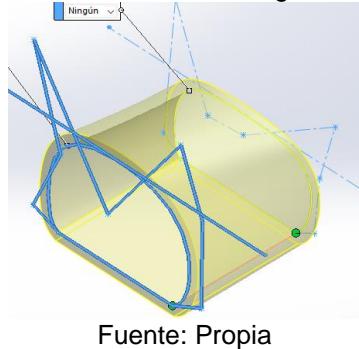
Ilustración 97. Diseño CAD base para el pulgar



Fuente: Propia

Siguiendo el diseño, se pasa a crear la muñeca. En esta sección se repite el proceso para la creación del dorso. La única diferencia es que un dibujo tiene mayor longitud, debido a que la parte que cubrirá de la muñeca es más grande que el otro extremo.

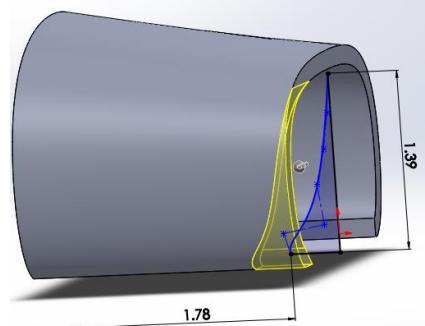
Ilustración 98. Diseño CAD configuración muñeca



Fuente: Propia

Siguiente se hizo un corte para quitar el exceso inferior y también se hizo un corte en la parte frontal en forma similar a un triángulo. Esto para permitir el movimiento de la muñeca sin afectarla, tal como se observa en la Ilustración 99.

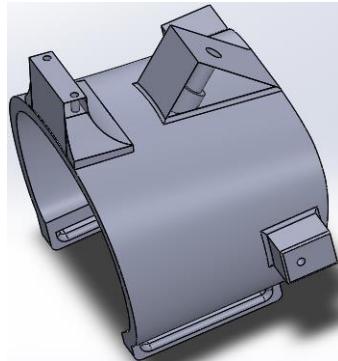
Ilustración 99. Diseño CAD corte en muñeca



Fuente: Propia

Adelante se hicieron planos para la creación de figuras que ayudarían a las conexiones con las demás partes del exoesqueleto. Se hicieron los cortes para las diferentes medidas de tornillos que se usarían y cortes para el uso de un cinto para sostener la pieza sobre la muñeca.

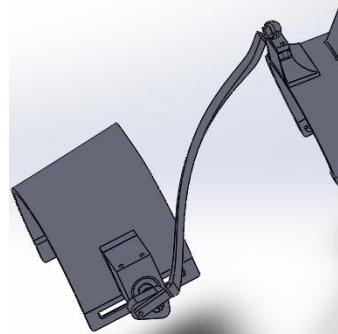
Ilustración 100. Diseño CAD muñeca



Fuente: Propia

Siguiendo con la parte trasera del antebrazo. Se utilizaron las medidas de la pieza elaborada de cartón, y se utilizó el mismo procedimiento que en las piezas anteriores con la herramienta “saliente/base por límite”.

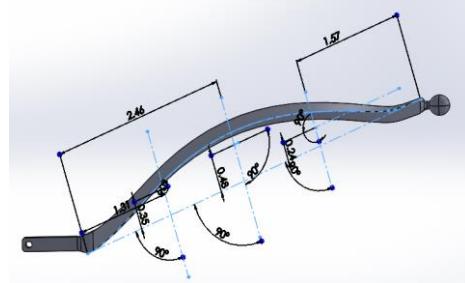
Ilustración 101. Diseño CAD unión antebrazo y muñeca



Fuente: Propia

Una vez creada esta pieza, se hizo la base de conexión para el potenciómetro y poder atornillarlo. Después se trabajó en la pieza que recorrería el antebrazo para unir con la muñeca.

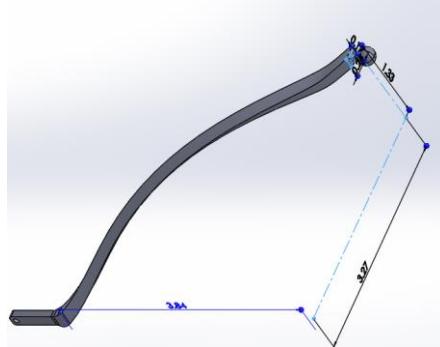
Ilustración 102. Diseño CAD pieza unión antebrazo



Fuente: Propia

Para esta pieza se utilizaron planos a distancia tanto superior como lateral para colorar el croquis de un extremo con respecto al primer croquis. Tal como se observa en la Ilustración 103.

Ilustración 103. Diseño CAD distancias para pieza de unión

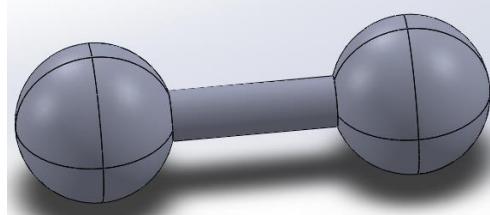


Fuente: Propia

Una vez hechos los croquis, se creó un plano transversal a estos primeros dos croquis y se hizo un nuevo croquis con la herramienta “spline”, uniendo los dos croquis para así poder utilizar la herramienta “recubrir” y darle una forma que pudiera recorrer el antebrazo sin problemas.

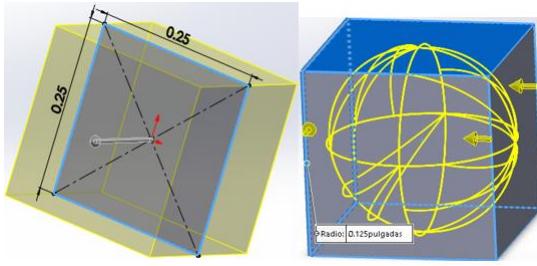
Para las uniones como se observan en la Ilustración 104, se elaboró un cubo al que simplemente se le aplicó la herramienta “redondeo”. Como se observa en la Ilustración 105.

Ilustración 104. Diseño CAD unión esférica



Fuente: Propia

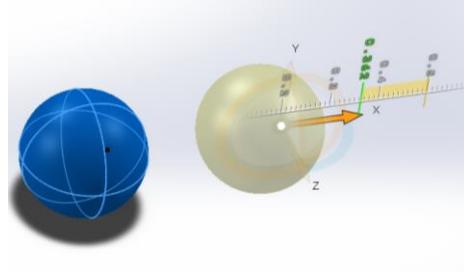
Ilustración 105. Diseño CAD redondeo cubo



Fuente: Propia

Después se utilizó la herramienta “mover/copiar sólidos” para hacer una copia de la esfera y utilizando un plano en medio de una de las esferas se extruyó un cilindro para conectarlas entre ellas.

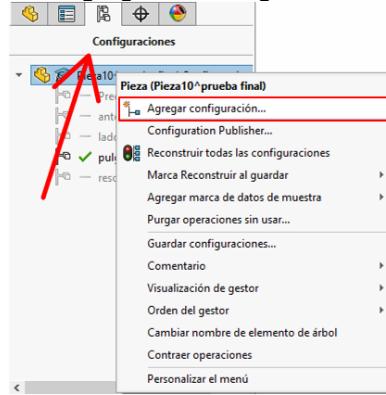
Ilustración 106. Diseño CAD copia sólido



Fuente: Propia

Para otro tipo de uniones que usaban una esfera en uno de sus extremos, se agregó una nueva configuración a la pieza, tal como se muestra en la Ilustración 107.

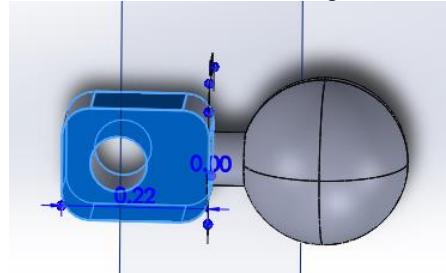
Ilustración 107. Agregando configuraciones a una pieza



Fuente: Propia

Con esta opción se pueden suprimir las operaciones que se desee quitar y reutilizar operaciones ya realizadas anteriormente. En la Ilustración 108 se muestra como se reutilizo la esfera y solo se añadió una nueva forma en lugar de la otra esfera.

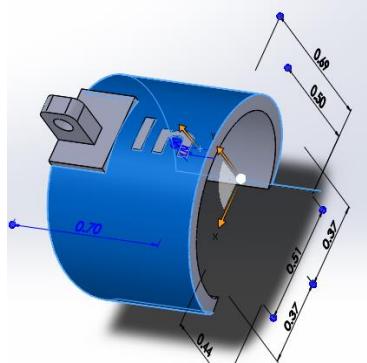
Ilustración 108. Diseño CAD configuración de pieza



Fuente: Propia

Terminando las figuras para las conexiones esféricas, se pasó al diseño de los anillos para los dedos. Estos utilizaron las mismas funciones que la elaboración de la muñeca. Solo se le añadieron una base para las conexiones con los eslabones y una escritura para identificar el anillo, tal como se observa en la Ilustración 109.

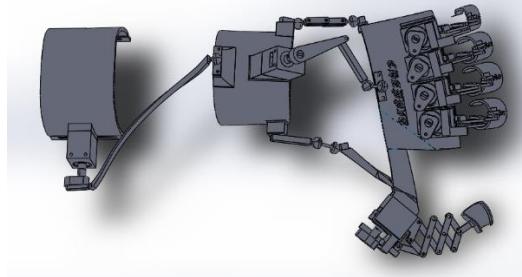
Ilustración 109. Diseño CAD anillo dedo índice



Fuente: Propia

Terminando así el primer diseño de exoesqueleto con un movimiento de 7 grados de libertad, utilizando 7 potenciómetros. Teniendo el diseño en Solid Works se imprimieron las piezas en una impresora 3D, obteniendo un total de 109 piezas.

Ilustración 110. Diseño CAD exoesqueleto



Fuente: Propia

Se utilizó el material PLA para las impresiones del exoesqueleto, ya que este tiene la ventaja de ser impreso de manera rápida y tiene una calidad promedio, aparte de tener un precio muy accesible.

Ilustración 111. Impresiones 3D de muñeca y antebrazo



Fuente: Propia

Ilustración 112. Impresiones 3D de eslabones para los dedos



Fuente: Propia

Una vez impresas las piezas, estas se limpiaron y lijaron para evitar mayor fricción entre ellas. Se utilizaron tornillos M2 5mm con cabeza de cruz y tornillos M2 8mm con cabeza allen para unir las piezas impresas. Se empezó con los eslabones de los dedos y las uniones en las bases de los potenciómetros, armando el dorso y los dedos como resultados.

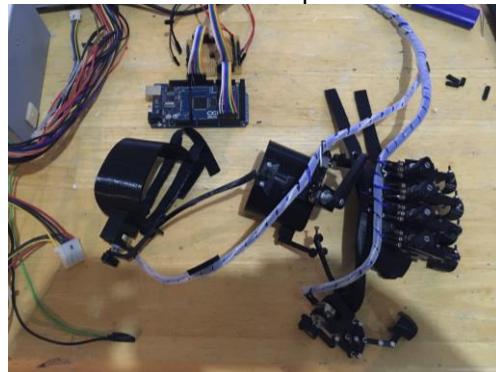
Ilustración 113. Dorso y dedos armados



Fuente: Propia

Se utilizó el mismo procedimiento para preparar las piezas en la muñeca y el antebrazo.

Ilustración 114. Exoesqueleto armado



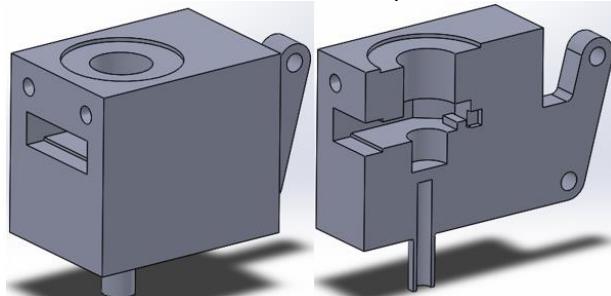
Fuente: Propia

Buscando obtener mejoras, se optó por agregar movimientos horizontales en los dedos para obtener mejor movimiento y agarre de parte de la mano robótica. Del canal de Will Cogley en YouTube, el mismo del que se obtuvo la idea para diseñar la mano robótica, se inspiró la mejora por utilizar potenciómetros

252G. Los cuales, al ser más pequeños y tener una movilidad de 360° grados, permitieron generar movimientos en el exoesqueleto para los movimientos horizontales.

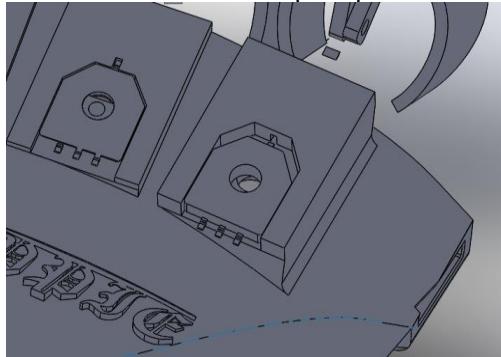
Se hizo el cambio necesario en el diseño en Solid Works para ajustar el mecanismo a los nuevos potenciómetros. Primeramente, se hicieron los cortes en la base del potenciómetro para permitir la entrada y se le añadió un poste en la parte inferior para conectar con el potenciómetro que ayudará al movimiento horizontal. Tal como se observa en la Ilustración 115 e Ilustración 116.

Ilustración 115. Diseño CAD base potenciómetro 252G



Fuente: Propia

Ilustración 116. Base horizontal para potenciómetro 252G



Fuente: Propia

Se añadió un eje para conectar con el potenciómetro y juntos con los eslabones lograr el giro para el movimiento de los dedos. A esta nueva pieza se le añadió un seguro para que el movimiento de los dedos no lo arrastrara fuera de su lugar, se hicieron agujeros para ensamblar estas piezas por medio de tornillos. Tal como se observa en la Ilustración 117, Ilustración 118 e Ilustración 119.

Ilustración 117. Eje para giro de potenciómetros

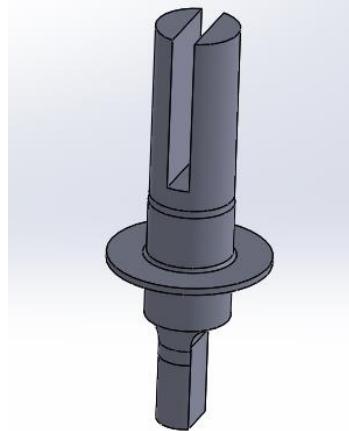


Ilustración 118. Seguro para eje de potenciómetro

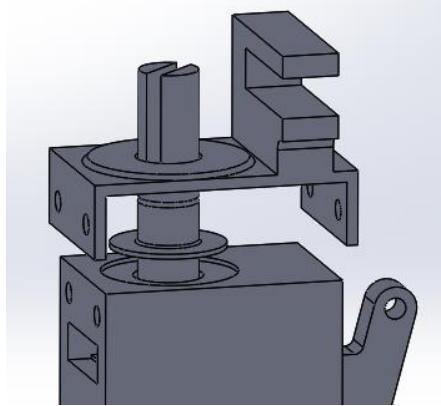
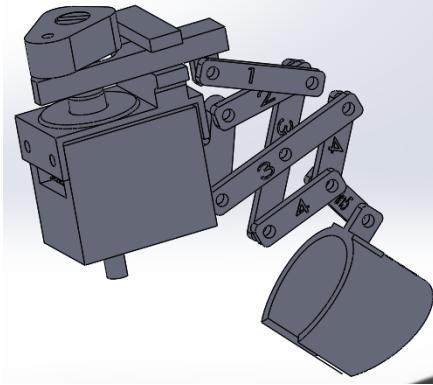


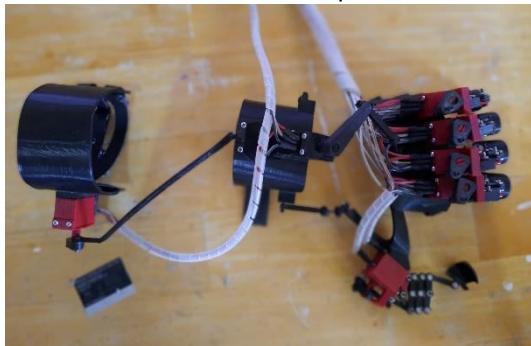
Ilustración 119. Diseño CAD nuevo potenciómetro



Fuente: Propia

Repetiendo estas mismas operaciones para todos los dedos se terminó el diseño final del exoesqueleto, con un total de 95 piezas, 104 tornillos y 12 potenciómetros (6 para el movimiento vertical y 6 para el movimiento horizontal).

Ilustración 120. Exoesqueleto final



Fuente: Propia

Elaboración código Arduino

Para la escritura de la programación en arduino se dividió en 3 partes, la programación para un filtro en los potenciómetros y su lectura; el control desde LabVIEW y la aplicación de celular; y la comunicación entre los LEDs y botones físicos con LabVIEW y la aplicación, para el encendido y apagado de ambas partes.

Filtro para potenciómetros

Esta etapa consiste en tomar varias lecturas del potenciómetro y generar un promedio de esas lecturas para eliminar ruido. Se puede consultar mejor este tipo de programación en la publicación de (Shin, 2011).

Ilustración 121. Programación filtro potenciómetros

```

Tesis_filtro.h
#ifndef Tesis_filtro_h
#define Tesis_filtro_h
#include <Servo.h>
Servo servo; //Pulgar
int P_val; //Pulgar
int Lec[10]; //Vector de lecturas
int lec[10]; //Vector de lecturas
int p = 0;
int To = 0;
int Pro = 0;

void setup() {
    // Dirección de servo
    servo.attach(2); //Pulgar
    // Posición inicial de servo
    servo.write(179); //Pulgar
    // Cantidad de valores a guardar en vector
    for(p=0; p<10; p++)
        Lec[p] = 0;
    p=0;
}

void loop() {
    // Lectura de potenciómetros y escritura en servomotores
    To = To - Lec[p];
    //Agrega una lectura a la posición actual dentro del vector
    Lec[p] = analogRead(A1);
    //Realiza la sumatoria entre lecturas
    To = To + Lec[p];
    p = p + 1;
    //Calcula el promedio y envía el resultado al servomotor
    //Dibujar
    if (p == 10) {
        p = 0;
        Pro = To / 10;
        Serial.print("#P1");
        Serial.print(",");
        Serial.print(Pro);
        Serial.print(",");
        P_val = map(Pro, 600, 710, 179, 1);
        if (P_val>144&P_val<=179) {
            Serial.print(P_val);
            servo.write(P_val);
        }
    }
}

```

Fuente: Propia

Se inició declarando las variables de tipo entero para realizar las operaciones del filtro. Donde se necesitaban 4 variables como se observan en la Ilustración 122.

Ilustración 122. Declaración de variables para filtro

```
int lec[10]; //Tamaño de array
int p = 0;
int To = 0;
int Pro = 0;
```

Fuente: Propia

Se declaró una operación FOR en el void setup para reunir 10 lecturas en el array “Lec”.

Ilustración 123. Operación de array filtro

```
void setup() {

    // Cantidad de valores a guardar en array
    for(p=0; p< 10; p++)
        Lec[p] = 0;
    p=0;
}
```

Fuente: Propia

Ya en el void loop se realizaron las operaciones para el redondeo, y estas se imprimieron en el puerto serial. Primeramente, se resta el valor actual con el valor anterior. Después se vuelve a generar el array de 10 lecturas y se suma al valor actual de “To”. Después de que suman las 10 lecturas, estas se dividen dentro de un IF para hacer el redondeo en una variable llamada “Pro”. Y finalmente se imprimen en el puerto serial. Tal como se observa en la Ilustración 124.

Ilustración 124. Redondeo de array filtro

```
void loop() {
    To = To - Lec[p];
    //Agrega una lectura a la posición actual dentro del array
    Lec[p] = analogRead(A11);
    //Realiza la sumatoria entre lecturas
    To = To + Lec[p];
    p = p + 1;
    if (p >= 10){
        p = 0;
        Pro = To / 10;
        Serial.print(Pro);
    }
}
```

Fuente: Propia

Esta misma programación se utilizó en los 12 potenciómetros del exoesqueleto, tan solo se adaptó para distribuir la impresión en el puerto serial separadamente y que el HMI pudiera recibir estos datos de una manera ordenada.

El resultado de la variable promedio se utilizó para realizar la operación de mapeo y enviar el valor al servomotor. Pero, primeramente, se obtuvieron los valores del movimiento mecánico en los potenciómetros del exoesqueleto. Pues este movimiento no abarcaba todos los valores del potenciómetro, se movía en un rango menor, una vez que tenían energía los potenciómetros, se podían tomar las lecturas del rango de estos en el movimiento mecánico. Ya obtenido este rango se utilizaron en el rango del mapeo. Un ejemplo se observa en la Ilustración 125.

Ilustración 125. Mapeo para servomotor

```
Serial.print(Pro);
Serial.print(",");
P_val = map(Pro, 600, 710, 179, 1);
if(P_val>=1&&P_val<=179){
    Serial.print(P_val);
    servo.write(P_val);
}
else{
    Serial.print("0");
}
```

Fuente: Propia

Control desde LabVIEW y aplicación de celular

En esta sección se declararon primeramente las variables que se utilizaban en el modo control.

Ilustración 126. Variable tipo carácter

```
//Variable para lectura de puerto serial
char b;
String readString;
```

Fuente: Propia

Ya en el while loop, se utilizó un IF para revisar si el puerto serial estaba recibiendo caracteres. Si la condición era TRUE, se ejecutaba un comando WHILE, el cual se encargaba de repetir la programación si el puerto serial estaba en uso. Después las lecturas que se estaban recibiendo en el puerto serial se

acumulaban en una variable de tipo carácter “c”. después se ejecutaba un comando IF, en el que si las lecturas eran mayores a “0” se escribían en el servomotor. Tal como se observa en la Ilustración 127.

Ilustración 127. Programación de control

```
//Pulgar
else if(b=='A'){
    delay(6);
    while (Serial.available()) {
        char c = Serial.read();
        readString += c;
    }
    if(readString.length() >0){
        servo2.write(readString.toInt());
        readString="";
    }
}
```

Fuente: Propia

Esta programación se utilizó para todos los servomotores que se trabajan en la mano robótica.

Comunicación entre botones físicos y virtuales

Primeramente, se declararon las variables de los indicadores y botones, tanto sus entradas como las variables para los valores virtuales.

Ilustración 128. Declaración de variables para botones

```
//Declarando pulsadores y LEDs de comunicación
const int btn_verde=26;
const int btn_rojo=27;
const int ledverde=22;
const int ledama=24;
const int ledrojo=23;
const int ledazul=25;

//Declarando variables de control
int btn_vread;//lectura de botón verde
int btn_rread;//lectura de botón rojo
bool lab_verde;
bool lab_rojo;
bool lab_azul;
bool lab_ama;
bool ant_val;
bool val_0;
bool val_25;
```

Fuente: Propia

En la sección del set up se declararon las salidas, las entradas y valores iniciales de variables selectas.

Ilustración 129. Entradas, salidas y valores iniciales

```

// Declarando entradas y salidas de botones e indicadores
pinMode (ledverde,OUTPUT);
pinMode (ledrojo,OUTPUT);
pinMode (ledazul,OUTPUT);
pinMode (ledama,OUTPUT);
pinMode (btn_verde,INPUT);
pinMode (btn_rojo,INPUT);
// Indicadores apagados
digitalWrite (ledverde,HIGH);
digitalWrite (ledrojo,HIGH);
digitalWrite (ledazul,HIGH);
digitalWrite (ledama,HIGH);
// Variables para uso de botones
lab_verde = false;
lab_rojo = true;
lab_azul = false;
lab_ama = false;
ant_val = true;
val_25 = true;
val_0 = false;

```

Fuente: Propia

Una vez en el while loop, el código empieza por defecto en apagado, solo está esperando a ser llamado para ser encendido. Esto se efectuó por medio de un WHILE para lograr que la lectura permaneciera en esta sección de código sin estar leyendo otras áreas del código innecesarias en el momento. Mientras permaneciera en apagado, los indicadores permanecían apagados excepto por el indicador rojo, que indica físicamente al usuario que el sistema está apagado. Tal como se observa en la Ilustración 130

Ilustración 130. Programación arduino apagado

```

void loop() {
    // Programa apagado
    while(lab_rojo==true&&lab_verde==false){
        digitalWrite (ledverde,HIGH);
        digitalWrite (ledrojo,LOW);
        digitalWrite (ledazul,HIGH);
        digitalWrite (ledama,HIGH);
        if(Serial.available()){
            b = Serial.read();
            if(b=='a'){
                delay(6);
                lab_verde=true;//Encendido Labview
                lab_rojo=false;
            }
        }
        btn_vread = digitalRead(btn_verde);
        btn_rread = digitalRead(btn_rojo);
        if(btn_vread==HIGH&&btn_rread==LOW){
            lab_verde=true;//Encendido Labview
            lab_rojo=false;
            Serial.println("encender");
        }
    }
}

```

Fuente: Propia

En caso de que se enviara un carácter desde LabVIEW o la aplicación de celular, se efectuaba el comando IF, en el cual si el carácter “a” es recibido se guardaba en una variable de tipo booleana llamada “lab_verde” la cual detendría la condición WHILE de apagado. En el caso de que se presionara el botón físico,

de igual forma se cambiaban los valores booleanos de las variables “lab_verde” y “lab_rojo” que detienen la condición de WHILE apagado. Pasando así a la siguiente parte de la programación.

Cuando el programa pasaba a encendido, entraba en otro WHILE en el que se efectuaba el encendido del indicador verde y el apagado del indicador rojo. Después pasaba a otro WHILE exclusivo para el modo lectura, en el que por defecto iniciaba con el indicador azul encendido y el modo lectura exoesqueleto. Mientras la condición de este WHILE fuera TRUE estaría esperando comandos por medio del puerto serial que efectuarían diferentes funciones. En el caso de que se recibiera el carácter “d” cambiarían los valores de las variables para avanzar a modo control y romper las condiciones WHILE de lectura. Si se recibía el carácter “b” se cambiaba a modo apagado. Si se recibía el carácter “e” se efectuaba el modo paro de emergencia. Si no se recibía ninguno de estos caracteres, el WHILE de lectura seguía avanzando en el código y efectuaba el comando “Exoesqueleto”, en donde estaba escrito el código del filtro y la escritura del redondeo en los servomotores. Tal como se observa en la Ilustración 131 e Ilustración 132.

Ilustración 131. Encendido de sistema Arduino

```
// Programa encendido
while(lab_verde==true&&lab_rojo==false&&btn_rread==LOW) {
    digitalWrite(ledverde,LOW);
    digitalWrite(ledrojo,HIGH);
    while(lab_ama==false&&(lab_azul==true||ant_val==true)&&lab_rojo==false){
        digitalWrite(Ledazul,LOW);
        digitalWrite(Ledama,HIGH);
        btn_vread = digitalRead(btn_verde);
        btn_rread = digitalRead(btn_rojo);
        if(Serial.available()){
            b = Serial.read();
            if(b=='d'){
                lab_ama=true;//Modo control
                lab_azul=false;
                ant_val=false;
            }
            else if(b=='b'){
                lab_rojo=true;//apagar
                lab_verde=false;
            }
            else if(b=='e'){
                digitalWrite(ledverde,HIGH);
            }
        }
    }
}
```

Fuente: Propia

Ilustración 132. Encendido de sistema Arduino 2

```

else if(b=='e'){
    digitalWrite(ledverde,HIGH);
    digitalWrite(ledrojo,HIGH);
    digitalWrite(ledazul,HIGH);
    digitalWrite(ledama,HIGH);
    lab_rojo=false;
    lab_verde=false;
    lab_ama=false;
    lab_azul=false;
    ant_val=false;
}
else{
    Exoesqueleto();
}
if(btn_rread==HIGH&btn_vread==LOW){
    lab_verde=false;
    lab_rojo=true;
}
}

```

Fuente: Propia

Después de efectuar el comando “Exoesqueleto” se pasaba a un IF, que permitía apagar el sistema con el botón físico si es que se deseaba.

Cuando el carácter “d” se recibía en el puerto serial se cambiaba a modo control, el cual rompía la condición del WHILE de lectura y se pasaba a la siguiente parte del código. En la que se entraría en otro WHILE exclusivo para el control.

En este WHILE, se efectuaban los mismos comandos que en el WHILE de lectura, solo que con las variables adaptadas para el control. Primeramente, se encendía el indicador amarillo, y el indicador azul se apagaba. Después lía las entradas de los botones físicos y pasaba a un IF. En este IF se detectaba la entrada de caracteres por el puerto serial al igual que en el WHILE de lectura, con la única diferencia de que el carácter “d” ya no realizaba ninguna operación y se cambiaba por el carácter “c”, que, si se llegaba a recibir en el puerto serial, activaría el modo lectura y terminaría con la condición WHILE de control.

Después, lo único que está haciendo este WHILE de control es estar esperando caracteres por el puerto serial, que serían los caracteres que indicaban que dedo de la mano robótica se estaba controlando. Tal como se observa en la Ilustración 133 y la Ilustración 134.

Ilustración 133. Modo lectura Arduino

```

while(lab_ama==true&lab_azul==false&lab_rojo==false) {
    digitalWrite(ledazul,HIGH);
    digitalWrite(ledama,LOW);
    btn_vread = digitalRead(btn_verde);
    btn_rread = digitalRead(btn_rojo);
    if(Serial.available()){
        b = Serial.read();
        if(b=='c'){
            delay(6);
            lab_ama=false;//Lectura exoesqueleto Labview
            lab_azul=true;
        }
        else if(b=='b'){
            delay(6);
            lab_rojo=true;//Apagar labview
            lab_verde=false;
        }
        else if(b=='e'){
            delay(6);
            digitalWrite(ledverde,HIGH);
            digitalWrite(ledrojo,HIGH);
            digitalWrite(ledazul,HIGH);
            digitalWrite(ledama,HIGH);
            lab_rojo=false;
            lab_verde=false;
            lab_ama=false;
        }
    }
}

```

Fuente: Propia

Ilustración 134. Modo lectura Arduino 2

```

else if(b=='e'){
    delay(6);
    digitalWrite(ledverde,HIGH);
    digitalWrite(ledrojo,HIGH);
    digitalWrite(ledazul,HIGH);
    digitalWrite(ledama,HIGH);
    lab_rojo=false;
    lab_verde=false;
    lab_ama=false;
    lab_azul=false;
    ant_val=false;
}

//Pulgar
else if(b=='A'){
    delay(6);
    while (Serial.available()){
        char c = Serial.read();
        readString += c;
    }
    if(readString.length() >0){
        servo2.write(readString.toInt());
        readString="";
        //Limpiar string
    }
}

```

Fuente: Propia

Elaboración de aplicación de celular

Para la elaboración de la aplicación se utilizó App Inventor, una plataforma que proporciona herramientas básicas y de manejo fácil para la elaboración sencilla de una aplicación exclusivamente para dispositivos Android. Aunque tiene la ventaja de desarrollarse desde una Mac, Linux o Windows, solamente se puede utilizar la aplicación desde un celular o tableta Android.

Para esta aplicación se elaboró una interfaz con 5 pestañas capaz de obtener control sobre los dedos de la mano robótica y la muñeca, también se le añadió control por medio de comandos de voz y, por último, lectura por medio de indicadores y gráficas.

Inicialmente se observa la pestaña principal, en la que se encuentra un menú. De este menú se tiene como primera opción la pestaña de “Control”, en segunda opción se tiene “Gráfica”, y en la tercera opción se tiene “Comando de voz”. Tal como se observa en la Ilustración 135.

En la pestaña principal se tienen las opciones diferentes para control o lectura, de esta pestaña el usuario puede desplazarse a la pestaña deseada.

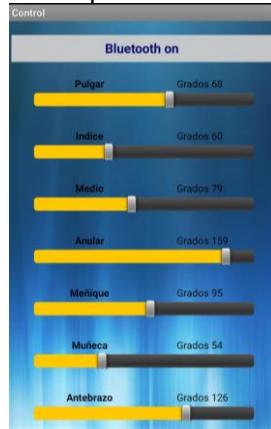
Ilustración 135. Aplicación de celular menú



Fuente: Propia

Si se selecciona la opción de “Control”, la aplicación se dirige a una pestaña en la que el usuario puede controlar los dedos y la muñeca para el movimiento vertical por medio de sliders.

Ilustración 136. Aplicación de celular control

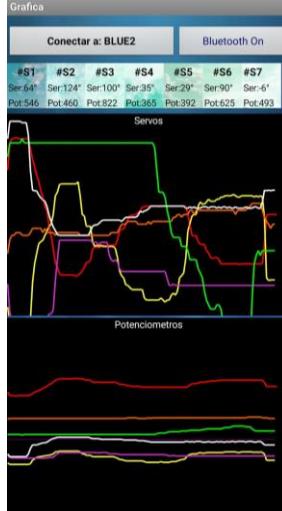


Fuente: Propia

En la opción de “Gráfica”, el usuario puede observar el comportamiento de los movimientos verticales de la mano robótica por medio de indicadores y

gráficas. Una vez conectado al módulo de bluetooth, se reciben los valores por el puerto serial de parte de arduino, los cuales se distribuyen en los indicadores para cada sensor y se concatenan en las gráficas de los servos y los potenciómetros. Tal como se observa en la Ilustración 137.

Ilustración 137. Aplicación de celular graficas

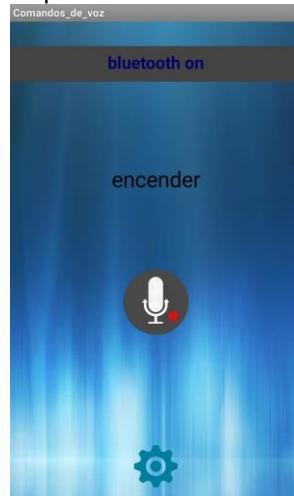


Fuente: Propia

En la última opción del menú se tiene “Comando de voz”. En esta pestaña se puede observar la herramienta de micrófono para efectuar comandos de voz. Estos comandos se pueden modificar por parte del usuario desde una pestaña exclusiva para configurar los valores a recibir y enviar. Una vez asignados los caracteres que se enviaran por medio del puerto serial, se guarda la configuración y se regresa a la pestaña de comandos. Tal como se observa en la Ilustración 138 e Ilustración 139.

Estos caracteres se reciben en el puerto serial de Arduino, cada uno a través de condiciones efectúan un movimiento en la mano robótica.

Ilustración 138. Aplicación de celular comando de voz



Fuente: Propia

Ilustración 139. Aplicación de celular configuración

Configuracion	
Voz a reconocer	Datos a enviar
Apagar	b
encender	a
Bluetooth	c
control	d
V5	D5

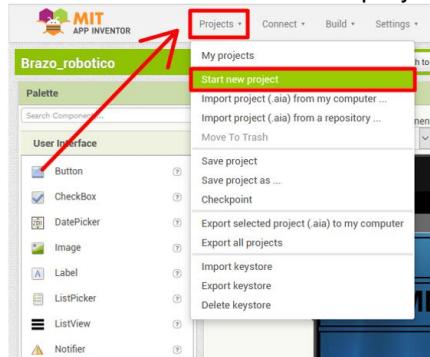
Fuente: Propia

La programación que se utiliza en la plataforma de App Inventor se divide en dos secciones como lo es LabVIEW, primeramente, se tiene el “App Inventor Designer” y en segundo “App Inventor Block Editor”.

En “App Inventor Designer” se creó la interfaz de usuario, en la que se seleccionaron los elementos y componentes con los que interactuó el usuario. Mientras que en el “App Inventor Block Editor”, se hizo la programación por bloques para hacer conexiones y dar vida a los elementos seleccionados en el “App Inventor Designer”.

Una vez ubicado en la página de App Inventor, se dirigió a la pestaña de “projects” » “start new project”, esta acción creó una nueva ventana en la que se observaron áreas con los títulos: “Palette”, “Viewer”, “Components”, “Media” y “Properties”. Tal como se observa en la Ilustración 140 e Ilustración 141.

Ilustración 140. Creación nuevo proyecto



Fuente: Propia

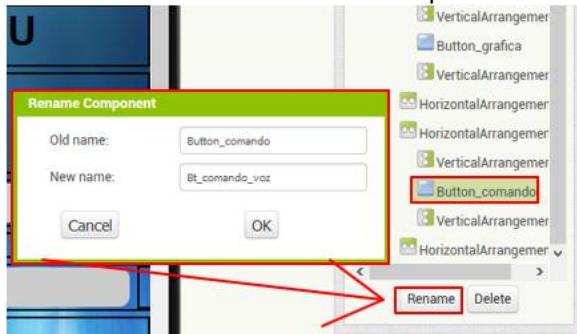
Ilustración 141. Herramientas para elaborar nuevo proyecto



Fuente: Propia

En el área de “Palette” se tienen los elementos que se utilizaron en la aplicación. De aquí se seleccionaron los elementos que se arrastran al área de “Viewer”, donde se ven virtualmente los elementos seleccionados como aparecen físicamente cuando la aplicación fue construida. En el área de “Components” aparecen los elementos arrastrados al área de “Viewer”, aparecen en el orden en que se agregaron y se pudieron modificar los nombres para obtener una mejor idea de lo que se estaba estructurando.

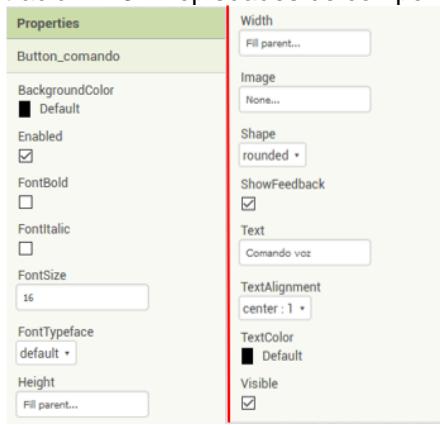
Ilustración 142. Renombrar componentes



Fuente: Propia

Para configurar estos componentes se dirigió al área de “Properties”, en esta sección el usuario tiene una variedad de opciones para configurar y darle la forma al componente seleccionado.

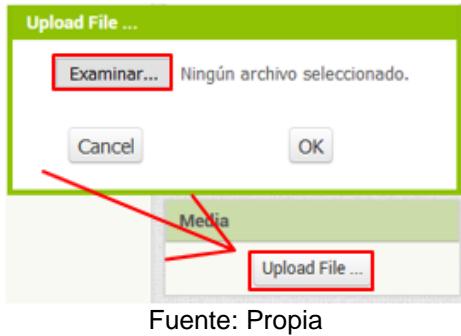
Ilustración 143. Propiedades de componente



Fuente: Propia

En el caso de las imágenes y audios, ya sea para adornar los fondos de pantalla o adornar los componentes seleccionados por el usuario, se buscó en el área de “Media”. Se dió click en “Upload file” » “Examinar” y se seleccionó el archivo que se deseó utilizar en la aplicación.

Ilustración 144. Subir archivo



Para la siguiente sección se dividirá la explicación de las pestañas de la aplicación, con el propósito de profundizar y analizar los elementos que la componen.

Empezando con la pestaña principal, se utilizaron los componentes “HorizontalArrangement” para adornar la pantalla con los elementos a utilizarse en ella. Una vez colocados los “HorizontalArrangement” y “VerticalArrangement”, se insertaron los botones que se utilizarían para cambiar de pestaña.

Ilustración 145. Inserción de "layout"



Una vez hecho esto, se modificaron las propiedades de los layouts para darle las dimensiones necesarias para acomodarse dentro de los rangos de la pantalla. En las opciones de propiedades se modificaron las herramientas “Height” y “Width”. La herramienta “Height” modificó la altura del layout y la herramienta “Width” modificó el ancho del layout.

Ilustración 146. Layout height

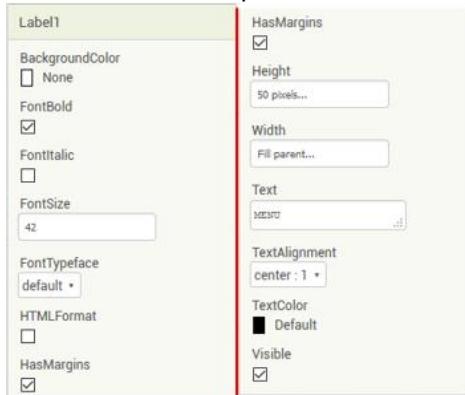


Fuente: Propia

Dando click en cualquiera de estas herramientas, aparecen 4 opciones para la configuración de este layout: “Automatic”, “Fill parent”, “pixels” y “percent”. La opción de “Automatic” coloca la dimensión del layout por defecto, la opción “Fill parent” abarca las dimensiones hasta donde otro layout o borde de la pantalla lo detengan. La opción de “pixels” le asigna una dimensión específica de pixeles en la pantalla. Y, por último, la opción de “percent” asigna un porcentaje de la dimensión de la pantalla al layout.

Pasando a configurar las propiedades del “label”, se le dió click a la casilla de “Fontbold” para marcar como negritas el texto. Se le asignó un tamaño de “42” en “FontSize” para abarcar un tamaño de letra considerable en la pantalla de la aplicación. Se le dió click a la casilla de “HasMargins” para crear líneas alrededor del label. En la opción de “Height” se le asignó un valor de “50 pixels”, en la opción de “Width” se le asignó “Fill parent”. En la opción de “Text” se le asignó el nombre de “Menú”. En la opción de “TextAlignment” se escogió “center:1” para que el texto se acomodara en medio del layout.

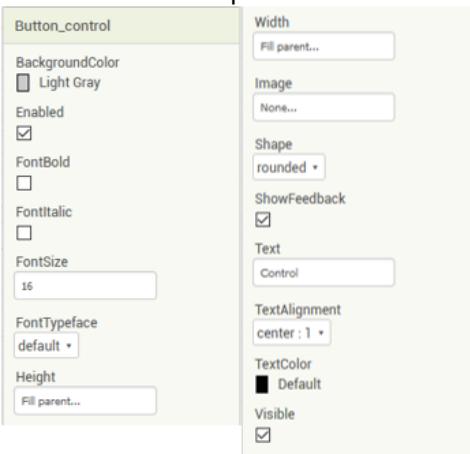
Ilustración 147. Propiedades de “label”



Fuente: Propia

Siguiendo con los botones, se escogió el color “Light Gray” para el color de fondo del botón. Se marcó la casilla de “Enable” para habilitar el botón desde que inicia la aplicación. En “FontSize” se utilizó el valor de “16” para el tamaño de la letra. En la opción de “Height” se escogió “Fill parent” y así mismo en la opción de “Width”. En la opción de “Shape” se seleccionó “rounded” para darle bordes redondos al botón. En la opción de “Text” se insertó el texto “Control”. En la opción de “TextAlignment” se seleccionó “center:1”. En la opción de “TextColor” se seleccionó el color negro para el color del texto.

Ilustración 148. Propiedades de botones

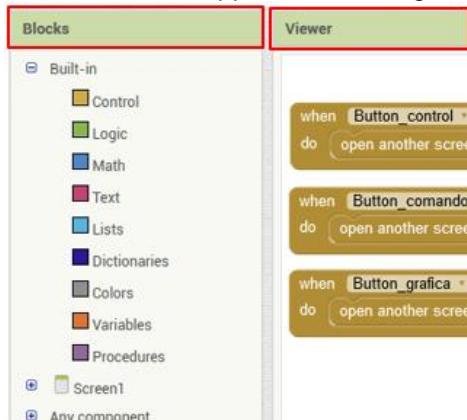


Fuente: Propia

Pasando a la programación en bloques, se tiene en el lado izquierdo de la pantalla la sección de “Blocks”, y del lado derecho se puede observar la sección de “Viewer”. En el área de “Blocks” se tienen los componentes que se utilizan para

la programación, estos componentes se dividen en diferentes tipos, incluso las operaciones de los componentes que se agregaron al “App Inventor Designer” aparecerán en esta área. En el área de “Viewer” se observan los bloques seleccionados del área de “Blocks” y se pueden unir entre ellos para lograr el funcionamiento requerido por parte del usuario.

Ilustración 149. App Inventor Designer



Fuente: Propia

En el caso de la programación para la pestaña de menú, se seleccionó el primer botón en el área de “Blocks”. Una vez hecho esto, apareció una lista con diferentes operaciones de esta función. Se seleccionó la primera opción “when Button_control Click” que sirve para efectuar una función cuando se le de click al botón seleccionado.

Ilustración 150. Función para botón control



Fuente: Propia

Dentro de este bloque se agregó la función “open another screen screenName”, que sirve para abrir otra pestaña de la aplicación.

Ilustración 151. Función para abrir pestaña



Fuente: Propia

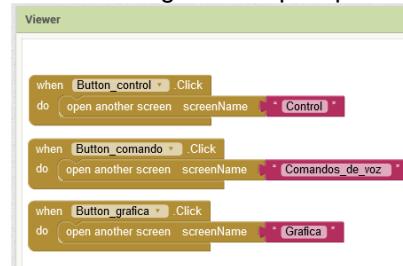
A este se le conectó un bloque de texto string con el nombre de la pestaña a abrir. Tal como se observa en la Ilustración 152 e Ilustración 153.

Ilustración 152. Dato string



Fuente: Propia

Ilustración 153. Código de bloques pestaña menú

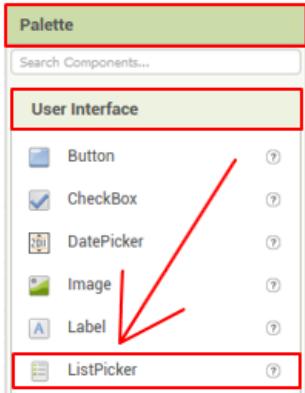


Fuente: Propia

Pasando al diseño de la pestaña de “Control”, se repitieron los pasos de inserción de layouts para adornar la pestaña en la sección de “App Inventor Designer”. Una vez insertados y configuradas las dimensiones se insertaron sliders, labels y un “ListPicker”.

El “ListPicker” se seleccionó del área de “Palette” » “User Interface” » “ListPicker”. Tal como se observa en la Ilustración 154.

Ilustración 154. Selección de "ListPicker"

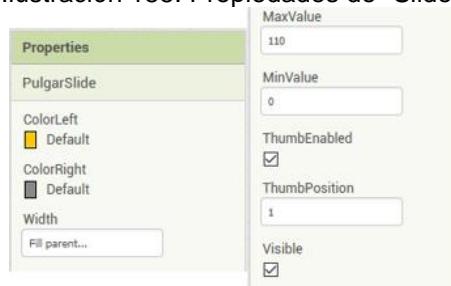


Fuente: Propia

Esta herramienta tiene la forma parecida de un botón, pero su función es la de llamar una lista de textos en la que el usuario puede escoger diferentes dispositivos disponibles en el celular. Y se le asignó la misma configuración que los botones en la pestaña de “Menú”.

En la configuración de los sliders, se les asignó el color amarillo en la opción “ColorLeft” y el color gris en la opción “ColorRight” para diferenciar cuando este se deslice por el usuario. En la opción de “Width” se seleccionó la opción “Fill parent...” para abarcar todo el espacio del layout en el que se encuentra adentro. En la opción de “.MaxValue” se insertó el valor “110” y en la opción “.MinValue” se insertó el valor “0”, estos son los valores máximo y mínimo del slider. En la opción de “ThumbEnabled” se insertó el valor de “1”, para que al iniciar la aplicación el slider tenga esa posición como valor.

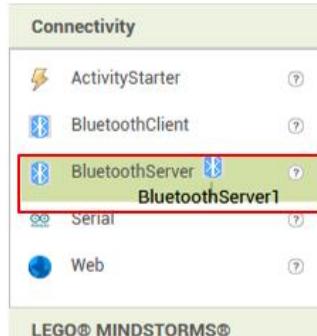
Ilustración 155. Propiedades de "Slider"



Fuente: Propia

Después de esto se agregó la herramienta de “BluetoothClient” que se encuentra en el área de “Palette” » “Connectivity” » “BluetoothClient”. Se arrastró hasta el área de “Viewer” y se colocó por defecto en los componentes no visibles.

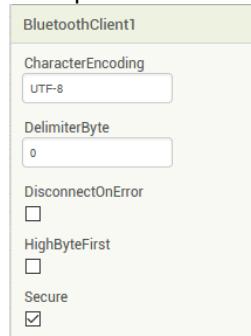
Ilustración 156. Ubicación de "BluetoothClient"



Fuente: Propia

En las propiedades de “BluetoothClient” se dejaron los valores por defecto.

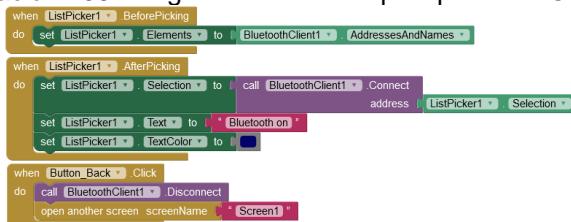
Ilustración 157. Propiedades de "BluetoothClient"



Fuente: Propia

Una vez terminado el diseño de la pestaña de “Control” se prosiguió a realizar la programación por bloques. En la que se inició configurando el “ListPicker” para llamar una lista de dispositivos bluetooth vinculados con el celular.

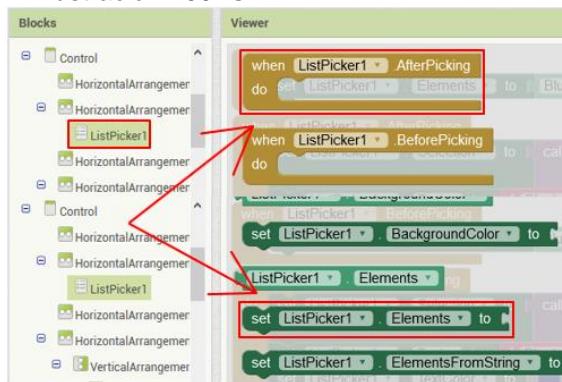
Ilustración 158. Programación en bloques pestaña “Control”



Fuente: Propia

En el área de “Blocks”, se dio click sobre “ListPicker” en el que apareció un menú de comandos de este componente. Inicialmente se escogió el comando “when ListPicker AfterPicking” que sirve para crear una lista de componentes a seleccionar por el usuario cuando se le da click al “ListPicker”. Después se arrastró el comando “set ListPicker Elements to” para conectarlo con “when ListPicker AfterPicking”, esta herramienta sirvió para seleccionar los elementos de “ListPicker” y aplicarlos a una siguiente instrucción.

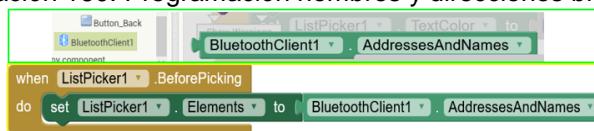
Ilustración 159. Comandos de “ListPicker”



Fuente: Propia

Después se utilizó el comando de “BluetoothClient AddressesAndNames” para conectarlo con los elementos seleccionados anteriormente. Este comando se encuentra en el área de “Blocks” » “BluetoothClient” » “BluetoothClient AddressesAndNames”.

Ilustración 160. Programación nombres y direcciones bluetooth



Fuente: Propia

Esta combinación de elementos creó una lista de dispositivos bluetooth que estaban vinculados con el celular. Una vez se seleccionó el dispositivo, se utilizó la siguiente programación:

Ilustración 161. Programación "AfterPicking"



Fuente: Propia

La programación que se muestra en la Ilustración 161, aparte de seleccionar el dispositivo bluetooth con el que se comunicó la aplicación, también realizó un cambio en el texto. Este cambió el texto de “Bluetooth” a “Bluetooth on” y el color negro a color azul.

El comando de “when ListPicker afterpicking” se encontró en el área de “Blocks” » “ListPicker” » “when ListPicker AfterPicking”. En seguida se le conectó en la sección de “do” el bloque de “set ListPicker Selection to”, que se localizó en los comandos de “ListPicker”. Después se le conectó un comando del componente de “BluetoothClient” que se llama “call BluetoothClient Connectaddress”, y en seguida se conectó “ListPicker Selection”.

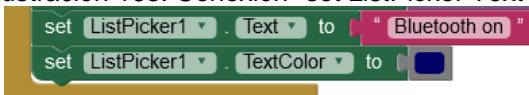
Ilustración 162. Conexión "call BluetoothClient Connectaddress"



Fuente: Propia

En seguida de estos elementos, se añadió el comando “set ListPicker Text to” y después se conectó un texto string con el texto “Bluetooth on”. Debajo de esta conexión se añadió un comando “set ListPicker TextColor to” y se le conectó un bloque de color azul. Todo esto para realizar un cambio que el usuario pueda reconocer, que el celular se ha conectado exitosamente por medio de bluetooth con el dispositivo externo.

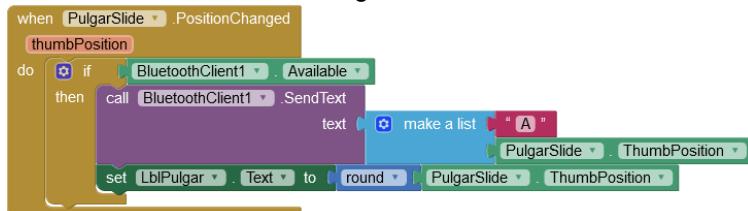
Ilustración 163. Conexión "set ListPicker Text to"



Fuente: Propia

Para el control por medio de los sliders se utilizó la siguiente programación de bloques:

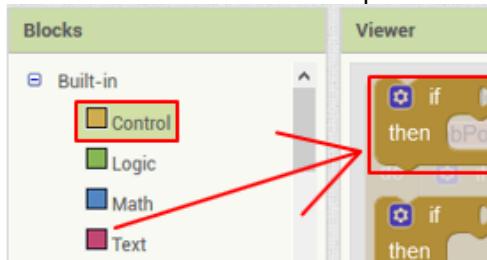
Ilustración 164. Programación control slider



Fuente: Propia

Para esta programación se utilizó un comando “when Slide PositionChanged” que se encuentra seleccionando en el área de “Blocks” » “(slider a programar)” » “when Slide PositionChanged”. A continuación se insertó en la configuración “do” un bloque “if then” que se ubica en el área de “Blocks” » “Built-in” » “Control” » “if then”. Tal como se observa en la Ilustración 165.

Ilustración 165. Ubicación bloque "if then"



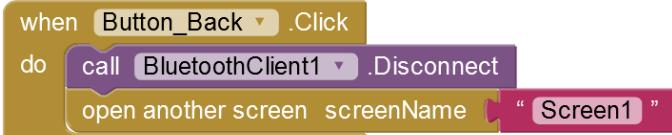
Fuente: Propia

En la conexión de “if” se insertó un bloque del bluetooth “BluetoothAvailable”. En la conexión del “then” se insertó un “call Bluetooth SendText” conectado a un bloque de “make a list” con dos entradas, a las que se conectaron un bloque string y un bloque del slider a manipular “Slide ThumbPosition”. Debajo del bloque “call BluetoothClient SendText” se insertó un bloque “set LbL Text to”, que se encuentra en los bloques del label que muestra al usuario los valores del slider a controlar. En seguida se conectó un bloque “round” y a este se le conectó un bloque “Slide ThumbPosition”. Esta programación ayudó a detectar si el puerto serial estaba libre, entonces se enviaban los datos por medio de una lista, que la conformaban el comando “A” y los valores del slider.

Esta misma programación se utilizó con todos los sliders y sus respectivos labels, solo se cambiaron los comandos a enviar por el puerto serial.

Por último, se tiene la programación del botón para salir de la pestaña y volver a la pestaña de menú. Tal como se observa en la Ilustración 166.

Ilustración 166. Programación botón de vuelta

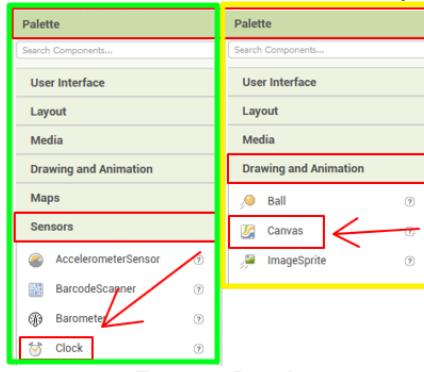


Fuente: Propia

Se utilizó un “when Button Click”, y en la conexión de “do” se insertó un bloque “call BluetoothClient Disconnect”, enseguida se insertó un bloque “open another screen screenName” conectado a un bloque string con el nombre de la pestaña de “Menú”.

Pasando a la programación de bloques de la pestaña de “Gráficas”, se inició editando la pestaña en “App Inventor Designer”, utilizando las mismas configuraciones que en las pestañas diseñadas anteriormente. Añadiendo algunos componentes extras como lo son: “Canvas” y “Clock”, que se ubican en el área de “Palette” » “Drawing and Animation” » “Canvas” y “Palette” » “Sensors” » “Clock”.

Ilustración 167. Ubicación se "Canvas" y "Clock"



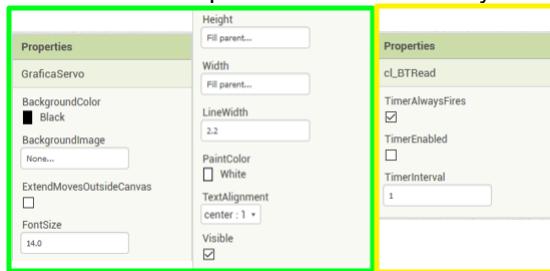
Fuente: Propia

La herramienta de “Canvas” ayudó a dibujar líneas en un área determinada de la pestaña para simular una gráfica en tiempo real. Y la herramienta “Clock” ayudó a crear lapsos de tiempo para correr los ciclos de la programación en un determinado tiempo. En las propiedades de “Canvas” se seleccionó el color negro para la opción de “BackgroundColor”, en la opción de “FontSize” se insertó el valor

“14.0”, en las opciones “Height” y “Width” se utilizó “Fill parent” para expandir las dimensiones hasta donde el layout lo permitiera, en la opción de “LineWidth” se utilizó el valor de “2.2”, y en la opción de “PaintColor” se escogió el color blanco. Tal como se observa en la Ilustración 168.

En las propiedades de “Clock” se marcó la casilla de “TimerAlwaysFires”, esto para que el temporizador se activara en todo momento. La casilla de “TimerEnabled” se marcó de igual manera para habilitar el uso del temporizador desde que se inicie la pestaña. Tal como se observa en la Ilustración 168.

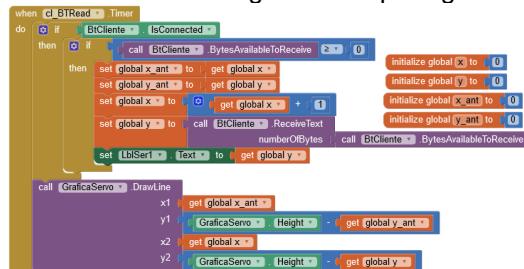
Ilustración 168. Propiedades de "Canvas" y "Clock"



Fuente: Propia

Una vez añadidos los componentes y configurado sus propiedades se pasó a la programación en bloques para esta pestaña.

Ilustración 169. Programación para gráfica



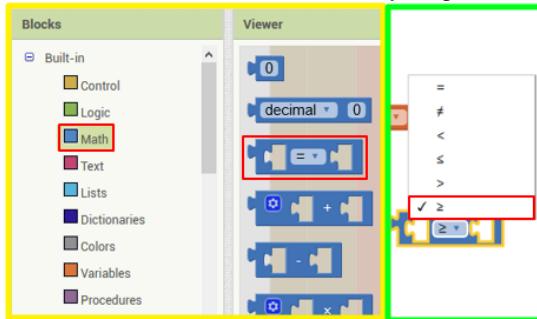
Fuente: Propia

Iniciando, se programó la conexión por bluetooth como se hizo anteriormente en la pestaña de “Control”, después se realizó la programación para la gráfica y lectura de datos a través de los labels.

Se seleccionó inicialmente un bloque perteneciente al componente “Clock” que se llama “when Clock Timer”. A este se le conectó un bloque “if then”,

después se le conectó al “if” un bloque de “BluetoothClient IsConnected”, en la conexión de “then” se conectó otro “if then”. A la conexión de este nuevo “if then” se introdujo un bloque mayor igual a que se ubica en “Blocks” » “Math”, y se seleccionó el mayor igual a, tal como se observa en la Ilustración 170.

Ilustración 170. Ubicación mayor igual a



Fuente: Propia

Dentro de este, se insertó un bloque del bluetooth “call BluetoothClient BytesAvailableToReceive” y un bloque numérico con el valor “0”. En la conexión “then”, se insertaron los valores de las variables globales para el eje “x” y eje “y” de la gráfica a dibujar.

Para este paso se crearon variables ubicadas en “Blocks” » “Variables” » “variable global name to”. Una vez arrastrada al área de ‘Viewer’ se insertó el nombre de “x” para la primera variable, el nombre de “y” para la segunda variable, el nombre de “x_ant” para la tercera variable y a la última variable se le asignó el nombre de “y_ant”. Y a todas estas variables se les conectó un valor numérico “0”. Puesto que se esperaba que al iniciar la pestaña iniciaran en cero. Tal como se observa en la Ilustración 171.

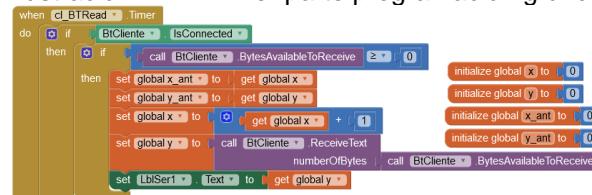
Ilustración 171. Ubicación de variables



Fuente: Propia

Una vez creadas estas variables se pasó a insertar funciones de estas a la programación principal. Se utilizó un bloque “set global x_ant to” y a este se conectó un “get global x”. Después se insertó un bloque “global y_ant to” y a este se le conectó un “get global y”. Esta programación se debe a que el componente “Canvas” dibuja por medio de un valor anterior y un valor actual para crear una línea, es decir, un punto inicial y un punto final por cada ciclo de tiempo. Después de estos bloques se insertó un “set global x” y a este se le conectó un bloque de suma. En su interior se insertaron un bloque “get global x” y un bloque numérico con el valor “1”. Después se insertó un bloque “set global y to” y se le conectó un bloque de bluetooth “call BluetoothClient” ReceiveTextnumberOfBytes” y un bloque “call BluetoothClient BytesAvailableToReceive”. Tal como se observa en la Ilustración 172. Esto con la intención de avanzar en la gráfica en el eje “x” siempre un espacio, e insertar los valores provenientes del puerto serial en el eje “y”. Finalmente, se insertó un bloque perteneciente al label en donde se insertarían los valores del puerto serial para que el usuario pudiera observarlos. Se insertó un bloque “set label text to” y se le conectó un bloque “get global y”.

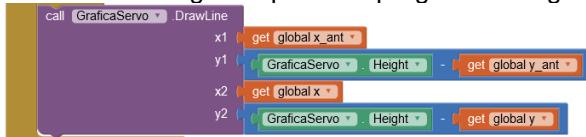
Ilustración 172. Primer parte programación gráfica



Fuente: Propia

Una vez terminada esta parte de la programación, se insertó un bloque perteneciente a “Canvas” llamado “call Canvas DrawLine”. En su entrada “x1” se le inserto un bloque “get global x_ant”, en la entrada “y1” se insertó un bloque de resta con los bloques “Canvas Height” y “get global y_ant”. En la entrada “x2” se insertó un bloque “get global x”. Y, por último, en la entrada “y2” se insertó un bloque numérico de resta, dentro de él se insertaron los bloques “Canvas Height” y “get global y”. Tal como se observa en la Ilustración 173.

Ilustración 173. Segunda parte de programación gráfica



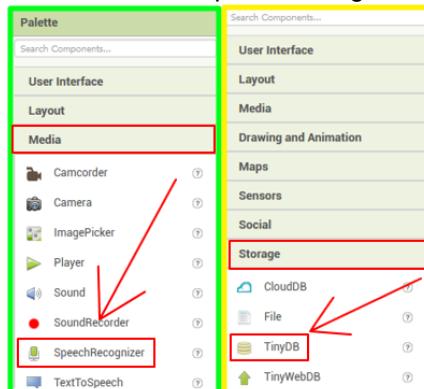
Fuente: Propia

Finalmente para esta pestaña, se hizo la programación para el botón de vuelta, en la cual se volvió a utilizar la programación de la pestaña de “Control”.

Para la pestaña de “Control de voz” se repitieron los ajustes ya realizados en las pestañas anteriores para el diseño. Añadiendo unos componentes extras como “SpeechRecognizer” y “TinyDB”. La herramienta “SpeechRecognizer” tiene la función de reconocer la voz del usuario y convertirla en texto. La herramienta “TinyDB” tiene la función de guardar información, incluso si la aplicación se cierra.

Estos componentes se encuentran en “Palette” » “Storage” » “TinyDB” y “Palette” » “Media” » “SpeechRecognizer”. Tal como se observa en la Ilustración 174.

Ilustración 174. Ubicación "SpeechRecognizer" y "TinyDB"

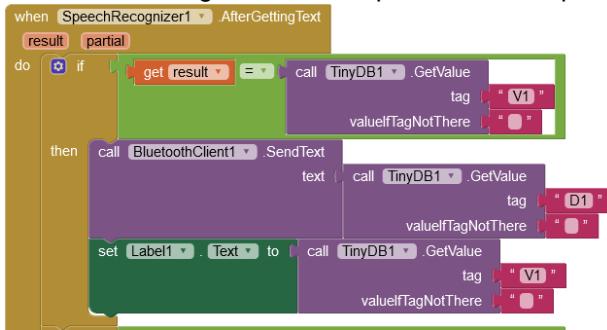


Fuente: Propia

En las propiedades de “SpeechRecognizer” y “TinyDB” se dejaron las propiedades por defecto. Pasando a la programación por bloques, para esta parte de la aplicación se dividió en dos pestañas. En la primera pestaña se hizo la programación para reconocer solamente la voz del usuario y en la segunda pestaña se hizo la programación para guardar los caracteres que se enviaban por medio del puerto serial.

Empezando por la primera parte de la programación de “Comando por voz” se utilizó un bloque perteneciente a “SpeechRecognizer” llamado “when SpeechRecognizer AfrerGettingText”, al que se le conectó un bloque “if then”. A la entrada “if” se insertó un bloque de igualdad, a este se le ingresó un bloque “get result” y “call TinyDB GetValue” a este último bloque se le conectó a sus entradas bloques de tipo chart. En la entrada “then” se conectó un bloque de bluetooth “call BluetoothClient SendText” y un bloque “call TinyDB GetValue” al que se le conectaron bloques de tipo chart en sus entradas. Por último, se añadió un bloque más para indicar al usuario el valor que se reconoce por medio de la voz. Se utilizó un bloque “set label Text to” y a este se le conectó un “call TinyDB GetValue”, a las entradas de este bloque se le conectaron bloques de tipo string. Tal como se observa en la Ilustración 175.

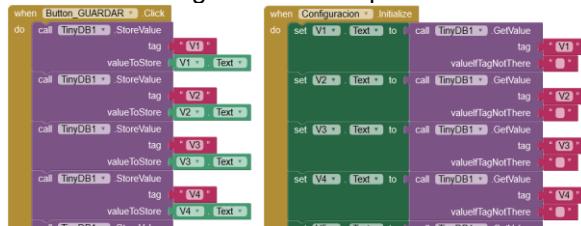
Ilustración 175. Programación 1ra parte comando por voz



Fuente: Propia

Esta misma programación se utilizó por cada uno de los componentes a guardar caracteres. Una vez realizada la programación se pasó a programar el botón que lleva a la segunda pestaña. Se utilizó la misma programación del botón de volver, pero en vez de enviar a la pestaña “Menú” se dirige a la pestaña “Configuración”. En esta pestaña se utilizó la siguiente programación.

Ilustración 176. Programación 1ra parte comando de voz



Fuente: Propia

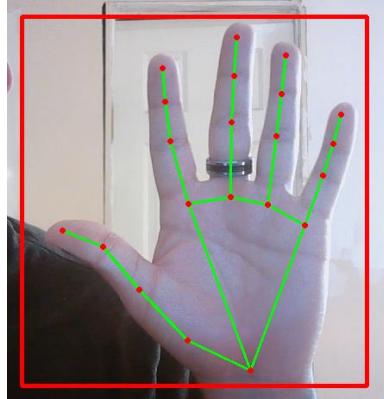
Se utilizó un bloque de “when SpeechRecognizer AfterGettingText” al que se le conectaron bloques “call TinyDB StoreValue”. A la primera entrada se le conectó un bloque de tipo string y a la segunda entrada un bloque “label Text”. Esta programación se repitió en todos los espacios donde el usuario guardaría caracteres. Tal como se observa en la Ilustración 176.

Siguiendo con la programación se utilizó un bloque “when nombre de pestaña Initialize”, al que se le insertaron bloques “set Label Text to”. A estos bloques se les conectaron “call TinyDB GetValue” junto con bloques string. Tal como se observa en la Ilustración 176.

Una vez terminada la programación para cada pestaña, se probó la aplicación con el emulador de parte de App Inventor, el cual se descargó de la página oficial.

Elaboración control por gesturas

Ilustración 177. Puntos de referencia para la mano



Fuente: Propia

Para la elaboración de control por gesturas se utilizó programación en Python. La cual tiene la ventaja de trabajar con OpenCV y Mediapipe. Utilizando pycharm, se instalaron las librerías “cv2”, la librería “mediapipe” y la librería “pyfirmata”. Tal como se muestra en la Ilustración 178.

Ilustración 178. Librerías para Python

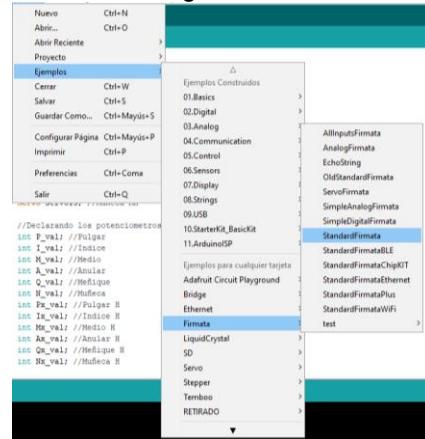
```
import cv2          #Libreria OpenCV
import mediapipe as mp #Libreria de mediapipe
from pyfirmata import Arduino, SERVO
```

Fuente: Propia

La librería “cv2”, nos permite utilizar las herramientas de OpenCV para el uso de la cámara web y el proceso de imágenes. La librería “mediapipe” nos permite utilizar los logaritmos de detección y herramientas de mediapipe. Y la librería “pyfirmata” nos permite tomar control sobre el arduino para programar desde Python.

De parte del arduino se tiene que cargar el programa de firmata para lograr esta comunicación. Tal como se observa en la Ilustración 179.

Ilustración 179. Programa Firmata en Arduino



Fuente: Propia

Una vez instaladas las librerías, se pasa a declarar las salidas del arduino.

Ilustración 180. Declarando salidas Arduino en Python

```
board = Arduino('com4') # Declarando puerto de arduino DUE
board.digital[2].mode = SERVO #Pulgar
board.digital[3].mode = SERVO #Indice
board.digital[4].mode = SERVO #Mediano
board.digital[5].mode = SERVO #Anular
board.digital[6].mode = SERVO #Meñique
board.digital[7].mode = SERVO #Muñeca
board.digital[8].mode = SERVO #Pulgar_Horizontal
board.digital[9].mode = SERVO #Indice_Horizontal
board.digital[10].mode = SERVO #Mediano_Horizontal
board.digital[11].mode = SERVO #Anular_Horizontal
board.digital[12].mode = SERVO #Meñique_Horizontal
board.digital[13].mode = SERVO #Muñeca_Horizontal
```

Fuente: Propia

Después se declaran las posiciones iniciales de los servomotores en la mano robótica.

Ilustración 181. Posición inicial de servos en Python

```
# Posiciones iniciales de la mano robótica
board.digital[2].write(179) #Pulgar
board.digital[3].write(179) #Indice
board.digital[4].write(179) #Medio
board.digital[5].write(1) #Anular
board.digital[6].write(1) #Meñique
board.digital[7].write(130) #Muñeca
board.digital[8].write(60) #Pulgar_Horizontal
board.digital[9].write(48) #Indice_Horizontal
board.digital[10].write(75) #Medio_Horizontal
board.digital[11].write(66) #Anular_Horizontal
board.digital[12].write(67) #Meñique_Horizontal
board.digital[13].write(110) #Muñeca_Horizontal
```

Fuente: Propia

Se continua con algunas declaraciones como las dimensiones que se desean utilizar en la pantalla que se mostrará. Algunas variables para guardar

valores de herramientas para el algoritmo de la mano, variables para guardar las dimensiones de la pantalla y variables para las operaciones de las posiciones de la mano en la pantalla. Tal como se observa en la Ilustración 182.

Ilustración 182. Variables en Python

```
# Dimensiones de la pantalla
screen_width = 1280
screen_height = 1024

# Variables para las herramientas del modelo
# de la mano de mediapipe
mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands

# Parametros para la salida de video en la pantalla
hands = mp_hands.Hands(min_detection_confidence=0.7,
                        min_tracking_confidence=0.8,
                        max_num_hands=1)
cap = cv2.VideoCapture(1) # Camara Web
cap.set(3, screen_width)
cap.set(4, screen_height)
cap_width = int(cap.get(3))
cap_height = int(cap.get(4))

# Variables para operaciones detección de mano
no = 1000 #Número de multiplicación
new_min = int(1) #Valor mínimo de servomotores
new_max = int(179) #Valor máximo de servomotores
```

Fuente: Propia

Después se inicia el WHILE que mantendrá la pantalla abierta con sus instrucciones para detección de mano y movimiento.

Ilustración 183. Inicio de bucle para detección de mano

```
while cap.isOpened():
    success, image = cap.read()
    if not success:
        print("No se detecta cámara")
        continue
    # Voltea la imagen inversamente para el reflejo en la pantalla
    # y convierte la imagen BGR a RGB
    image = cv2.cvtColor(cv2.flip(image, 1), cv2.COLOR_BGR2RGB)
    results = hands.process(image)
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    if results.multi_hand_landmarks:
```

Fuente: Propia

Se pasa a la sección en que se declara las variables de las herramientas que detectan los movimientos del eje X y el eje Y en la pantalla. Estas variables nos permitirán utilizar un rango de valores aceptable para los servomotores, algo parecido al mapeo que se utiliza en Arduino.

Ilustración 184. Variables para eje X y eje Y

```
# Posiciones de los dedos en el eje X y eje Y
pulgarx = int(no * (results.multi_hand_landmarks[0].landmark[mp_hands.HandLandmark.THUMB_TIP].x))
pulgary = int(no * (results.multi_hand_landmarks[0].landmark[mp_hands.HandLandmark.THUMB_TIP].y))
indicex = int(no * (results.multi_hand_landmarks[0].landmark[mp_hands.HandLandmark.INDEX_FINGER_TIP].x))
indicey = int(no * (results.multi_hand_landmarks[0].landmark[mp_hands.HandLandmark.INDEX_FINGER_TIP].y))
mediox = int(no * (results.multi_hand_landmarks[0].landmark[mp_hands.HandLandmark.MIDDLE_FINGER_TIP].x))
medioy = int(no * (results.multi_hand_landmarks[0].landmark[mp_hands.HandLandmark.MIDDLE_FINGER_TIP].y))
anularx = int(no * (results.multi_hand_landmarks[0].landmark[mp_hands.HandLandmark.RING_FINGER_TIP].x))
anulary = int(no * (results.multi_hand_landmarks[0].landmark[mp_hands.HandLandmark.RING_FINGER_TIP].y))
meniquex = int(no * (results.multi_hand_landmarks[0].landmark[mp_hands.HandLandmark.PINKY_TIP].x))
meniquey = int(no * (results.multi_hand_landmarks[0].landmark[mp_hands.HandLandmark.PINKY_TIP].y))
nuñecax = int(no * (results.multi_hand_landmarks[0].landmark[mp_hands.HandLandmark.PINKY_MCP].x))
nuñecay = int(no * (results.multi_hand_landmarks[0].landmark[mp_hands.HandLandmark.PINKY_MCP].y))
```

Fuente: Propia

Una vez que se obtienen los rangos en la pantalla, tanto para el eje X, que ayudara a los movimientos horizontales; y el eje Y, que permite los movimientos verticales. Se utiliza una conversion de valores para adaptar un rango aceptable para los servomotores. Tal como se observa en la Ilustración 185

Ilustración 185. Conversión para rango de servos

```
# Condición para efectuar movimientos de La mano robotica
if muñecay >= 750 and muñecay <= 880 and muñecax >= 585 and muñecax <= 700:

    # Pulgar
    Py_min = int(580)
    Py_max = int(10)
    Py = int((ouptpx - Py_min) / (Py_max - Py_min)) * (new_max - new_min) + new_min
    if Py >= new_min and Py <= new_max:
        # print(med_y)
        board.digital[2].write(Py)

    # Indice
    Iy_min = int(400)
    Iy_max = int(400)
    Iy = int((indicey - Iy_min) / (Iy_max - Iy_min)) * (new_max - new_min) + new_min
    if Iy >= new_min and Iy <= new_max:
        # print(med_y)
        board.digital[3].write(Iy)

    # Medio
    My_min = int(400)
    My_max = int(240)
    My = int((medioy - My_min) / (My_max - My_min)) * (new_max - new_min) + new_min
    if My >= 10 and My <= new_max:
```

Fuente: Propia

Se continua con los dibujos que aparecerán en la pantalla del usuario. Donde se podrán apreciar las marcas de los puntos de referencia para la mano y un cuadro de color rojo para colocar la mano, ayudando al usuario a no salirse de los rangos establecidos. Se termina con las instrucciones para cerrar la pantalla presionando la tecla “esc”, finalizando así el programa.

Ilustración 186. Dibujos en la pantalla e instrucciones de cierre

```
for hand_landmarks in results.multi_hand_landmarks:
    mp_drawing.draw_landmarks(image, hand_landmarks, mp_hands.HAND_CONNECTIONS)

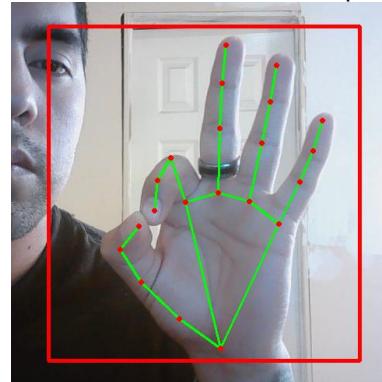
    # Se muestra un rectángulo rojo en la pantalla
    cv2.rectangle(image, (500, 150), (900, 580), color=(0, 0, 255), thickness=4)
    cv2.imshow('Hands Gesture Detection', image)
    if cv2.waitKey(1) & 0xFF == 27:
        break

hands.close()
cap.release()
```

Fuente: Propia

Si se necesitara más información del algoritmo de rastreo para la mano, se puede consultar en la página oficial de Mediapipe.

Ilustración 187. Movimientos en la mano con puntos de referencia



Fuente: Propia

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

Tomando el interés del lector acerca de las características principales del documento se redactarán primeramente conclusiones generales que se derivan de cada una de las interfaces realizadas en la documentación, las conclusiones acompañadas de los objetivos realizados en esta investigación y, finalizando, con las recomendaciones y sugerencias para futuras investigaciones a seguir.

De las interfaces propuestas en la documentación, se considera que el HMI se puede caracterizar por ser la interfaz más completa y profesional a la hora de trabajar con una mano robótica. Puesto que tiene bastantes elementos con los que el usuario puede interactuar, y tiene la ventaja de comunicarse con las demás interfaces al mismo tiempo que se está utilizando. Otra ventaja es que se pueden manejar muchísimos protocolos de comunicación, por lo que la obtención de datos y control sobre otros dispositivos la califican como una gran opción. La desventaja de esta interfaz es que es complicada de elaborar a comparación de las demás interfaces.

La interfaz del exoesqueleto tiene un aspecto muy llamativo, pero muy incómodo a la hora de utilizarse. Su elaboración llevó varias pruebas, por lo que se utilizó bastante filamento, por lo tanto, más gasto; otra desventaja es que antes de utilizarse se tienen que calibrar los rangos de valores en los potenciómetros junto con los ángulos de los servomotores antes de poder utilizarse. Mas cabe recalcar que su precisión, por medio del filtro realizado en la programación de arduino, era bastante efectivo.

Mientras que la interfaz de la aplicación de celular tiene la ventaja de elaborarse demasiado rápido, y su manipulación permite el control de manera precisa sobre la mano robótica. Sus más grandes desventajas son que la comunicación por medio de bluetooth se tiene que realizar cada vez que se cambia de pantalla y esta falla constantemente, otra desventaja es que su alcance se ve limitado por la capacidad del módulo de bluetooth.

Llegando a la interfaz de control por gesturas, resultó ser de lo más magnífica. Tiene la gran ventaja de utilizar cualquier tamaño de mano y sin necesidad de calibrar potenciómetros, la precisión de movimiento es de los más excelentes; al igual que sus librerías se descargan de una manera sencilla. Tiene la desventaja de tener la necesidad de realizar bastantes operaciones aritméticas para la detección de la mano de parte del usuario en la pantalla que se está proyectando para realizar los movimientos deseados, otra de sus desventajas son los bordes de la pantalla y la mano sobrepuerta en sí misma, pierden el rastreo correcto de la mano.

Tocando el tema de los objetivos establecidos en la presente documentación, se fijó como uno de los objetivos el analizar las necesidades de los estudiantes para elaborar una interfaz adecuada. La respuesta general de las entrevistas fue la falta de aprendizaje para programar en esta área. Con la intención de cubrir esta solicitud, se explicó muy a detalle cómo realizar las diferentes interfaces para el control de la mano robótica. Se elaboraron diferentes interfaces con la finalidad de ayudar al lector a obtener una variedad de diferentes formas para cubrir sus necesidades con respecto al tema. Otras de las respuestas que se obtuvieron en las entrevistas, fue la falta de recursos para la elaboración de proyectos similares. Con la finalidad de cubrir esta necesidad se elaboraron las interfaces con softwares de código abierto y al alcance del lector. Se explicó la obtención de archivos y librerías para realizar las tareas necesarias para el análisis y movimientos de la mano robótica desde diferentes interfaces.

Como resultados, se obtuvo finalizar con cada una de las interfaces cumpliendo las expectativas y objetivos, utilizando diferentes maneras con las que se puede manipular y monitorear una mano robótica, tanto sus ventajas como sus desventajas se vieron en el tránscurso de su elaboración, pudiendo recalcarlas para facilitar una mejor percepción acerca de las interfaces de parte del lector.

Terminando con las recomendaciones, se insta al lector a investigar acerca del lenguaje Python para la realización de futuros proyectos, este lenguaje está

obteniendo bastante popularidad y diferentes aplicaciones en el área tecnológica. Con este lenguaje se pueden entrelazar demasiados softwares, con una de las finalidades de efectuar diferentes tareas a realizar dependiendo las necesidades del usuario. Esta ventaja puede llevar al desarrollador a efectuar mejores operaciones para el control y monitoreo, tratándose del tema de la manipulación de una mano robótica.

REFERENCIAS

Amanda. (09 de Abril de 2018). *recursosdeautoayuda.com*. Obtenido de
<https://www.recursosdeautoayuda.com/metodo-analitico/>

Andrade Zeas, D. M., & Zuñiga Tenesaca, D. A. (19 de Agosto de 2011).
dspace.ups.edu.ec. Obtenido de
<https://dspace.ups.edu.ec/bitstream/123456789/1068/13/UPS-CT002117.pdf>

AUTYCOM. (s.f.). *autycom.com*. Obtenido de <https://www.autycom.com/que-es-un-sistema-hmi/>

Cahuana, J. L. (30 de mayo de 2020). *www.nettix.com*. Obtenido de Nettix:
<https://www.nettix.com.pe/documentacion/web/que-es-phpmyadmin-y-como-puedo-usarlo>

DISTRELEC. (s.f.). *www.distrelec.de*. Obtenido de
<https://www.distrelec.de/en/rotary-potentiometer-5kohm-surface-mount-bourns-3382g-252g/p/30156540>

E-Marmolejo, D. (s.f.). *hetpro-store.com*. Obtenido de <https://hetpro-store.com/TUTORIALES/microcontrolador/>

Garmendia, I. (12 de Febrero de 2016). *orekait.com*. Obtenido de
<https://orekait.com/blog/interfaz-de-usuario-historia-de-la-interfaz-hombre-mquina/>

Google Open Source. (s.f.). *opensource.google*. Obtenido de
<https://opensource.google/projects/mediapipe>

Hernandez Latorre, V. (s.f.). *www.inc.cl*. Obtenido de INC:
<https://www.inc.cl/blog/hosting/que-es-phpmyadmin>

Ingenieria Mecafenix. (21 de Abril de 2017). www.ingmecafenix.com. Obtenido de
<https://www.ingmecafenix.com/electronica/potenciómetro/>

Isaac. (s.f.). www.hwlible.com. Obtenido de <https://www.hwlible.com/arduino-due/>
Jolly, A. (s.f.). [ajolly.com](http://www.ajolly.com.mx). Obtenido de <http://www.ajolly.com.mx/es/desarrollo-consulta-sistema-supervisor-control-prueba-medida/19-LabVIEW-que-es.html>

Middleware. (s.f.). www.redhat.com. Obtenido de
<https://www.redhat.com/es/topics/middleware/what-is-ide>
[monografias.com](http://www.monografias.com). (s.f.). Obtenido de
<https://www.monografias.com/docs/Organizacion-Del-Tiempo-De-Estudio-P3J6SGFC8UNY>

Muñoz de frutos, A. (04 de Marzo de 2017). computerhoy.com. Obtenido de
<https://computerhoy.com/noticias/hardware/que-es-exoesqueleto-59152>

neoattack. (s.f.). neoattack.com. Obtenido de <https://neoattack.com/neowiki/mysql/PROGRAMO>. (s.f.). www.programoergosum.com. Obtenido de
<https://www.programoergosum.com/cursos-online/appinventor/27-curso-de-programacion-con-app-inventor/primeros-pasos>

RIPIPSA. (2019). ripipsacobots.com. Obtenido de
<https://ripipsacobots.com/brazos-roboticos-industriales/>

robologs.net. (s.f.). Obtenido de <https://robologs.net/tutoriales/tutoriales-opencv/>
Rossana, A. (30 de marzo de 2020). conceptodefinicion.de. Obtenido de
<https://conceptodefinicion.de/estudiante/>

Salud Bupa. (marzo de 2020). www.bupasalud.com.mx. Obtenido de
<https://www.bupasalud.com.mx/salud/coronavirus>

SanDoRobotics. (s.f.). *sandorobotics.com*. Obtenido de

<https://sandorobotics.com/producto/mg995/>

Shin, M. (30 de abril de 2011). *masamuneshin.wordpress.com*. Obtenido de Blog =

Release: [https://masamuneshin.wordpress.com/2011/04/30/proyecto-arduino-i-lectura-limpia-de-potencímetro-para-control-de-servomotor/](https://masamuneshin.wordpress.com/2011/04/30/proyecto-arduino-i-lectura-limpia-de-potenciómetro-para-control-de-servomotor/)

significados.com. (28 de Noviembre de 2019). Obtenido de

<https://www.significados.com/recursos/>

Sy Corvo, H. (12 de febrero de 2021). *www.lifeder.com*. Obtenido de

<https://www.lifeder.com/interfaz-informatica/>

TRESDE. (24 de Febrero de 2020). *tresde.pe*. Obtenido de [https://tresde.pe/todo-](https://tresde.pe/todo-lo-que-debes-saberacerca-del-filamento-pla/)

[lo-que-debes-saber-acerca-del-filamento-pla/](#)

Velez Ortiz, T. F. (2015). *dspace*. Obtenido de [despace.ucuenca.edu.ec:](https://dspace.ucuenca.edu.ec:file:///C:/Users/HPCORE~1/AppData/Local/Temp/tesis-3.pdf)

<file:///C:/Users/HPCORE~1/AppData/Local/Temp/tesis-3.pdf>

www.atecnologia.com. (s.f.). Obtenido de

<https://www.atecnologia.com/informatica/impresoras-3d.html>

www.ecured.cu. (s.f.). Obtenido de

https://www.ecured.cu/Dise%C3%B1o_de_Interfaces_de_Usuario

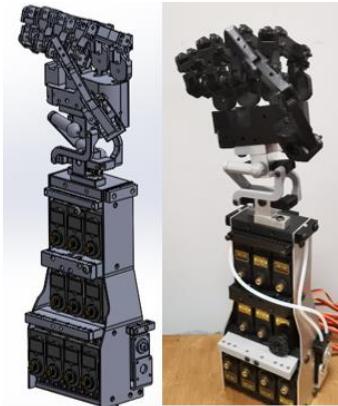
www.ionos.mx. (03 de Septiembre de 2019). Obtenido de IONOS:

<https://www.ionos.mx/digitalguide/servidores/herramientas/instala-tu-servidor-local-xampp-en-unos-pocos-pasos/>

ANEXOS

Elaboración de la mano robótica

Ilustración 188. Mano robótica doble vista

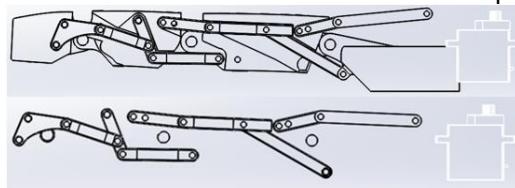


Fuente: Dionicio de la Cruz (2020)

La mano robótica se diseñó y armo por Raúl Dionicio de la Cruz, estudiante de la Universidad Tecnológica de Tijuana en la carrera ingeniería de mecatrónica para el proyecto grupal del décimo cuatrimestre. A continuación, se mostrará una explicación de su historia y elaboración.

Inicialmente se elaboró como prototipo un dedo con múltiples piezas para su movimiento por medio de un servo motor.

Ilustración 189. Diseño CAD dedo robótico de prueba



El cual se podía extender y retraer con un poco de dificultad. Se diseñó en Solid Works con un total de 19 piezas. Tal como se muestra en la Ilustración 190, Ilustración 191, Ilustración 192, Ilustración 193 e Ilustración 194.

Ilustración 190. Diseño CAD dedo robótico de prueba contraído

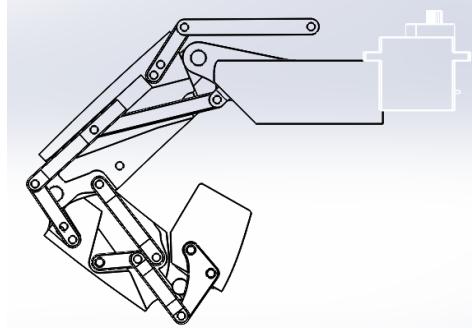


Ilustración 191. Diseño CAD dedor robotico de prueba vista trasnparente

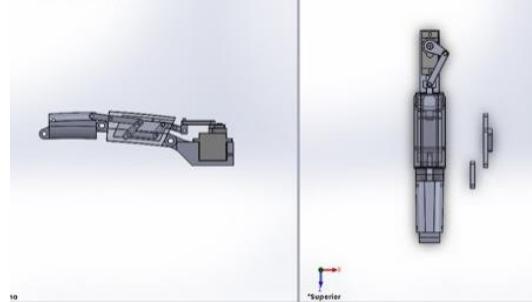


Ilustración 192. Diseño CAD dedo robótico de prueba contraído vista transparente

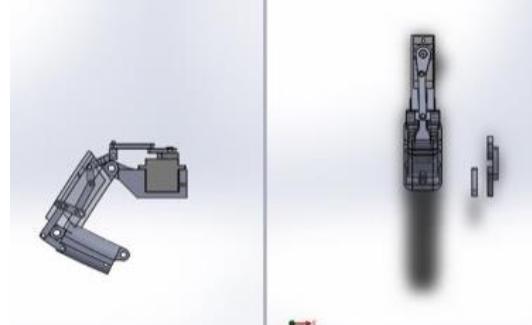


Ilustración 193. Diseño CAD dedo robótico de prueba contraído con vista de sección

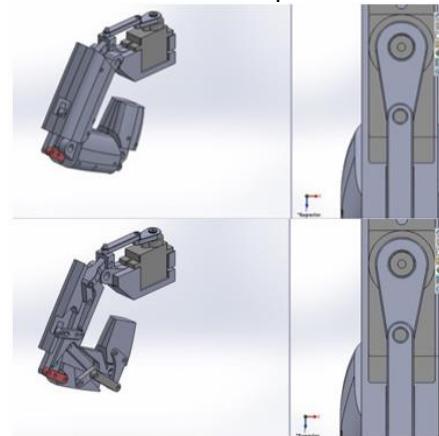
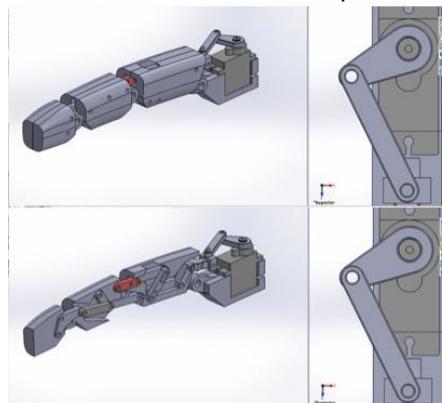


Ilustración 194. Diseño CAD dedo robótico de prueba con vista de sección



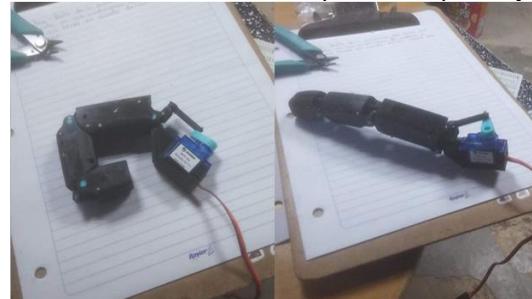
Se imprimió con un filamento PLA

Ilustración 195. Impresiones 3D dedo robótico de prueba



Una vez impreso, se armó y se obtuvo el resultado de la Ilustración 196.
Una de las desventajas de este prototipo era el tamaño y la dificultad de movimiento. Así que se optó por realizar un nuevo prototipo.

Ilustración 196. Dedo robótico de prueba impreso y armado



Investigando se encontró un canal en YouTube llamado “Will Cogley”, en el que se muestran varios proyectos avanzados de exoesqueletos y manos robóticas.

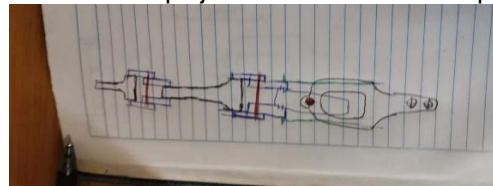
Ilustración 197. Mano robótica del canal de Will Cogley



Fuente: Will Cogley (2018)

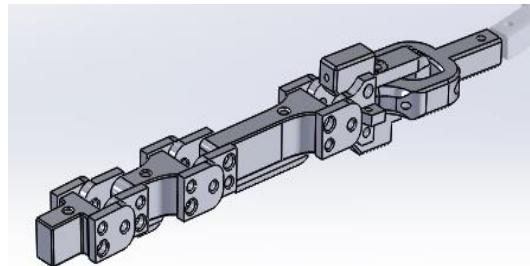
De este canal se obtuvieron ideas para crear el prototipo a continuación.

Ilustración 198. Bosquejo dedo robótico nuevo prototipo



Se hizo un bosquejo para obtener una mejor idea de lo que se elaboraría en Solid Works.

Ilustración 199. Diseño dedo robótico



Inicialmente, se realizaron los dibujos de los dedos en Solid Works. Una vez finalizados los diseños CAD, se imprimieron las piezas con filamento PLA. Tal como se observa en la Ilustración 200 e Ilustración 201.

Ilustración 200. Impresiones 3D eslabones dedo robótico



Ilustración 201. Impresiones 3D dedo robótico



Viendo la funcionalidad y efectividad del movimiento de los dedos se continuo el diseño CAD de la mano. Tal como se observa en la Ilustración 202 e Ilustración 203.

Ilustración 202. Diseño CAD mano robótica vista lateral

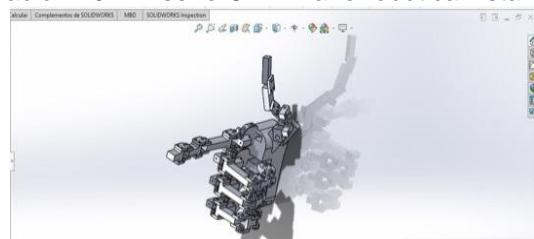
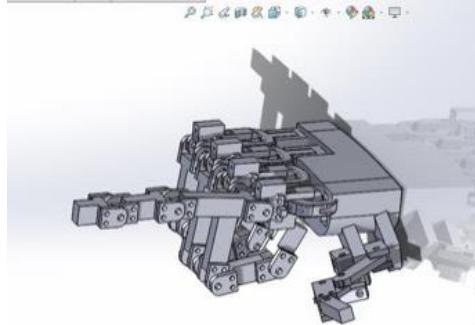
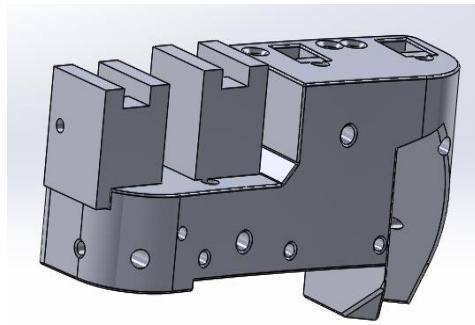


Ilustración 203. Diseño CAD mano robótica



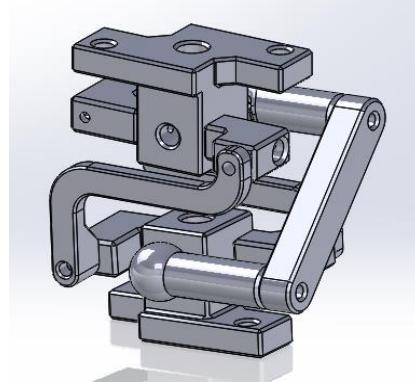
Después de la elaboración de los dedos, se diseñó el dorso de la mano robótica.

Ilustración 204. Diseño CAD dorso



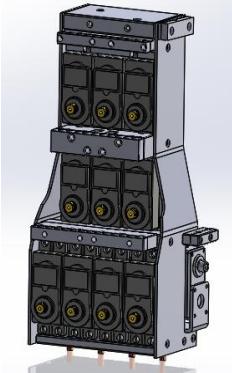
A continuación, se elaboró el diseño de la muñeca.

Ilustración 205. Diseño CAD muñeca



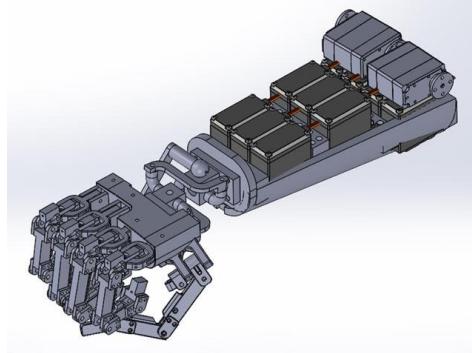
Continuando con el diseño del antebrazo, en donde se colocarían los servomotores.

Ilustración 206. Diseño CAD antebrazo con motores



Una vez terminados las partes de la mano, se ensamblaron unos con otros.

Ilustración 207. Diseño CAD mano robótica vista acostada



Se imprimieron las piezas con material PETG, ya que este material tiene la ventaja de ser más resistente.

Ilustración 208. Impresiones 3D mano robótica



Continuando con el diseño, se maquinaron las laterales del antebrazo.

Ilustración 209. Maquinado pieza de metal



Ilustración 210. Pieza con cortes en taller de maquinado

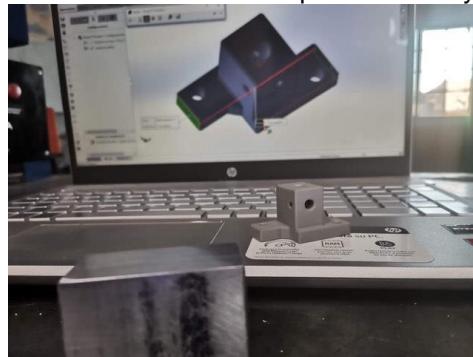


En el taller de maquinado también se maquinó el conector del antebrazo con la muñeca. Puesto que esta pieza necesitaría bastante resistencia para soportar el movimiento y peso de la mano.

Ilustración 211. Piezas maquinadas terminadas



Ilustración 212. Pieza de conexión para muñeca y antebrazo



Una vez impresas las piezas y maquinadas, se empezaron las conexiones de los soportes del antebrazo para colocar los servomotores.

Ilustración 213. Armado antebrazo con motores



Ilustración 214. Ensamble motores y antebrazo



Llevando así a insertar 12 motores en el antebrazo. 10 servo motores de 12 kilos se colocaron en la parte frontal y 2 servomotores de 25 kilos se colocaron en la parte trasera del antebrazo.

Ilustración 215. Ensamble motores terminado



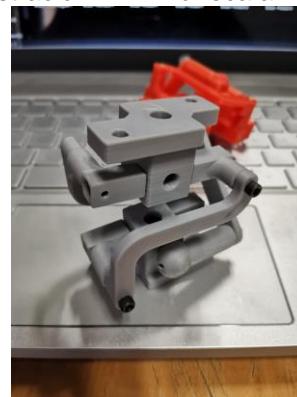
Se prosiguió a hacer las primeras conexiones de los dedos con el dorso.

Ilustración 216. Palma con bases para dedos



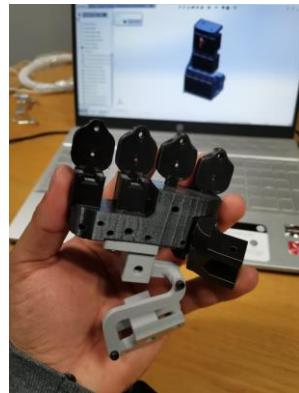
Y después, a la elaboración de la muñeca.

Ilustración 217. Muñeca armada



Una vez terminada la muñeca, esta se unió con el dorso.

Ilustración 218. Palma con muñeca



Y, después, se conectó con el antebrazo.

Ilustración 219. Palma con muñeca y antebrazo



Terminando las conexiones con los dedos.

Ilustración 220. Mano robótica armada



Finalmente se añadieron tubos delgados para usarse como guías para el hilo que ayudaría al cerrado de los dedos. Para la retracción de los dedos se utilizaron ligas. Para que pudiera sostener objetos, se consiguió un material de tapicería y se pegó a los eslabones de los dedos.

Ilustración 221. Mano robótica terminada vista frontal



Ilustración 222. Mano robótica terminada vista trasera



Ilustración 223. Mano finalizada

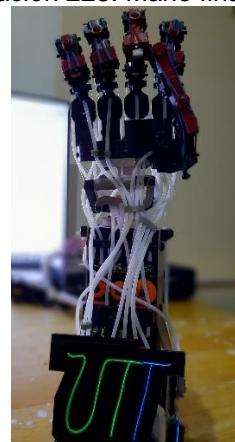
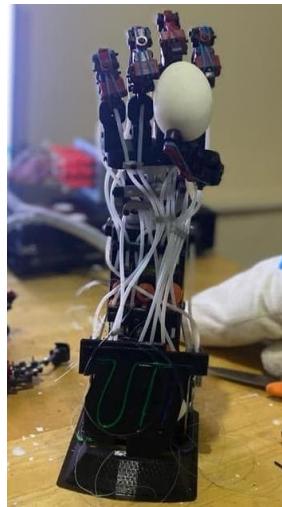


Ilustración 224. Mano sosteniendo objeto delicado



Elaboración caja de control

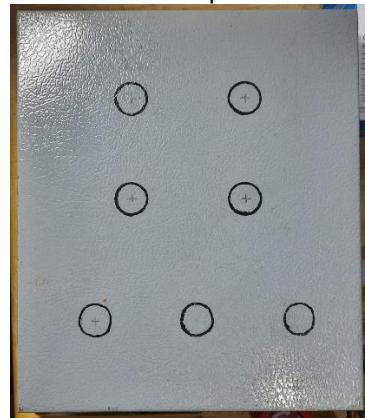
Ilustración 225. Caja de control



Para la elaboración de la caja de control, se llevó a un taller de maquinado. En el taller se efectuaron los agujeros necesarios para la inserción de indicadores y botones.

Se inició tomando medidas y dibujando marcas para la realización de los agujeros.

Ilustración 226. Marcas para realizar orificios



En el taller de maquinado, se utilizó la fresadora para elaborar los agujeros. Tal como se observa en la Ilustración 227, Ilustración 228 e Ilustración 229.

Ilustración 227. Maquinado caja de control



Ilustración 228. Maquinado caja de control 2

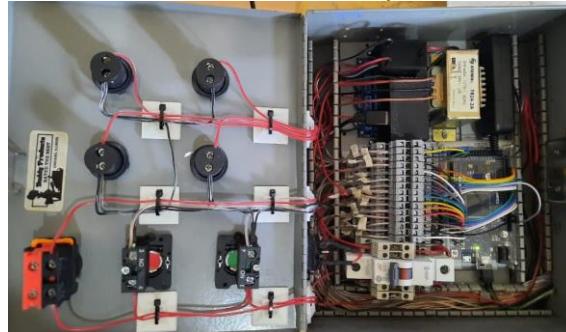


Ilustración 229. Maquinado caja de control 3



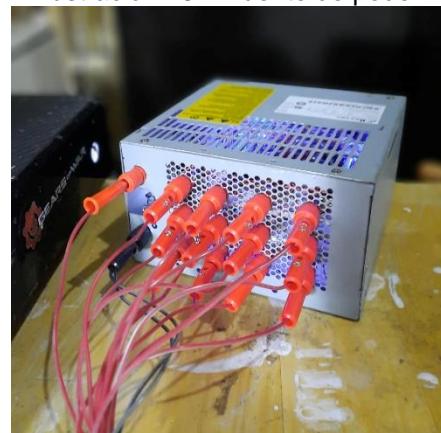
Una vez terminado los agujeros, se colocaron los componentes y se realizó el cableado para la manipulación y monitoreo de la mano robótica.

Ilustración 230. Cableado caja de control



Elaboración fuente de poder

Ilustración 231. Fuente de poder

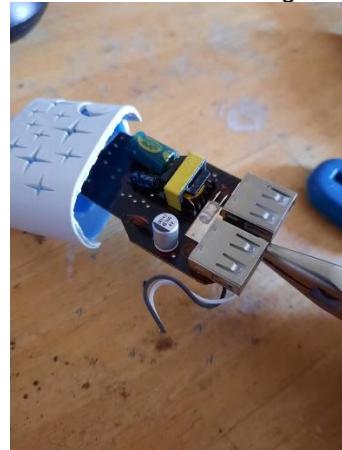


Para la fuente de poder, se utilizó una fuente de computadora. A la cual se le vacío su interior para utilizar nuevos componentes. Se le hicieron agujeros para conectar bornes sobre ella. También se abrieron los cargadores de viaje para sacar de ellos sus protoboards. Tal como se observa en la Ilustración 232 e Ilustración 233.

Ilustración 232. Cargador Kingsman



Ilustración 233. Protoboard de cargador Kingsman



Estos protoboards se colocaron de 4 en 4, en piezas de plástico formando 3 pisos. Después se puentearon las entradas de 110\120 CA y se cablearon las salidas de 5v a los bornes. Tal como se observa en la Ilustración 234.

Ilustración 234. Fuente de poder vista interior

