Industria y Comercio

SUPERINTENDENCIA

DELEGATURA DE PROPIEDAD INDUSTRIAL

SOLICITUD

PATENTE DE INVENCIÓN

SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO
No. 15-278320- -00000-0000
Fecha: 2015-11-23 09:36:10 Dep: 2020 DIR NUEVASCF
Tra: 2 PATENTES Eve: 1 REGDEPOSITO
Act: 411 PRESENTACION Folios: 446 447

21. **EXPEDIENTE No.** _____

54. **TÍTULO** PROTOTIPO EXOESQUELETO DE MANO Y MUÑECA CON APLICACIONES DE REHABILITACION Y ASISTENCIA " PROTOTIPO EMMRA - 1"

51. **CLASIFICACIÓN INTERNACIONAL** _____

71. **SOLICITANTE** Juan David Moreno Arango

DOMICILIO Calle 56A #18-27 Manizales/Caldas

74. **APODERADO** _____

22. **BOGOTÁ, D. C.,** _____

(FORMA P-10)

209984 Imprenta Nacional de Colombia 2014

Industria y Comercio
SUPERINTENDENCIA

SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO

No. 15-278320- -00000-0000

Fecha: 2015-11-23 09:36:10 Dep: 2020 DIR NUEVAS CR
Tra: 2 PATENTES Eve: 1 REGDEPOSITO
Act: 411 PRESENTACION Folios: 447

1 / 447

DIRECCIÓN DE NUEVAS CREACIONES
SOLICITUD DE PATENTE - NACIONAL

1 TIPO DE SOLICITUD ☒ Patente de invención ☐ Patente de Modelo de Utilidad

2 TÍTULO DE LA INVENCIÓN (200 caracteres o espacios máximos)

PROTOTIPO EXOESQUELETO DE MANO Y MUÑECA CON APLICACIONES DE REHABILITACION Y ASISTENCIA "PROTOTIPO EMMRA-1"

3 CLASIFICACIÓN INTERNACIONAL (CIP)

4 SOLICITANTE (S) ☒ Esta persona también es inventor. Para datos adicionales utilizar hoja de información complementaria

	APELLIDOS O RAZÓN SOCIAL	NOMBRE	IDENTIFICACIÓN	TIPO
1	MORENO ARANGO	JUAN DAVID	75076197	CC

5 DATOS DEL SOLICITANTE

DIRECCIÓN Calle 56A No. 18-27
CIUDAD Manizales
DEPARTAMENTO/ESTADO Caldas
PAÍS DE RESIDENCIA Colombia

No. TELÉFONO (6) 8917406
CORREO ELECTRÓNICO ciberr-m317@hotmail.com
NACIONALIDAD O LUGAR DE CONSTITUCIÓN Colombiano

6 INVENTOR (ES) Para datos adicionales utilizar hoja de información complementaria

	APELLIDOS	NOMBRES	NACIONALIDAD
1	Moreno Arango	Juan David	Colombiano
2	Moreno Arango	Julián Alberto	Colombiano
3			
4			

DIRECCIÓN DE CORREO ELECTRÓNICO: ciberr-m317@hotmail.com

7 DATOS INVENTOR (ES) Para datos adicionales utilizar hoja de información complementaria

	PAÍS RESIDENCIA	DEPARTAMENTO/ESTADO	CIUDAD	DIRECCIÓN
1	Colombia	Caldas	Manizales	Calle 56A No. 18-27
2	Colombia	Caldas	Manizales	Calle 55B No. 10-27
3				

OTRO(S) SOLICITANTE(S) Y/O (OTRO(S)) INVENTOR(ES)

☐ Los demás solicitantes y/o (demás) inventores se indican en hoja de información complementaria.

8 ☐ REPRESENTANTE LEGAL ☐ APODERADO

APELLIDOS	NOMBRES	IDENTIFICACIÓN
		C.C. T.P.

DIRECCIÓN
CIUDAD
PAÍS

No. TELÉFONO
CORREO ELECTRÓNICO
No. RADICACIÓN O PROTOCOLO DE PODER GENERAL

9 DECLARACION(ES) DE PRIORIDAD ☐ SI ☒ NO

	(33) PAÍS DE ORIGEN	CÓDIGO PAÍS	(31) NUMERO	(32) FECHA (AAAA/MM/DD)
1				
2				
3				

10	DECLARACIÓN SOBRE USO DE RECURSOS GENÉTICOS O BIOLÓGICOS

Declaro que el objeto de la presente solicitud de patente fue obtenido a partir de recursos genéticos o biológicos de los que cualquiera de los países miembros de la Comunidad Andina es país de origen.

☐ SI ☒ NO

Nota: En caso afirmativo deberá anexar copia del contrato de acceso de recursos genéticos o productos derivados, o certificado o número de registro, expedido por la Autoridad competente.

11	DECLARACIÓN SOBRE USO DE CONOCIMIENTOS TRADICIONALES

Declaro que el objeto de la presente solicitud de patente fue obtenido a partir de conocimientos tradicionales de comunidades indígenas, afroamericanas o locales de países miembros de la Comunidad Andina.

☐ SI ☒ NO

Nota: En caso afirmativo deberá anexar la licencia o autorización de uso de conocimiento tradicional, o certificado o número de registro.

12	REDUCCIÓN DE TASAS

Declaro que carezco de medios económicos para presentar la solicitud de patente.

☐ SI ☒ NO

Nota: En caso de ser persona natural y carecer de medios económicos, y por lo tanto, aplique la reducción de tasas a que se refiere la resolución vigente en tarifas, debe firmar la presente solicitud bajo la gravedad de juramento.

Micro, pequeñas y medianas empresas ☐

Universidades públicas o privadas ☐

Entidades sin ánimo de lucro ☐

Debe aportar los documentos que se indican en el numeral 17 de anexos

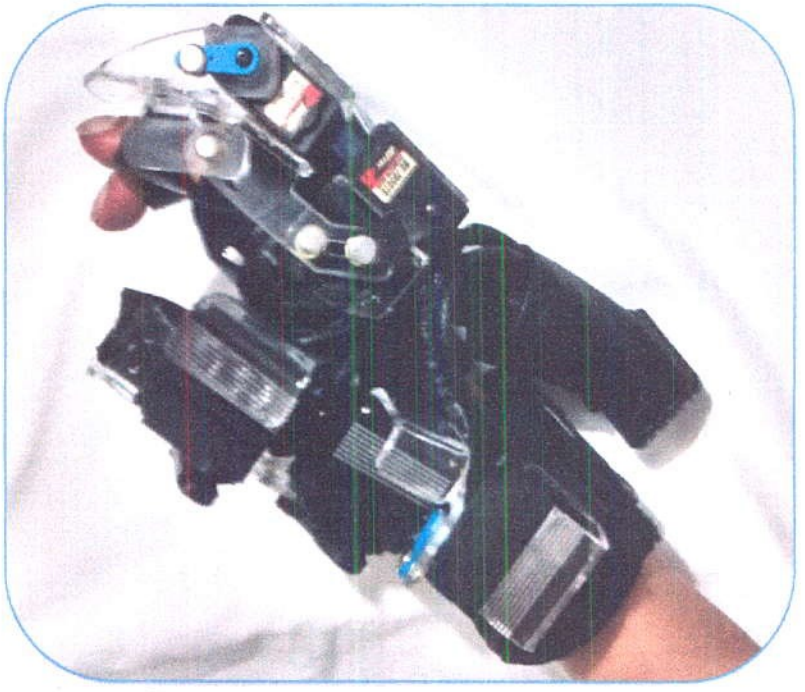
13	PARA PUBLICAR A PARTIR DE LA FECHA DE PRESENTACIÓN O DE LA PRIORIDAD INVOCADA:

Si es Patente de Invención	Si es Patente de Modelo de Utilidad
☐ 6 meses ☐ 12 meses ☒ 18 meses ☐ Otro Cual:	☐ 6 meses ☐ 12 meses ☐ Otro Cual:

14	RESUMEN

El "Prototipo EMMRA-1" es un exoesqueleto mecatrónico de mano y muñeca con aplicaciones de rehabilitación y asistencia, se compone de una interface órtesis piel (guante de cuero y neopreno, una interface órtesis pasiva (eslabones de acrílico 3mm y ejes articulares en duralumnio), una interface órtesis activa (servomotores), sensores de señal muscular, sensores de fuerza resistivos y un módulo de control electrónico por comandos de voz y electromiografía (Tarjeta EasyVR Shield 3.0 integrada a la tarjeta Arduino Mega 2560). Presenta beneficios al paciente con discapacidad motriz de mano y muñeca a causa de lesiones neurológicas centrales y periféricas, facilitando la rehabilitación en el ambiente doméstico y la asistencia en la manipulación de objetos.

15	FIGURA CARACTERISTICA



16	COMPROBANTE DE PAGO O PAGO ELECTRÓNICO	N° 68285038-3	Fecha 17/11/2015

17	FIRMA DEL SOLICITANTE. DEL APODERADO O DEL REPRESENTANTE LEGAL *Junto a cada firma, indicar el nombre del firmante y su calidad (si tal calidad no es obvia al leer el petitorio)*

• Juan David Moreno A.
• Inventor.

Juan David Moreno A.

18	DOCUMENTOS QUE ACOMPAÑAN LA SOLICITUD

Documentación Técnica

1. [X] Descripción — N° de folios: 426
2. [ ] Reivindicaciones — N° Reivindicaciones:
3. [X] Dibujos y/o figuras — N° folios:
4. [X] Resumen
5. [ ] Documento de Prioridad.
6. [ ] Traducción del documento de prioridad.
7. [ ] Certificado de depósito de material biológico si fuera el caso.
8. [ ] Listado de secuencias de nucleótidos y/o aminoácidos en forma digital si fuera el caso.
9. [X] Arte final 12 x 12.
10. [X] Anexo formato digital.

Documentación Jurídica

11. [ ] Poderes, si fuera el caso.
12. [ ] Documento que legalmente pruebe la cesión del inventor al solicitante o a su causante.
13. [ ] Copia del contrato de acceso de recursos genéticos o productos derivados, o certificado o número de registro, si fuera el caso.
14. [ ] Copia de la licencia o autorización de conocimientos tradicionales, o Certificado o número de registro, si fuera el caso.
15. **Reducción de tasas**
    - **Micro, pequeñas o medinas empresas**
      - [ ] Copia simple de la declaración de renta del año inmediatamente anterior, o en su defecto prueba documental idónea.
      - [ ] Documento de constancia de cumplimiento con lo establecido en la ley 905 de 2004.
    - **Universidades públicas o privadas**
      - [ ] Copia acto de reconocimiento institucional emitido por el Ministerio de Educación
    - **Entidades sin ánimo de lucro**
      - [ ] Copia de registro vigente en Cámara de comercio.
      - [ ] Hoja de información complementaria.
      - [ ] Otros, especificar
16. [ ] Comprobante de pago de la tasa de presentación de la solicitud.
17. [ ] Comprobante de pago por reivindicación de prioridad.
18. [ ] Comprobante de pago por reivindicación adicional a 10.

4 1

EXOESQUELETO DE MANO Y MUÑECA CON APLICACIONES DE REHABILITACION Y ASISTENCIA (PROTOTIPO EMMRA-I)

REGISTRO DE PATENTE

INVENTORES:

JUAN DAVID MORENO ARANGO

JULIAN ALBERTO MORENO ARANGO

Manizales, 17 de Noviembre de 2015

1. RESUMEN

El "Prototipo EMMRA-1" es un exoesqueleto mecatrónico de mano y muñeca con aplicaciones de rehabilitación y de asistencia, se compone de una interface órtesis-piel (guante de cuero y neopreno), una interface órtesis pasiva (eslabones en acrílico 3mm y ejes articulares en duraluminio), una interface órtesis activa (servomotores), sensores de señal muscular, sensores de fuerza resistivos y un módulo de control electrónico por comandos de voz y electromiografía (Tarjeta EasyVR Shield 3.0 integrada a la tarjeta Arduino Mega 2560). Presenta beneficios al paciente con discapacidad motriz de mano y muñeca a causa de lesiones neurológicas centrales y periféricas, facilitando la rehabilitación en el ambiente doméstico y la asistencia en la manipulación de objetos.

2. DESCRIPCIÓN DEL INVENTO

2.1 Sector tecnológico al que pertenece la invención:

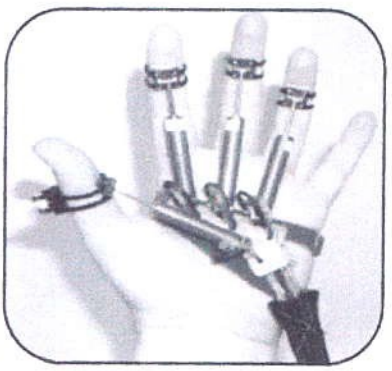
Innovación tecnológica en salud: El proyecto de exoesqueleto de mano y muñeca con aplicaciones de rehabilitación y asistencia "Prototipo EMMRA-1" hace parte del desarrollo tecnológico en nuestro país, presentará al mundo un nuevo modelo de exoesqueletos prácticos para uso doméstico. Aunque algunos de los componentes electrónicos son importados, la estructura mecánica y el software son diseños propios que cumplirán con los estándares de bioseguridad internacionales para su aplicación en humanos. El prototipo requiere ser patentado como innovación en dispositivos biomédicos para la rehabilitación de personas con discapacidad motriz en muñeca y mano a causa de lesiones neurológicas centrales y periiféricas, además, tendrá una aplicación para asistir al paciente con la manipulación de objetos para comer alimentos, escribir y agarrar objetos), sus principios de funcionamiento serán la base de conocimientos para futuras aplicaciones en modelos de exoesqueletos que permitan la rehabilitación de discapacidad motriz en otras áreas del cuerpo

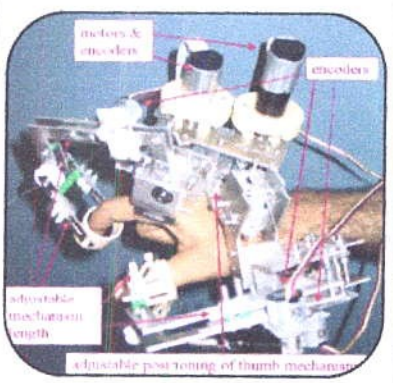
2.2 Tecnología Anterior conocida por el inventor:

- Exoesqueletos para Miembros Superiores: La cinemática del exoesqueleto se adapta a la estructura anatómica del cuerpo humano, su control puede provenir de dispositivos externos o de señales electromiográficas detectadas por sensores conectados a la piel. La señal detectada puede ser procesada y generar una respuesta comandada por actuadores analógicos a las funciones de la extremidad [1]. Los exoesqueletos de miembros superiores forman parte de un grupo de sistemas robóticos denominados robots portátiles (wereable robots), de acuerdo a sus funciones se clasifican en: a) Robots Ortópticos: Su función es reemplazar las funciones de un miembro perdido. b) Robots Prostéticos: Su función es la de rehabilitar un miembro que se encuentra con limitaciones funcionales. c) Robots extensores: Su función es la de aumentar la potencia y la fuerza del miembro al que está unido [1,2]. Las aplicaciones potenciales de los wereable robots incluyen: 1) Dispositivos diagnósticos y terapéuticos para fisioterapia, 2) Órtesis para amplificación de la potencia y la fuerza, 3) Dispositivos hápticos para simulaciones virtuales, 4) Dispositivos maestros para procesos de Teleoperación. [3]

- Estado del Arte en Exoesqueletos de Mano: Desde los años 80 muchos investigadores han estado tratando de desarrollar manos robóticas destinadas a replicar las funciones de la mano humana. Las principales aplicaciones de estos dispositivos están relacionadas con la robótica industrial y prótesis de miembros superiores. En la práctica, el exoesqueleto puede aplicar fuerzas para los dedos con el fin de obligarlos a realizar una trayectoria determinada según los Grados de Libertad (GDL), o para mejorar la fuerza que el dedo normalmente podría aplicar. Las principales aplicaciones de los exoesqueletos de mano existentes son: Dispositivos de Rehabilitación, Dispositivos hápticos, y Dispositivos de Asistencia [4]. Los exoesqueletos de mano que se encuentran publicados en revistas científicas, han sido presentados en Congresos Internacionales de Mecatrónica o Robótica de Rehabilitación. A continuación, describiremos sus principales características y desventajas frente a nuestro innovador prototipo de exoesqueleto de mano y muñeca:

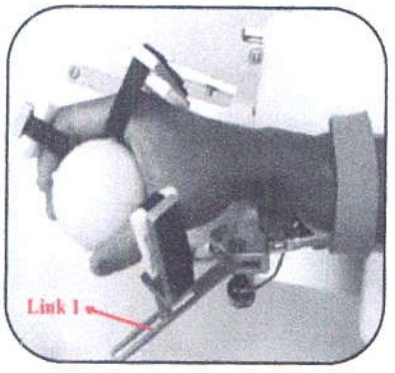
- **Estado del arte en exoesqueletos de Mano y Muñeca**:

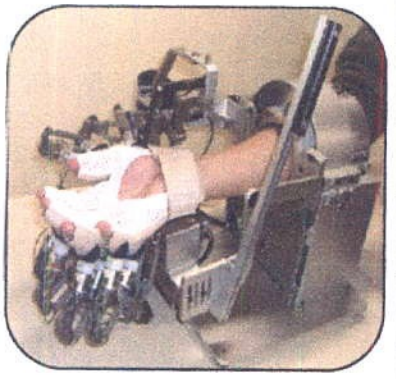
Rutgers Master II	
	**APLICACION:** Dispositivo Hápico
	GDL: (4) (1 GDL para cada dedo del I al IV)
	ESTRUCTURA: Ejes (Articulaciones) y Cables (Tendones)
	ACTUACIONES: Neumáticos
	CONTROL: Programado por Software
	SENSORES: Posición y Fuerza
	ENTORNO DE TRABAJO: Institucional
	PUBLICACION: IEEE-ASME, Junio 2002 [5]

Robotics Center-Ecole des Mines de Paris Hand	
	**APLICACION:** Dispositivo Háptico
	GDL: (6) (Pulgar 3 GDL e Índice 3 GDL)
	ESTRUCTURA: Poleas (articulaciones) y Palancas (Tendones)
	ACTUACIONES: Motor CC
	CONTROL: Programable por Software
	SENSORES: Posición
	ENTORNO DE TRABAJO: Institucional
	PUBLICACION: University Magazine, Julio 2003 [6]

Exoesqueleto de Mano	
	**APLICACION:** Plataforma dinámica para estudiar los movimientos de la mano en 3D, midiendo la posición y el torque de cada articulación a una taza de muestreo de 960Hz, además calculando la velocidad y aceleración a partir de la posición.
	GDL: (7) (flexión y extensión dedos de mano)
	ESTRUCTURA: Ejes (Articulaciones) y Actuadores (Músculos)
	ACTUADORES: 1 Servomotor para varias articulaciones
	CONTROL: Movimiento Programático
	SENSORES: Sensores de Posición
	ENTORNO DE TRABAJO: Institucional
	PUBLICACION: IEEE- ICORR, Junio 2005 [7]

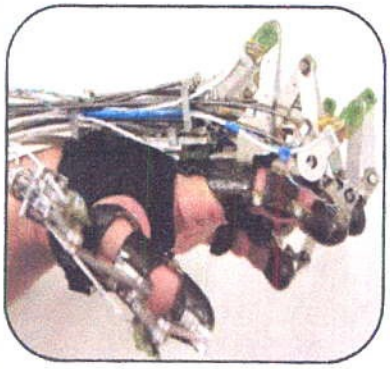
Milan University Hand	
	**APLICACION:** Rehabilitación Mano y Muñeca
	GDL: (2) (1 GDL para los dedos y 1 GDL para muñeca)
	ESTRUCTURA: Poleas (Articulaciones) y Cables (Tendones)
	ACTUADORES: Servomotores
	CONTROL: Programado por Software
	SENSORES: Posición y Fuerza Interna
	ENTORNO DE TRABAJO: Institucional
	PUBLICACION: IEEE-ICORR, Junio 2005 [8]

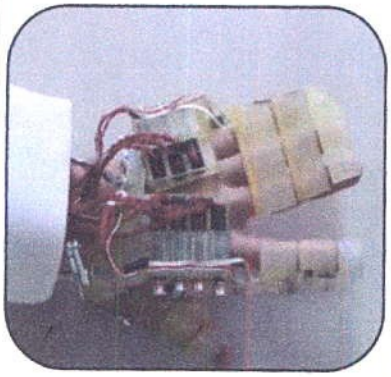
HWARD	
	**APLICACION:** Rehabilitación Mano
	GDL: (3) (2 GDL para los dedos y 1 GDL para muñeca)
	ESTRUCTURA: Giro de Palanca (Articulaciones)
	ACTUADORES: Neumáticos
	CONTROL: Programado por Software
	SENSORES: Posición y Fuerza Interna
	ENTORNO DE TRABAJO: Institucional
	PUBLICACION: IEEE-ICORR, Junio 2005 [9]

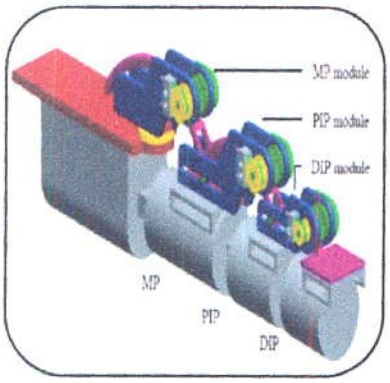
Gifu University Hand	
	**APLICACION:** Rehabilitación Mano y Muñeca
	GDL: (21) (Pulgar 4 GDL, Muñeca y otros dedos 3 GDL)
	ESTRUCTURA: Articulaciones Mecánicas y Soporte Metálico
	ACTUADORES: 1 Servomotor para cada articulación
	CONTROL: Guante de Datos en la Mano Sana
	SENSORES: Sensores de Posición y Fuerza
	ENTORNO DE TRABAJO: Institucional
	PUBLICACION: IEEE-ICORR Junio 2007 [10]

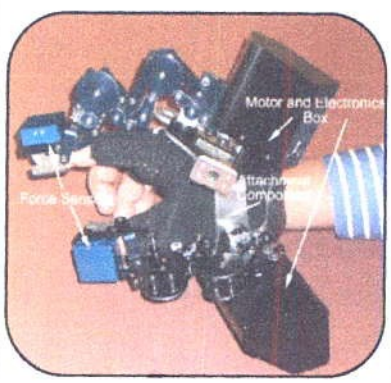
Gentle/G Hand	
	**APLICACION:** Rehabilitación Dedos de Mano
	GDL: (6) (Muñeca 3 GDL, Pulgar 1 GDL y Otros dedos 2 GDL)
	ESTRUCTURA: Piñones (articulaciones) y Cadenas (Tendones)
	ACTUADORES: Motores CC
	CONTROL: Programación por Software
	SENSORES: Sensores de Posición y Fuerza
	ENTORNO DE TRABAJO: Institucional
	PUBLICACION: IEEE-ICORS Junio 2007 [11]

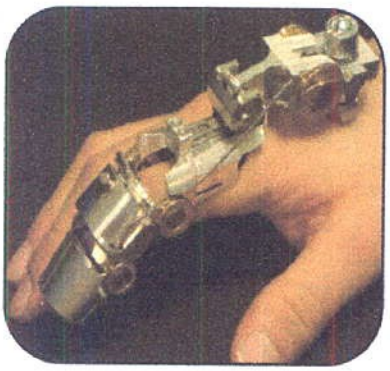
Pittsburgh University Hand	
	**APLICACION:** Rehabilitación Dedo Índice
	GDL: (2) (Flexión y Extensión 1 GDL)
	ESTRUCTURA: Bandas (articulaciones) y Cables (Tendones)
	ACTUADORES: Neumático
	CONTROL: Programación por Software
	SENSORES: No tiene
	ENTORNO DE TRABAJO: Institucional
	PUBLICACION: Journal of Robotics, Junio 2007 [12]

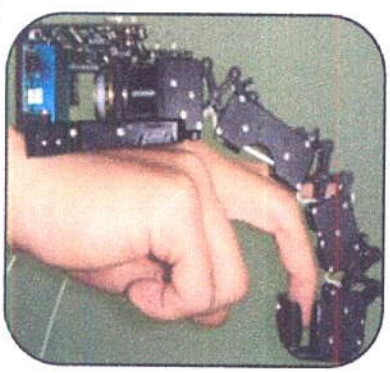
Berlin University Hand	
	**APLICACION:** Rehabilitación Mano
	GDL: (20) (flexión y extensión de todos los dedos de la mano)
	ESTRUCTURA: Poleas (Articulaciones) y Cables (Tendones)
	ACTUADORES: 4 Motores CC para cada dedo
	CONTROL: Señal EMG en el antebrazo (electrodos para EMG)
	SENSORES: Posición, Codificadores Ópticos y de Fuerza
	ENTORNO DE TRABAJO: Institucional
	PUBLICACION: IEEE-ICRB, Diciembre 2007 [13]

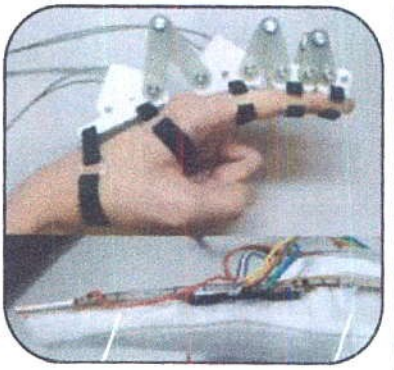
University of Tsukuba Hand	
	**APLICACION:** Rehabilitación Mano
	GDL: (8) (Pulgar 2 GDL, Índice 3 GDL, y Anular 3 GDL)
	ESTRUCTURA: Ejes (articulaciones) y Palancas (Tendones)
	ACTUADORES: Motores DC
	CONTROL: Programable por Software
	SENSORES: Posición
	ENTORNO DE TRABAJO: Institucional
	PUBLICACION: IEEE-ICORA, Mayo 2008 [14]

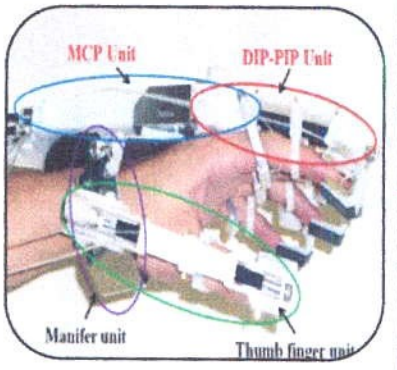
Harbin Institute of Technology Hand	
	**APLICACION:** Rehabilitación Dedos de Mano
	GDL: (3) (Flexión y Extensión Articulaciones Dedos)
	ESTRUCTURA: Piñones (articulaciones) y Cadenas (Tendones)
	ACTUADORES: Servomotores
	CONTROL: Programación por Software
	SENSORES: Sensores de Posición y Fuerza
	ENTORNO DE TRABAJO: Institucional o Doméstico
	PUBLICACION: IEEE-ICIRS Septiembre 2008 [15]

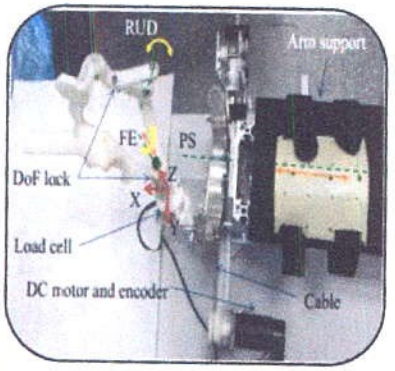
PERCRO Hand I	
	**APLICACION:** Dispositivo Háptico
	GDL: (6) (Pulgar 3 GDL e Índice 3 GDL)
	ESTRUCTURA: Piñones (articulaciones) y Palancas (Tendones)
	ACTUADORES: Motor CC
	CONTROL: Programable por Software
	SENSORES: Posición
	ENTORNO DE TRABAJO: Institucional
	PUBLICACION: IEEE-ICORR, Mayo 2009 [16]

HANDEXOS	
	**APLICACION:** Rehabilitación Mano
	GDL: (1) (1 GDL para cada dedo)
	ESTRUCTURA: Ejes y Poleas (Articulaciones) y Cables
	ACTUADORES: Motor CC
	CONTROL: Programado por Software
	SENSORES: Sin sensores
	ENTORNO DE TRABAJO: Institucional o Doméstico
	PUBLICACION: IEEE-ICIRS, Octubre 2009 [17]

DLR/HIT Hand	
	**APLICACION:** Rehabilitación Dedo Índice
	GDL: (1) (3 GDL para cada dedo)
	ESTRUCTURA: Ejes y Poleas (Articulaciones) y Cables
	ACTUADORES: Motor CC
	CONTROL: Programado por Software
	SENSORES: Posición y Fuerza. "Sensación de Tocar"
	ENTORNO DE TRABAJO: Institucional o Doméstico
	PUBLICACION: IEEE-ICIRS, Octubre 2009 [18]

HAND 1 – Universidad de Tokio	
	**APLICACION:** Rehabilitación Dedo Índice
	GDL: (2) (flexión y extensión del dedo)
	ESTRUCTURA: Poleas (Articulaciones) y Cables (Tendones)
	ACTUADORES: 1 Servomotor para cada articulación
	CONTROL: Movimiento del índice de la Mano Sana
	SENSORES: Sensores de Posición
	ENTORNO DE TRABAJO: Institucional
	PUBLICACION: IEEE-EMB, Septiembre 2009 [19]

Tokyo Institute and Technology Hand	
	**APLICACION:** Asistente para mejorar la fuerza
	GDL: (10) (2 GDL para cada)
	ESTRUCTURA: Poleas (Articulaciones) y Cables (Tendones)
	ACTUADORES: Neumático
	CONTROL: Señal Electromiográfica
	SENSORES: Posición y Fuerza
	ENTORNO DE TRABAJO: Institucional
	PUBLICACION: IEEE-ICRA, Mayo 2010 [20]

Tokyo Institute and Technology Hand	
	**APLICACION:** Estudio de fuerzas de reacción en Muñeca
	GDL: (5) (Flexión, Extensión, Pronación, Supinación, Rotación)
	ESTRUCTURA: Ejes (Articulaciones) y Joystick (Guías)
	ACTUADORES: Motores CC
	CONTROL: Programado por Software
	SENSORES: Posición y Movimiento
	ENTORNO DE TRABAJO: Institucional
	PUBLICACION: IEEE-ICORR, June 2013 [21]

2.3. Descripción en términos que permita la comprensión del problema técnico y de la solución aportada por la invención

- **Problema Técnico:**

La discapacidad es una problemática de salud pública mundial, una de las entidades más discapacitantes es la lesión medular traumática, sus principales causas son los accidentes de vehículos, la violencia y caídas en deportes extremos, afectando a la población más joven y haciendo el costo económico del tratamiento, rehabilitación y pérdida de años de vida productiva muy elevado. Aunque los avances de la medicina y la tecnología han mejorado la sobrevida y la expectativa de vida de los pacientes, se necesita impactar en su calidad de vida superando la discapacidad funcional y trascendiendo a verdaderos procesos de inclusión social. Los exoesqueletos y prótesis robóticas parecen ser la esperanza para superar las discapacidades motrices, sin embargo los modelos actuales están en fase experimental y presentan limitaciones técnicas importantes que deben resolverse, para permitir a los pacientes mayor funcionalidad en las actividades básicas de su vida diaria y la posibilidad de seguir participando efectivamente en su entorno familiar y social.

- **Solución Aportada por la invención:**

La práctica médica en el manejo de la urgencia y las complicaciones del trauma al Sistema Nervioso Central **SNC**, evolucionó en la creación de protocolos clínico-quirúrgicos con altos niveles de evidencia científica, logrando un impacto positivo en la sobrevida de los pacientes durante las fases críticas del trauma. Igualmente, los programas de rehabilitación conformados por grupos interdisciplinarios de especialistas médicos y terapeutas, han demostrado un impacto positivo en la expectativa de vida de los pacientes.

Hoy es conveniente para la medicina buscar opciones para superar las secuelas generadoras de discapacidad motriz, la esperanza está centrada en la incorporación de tecnología robótica mediante el diseño de exoesqueletos y prótesis mecatrónicas, las cuales se presume impactaran positivamente en la calidad de vida de los pacientes con Lesión Medular Traumática Cervical.

El desarrollo tecnológico de exoesqueletos para el movimiento de extremidades se proyecta como un complemento a los programas de renabilitación, permitiendo que personas discapacitadas de todo el mundo puedan realizar las terapias físicas con más frecuencia e intensidad, evitando el agotamiento de los terapeutas y disminuyendo los costos en la prestación de servicios de salud. Sin embargo, la aplicación más ambiciosa es que los exoesqueletos puedan mejorar las limitaciones físicas con respecto a las actividades manuales y la marcha, ofreciendo a los pacientes mayor independencia en sus hogares en actividades básicas y lúdicas como: caminar, comer, vestirse, peinarse, cepillarse los dientes, pasar las páginas de un libro, utilizar dispositivos electrónicos (teléfonos, controles remotos de audio y video, videojuegos, etc.)

Colombia, no es un país ajeno a la problemática social que genera la discapacidad, las víctimas sobrevivientes de la violencia y los accidentes de tránsito que demandan atención multidisciplinaria, también se han convertido en un problema de salud pública. MinSalud ha adoptado políticas en tema de discapacidad para generar conciencia en las entidades públicas y privadas sobre la necesidad de buscar soluciones orientadas a la prevención y tratamiento de este grupo de personas. La invención de nuevas tecnologías (telemedicina, órtesis, prótesis y exoesqueletos), es una estrategia trascendental para brindar muchas soluciones a las deficiencias en los programas de rehabilitación e incorporación social de los pacientes, buscando superar las limitantes geográficas en el acceso a los servicios de salud, al igual que las deficiencias del número de terapeutas y médicos.

En un principio el prototipo de exoesqueleto de mano y muñeca con aplicaciones de rehabilitación y asistencia EMMRA-1 está diseñado para los pacientes con Lesión Medular Traumática de Columna Cervical, sin embargo, su alcance puede también incluir pacientes con discapacidad motriz de mano y muñeca por lesiones medulares no traumáticas o patologías del Sistema Nervioso Central y Periférico. Es un dispositivo biomédico útil para la rehabilitación funcional de mano y muñeca, permitiendo optimizar la frecuencia y cantidad de los ejercicios de flexión y extensión de muñeca y los ejercicios de flexión, extensión y prensión de la mano. Su beneficio principal es complementar el plan casero de terapia física, lejos del ambiente institucional, sin depender de la presencia de los terapeutas.

La creación de un nuevo modelo de exoesqueletos aportará a la comunidad académica y científica novedosas aplicaciones de los conceptos teóricos en biología, física, mecánica, electrónica y arte; además, será una nueva fuente de trabajos de investigación para realizar estudios experimentales durante su aplicación en humanos.

En Colombia, la locomotora de la innovación para el apoyo del desarrollo tecnológico convocada por el Departamento Administrativo de Ciencia, Tecnología e Innovación **COLCIENCIAS**, es una plataforma ideal para promover el desarrollo, construcción y registro de patente del Prototipo EMMRA-1, el cual será diseñado y fabricado en Manizales Caldas, hará parte de la evolución tecnológica en el Eje Cafetero, tendrá un impacto en la generación de empleo regional y motivará la creación de dispositivos biomédicos e industriales en nuestro país, ya que sus principios de funcionamiento pueden tener múltiples aplicaciones médicas e industriales.

2.4 Reseña de los dibujos:

- **INTERFACE ORTESIS PIEL:**

Estas fotos son tomadas con el guante de cuero y neopreno cubriendo la mano del paciente discapacitado.



Figura 1. Guante de cuero y neopreno mano derecha cara lateral interna



Figura 2. Guante de cuero y neopreno cara mano izquierda lateral interna



Figura 3. Guante de cuero y neopreno cara palmar y dorsal mano izquierda



Figura 4. Guante de cuero y neopreno cara dorsal ambas manos

- **INTERFACE ÓRTESIS PASIVA:**

Estas fotos son tomadas con el guante de cuero y neopreno cubriendo la mano del paciente discapacitado.

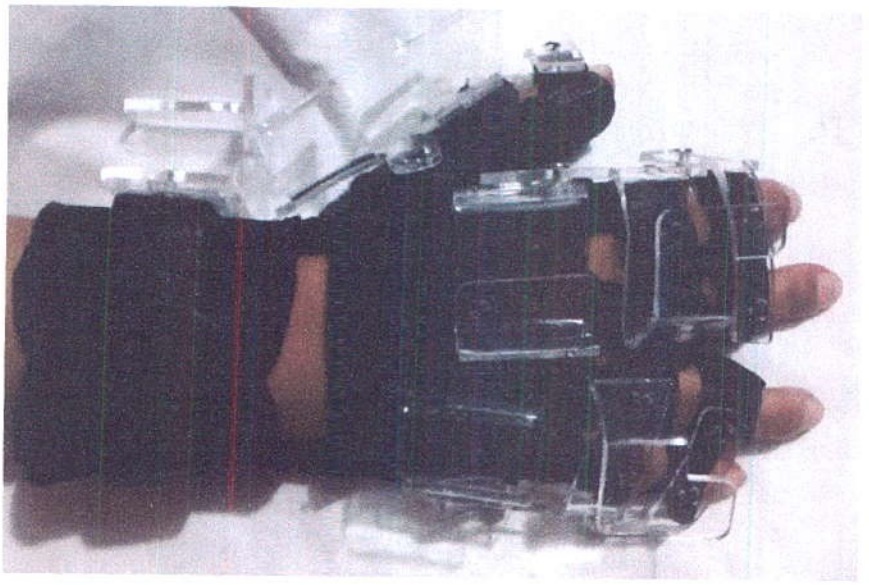


Figura 5. Órtesis Pasiva, cara dorsal mano derecha.



Figura 6. Órtesis Pasiva, cara palmar mano derecha.



Figura 7. Órtesis Pasiva, cara interna mano izquierda



Figura 8. Órtesis Pasiva, cara dorsal ambas manos.



Figura 9. Órtesis Pasiva, cara interna ambas manos.



Figura 10. Órtesis Pasiva, cara palmar ambas manos

- **ÓRTESIS ACTIVA**

Estas fotos son tomadas con el guante de cuero y neopreno cubriendo la mano del paciente discapacitado.

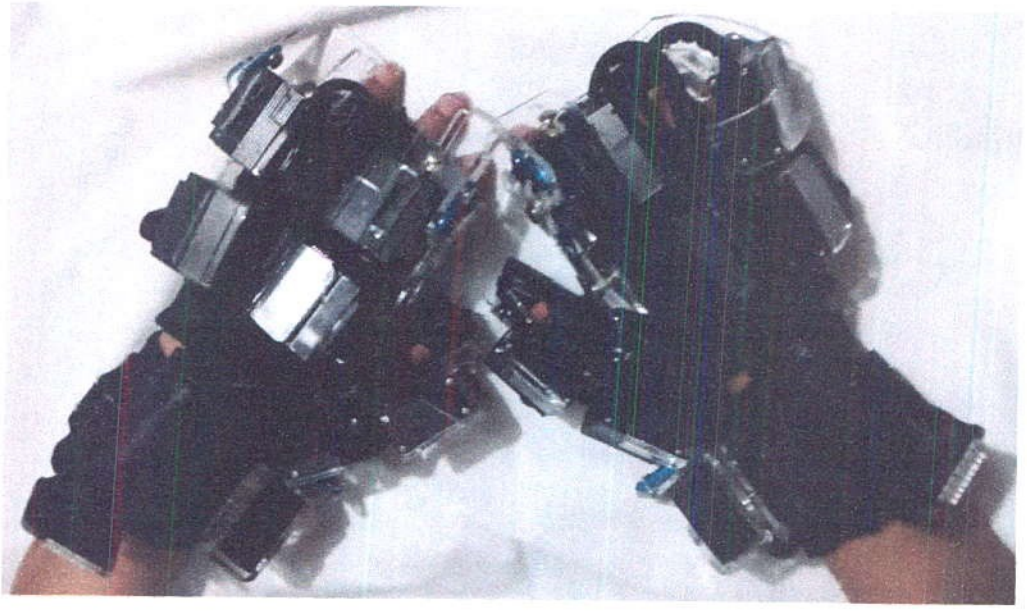


Figura 11. Exoesqueleto Mecatrónico (ambas manos – cara dorsal)

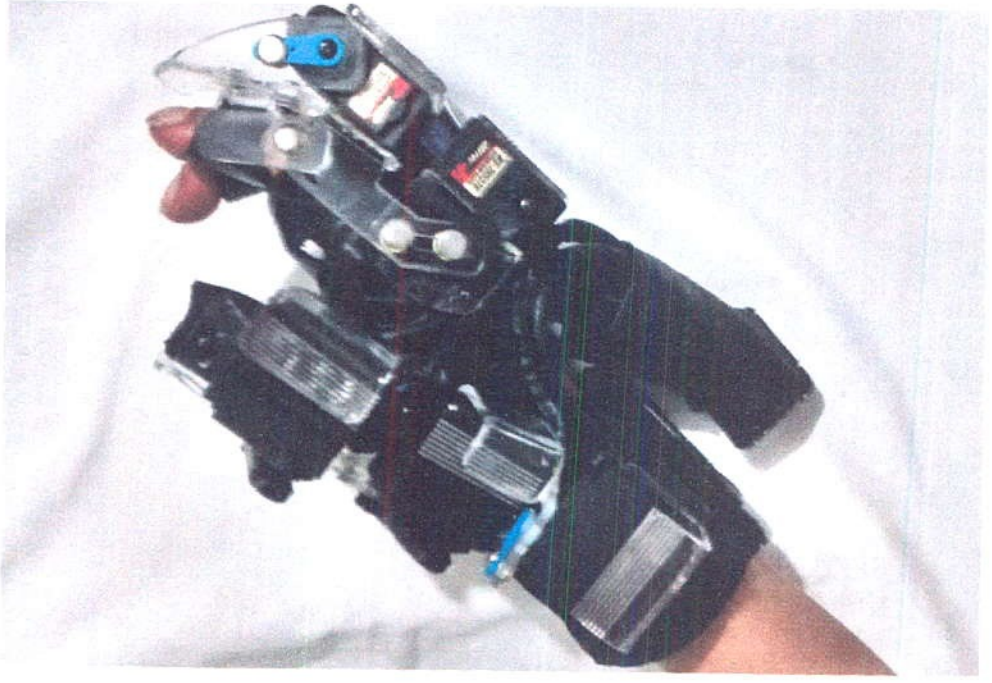


Figura 12. Exoesqueleto Mecatrónico (mano derecha – cara interna)



Figura 13. Exoesqueleto Mecatrónico (mano derecha – cara palmar)

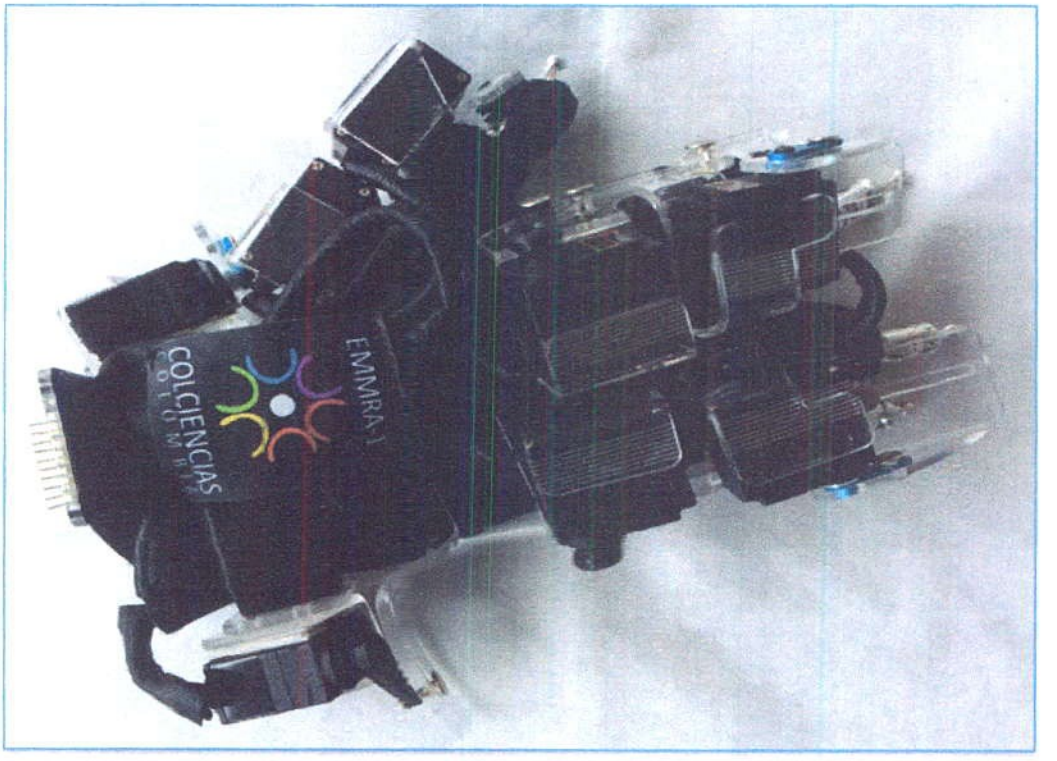


Figura 14. Exoesqueleto Mecatrónico (mano izquierda– cara dorsal)

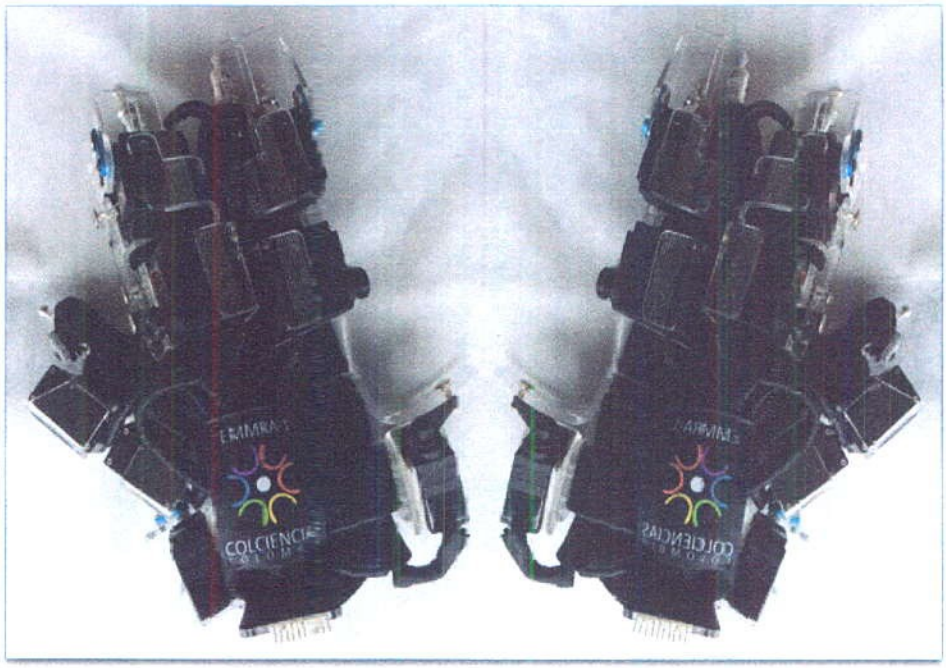


Figura 15. Exoesqueleto Mecatrónico (ambas manos– cara dorsal)

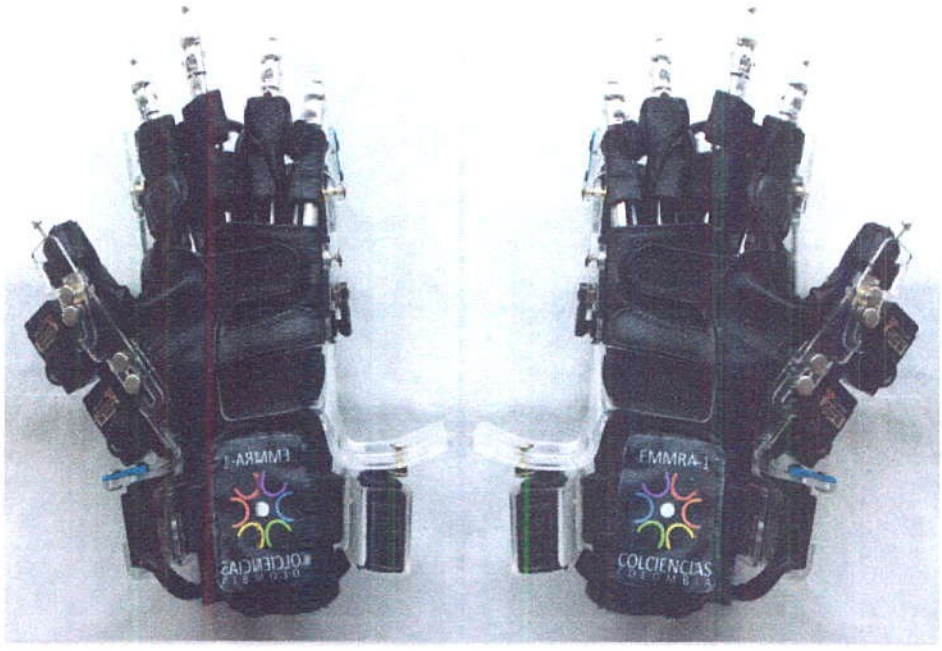


Figura 16. Exoesqueleto Mecatrónico (ambas manos– cara palmar)

- **SENSORES DE FUERZA RESISTIVOS UBICADOS EN EL GUANTE**

Ubicación de los Sensores de Fuerza Resistivos en el guante:

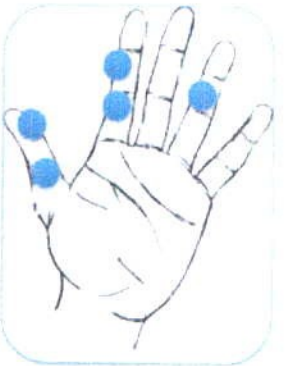


Figura 17. Ubicación de los SFR en el guante



Figura 18. Sensores de Fuerza Resistiva en el Exoesqueleto

- **SENSORES DE SEÑAL MUSCULAR UBICADOS EN LA CONEXIÓN DEL EXOESQUELETO CON EL MODULO DE CONTROL**

Los sensores de seña muscular (círculo amarillo) se ubican en la vaina de cuero que conecta el exoesqueleto con el módulo de control electrónico, a la altura de los antebrazos, tienen tres electrodos de electromiografía (círculo rojo) para sensar la actividad muscular del paciente y conexión antiestática (círculo verde) para evitar interferencias.

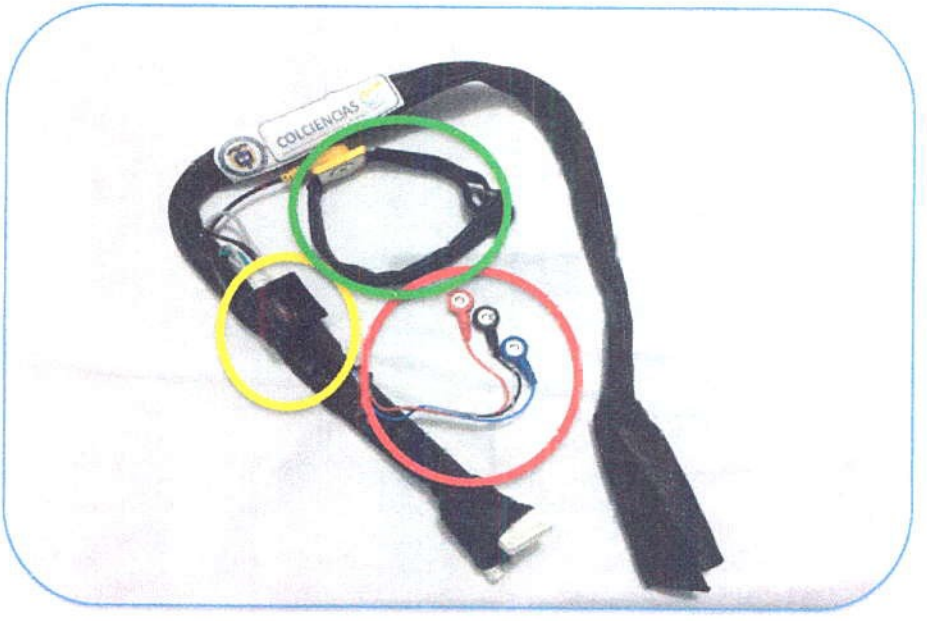


Figura 19. Ubicación Sensor Muscular en correas al módulo de control

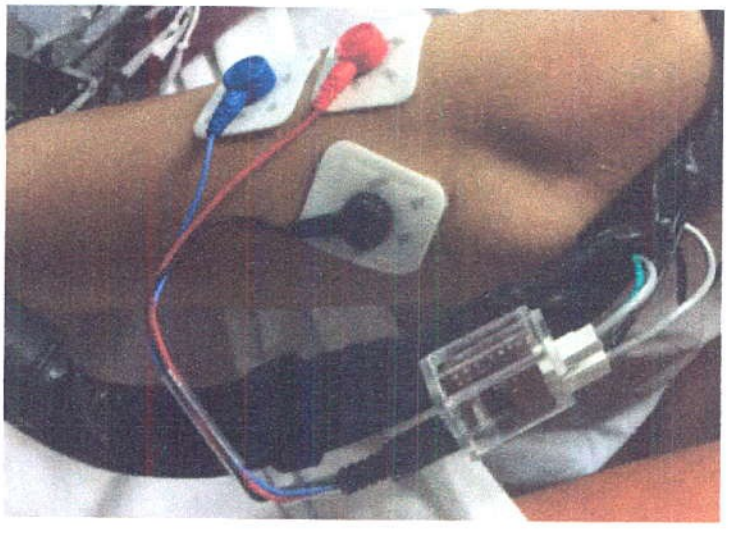


Figura 20. Sensor y Electrodos de EMG en antebrazo

- **INDICADORES LED DE ALTA LUMINOSIDAD**

La interconectividad entre el exoesqueleto mecatrónico y la conexión hacia el módulo de control electrónico, se encuentra a nivel de la muñeca del paciente, se diseño un nodo de interconexión que cuenta con dos Led de alta luminosidad (verde y rojo) en cada lado, los Led indican al paciente en que momento puede decir comandoz de voz para el reconocimiento por el módulo de voz. Cuando se activa el Led verde el paciente puede pronunciar un comando y cuando se activa el Led rojo el módulo de control está procesando la información y no puede atender las indicaciones del usuario.

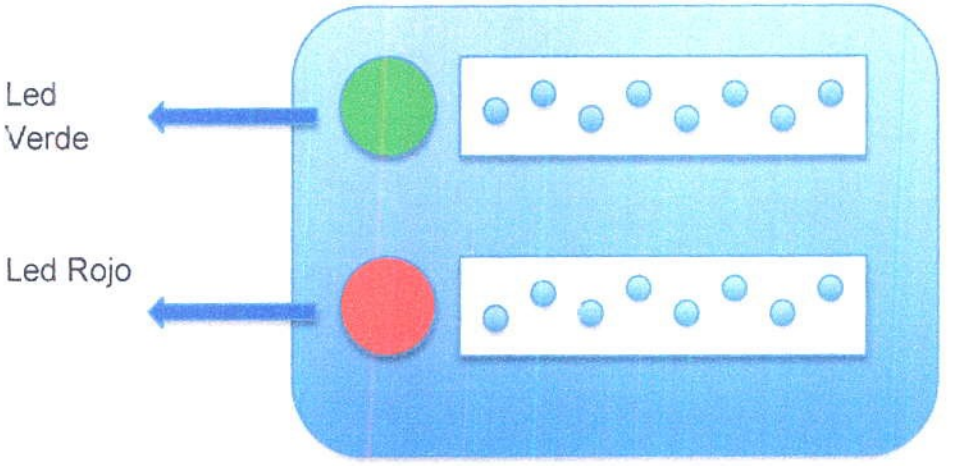


Figura 21. Diseño nodo interconectividad

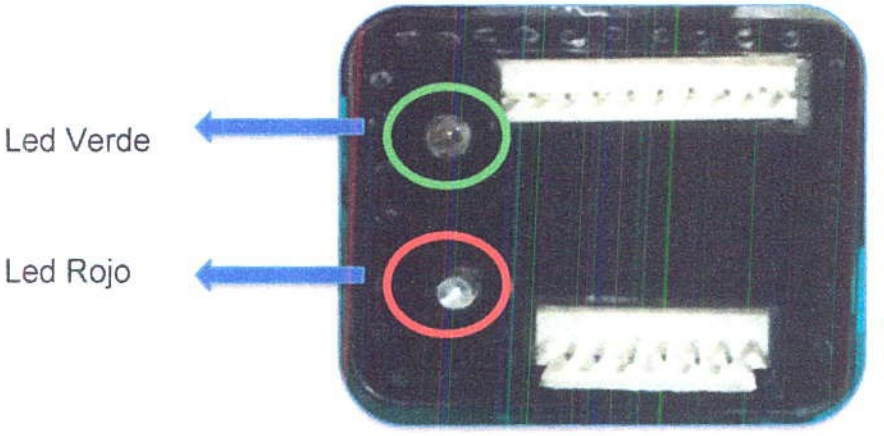


Figura 22. Nodo de interconexión a nivel de muñeca

- **MODULO DE CONTROL ELECTRÓNICO:**

El módulo de control electrónico esta compuesto por un módulo de reconocimiento de voz (EasyVR Shield 3.0), acoplado a una tarjeta electrónica Arduino Mega 2560. La programación del módulo se realiza en Software Arduino, permite integrar los diferentes componentes (Sensores de Electromiografía, Sensores de Fuerza Resistivos, Fuente de Alimentación Conmutada y Servomotores del exoesqueleto).

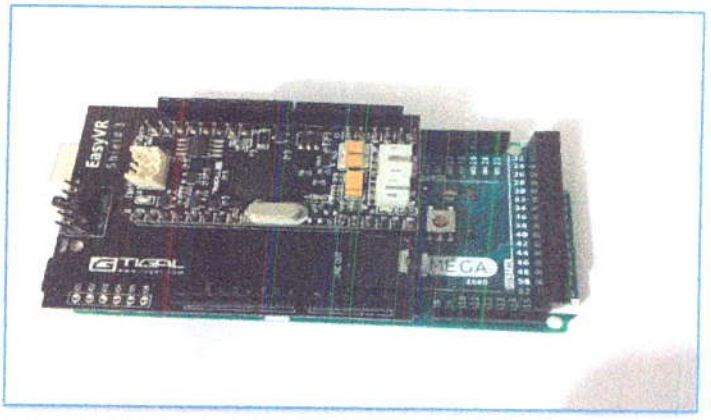


Figura 23. Módulo de control electrónico

Se diseño y fabricó una plataforma de interconectividad entre la tarjeta Arduino Mega 2560 y todos los componentes electrónicos del sistema, esta plataforma se diseña en software ISIS-ARES versión estudiantil y se fabricó en baquelas de fibra de vidrio. La plataforma electrónica contiene dos baquelas de fibra de vidrio, que permiten la interconectividad con todos los elementos del sistema descritos anteriormente.

En el plano inferior se conectan las lineas de alimentación de las fuentes de alimentación conmutada y se crea un divisor de corriente de alta impedancia para proteger la tarjeta Arduino Mega 2560 del Amperaje entregado por las Switching Power Supply. Además se conecta las lineas de tierra del circuito electrónico y la señal de los Sensores de Fuerza Resistivos de la mano izquierda y los sensores de Electromiografía de ambas manos. En el plano superior se conectan las señales de los Sensores de Fuerza Resistivos de la mano derecha y todos los cables de los servomotores de ambas manos.

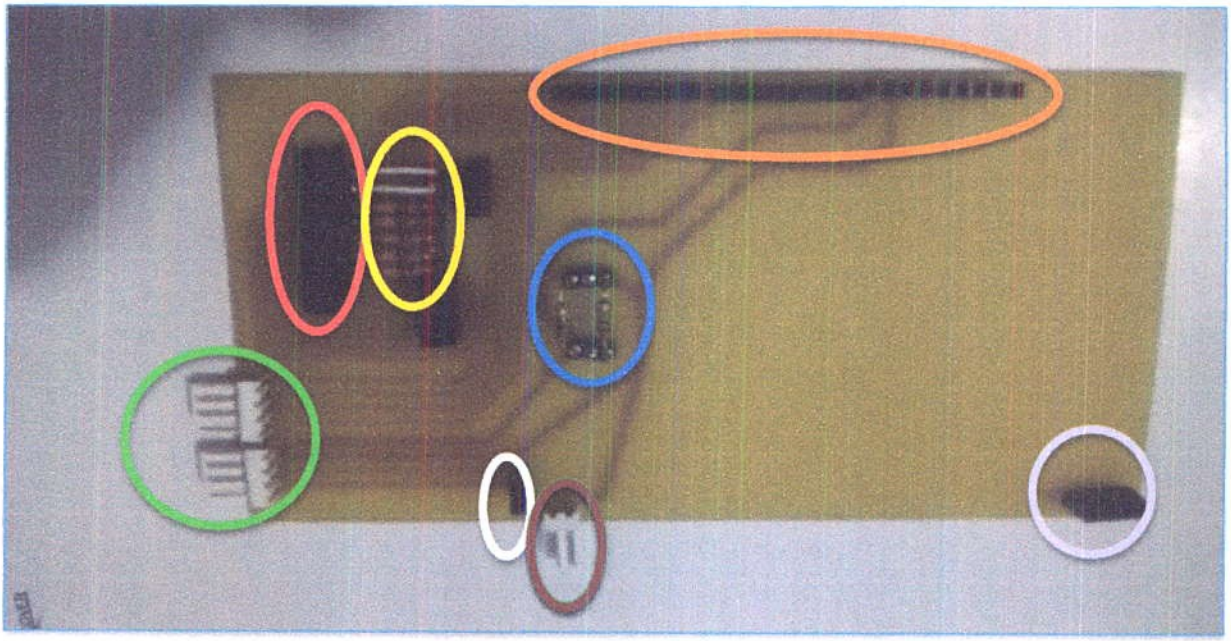


Figura 24. Tarjeta Electrónica de la placa inferior

- La plataforma inferior contiene las conexiiones representadas en la gráfica:

  - **Verde:** Conectores GP de 4 pines. Para la entrada de voltaje positivo y negativo al sistema. Para la salida de los voltajes a los SFR, Para la entrada de la señal de electromiografía.
  - **Roja:** Conectores para la entrada de la señal de los Servomotores de la mano derecha.
  - **Amarilla:** Resistencias para la conexión a tierra de los SFR.
  - **Naranja:** Conexión entre la plataforma inferior y la superior
  - **Azul:** Resistencias de alta impedancia (1 MΩ) para atenuar el amperaje que ingresa a la tarjeta arduino.
  - **Blanco:** Conexión entre la plataforma inferior y la superior para el Voltaje de los SFR.
  - **Café:** Conexión a tierra de la fuente de alimentación y la plataforma.
  - **Lila:** Conexión entre la plataforma inferior y la superior

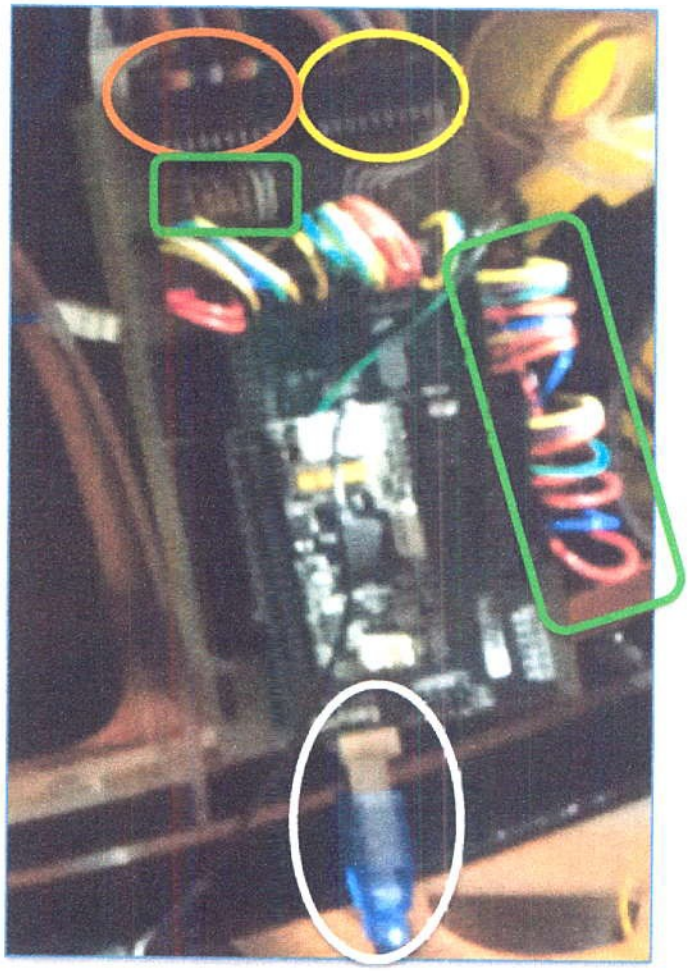


Figura 25. Módulo de control ensamblado en la plataforma superior

- ✓ **Conectores de seguridad:** Conectan los cables de los servomotores de la mano derecha (círculo naranja), y los de la mano izquierda (círculo amarillo).
- ✓ **Resistencias a tierra de los SFR:** Las resistencias de 10KΩ conectan la salida de la señal a la tierra del módulo de control (recuadro verde).
- ✓ **Conectores de circuitos:** Conectan las salidas de la plataforma al módulo de control del dispositivo. Se realiza de esta forma con propósitos educativos (recuadro rojo).
- ✓ **Conector USB:** Cable USB macho hembra conecta el módulo de control con el computador.

2.5 Descripción de la mejor manera de ejecutar la invención

El proyecto: Prototipo exoesqueleto de mano y muñeca con aplicaciones de rehabilitación y asistencia (**Prototipo EMMRA-1**), hace parte de los proyectos financiados por la convocatoria 642 de **COLCIENCIAS** "La locomotora de la innovación para el apoyo del desarrollo tecnológico", está orientado a beneficiar los procesos de asistencia y rehabilitación a la discapacidad motríz en la mano y la muñeca de los pacientes con secuelas por lesiones neurológicas.

El exoesqueleto mecatrónico es un producto original, que implica la aplicación de la ingeniería electrónica, biomédica, mecánica y de sistemas, para generar innovación tecnológica en el área de la salud. Concientes de la necesidad de que el resultado final de la investigación se traduzca en un beneficio para la humanidad, realizamos el presente documento para describir con detalle los procesos de diseño y fabricación del Prototipo EMMRA-1, con el propósito de registrar la patente nacional e internacional en Colombia.

El Prototipo EMMRA-1 está compuesto por las siguientes Interfaces: Órtesis - Piel, Órtesis Pasiva y Órtesis Activa. La Interface Órtesis- Piel está diseñada para ser fácilmente aplicada al paciente discapacitado, fabricada en cuero textil, neopreno, hilo nylon y velcro. La Órtesis Pasiva está diseñada para permitir a cada mano siete Grados de Libertad (**GDL**), con articulaciones elaboradas de acrílico 3mm para los servomotores y ejes con remaches doble cabeza para evitar la fricción. La Órtesis Activa contiene Servomotores Hitec HS-7955TG, proporcionan un torque de 20Kg/cm y un movimiento de 90 grados programables, para evitar sobrepasar los límites anatómicos de las articulaciones de la mano.

El alcance del proyecto "Prototipo EMMRA-1" involucra la descripción del diseño del exoesqueleto mecatrónico (anexo 1), la descripción de la fabricación del Prototipo EMMRA-1 (anexo 2), el diseño del módulo de control del dispositivo (anexo 3), la adaptación de una silla de descanso y equipo de transmisión de movimiento a la silla de descanso (anexo 4) y el código de programación del módulo de control (anexo 5).

3. Reivindicaciones:

Solicitamos la patente del Prototipo Excesqueleto de Mano y Muñeca con Aplicaciones de Rehabilitación y Asistencia "Prototipo EMMRA-1", el cual involucra la Interface Órtesis-Piel, la interface Órtesis Pasiva, y la interface Órtesis Activa.

Los componentes del exoesqueleto mecatrónico integrados al dispositivo para mejorar sus prestaciones y que no pueden ser patentados son (sensores de fuerza resistivos, indicadores led de alta luminosidad, sensores de electromiografía, fuentes de alimentación conmutada con certificación medica, equipo de transmisión de movimiento, servomotores, actuadores lineales, la silla de descanso filadelfia y el módulo de control electrónico con tarjeta EasyVR Shield 3.0 y Arduino Mega 2560)

La programación de los módulos de control del dispositivo se realizó en Software Arduino, EasyVR Commander, Audacity, QuickSintesys 5) y no puede ser patentada.

El código de programación del dispositivo se realiza en Código Fuente Arduino con modificaciones en su contenido más no en su estructura, por lo tanto no puede ser patentado.

Industria y Comercio SUPERINTENDENCIA	TESTIGO DOCUMENTAL	Folio 36

Dependencia Jerárquica: DEL PROPIEDAD INDUSTRIAL	**Código**	2000
Dependencia Productora: GRUPO DE NUEVAS CREACIONES	**Código**	2020
Tipo de Radicación: Entrada: X Salida: ☐ Traslado: ☐ Convenio: ☐		
Radicador:		

DESCRIPCIÓN DE LA PRUEBA

Nombre de la Serie / Subserie:	Código:
Expediente: 15- 278320 - 0	Solicitante / ***Quejoso / Demandante / otros.***
Prueba excluida: CD	
N°. de Pruebas excluidas: 1	
Fecha de emisión de la Prueba:	

UBICACIÓN DE LA PRUEBA EN LA SECCIÓN DE MATERIAL GRÁFICO

N°. de bloque:		Otras especificaciones:
Nº. de rodante:		
Nº. de estante:		
N°. de archivador:		
Nº. de entrepaño:		
Nº. de gaveta:		
N°. de caja:		

IMPORTANTE:

- El material gráfico ***o especial*** que se encuentra ***dentro o anexo a*** las unidades documentales o expedientes, debe extraerse y llevarse a aquella sección del archivo que se haya dispuesto para conservar documentos en distintos formatos.
- Entiéndase por material gráfico todos aquellos dibujos, croquis, mapas, planos, fotografías, ilustraciones, pictografías, códices, prensa, objetos tridimensionales, entre otros.

Observaciones:

El cd contiene
1 archivo mp4 sin carpeta

Andrej

Unidad de Conservación

- ☐ Bolsa
- ☐ Caja
- ☐ Libro
- ☐ Sobre
- ☐ Tomo
- ☐ Otro

Soporte Documental

- X CD
- ☐ Disquete
- ☐ DVD
- ☐ Folleto
- ☐ Periódico
- ☐ Plano
- ☐ Publicación Periódica
- ☐ USB
- ☐ Otro

W

5. Figuras o Ejemplos:

Se anexa el video EMRRA 1 FINAL.mp4, donde se describe con detalle el financiamiento del proyecto por Colciencias, su alcance, las aplicaciones, sus beneficios, las pruebas con un paciente discapacitado y la opinión de médicos especialistas acerca del proyecto.

4. Bibliografía

[1] Jose L. Pons. Wearable Robots: Biomechatronic Exoskeleton. Edited By Wiley & Sons Ltd. ISBN: 978-0-470-51294-4. 2008. Pp 1-15.

[2] H. Kazerooni. "Extender: A case study for human-robot interaction via transfer of power and information signals". 2nd IEEE International Workshop on Robot and Human Communication, November 1993. pp. 10-20.

[3] Perry J. C., Rosen, J. y Burns, S., Upper limb powered exoskeleton design, IEEE/ASME Transactions on Mechatronics, Volumen 12, Issue 4. pp 408-417. Agosto 2007[33] Mozaffari F. M., Troncossi M., Parenti C.V., State of the art of Hand Exoskeleton Systems. Universitá di Bologna. 2011.

[4] Mozaffari F. M., Troncossi M., Parenti C.V., State of the art of Hand Exoskeleton Systems. Universitá di Bologna. 2011.Mozaffari F. M., Troncossi M., Parenti C.V., State of the art of Hand Exoskeleton Systems. Universitá di Bologna. 2011.

[5] M. Bouzit, G. Burdea, G. Popescu and R. Boian. "The Rutgers Master II—New Design Force-Feedback Glove", IEEE/ASME Transactions On Mechatronics, 7(2):256-263, 2002.

[6] P. Stergiopoulos, P. Fuchs and C. Laurgeau. "Design of a 2-Finger Hand Exoskeleton for VR Grasping Simulation", Proceedings of EuroHaptics 2003, July 6-9, 2003. Dublin (Ireland).

[7] Michael Mistry, Peyman Mohajerian, y Stefan Schaal. Arm Movement Experiments with joint space force fields using an exoskeleton robot. 9th International Conference on Rehabilitation Robotics. June 2005. pp 408-413.

[8] M. Mulas, M. Folgheraiter and G. Gini. "An EMG-controlled Exoskeleton for Hand Rehabilitation", Proceedings of the IEEE 9th International Conference on Rehabilitation Robotics, June 28 - July 1, 2005, Chicago, IL(USA), pp. 371-374.

[9] C.D. Takahashi, L. Der-Yeghiaian, V.H. Le, S.C. Cramer. "A Robotic Device for Hand Motor Therapy After Stroke", Proceedings of the IEEE 9th International Conference on Rehabilitation Robotics, June 28 - July 1, 2005, Chicago, IL (USA), pp. 17-20.

[10] H. Kawasaki, S. Ito, Y. Ishigure,Y. Nishimoto, T. Aoki, T. Mouri, H. Sakaeda, and M. Abe. "Development of a Hand Motion Assist Robot for Rehabilitation Therapy by Patient Self-Motion Control", Proceedings of the IEEE 10th International Conference on Rehabilitation Robotics. June 12-15, 2007, Noordwijk, The Netherlands, pp. 234-240.

[11] Rui C.C. Loureiro and W.S. Harwin. "Reach & Grasp Therapy: Design and Control of a 9 DOF Robotic Neuro-rehabilitation System", Proceedings of the IEEE 10th International Conference on Rehabilitation Robotic, June 12.15, 2007, Noordwijk, The Netherlandnds pp. 757-763.

[12] L. Lucas, M. DiCicco, and Y. Matsuoka. "An EMG-Controlled Hand Exoskeleton for Natural Pinching", Journal of Robotics and Mechatronics, Carnegie Mellon University, Pittsburgh. Rb16(5):1-7, 2007.

[13] A. Wege and G. Hommel. "Development and Control of a Hand Exoskeleton for Rehabilitation of Hand Injuries", Proceedings of the 2007 IEEE International Conference on Robotics and Biomimetics. December 2007, Sanya, China.

[14] Y. Hasegawa, Y. Mikami, K. Watanabe and Y. Sankai. "Five-Fingered Assistive Hand with Mechanical Compliance of Human Finger", Proceedings of the IEEE International Conference on Robotics and Automation, May 19-23, 2008, Pasadena, CA (USA).

[15] Y. Fu, P. Wang, S. Wang. "Development of a Multi-DOF Exoskeleton Based Machine for Injured Fingers", IEEE/RSJ International Conference on Intelligent Robots and Systems, September 22-26, 2008, Nice, France.

[16] M. Fontana, A. Dettori, F. Salsedo and M. Bergamasco. "Mechanical design of a novel Hand Exoskeleton for accurate force displaying", Proceedings of the IEEE International Conference on Robotics and Automation, May 12-17, 2009, Kobe, Japan.

[17] A. Chiri, F. Giovacchini, N. Vitiello, E. Cattin, S. Roccella, F. Vecchi, M.C. Carrozza. "HANDEXOS: towards an exoskeleton device for the rehabilitation of the hand", Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 11-15, 2009, St. Louis, USA.

[18] H. Fang, Z. Xie, H. Liu. "An Exoskeleton Master Hand for Controlling DLR/HIT Hand", Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, October 11-15, 2009, St. Louis, USA.

[19] H. Yamaura, K. Matsushita, R. Kato, and H. Yokoi. "Development of Hand Rehabilitation System for Paralysis Patient – Universal Design Using Wire-Driven Mechanism", 31st Annual International Conference of the IEEE EMBS, September 2-6, 2009, Minneapolis, Minnesota (USA).

[20] K. Tadano, M. Akai, K. Kadota, K. Kawashima. " Development of grip amplified glove using bi-articular mechanism with pneumatic artificial rubber muscle", Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), May 3 - 8, 2010, Anchorage, Alaska, USA, pp. 2363.

[21] Mohammad Esmaeili, Nathanael Jarrass, Wayne Dailey, Etienne Burdet and Domenico Campolo. Hyperstaticity for Ergonomic Design of a Wrist Exoskeleton. Proceedings of the IEEE 10th Internaticnal Conference on Rehabilitation Robotic, June 24-26, 2013, Seattl, Washington USA.

Juan David Moreno Arango
INVESTIGADOR PRINCIPAL
Prototipo EMMRA-1

40 37

INFORME FINAL DEL PROYECTO
EXOESQUELETO DE MANO Y MUÑECA CON APLICACIONES DE REHABILITACION Y ASISTENCIA (PROTOTIPO EMMRA-I)

CAPITULO I
DISEÑO DEL EXOESQUELETO MECATRONICO DE MANO Y MUÑECA

INVESTIGADORES:

JUAN DAVID MORENO ARANGO
JULIAN ALBERTO MORENO ARANGO

ASESORES:

VICTOR MAURICIO ACEVEDO CORREA
ALEXANDER RAMOS RODRIGUEZ

COLABORADORES

MARIA ISABEL RIVERA ARANGO
JAIRO MARTINEZ MARIN

Manizales, 01 de Noviembre de 2015

INTRODUCCIÓN

El proyecto: Prototipo exoesqueleto de mano y muñeca con aplicaciones de rehabilitación y asistencia (**Prototipo EMMRA-1**), es un proyectos financiado por la convocatoria 642 de **COLCIENCIAS** "La locomotora de la innovación para el apoyo del desarrollo tecnológico", está orientado a beneficiar los procesos de asistencia y rehabilitación a la discapacidad motríz en la mano y la muñeca de los pacientes con secuelas por lesiones neurológicas.

El exoesqueleto mecatrónico es un producto original, que implica la aplicación de la ingeniería electrónica, biomédica, mecánica y de sistemas, para generar innovación tecnológica en el área de la salud. Concientes de la necesidad de que el resultado final de la investigación se traduzca en un beneficio para la humanidad, realizamos el presente documento para describir con detalle los procesos de diseño y fabricación del Prototipo EMMRA-1, con el propósito de registrar la patente nacional e internacional en Colombia.

El Prototipo EMMRA-1 está compuesto por las siguientes Interfaces: Órtesis - Piel, Órtesis Pasiva y Órtesis Activa. La Interface Órtesis- Piel está diseñada para ser fácilmente aplicada al paciente discapacitado, fabricada en cuero textil, neopreno, hilo nylon y velcro. La Órtesis Pasiva está diseñada para permitir a cada mano siete Grados de Libertad (**GDL**), con articulaciones elaboradas de acrílico 3mm para los servomotores y ejes con remaches doble cabeza para evitar la fricción. La Órtesis Activa contiene Servomotores Hitec HS-7955TG, proporcionan un torque de 20Kg/cm y un movimiento de 90 grados programables, para evitar sobrepasar los límites anatómicos de las articulaciones de la mano.

Cada mano cuenta con cinco Sensores de Fuerza Resistivos (**FSR**) ubicados en las áreas de contacto digital en la palma de la mano, regulan la intensidad de la fuerza aplicada por el exoesqueleto. En cada antebrazo se ubican sensores de señal muscular con electrodos de electromiografía, para el proceso de rehabilitación. Contiene indicadores Led de alta luminosidad de color verde y rojo, para informar al paciente cuando puede dar instrucciones al exoesqueleto por comandos de voz o electromiografía.

El Módulo de control está conformado por un Módulo de Reconocimiento de Voz EasyVR Shield 3.0, integrado a una tarjeta Arduino Mega 2560, que procesa las señales de voz, electromiografía y FSR, para envíar señales de actuación a los servomotores.

TABLA DE CONTENIDO

1. **DISEÑO INTERFACE MANO – ÓRTESIS**..... 04
1.1 Diseño protector cara dorsal de la mano..... 04
1.2 Diseño protector cara ventral de la mano..... 05
1.3 Diseño protector base del I dedo..... 06
1.4 Diseño protector falange distal I dedo..... 07
1.5 Diseño protector falange proximal II dedo..... 08
1.6 Diseño protector falange media II dedo..... 10
1.7 Diseño protector falange proximal III dedo..... 11
1.8 Diseño protector falange distal III dedo..... 12
1.9 Diseño protector falange proximal IV dedo..... 13
1.10 Diseño protector falange medial IV dedo..... 14
1.11 Diseño protector falange proximal V dedo..... 15
1.12 Diseño protector falange medial V dedo..... 16
1.13 Diseño protector lado interno de la muñeca..... 17
1.14 Diseño protector lado externo de la muñeca..... 18
2. **DISEÑO INTERFACE ÓRTESIS PASIVA**..... 19
2.1 Diseño protector servomotor 1..... 19
2.2 Diseño protector servomotor 2..... 21
2.3 Diseño protector servomotor 3..... 23
2.4 Diseño protector servomotor 4..... 25
2.5 Diseño protector servomotor 5..... 26
2.6 Diseño protector servomotor 6..... 29
2.7 Diseño protector servomotor 7..... 32
2.8 Diseño protector servomotor 8..... 33
3. **DISEÑO INTERFACE ÓRTESIS ACTIVA**..... 37
3.1 Diseño movimiento I metacarpiano..... 37
3.2 Diseño movimiento articulación metacarpofalangica I dedo..... 41
3.3 Diseño movimiento articulación interfalangica I dedo..... 43
3.4 Diseño movimiento articulaciónes metacarpofalangicas II y III dedos..... 45
3.5 Diseño movimiento articulación interfalangica proximal II y III dedos..... 47
3.6 Diseño movimiento articulaciónes metacarpofalangicas IV y V dedos..... 49
3.7 Diseño movimiento articulación interfalangica proximal IV y V dedos..... 51
4. **IMPLEMENTACION SENSORES E INDICADORES**..... 53
4.1 Sensores de Fuerza Resistivos..... 53
4.2 Indicadores LED de alta lumiosidad..... 54
4.3 Sensores de Señal Muscular..... 55
5. Fuente de Alimentación MPT 200ª..... 56

1. **INTERFACE MANO – ORTESIS:** La piel del paciente discapacitado, es muy suceptible de lesiones de contacto ante el calor, la humedad y la fricción. Para evitar dermatitis, ulceras de presión, quemaduras por fricción o calor, se diseña un guante elaborado en cuero textil y neopreno, unidos por costura con hilo nylon y velcro.

   El guante de cuero y neopreno, evita que la piel del paciente, entre en contacto con las piezas activas del dispositivo (piezas de acrílico, remaches de aluminio, cables, sensores, electrodos, etc.). Es una pieza fundamental que debe prestar los siguientes requisitos de diseño:

   - Facilmente puesto en la mano de un paciente discapacitado.
   - Proteger del calor generado por los servomotores.
   - Evitar que el exoesqueleto mecánico entre en contacto con la piel.
   - Flexible en los puntos articulares.
   - Adaptarse a la anatomía de la mano y muñeca del paciente.

1.1 PROTECTOR CARA DORSAL DE LA MANO (CARPO Y METACARPO): Elaborado en neopreno, obtenido de una ferula muslera (producto con registro INVIMA para comercializar en nuestro pais y aprobado por la FDA como materia prima para ortesis).

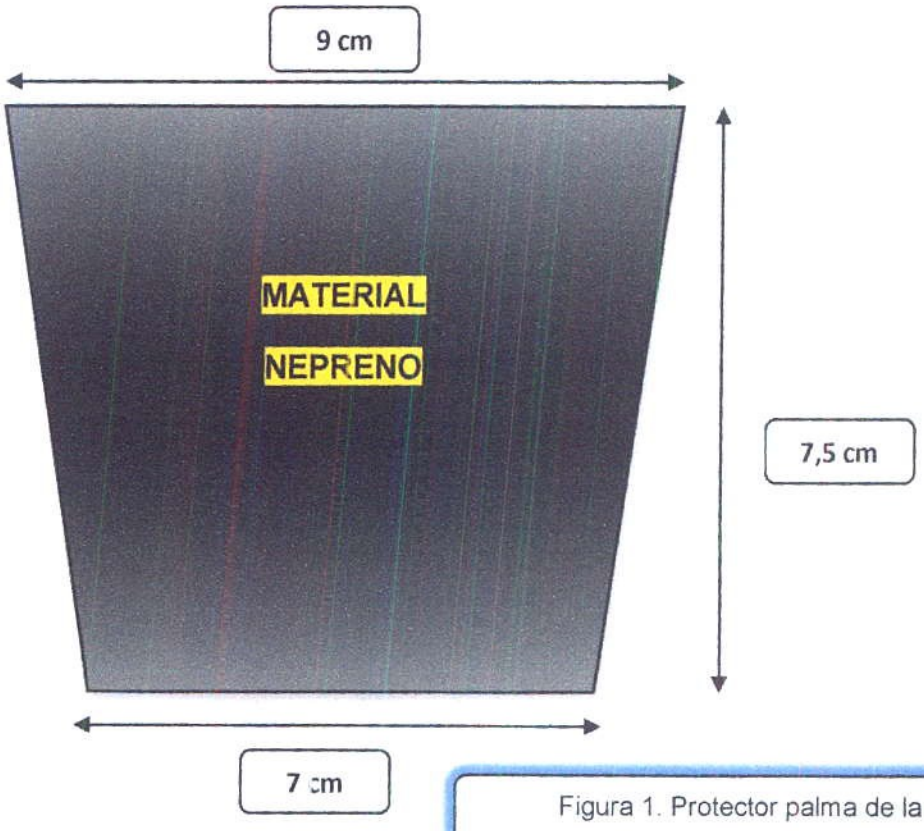


Figura 1. Protector palma de la mano

1.2 PROTECTOR CARA VENTRAL DE LA MANO (CARPO Y METACARPO):

Elaborado en cuero textil y correas de velcro para el ajuste.

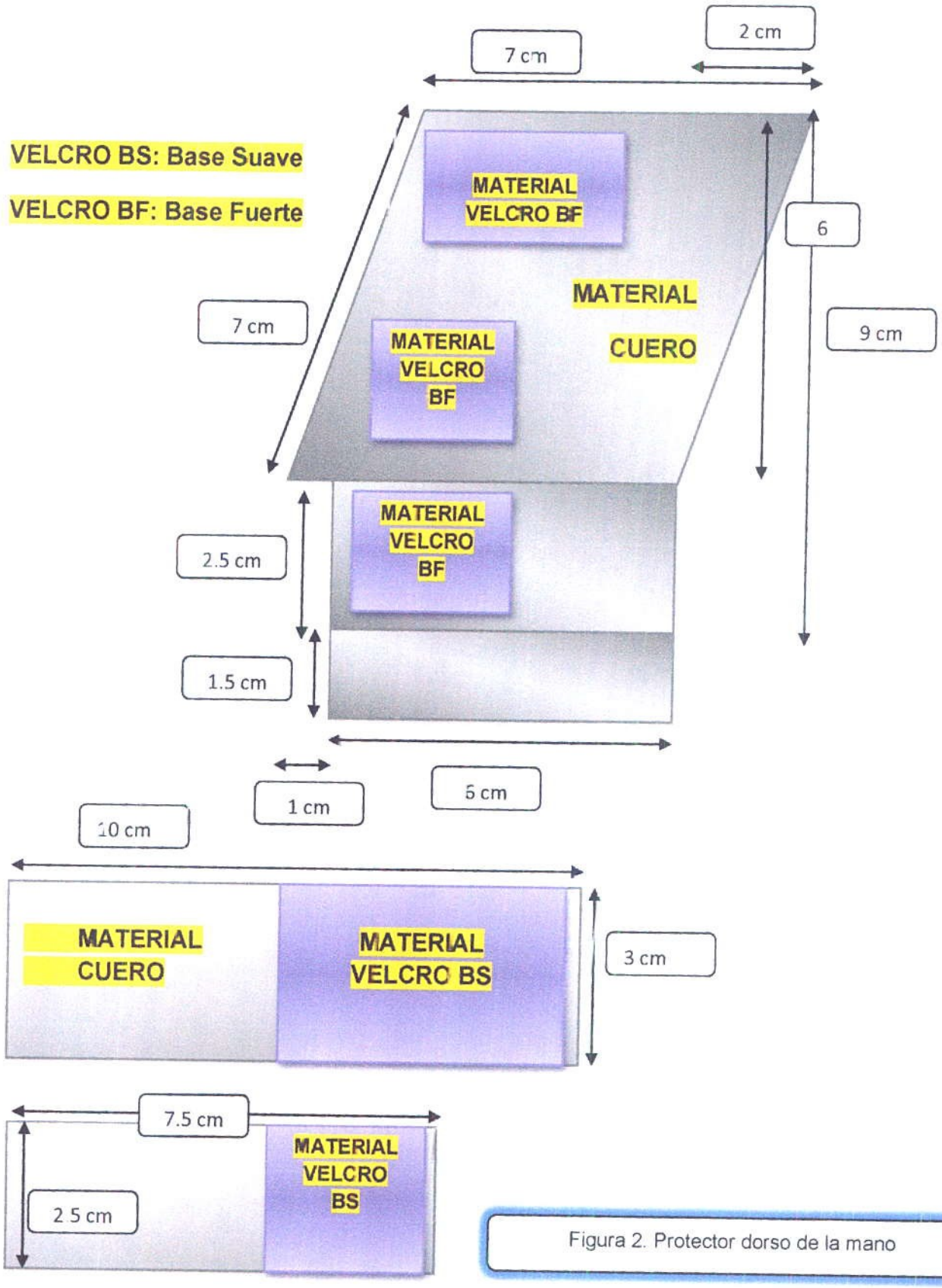


Figura 2. Protector dorso de la mano

1.3 PROTECTOR BASE PRIMER DEDO (METACARPO Y FALANGE PROXIMAL)

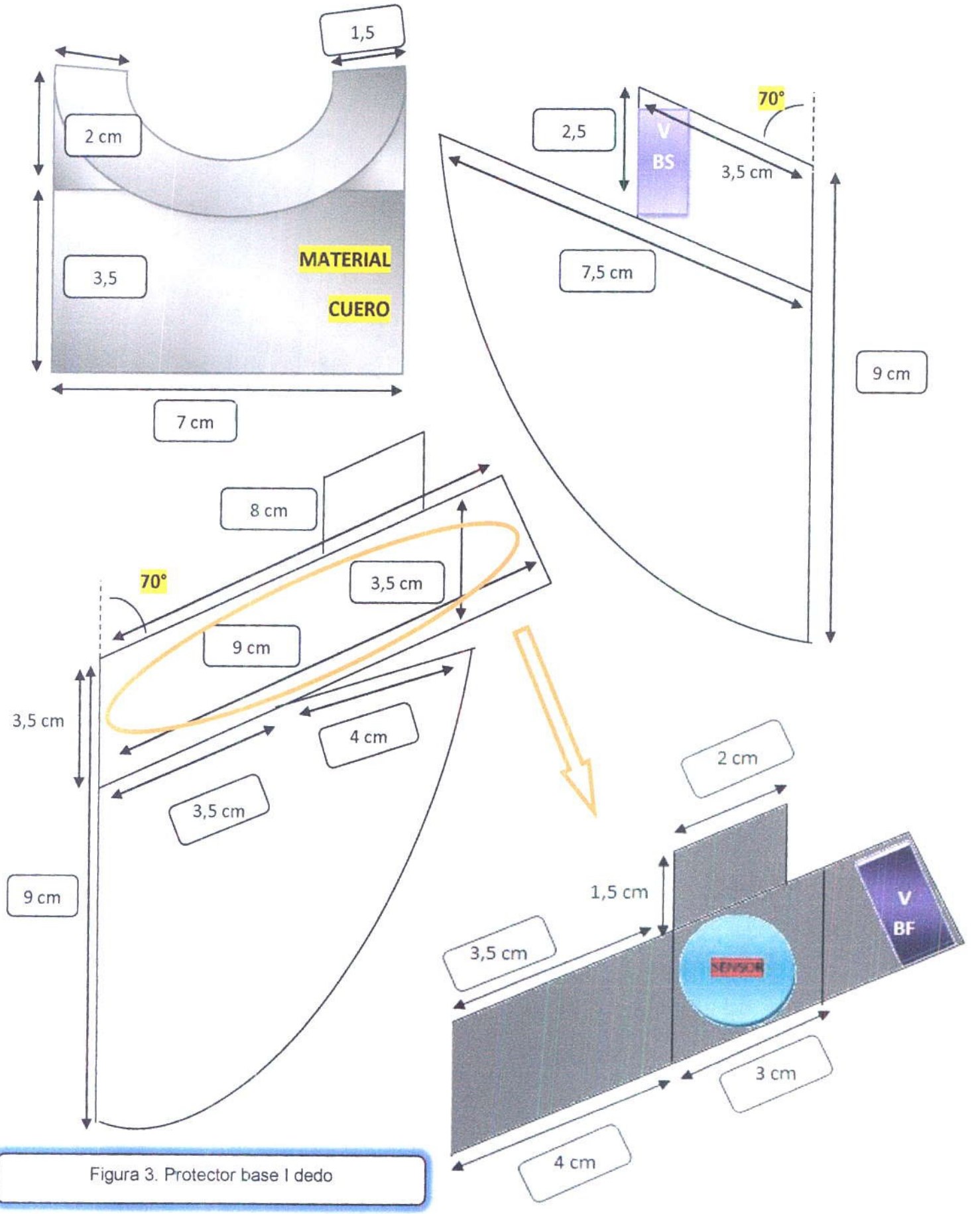
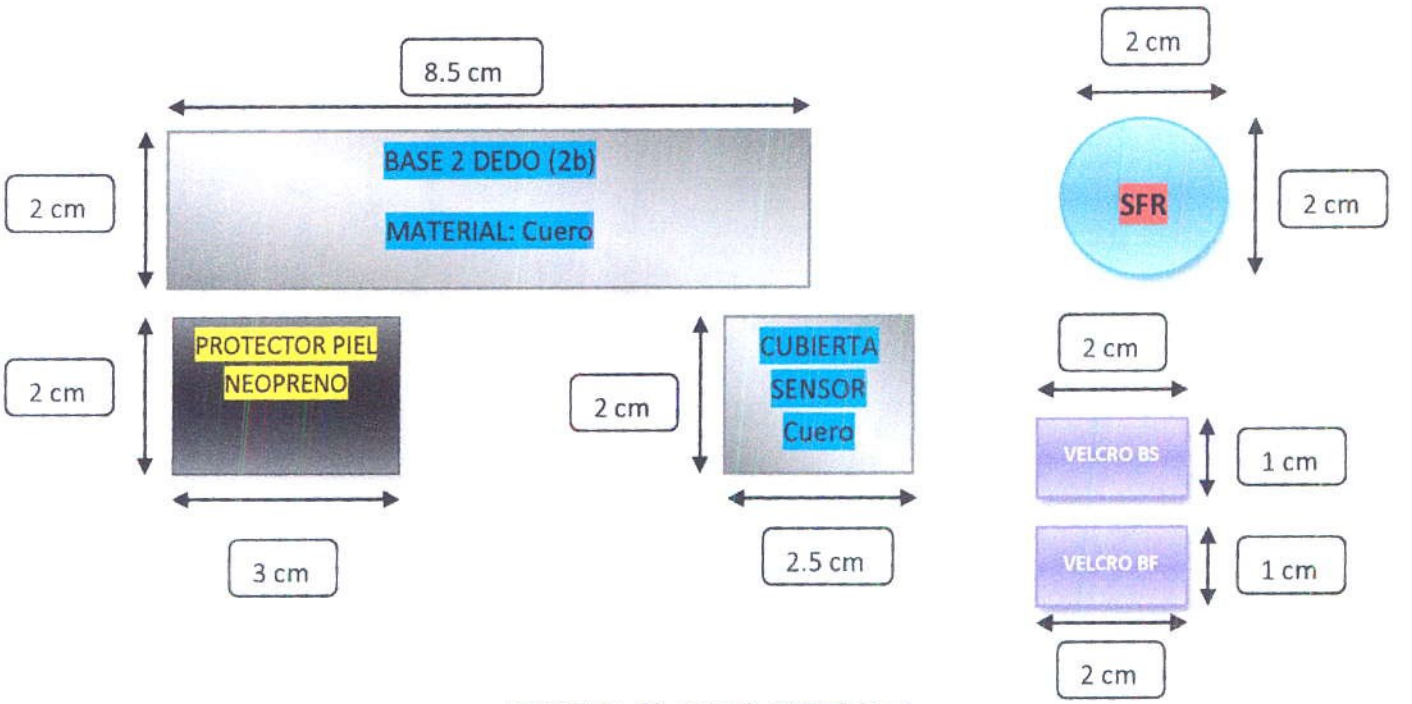
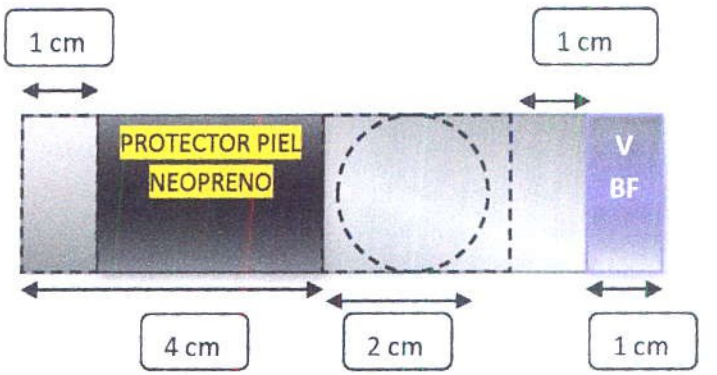


Figura 3. Protector base I dedo

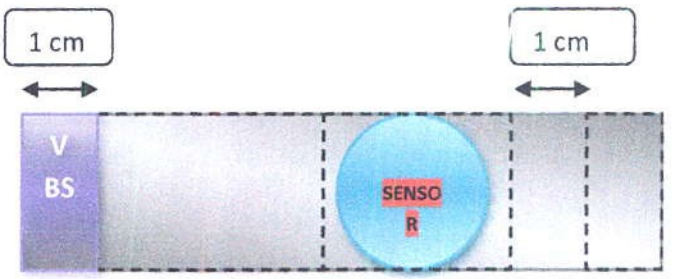
1.4 PROTECTOR BASE PRIMER DEDO (FALANGE DISTAL)

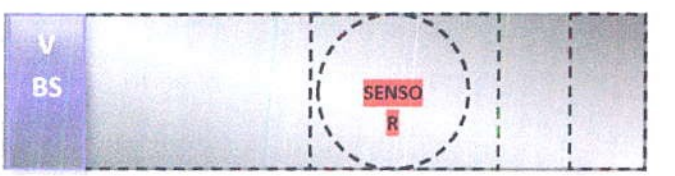


MONTAJE CARA INTERNA



MONTAJE CARA EXTERNA





SENSOR SFR: Sensor de Fuerza Resistivo

Figura 4. Protector base I dedo FD

1.5 PROTECTOR BASE SEGUNDO DEDO (FALANGE PROXIMAL)

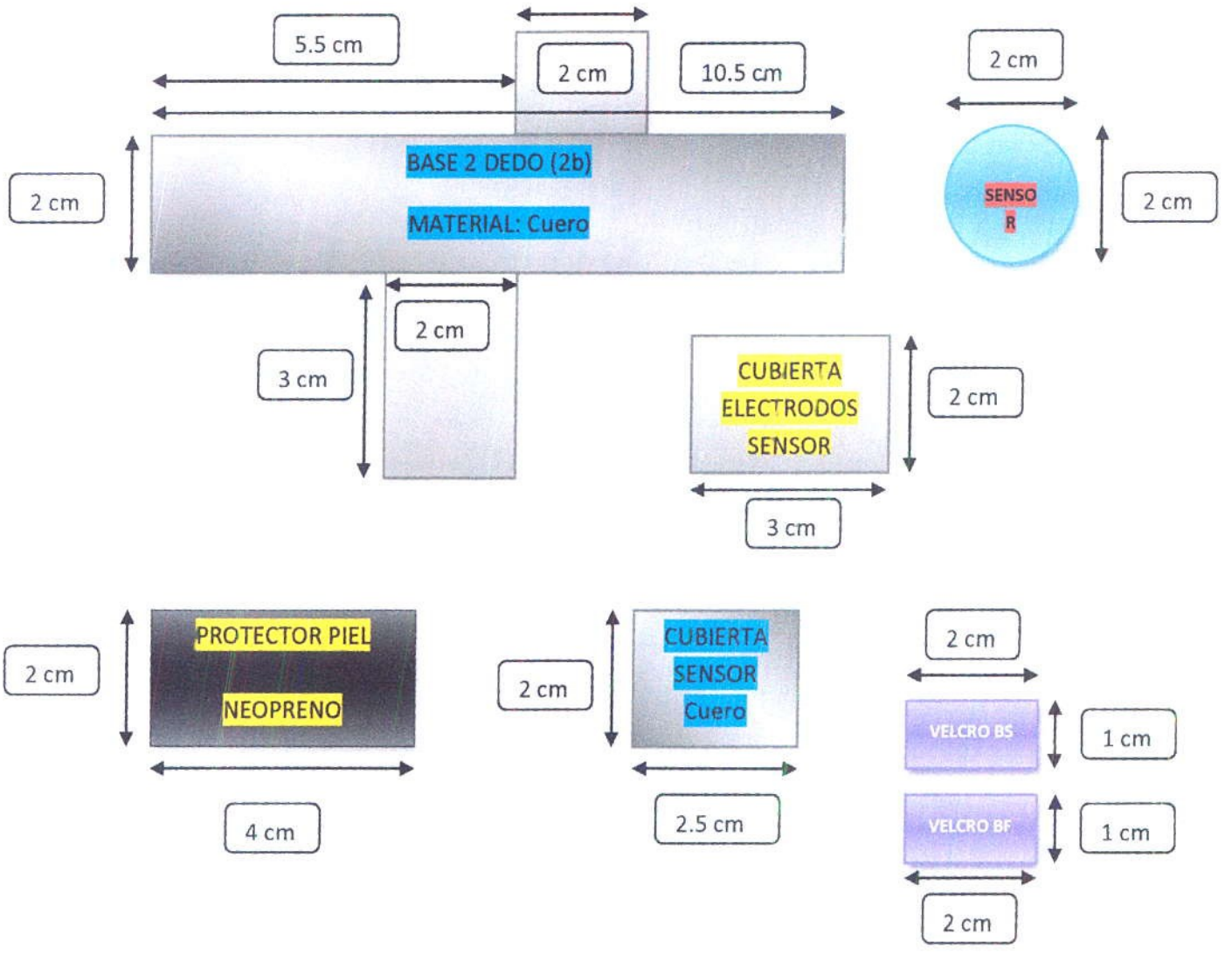
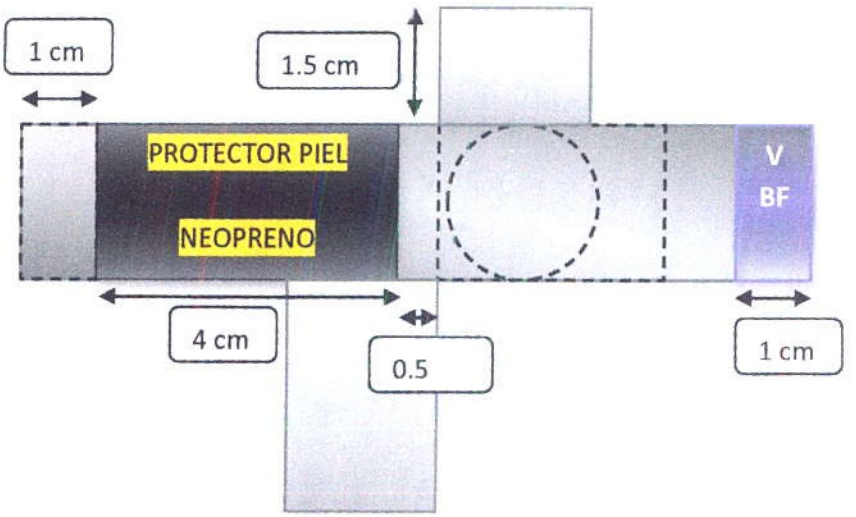
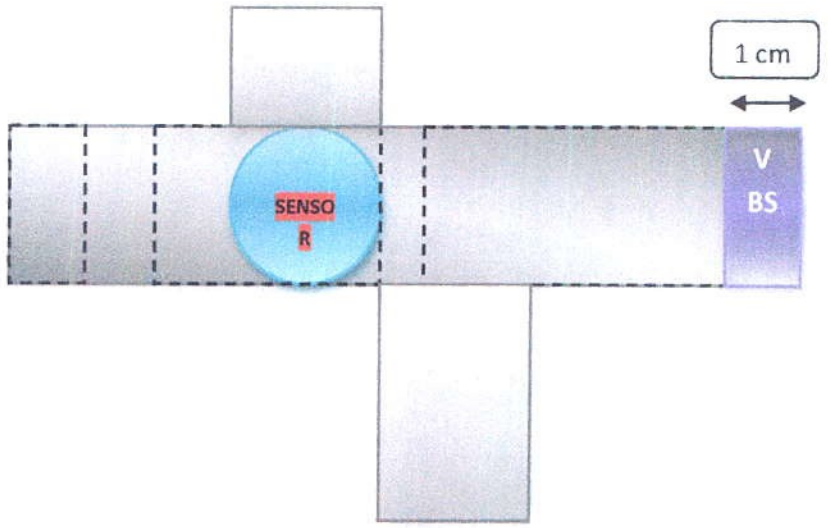


Figura 5. Protector base II dedo FP

MONTAJE CARA INTERNA



MONTAJE CARA EXTERNA



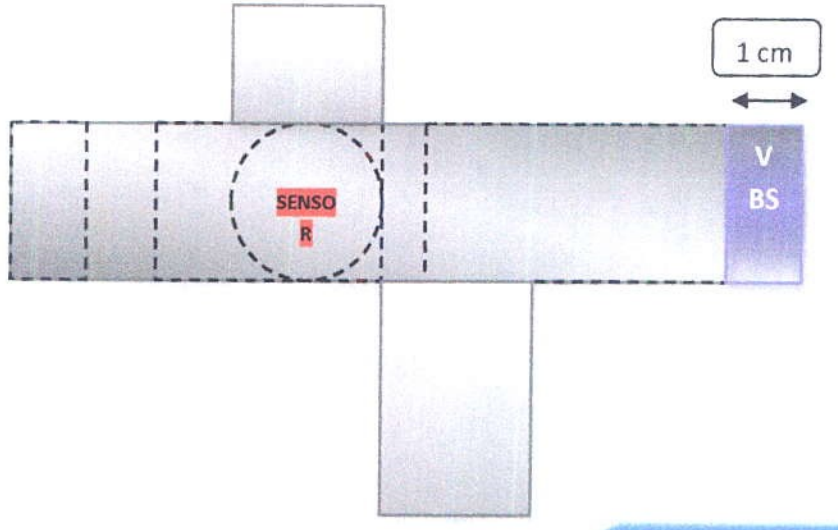
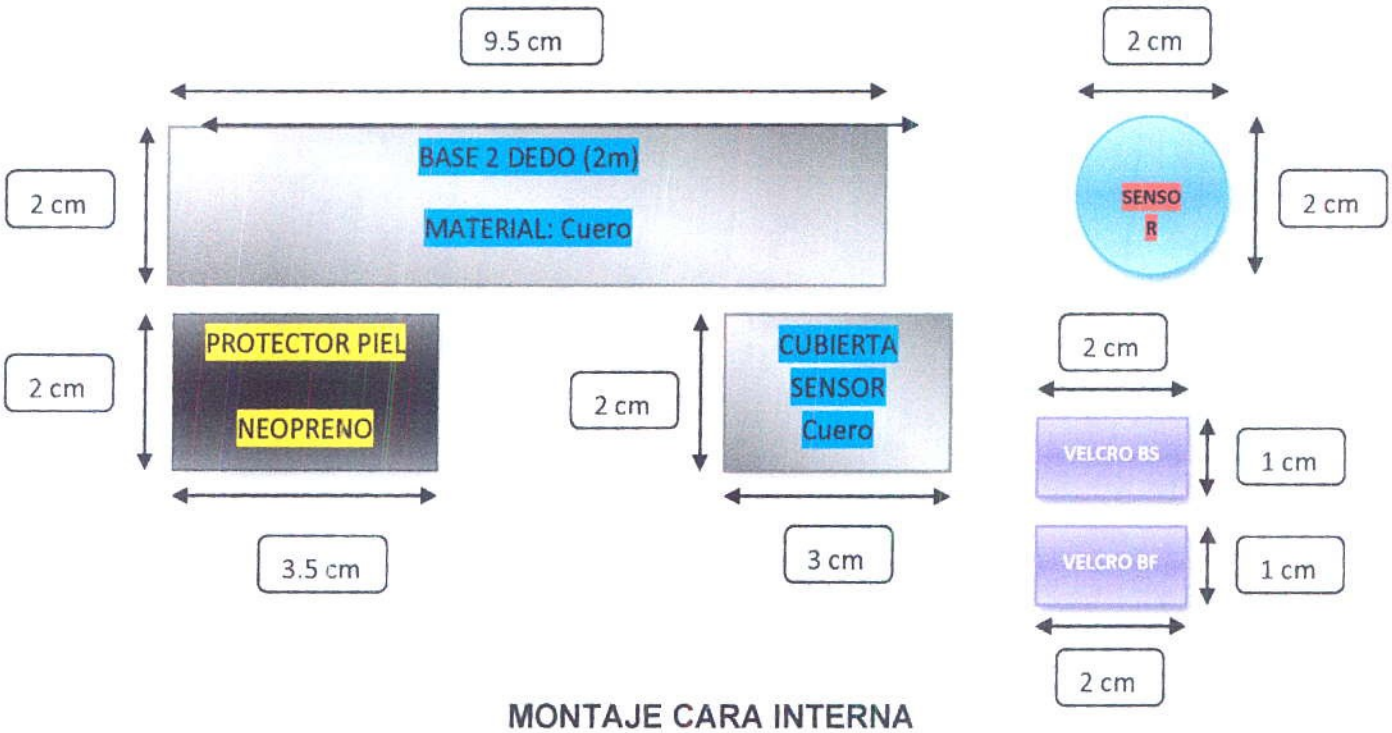
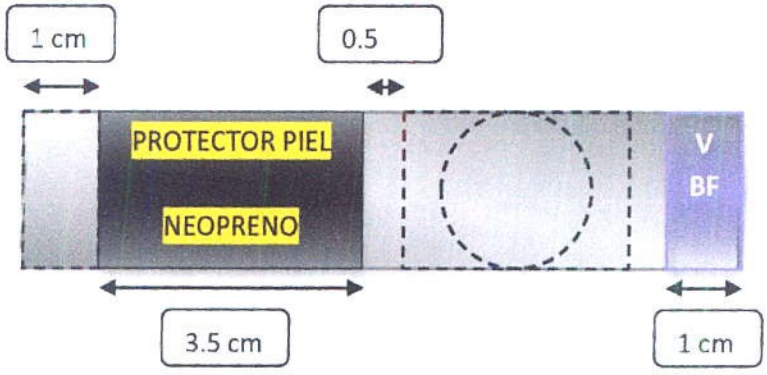


Figura 6. Protector base II dedo FP

1.6 PROTECTOR BASE SEGUNDO DEDO (FALANGE MEDIA)



MONTAJE CARA INTERNA



MONTAJE CARA EXTERNA

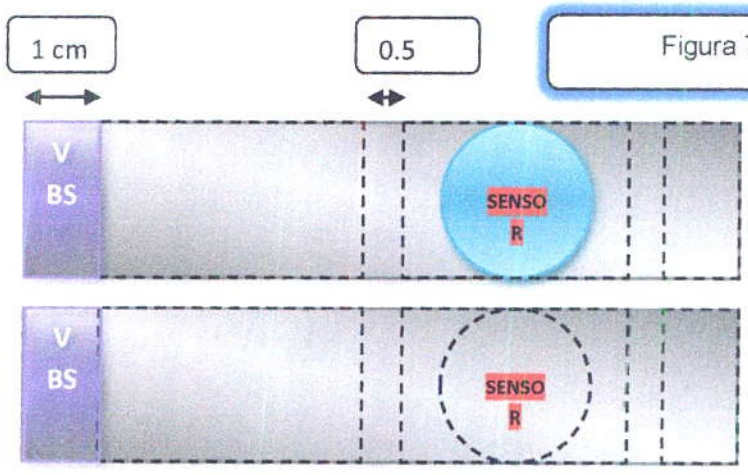
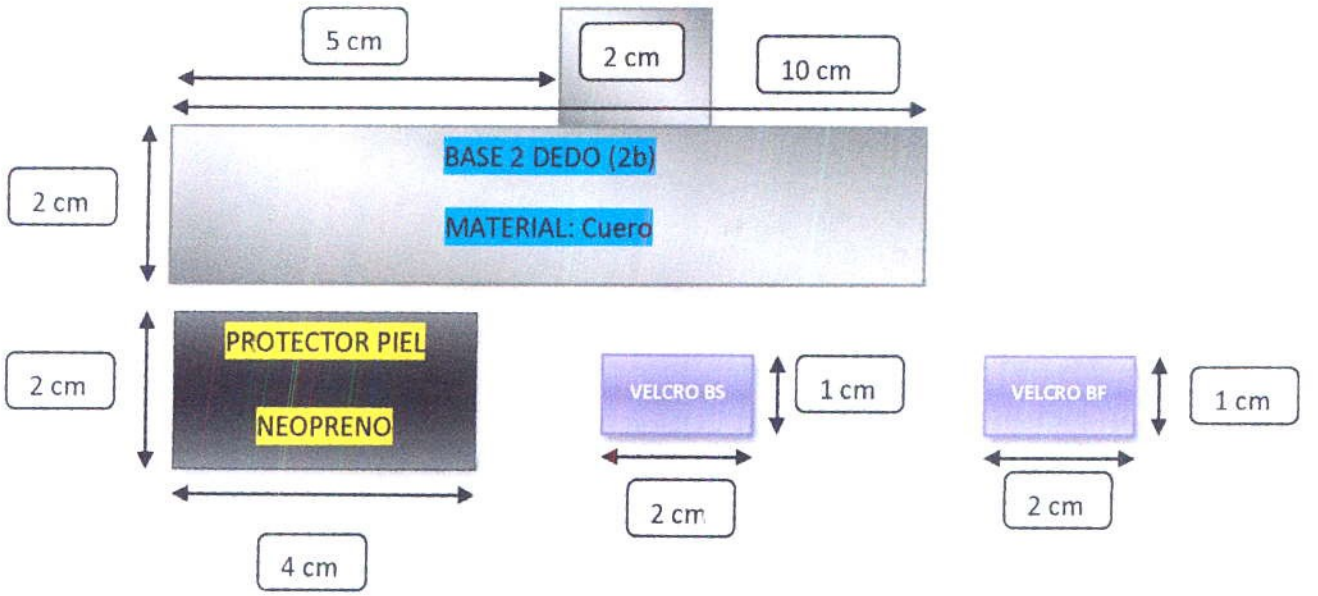
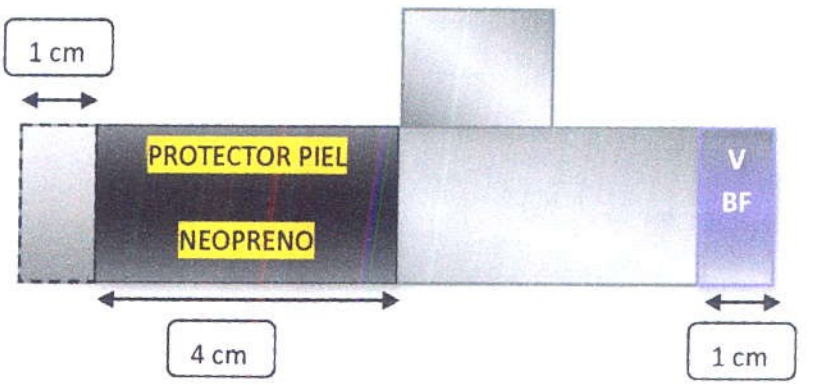


Figura 7. Protector base II dedo FM

1.7 PROTECTOR BASE TERCER DEDO (FALANGE PROXIMAL)



MONTAJE CARA INTERNA



MONTAJE CARA EXTERNA

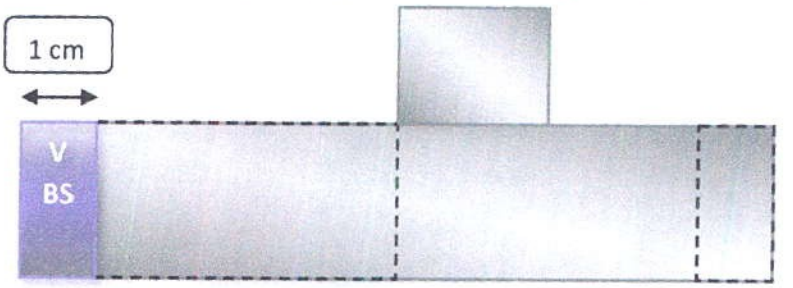
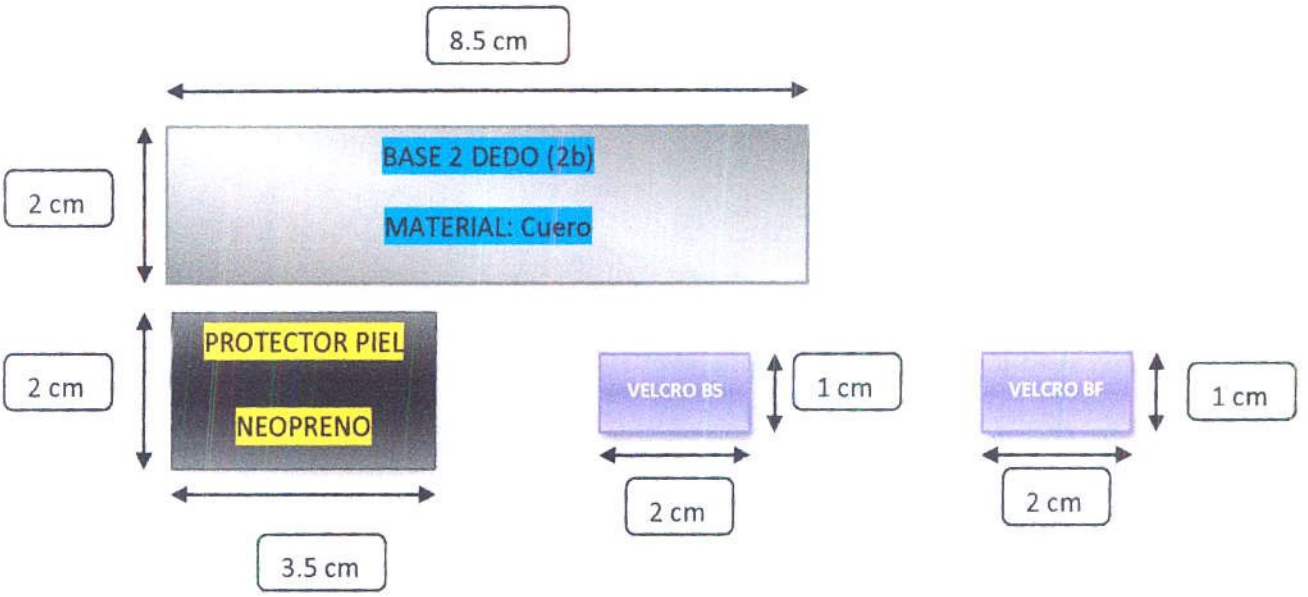
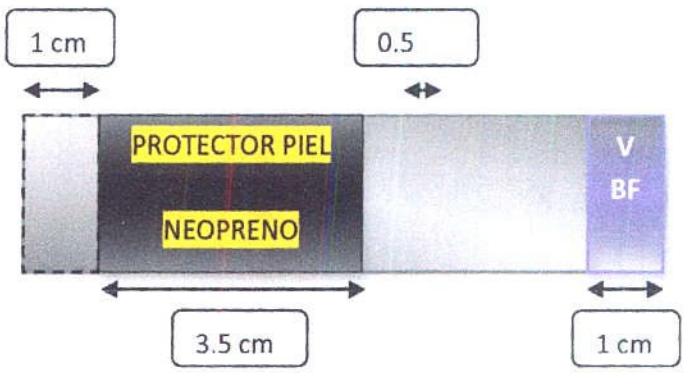


Figura 8. Protector base III dedo FP

1.8 PROTECTOR BASE TERCER DEDO (FALANGE MEDIA)



MONTAJE CARA INTERNA



MONTAJE CARA EXTERNA

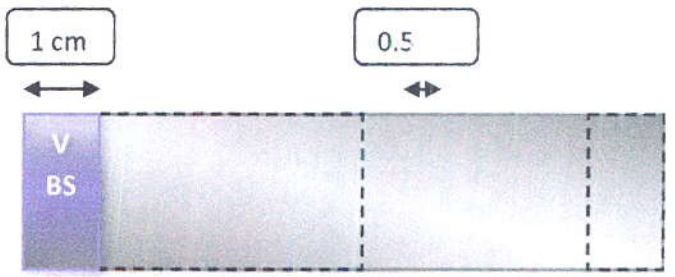
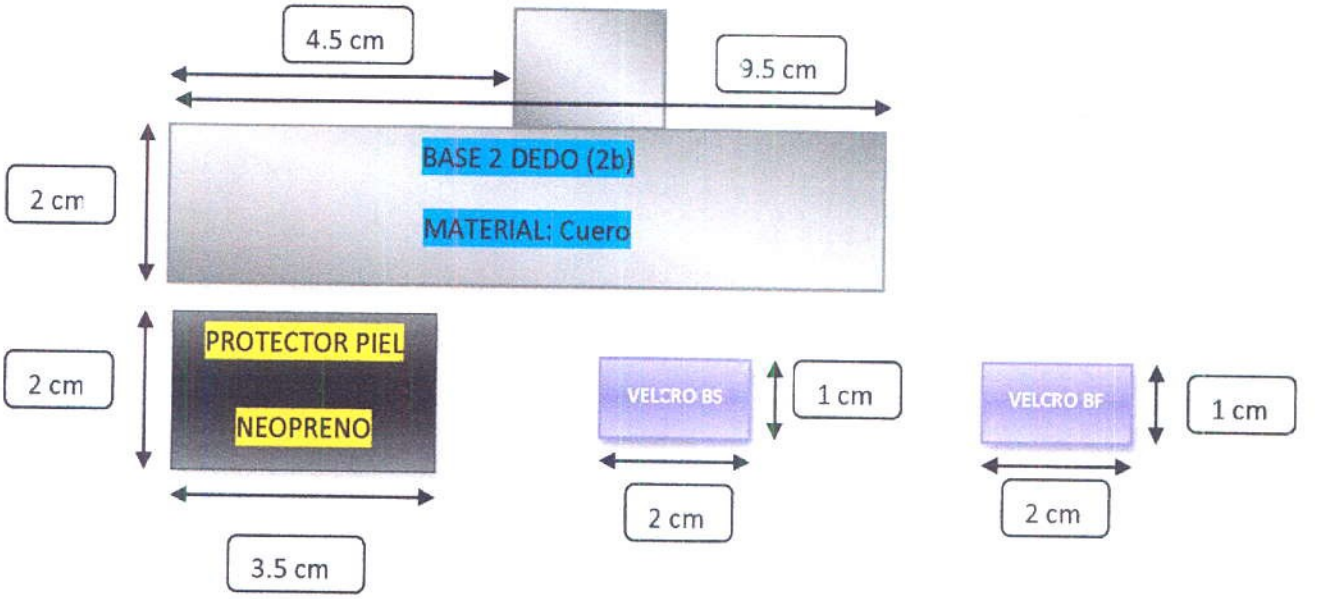
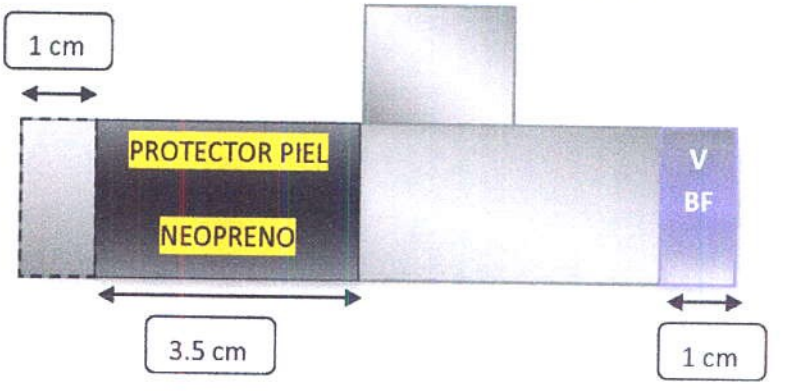


Figura 9. Protector base III dedo FM

1.9 PROTECTOR BASE CUARTO DEDO (FALANGE PROXIMAL)



MONTAJE CARA INTERNA



MONTAJE CARA EXTERNA

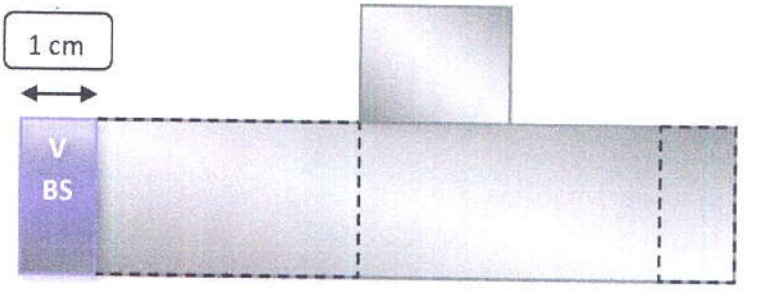
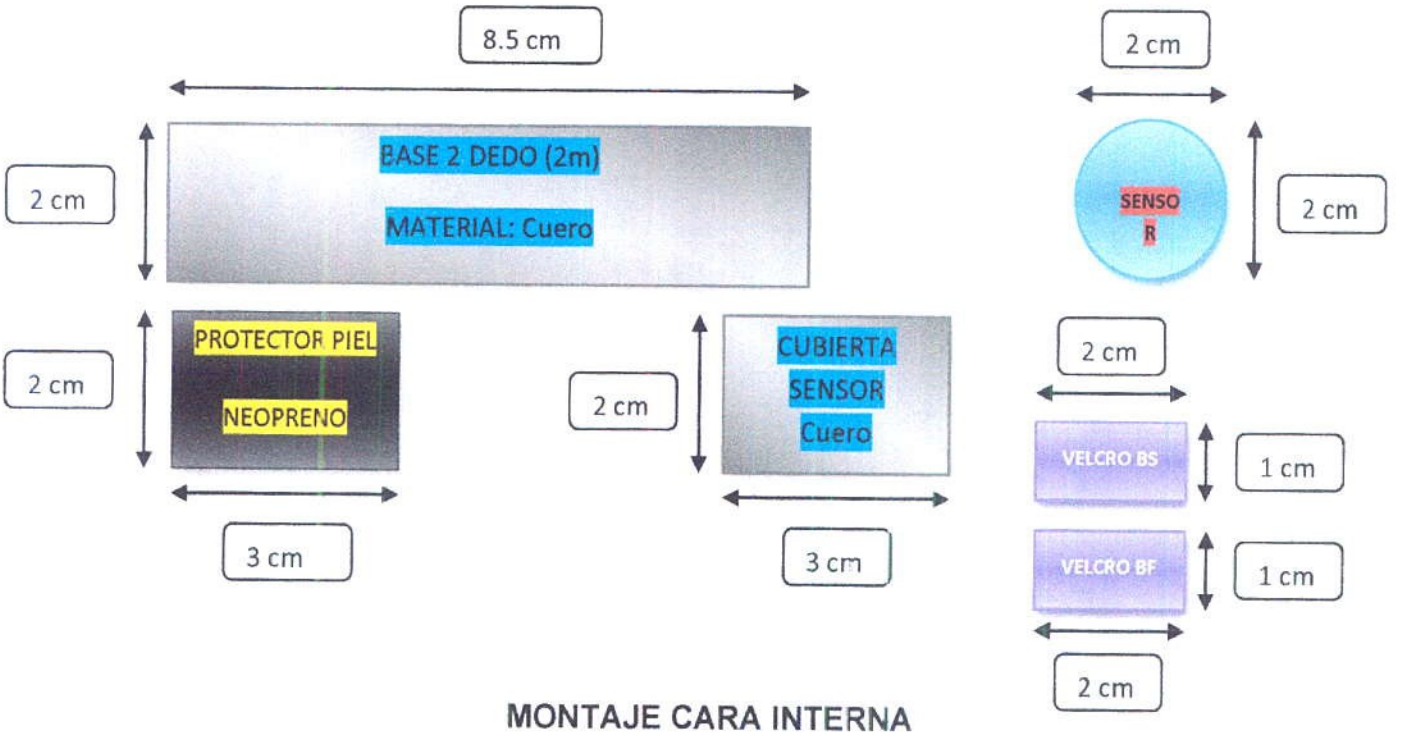
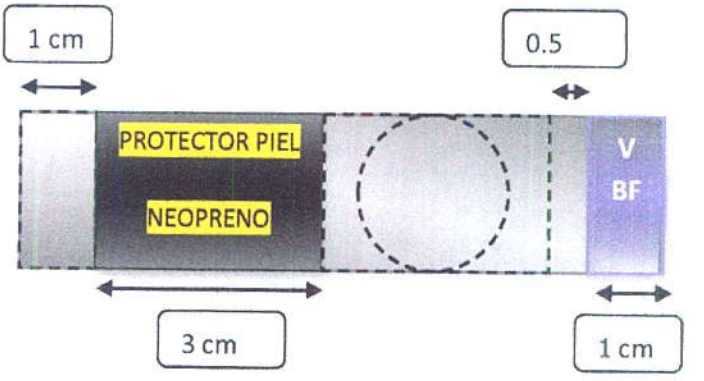


Figura 10. Protector base IV dedo FP

1.10 PROTECTOR BASE CUARTO DEDO (FALANGE MEDIA)



MONTAJE CARA INTERNA



MONTAJE CARA EXTERNA

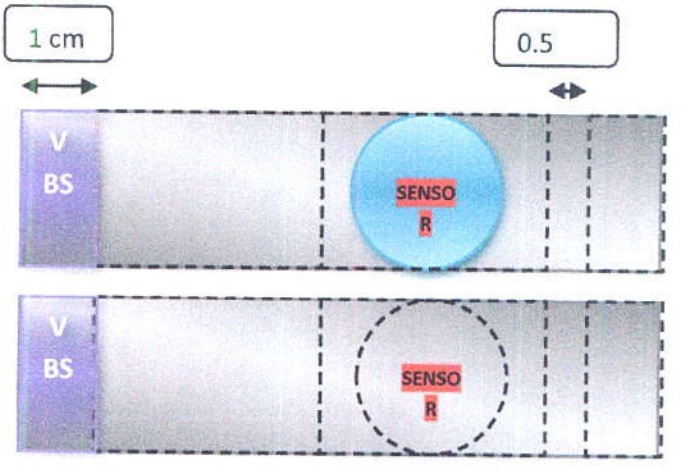
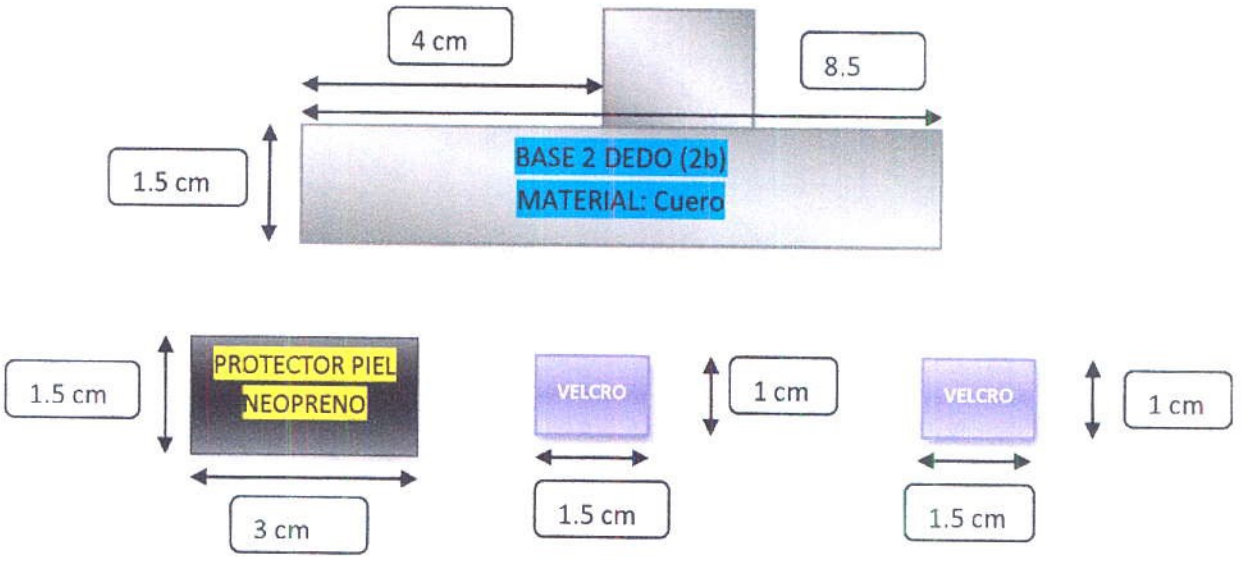


Figura 11. Protector base IV dedo FM

1.11 PROTECTOR BASE QUINTO DEDO (FALANGE PROXIMAL)



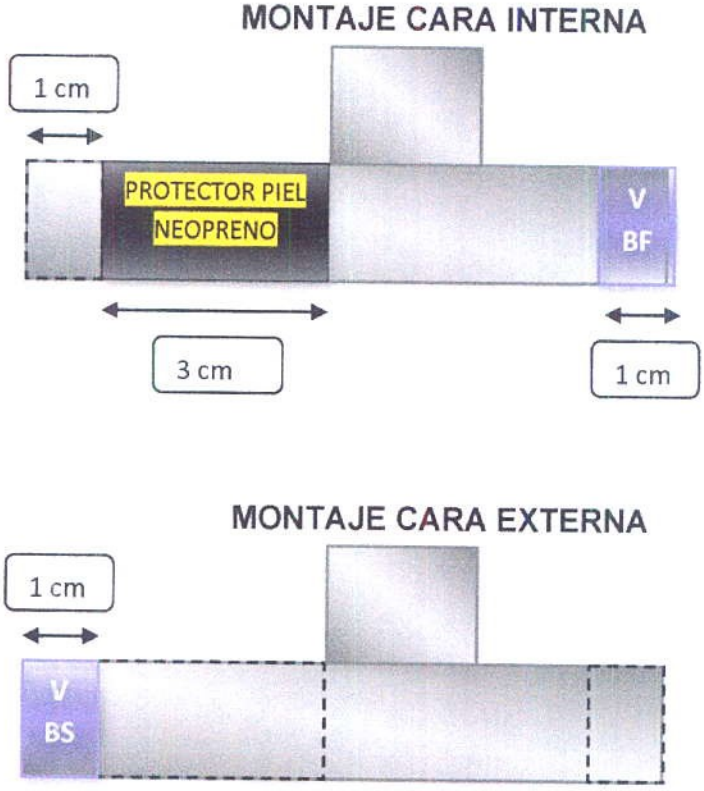
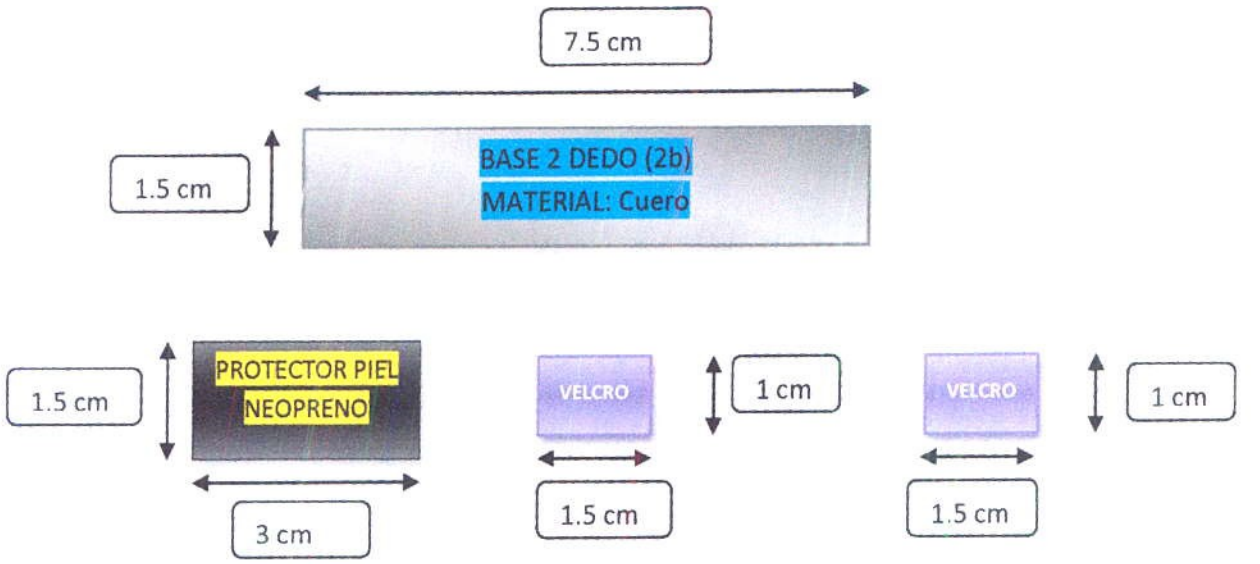
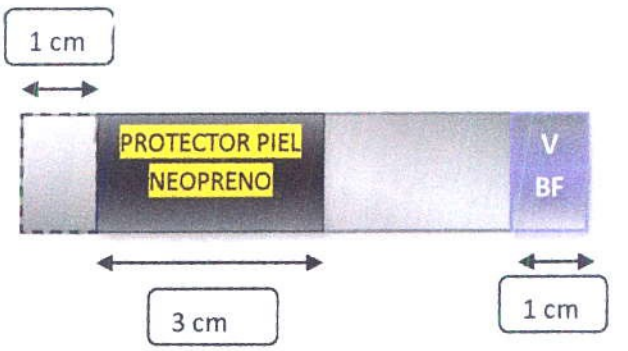


Figura 12. Protector base V dedo FP

1.12 PROTECTOR BASE QUINTO DEDO (FALANGE MEDIA)



MONTAJE CARA INTERNA



MONTAJE CARA EXTERNA

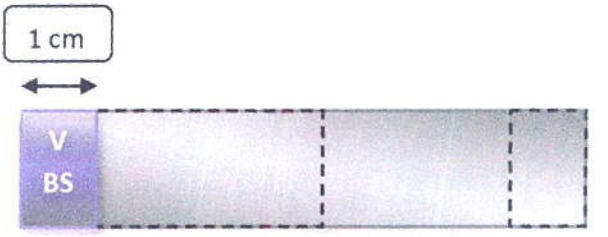
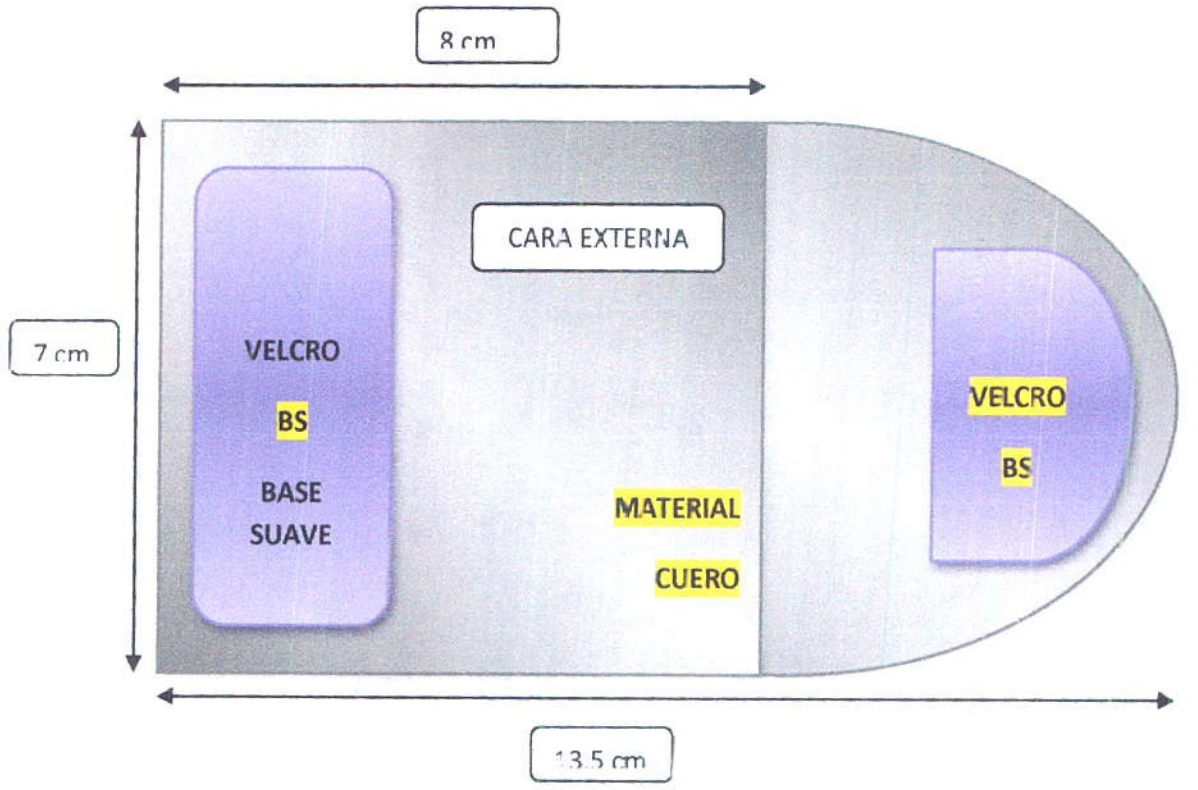


Figura 12. Protector base V dedo FM

1.13 PROTECTOR DE MUÑECA LADO INTERNO



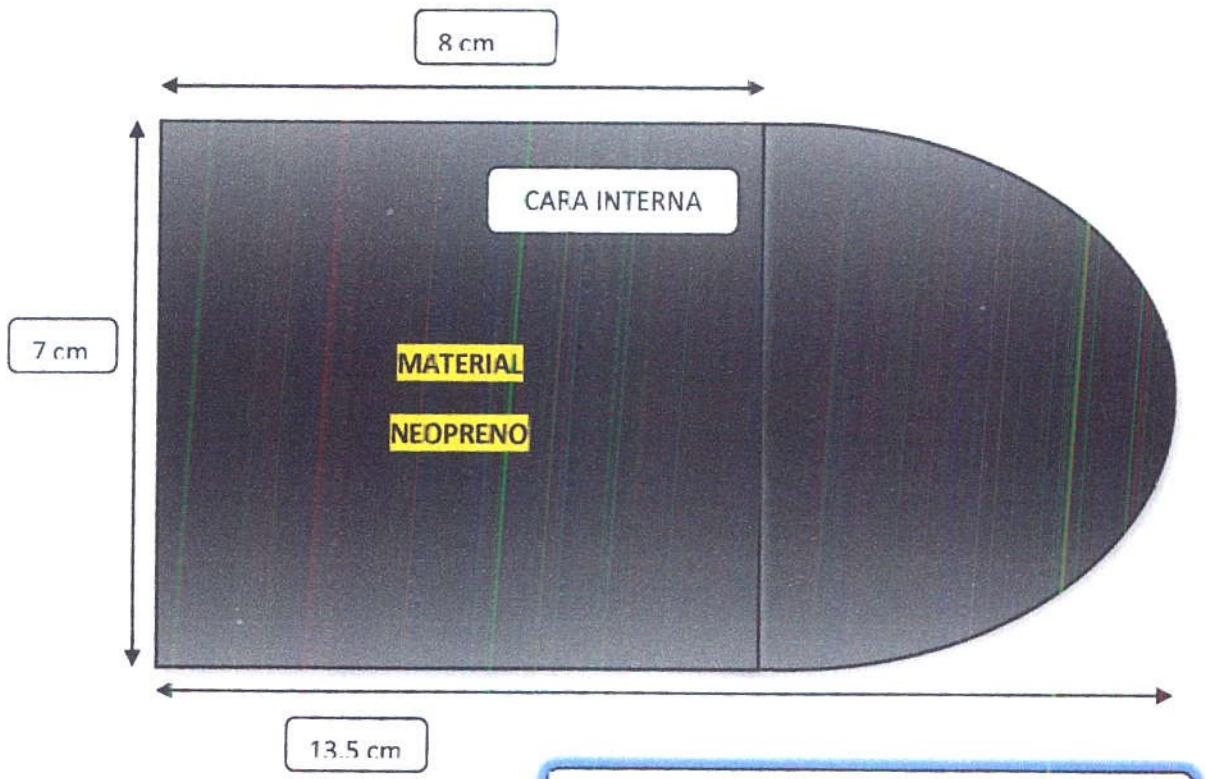
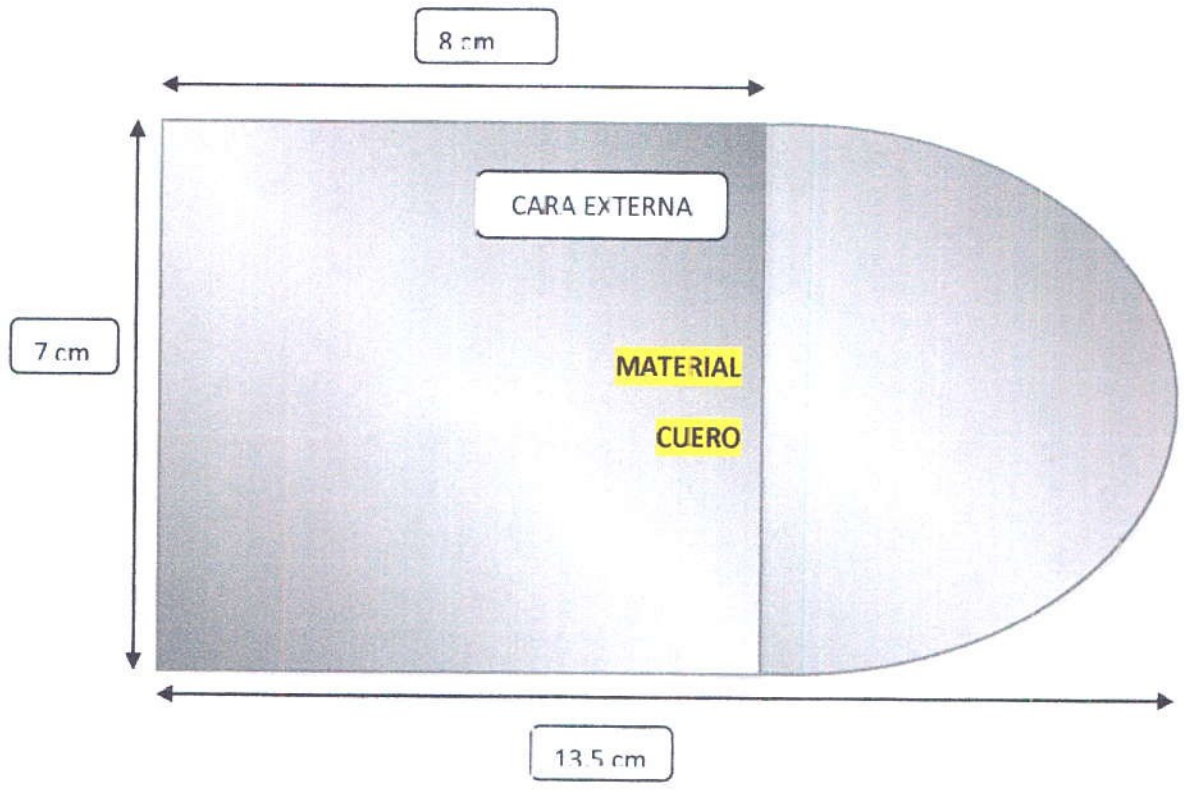


Figura 13. Protector de muñeca lado interno

1.14 PROTECTOR DE MUÑECA LADO EXTERNO



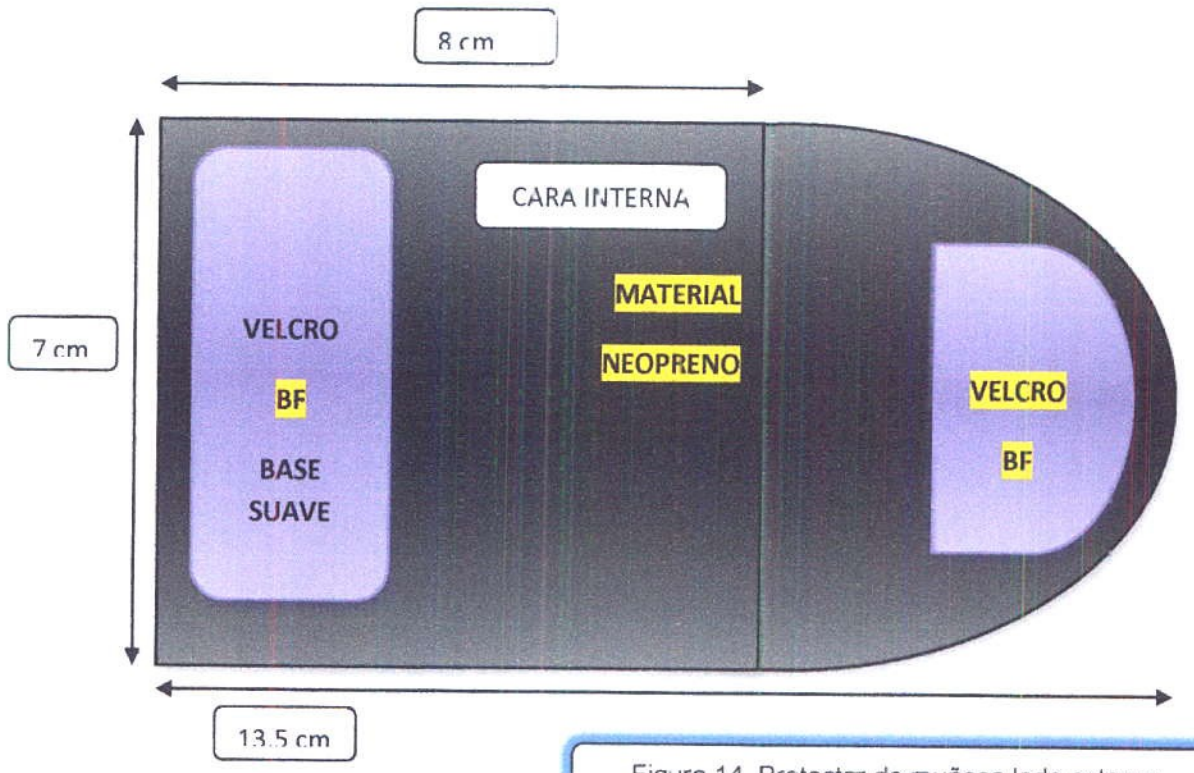
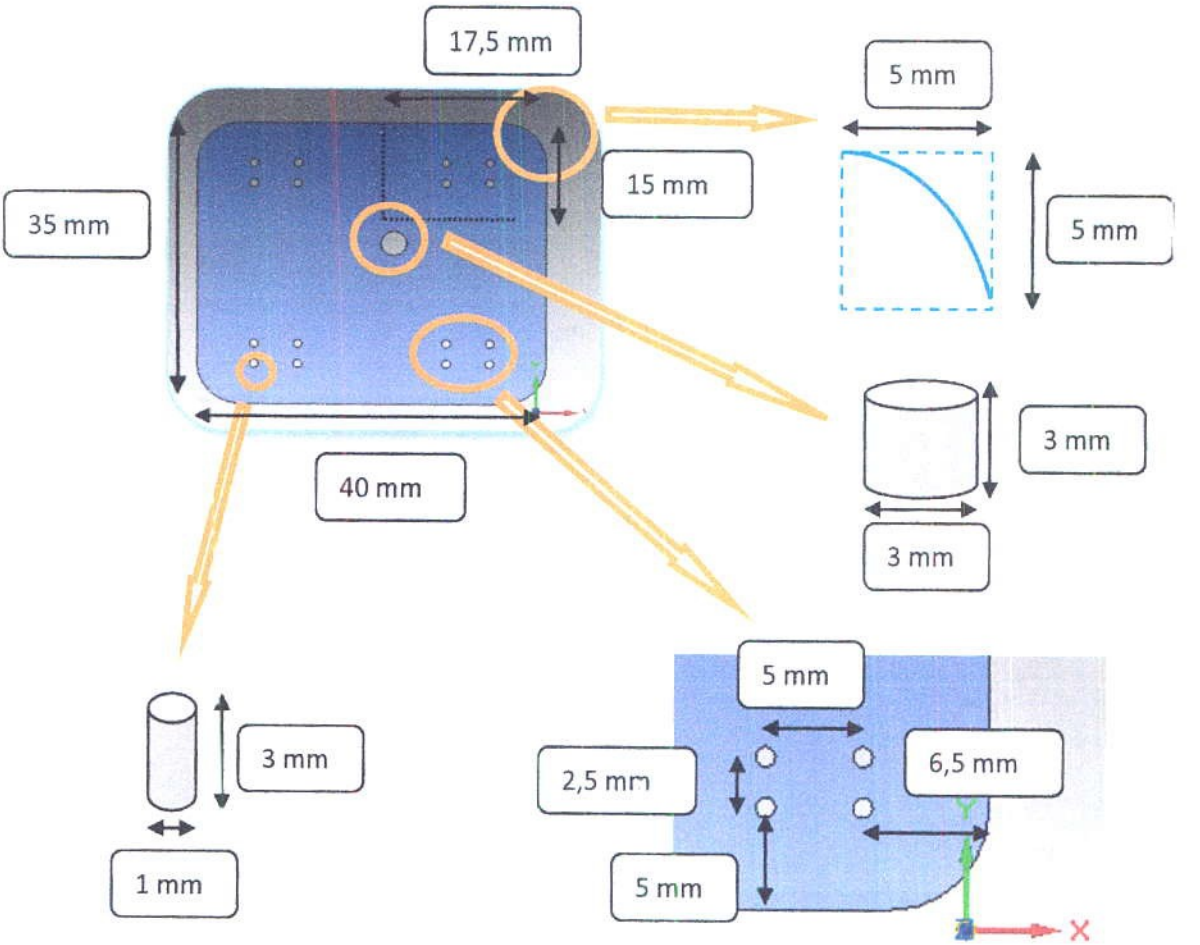


Figura 14. Protector de muñeca lado externo

2. ÓRTESIS PASIVA

2.1 Protector Servomotor 1

(BASE SERVO 1 _ MUÑECA (BS1_MN))



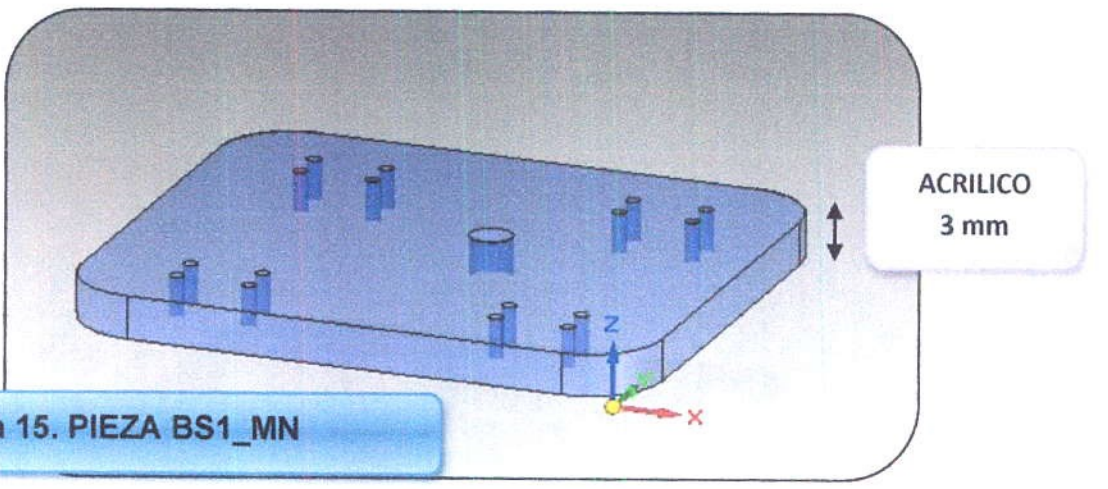
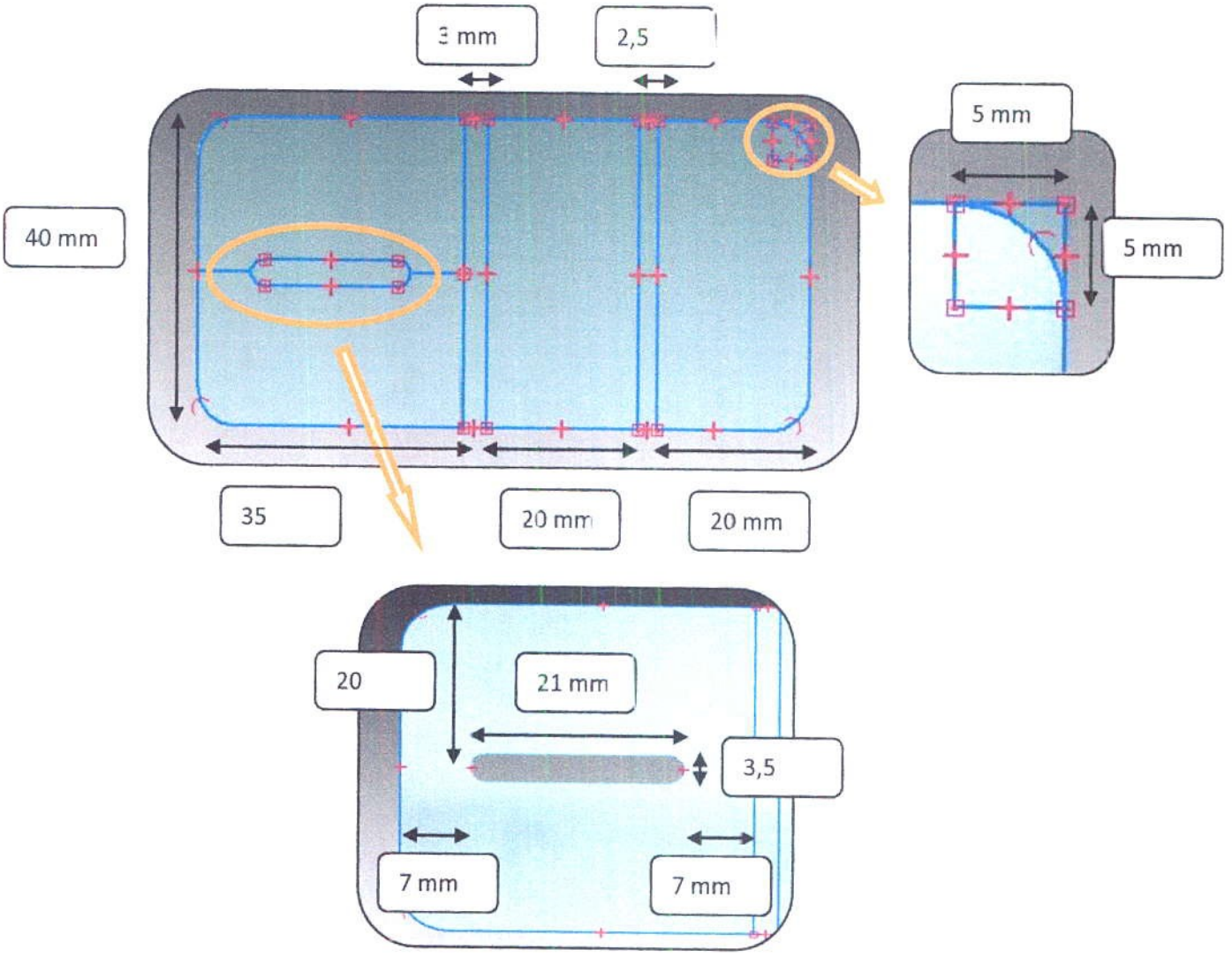


Figura 15. PIEZA BS1_MN

PROTECTOR SERVO 1 _ MUÑECA (PS1_MN)



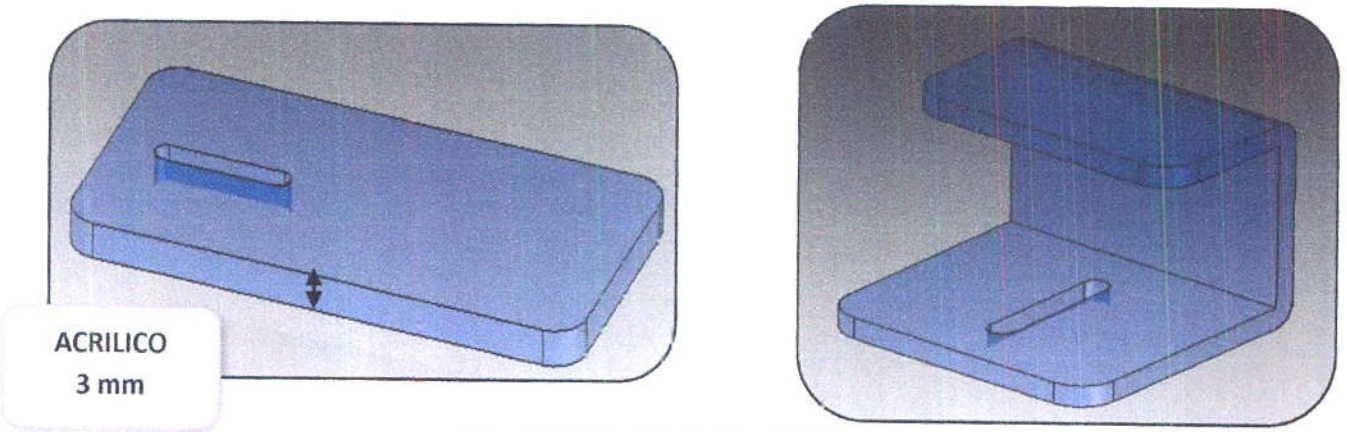
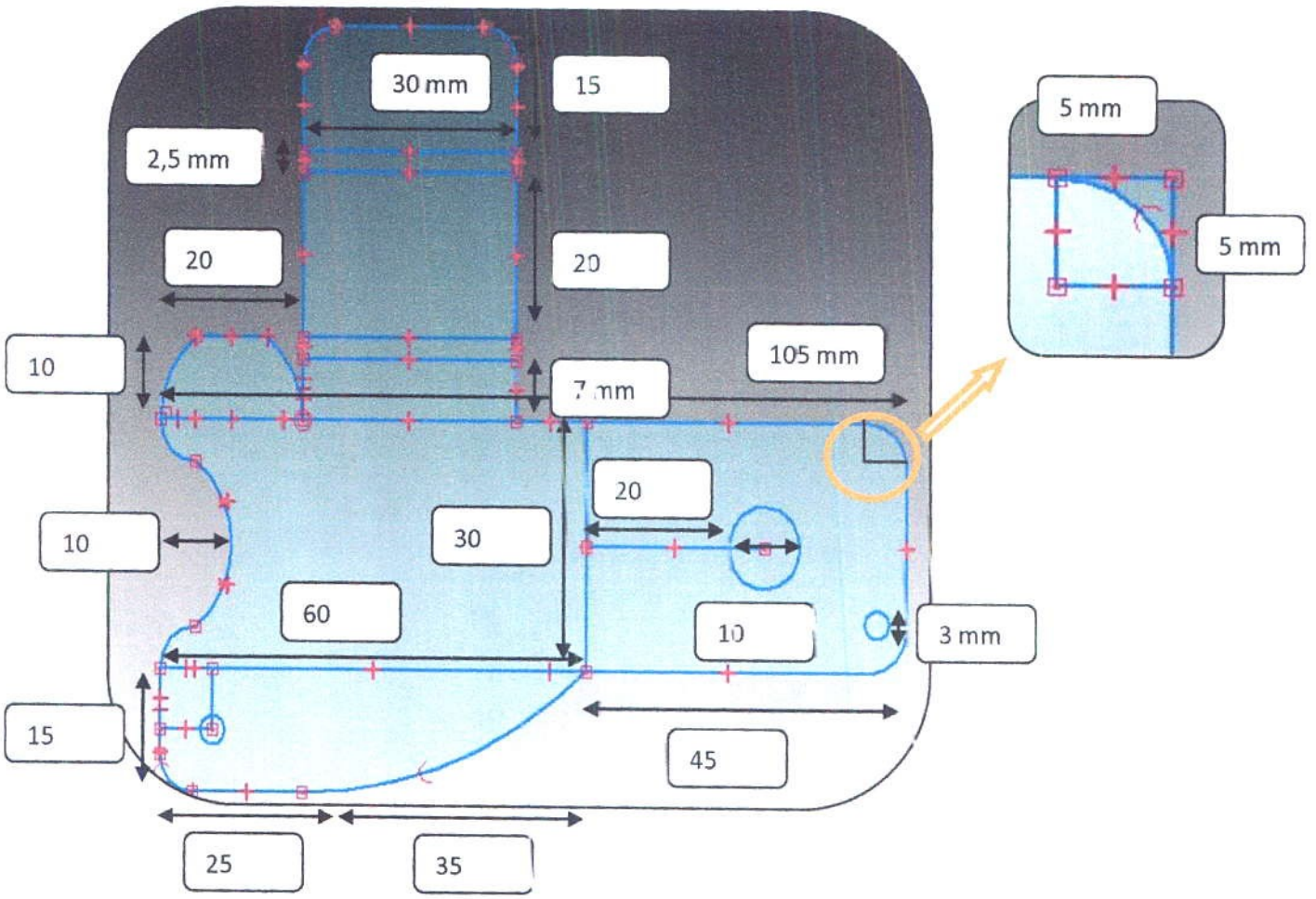
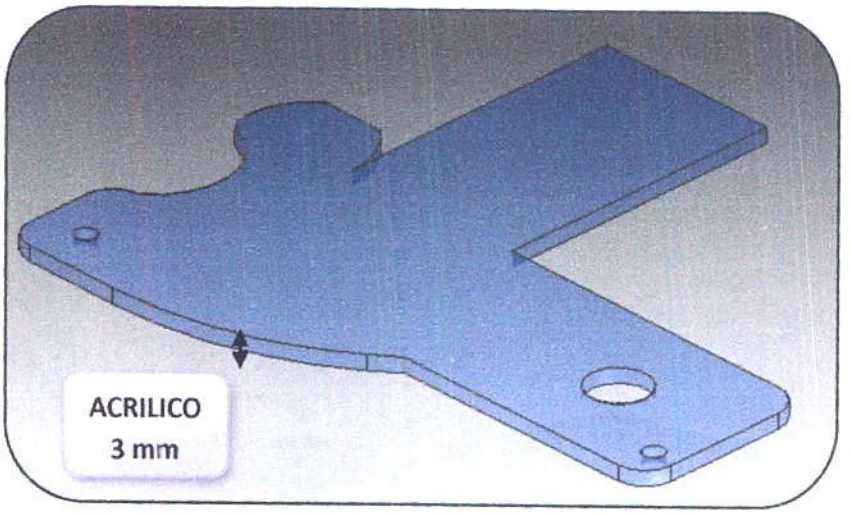


Figura 16. PIEZA PS1_MN

2.2 Protector Servomotor 2

PROTECTOR SERVO 2 _ METACARPIANO I DEDO (PS2_MI)





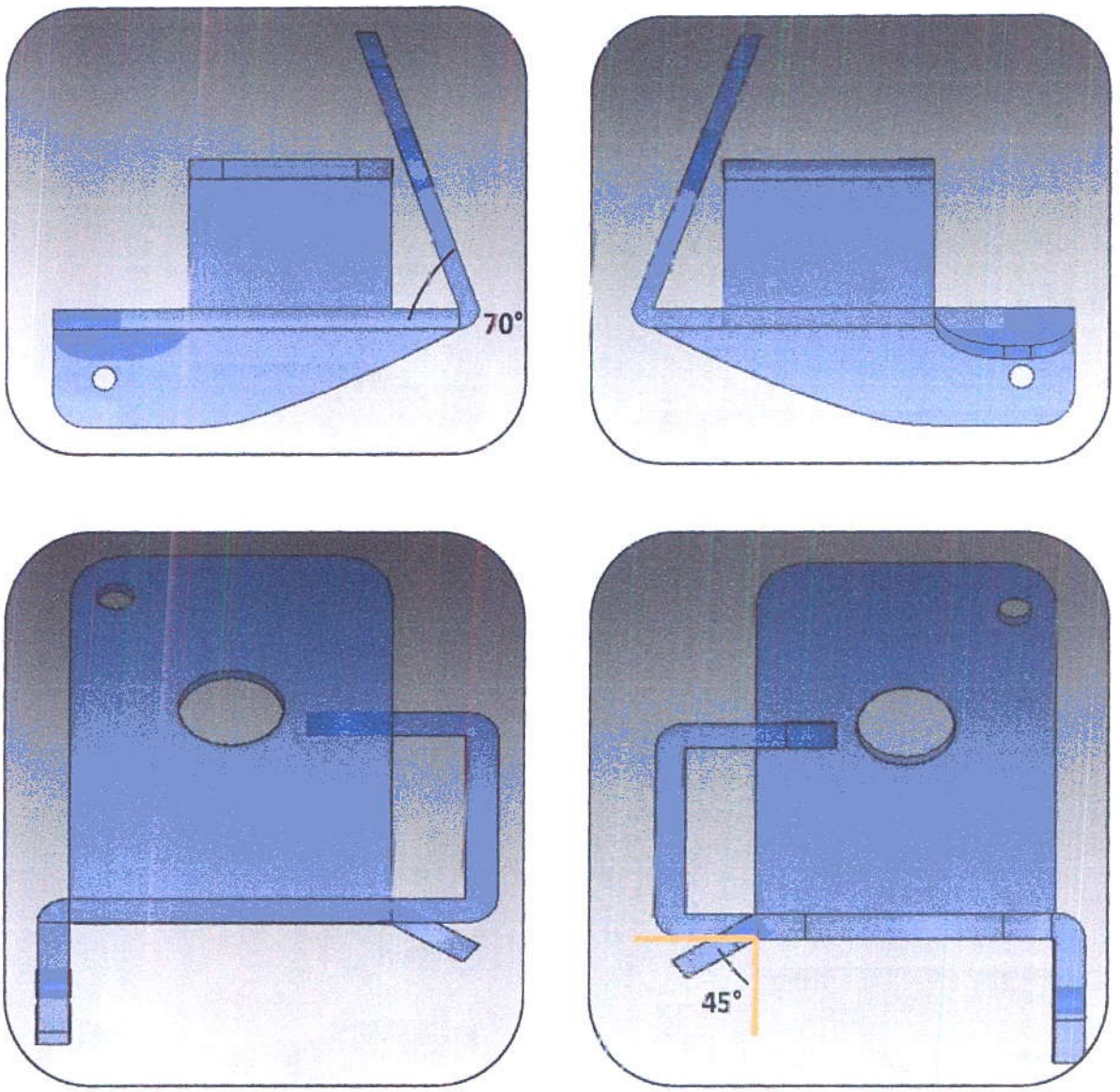


Figura 17. PIEZA PS2_MI

2.3 Protector Servomotor 3

PROTECTOR SERVO 3 _ FALANGE PROXIMAL I DEDO (PS3_FPI)

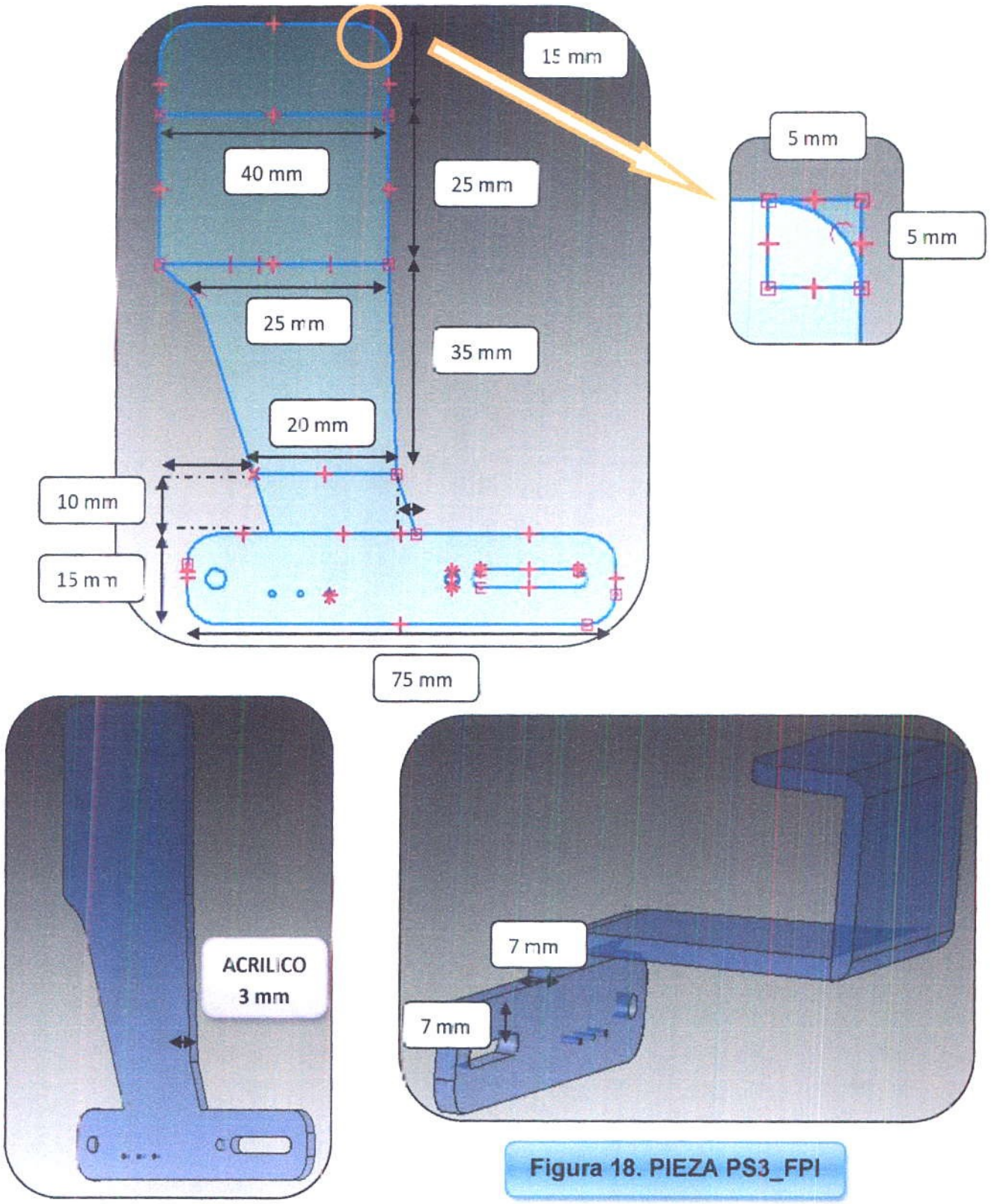
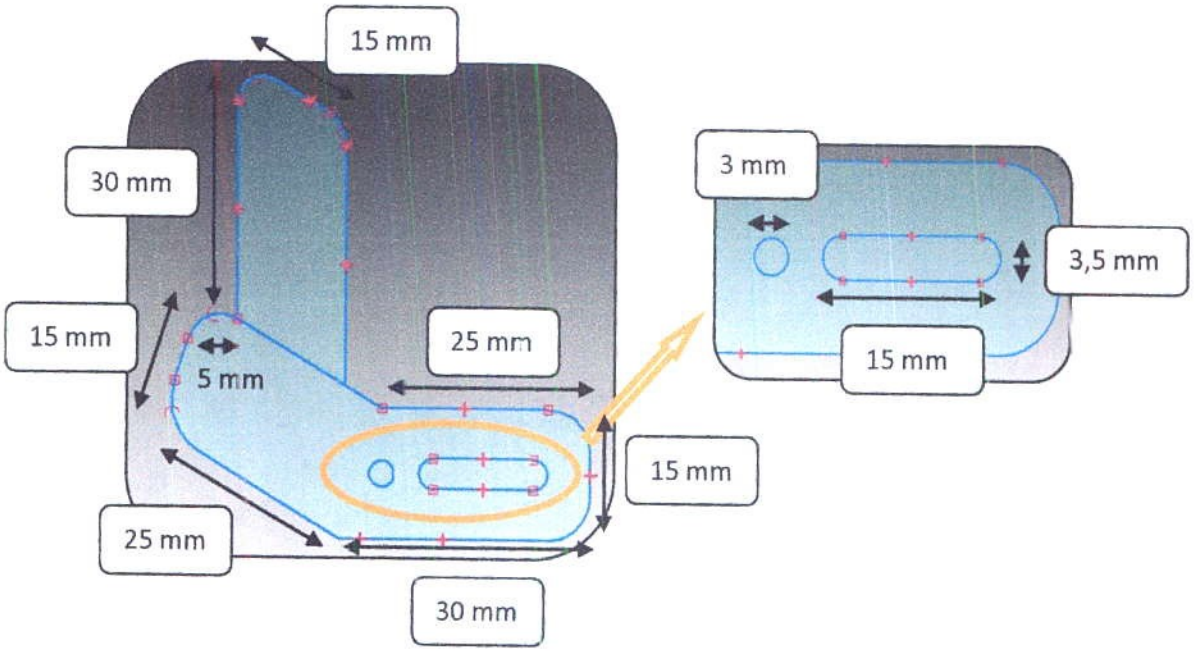


Figura 18. PIEZA PS3_FPI

BASE Y PROTECTOR_ FALANGE DISTAL I DEDO (BP_FDI)



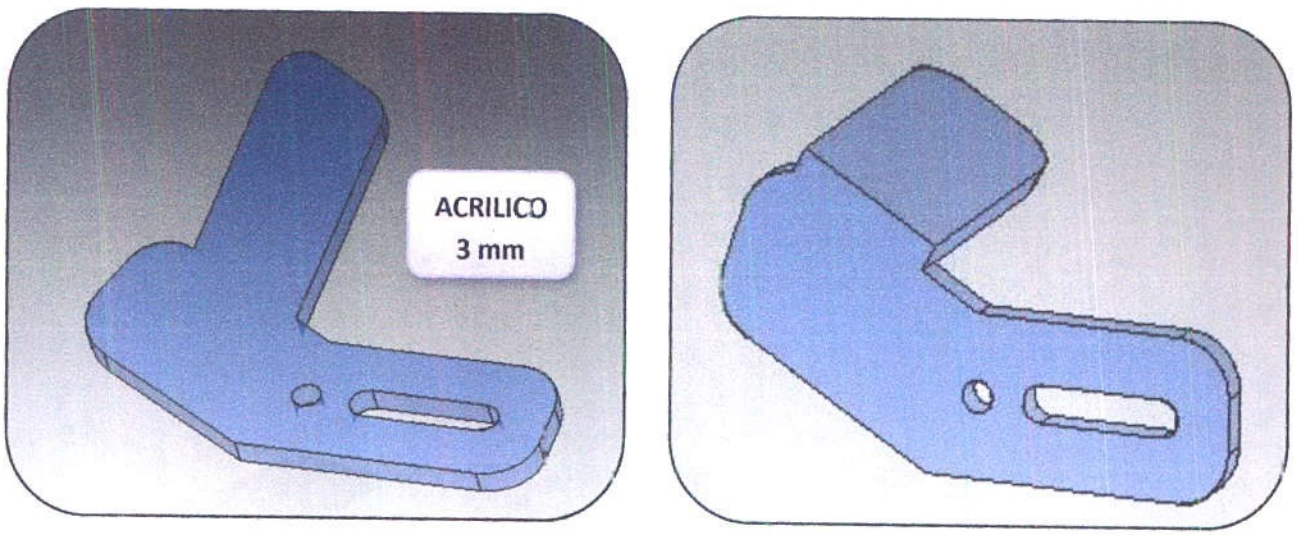


Figura 19. PIEZA BP_FDI

2.4 Protector Servomotor 4

BASE SERVO 4_ METACARPIANO II DEDO (BS4_MII)

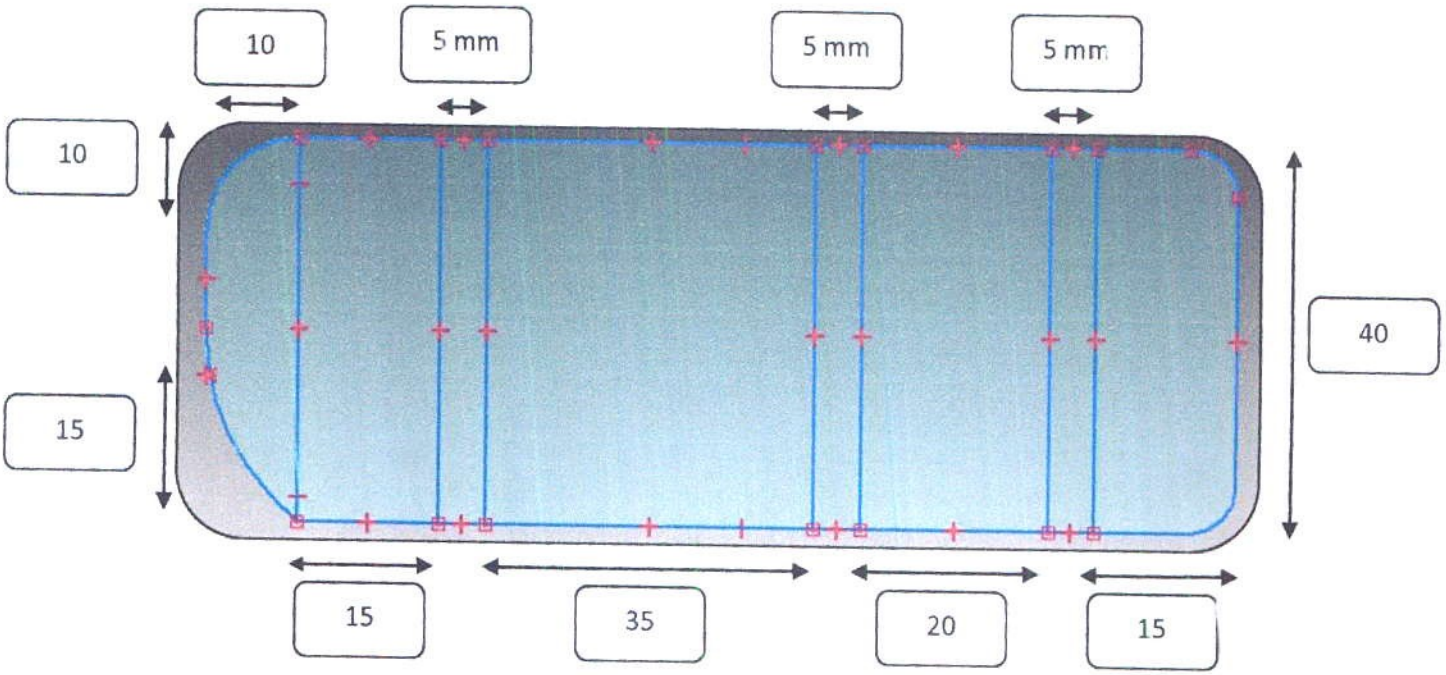
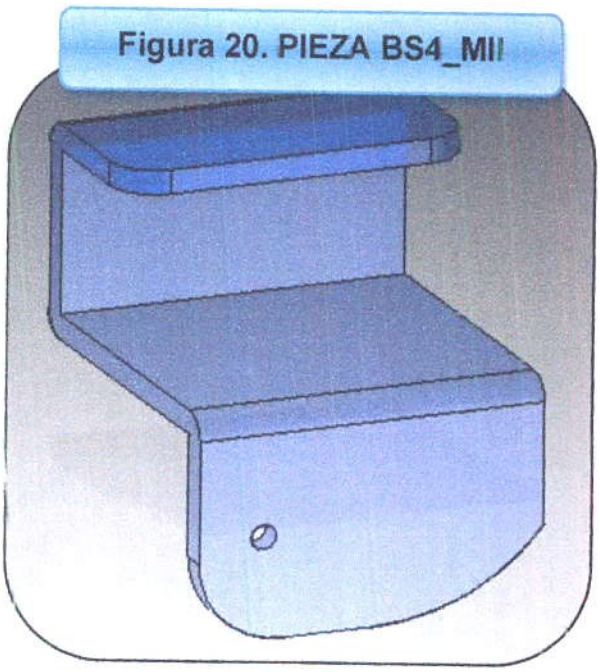




Figura 20. PIEZA BS4_MII



2.5 Protector Servomotor 5

PROTECTOR SERVO 5 _ FALANGE PROXIMAL II - III DEDO (PS5_FPII)

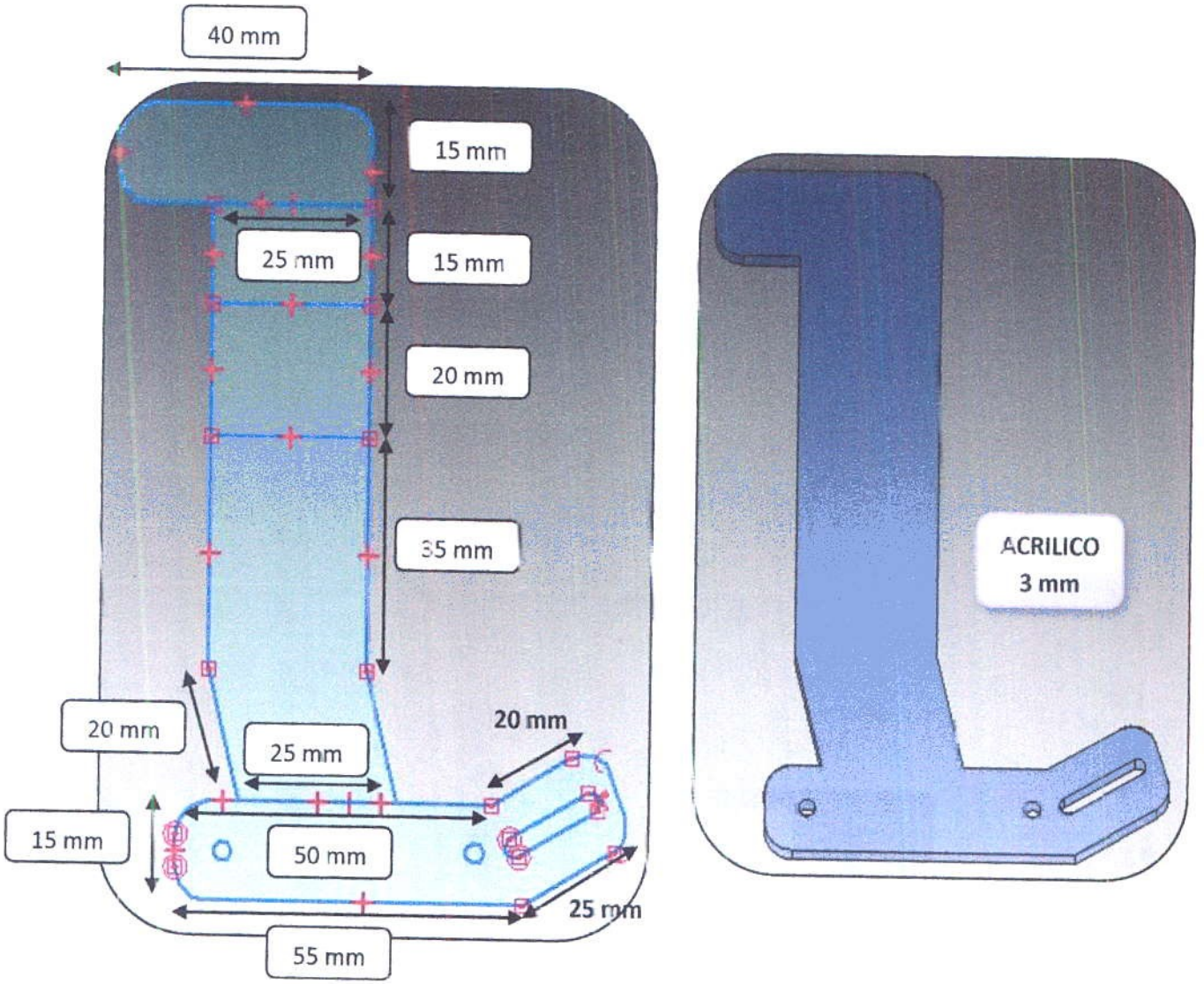


Figura 21. PIEZA PS5_FPII

BASE _ FALANGE MEDIAL II - III DEDO (B_FMII)

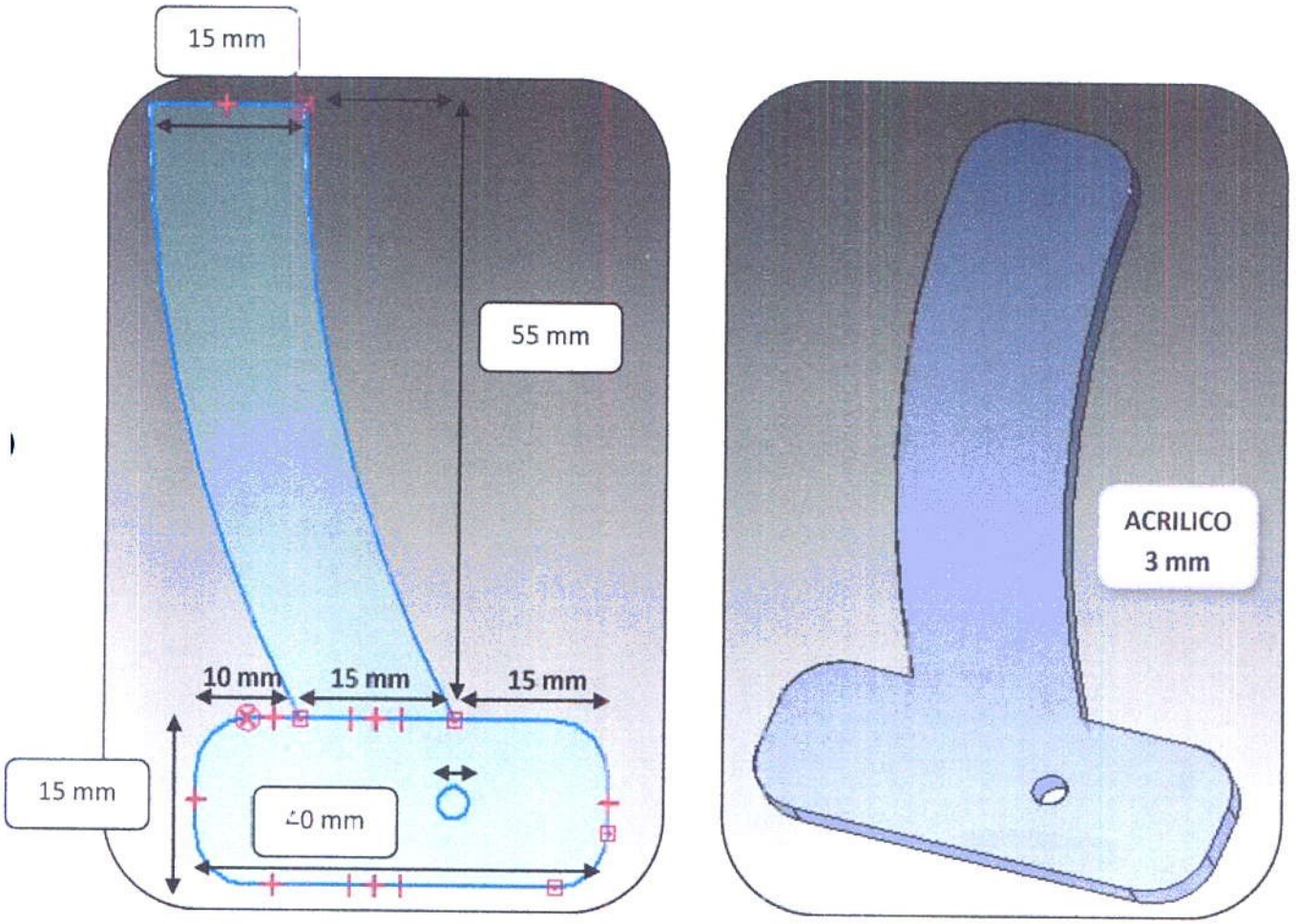
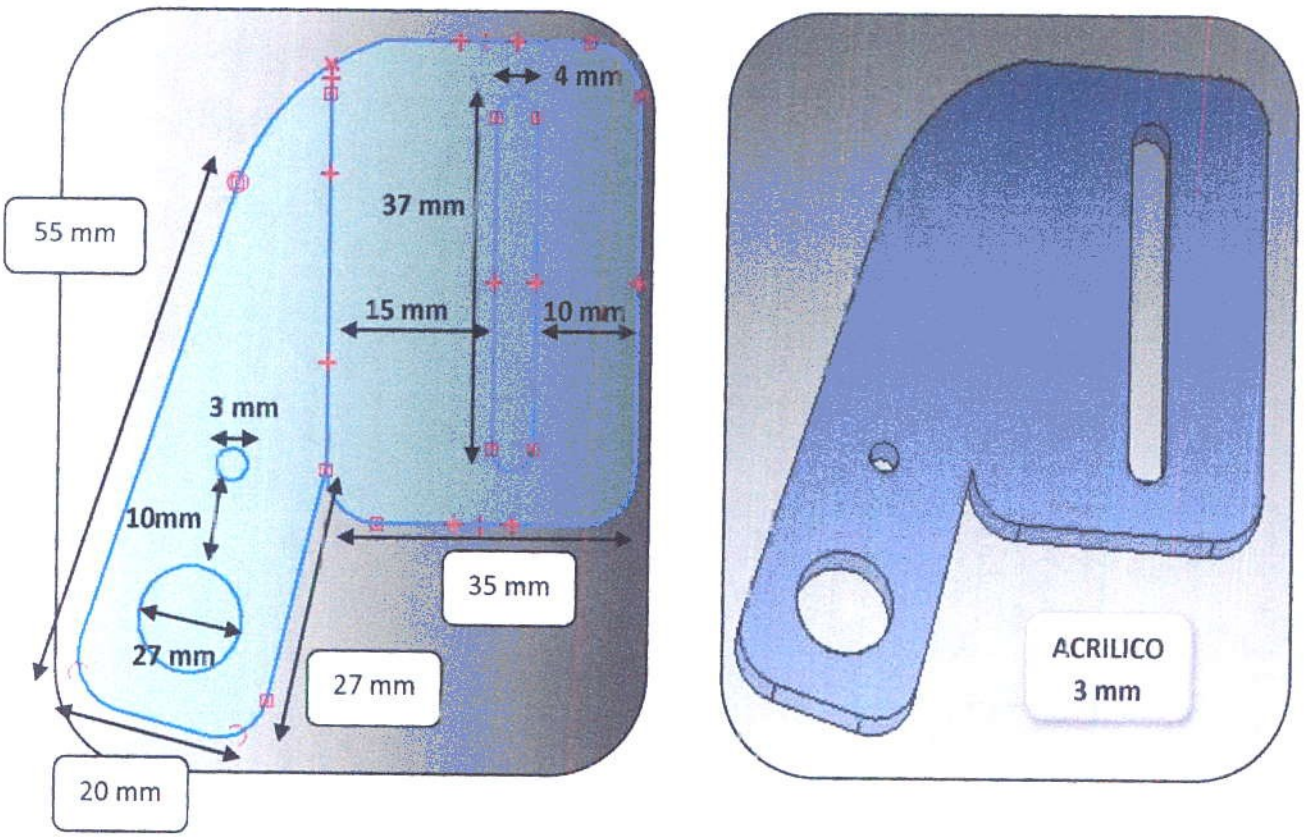


Figura 22. PIEZA B_FMII

PALANCA SERVO 5 _ FALANGE MEDIAL II - III DEDO (P5_FMII)



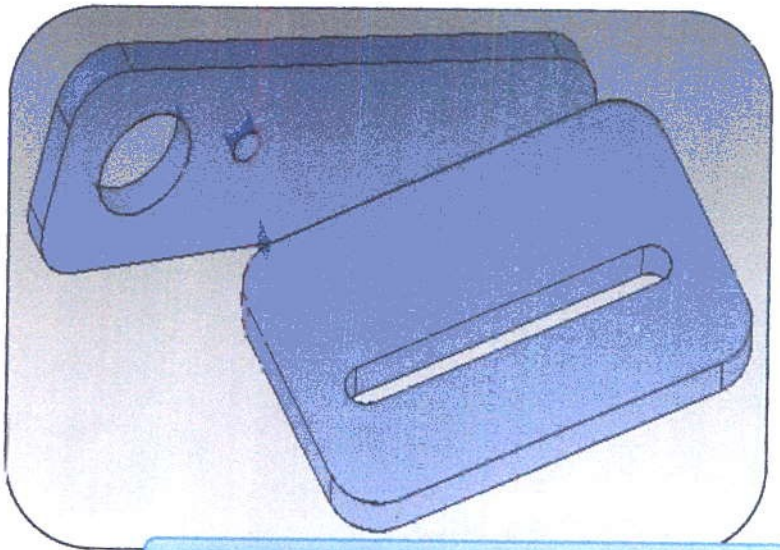


Figura 23. PIEZA P5_FMII

2.6 Protector Servomotor 6

PROTECTOR SERVO 6 _ FALANGE PROXIMAL IV - V DEDO (PS6_FPIV)

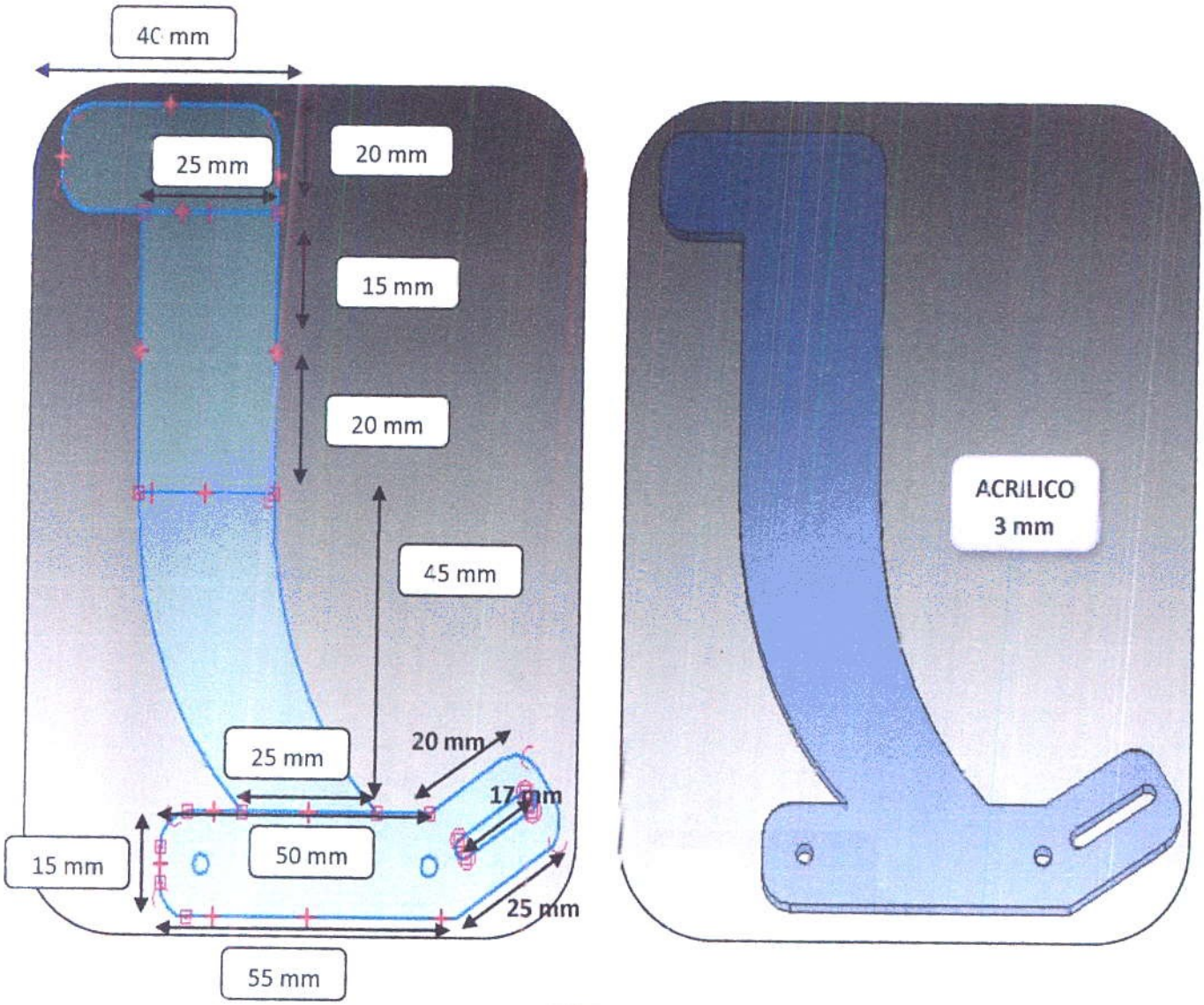
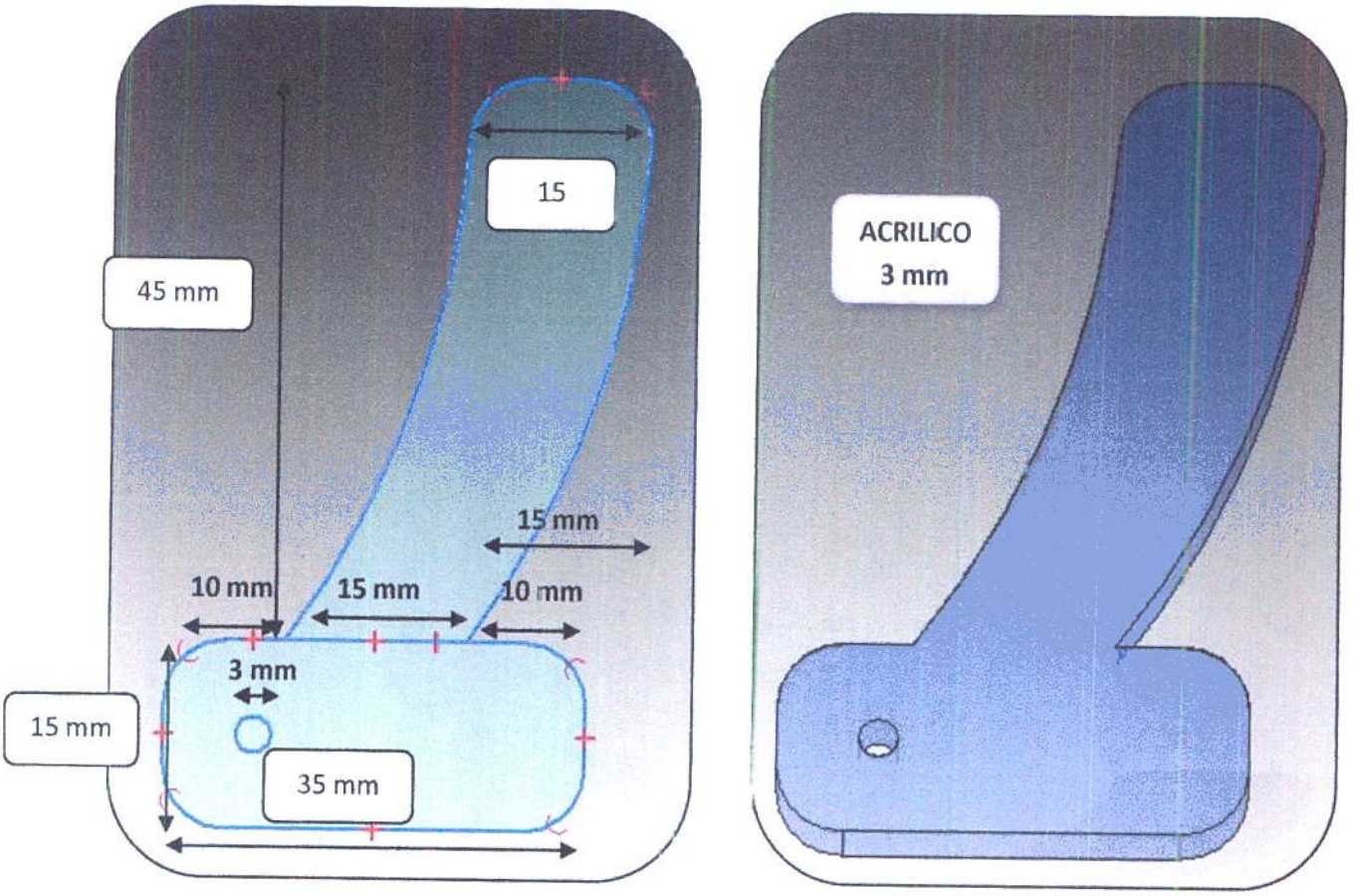


Figura 24. PIEZA PS6_FPIV

BASE _ FALANGE MEDIAL IV - V DEDO (B_FMIV)



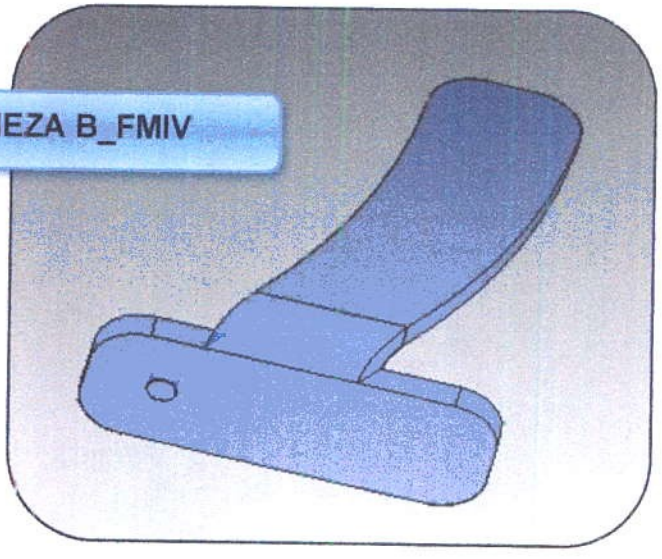
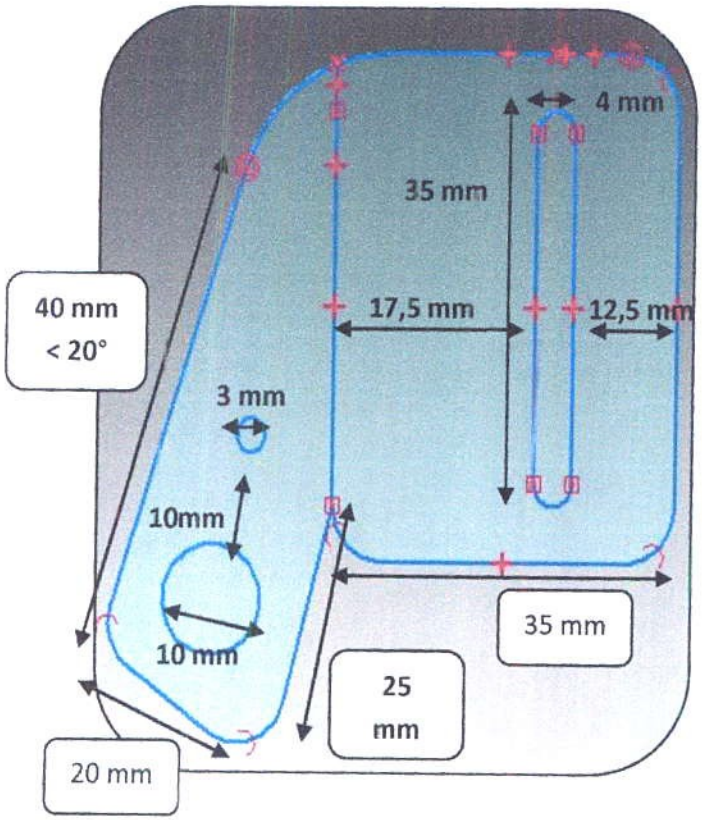


Figura 25. PIEZA B_FMIV

PALANCA SERVO 6 _ FALANGE MEDIAL IV – V DEDO (P7_FMIV)





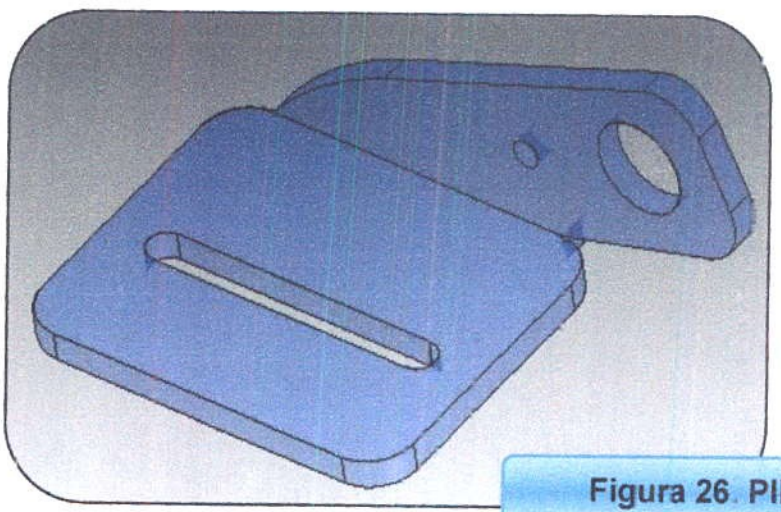


Figura 26. PIEZA P_FMIV

2.7 Protector Servomotor 7

BASE SERVO 7_ METACARPIANO IV – V DEDO (BS6_MIV)

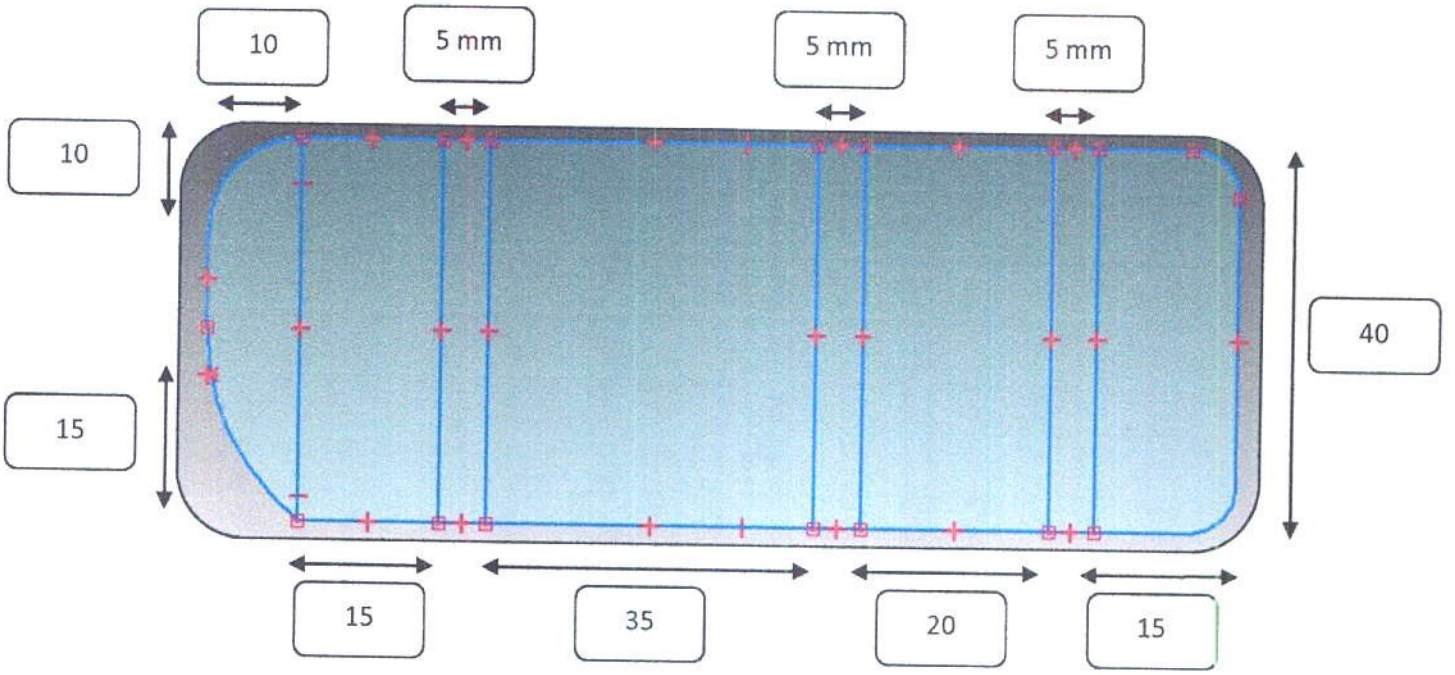
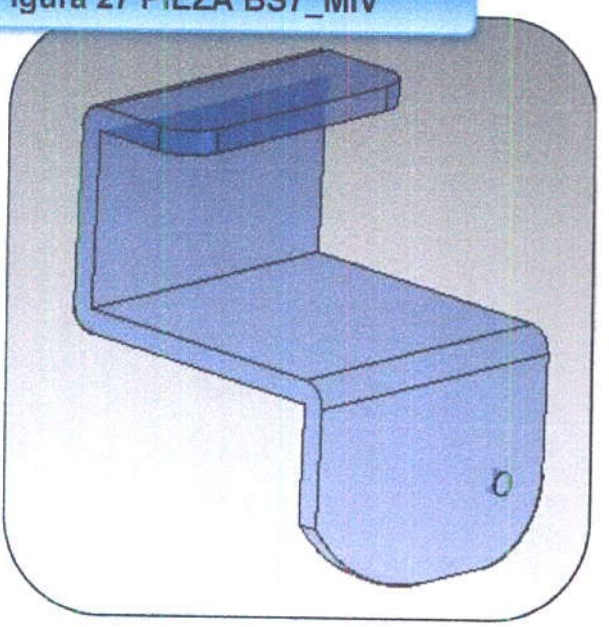


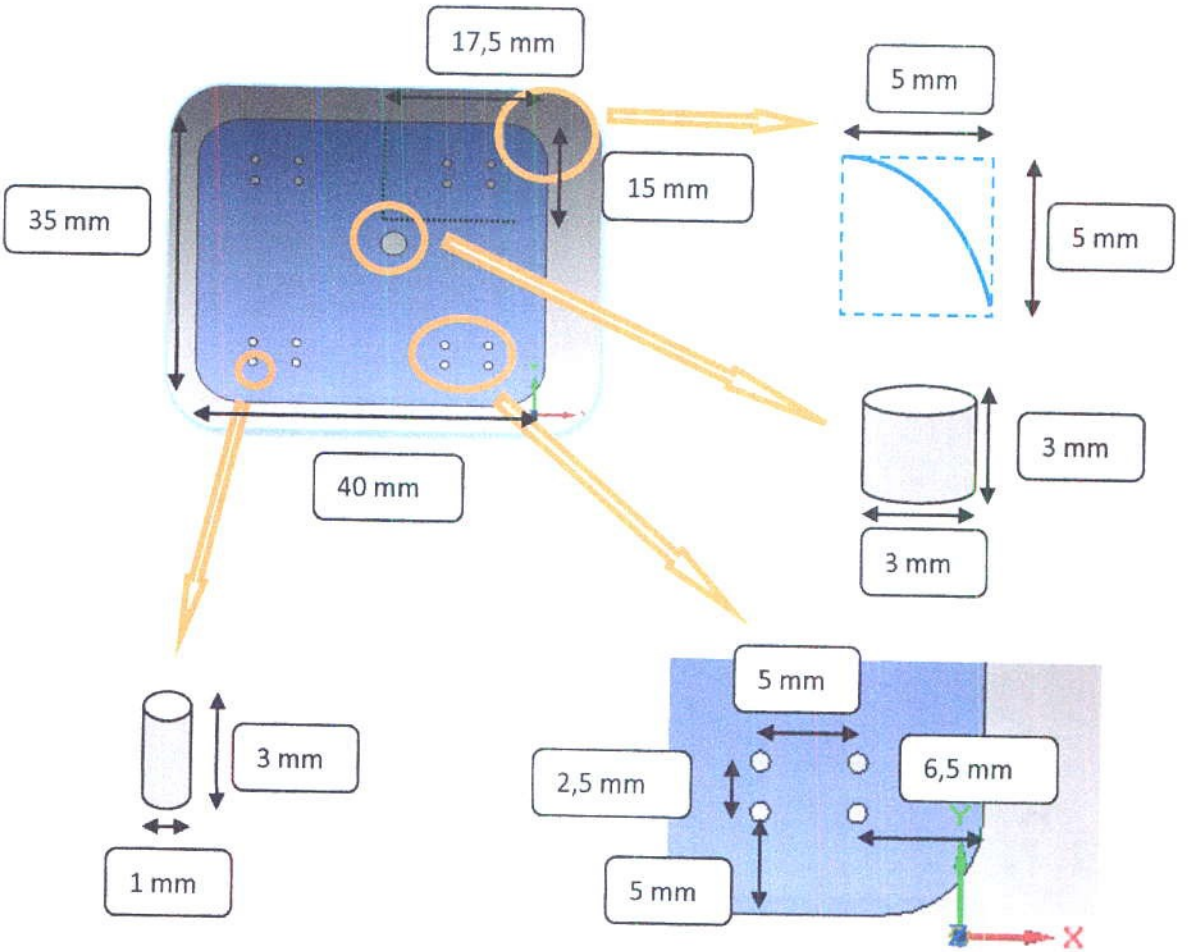


Figura 27 PIEZA BS7_MIV



2.8 Protector Servomotor 8

(BASE SERVO 8 _ MUÑECA (BS8_MN))



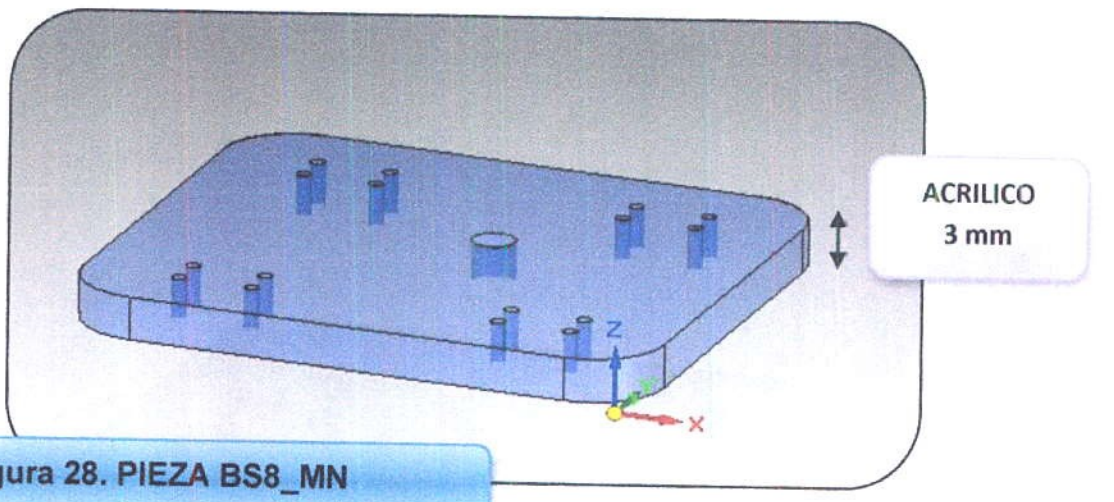


Figura 28. PIEZA BS8_MN

PROTECTOR SERVO 8 _ MUÑECA (PS8_MN)

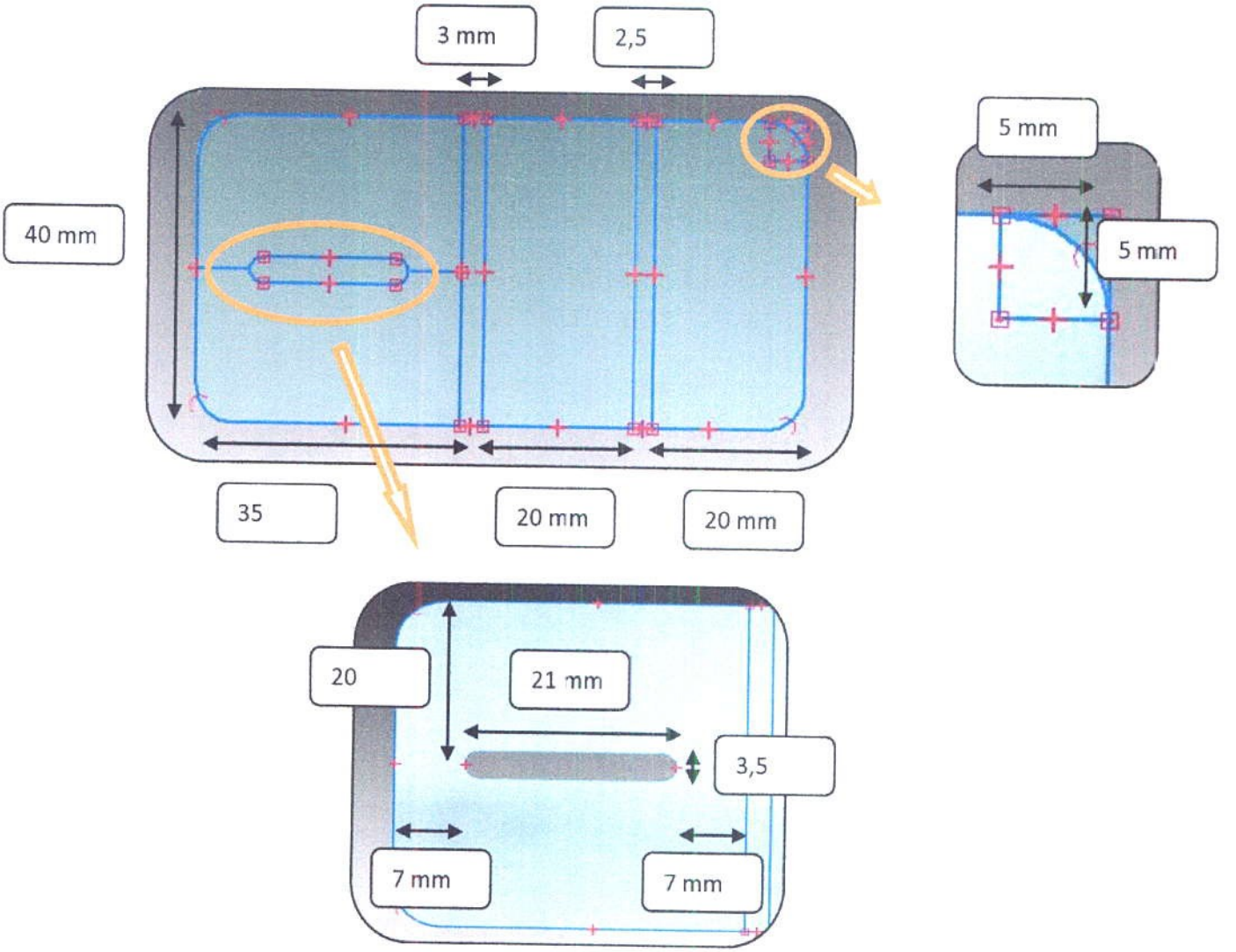
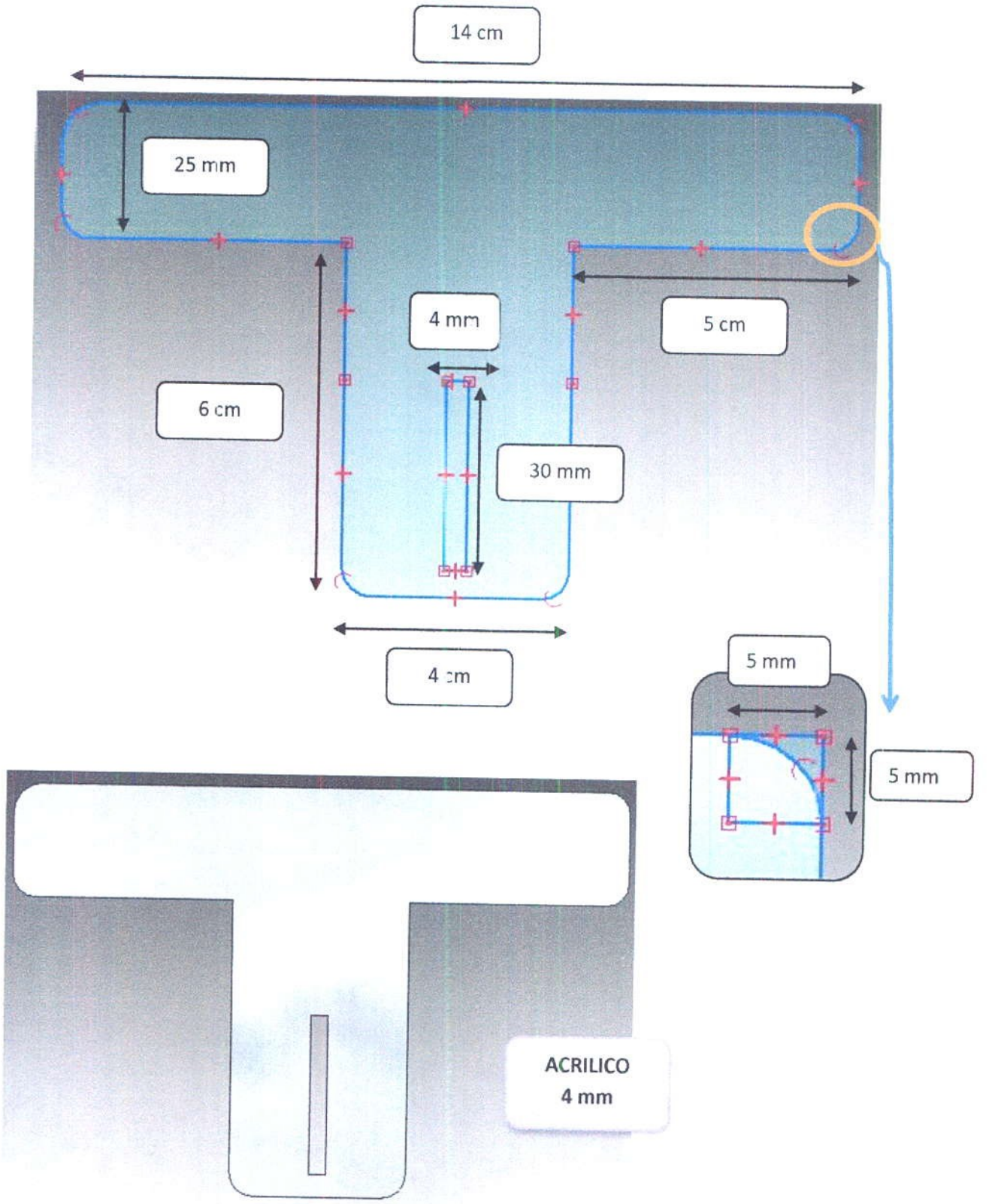


Figura 29. PIEZA PS8_MN

PALANCA SERVO 8 _ MUÑECA DEDO (PLS8_MN)



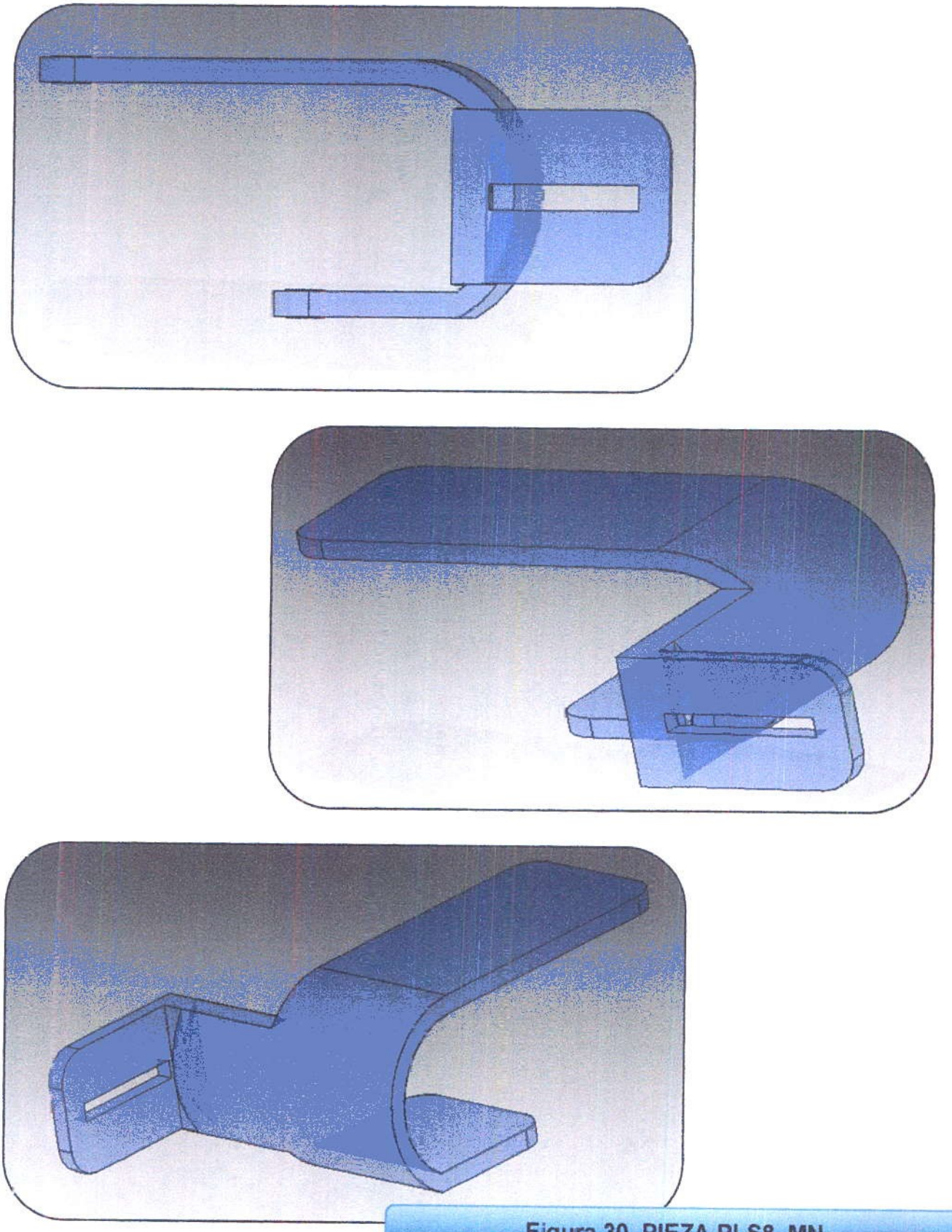


Figura 30. PIEZA PLS8_MN

3. ÓRTESIS ACTIVA:

3.1 DISEÑO DEL MOVIMIENTO I METACARPIANO: Componentes

Fig 31. PIEZA BS1_MN

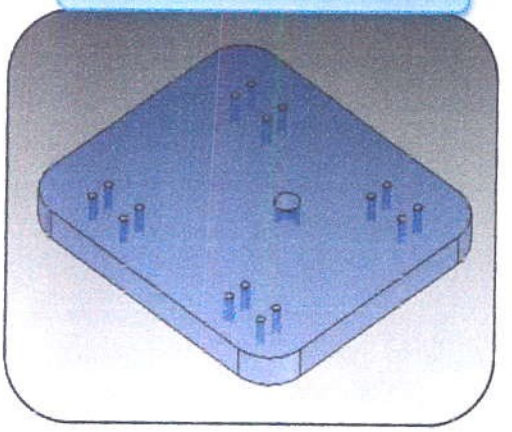


Fig 32. ESLABON

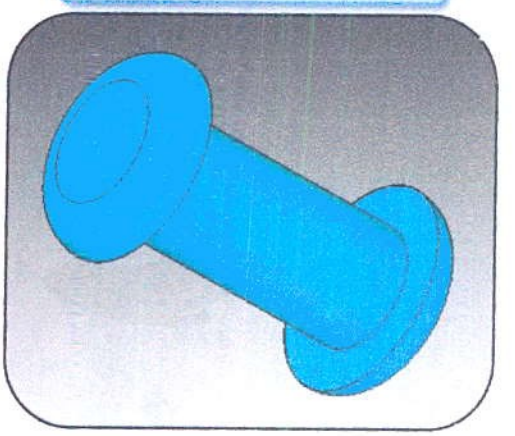


Fig 33. PIEZA PS1_MN

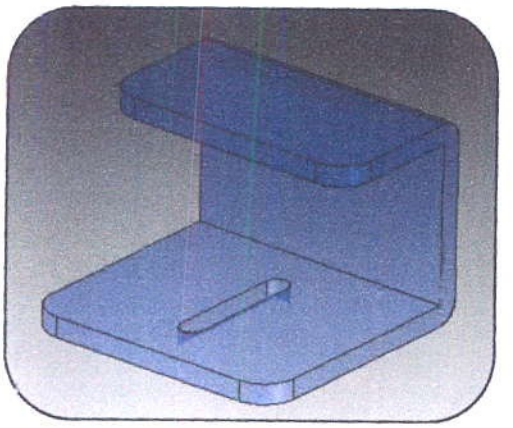


Fig 34. SERVOMOTOR

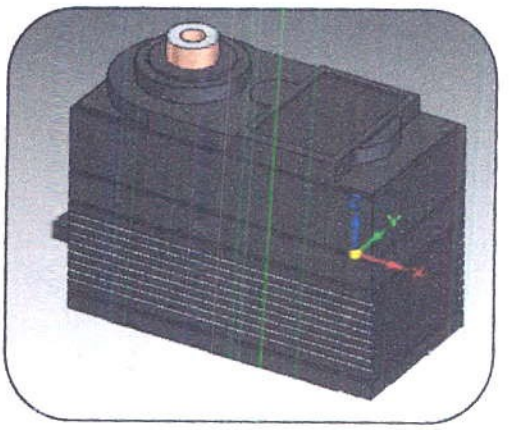


Fig 35. PALANCA SERVO

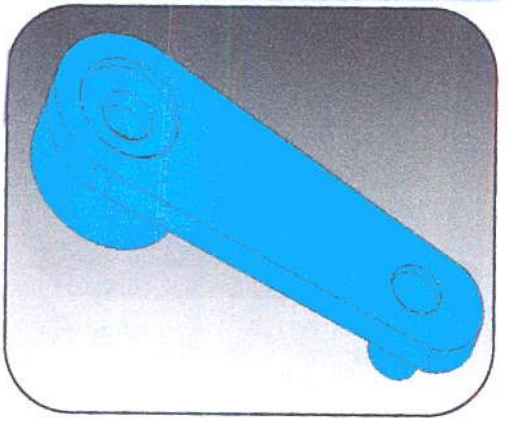
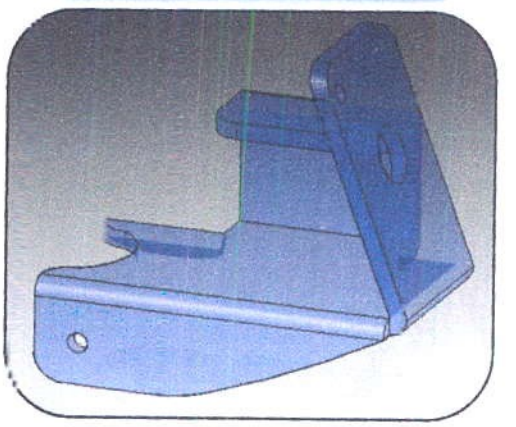
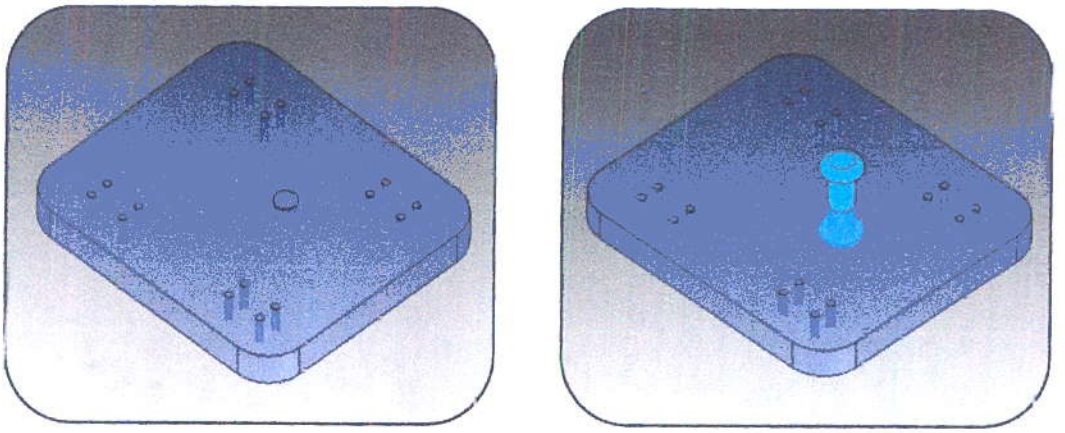


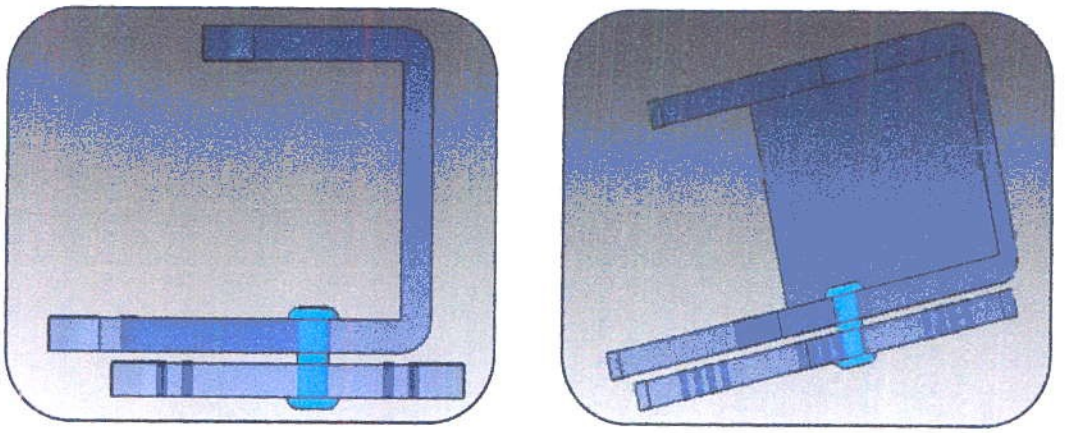
Fig 36. PIEZA PS2_MI



- El remache en duraluminio tiene alineación axial con la base en acrílico.



- El remache en duraluminio une las bases en acrílico.



- El servomotor se integra a las bases de acrílico que dan protección

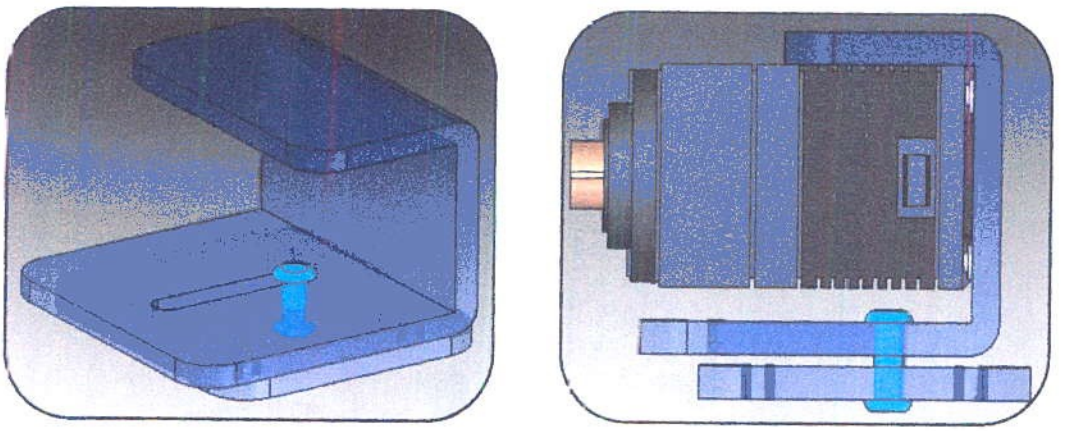
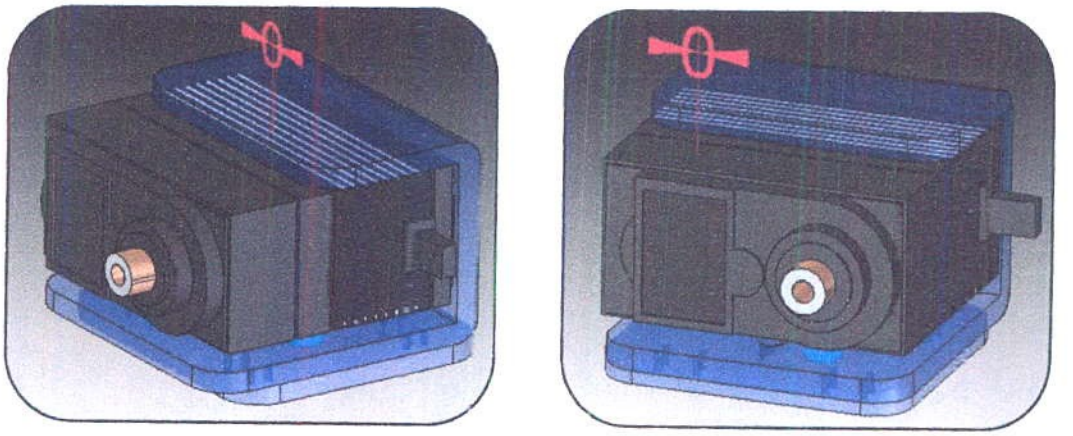
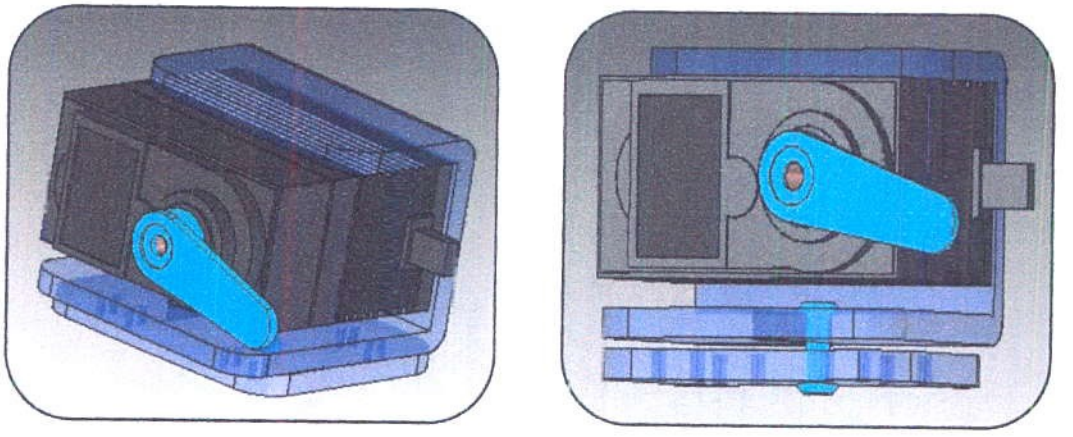


Figura 37. Montaje Componentes Exoesqueleto Base I dedo

- La ubicación del servomotor es estratégica para definir el movimiento



- La palanca del servomotor se integra a la pieza



- La palanca del servomotor se desplaza en forma circular

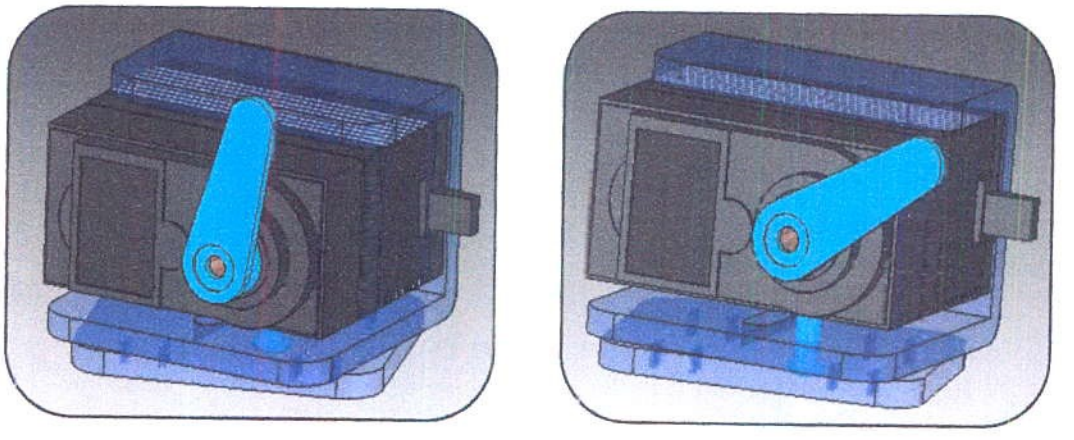
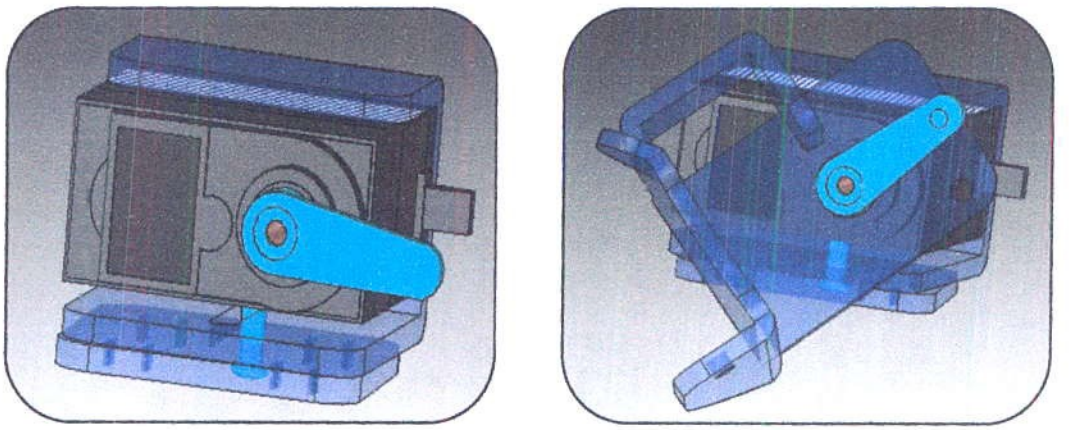


Figura 38. Montaje Componentes Exoesqueleto Base I dedo

- La siguiente pieza se une al conjunto.



- Se describe el movimiento rotacional de la pieza

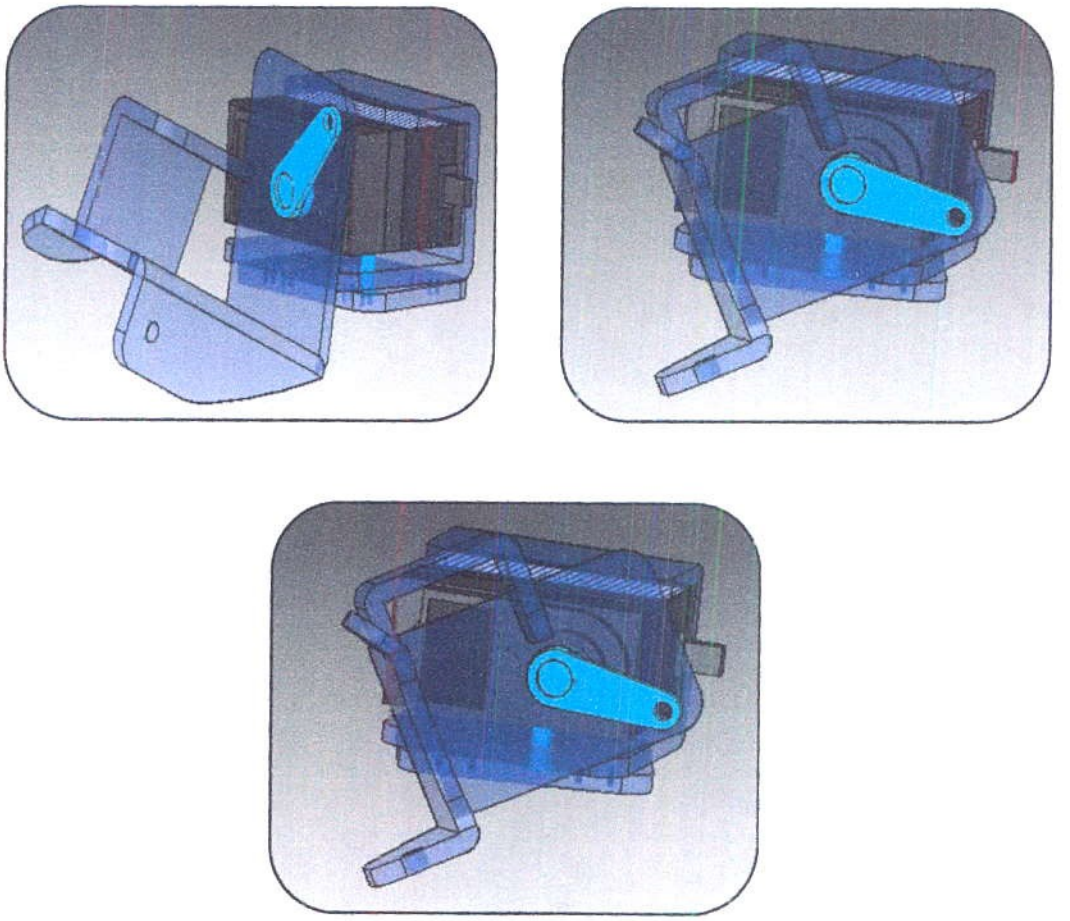


Figura 39. Montaje Componentes Exoesqueleto Base I dedo

3.2 DISEÑO DEL MOVIMIENTO ARTICULACION METACARPOFALANGICA I DEDO

Fig 40. PIEZA PS2_MI

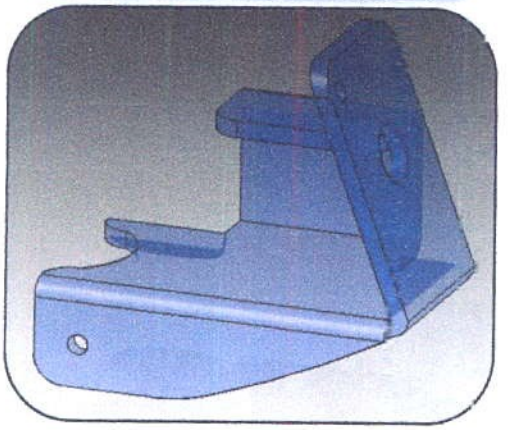


Fig 41. SERVOMOTOR

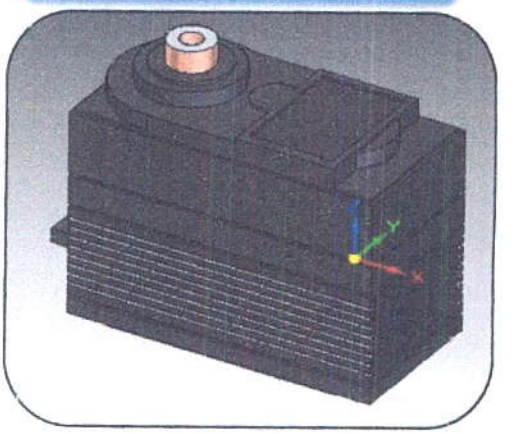


Fig 42. PALANCA SERVO

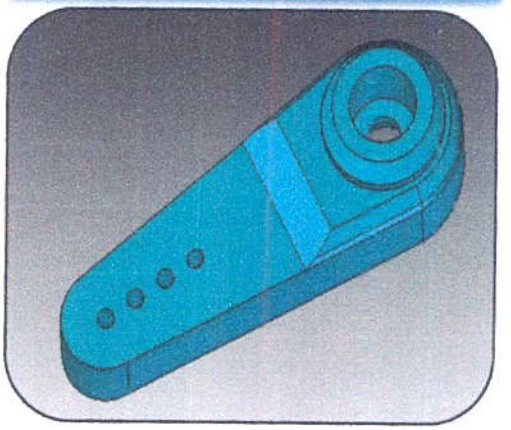


Fig 43. ESLABON

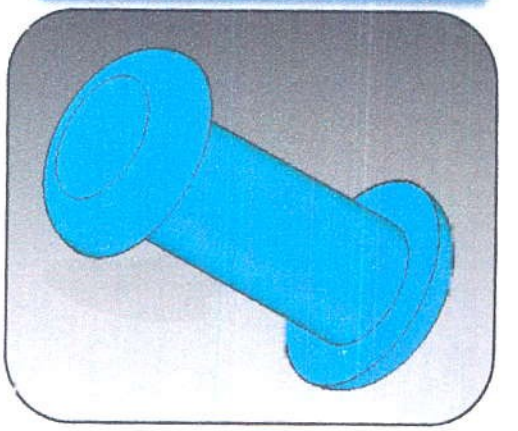
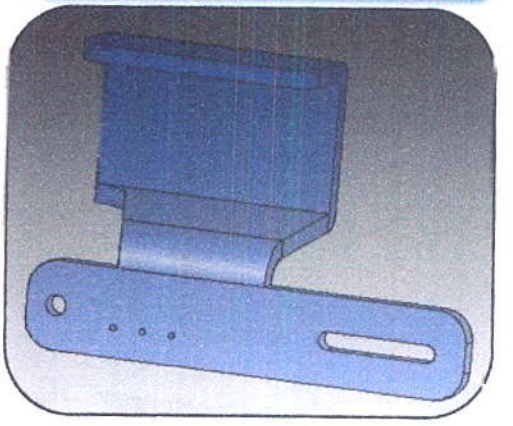
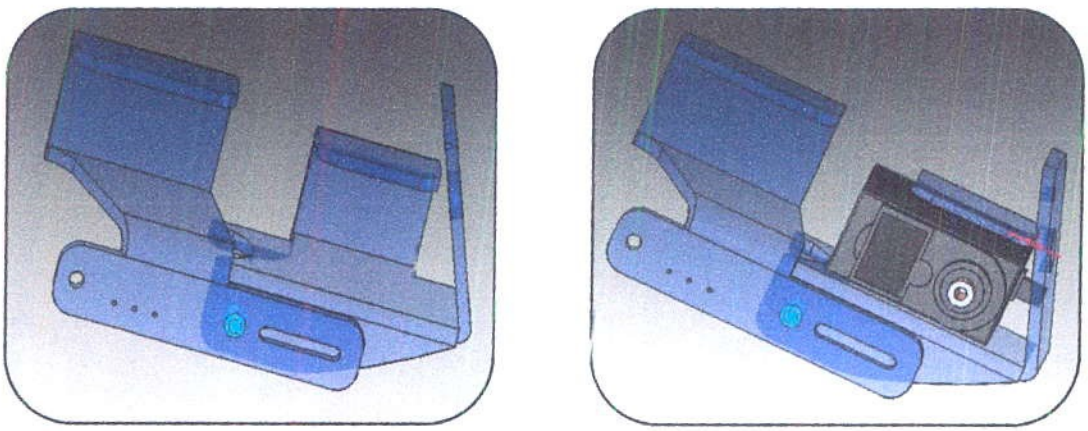


Fig 44. PIEZA PS3_FPI



- El servomotor se integra a las bases de acrílico que dan protección.



- La palanca del servomotor se desplaza en forma circular, generando un movimiento angular de la pieza distal

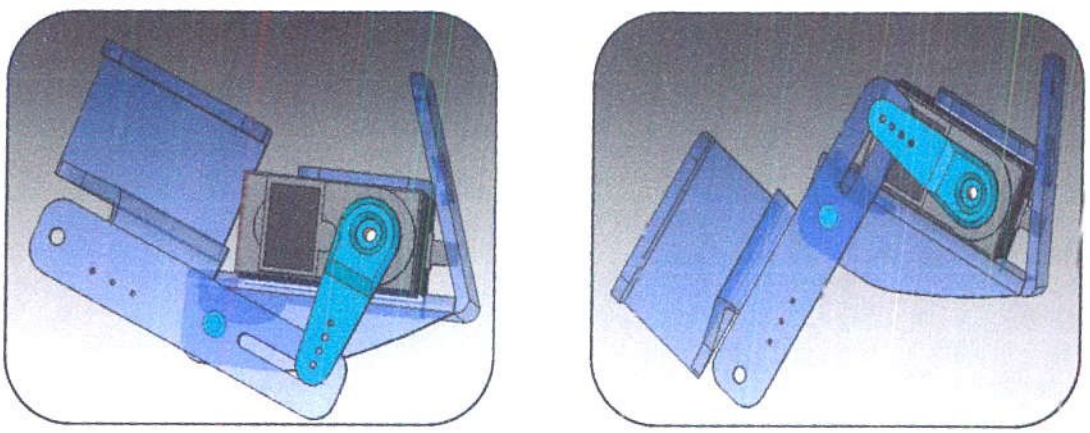


Figura 45. Montaje Componentes Articulación MF I dedo

3.3 DISEÑO DEL MOVIMIENTO ARTICULACION INTERFALANGICA I DEDO

Fig 46. PIEZA PS3_FPI

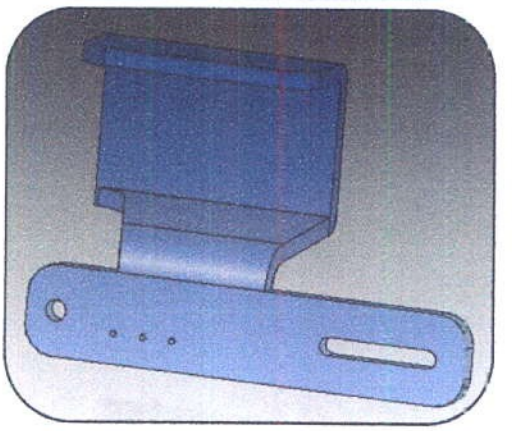


Fig 47. SERVOMOTOR

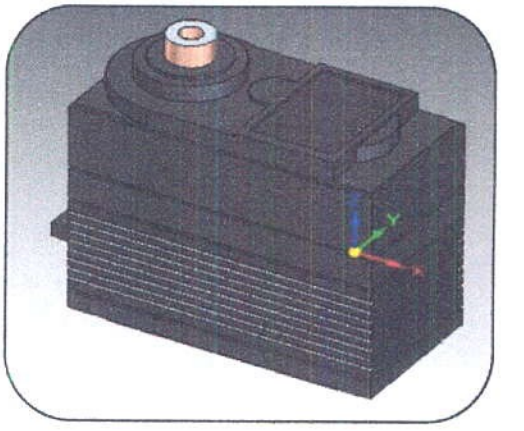


Fig 48. PALANCA SERVO

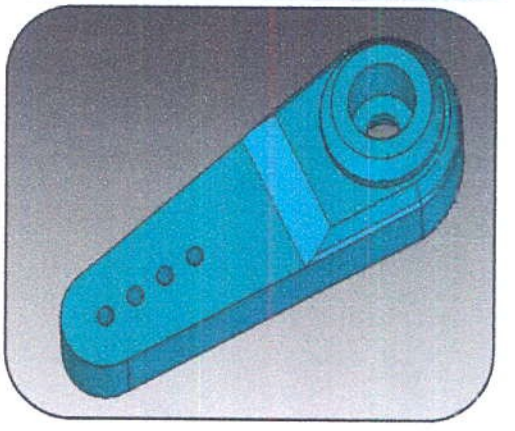


Fig 49. ESLABON

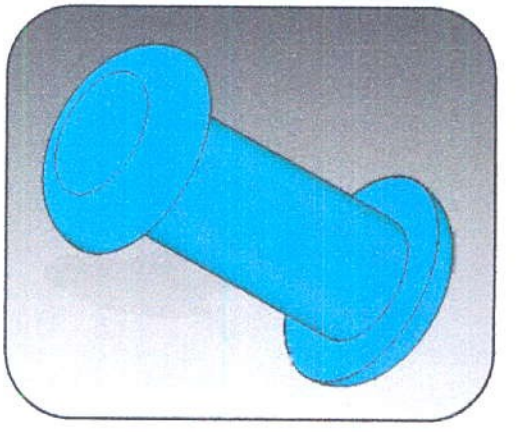
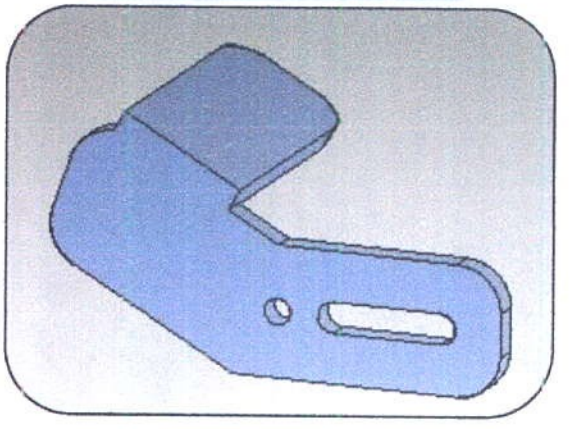
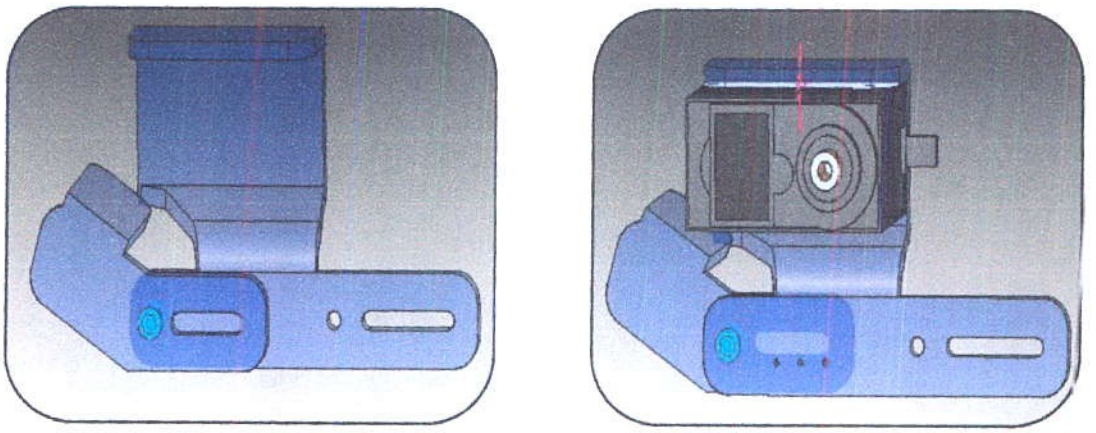


Fig 50. PIEZA BP_FDI



- El servomotor se integra a las bases de acrílico que dan protección.



- La palanca del servomotor se desplaza en forma circular, generando un movimiento angular de la pieza distal.

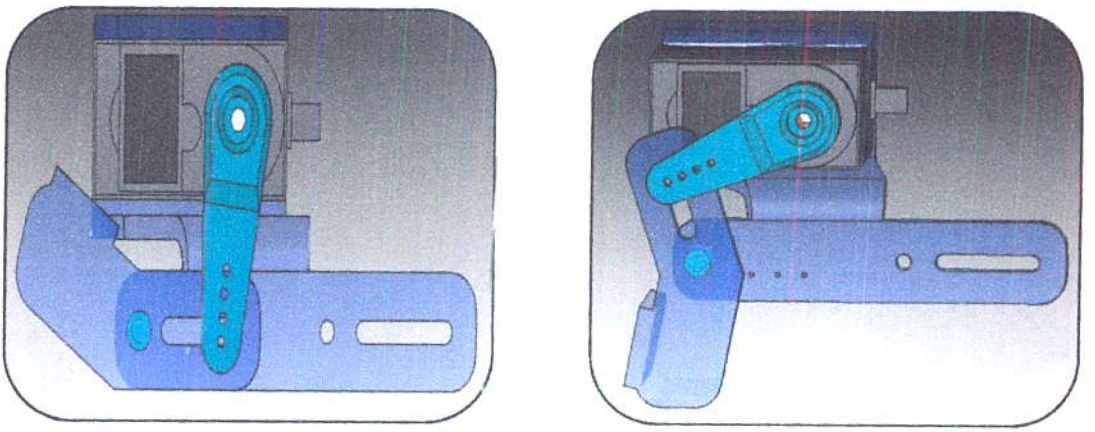


Figura 51. Montaje Componentes Articulación IF I dedo

3.4 DISEÑO DEL MOVIMIENTO ARTICULACION METACARPOFALANGICA II Y III DEDOS

Fig 52. PIEZA BS4_MII

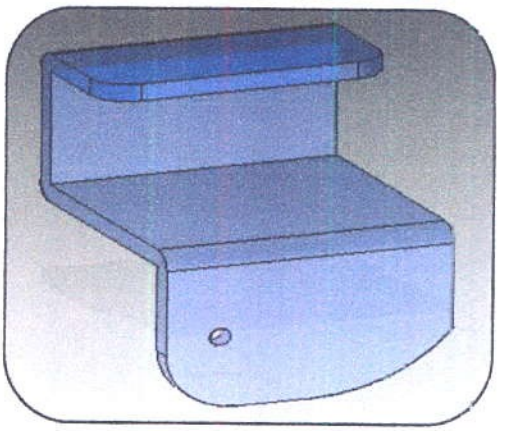


Fig 53. SERVOMOTOR

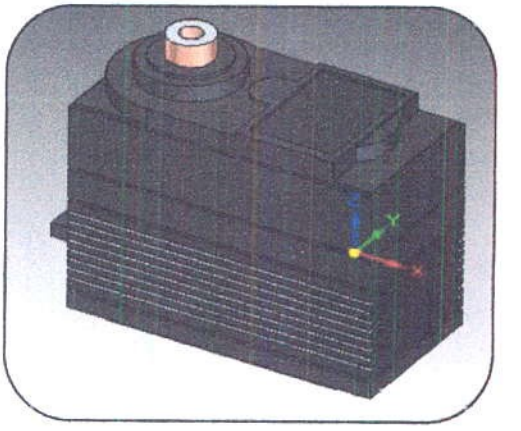


Fig 54. PALANCA SERVO

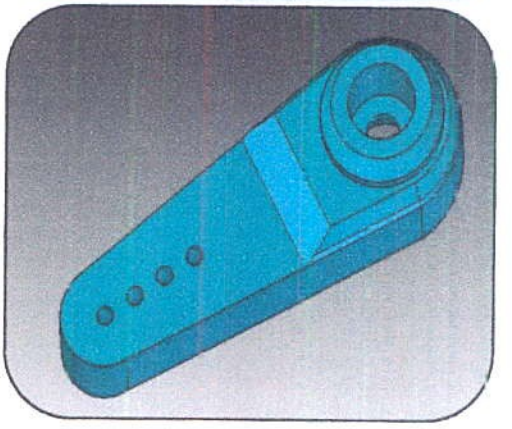


Fig 55. ESLABON

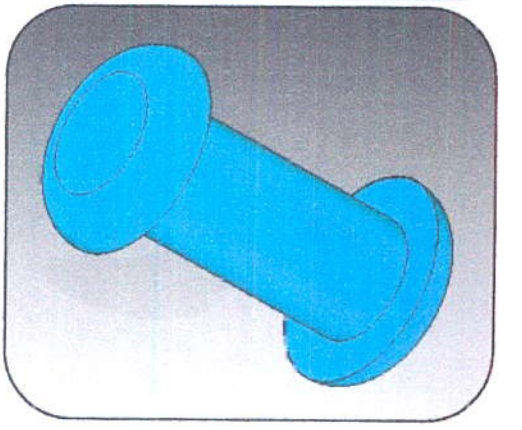
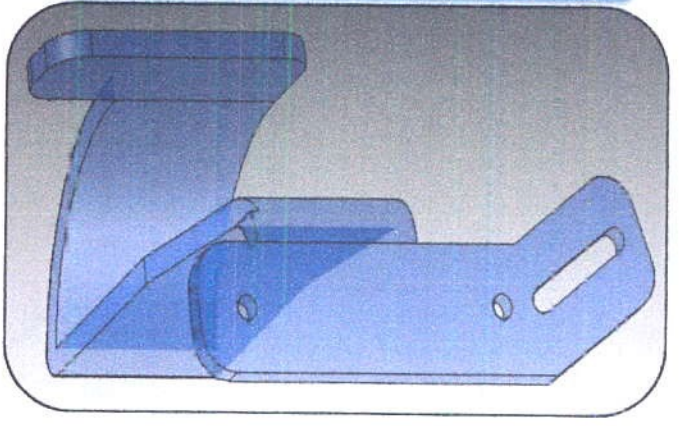
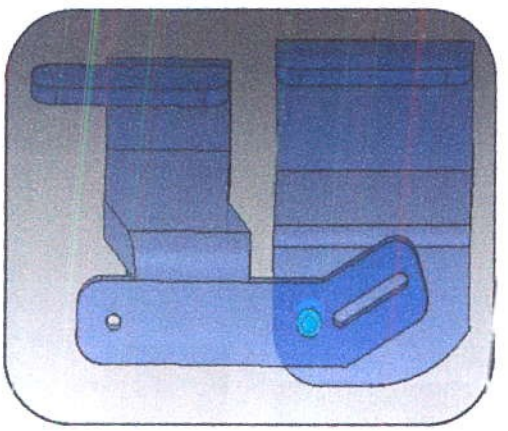
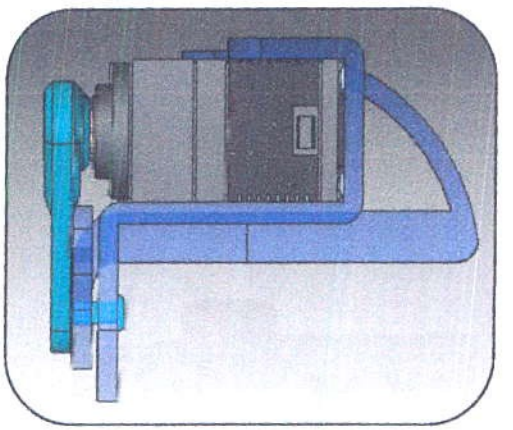


Fig 56. PIEZA PS5_FPII

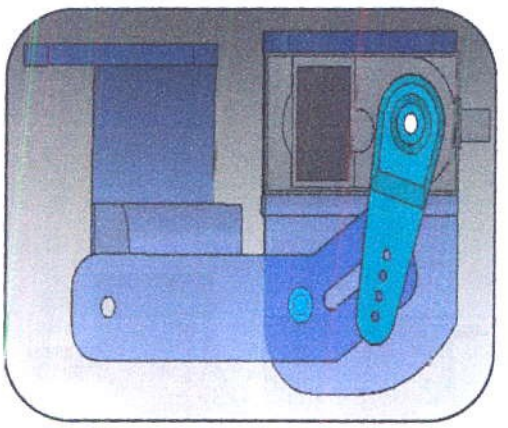


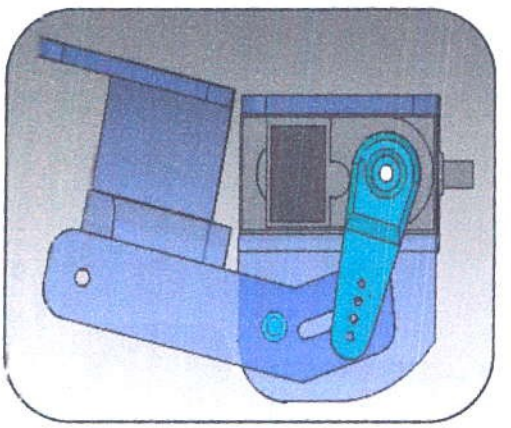
- El servomotor se integra a las bases de acrílico que dan protección.





- La palanca del servomotor se desplaza en forma circular, generando un movimiento angular de la pieza distal.





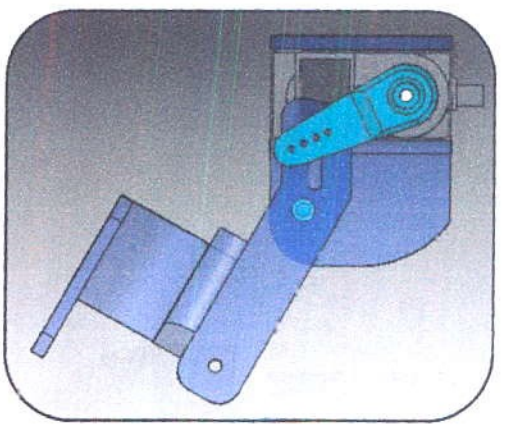


Figura 57. Montaje Componentes Articulación MF II y III dedos

3.5 DISEÑO DEL MOVIMIENTO ARTICULACION INTERFALANGICA PROXIMAL II Y III DEDOS

Fig 58. PIEZA PS5_FPIV

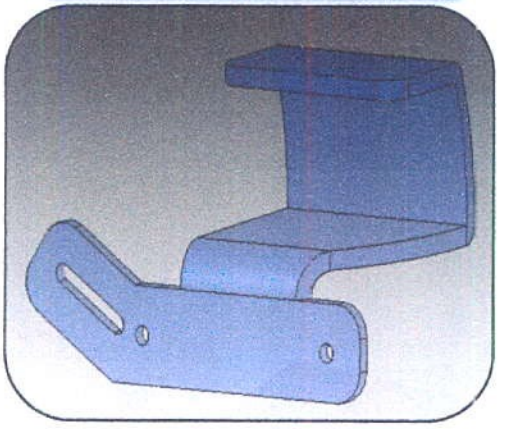


Fig 59. SERVOMOTOR

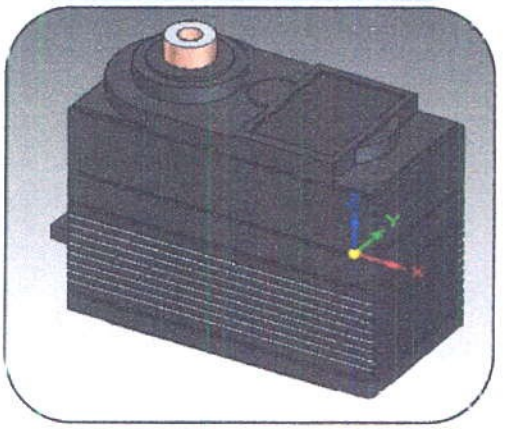


Fig 60. PALANCA SERVO

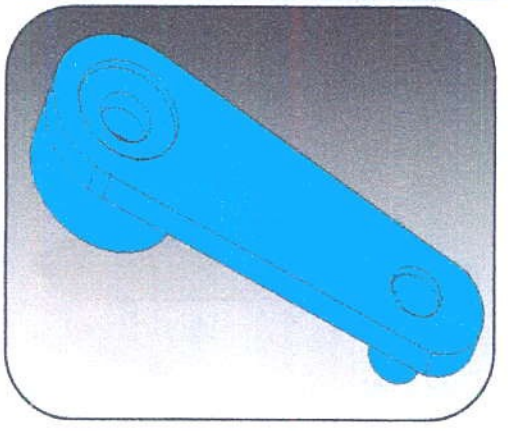


Fig 61. ESLABON

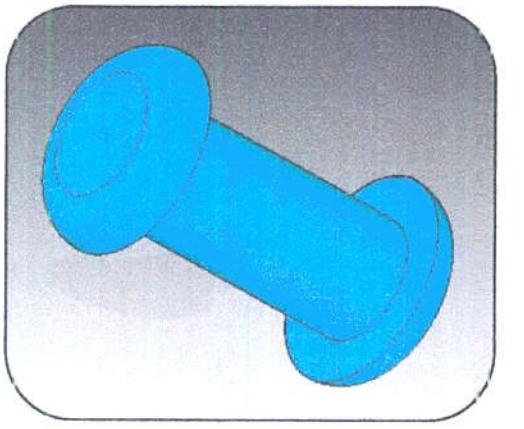


Fig 62. PIEZA B_FMIV

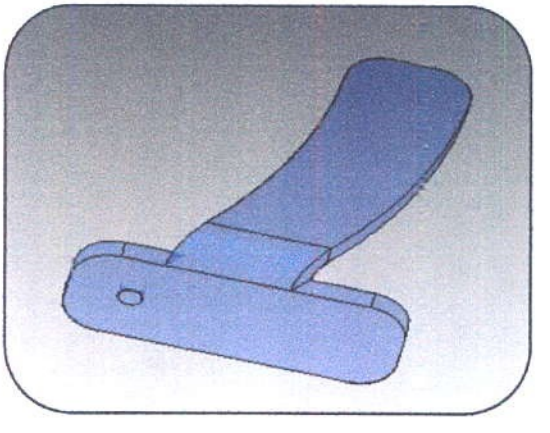
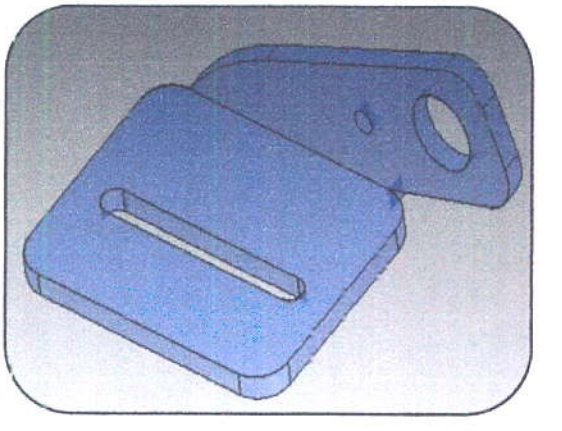
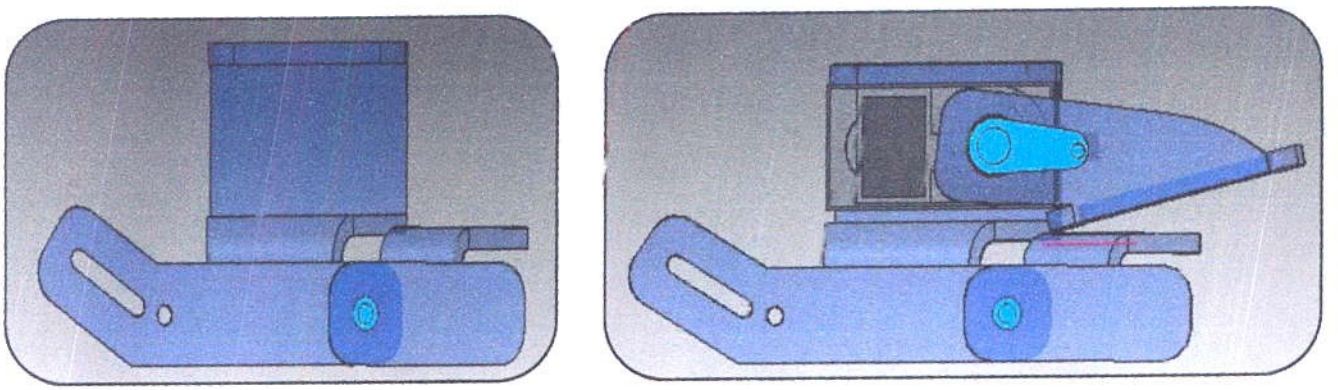


Fig 63, PIEZA P_FMIV



- El servomotor se integra a las bases de acrílico que dan protección.



- La palanca del servomotor se desplaza en forma circular, generando un movimiento angular de la pieza distal.

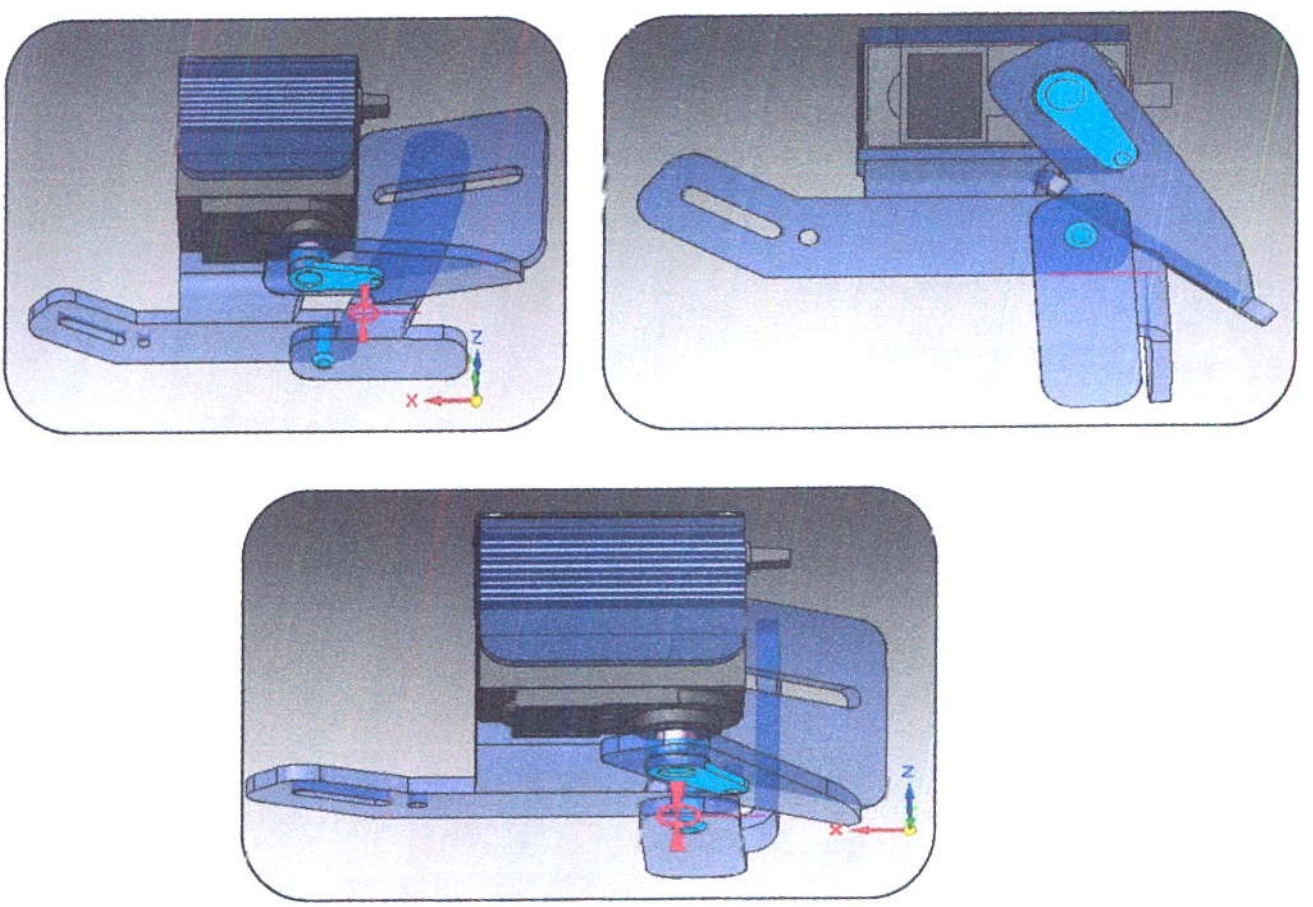


Figura 64. Montaje Componentes Articulación IFP II y III dedos

3.6 DISEÑO DEL MOVIMIENTO ARTICULACION METACARPOFALANGICA IV Y V DEDOS

Fig 65. PIEZA BS6_MIV

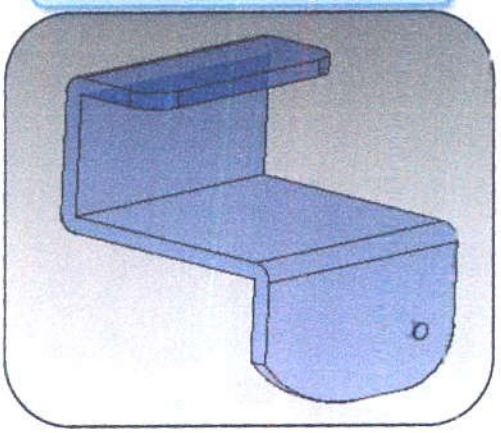


Fig 66. SERVOMOTOR

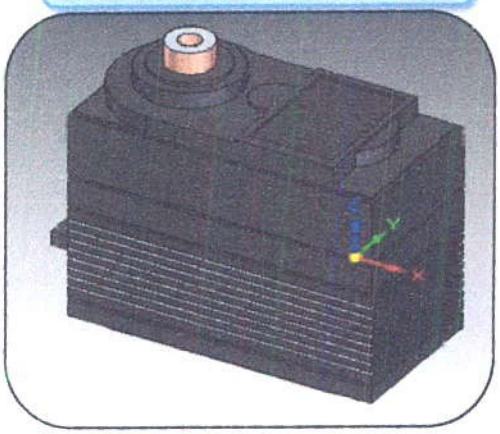


Fig 67. PALANCA SERVO

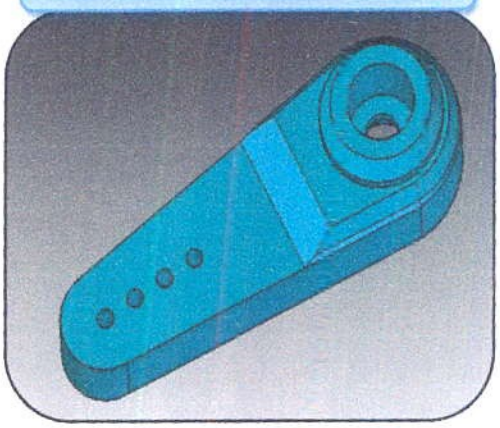


Fig 68. ESLABON

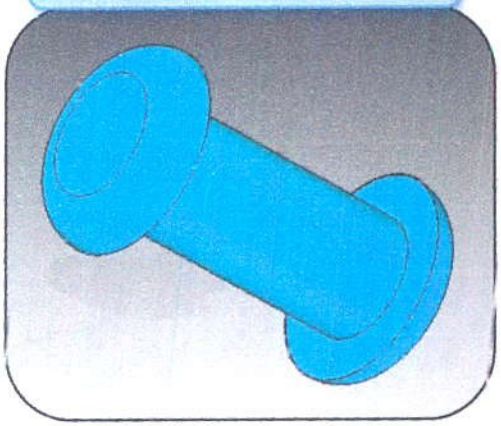
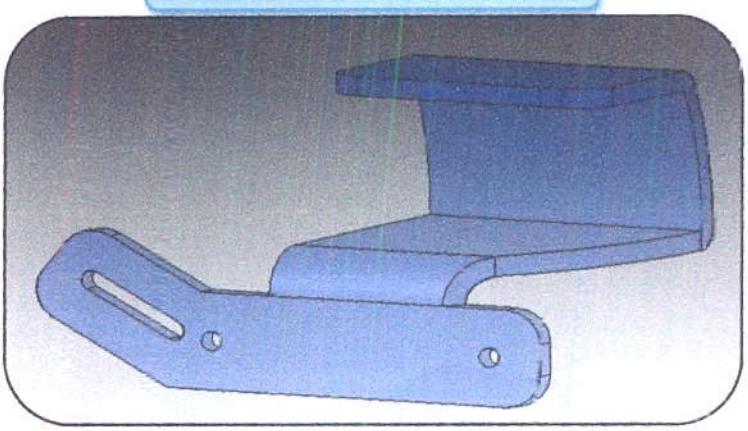
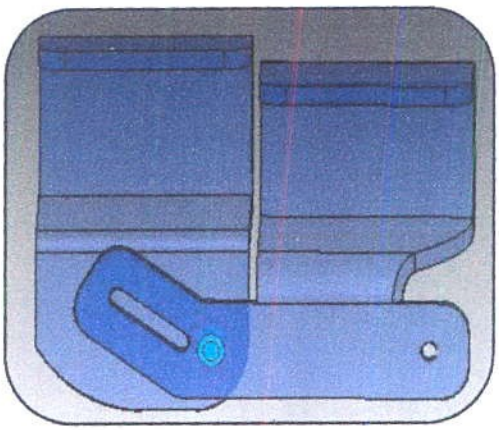
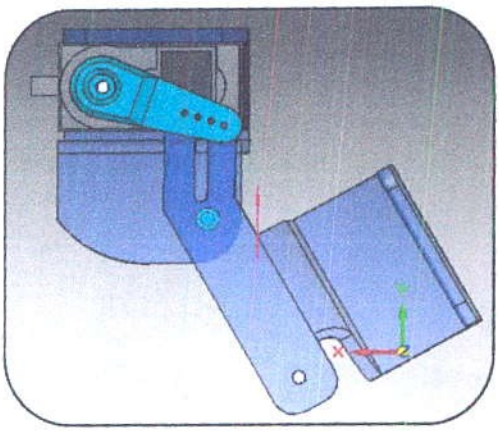


Fig 69. PIEZA PS5_FPIV



- El servomotor se integra a las bases de acrílico que dan protección.





- La palanca del servomotor se desplaza en forma circular, generando un movimiento angular de la pieza distal.

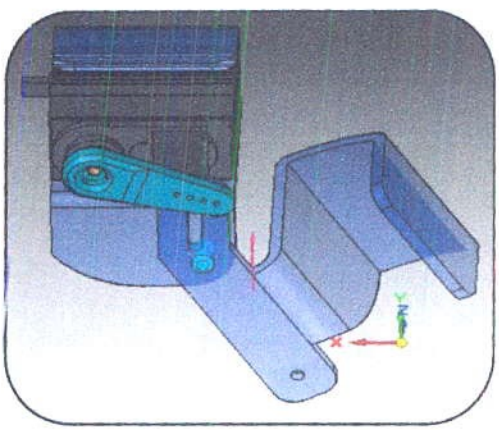


Figura 70. Montaje Componentes Articulación MF IV y V dedos

3.7 DISEÑO DEL MOVIMIENTO ARTICULACION INTERFALANGICA PROXIMAL IV Y V DEDOS

Fig 71. PIEZA PS7_FPII

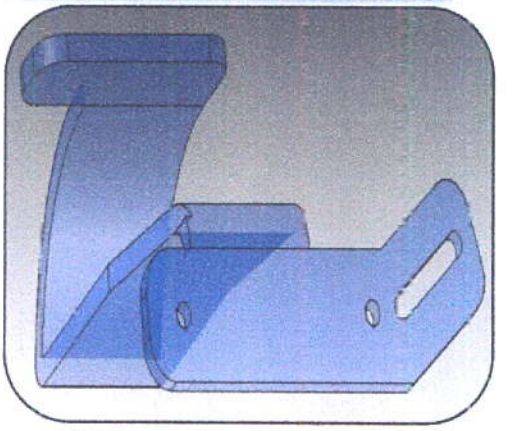


Fig 72. SERVOMOTOR

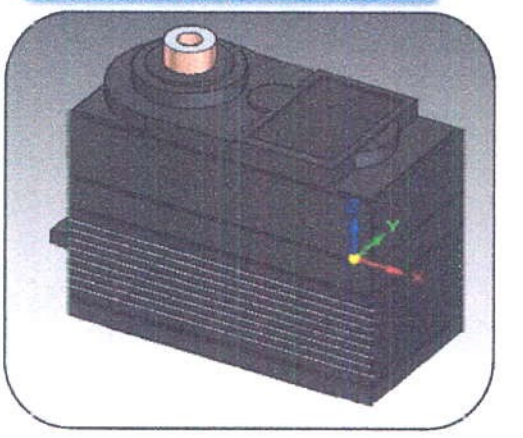


Fig 73. PALANCA SERVO

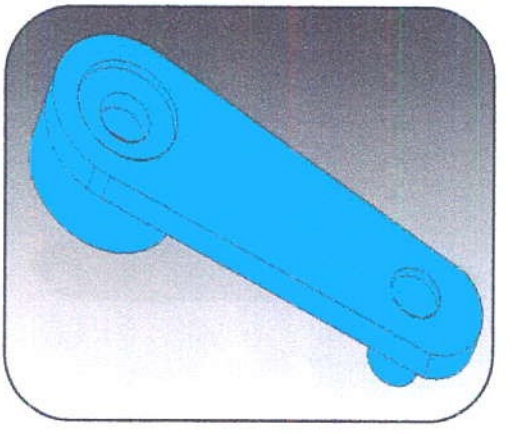


Fig 74. ESLABON

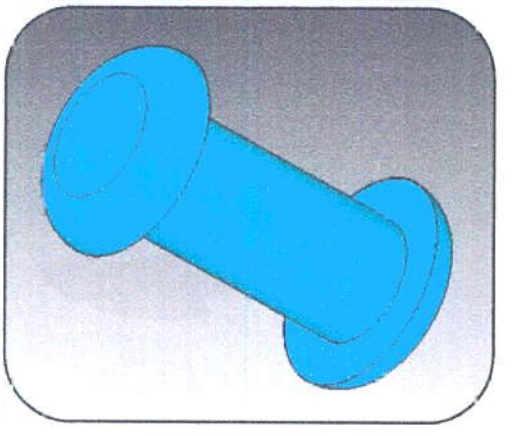


Fig 75. PIEZA B_FMII

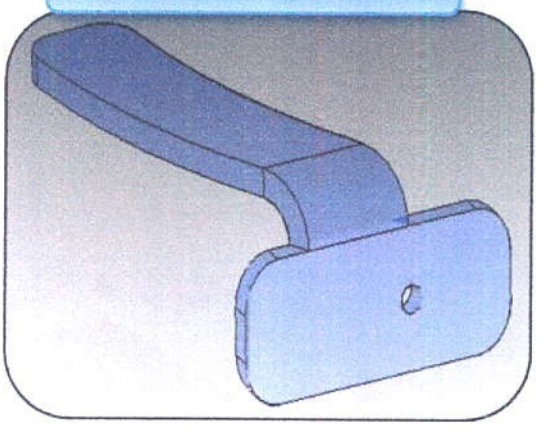
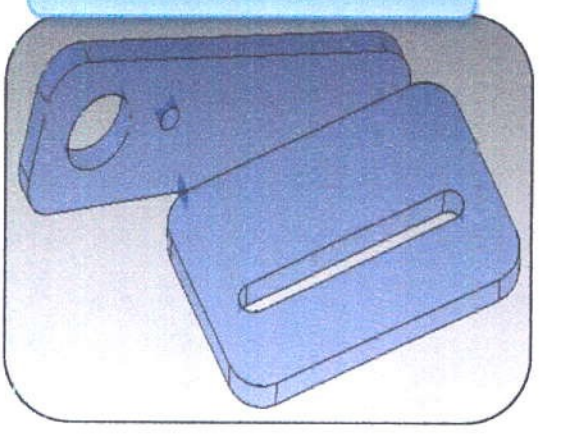
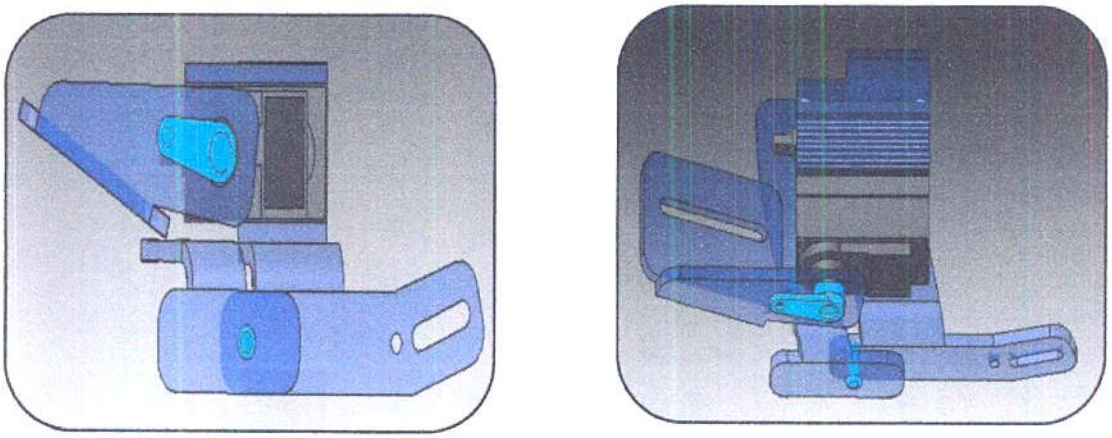


Fig 76. PIEZA P_FMII



- El servomotor se integra a las bases de acrílico que dan protección.



- La palanca del servomotor se desplaza en forma circular, generando un movimiento angular de la pieza distal.

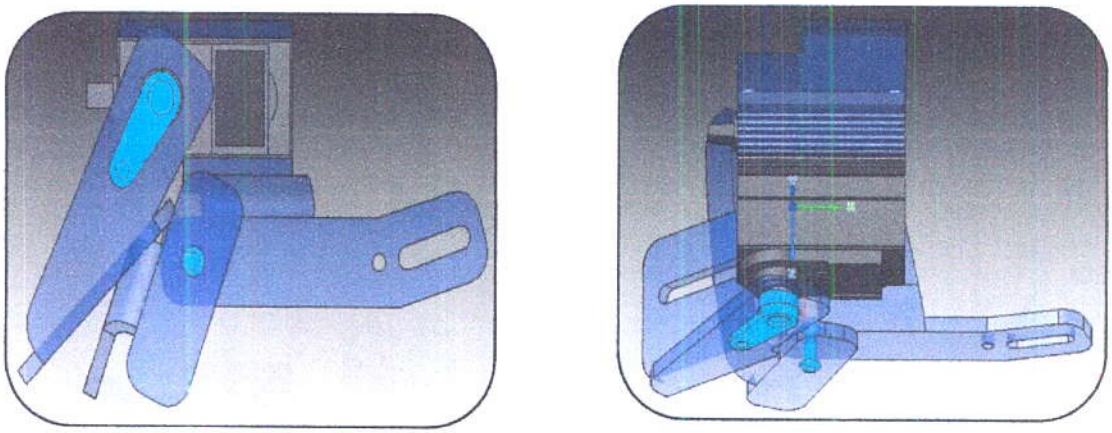


Figura 77. Montaje Componentes Articulación IFP IV y V dedos

4. IMPLEMENTACIÓN SENSORES E INDICADORES:

4.1 SENSORES DE FUERZA RESISTIVOS

El sensor de fuerza resistivo (**FSR**) es un dispositivo de película de polímero (PTF) que presenta una disminución de la resistencia cuando aumenta la fuerza aplicada a la superficie activa. Su sensibilidad a la fuerza está optimizada para uso en el control por toque humano de dispositivos electrónicos. Las FSRs no son células de carga o galgas extensiométricas aunque tengan propiedades similares. Las FSRs no son acecuadas para medidas de precisión.

Para nuestro proyecto utilizamos Modelo FSR 402, con un área activa circular de12.7mm de diámetro, espesor nominal de 0,46mm, capa semiconductora, espacio adhesivo de acrílico, capa conductiva.

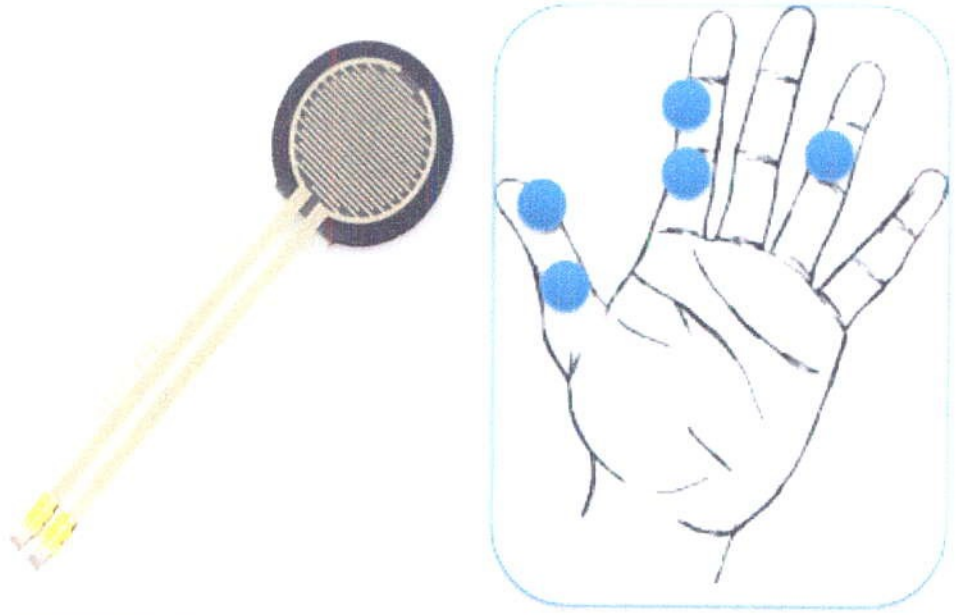


Figura 78. FSR y su ubicación en el guante de cada mano

En cada mano se ubicaron cinco (5) **FSR** en las zonas de contacto digital con objetos:

- 1 FSR en la cara palmar de la I falange del I dedo.
- 2 FSR en la cara palmar de la II falante del I dedo.
- 3 FSR en la cara palmar de la I falange del II dedo.
- 4 FSR en la cara palmar de la II falange del II dedo.
- 5 FSR en la cara palmar de la II falange del IV dedo.

4,2. INDICADORES LED

El módulo de reconocimiento de voz, a través de un sonido indica al usuario en que momento pronunciar un comando de voz, para que el exoesqueleto realice la una acción como abrir, cerrar, agarrar, soltar, etc. Sin embargo, hay un tiempo de milisegundos en el que el EasyVR shield 3.0 no reconoce ninguna instrucción, con el indicador Led de alta luminosidad de color verde, se muestra al usuario el tiempo en que puede pronunciar un comando de voz, cuando se termina este tiempo, se enciende el Led de alta luminosidad de color rojo, durante este tiempo, el módulo está ejecutando el ciclo de programa y no reconocerá ningún sonido pronunciado por el usuario.

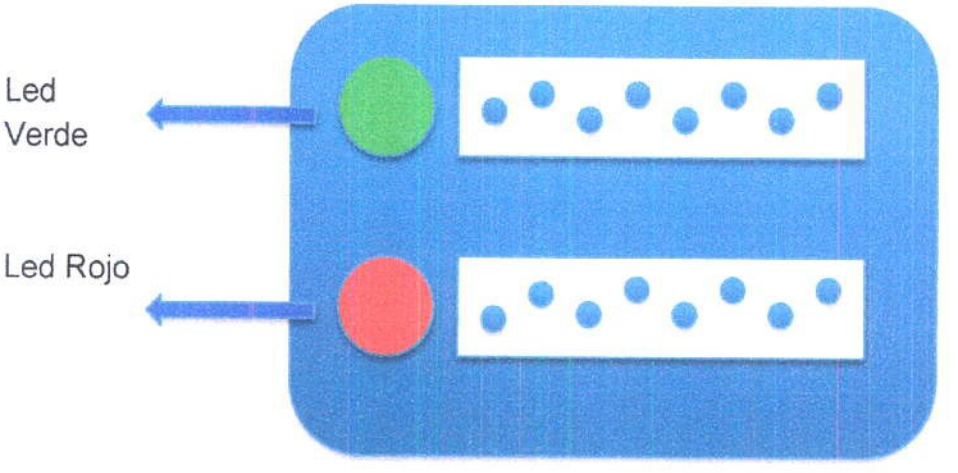


Figura 79. Base de indicadores led en la muñeca

Led Verde: Tiempo en el que el usuario puede pronunciar los comandoz de voz pregrabados en el módulo de reconocimiento de voz EasyVR Shield 3.0

Led Rojo: Tiempo en el que el usuario el módulo de reconocimiento de voz está procesando la información o indicando instrucciones al usuario, las palabras del usuario no serán reconocidas por el sistema.

4.3 SENSORES DE SEÑAL MUSCULAR:

Los sensores de señal muscular se encuentran ubicados en el antebrazo del paciente, recogen la información de tres electrodos (Rojo, Azul y Negro). Mide la diferencia de presión entre los electrodos rojo y azul, los convierte en un voltaje y los envía al módulo de control electrónico (Tarjeta Arduino Mega 2650). El electrodo negro es la tierra del paciente al circuito electrónico.

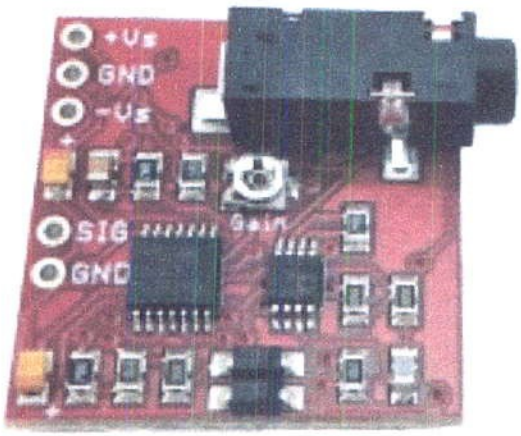


Figura 80. Sensor EMG. Tomado del Data Sheet del Fabricante

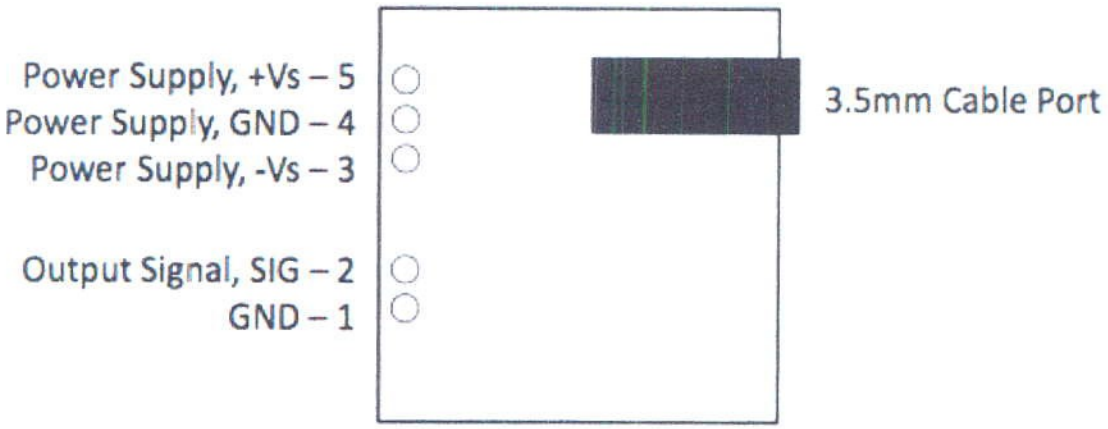


Figura 81. Esquema Sensor EMG. Tomado del Data Sheet del Fabricante

5. FUENTE DE ALIMENTACIÓN CONMUTADA

El Prototipo EMMRA-1 debe cumplir con las normas internacionales para fabricación de dispositivos biomédicos, para ello fue necesario adquirir las fuentes de alimentación conmutadas certificadas para uso médico (Switching Power Supply **SPS**), las cuales vienen fabricadas con certificación en seguridad eléctrica (IEC 60601-1) y compatibilidad electromagnética (IEC 6060-1-2).

Para alimentar los servomotores de cada mano, se implementó la SPS MPT200A en cada lado de la silla de descanso del paciente discapacitado. Esta fuente adquiere la energía eléctrica orriente Alterna de la red pública (corriente alterna) y proporciona energía al dispositivo (corriente continua). Presenta protecciones contra sobrevoltajes, cortocircuitos y cambios de tensión, entrega 4 salidas de 5 Voltios con 20 Amperios en total, una salida de 12 Voltios 7 Amperios, y una salida de -5 Voltios 1 Amperios.

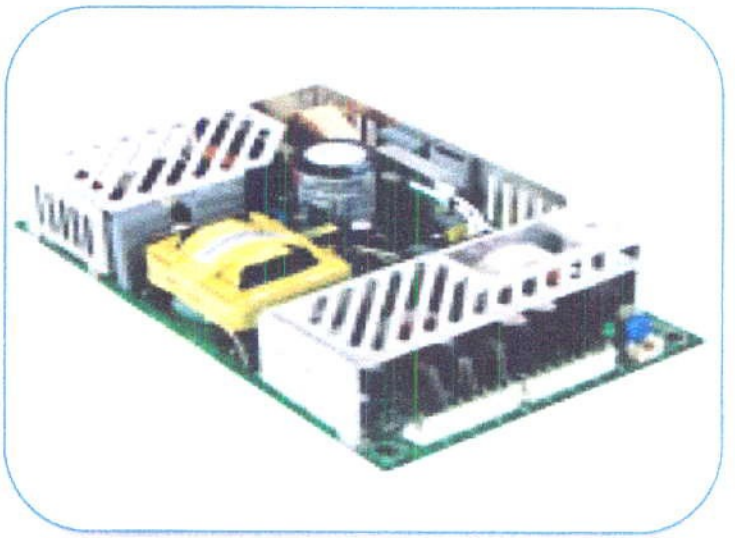


Figura 82. Fuente de alimentación MPT200A

INFORME FINAL DEL PROYECTO
PROTOTIPO EXOESQUELETO DE MANO Y MUÑECA CON APLICACIONES DE REHABILITACION Y ASISTENCIA (PROTOTIPO EMMRA-I)

CAPITULO II
FABRICACIÓN DEL EXOESQUELETO MECATRONICO DE MANO Y MUÑECA

INVESTIGADORES:

JUAN DAVID MORENO ARANGO
JULIAN ALBERTO MORENO ARANGO

ASESORES:

VICTOR MAURICIO ACEVEDO CORREA
ALEXANDER RAMOS RODRIGUEZ

COLABORADORES

MARIA ISABEL RIVERA ARANGO
JAIRO MARTINEZ MARIN

Manizales, 01 de Noviembre de 2015

INTRODUCCIÓN

El proyecto: Prototipo exoesqueleto de mano y muñeca con aplicaciones de rehabilitación y asistencia (**Prototipo EMMRA-1**), hace parte de los proyectos financiados por la convocatoria 642 de **COLCIENCIAS** "La locomotora de la innovación para el apoyo del desarrollo tecnológico", está orientado a beneficiar los procesos de asistencia y rehabilitación a la discapacidad motríz en la mano y la muñeca de los pacientes con secuelas por lesiones neurológicas.

El exoesqueleto mecatrónico es un producto original, que implica la aplicación de la ingeniería electrónica, biomédica, mecánica y de sistemas, para generar innovación tecnológica en el área de la salud. Concientes de la necesidad de que el resultado final de la investigación se traduzca en un beneficio para la humanidad, realizamos el presente documento para describir con detalle los procesos de diseño y fabricación del Prototipo EMMRA-1, con el propósito de registrar la patente nacional e internacional en Colombia.

El Prototipo EMMRA-1 está compuesto por las siguientes Interfaces: Órtesis - Piel, Órtesis Pasiva y Órtesis Activa. La Interface Órtesis- Piel está diseñada para ser fácilmente aplicada al paciente discapacitado, fabricada en cuero textil, neopreno, hilo nylon y velcro. La Órtesis Pasiva está diseñada para permitir a cada mano siete Grados de Libertad (**GDL**), con articulaciones elaboradas de acrílico 3mm para los servomotores y ejes con remaches doble cabeza para evitar la fricción. La Órtesis Activa contiene Servomotores Hitec HS-7955TG, proporcionan un torque de 20Kg/cm y un movimiento de 90 grados programables, para evitar sobrepasar los límites anatómicos de las articulaciones de la mano.

Cada mano cuenta con cinco Sensores de Fuerza Resistivos (**FSR**) ubicados en las áreas de contacto digital en la palma de la mano, regulan la intensidad de la fuerza aplicada por el exoesqueleto. En cada antebrazo se ubican sensores de señal muscular con electrodos de electromicgrafía, para el proceso de rehabilitación. Contiene indicadores Led de alta luminosidad de color verde y rojo, para informar al paciente cuando puede dar instrucciones al exoesqueleto por comandos de voz o electromiografía.

El Módulo de control está conformado por un Módulo de Reconocimiento de Voz EasyVR Shield 3.0, integrado a una tarjeta Arduino Mega 2560, que procesa las señales de voz, electromiografía y FSR, para envíar señales de actuación a los servomotores.

TABLA DE CONTENIDO

1. **FABRICACIÓN INTERFACE MANO – ÓRTESIS** 04
1.1 Materiales y equipos 04
1.2 Preparación piezas de cuero textil 05
1.3 Preparación piezas de neopreno 12
1.4 Preparación piezas en acero inoxidable 14
1.4 Costura de piezas en cuero y neopreno 15
1.5 Fotos del producto terminado 19
2. **FABRICACIÓN INTERFACE ÓRTESIS PASIVA** 21
2.1 Protectores servomotor 1 21
2.2 Protectores servomotor 2 22
2.3 Protectores servomotor 3 23
2.4 Protectores servomotor 4 24
2.5 Protectores servomotor 5 25
2.6 Protectores servomotor 6 26
2.7 Protectores servomotor 7 27
2.8 Protectores servomotor 8 28
2.9 Fotos del producto terminado 30
3. **FABRICACIÓN INTERFACE ÓRTESIS ACTIVA** 33
3.1 Diseño movimiento I metacarpiano 33
3.2 Diseño movimiento articulación metacarpofalangica I dedo 37
3.3 Diseño movimiento articulación interfalangica I dedo 39
3.4 Diseño movimiento articulaciónes metacarpofalangicas II y III dedos 41
3.5 Diseño movimiento articulación interfalangica proximal II y III dedos 43
3.6 Diseño movimiento articulaciónes metacarpofalangicas IV y V dedos 45
3.7 Diseño movimiento articulación interfalangica proximal IV y V dedos 47
3.8 Diseño movimiento muñeca 49
3.9 Secuencia de ensamblaje exoesqueleto 52
3.9 Fotos del producto terminado 59
4. **IMPLEMENTACION SENSORES E INDICADORES** 62
4.1 Sensores de Fuerza Resistivos 62
4.2 Indicadores LED de alta lumiosidad 63
4.3 Sensores de Señal Muscular 64
4.9 Fotos del producto terminado 65
5. Fuente de Alimentación Conmutada MPT200A 66

1. FABRICACIÓN INTERFACE MANO – ÓRTESIS:

Se fabrica un guante de cuero textil y neopreno, para brindar múltiples beneficios al paciente con discapacidad motriza de mano y muñeca:

- Protección de piel antiescara en superficies de contacto con componentes metálicos o plásticos del exoesqueleto.
- Protección de piel antitermica, regulando la temperatura generada por el calentamiento de los servomotores.
- Facilidad en colocarse y retirarse de la mano de un paciente con discapacidad de mano y muñeca.

1.1 MATERIALES Y EQUIPOS:

EQUIPOS:

- Máquina de coser cuero textil.
- Fileteadora con puntada de seguridad.
- Máquina de corte laser de acrílico.
- Máquina Remachadora.
- Resistencia calefacción lineal.

MATERIALES:

- Cuero textil.
- Ferulas de Neopreno.
- Velcro.
- Hilo Nylon
- Hilasa Negra
- Aguja 3,5 cm.
- Tijeras.
- Encendedor.
- Chequeador de hilo.
- Cinta doble fax transparente
- Software Solide Edge
- Software Autocad
- Diodos Led Alta Luminosidad Verde y Azul
- Conectores GP
- Cables Vehicular 18F
- Cable ribon de colores

1.2 PREPARACIÓN DE PIEZAS EN CUERO TEXTIL:

1.2.1 Protectores de la base del I dedo (Cuero textil)

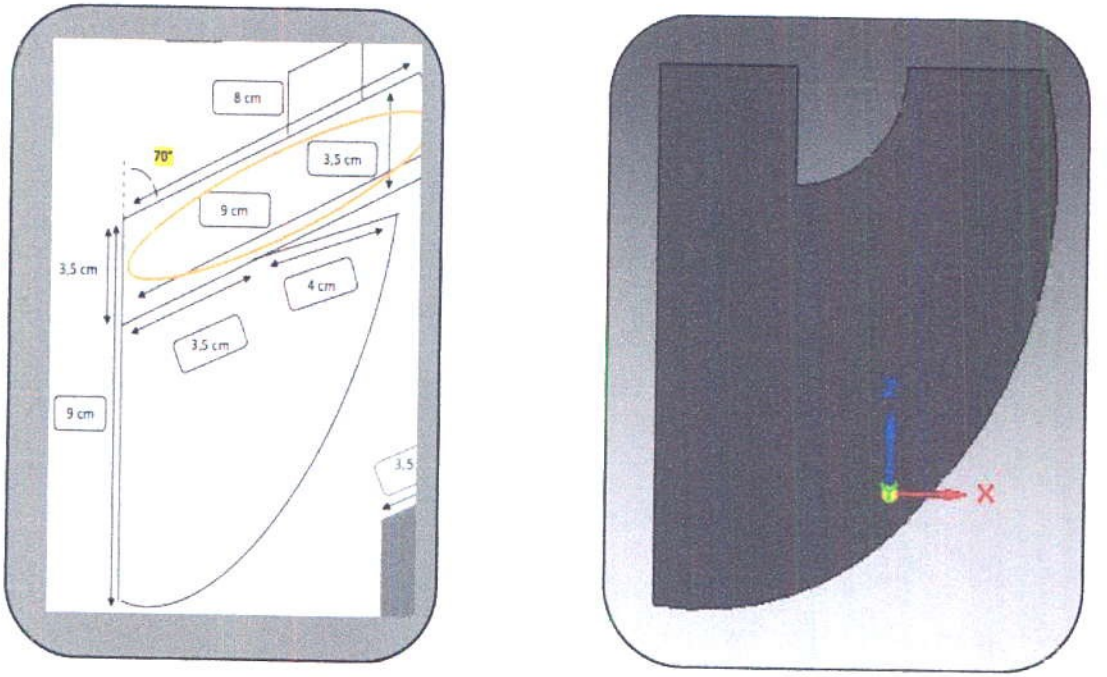


Figura 1. Protector base I dedo cara palmar mano derecha

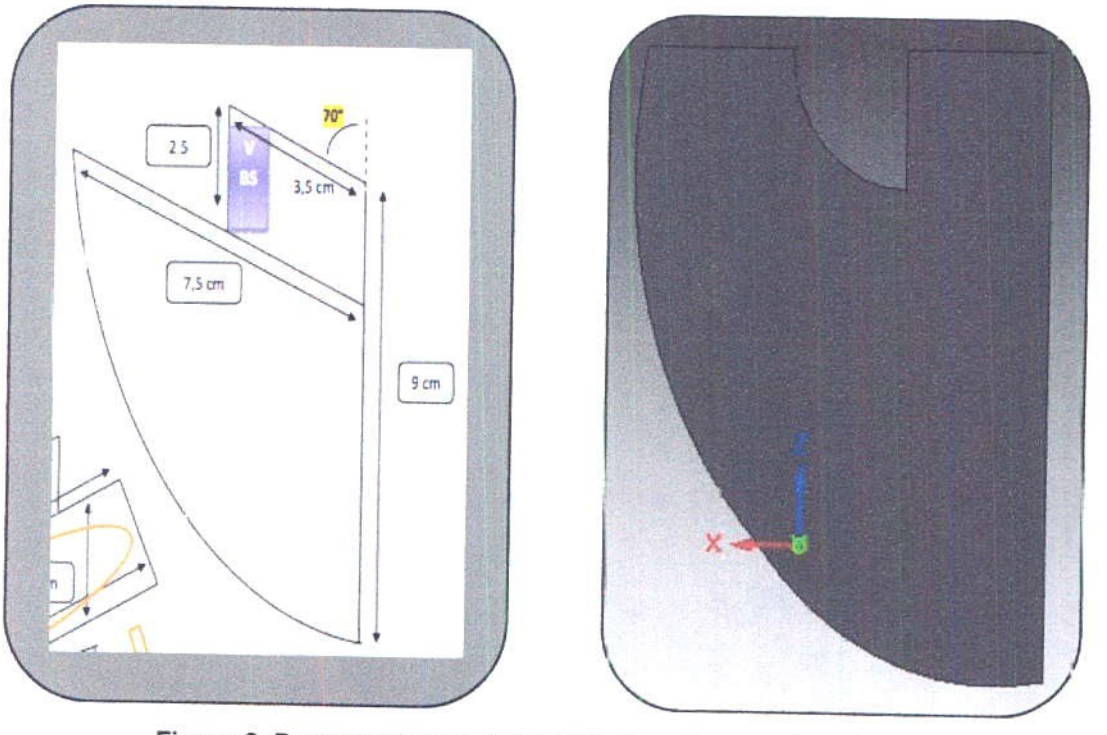


Figura 2. Protector base I dedo cara dorsal mano derecha

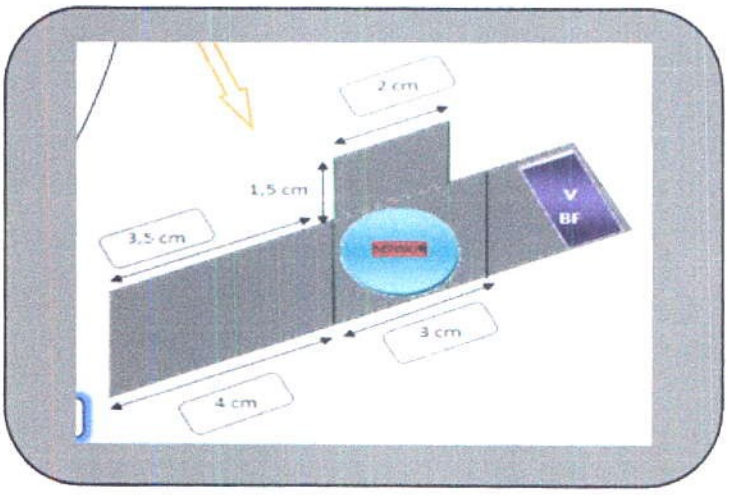


Figura 3. Protector falange proximal de I dedo mano derecha

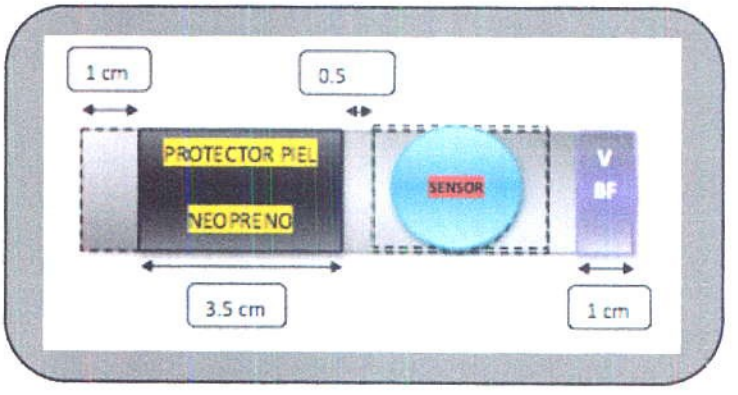


Figura 4. Protector falange distal I dedo mano derecha

1.2.2 Protector cara interna de la muñeca (Cuero textil)

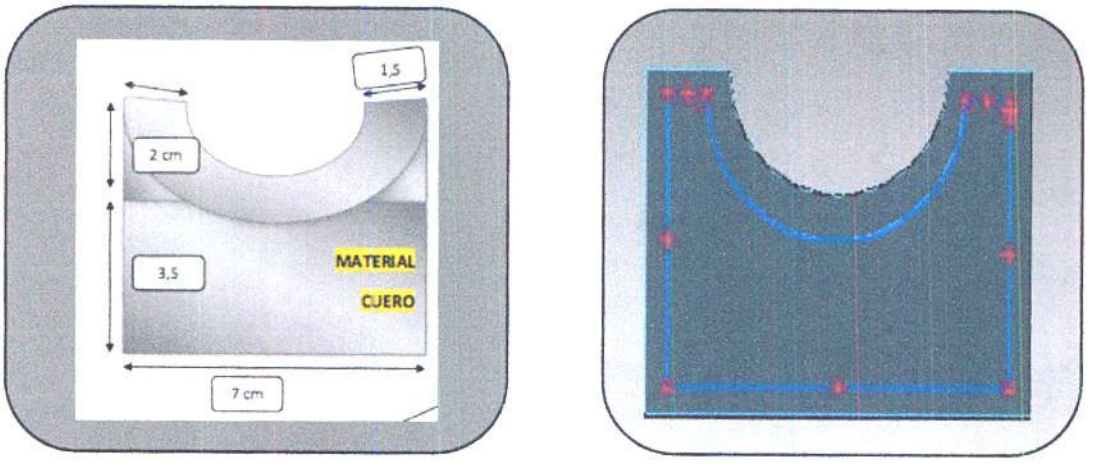


Figura 5. Protector base interna muñeca derecha

1.2.3 Protectores palma de la mano (Cuero textil)

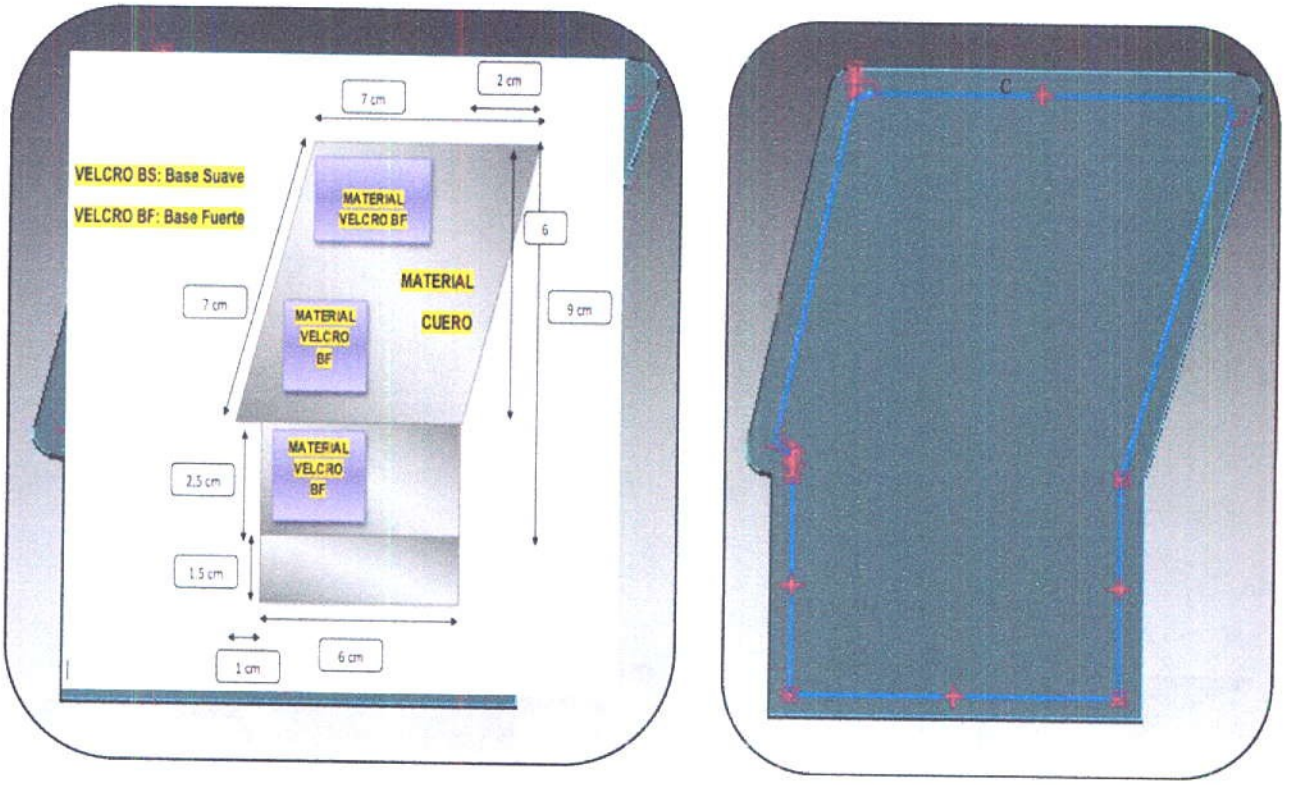


Figura 6. Protector cara palmar mano derecha

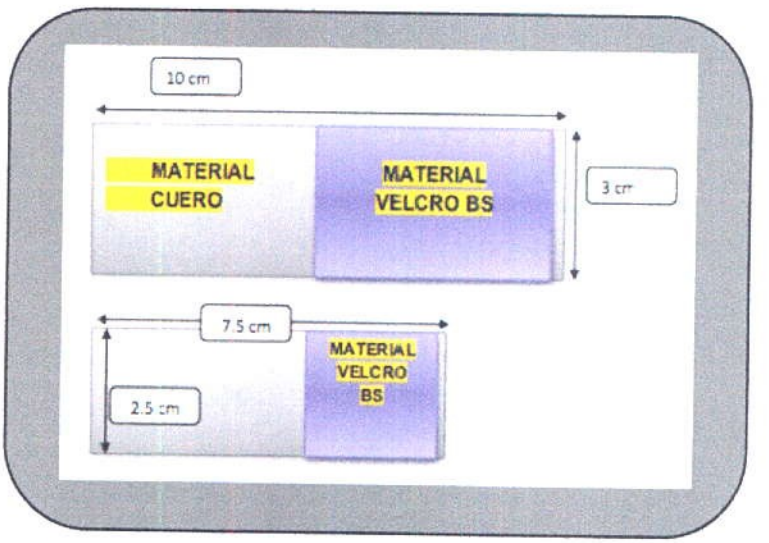


Figura 7. Correas de cuero y velcro ajuste palma mano derecha

1.2.4 Protectores II dedo mano derecha.

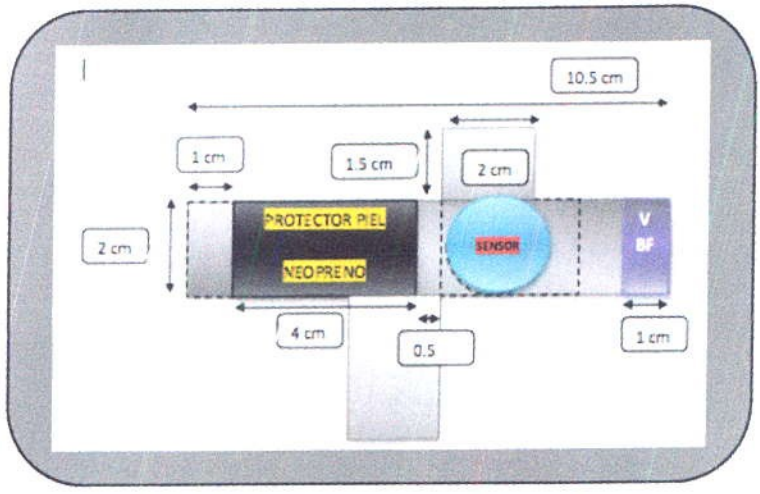


Figura 8. Protector falange proximal II dedo mano derecha

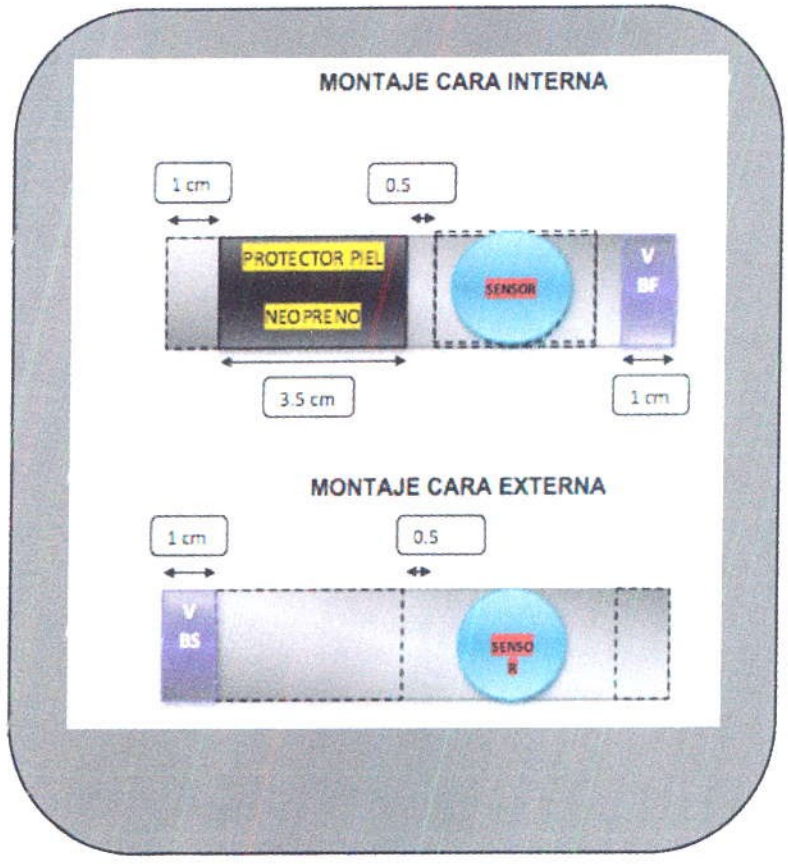


Figura 9. Protector falange medial II dedo mano derecha

1.2.5 Protectores III dedo mano derecha.

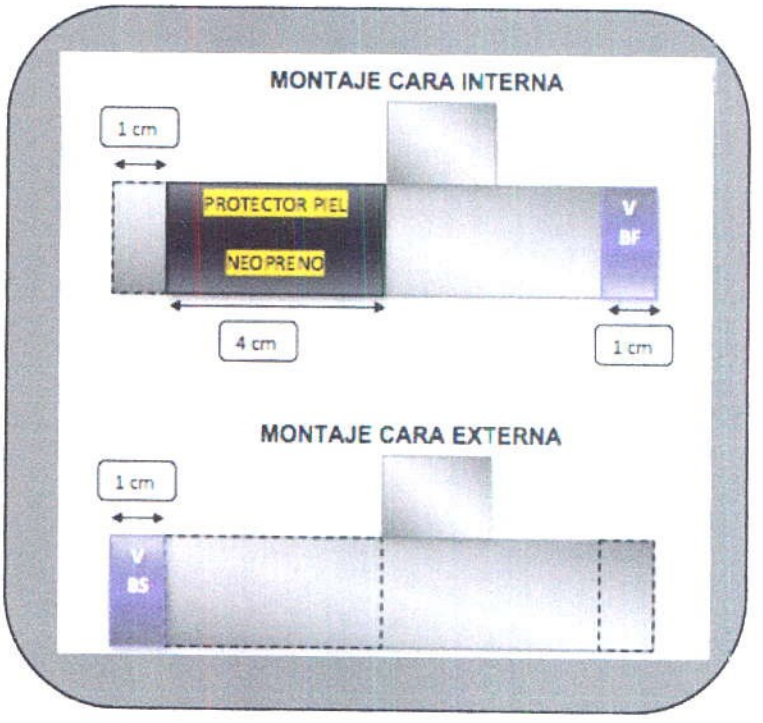


Figura 10. Protector falange proximal III dedo mano derecha

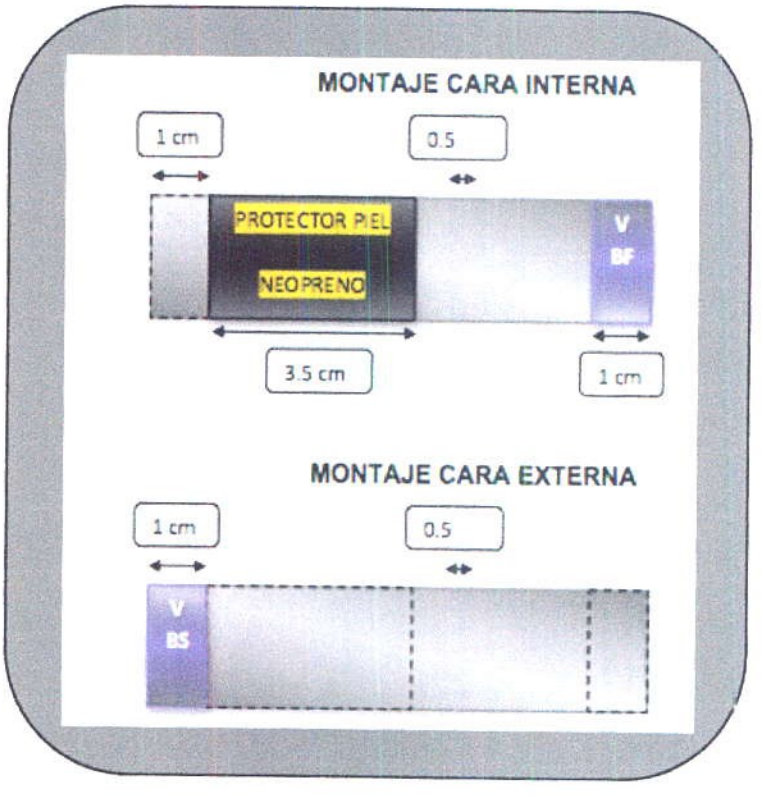


Figura 11. Protector falange medial III dedo mano derecha

1.2.6 Protectores IV dedo mano derecha.

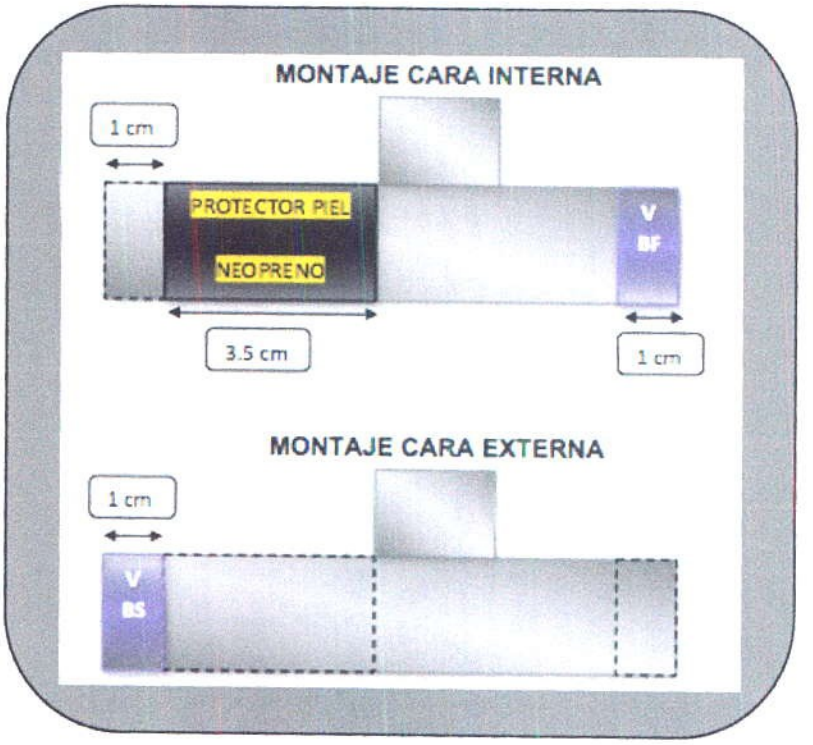


Figura 12. Protector falange proximal IV dedo mano derecha

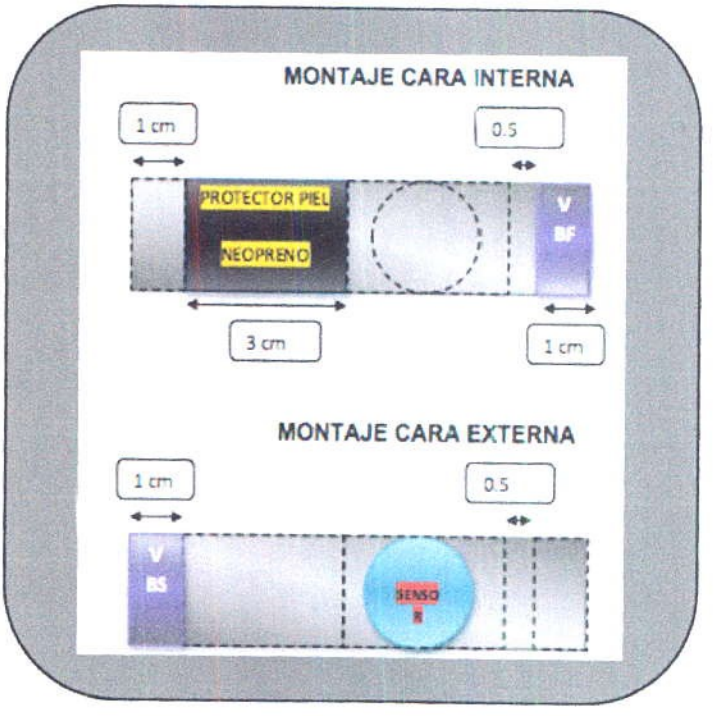


Figura 13. Protector falange medial IV dedo mano derecha

1.2.7 Protectores V dedo mano derecha.

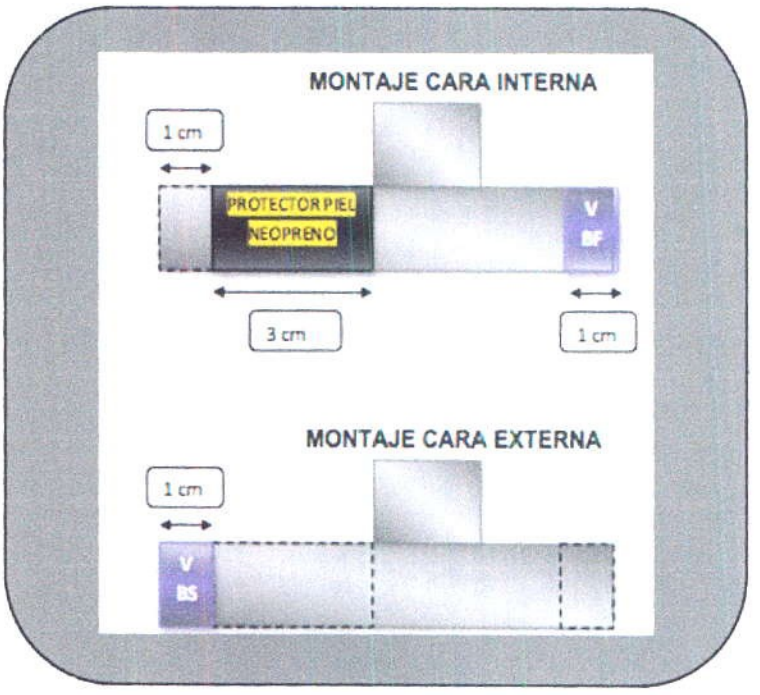


Figura 14. Protector falange proximal V dedo mano derecha

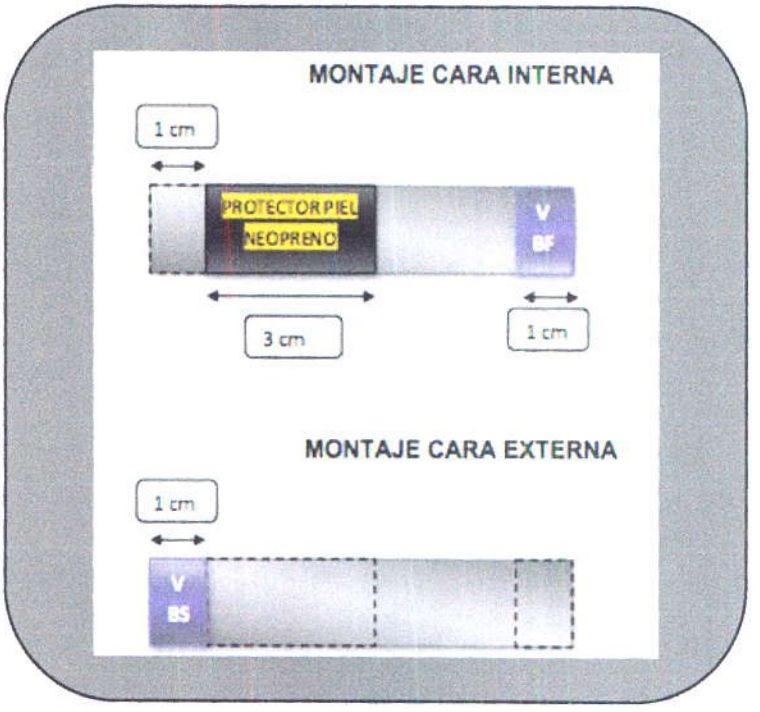


Figura 15. Protector falange medial V dedo mano derecha

1.3 PREPARACIÓN DE PIEZAS EN NEOPRENO:

1.3.1 Protector en neopreno cara dorsal mano derecha.

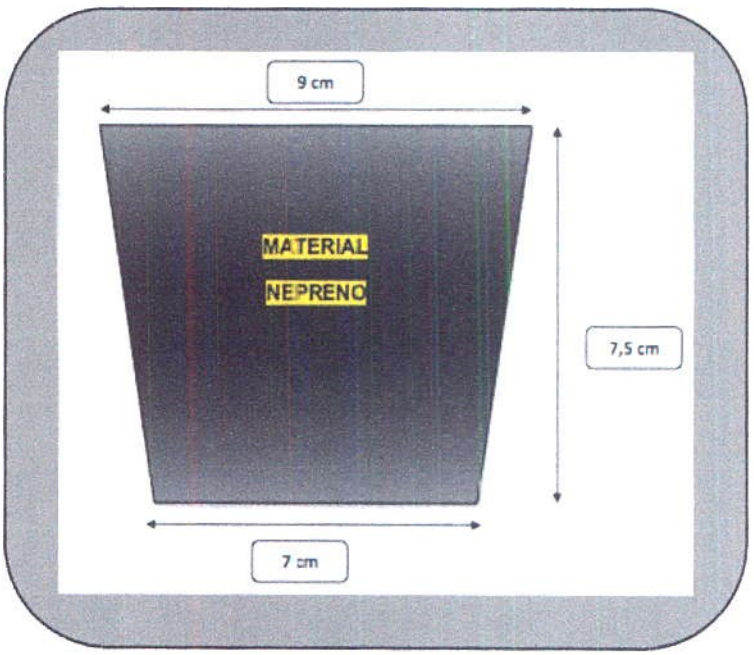


Figura 16. Protector cara dorsal mano derecha

1.3.2 Protector en neopreno cara interna de muñeca derecha.

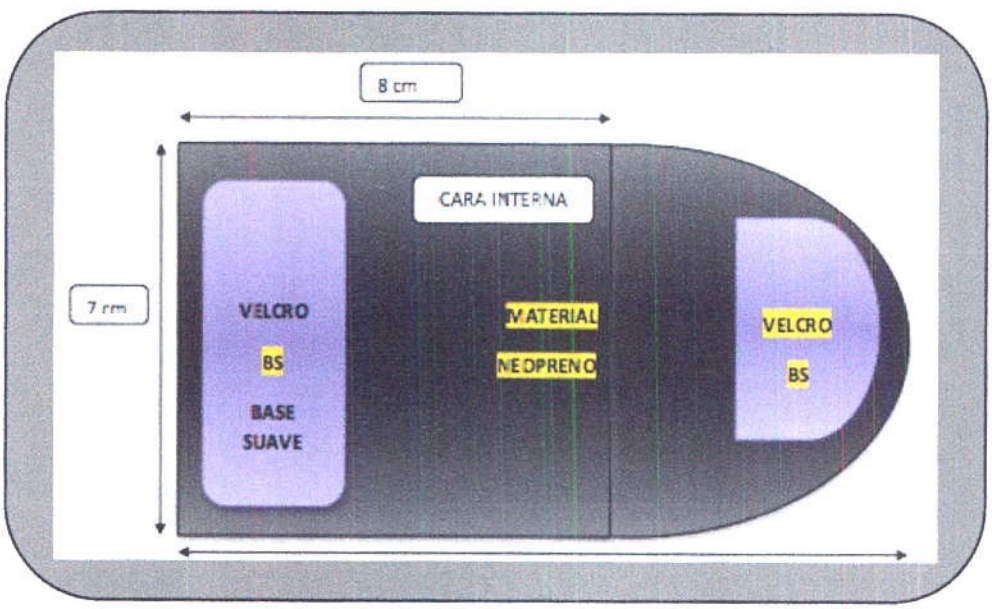


Figura 17. Protector cara interna muñeca derecha

1.3.3 Protector en neopreno cara externa de muñeca derecha.

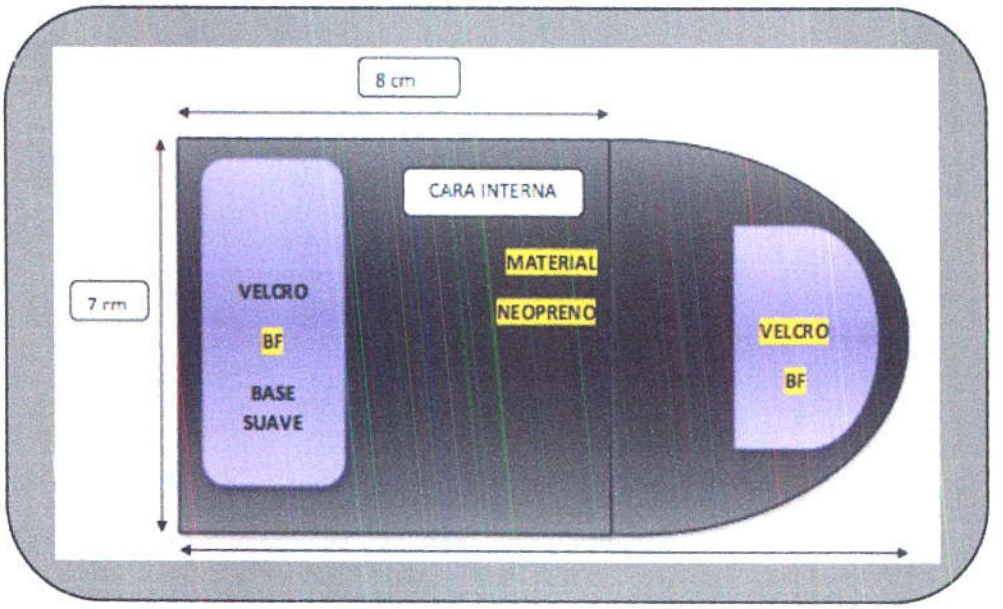


Figura 18. Protector cara externa muñeca derecha

1.4 PREPARACION DE PIEZAS EN ACERO INOXIDABLE

1.4.1 Piezas en acero inoxidable para el soporte de la muñeca a los servomotores:

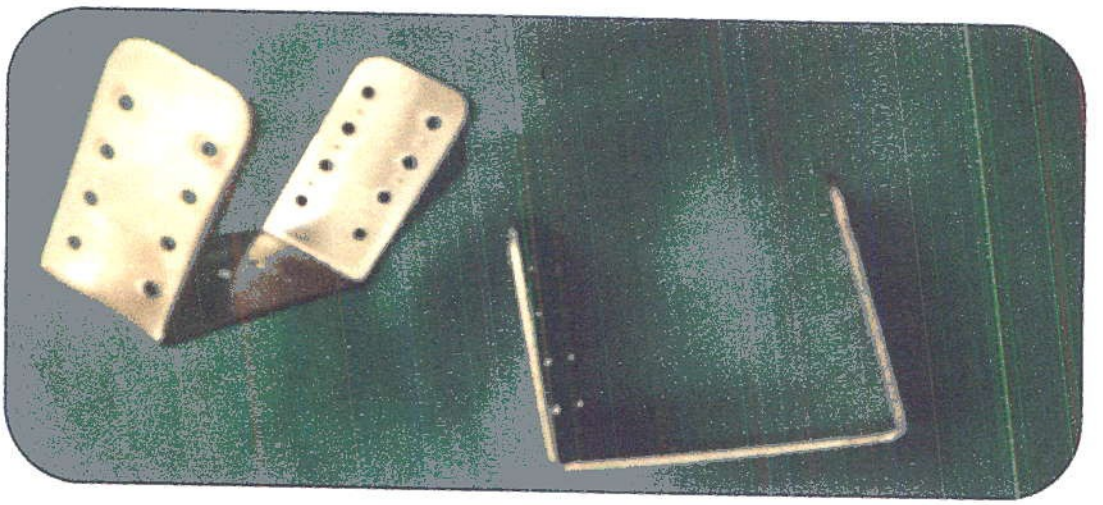


Figura 19. Piezas en acero inoxidable para protección de muñeca derecha

1.4.2 Piezas en acero inoxidable cubiertas en cuero:

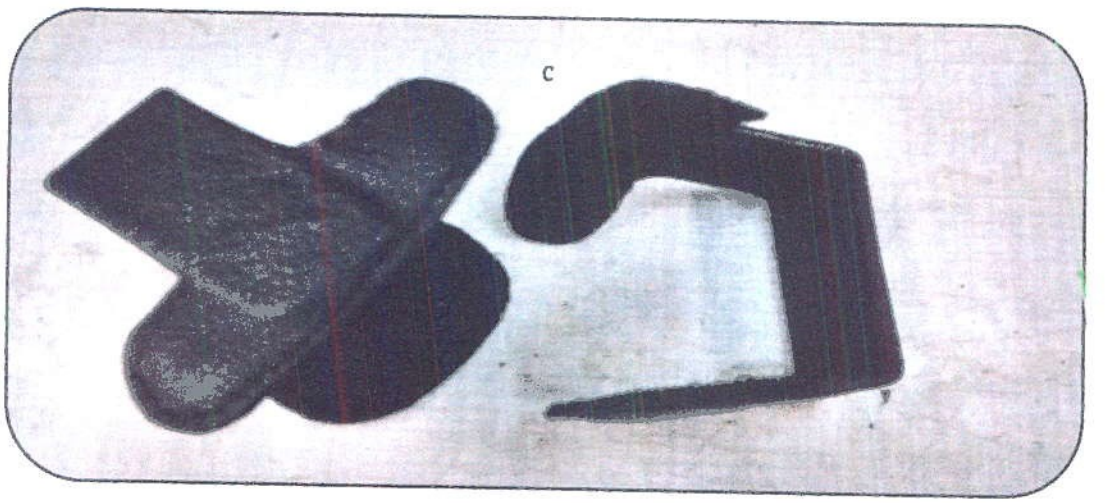


Figura 20. Piezas en acero inoxidable cubiertas en cuero textil

1.4 COSTURA DE PIEZAS EN CUERO Y NEOPRENO:

La costura de todas las piezas de cuero y neopreno, se realiza en una **máquina industrial para coser cuero textil**, utilizando hilo nylon, como se describe a continuación gráficamente:

- Costura de todos los bordes de las piezas de cuero a 5 mm de los extremos distales, para proteger los bordes.
- Costura de todos los bordes de las piezas de neopreno con **fileteadora industrial con punto de seguridad**, para proteger los bordes.
- Costura de los protectores dorsal y palmar del I dedo, se unen en cara lineal y se pasan dos lineas de costura a 5 mm y 10 mm del extremo de unión.

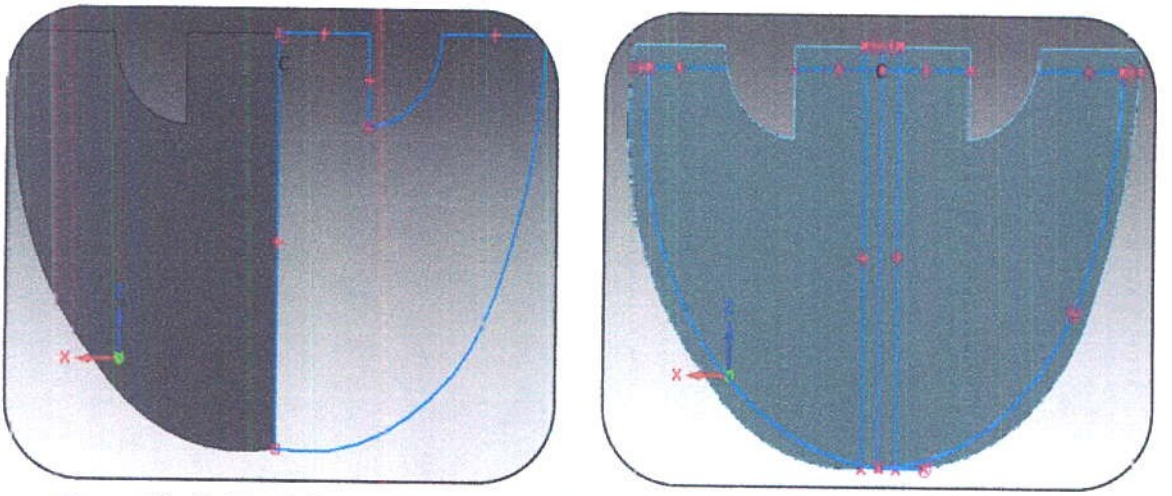


Figura 21. Unión de los protectores ventral y dorsal del I dedo mano derecha

- Costura del protector dorsal de neopreno, el protector palmar de cuero y las correas de cuero para asegurar el guante a nivel palmar.

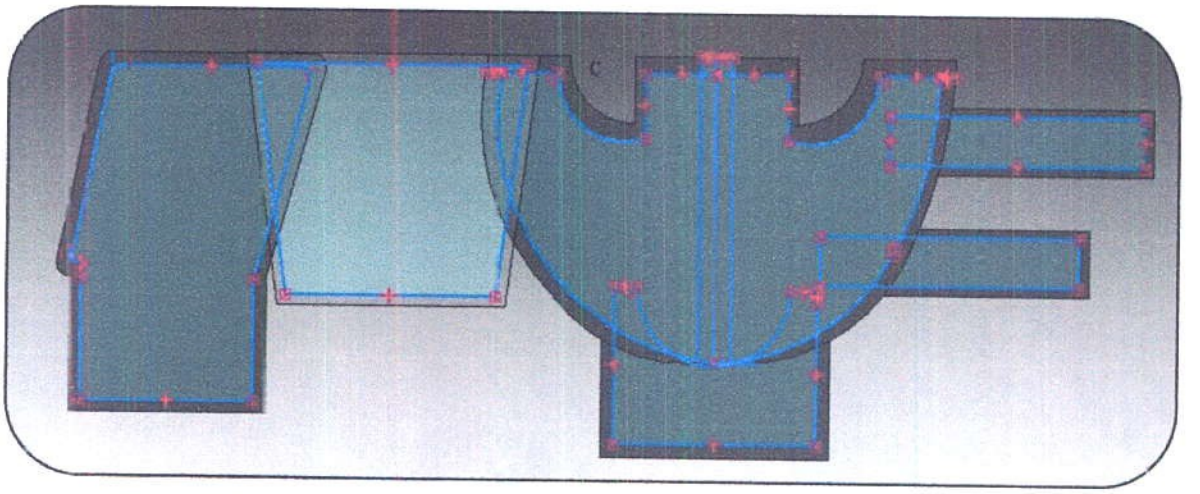


Figura 22. Unión de protectores de dorso y palma de mano derecha.

- Costura de los protectores en neopreno de lados interno y externo de muñeca derecha.

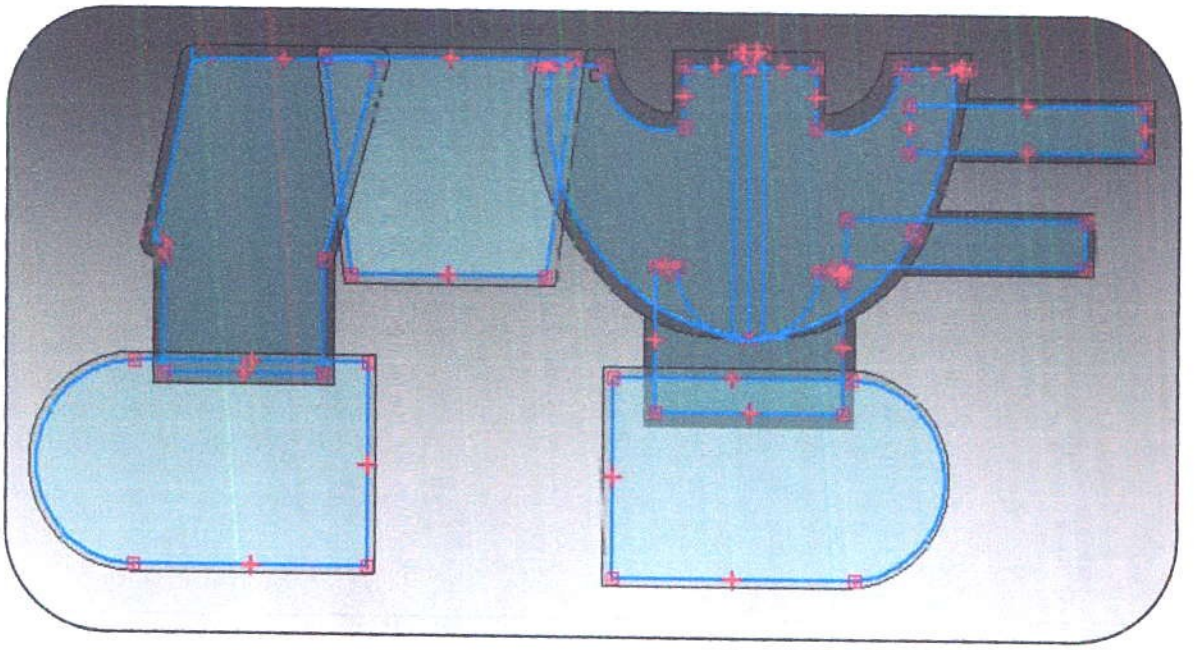


Figura 23. Unión de protectores de muñeca derecha.

- Costura de los protectores de los dedos, se realiza empezando por el V dedo hasta el II dedo.

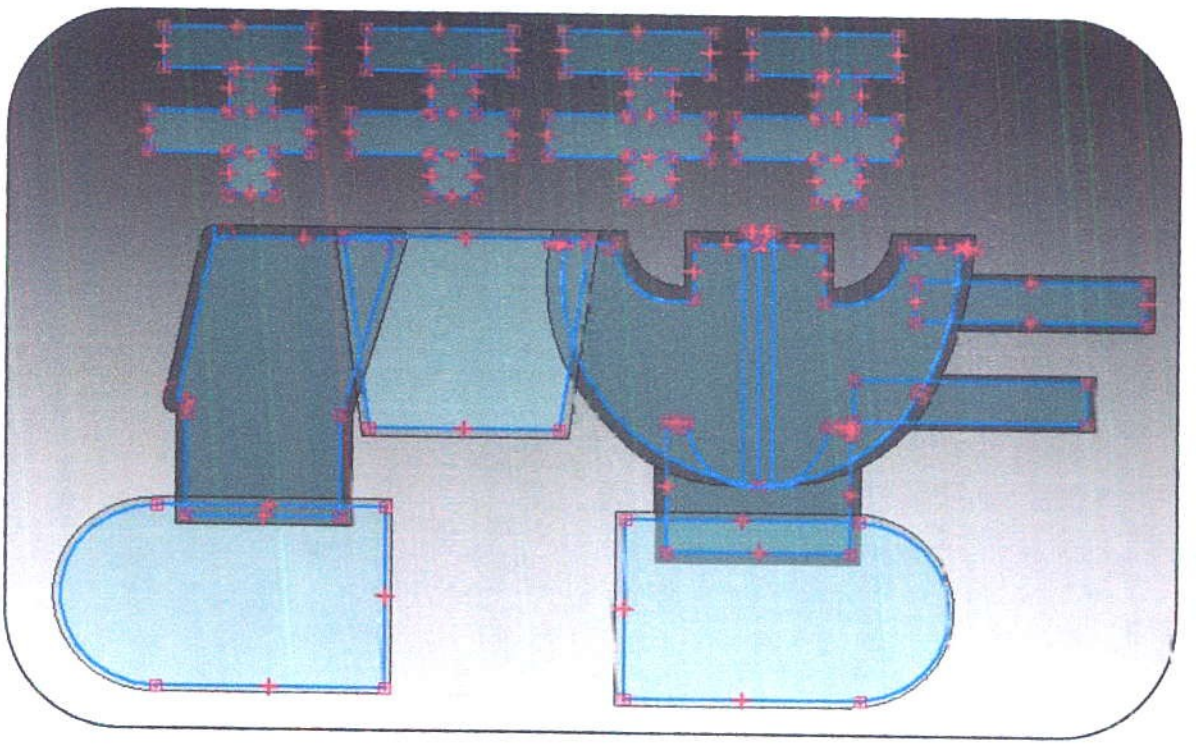


Figura 24. Unión de protectores de los dedos

- Costura del protector del V dedo de mano derecha, en la base del protector en neopreno de la cara dorsal de la mano derecha.

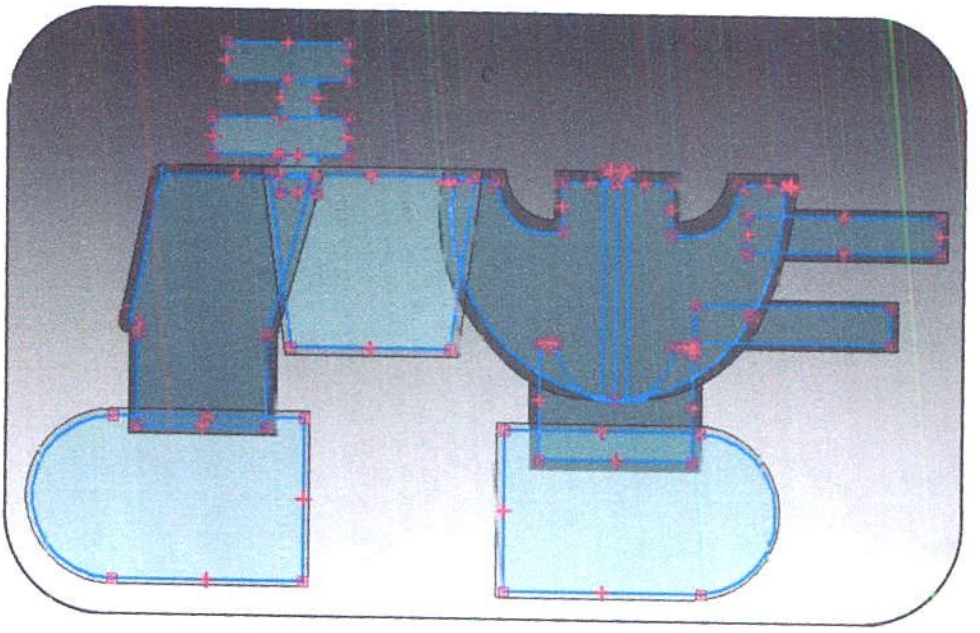


Figura 25. Unión de protectores del V dedo mano derecha

- El protector del V dedo se une en las superficies con velcro, para iniciar la costura de los otros protectores de los dedos.

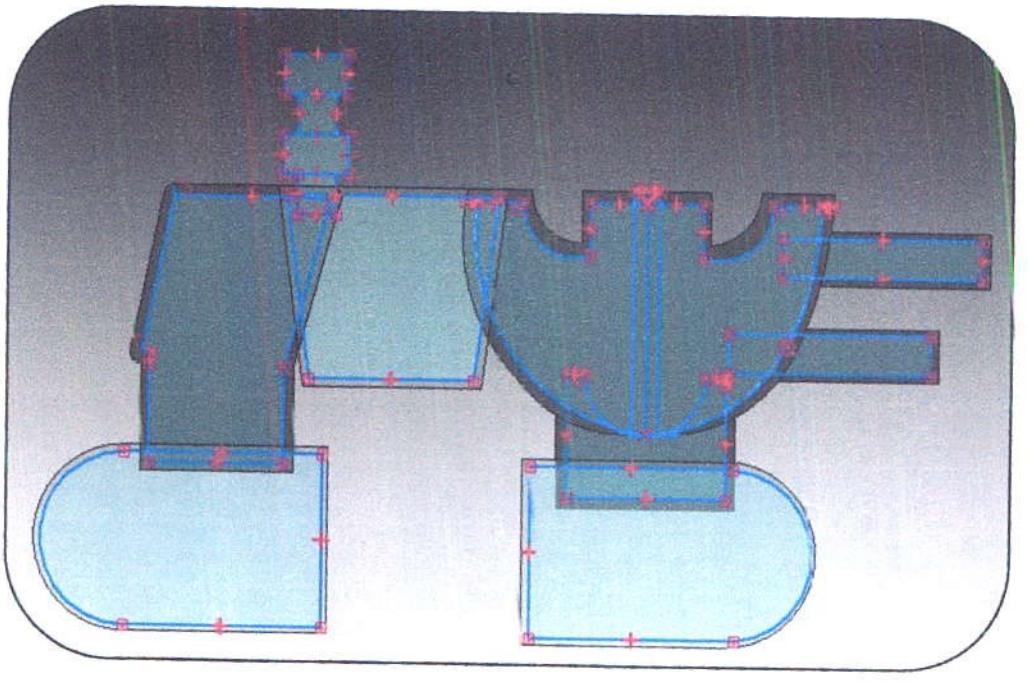


Figura 26. Unión y plegaje de protectores del V dedo mano derecha

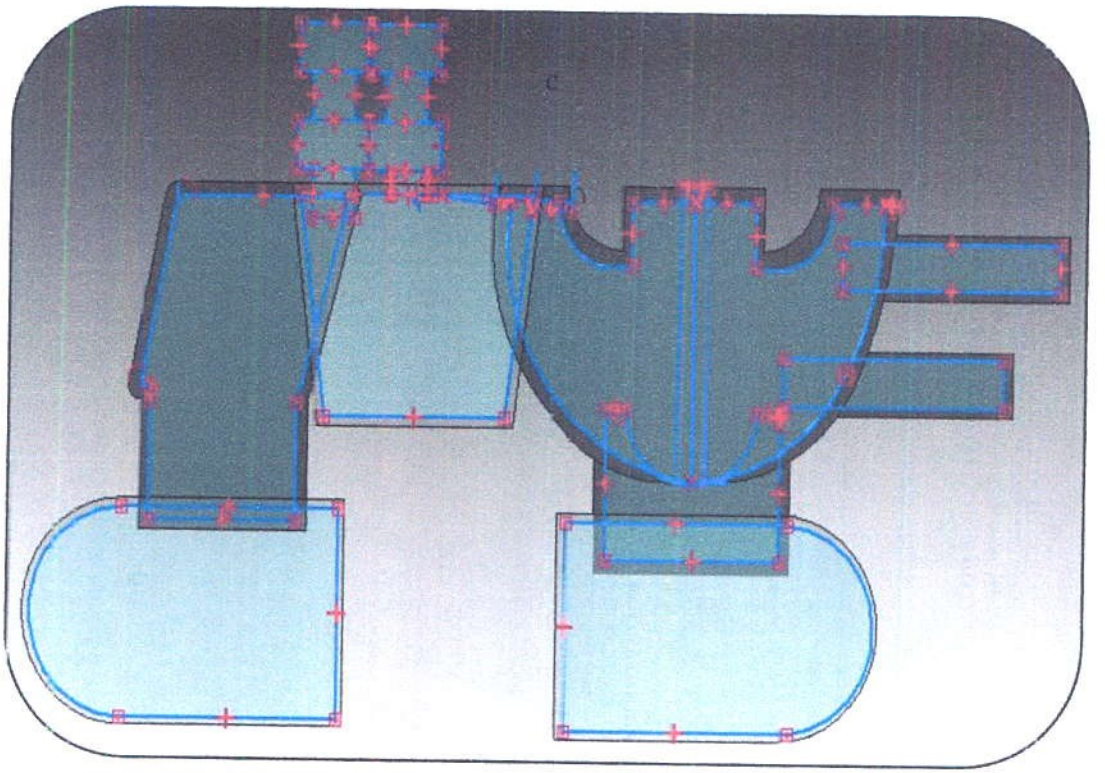


Figura 27. Unión y plegaje de protectores del IV dedo mano derecha

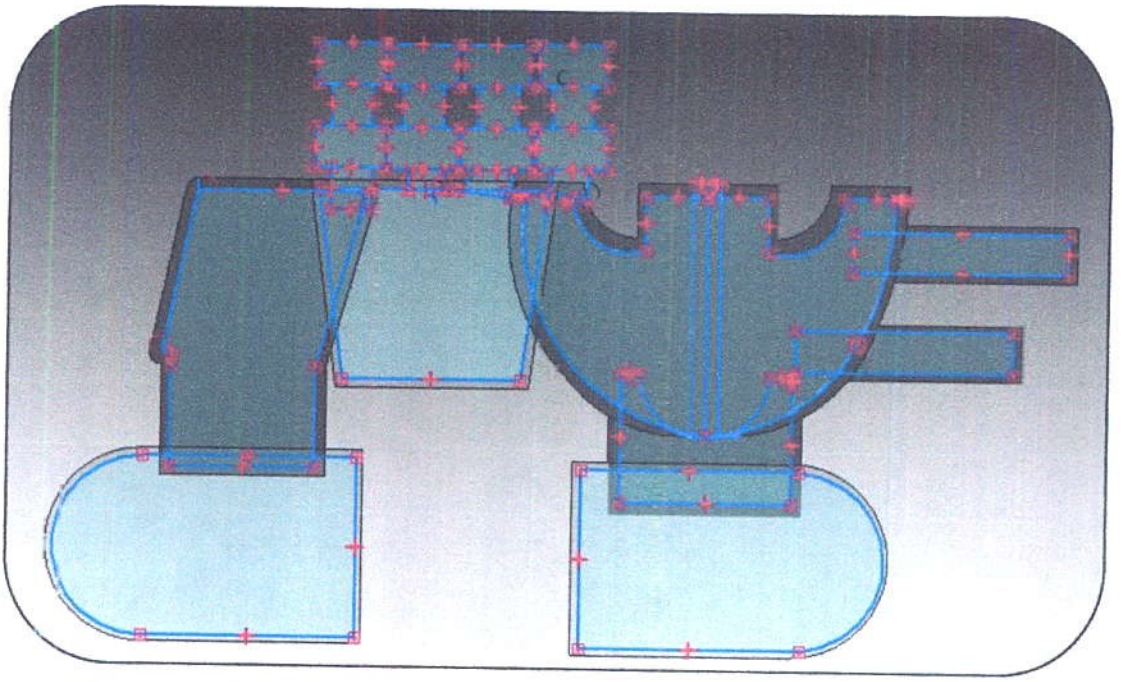


Figura 28. Unión y plegaje de protectores del I a V dedo mano derecha

1.5 Fotos del producto terminado: Interface Órtesis Piel

Estas fotos son tomadas con el guante de cuero y neopreno cubriendo la mano del paciente discapacitado.



Figura 29. Guante de cuero y neopreno mano derecha cara lateral interna

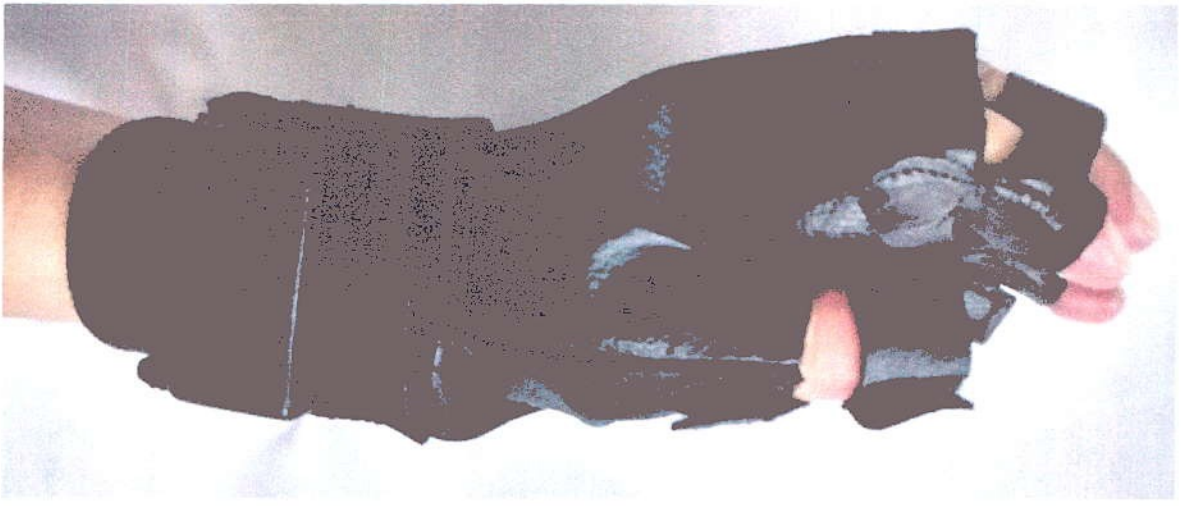


Figura 30. Guante de cuero y neopreno cara mano izquierda lateral interna



Figura 31. Guante de cuero y neopreno cara palmar y dorsal mano izquierda



Figura 32. Guante de cuero y neopreno cara dorsal ambas manos

2. FABRICACION DE LA INTERFACE ÓRTESIS PASIVA

2.1 Protectores Servomotor 1:

- Se utiliza una **máquina cortadora laser de acrílico**, se cortan las piezas con las dimensiones especificadas en el CAPITULO I (diseño de piezas de acrílico).

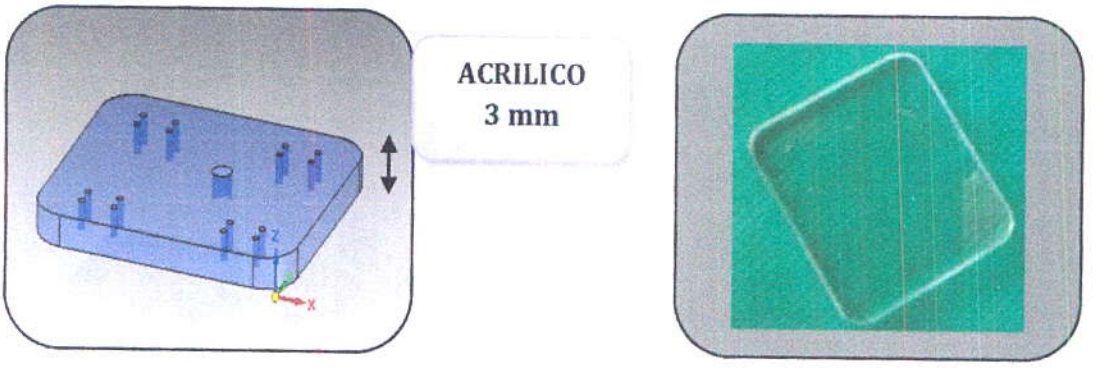


Figura 33. Pieza en acrílico BS1_MN a) Diseño Solid Edge y b) Pieza Fabricada

- Las piezas de acrílico deben calentarse en una **resistencia de calefacción,** para darle forma a cada pieza en particular, evitando que el acrílico se deforme.

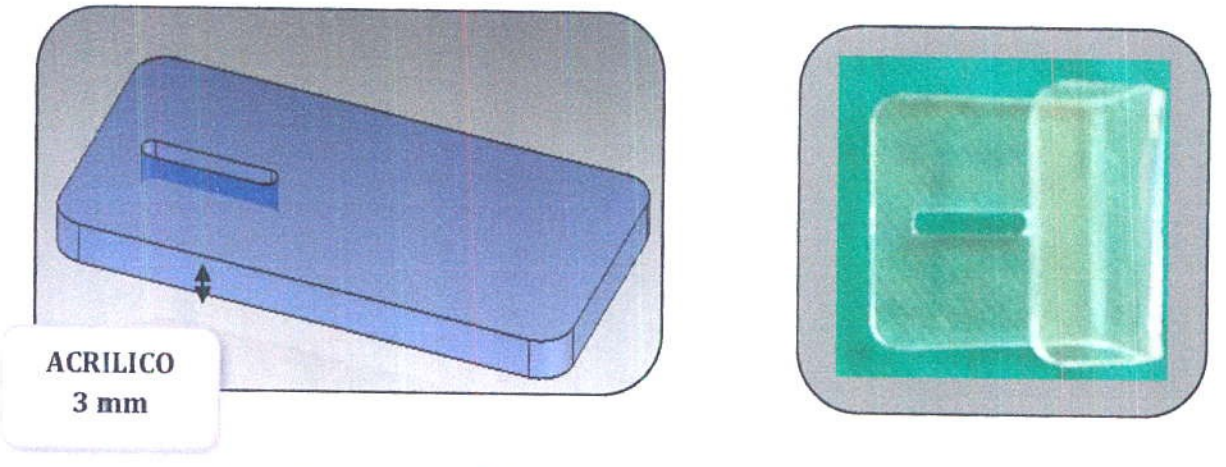


Figura 34. Pieza en acrílico PS1_MN a) Diseño Solid Edge y b) Pieza Fabricada

2.2 Protectores Servomotor 2:

- Se utiliza una **máquina cortadora laser de acrílico**, se cortan las piezas con las dimensiones especificadas en el CAPITULO I (diseño de piezas de acrílico).
- Las piezas de acrílico deben calentarse en una **resistencia de calefacción,** para darle forma a cada pieza en particular, evitando que el acrílico se deforme.

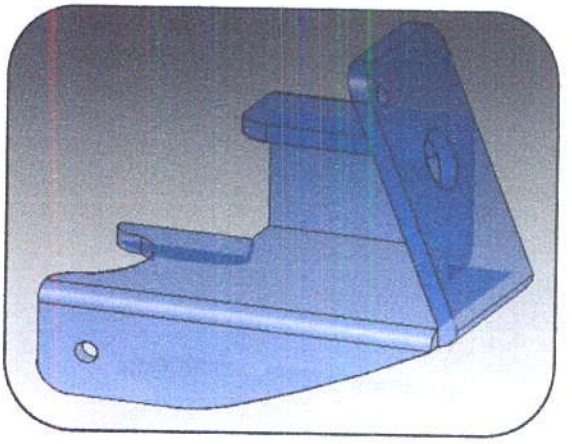


Figura 35. Pieza PS2_MI a) Diseño en Solid Edge



Figura 36. Pieza PS2_MI b) Pieza fabricada

2.3 Protectores Servomotor 3:

- Se utiliza una **máquina cortadora laser de acrílico**, se cortan las piezas con las dimensiones especificadas en el CAPITULO I (diseño de piezas de acrílico).

- Las piezas de acrílico deben calentarse en una **resistencia de calefacción,** para darle forma a cada pieza en particular, evitando que el acrílico se deforme.

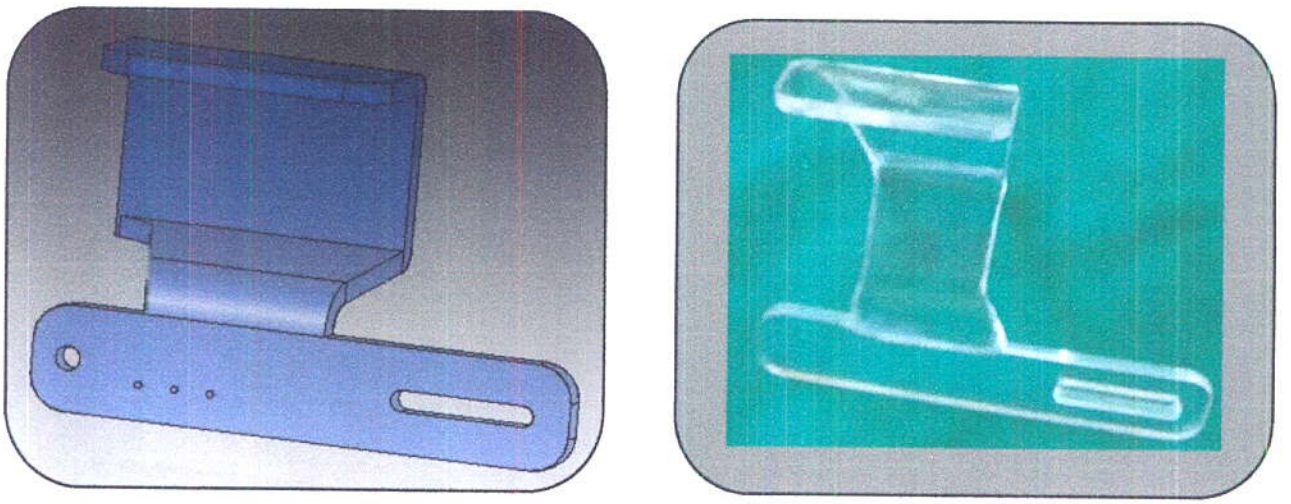


Figura 37. Pieza PS3_FPI a) Diseño en Solid Edge y b) Pieza fabricada.

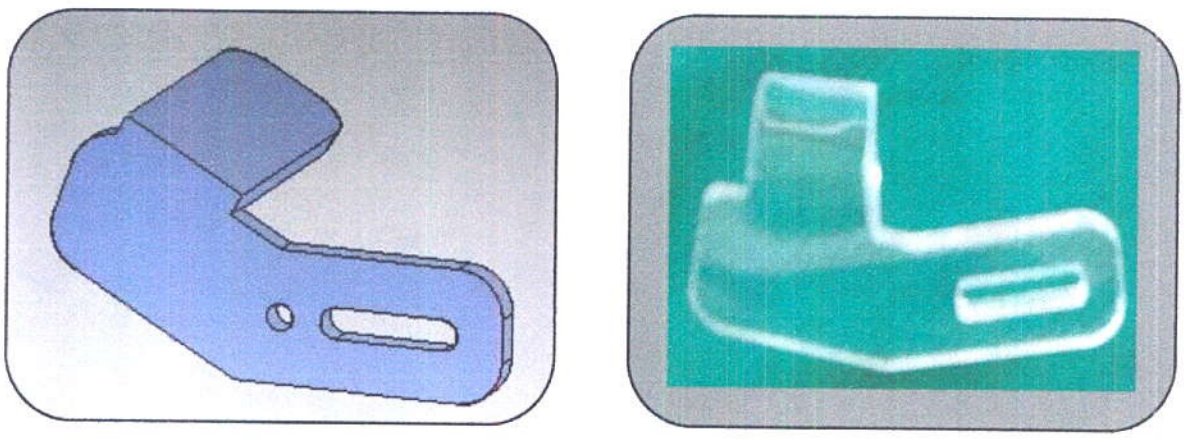


Figura 38. Pieza BP_FDI a) Diseño en Solid Edge y b) Pieza fabricada.

2.4 Protectores Servomotor 4:

- Se utiliza una **máquina cortadora laser de acrílico**, se cortan las piezas con las dimensiones especificadas en el CAPITULO I (diseño de piezas de acrílico).

- Las piezas de acrílico deben calentarse en una **resistencia de calefacción,** para darle forma a cada pieza en particular, evitando que el acrílico se deforme.

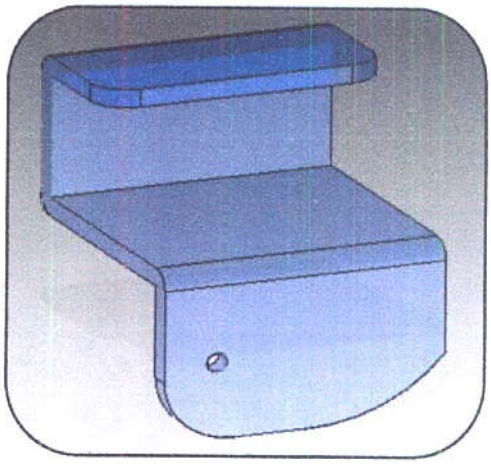


Figura 39. Pieza BS4_MII a) Diseño en Solid Edge



Figura 40. Pieza BS4_MII b) Pieza fabricada.

2.5 Protectores Servomotor 5:

- Se utiliza una **máquina cortadora laser de acrílico**, se cortan as piezas con las dimensiones especificadas en el CAPITULO I (diseño de piezas de acrílico).

- Las piezas de acrílico deben calentarse en una **resistencia de calefacción,** para darle forma a cada pieza en particular, evitando que el acrílico se deforme.

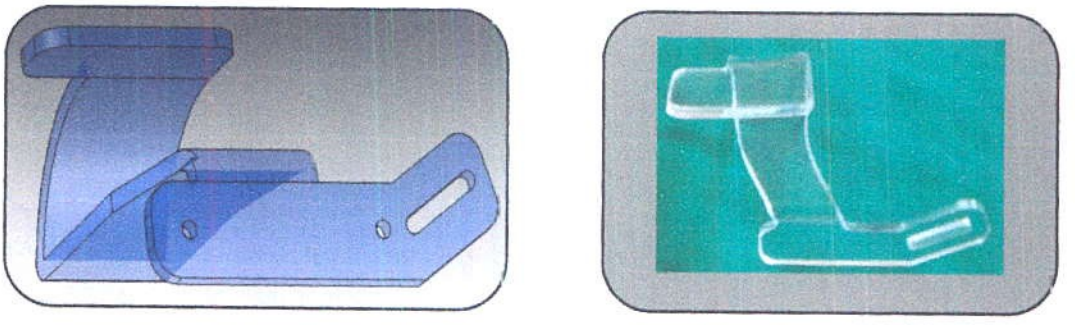


Figura 41 Pieza PS5_FPII a) Diseño en Solid Edge y b) Pieza fabricada

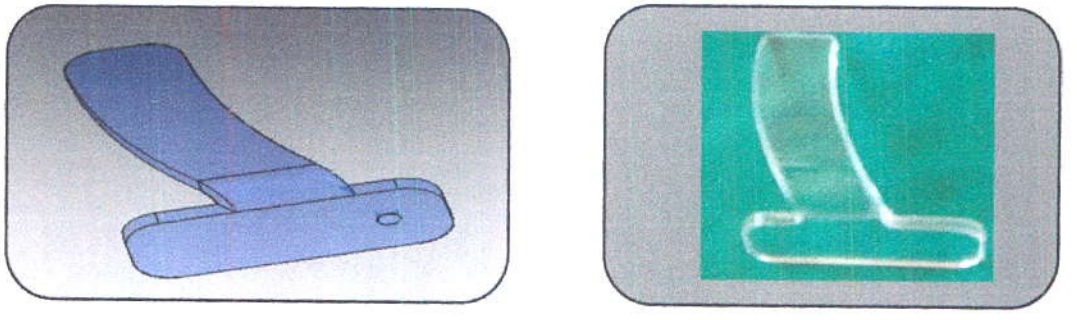


Figura 42 Pieza B_FMII a) Diseño en Solid Edge y b) Pieza fabricada

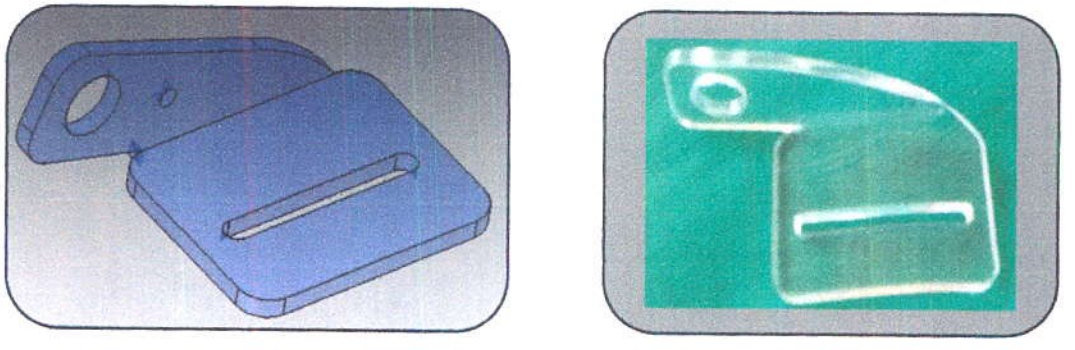


Figura 43 Pieza P_FMII a) Diseño en Solid Edge y b) Pieza fabricada

2.6 Protectores Servomotor 6:

- Se utiliza una **máquina cortadora laser de acrílico**, se cortan las piezas con las dimensiones especificadas en el CAPITULO I (diseño de piezas de acrílico).

- Las piezas de acrílico deben calentarse er una **resistencia de calefacción,** para darle forma a cada pieza en particular, evitando que el acrílico se deforme.

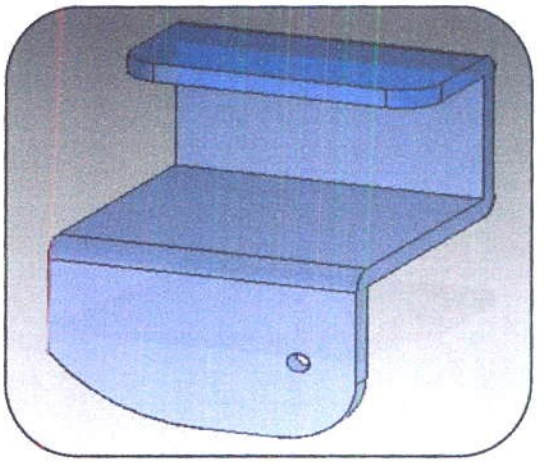


Figura 44. Pieza BS6_MIV a) Diseño en Solid Edge



Figura 45. Pieza BS6_MIV b) Pieza fabricada.

2.7 Protectores Servomotor 7:

- Se utiliza una **máquina cortadora laser de acrílico**, se cortan las piezas con las dimensiones especificadas en el CAPITULO I (diseño de piezas de acrílico).

- Las piezas de acrílico deben calentarse en una **resistencia de calefacción,** para darle forma a cada pieza en particular, evitando que el acrílico se deforme.

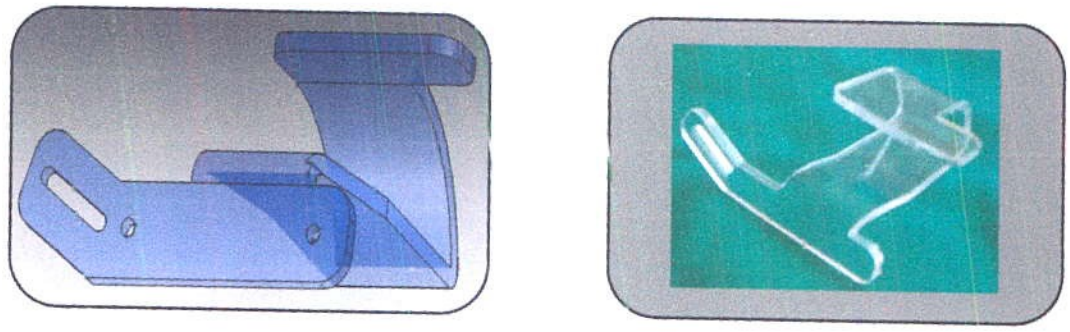


Figura 46. Pieza PS7_FPIV a) Diseño en Solid Edge y b) Pieza fabricada

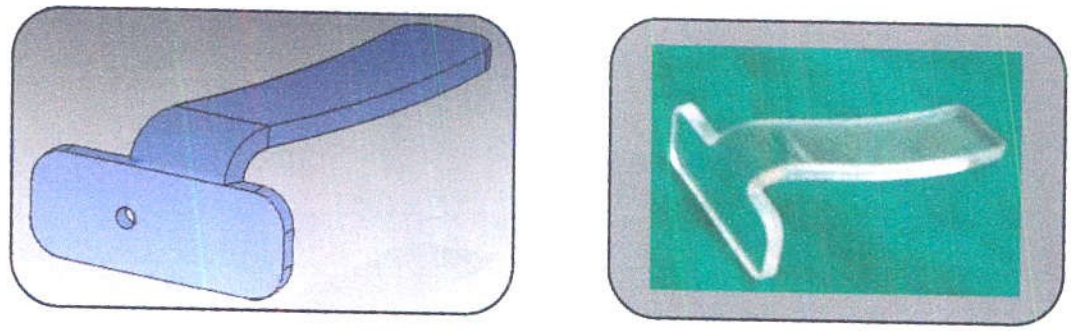


Figura 47. Pieza B_FMIV a) Diseño en Solid Edge y b) Pieza fabricada

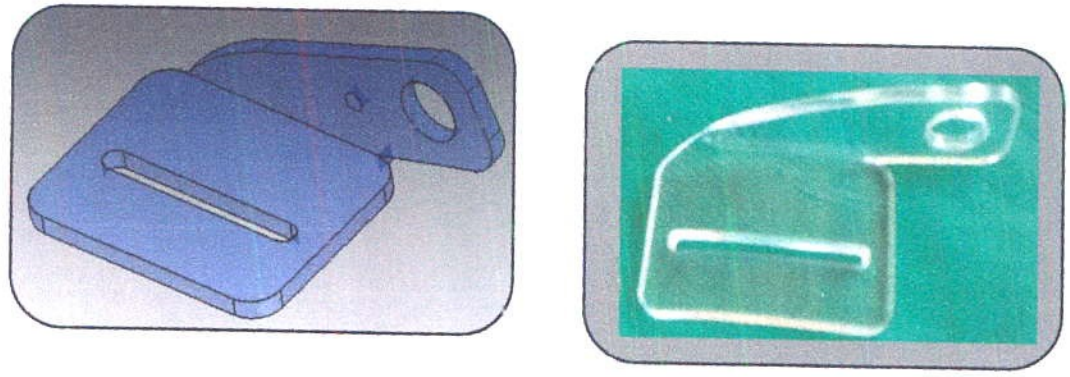


Figura 48. Pieza P_FMIV a) Diseño en Solid Edge y b) Pieza fabricada

2.8 Protectores Servomotor 8:

- Se utiliza una **máquina cortadora laser de acrílico**, se cortan las piezas con las dimensiones especificadas en el CAPITULO I (diseño de piezas de acrílico).

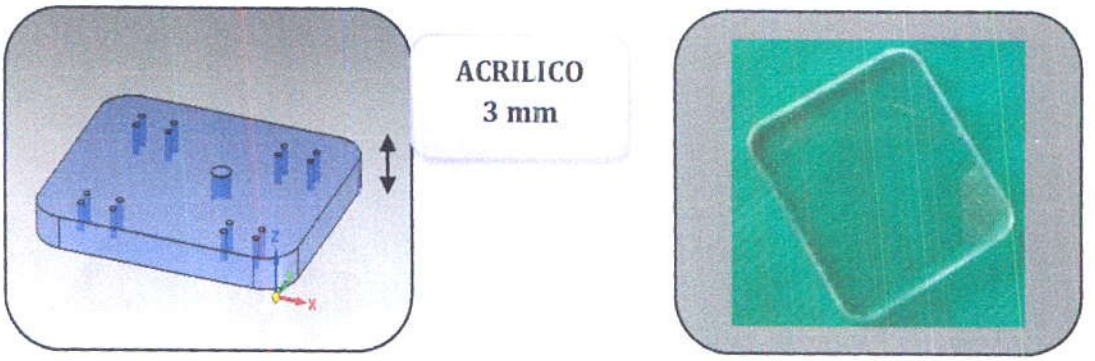


Figura 49. Pieza en acrílico BS8_MN a) Diseño Solid Edge y b) Pieza Fabricada

- Las piezas de acrílico deben calentarse en una **resistencia de calefacción,** para darle forma a cada pieza en particular, evitando que el acrílico se deforme.

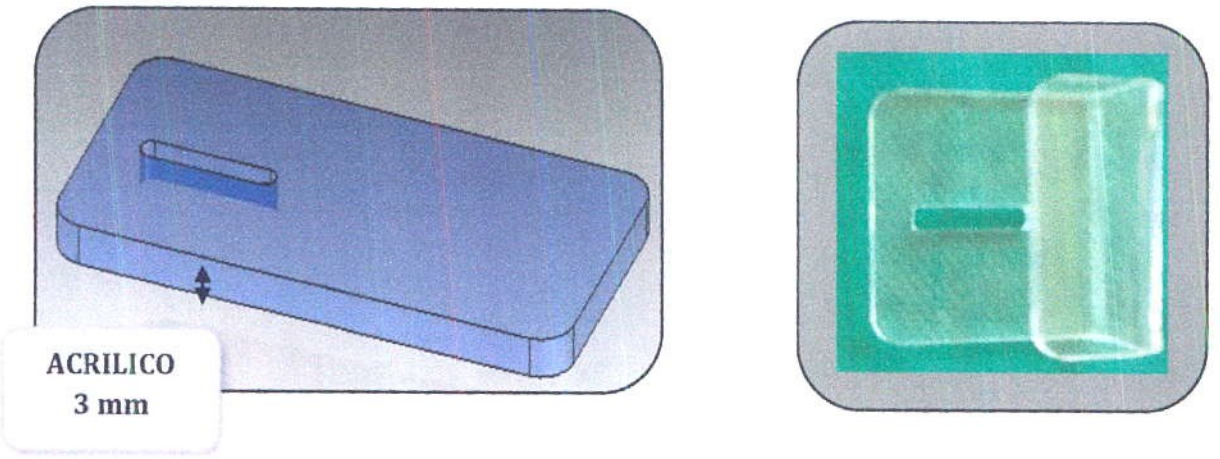


Figura 50. Pieza en acrílico PS8_MN a) Diseño Solid Edge y b) Pieza Fabricada

- Para el movimiento de muñeca, se ha diseñado una palanca que abarca la mitad de la muñeca dorsal y ventralmente.

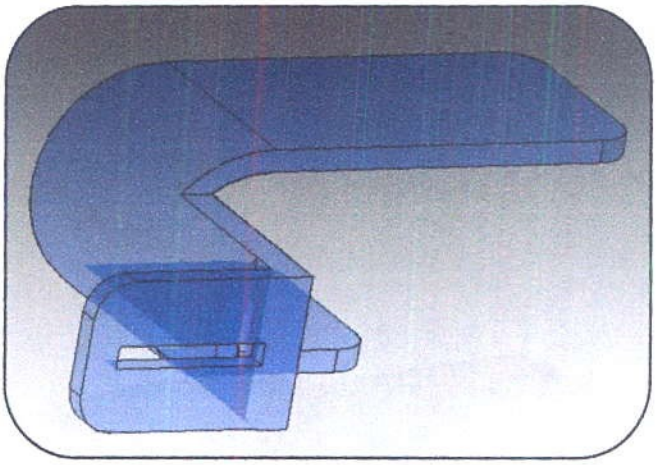


Figura 51. Pieza PLS8_MN a) Diseño en Solid Edge

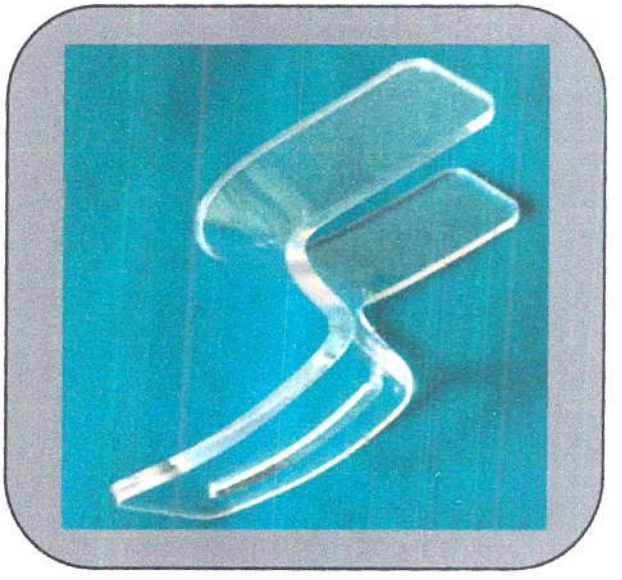


Figura 52. Pieza PLS8_MN b) Pieza fabricada.

2.9. Fotos del Producto Terminado: Interface Órtesis Pasiva:

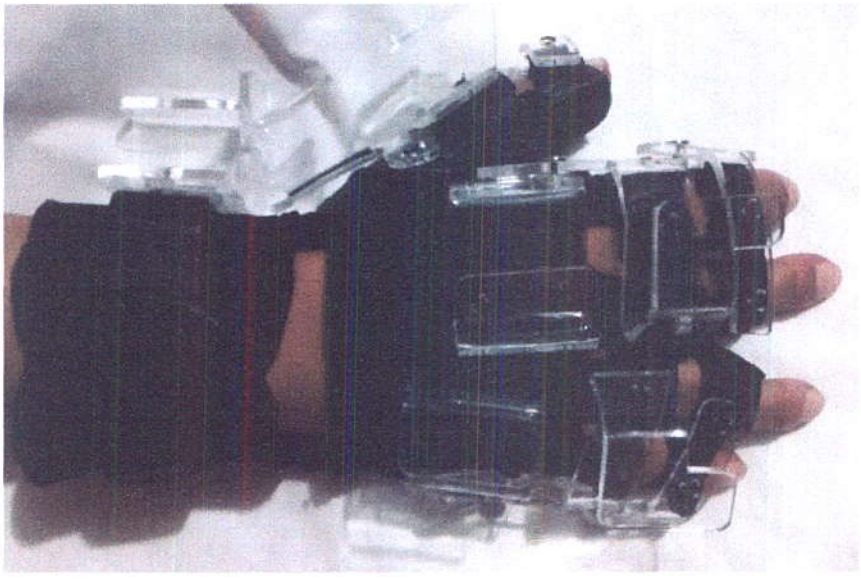


Figura 53. Órtesis Pasiva, cara dorsal mano derecha.



Figura 54. Órtesis Pasiva, cara palmar mano derecha.

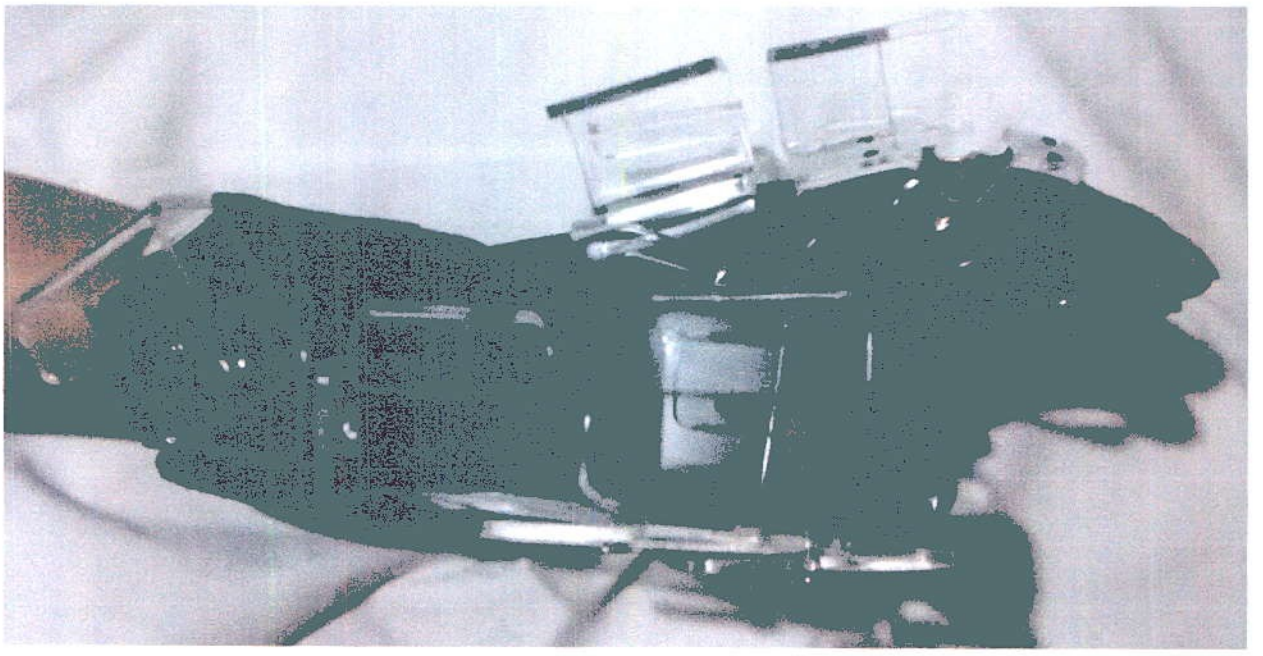


Figura 55. Órtesis Pasiva, cara interna mano izquierda

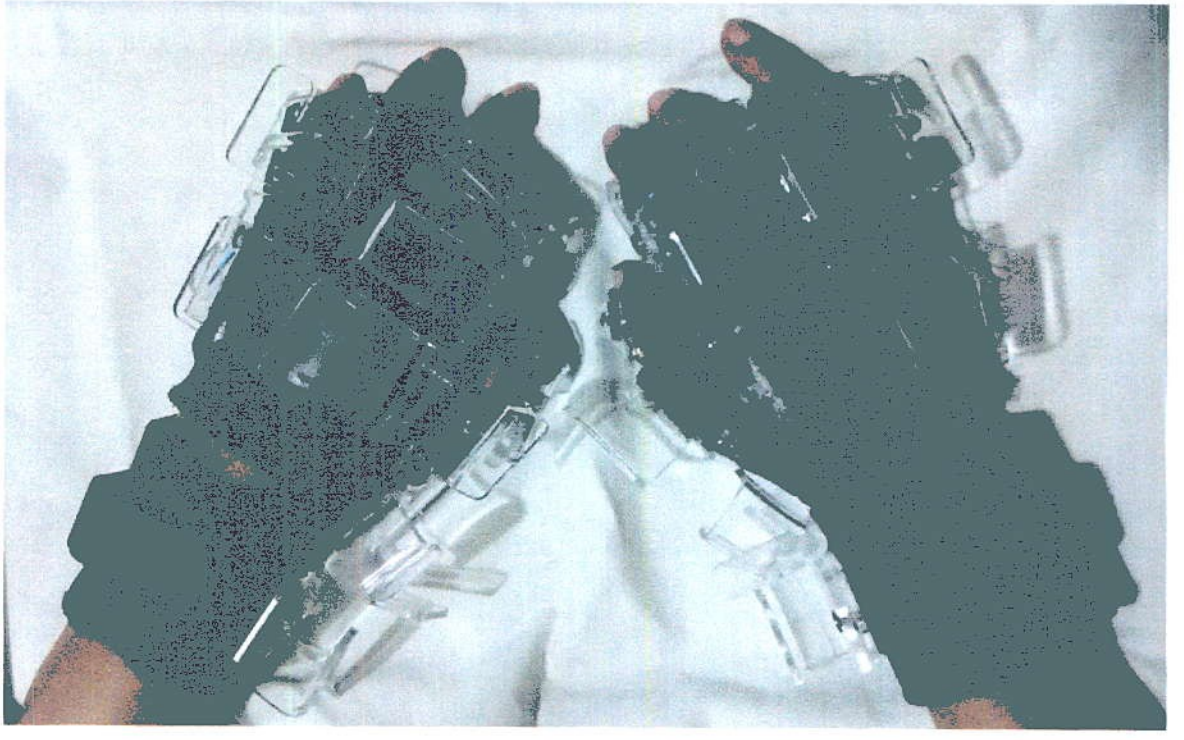


Figura 56. Órtesis Pasiva, cara dorsal ambas manos.



Figura 57. Órtesis Pasiva, cara interna ambas manos.



Figura 58. Órtesis Pasiva, cara palmar ambas manos

3. FABRICACIÓN INTERFACE ÓRTESIS ACTIVA

3.1 Diseño movimiento I metacarpiano

- Para la ubicación de cada una de las piezas de acrílico en el exoesqueleto de mano y muñeca, realizaremos la simulación en Solid Edge progresivamente a medida que se ensamblan las piezas desde el Servomotor 1 hasta el Servomotor 8.

- Todas las piezas tienen el nombre y las dimensiones, de acuerdo a las especificaciones del CAPITULO I del informe final.

- Primero tomamos como referencia las piezas metálicas que dan soporte al exoesqueleto a nivel de la muñeca:

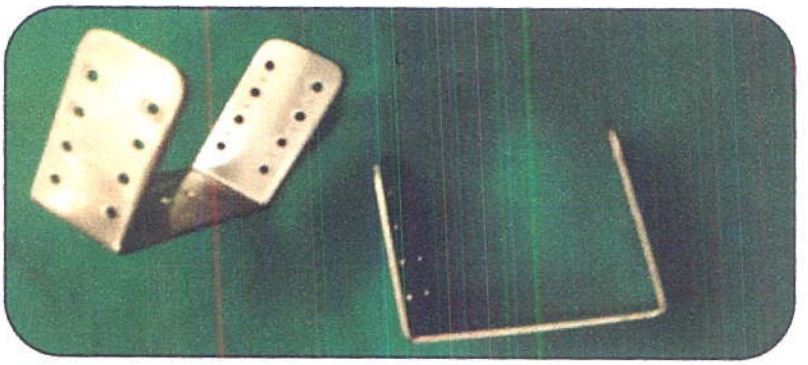


Figura 59. Piezas en acero inoxidable para protección de muñeca derecha

- Luego las ubicamos espacialmente el software Solid Edge

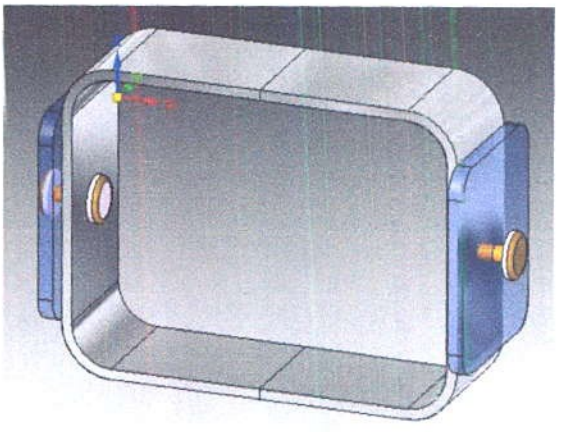


Figura 60. Piezas en acero inoxidable de la muñeca. Simulación Ensamblaje Solid Edge

- Elegimos los componentes que debe tener la simulación de la articulación que dará movimiento al I metacarpiano, este representa el movimiento circular del I dedo, para lograr la prensión de objetos.

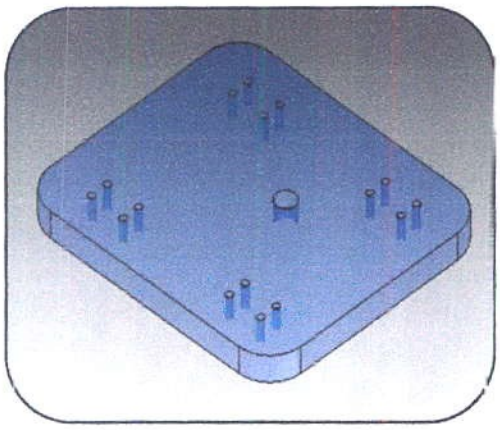


Fig 61. PIEZA BS1_MN

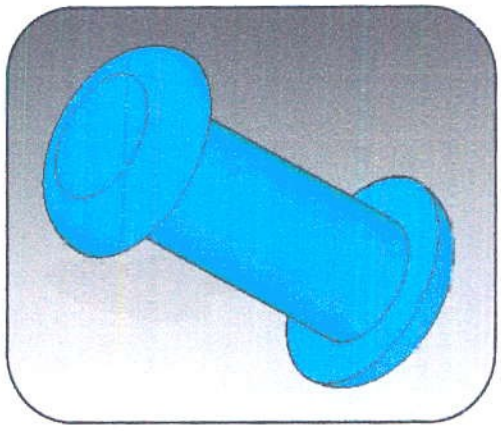


Fig 62. ESLABON

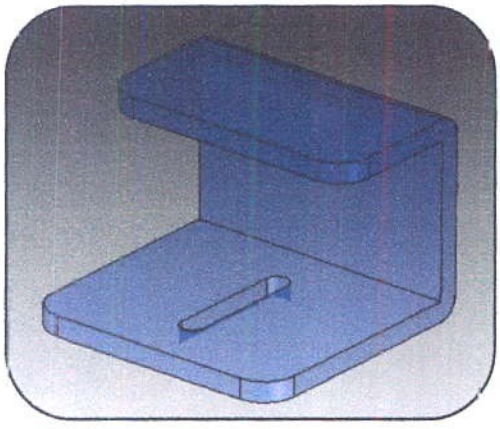


Fig 63. PIEZA PS1_MN

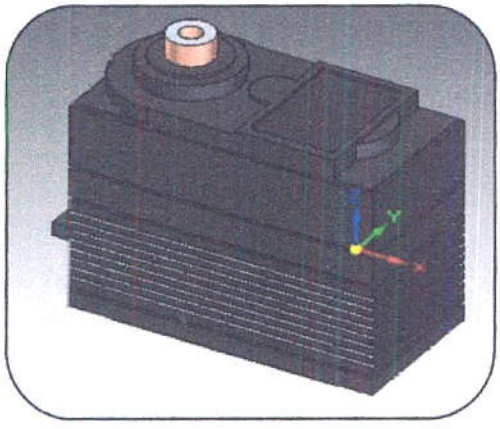


Fig 64. SERVOMOTOR

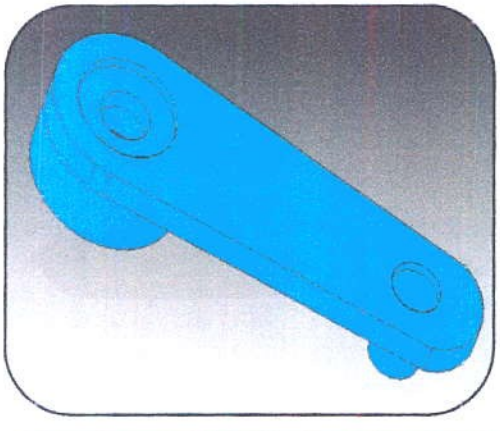


Fig 65. PALANCA SERVO

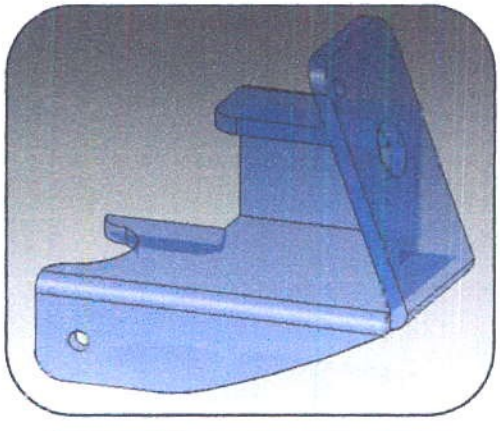


Fig 66. PIEZA PS2_MI

- El remache en duraluminio tiene alineación axial con la base en acrílico.

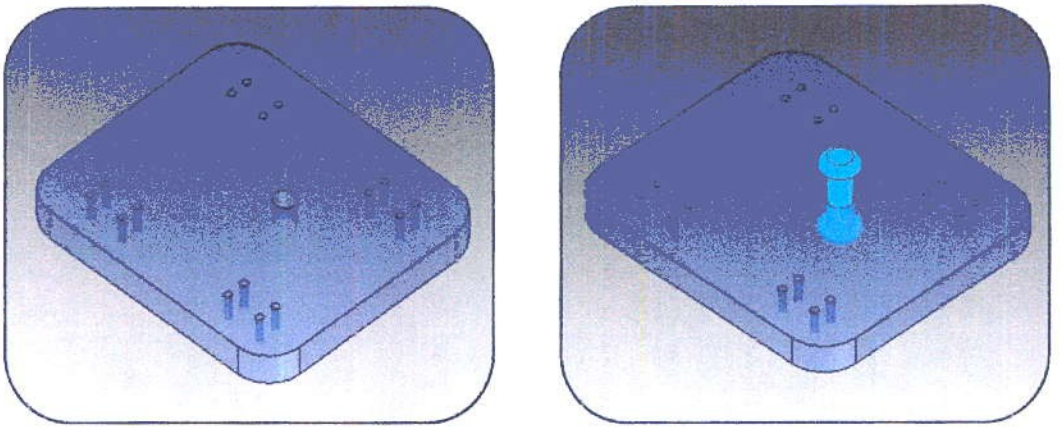


Figura 67. Ubicación del remache de duraluminio en la Pieza BS1_MN

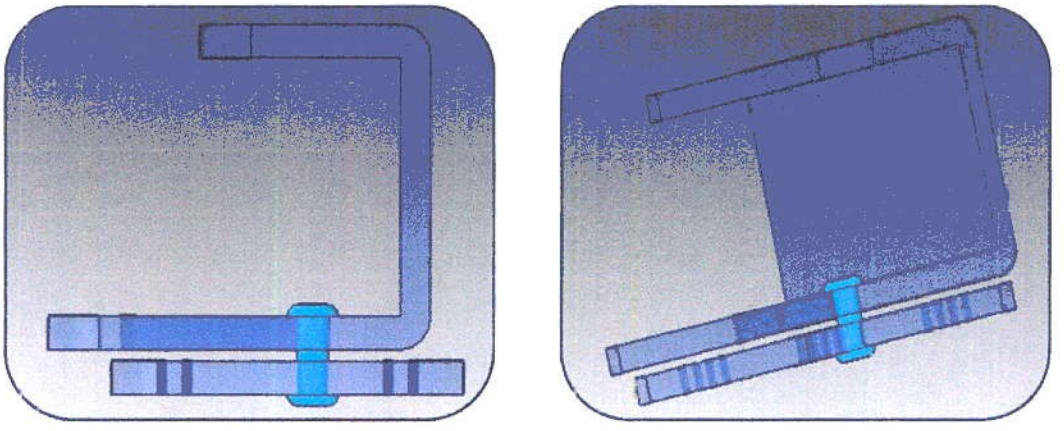


Figura 68. Ubicación del remache de duraluminio en la Pieza PS1_MN y

- El **Servomotor** 1 se integra a las bases de acrílico mediante cinta doble fax transparente.

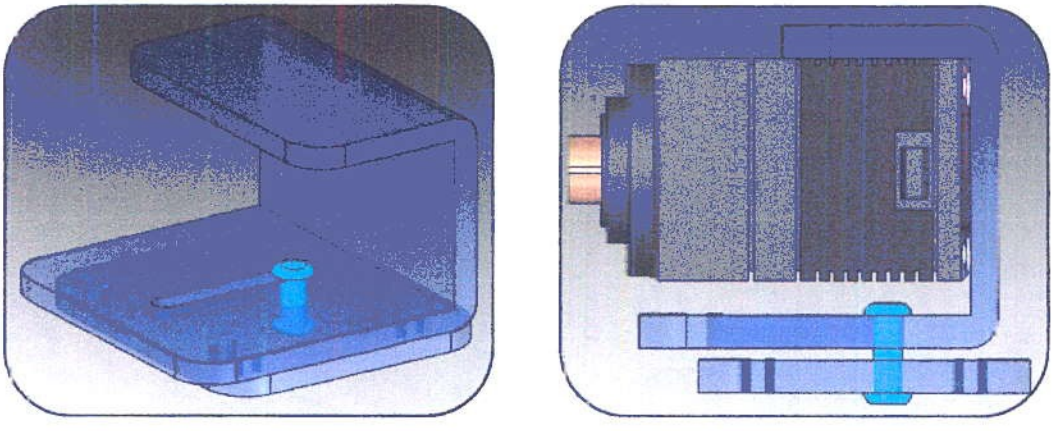


Figura 69. Ensamble del Servomotor 1 a las piezas BS1_MN y PS1_MN

- La ubicación del servomotor es estratégica para definir el movimiento

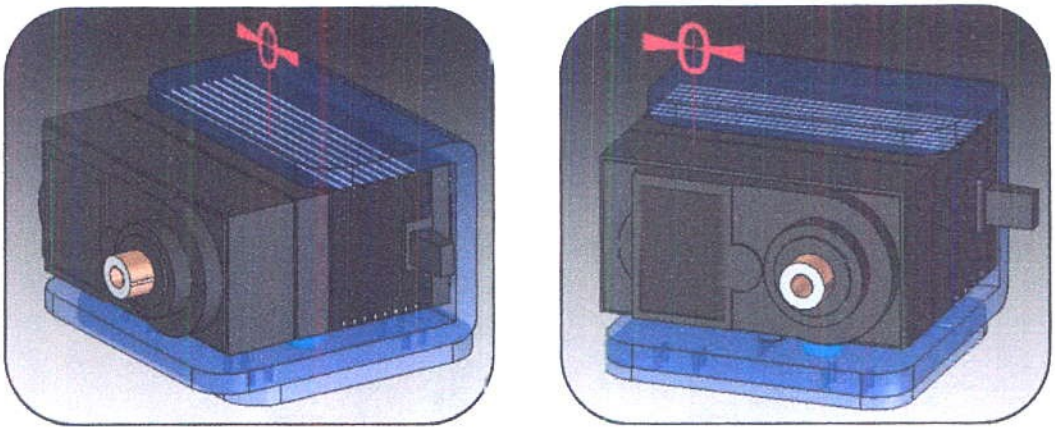


Figura 70. Ensamble del Servomotor 1 a las piezas BS1_MN y PS1_MN

- Ensamblamos la pieza **PS2_MI** al conjunto, unida por la palanca del Servomotor 1

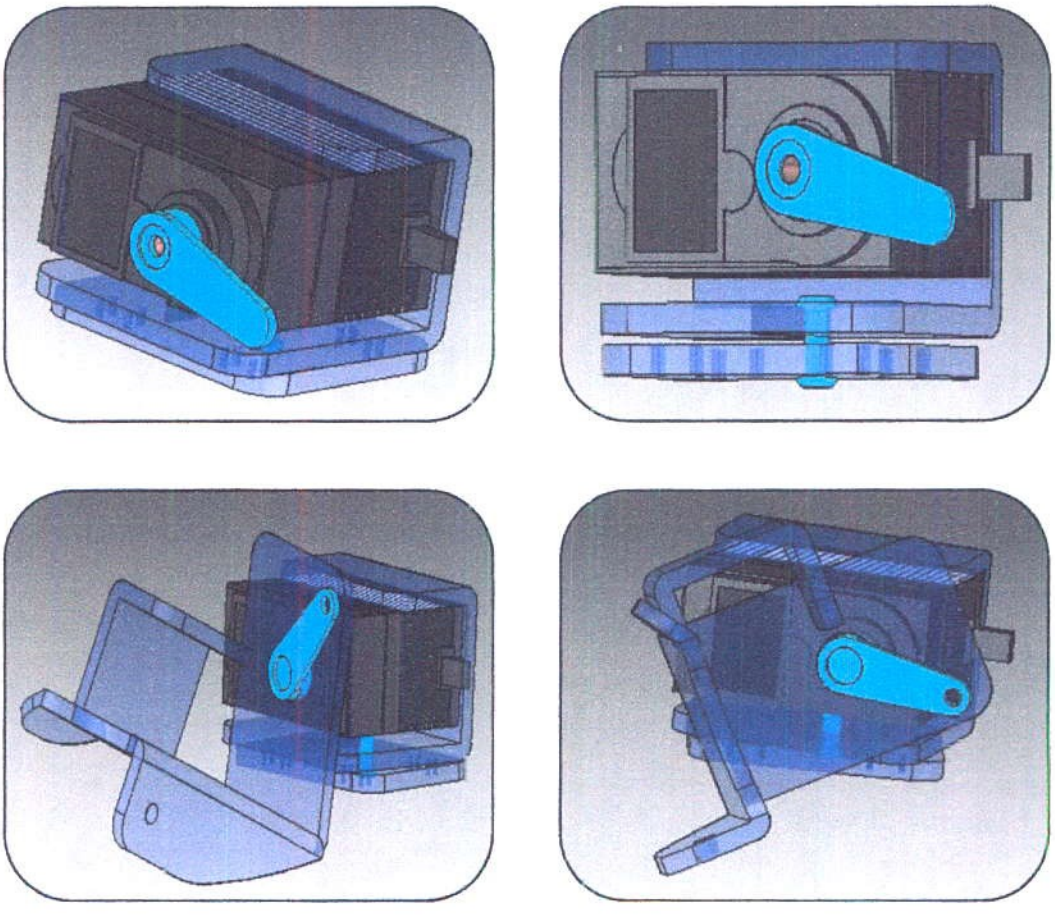


Figura 71. Ensamblaje componentes al Servomotor 1 y movimiento del I metacarpiano

3.2 Diseño movimiento articulación metacarpofalangica I dedo

- Elejimos los componentes que se integran a la articulación paralela a la metacarpofalángica del I dedo, para simular el movimiento en el Software Solide Edge.

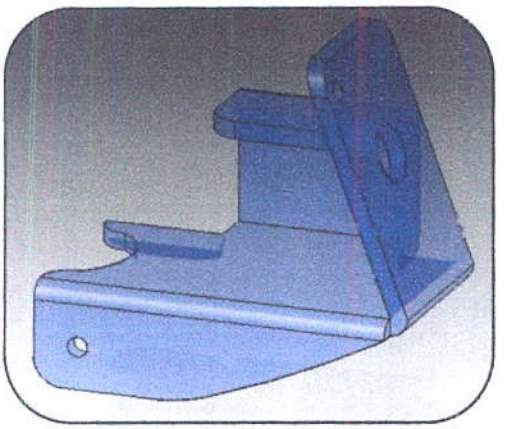


Fig 72. PIEZA PS2_MI

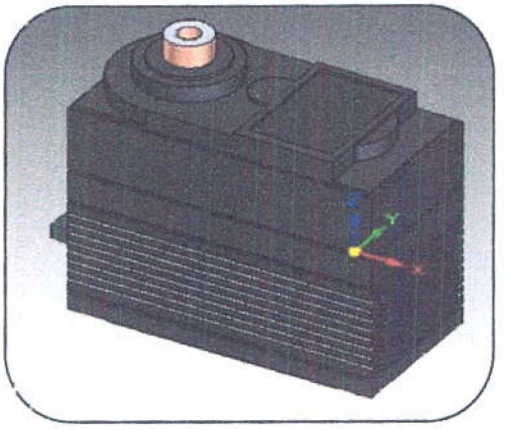


Fig 73. SERVOMOTOR

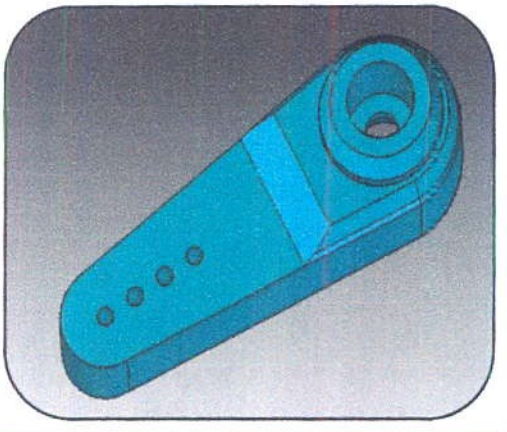


Fig 74. PALANCA SERVO

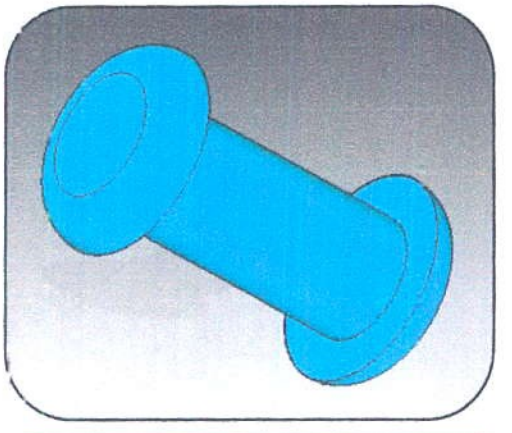


Fig 75. ESLABON

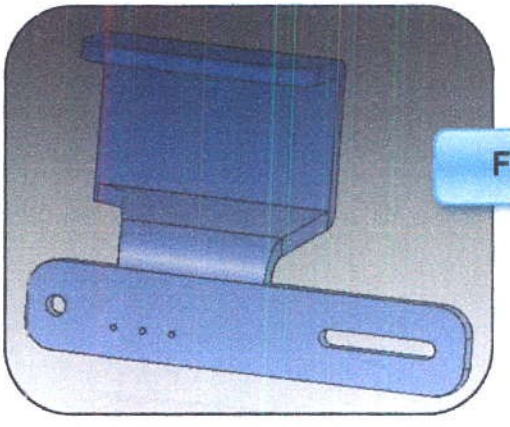


Fig 76. PIEZA PS3_FPI

- Se ensambla la Pieza **PS2_MI** a la Pieza **PS3_FPI**, unidas ambas piezas de acrílico por un remache doblecabeza de duraluminio. Posteriormente, se integra el **Servomotor 2** a la Pieza **PS2_MI**.

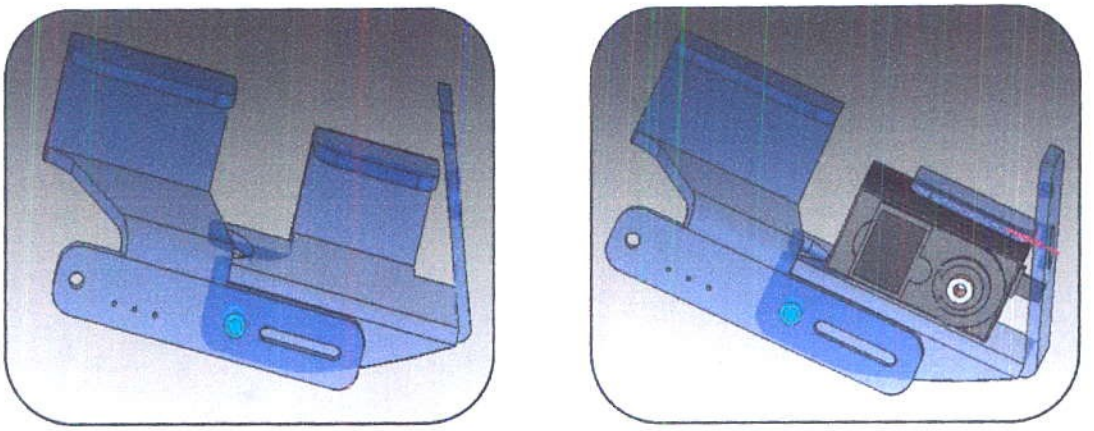


Figura 77. Ensamblaje componentes al Servomotor 2 para el movimiento del I dedo

- La palanca del servomotor se desplaza en forma circular, generando un movimiento angular de la pieza distal.

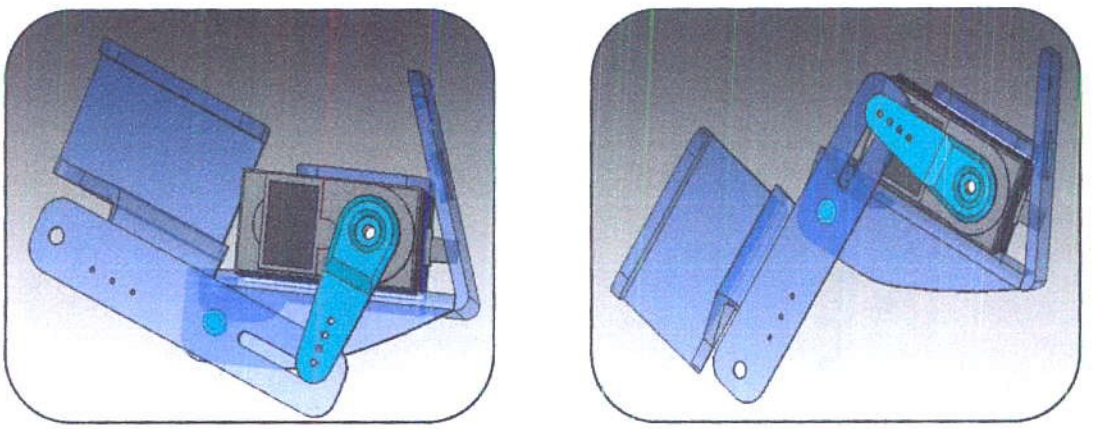


Figura 78. Ensamblaje de piezas para el movimiento de la falange proximal del I dedo

3.3 Diseño movimiento articulación interfalangica I dedo

- Elejimos los componentes que se integran a la articulación paralela a la metacarpofalángica del I dedo, para simular el movimiento en el Software Solide Edge.

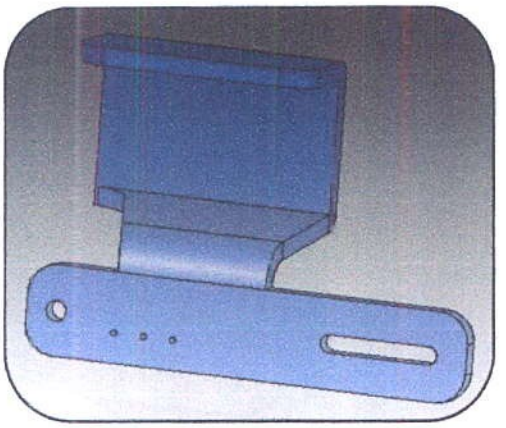


Fig 79. PIEZA PS3_FPI

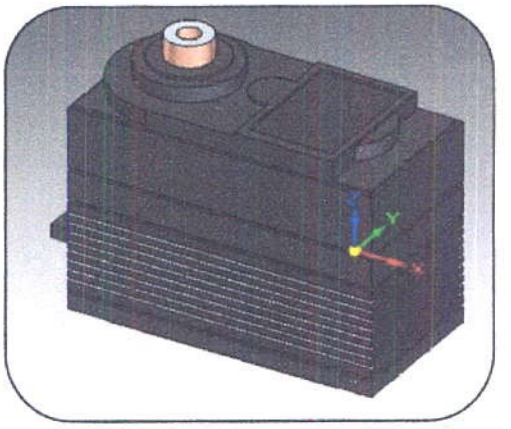


Fig 80. SERVOMOTOR

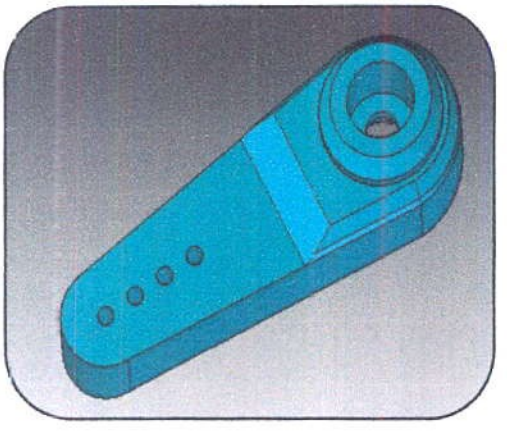


Fig 81. PALANCA SERVO

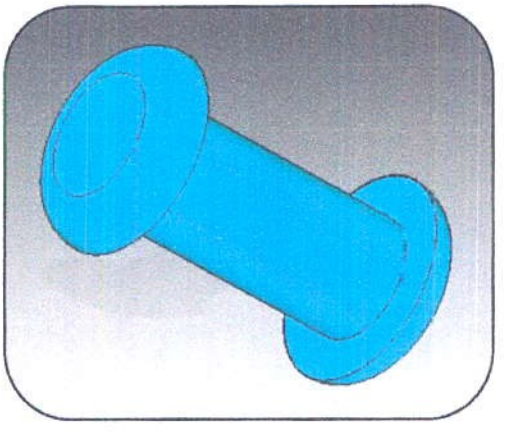


Fig 82. ESLABON

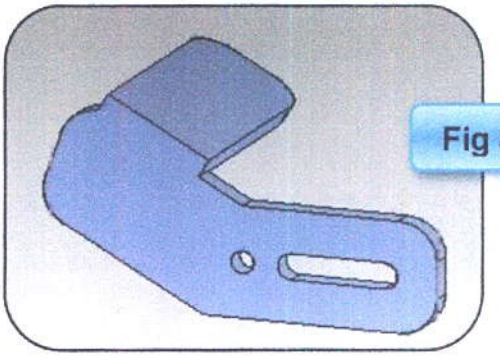


Fig 83. PIEZA BP_FDI

- Se unen las piezas **PS3_FPI** y la **BP_FDI** con un eslabon tipo remache doblecabeza en duraluminio. Posteriormente, el **Servomotor 3** se integra a la pieza **PS3_FPI**

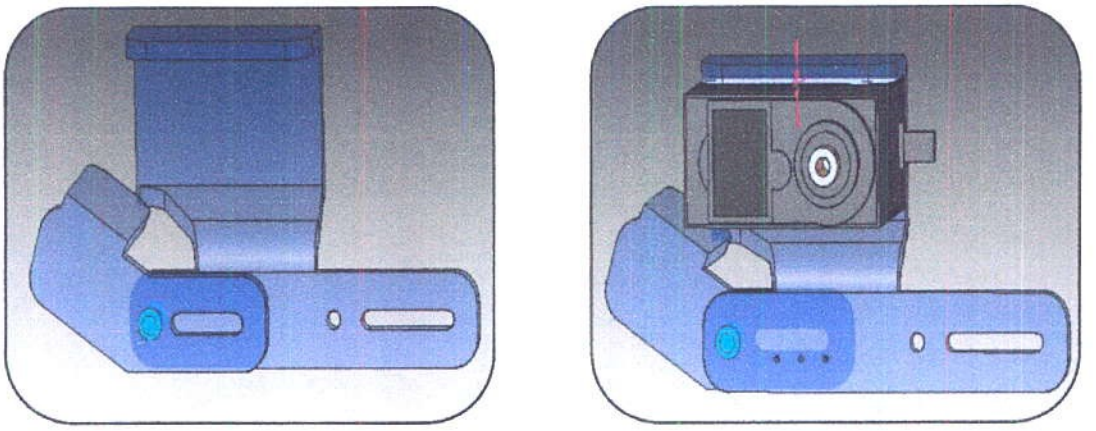


Figura 84. Ensamblaje de las piezas PS3_FPI y BP_FDI al Servomotor I

- La palanca del servomotor se desplaza en forma circular, generando un movimiento angular de la pieza distal.

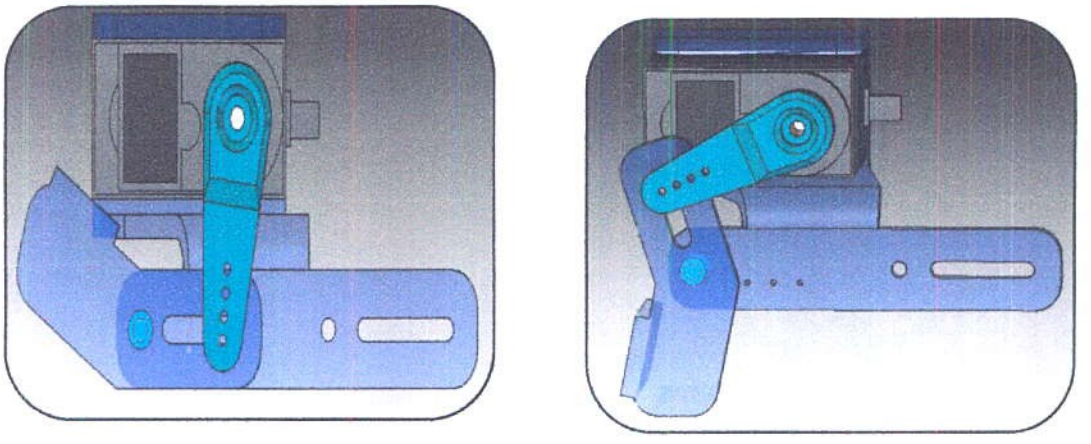


Figura 85. Ensambleje de piezas para el movimiento de la falange distal i dedo.

3.4 Diseño movimiento articulaciónes metacarpofalangicas II y III dedos

- Elejmos los componentes que se integran a la articulación paralela a la metacarpofalángica del I dedo, para simular el movimiento en el Software Solide Edge.

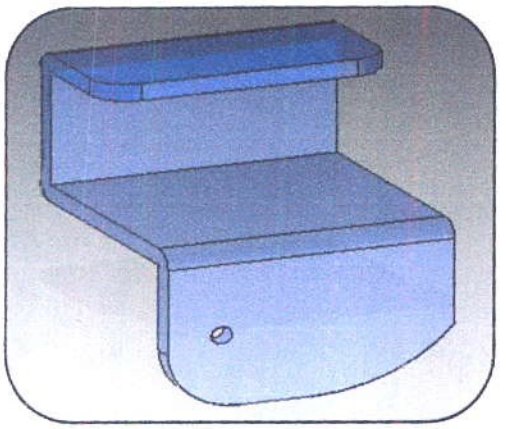


Fig 86. PIEZA BS4_MII

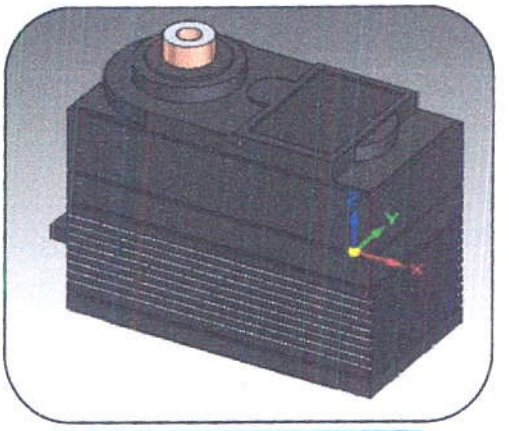


Fig 87. SERVOMOTOR

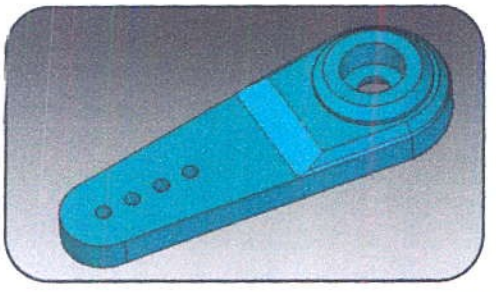


Fig 88. PALANCA SERVO

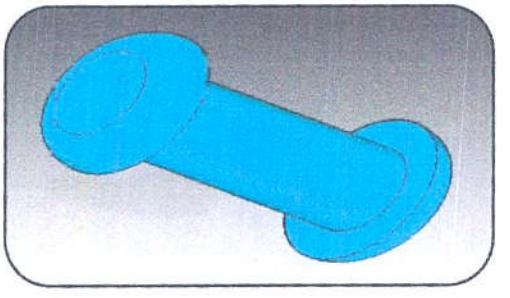


Fig 89. ESLABON

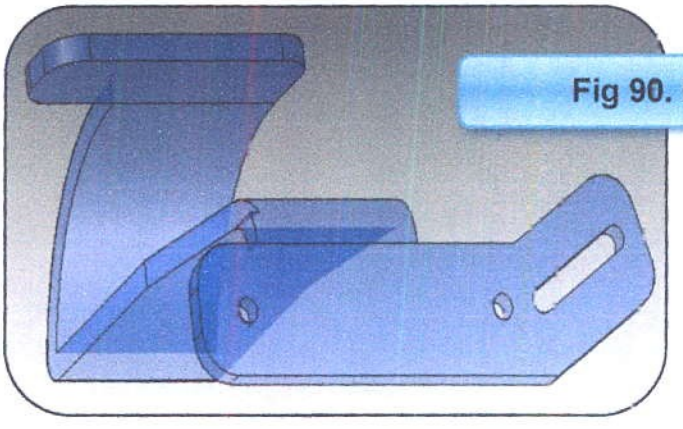


Fig 90. PIEZA PS5_FPII

- Se unen las piezas **BS4_MII** y la **PS5_FPII** con un eslabon tipo remache doblecabeza en duraluminio. Posteriormente, el **Servomotor 4** se integra a la pieza **BS4_MII**

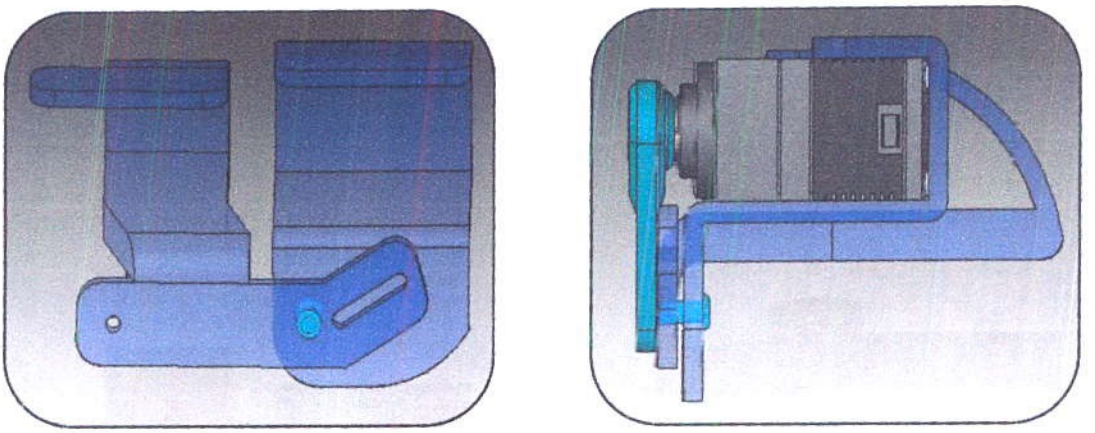


Figura 91. Ensambleje de piezas para el movimiento de la falange proximal de il y III dedos.

- La palanca del servomotor se desplaza en forma circular, generando un movimiento angular de la pieza distal.

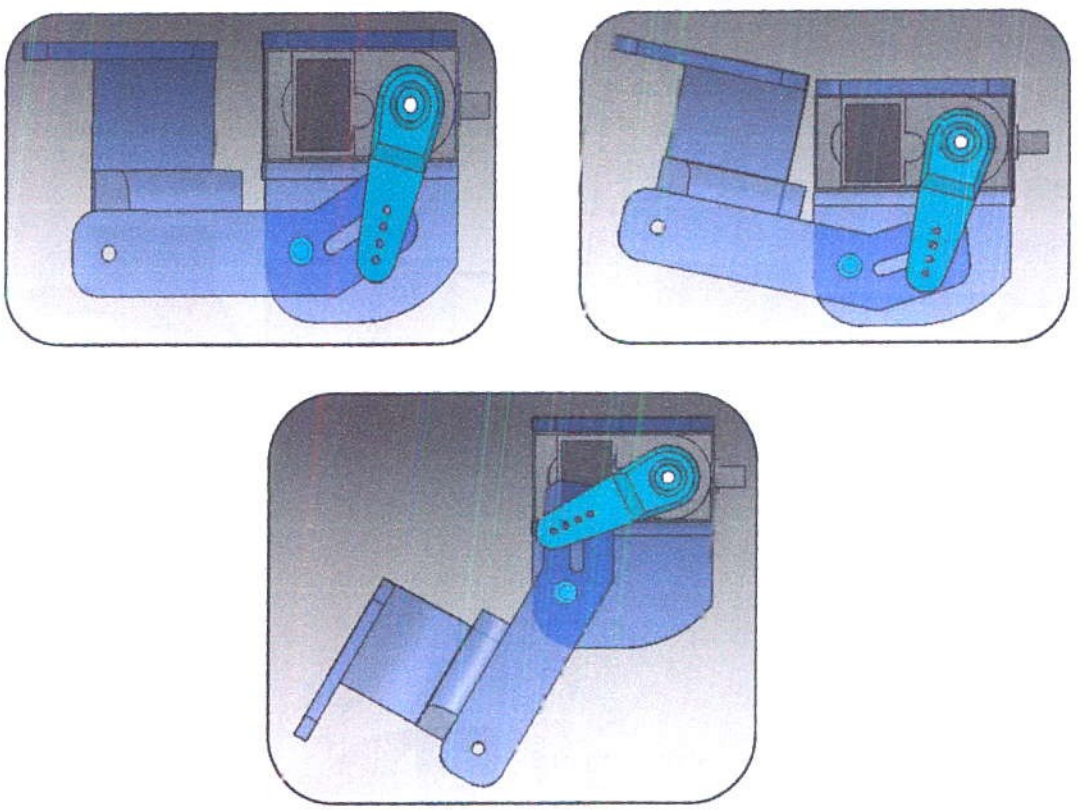


Figura 92. Ensambleje de piezas para el movimiento de la falange proximal de il y III dedos.

3.5 Diseño movimiento articulación interfalangica proximal II y III dedos

- Elejimos los componentes que se integran a la articulación paralela a la metacarpofalángica del I dedo, para simular el movimiento en el Software Solide Edge.

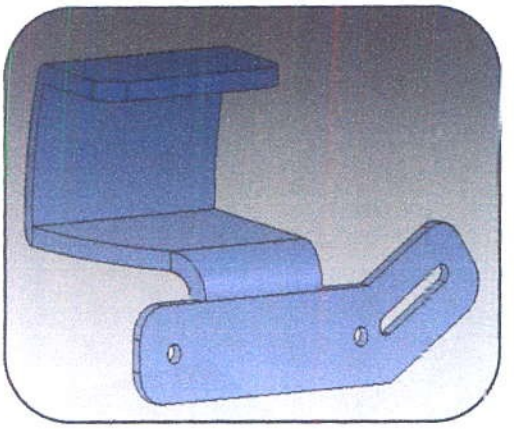


Fig 93. PIEZA PS5_FPII

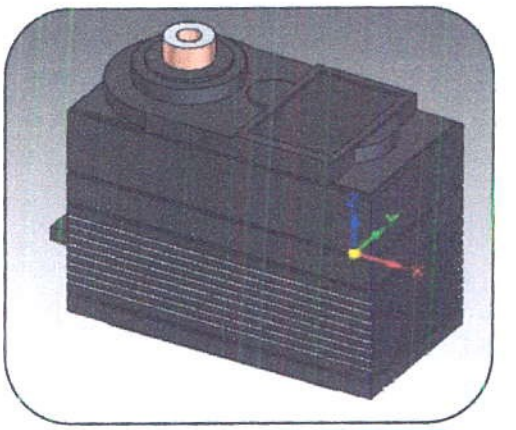


Fig 94. SERVOMOTOR

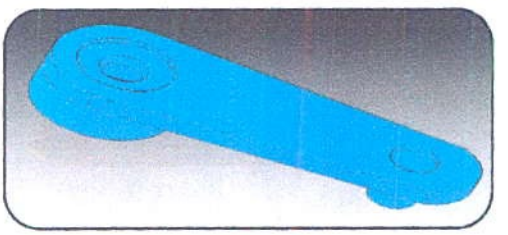


Fig 95. PALANCA SERVO

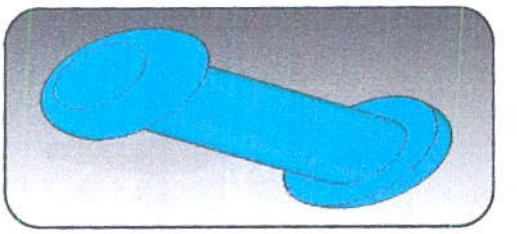


Fig 96. ESLABON

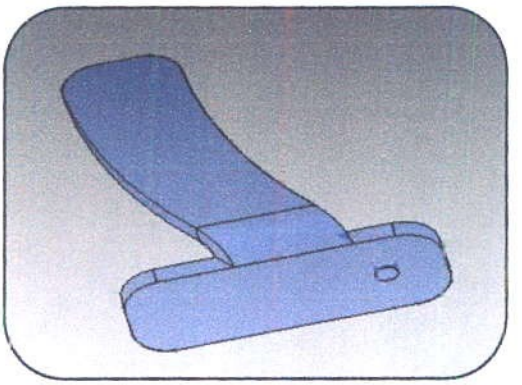


Fig 97. PIEZA B_FMII

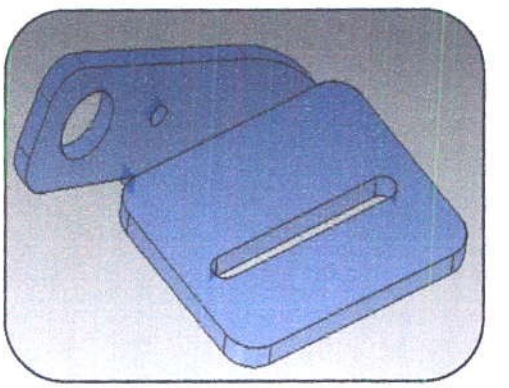


Fig 98, PIEZA P_FMII

- Se unen las piezas **PS5_FPII** y la **P_FMII** con un eslabon tipo remache doblecabeza en duraluminio. Posteriormente, el **Servomotor 5** se integra a la pieza **PS5_FPII**

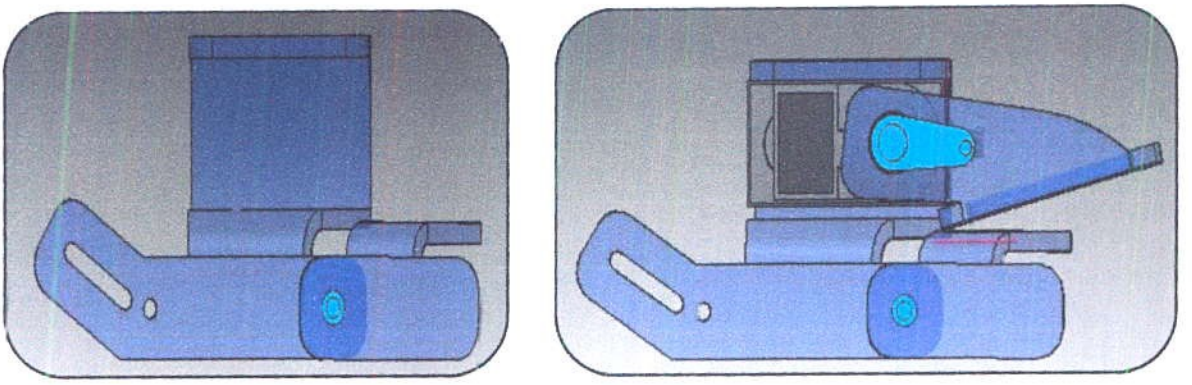


Figura 99. Ensambleje de piezas para el movimiento de la falange medial de II y III dedos.

- La palanca del servomotor se desplaza en forma circular, generando un movimiento angular de la pieza distal.

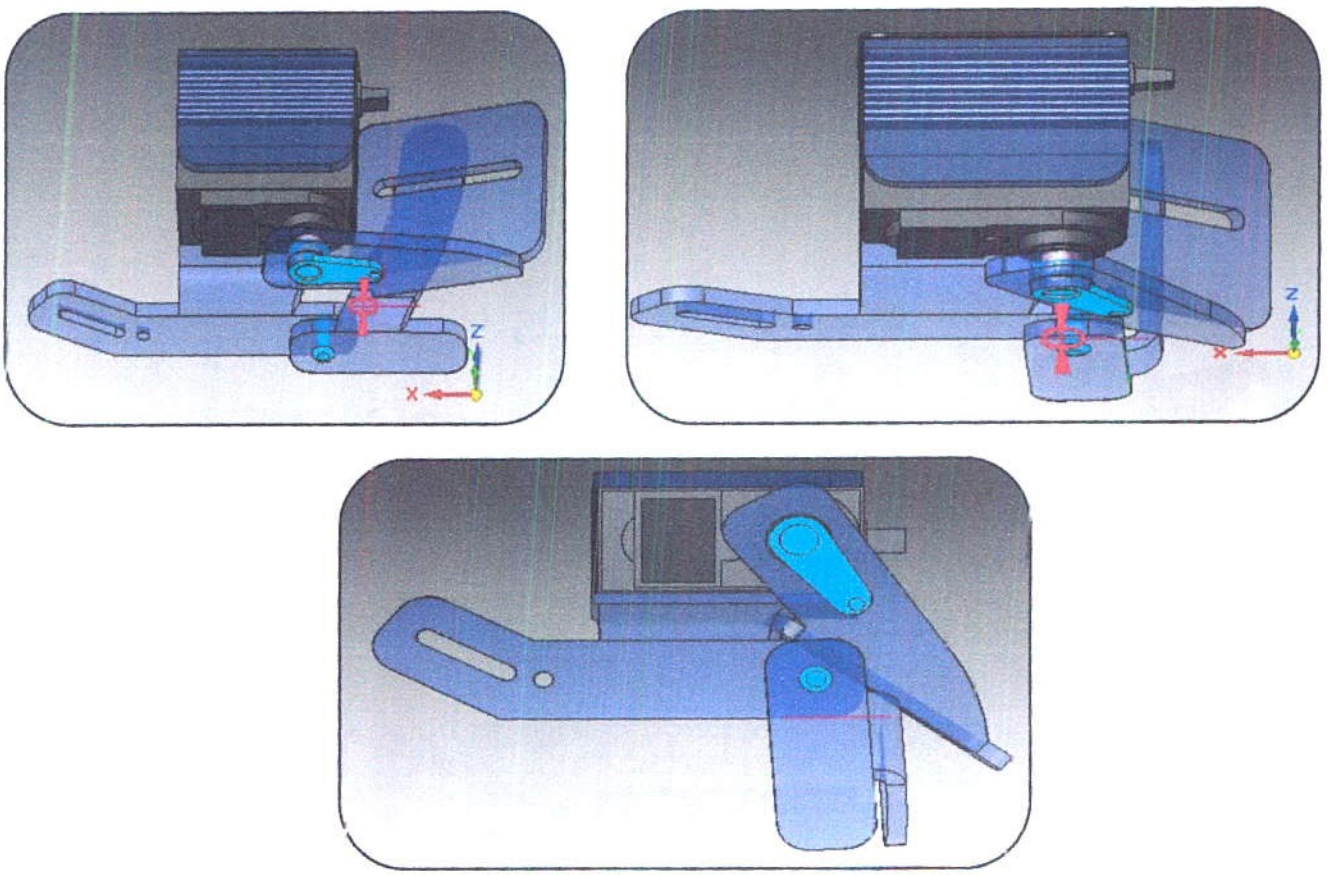


Figura 100. Ensambleje de piezas para el movimiento de la falange medial de II y III dedos.

3.6 Diseño movimiento articulaciónes metacarpofalangicas IV y V dedos

- Elejimos los componentes que se integran a la articulación paralela a la metacarpofalángica del I dedo, para simular el movimiento en el Software Solide Edge.

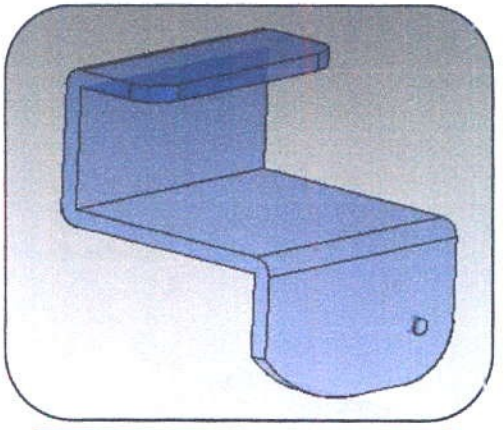


Fig 101. PIEZA BS6_MIV

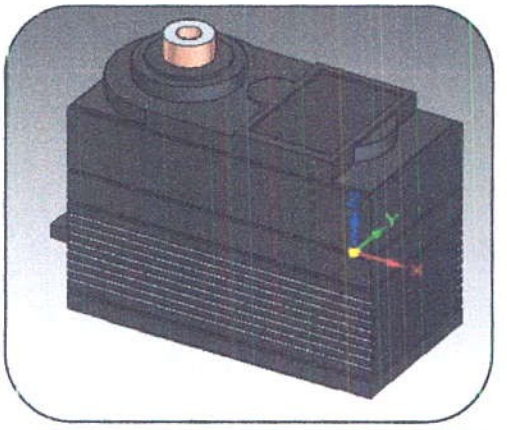


Fig 102. SERVOMOTOR

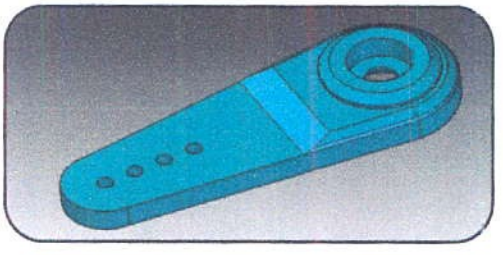


Fig 103. PALANCA SERVO

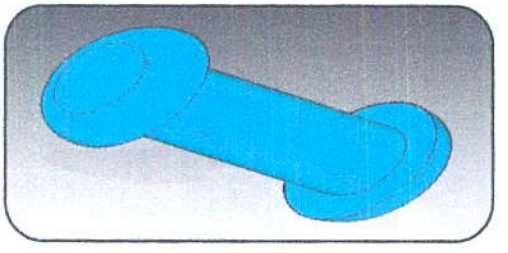


Fig 104. ESLABON

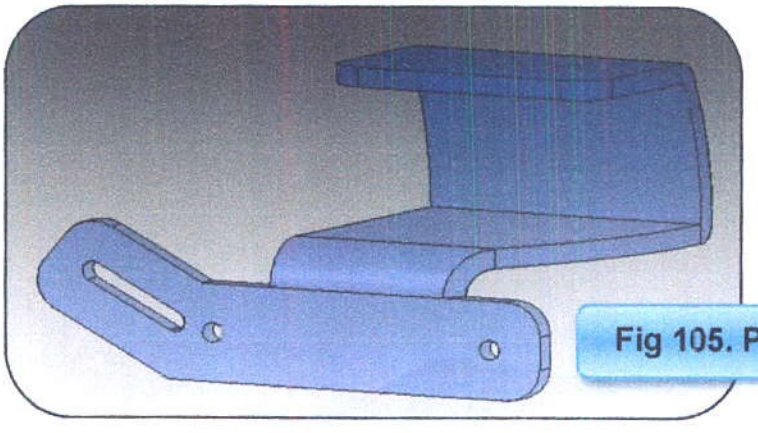


Fig 105. PIEZA PS7_FPIV

- Se unen las piezas **BS6_MIV** y la **PS7_FPIV** con un eslabon tipo remache doblecabeza en duraluminio. Posteriormente, el **Servomotor 5** se integra a la pieza **BS6_MIV**

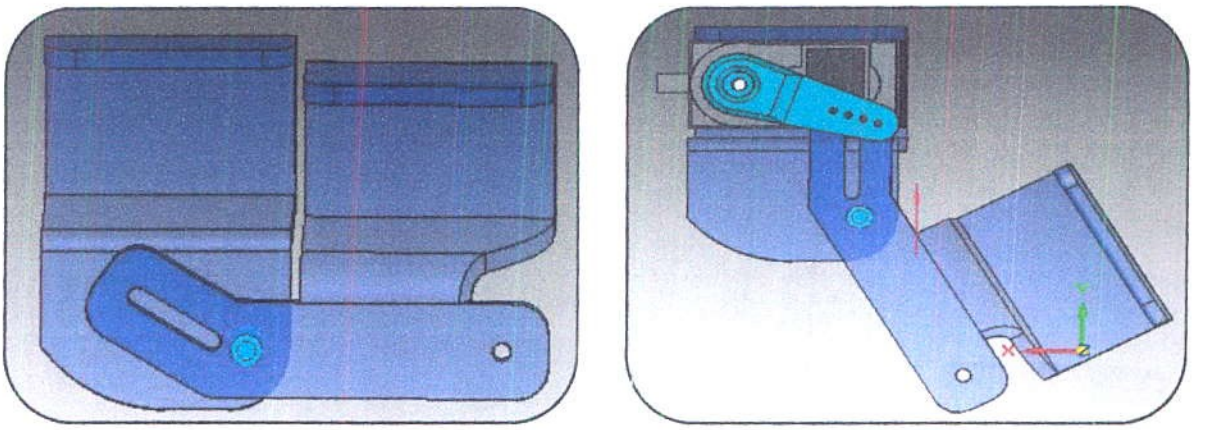


Figura 106. Ensambleje de piezas para el movimiento de la falange proximal de IV Y V dedos.

- La palanca del servomotor se desplaza en forma circular, generando un movimiento angular de la pieza distal.

3.7 Diseño movimiento articulación interfalangica proximal IV y V dedos

- Elejimos los componentes que se integran a la articulación paralela a la metacarpofalángica del I dedo, para simular el movimiento en el Software Solide Edge.

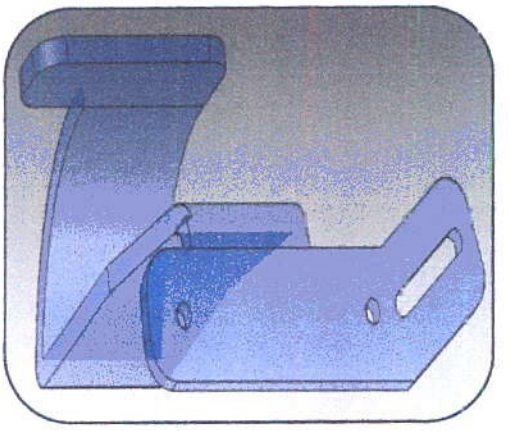


Fig 107. PIEZA PS7_FPIV

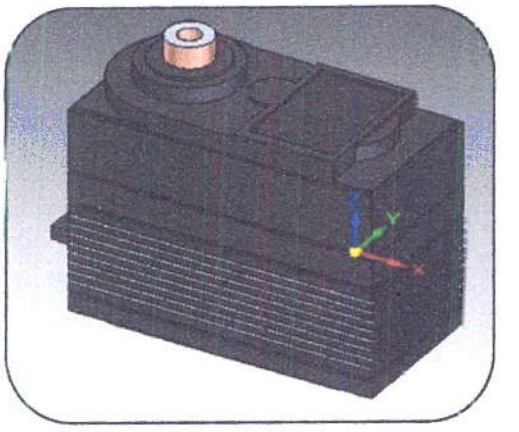


Fig 108. SERVOMOTOR

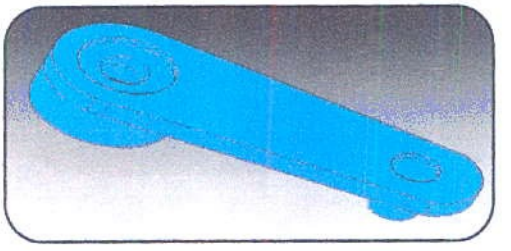


Fig 109. PALANCA SERVO

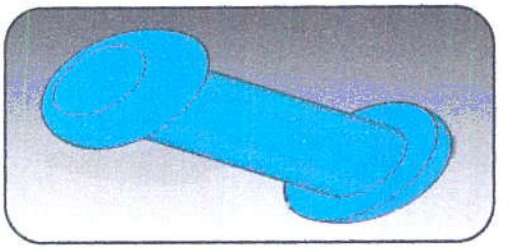


Fig 110. ESLABON

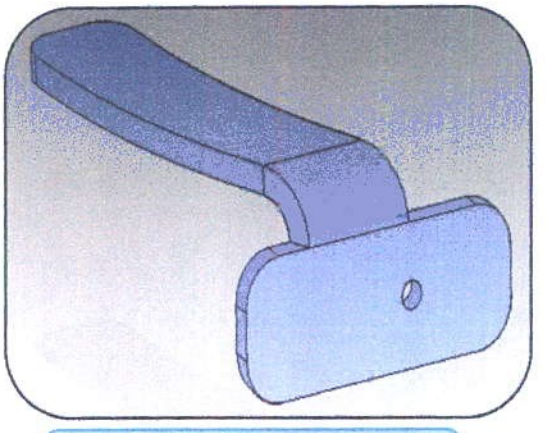


Fig 111. PIEZA B_FMII

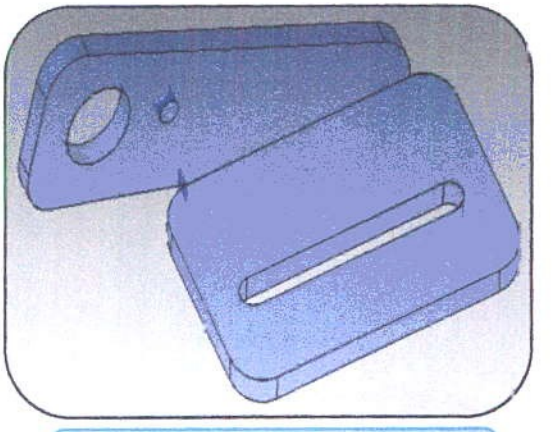


Fig 112. PIEZA P_FMIV

- Se unen las piezas **PS7_FPIV** y la **P_FMIV** con un eslabon tipo remache doblecabeza en duraluminio. Posteriormente, el **Servomotor 5** se integra a la pieza **PS7_FPIV**

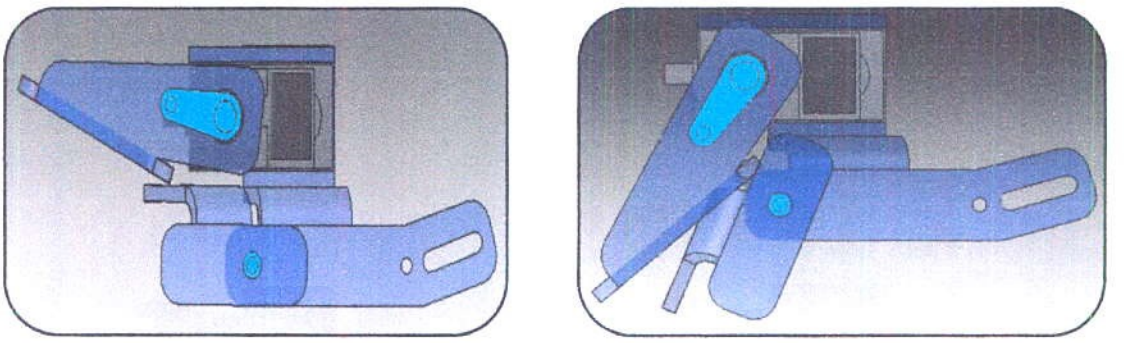


Figura 113. Ensambleje de piezas para el movimiento de la falange medial de IV Y V dedos.

- La palanca del servomotor se desplaza en forma circular, generando un movimiento angular de la pieza d stal.

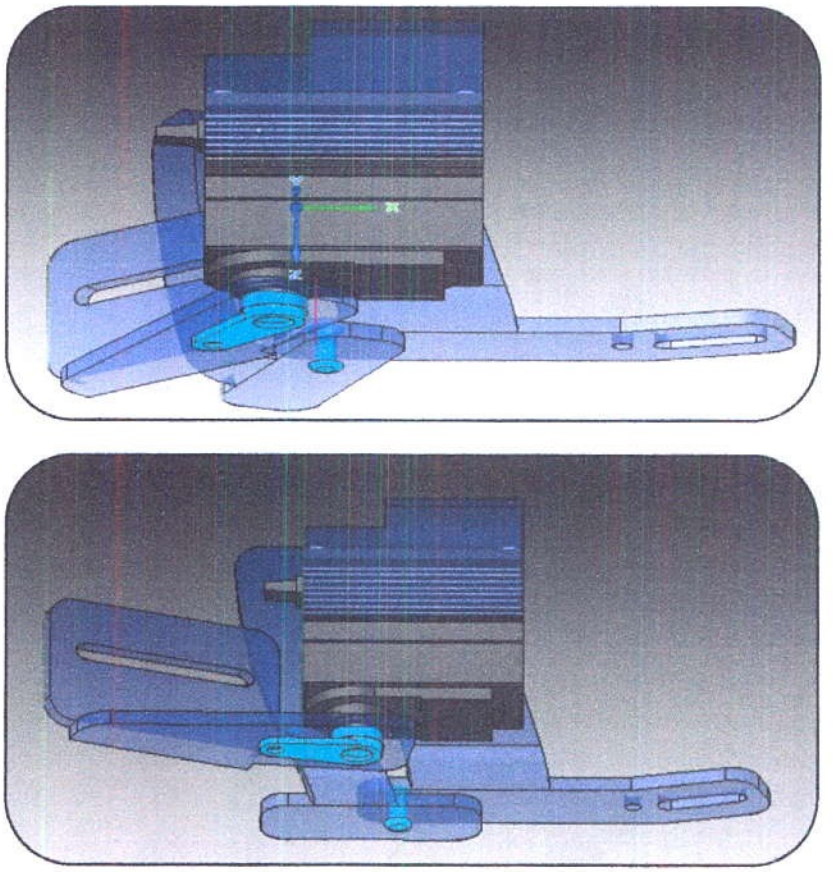


Figura 114. Ensambleje de piezas para el movimiento de la falange medial de IV Y V dedos.

3.8 Diseño movimiento muñeca

- Elegimos los componentes que debe tener la simulación de la articulación que dará movimiento al I metacarpiano, este representa el movimiento circular del I dedo, para lograr la prensión de objetos.

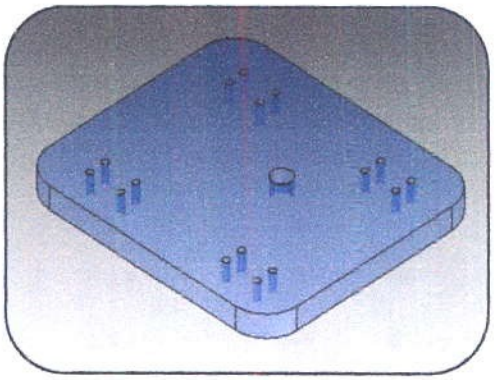


Fig 115. PIEZA BS8_MN

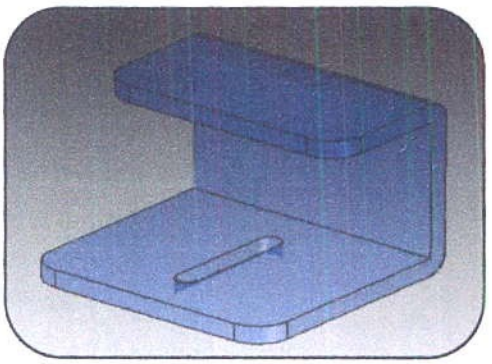


Fig 116. PIEZA PS8_MN

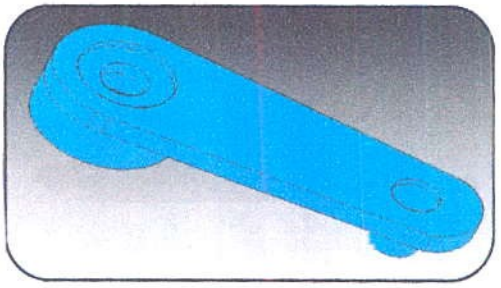


Fig 117. PALANCA SERVO

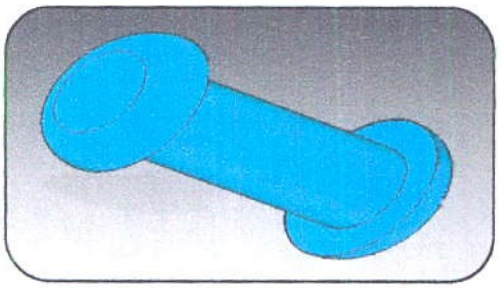


Fig 118. ESLABON

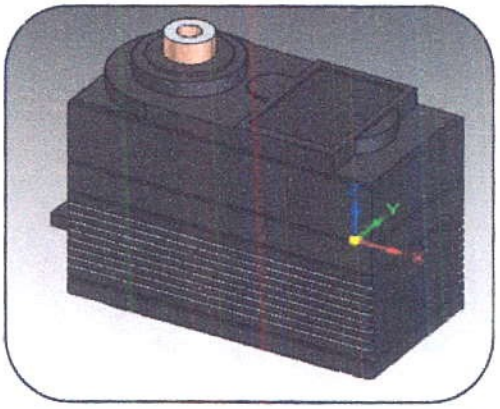


Fig 119. SERVOMOTOR

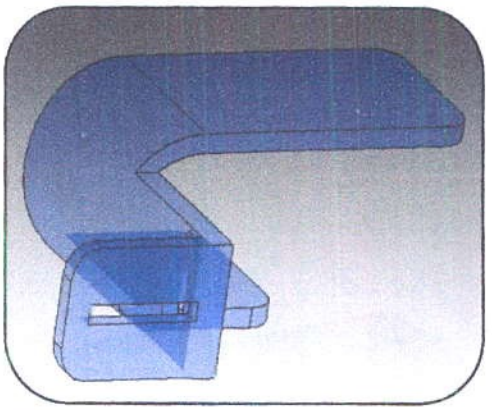


Fig 120. PIEZA PLS8_MN

- El remache en duraluminio tiene alineación axial con la base en acrílico.

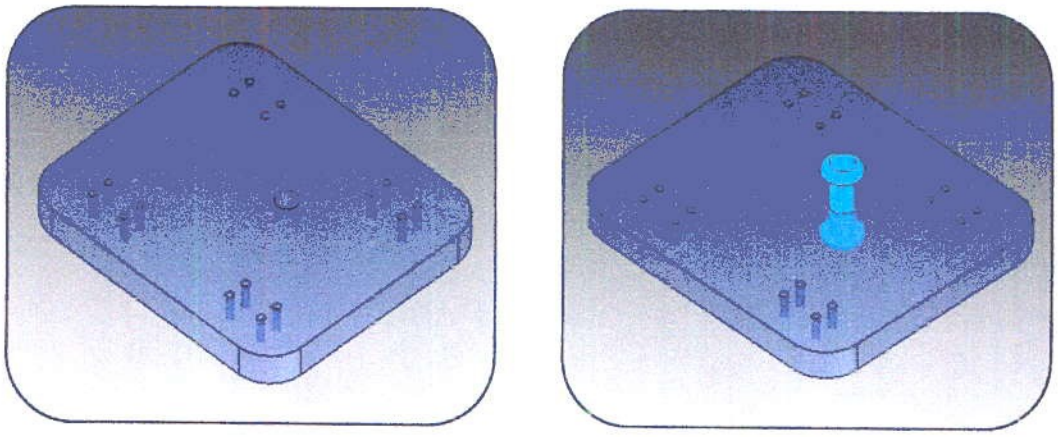


Figura 121. Ubicación del remache de duraluminio en la Pieza BS8_MN

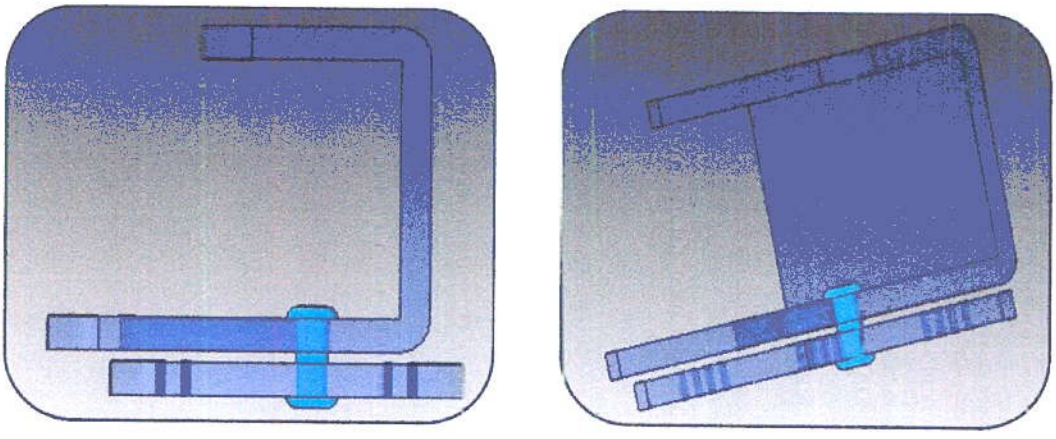


Figura 122. Ubicación del remache de duraluminio en la Pieza PS8_MN y

- El **Servomotor 8** se integra a las bases de acrílico mediante cinta doble fax transparente.

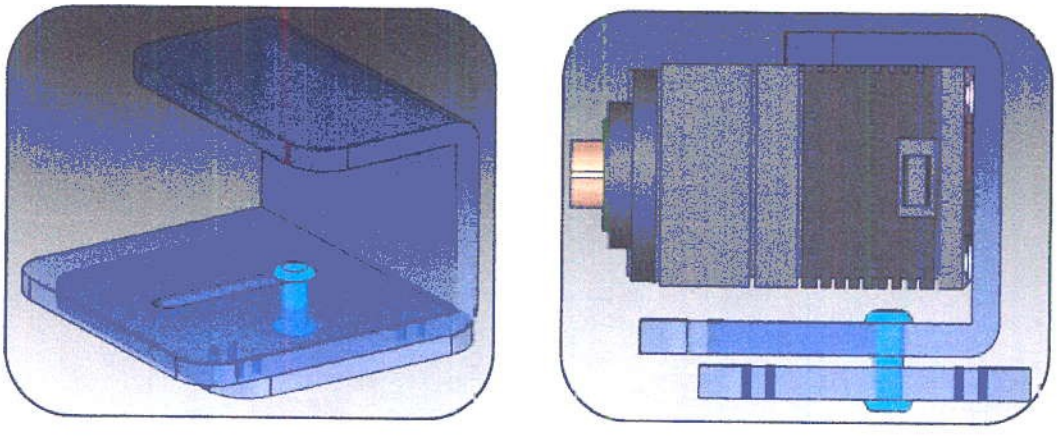


Figura 123. Ensamble del Servomotor 1 a las piezas BS8_MN y PS8_MN

- La ubicación del servomotor es estratégica para definir el movimiento y ensamblamos la pieza **PS8_MN** al conjunto, unida por la palanca del Servomotor 8

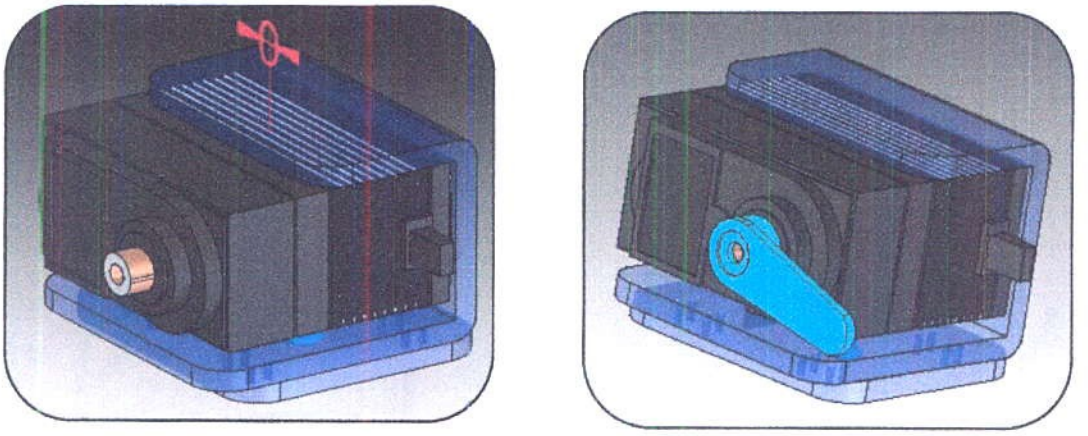


Figura 124. Ensamble del Servomotor 8 a las piezas BS8_MN y PS8_MN

- Finalmente integramos la palanca en acrílico **PLS8_MN** que dará movimiento en flexión y extensión a la muñeca.

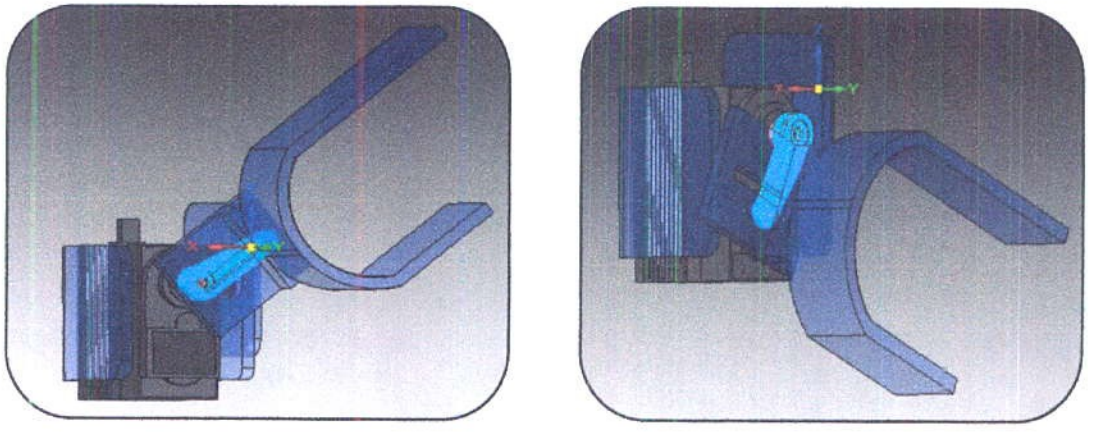


Figura 125. Ensamble del Servomotor 8 a las piezas BS8_MN y PS8_MN

3.9 Secuencia ensamblaje exoesqueleto

- Ensamblaje de Piezas **BS1_MN** y **BS8_MN** unidas a las piezas metálicas en acero inoxidable que dan soporte a la muñeca.

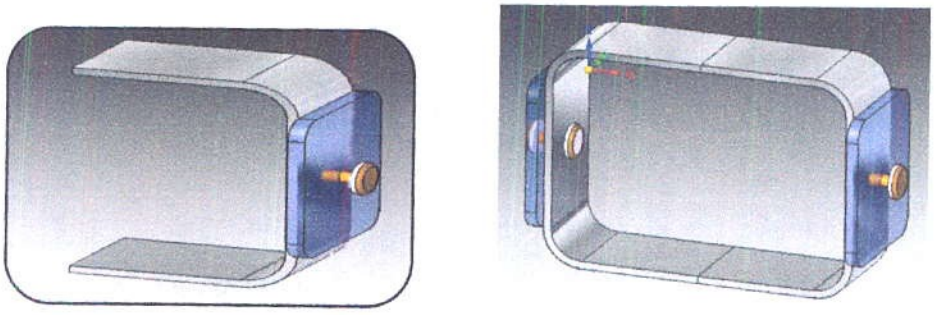


Figura 126. Ensamble de las piezas BS1_MN y BS8_MN

- Ensamblaje de Piezas **PS1_MN** y **PS8_MN** unidas a las piezas Piezas **BS1_MN** y **BS8_MN**

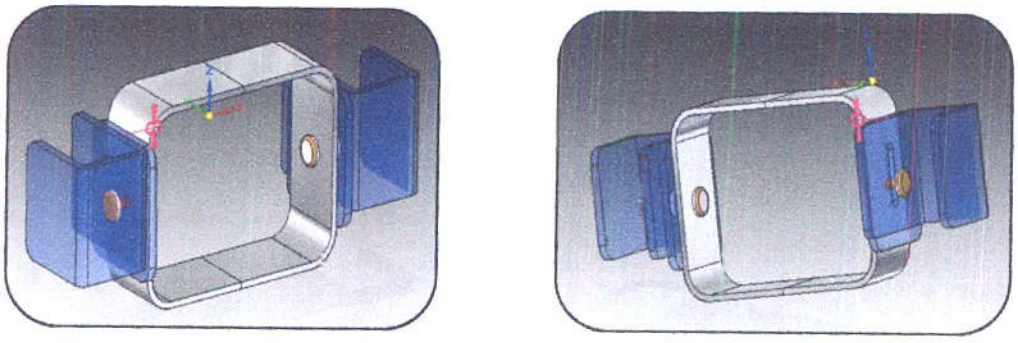
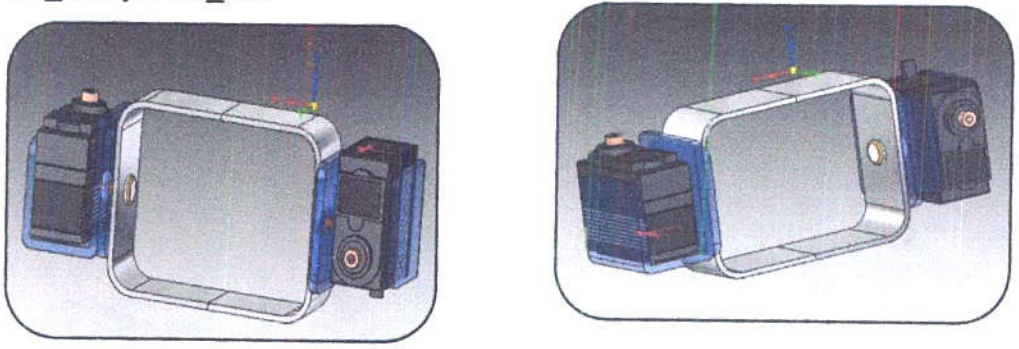


Figura 127. Ensamble de las piezas PS1_MN y PS8_MN

- Ensamblaje de Piezas **Servomotor 1** y **Servomotor 8** unidas a las Piezas **PS1_MN** y **PS8_MN**



- **Figura 128. Ensamble de los Servomotores 1 y 8**

- Ensamblaje de Piezas **PS2_MI** y **Servomotor 2** a la secuencia.

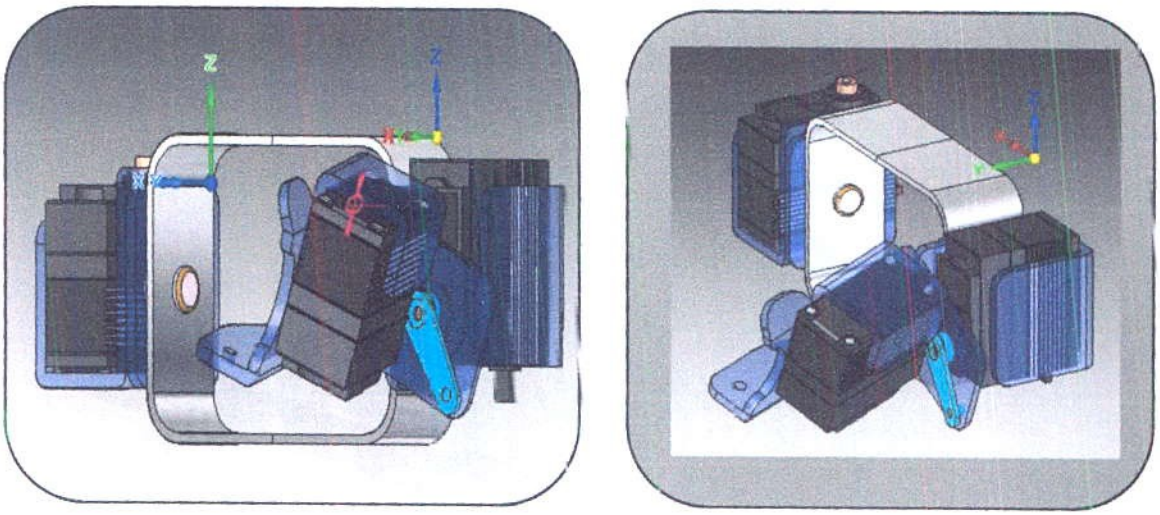


Figura 129. Ensamble de la pieza PS2_MI y Servomotor 2

- Ensamblaje de Piezas **PS3_FPI** a la secuencia:

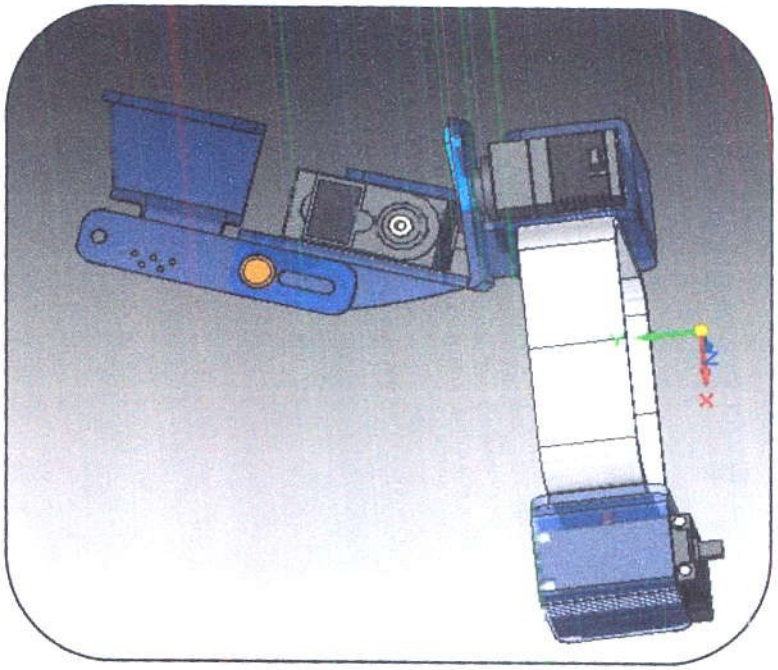


Figura 130. Ensamble de las piezas PS3_FPI

- Ensamblaje del **Servomotor 3** y la Pieza **BP_FDI** a la secuencia:

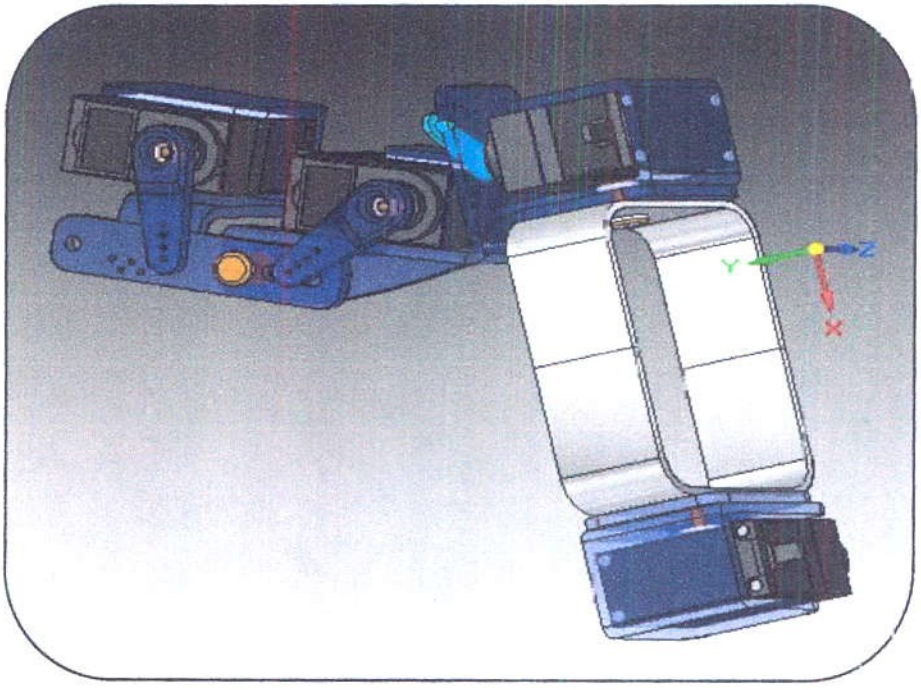


Figura 131. Ensamble del Servomotor 3 (Vista Lateral Interna)

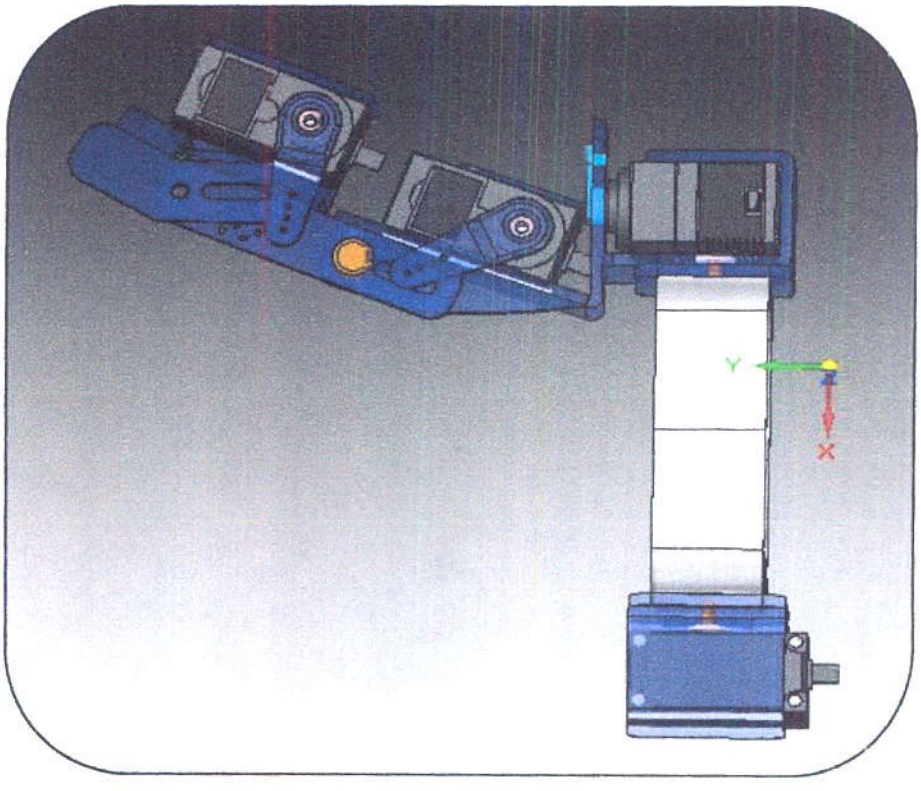


Figura 132. Ensamble del de la Pieza BP_FDI

- Ensamblaje del Piezas **BS4_MII** y **BS6_MIV** a la secuencia:

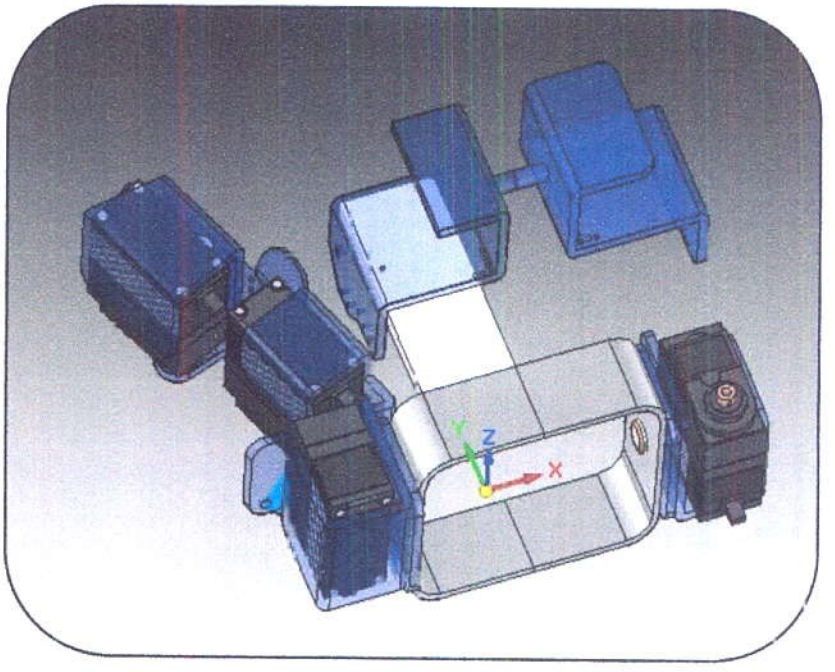


Figura 133. Ensamble de Piezas BS4_MII y BS6_MIV

- Ensamblaje del Servomotores 4 y 6 a la secuencia:

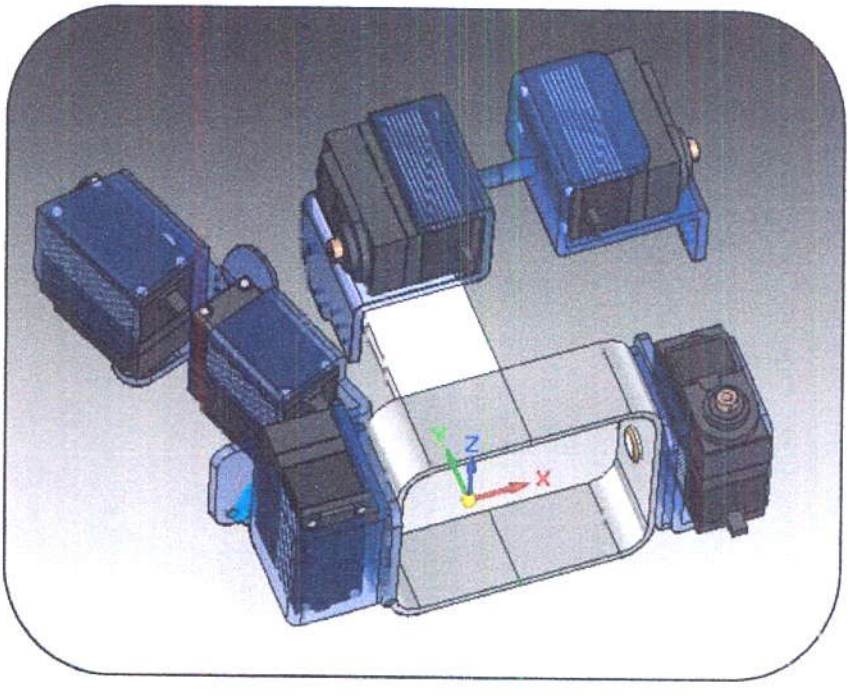


Figura 134. Ensamble de Servomotores 4 y 6

- Ensamblaje del Piezas **PS3_FPII** y **PS7_FPIV** a la secuencia:

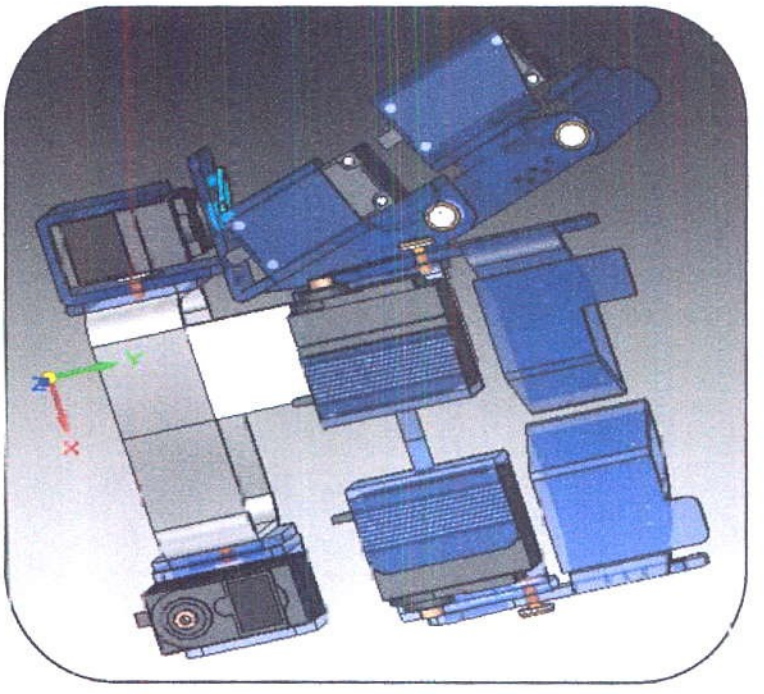


Figura 135. Ensamble de Piezas PS3_FPII y PS6_FPIV

- Ensamblaje del Servomotores 5 y 7 a la secuencia:

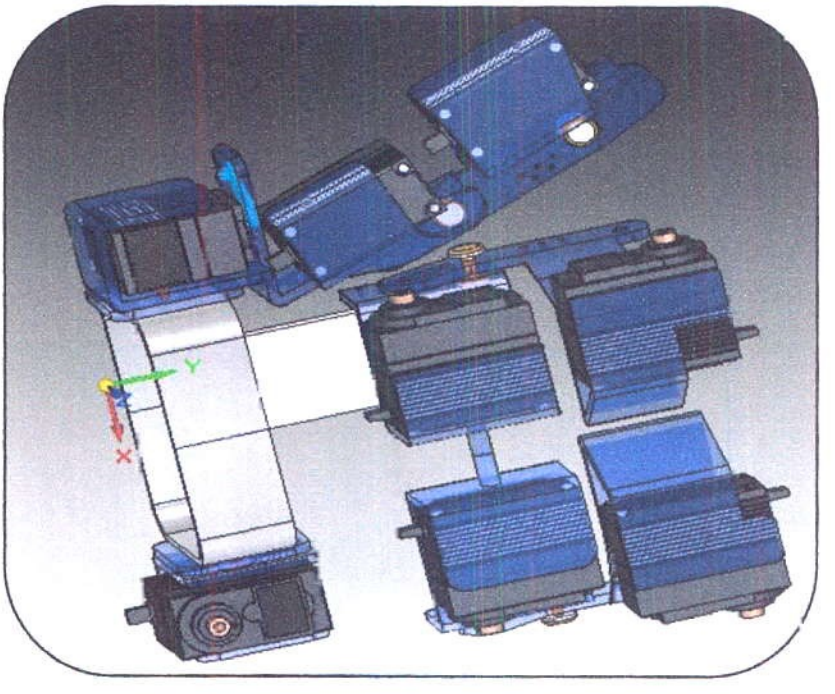


Figura 136. Ensamble de Servomotores 4 y 6

- Ensamblaje del Piezas **P5_FMII** y **P7_FMIV** a la secuencia:

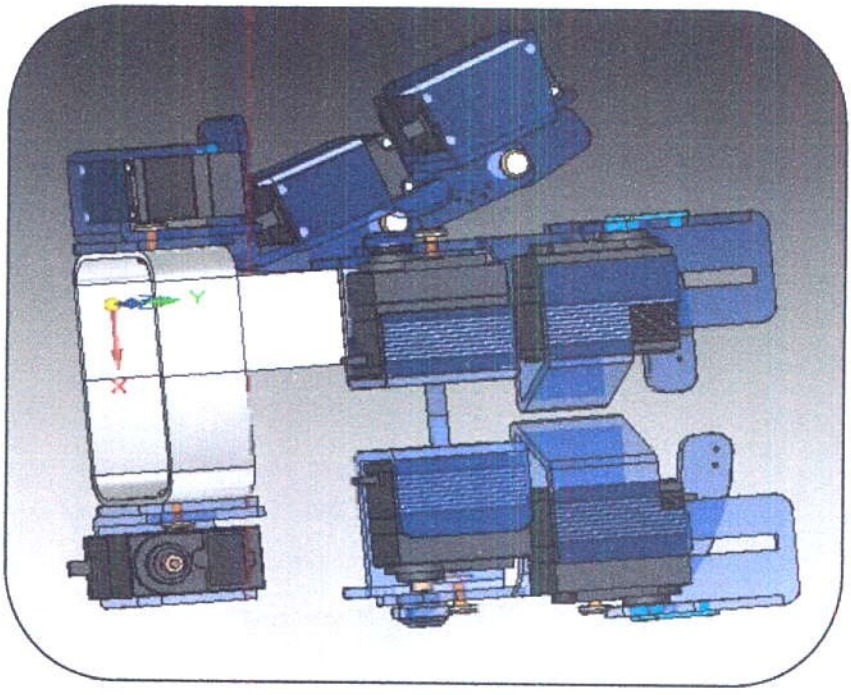


Figura 137. Ensamble de Piezas P5_FMII y P7_FMIV (cara dorsal)

- Ensamblaje del Servomotores 5 y 7 a la secuencia:

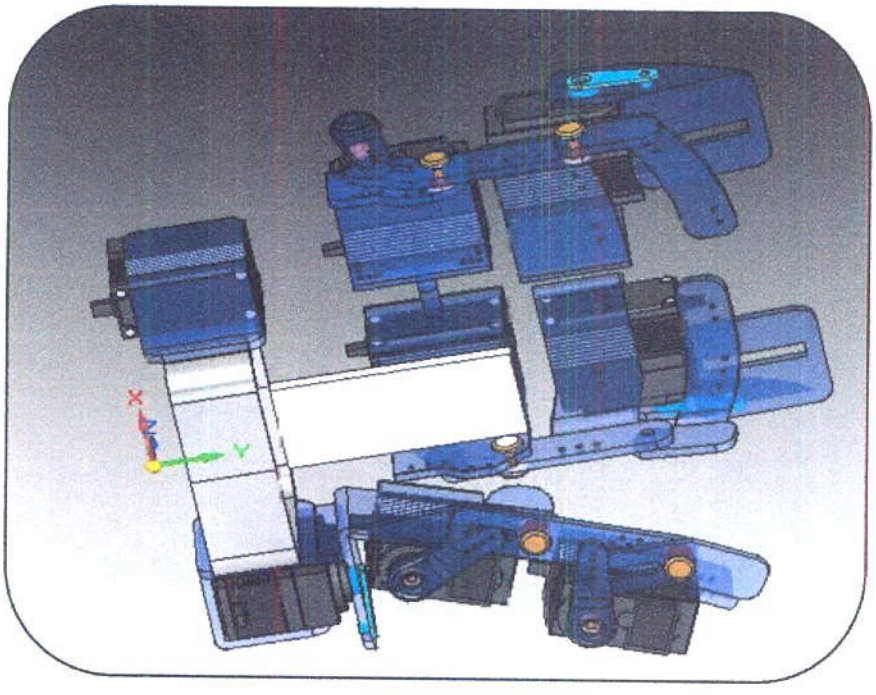


Figura 138. Ensamble de Piezas P5_FMII y P7_FMIV (cara ventral)

- Ensamblaje del Piezas **PLS8_MN** a la secuencia:

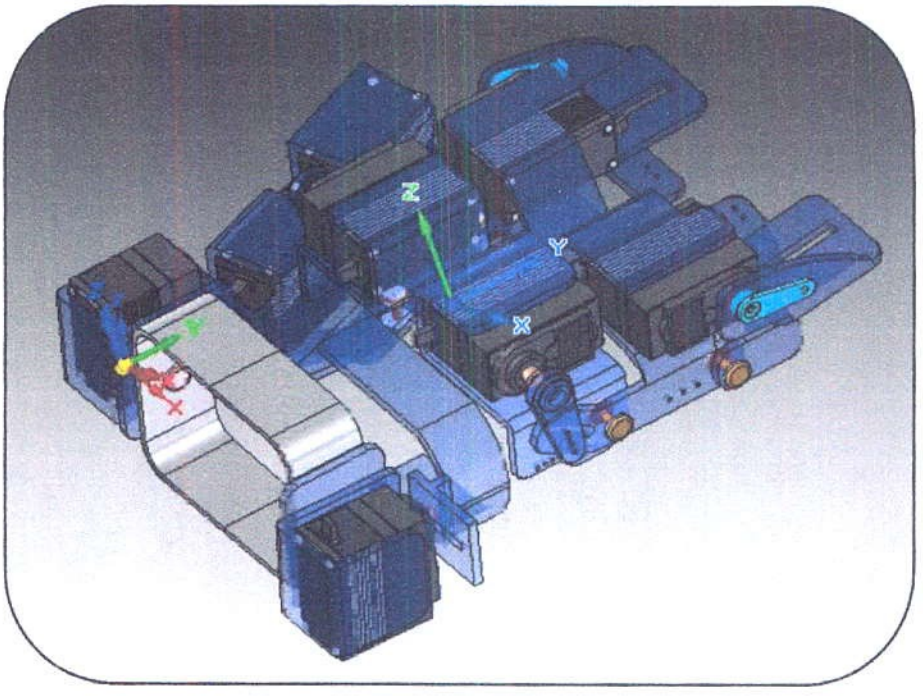


Figura 139. Ensamble de Piezas PLS8_MN (cara dorsal)

- Ensamblaje del Servomotores 5 y 7 a la secuencia:

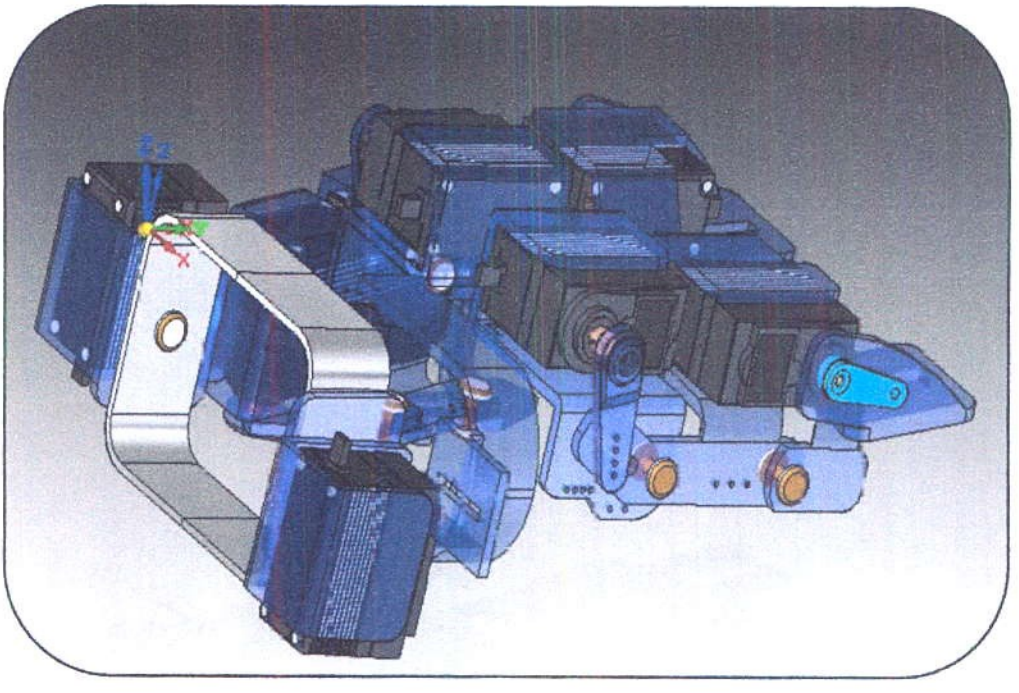


Figura 140. Ensamble de Piezas PLS8_MN (cara lateral externa)

3.9 Fotos del producto terminado

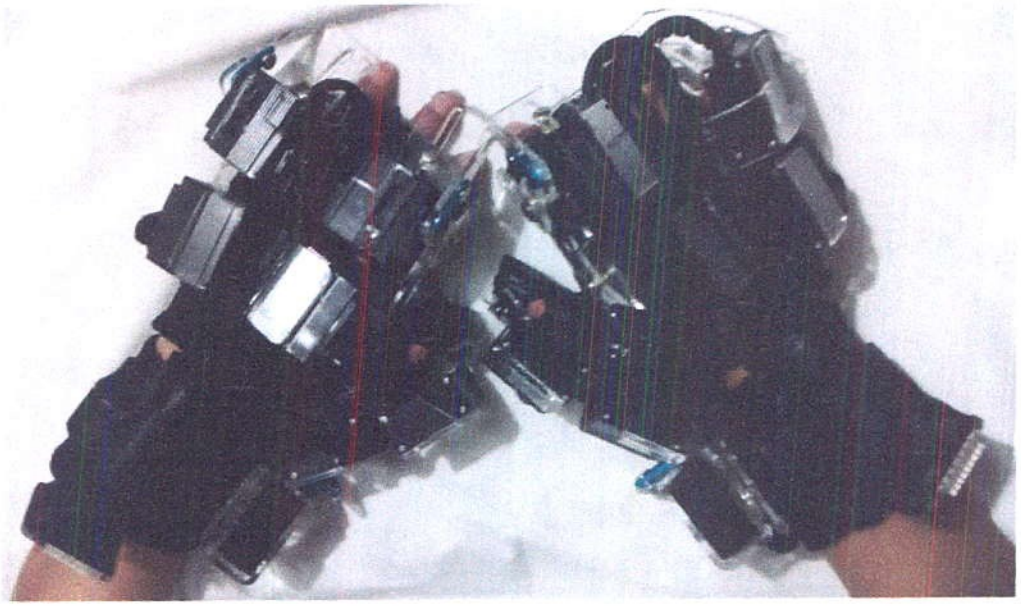


Figura 141. Exoesqueleto Mecatrónico (ambas manos – cara dorsal)

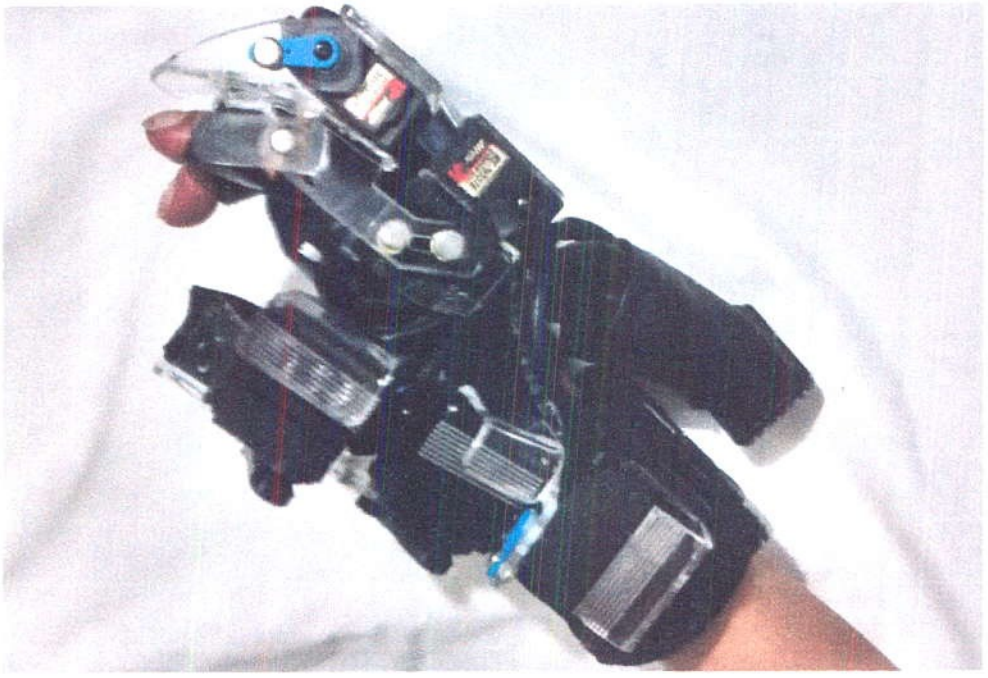


Figura 142. Exoesqueleto Mecatrónico (mano derecha – cara interna)



Figura 143. Exoesqueleto Mecatrónico (mano derecha – cara palmar)

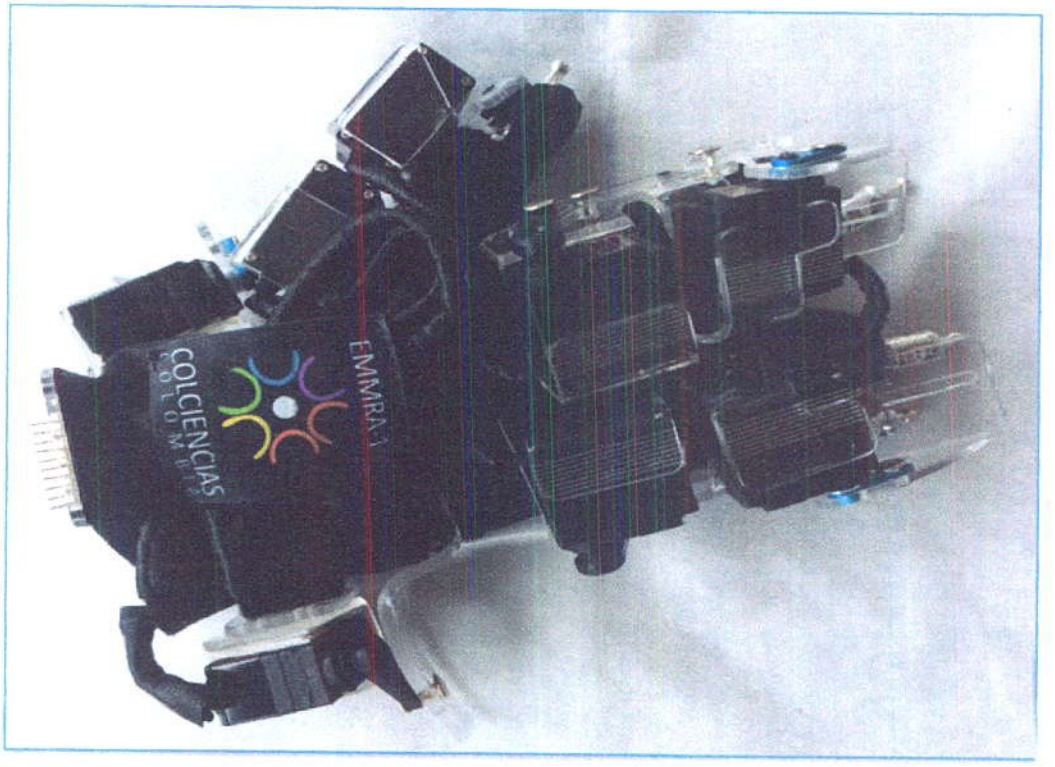


Figura 144. Exoesqueleto Mecatrónico (mano izquierda– cara dorsal)

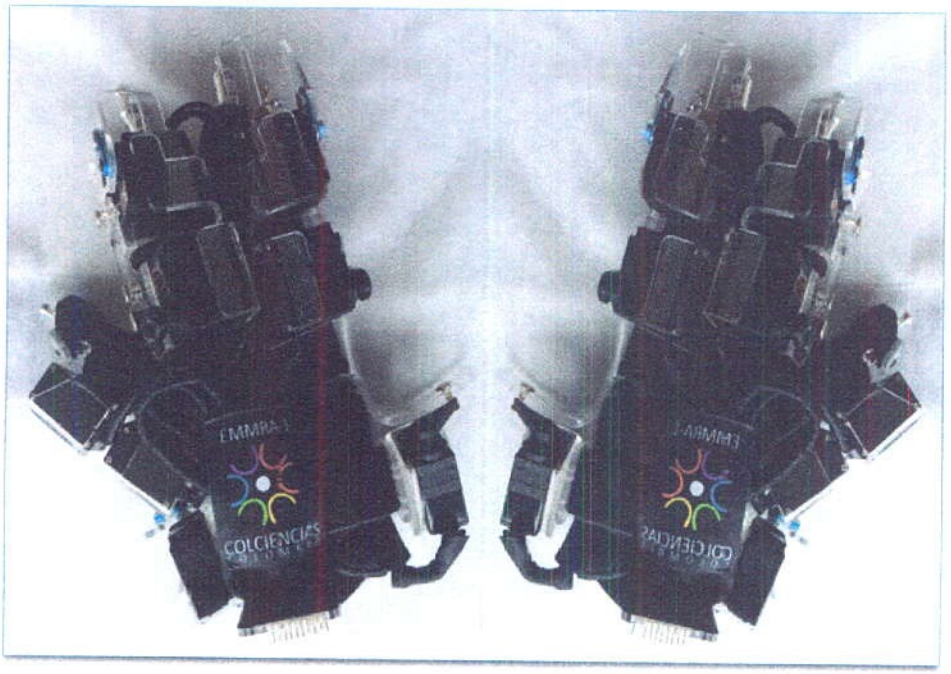


Figura 145. Exoesqueleto Mecatrónico (ambas manos– cara dorsal)

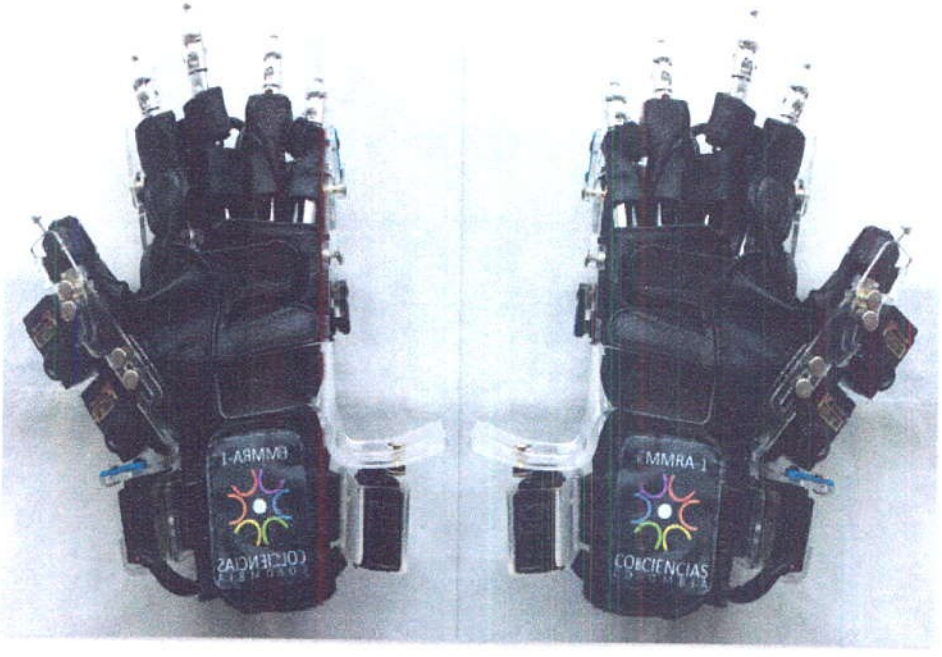


Figura 146. Exoesqueleto Mecatrónico (ambas manos– cara palmar)

4 IMPLEMENTACIÓN SENSORES E INDICADORES:

4.1 Sensor de Fuerza Resistivo:

El sensor de fuerza resistivo (**FSR**) es un dispositivo de película de polímero (PTF) que presenta una disminución de la resistencia cuando aumenta la fuerza aplicada a la superficie activa. Su sensibilidad a la fuerza está optimizada para uso en el control por toque humano de dispositivos electrónicos. Las FSRs no son células de carga o galgas extensiométricas aunque tengan propiedades similares. Las FSRs no son adecuadas para medidas de precisión.

Para nuestro proyecto utilizamos Modelo FSR 402, con un área activa circular de12.7mm de diámetro, espesor nominal de 0,46mm, capa semiconductora, espacio adhesivo de acrílico, capa conductiva.

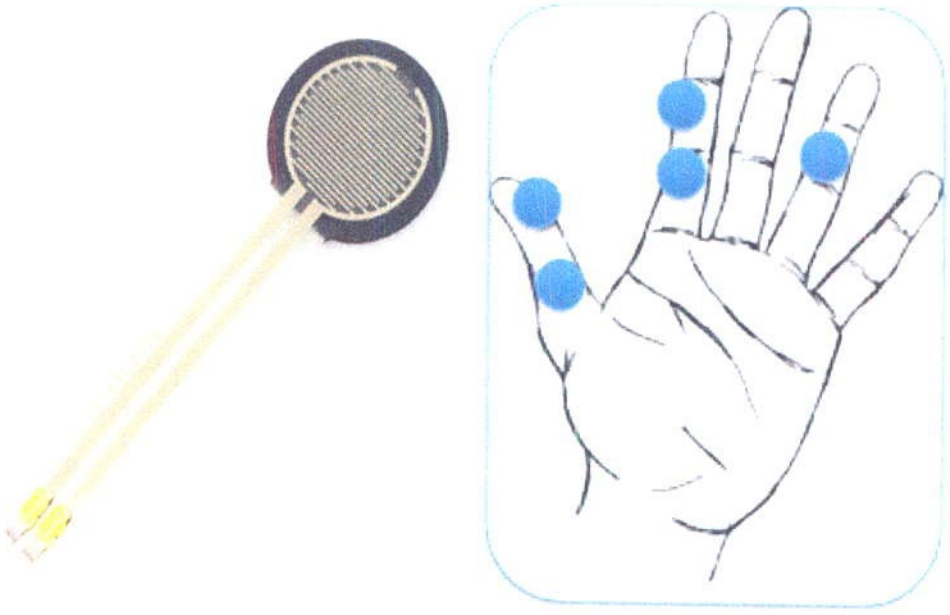


Figura 147. Sensores de Fuerza Resistivos y su ubicación en la mano

En cada mano se ubicaron cinco (5) **FSR** en las zonas de contacto digital con objetos:

- 1 FSR en la cara palmar de la I falange del I dedo.
- 2 FSR en la cara palmar de la II falante del I dedo.
- 3 FSR en la cara palmar de la I falange del II dedo.
- 4 FSR en la cara palmar de la II falange del II dedo.
- 5 FSR en la cara palmar de la II falange del IV dedo.

4.2 Indicadores Led

El módulo de reconocimiento de voz, a través de un sonido indica al usuario en que momento pronunciar un comando de voz, para que el exoesqueleto realice la una acción como abrir, cerrar, agarrar, soltar, etc. Sin embargo, hay un tiempo de milisegundos en el que el EasyVR shield 3.0 no reconoce ninguna instrucción, con el indicador Led de alta luminosidad de color verde, se muestra al usuario el tiempo en que puede pronunciar un comando de voz, cuando se termina este tiempo, se enciende el Led de alta luminosidad de color rojo, durante este tiempo, el módulo está ejecutando el ciclo de programa y no reconocerá ningún sonido pronunciado por el usuario.

Los indicadores Led se ubican en una tarjeta electrónica que sirve como nodo de interconexión entre los cables que conectan los servomotores y los sensores FSR con el módulo de control del dispositivo.

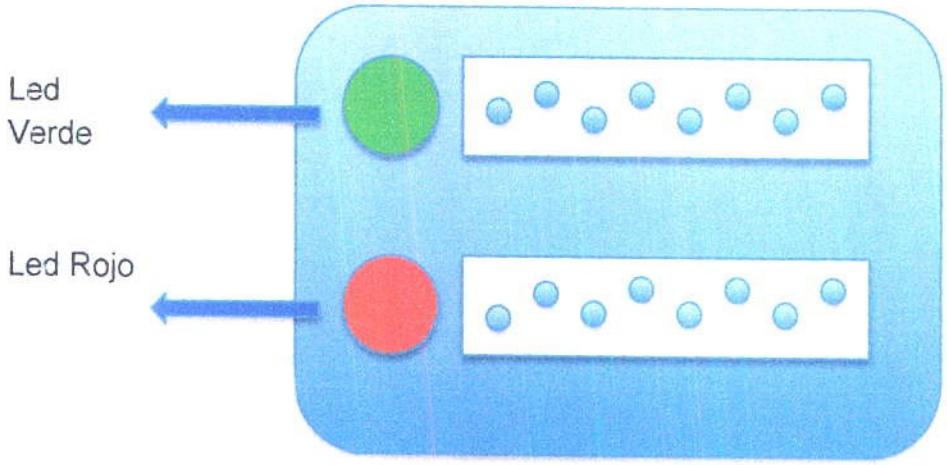


Figura 148. Diseño del Nódo de Interconexión

Led Verde: Tiempo en el que el usuario puede pronunciar los comandoz de voz pregrabados en el módulo de reconocimiento de voz EasyVR Shield 3.0

Led Rojo: Tiempo en el que el usuario el módulo de reconocimiento de voz está procesando la información o indicando instrucciones al usuario, las palabras del usuario no serán reconocidas por el sistema.

Se realizó el montaje de un circuito impreso para servir de nodo entre los múltiples cables que provienen del exoesqueleto de mano y su interconexión con el módulo de control electrónico.

El diseño del nodo se realizo en el Software ISIS – ARES versión libre para estudiantes.

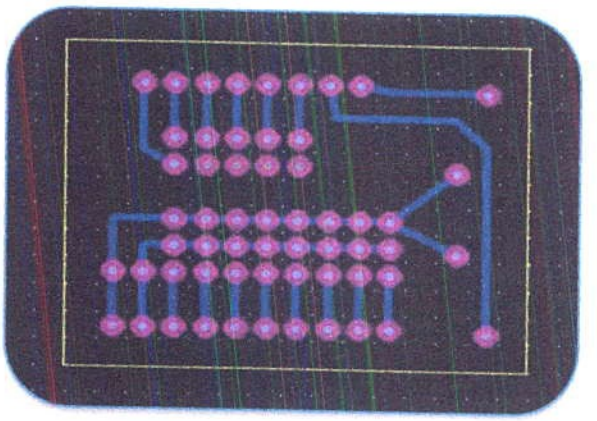


Figura 149. Diseño del circuito electrónico en Software ISIS - ARES

Posteriormente se pasa a un formato PDF para ser impreso en una Impresora Laser EPSON.

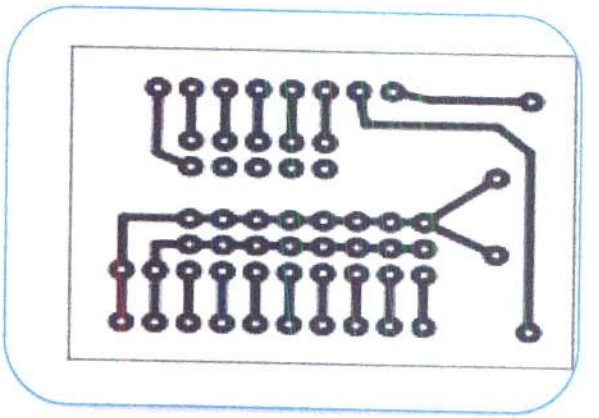


Figura 150. Diseño del circuito electrónico Nodo en PDF

Finalmente se pasa el circuito a una vaquela en fibra de vidrio, para obtener la tarjeta electrónica que se convertirá al nodo de interconexión.

Finalmente se pasa el circuito a una vaquela en fibra de vidrio, para obtener la tarjeta electrónica que se convertirá al nodo de interconexión.

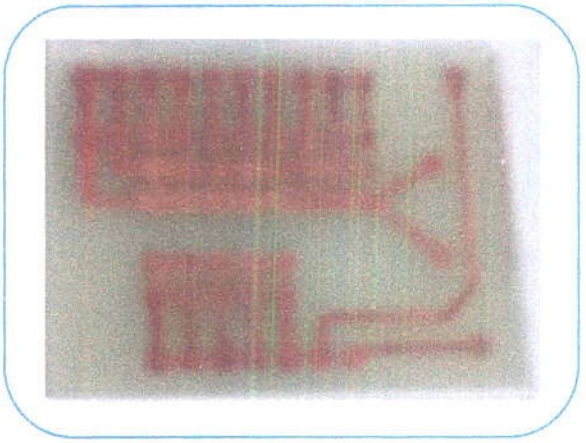


Figura 151. Diseño del circuito electrónico vaquela de fibra de vidrio

Se realiza el ensamblaje de conectores, utilizando pines y conectores GP de 8 y 10 pines, para realizar la interconexión

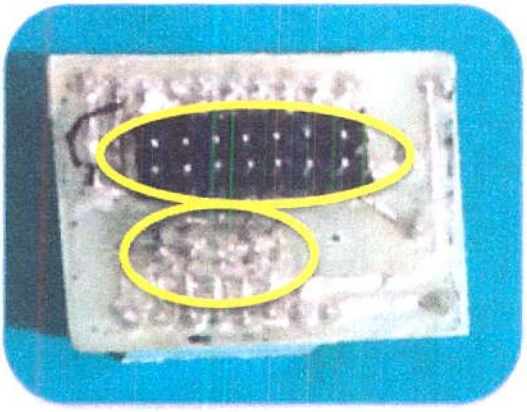


Figura 152. Pines interconectores con los servomotores y los SFR (cara distal)

Ubicamos el Led Verde en la parte superior del nodo y el Led Rojo en la parte inferior del nodo. Acoplamos los conectores GP para la interconexión de los cables con el módulo de control del dispositivo.



Figura 153. Pines GP para interconectarse con los servomotores y los SFR (cara proximal)

Pegamos cinta adhesiva blanca doble cara, para acoplar la Pieza en acrílico que dará mejor apariencia estética al dispositivo.

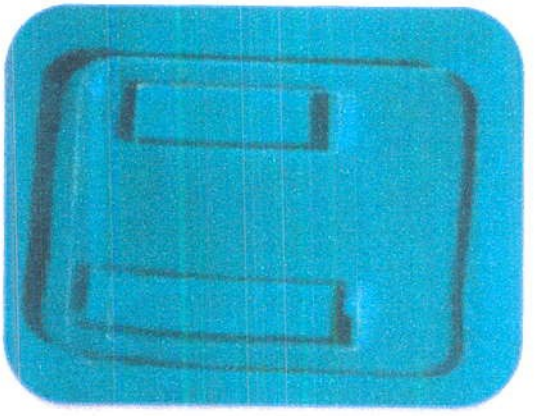


Figura 154. Pieza en acrílico para cubrir el nodo

El nodo tiene una cara externa que permite acoplar los 3 cables de cada servomotor (negro – tierra, rojo – voltaje, amarillo – señal), en total 24 cables de los 8 servomotores. Igualmente permite acoplar los 10 cables de los Sensores de Fuerza Resistivos (voltaje y tierra), además, permite acoplar los cables de voltaje y tierra que alimentan los Servomotores y los SFR.

El nodo tiene una cara interna, que permite dar continuidad a la comunicación de todos los cables descritos anteriormente, con el módulo de control del dispositivo, esto se fabricó con cables vehiculares de bajo calibre (18F) y cables ribon (para los SFR y señal de los servomotores). Todos cubiertos por una vaina en cuero textil, cuyos extremos se interconectan con los nódulos a través de pines.

El nodo es una plataforma elaborada en una baquela de fibra de vidrio, con circuito impreso para que en cada mano, los sensores queden al lado interno y puedan ser vistos fácilmente por el usuario.

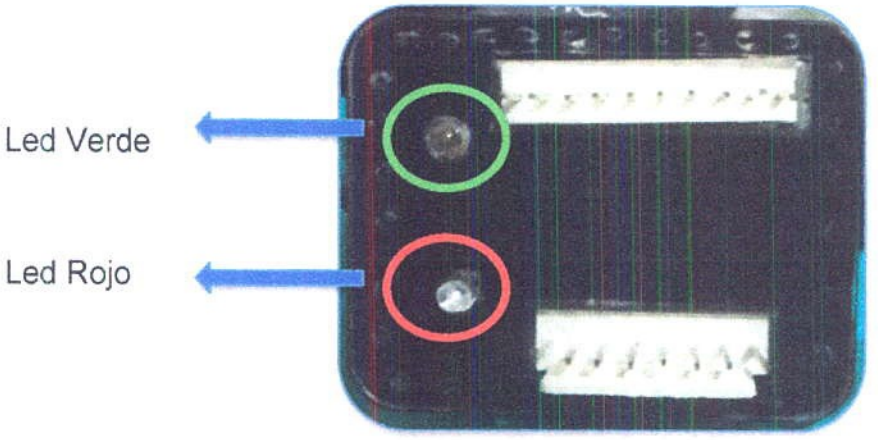


Figura 155. Nodo de interconexión cara interna

Los conectores GP en la parte superior reciben de izquierda a derecha:

- 1 pin: Conexión negativa del Voltaje para los Servomotores
- 2 pin: Conexión positiva del Voltaje para los Servomotores
- 3 pin: Señal para el movimiento del Servomotor 8
- 4 pin: Señal para el movimiento del Servomotor 7
- 5 pin: Señal para el movimiento del Servomotor 6
- 6 pin: Señal para el movimiento del Servomotor 5
- 7 pin: Señal para el movimiento del Servomotor 4
- 8 pin: Señal para el movimiento del Servomotor 3
- 9 pin: Señal para el movimiento del Servomotor 2
- 10 pin: Señal para el movimiento del Servomotor 1

Los conectores GP en la parte inferior reciben de izquierda a derecha:

- 1 pin: Conexión de Voltaje del Led Rojo
- 2 pin: Conexión de Voltaje del Led Verde
- 3 pin: Señal del SFR 1
- 4 pin: Señal del SFR 2
- 5 pin: Señal del SFR 3
- 6 pin: Señal del SFR 4
- 7 pin: Señal del SFR 5
- 8 pin: Conexíon de Voltaje 5 Voltios para alimentar los SFR

4.3 Sensor de señal muscular:

Los sensores de señal muscular se encuentran ubicados en el antebrazo del paciente, recogen la información de tres electrodos (Rojo, Azul y Negro). Mide la diferencia de presión entre los electrodos rojo y azul, los convierte en un voltaje y los envía al módulo de control electrónico (Tarjeta Arduino Mega 2650). El electrodo negro es la tierra del paciente al circuito electrónico.

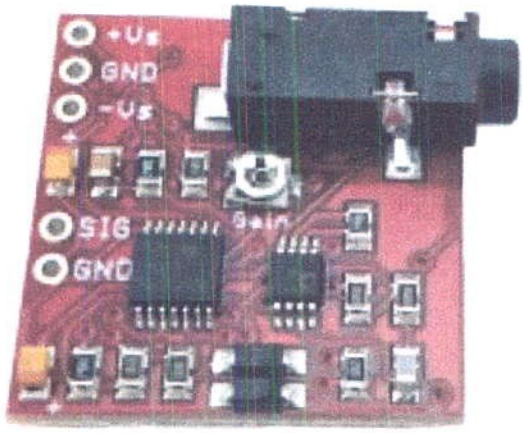


Figura 156. Sensor de Electromiografía

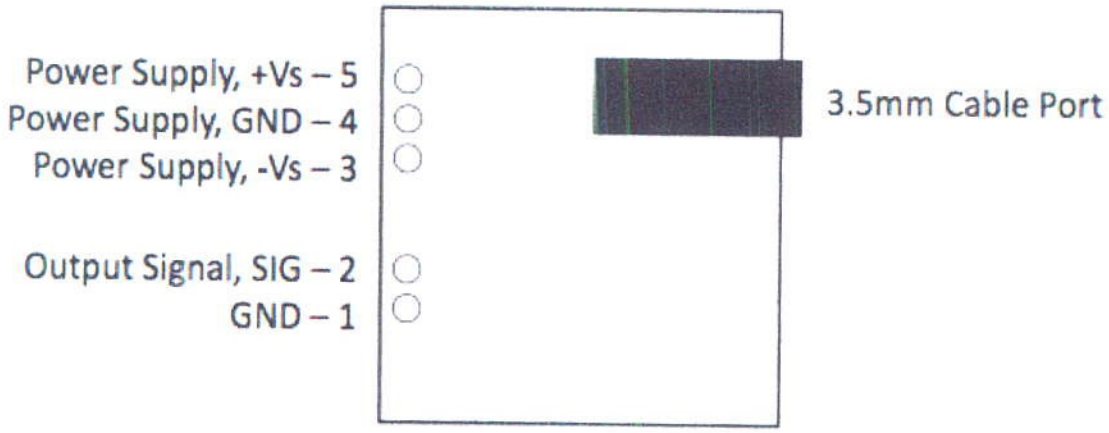


Figura 157. Entradas y Salidas del Sensor de Electromiografía

4.3 Fotos del producto terminado:

- Ubicación de los Sensores de Fuerza Resistivos en el guante:

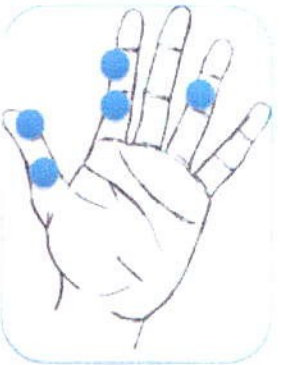


Figura 158. Ubicación de los SFR en el guante



Figura 159. Ubicación de los SFR en el guante

- Ubicación de los Indicadores Led en el guante:

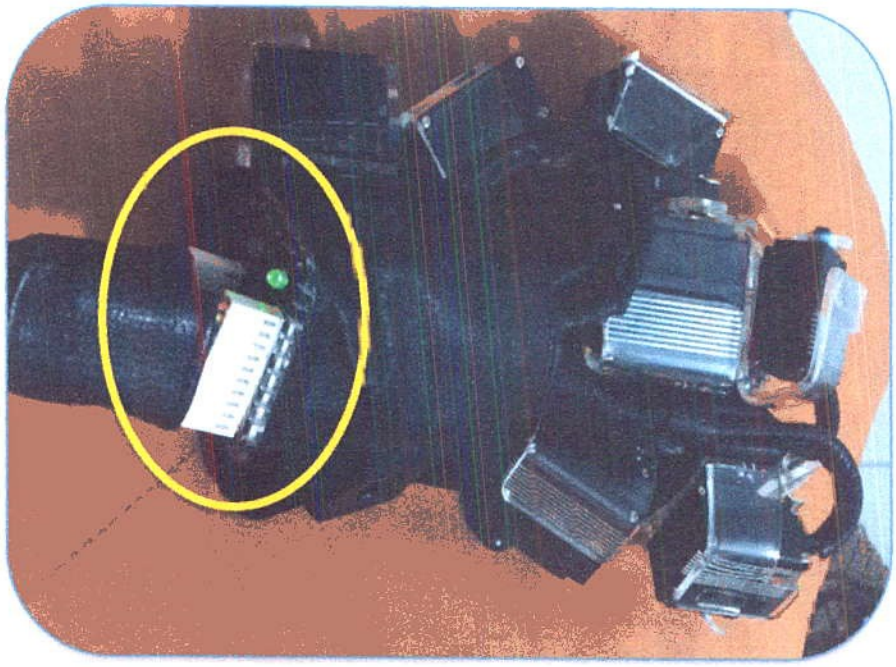


Figura 160. Ubicación de los Indicadores Led (Verde)

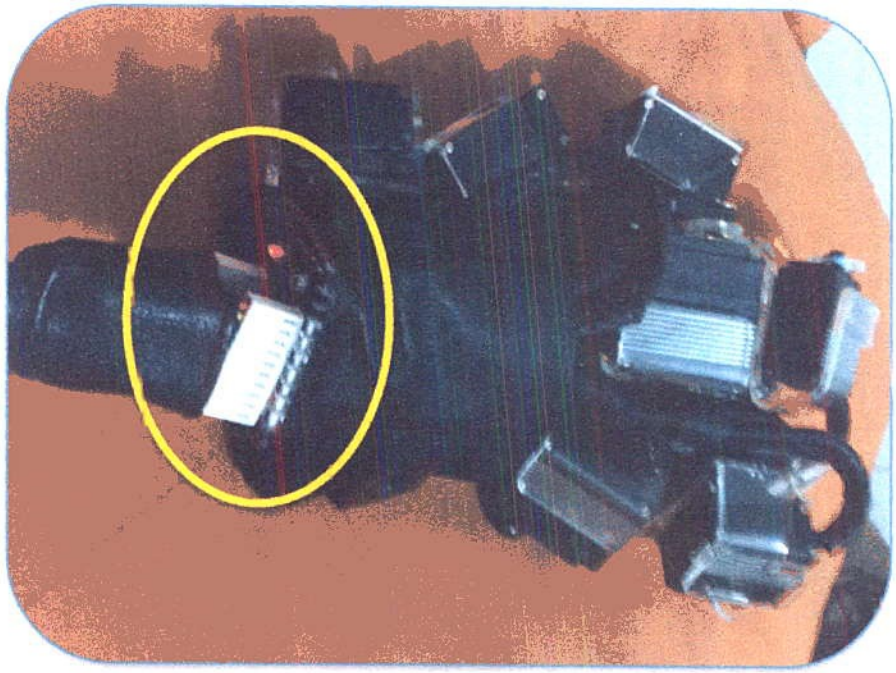


Figura 161. Ubicación de los Indicadores Led (Rojo)

- Ubicación de los sensores musculares y electrodos de electromiografía en las correas laterales que llevan los cables que interconectan los Servomotores, Led y SFR al Módulo de control del dispositivo

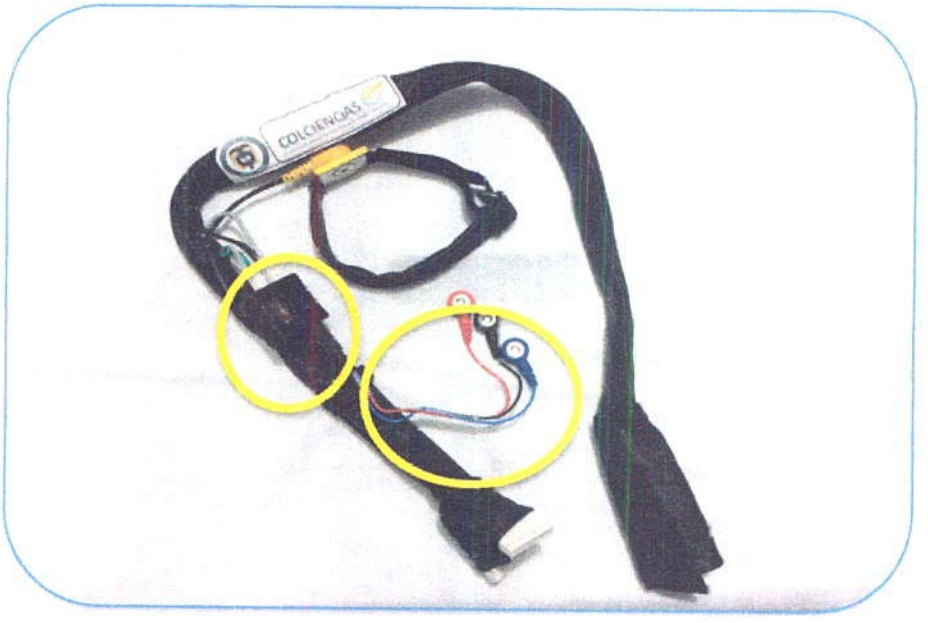


Figura 162. Ubicación Sensor Muscular en correas al módulo de control

- Ubicación de los Nodos de interconexión en las bases del exoesqueleto mecatrónico.

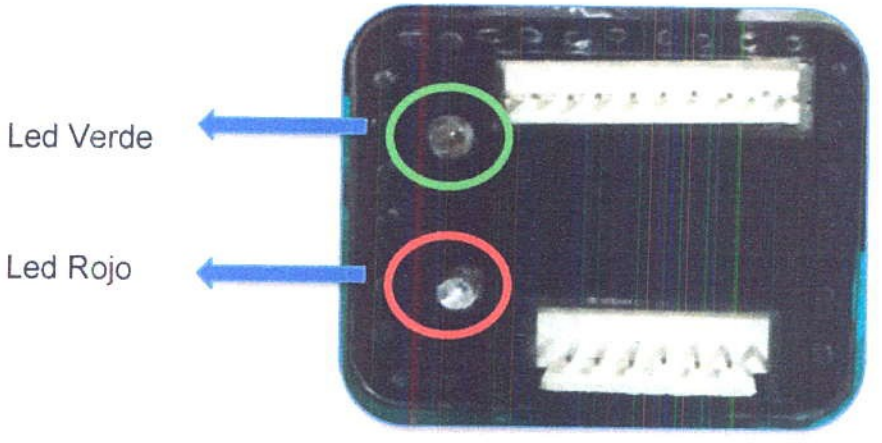


Figura 163. Nodo de interconexión cara interna

5. FUENTE DE ALIMENTACIÓN CONMUTADA

El Prototipo EMMRA-1 debe cumplir con las normas internacionales para fabricación de dispositivos biomédicos, para ello fue necesario adquirir las fuentes de alimentación conmutadas certificadas para uso médico (Switching Power Supply **SPS**), las cuales vienen fabricadas con certificación en seguridad eléctrica (IEC 60601-1) y compatibilidad electromagnética (IEC 6060-1-2).

Para alimentar los servomotores de cada mano, se implementó la SPS MPT200A en cada lado de la silla de descanso del paciente discapacitado. Esta fuente adquiere la energía eléctrica orriente Alterna de la red pública (corriente alterna) y proporciona energía al dispositivo (corriente continua). Presenta protecciones contra sobrevoltajes, cortocircuitos y cambios de tensión, entrega 4 salidas de 5 Voltios con 20 Amperios en total, una salida de 12 Voltios 7 Amperios, y una salida de -5 Voltios 1 Amperios.

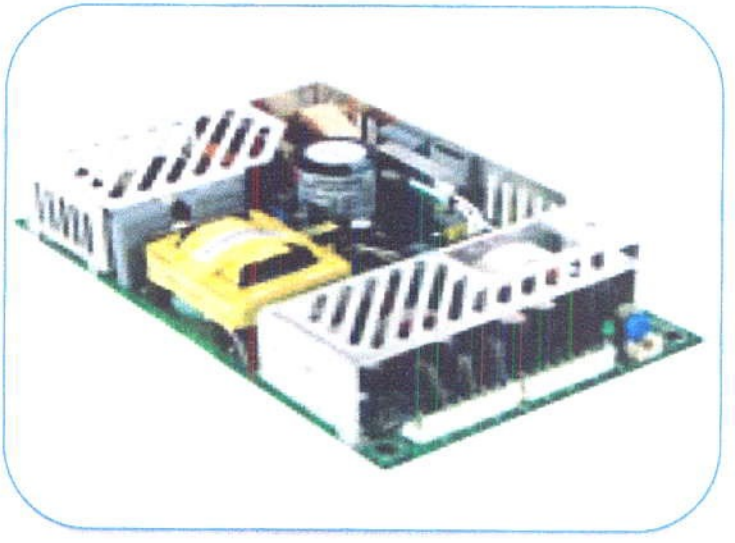


Figura 163. Fuente de Alimentación MPT200A

INFORME FINAL DEL PROYECTO
EXOESQUELETO DE MANO Y MUÑECA CON APLICACIONES DE REHABILITACION Y ASISTENCIA (PROTOTIPO EMMRA-I)

CAPITULO III
MODULO DE CONTROL DE VOZ Y ELECTROMIOGRAFIA

INVESTIGADORES:

JUAN DAVID MORENO ARANGO
JULIAN ALBERTO MORENO ARANGO

ASESORES:

VICTOR MAURICIO ACEVEDO CORREA
ALEXANDER RAMOS RODRIGUEZ

Manizales, 01 de Noviembre de 2015

INTRODUCCIÓN

Los pacientes con Lesión Medular Cervical Traumática presentan una interrupción de la información eléctrica entre la motoneurona periférica y el sistema nervioso central (SNC), resultante en limitación funcional motriz, que de acuerdo a la severidad, amplitud y altura del trauma, ocasionará mayor o menor grado de discapacidad.

El prototipo EMMRA-1 es controlado por un **Módulo de Reconocimiento de Voz (MRV)** para superar las limitaciones motrices en la discapacidad de mano y muñeca, permitiendo aplicaciones de asistencia al paciente en la manipulación de objetos. La producción del MRV requiere integrar diferentes procesos de diseño, los cuales involucran la utilización de diferentes software libre, programación de tablas electrónicas y adaptar la arquitectura del hardware al exoesqueleto prototipo.

- Hardware:
  - EasyVR Shield 2.0
  - Arduino Uno R3

- Software Libre:
  - Arduino
  - Audacity
  - Sensory QuickSynthesis 5
  - EasyVR Commander

Seleccionamos una tarjeta electrónica de reconocimiento de voz con hardware flexible (**Módulo EasyVR Shield 3.0)** por su tamaño reducido, facilidad de programación y bajo costo; características de diseño que permiten integrarlo al hardware compatible (**Arduino Mega**) para adaptarlc al exoesqueleto prototipo. En la programación del módulo se utiliza software libre (**EasyVR Commander, Arduino, Audacity, Sensory QuickSynthesis 5**), los cuales se descargan gratuitamente de las páginas web de los proveedores y fabricantes.

Nota: Parte de la información teórica de este documento, se encuentra en los manuales de usuario para el Módulo EasyVR, Arduino Mega, Software Arduino, EasyVR Commander, Audacty, y QuickSynthesis 5. Los manuales son creados por los fabricantes y están disponibles en las páginas web de los proveedores.

TABLA DE CONTENIDO

1. Modulo EasyVR..... 04
2. EasyVR Shield para Arduino..... 11
3. Arduino Mega 2560..... 15
4. Software Arduino..... 19
5. Software Audacity..... 20
6. Software Sensory QuickSynthesis 5..... 24
7. EasyVR Commander..... 27
8. Diseño del Modulo de Control Activado por voz..... 33
7.1 Implementación EasyVR Shield 3.0..... 33
7.2 Implementación Arduino Mega 2560..... 37
7.3 Implementación del Software Audacity..... 39
7.4 Implementación del Software QuickSynthesis 5..... 48
7.5 Implementación del EasyVR Commander..... 55
7.6 Subir tablas de sonido al módulo de voz..... 71
7.7 Generar código fuente para la programación del MRV..... 70
7.8 Implementación del Software Arduino..... 89
8. Diseño del Módulo de Control por Electromiografía..... 99
8.1 Sensor de Señal Muscular..... 100
8.2 Electrodos de Electromiografía..... 102
8.3 implementación del Software Arduino..... 104
9. Plataforma electrónica del módulo de control..... 105
10. Fotos del Producto terminado..... 110

1. Modulo EasyVR: (Module Easy Voice Recognition)

EasyVR 3.0 Es un módulo de reconocimiento de voz multipropósito, diseñado para el reconocimiento de voz y reproducción de sonidos de forma fácil, robusta, versátil y a bajo costo. Las voces a reconocer pueden estar predefinidas y grabadas internamente (SI: voces independientes del micrófono), también se pueden grabar y reconocer voces definidas por el propio usuario y en cualquier idioma (SD: voces dependientes del micrófono), que se supone previamente han sido enseñadas y grabadas en la memoria interna del módulo.

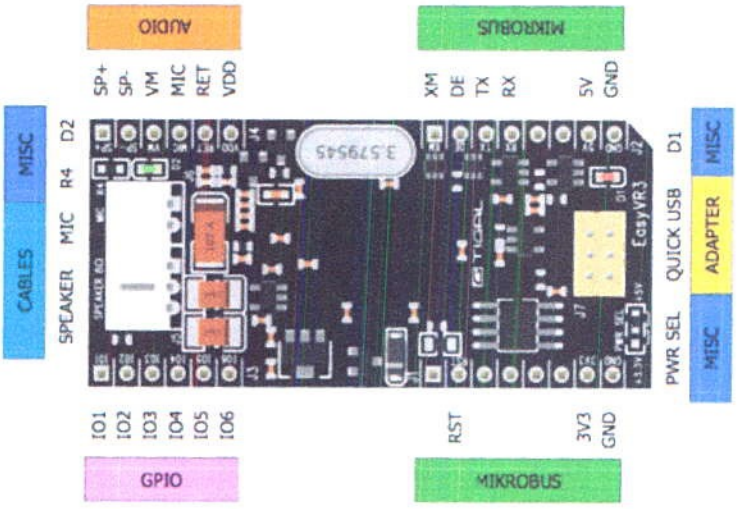


Figura 1. Modulo EasyVR. www.sparkfun.com/products

El Módulo EasyVR 3.0 puede ser usado en cualquier dispositivo con una interface UART que trabaje con voltajes entre 3.3V y 5V, tal como un PICs y Arduino Boards.

- **Aplicaciones:**

✓ Sistemas de control de propósito general gobernadas mediante la voz.
✓ Automatización de aplicaciones en el ámbito doméstico
✓ Control d acceso por voz.
✓ Sistemas rcbóticos controlados por la voz

- **Características del Módulo EasyVR 3.0**

✓ 28 comandos y voces pre-programadas (SI) en varios idiomas: inglés, italiano, japonés, alemán, francés, español.
✓ Admite hasta 32 comandos y sus correspondientes voces definidas por el usuario (SD) en cualquier idioma.
✓ Los comandos y voces SD se pueden organizar en hasta 16 grupos.
✓ Se dispone de un interface gráfico de usuario (GUI) para Windows que permite una rápida familiarización en el uso del módulo.
✓ Dispone de un potente protocolo de comandos para el desarrollo de cualquier aplicación basada en el reconocimiento de voz.
✓ Alimentación de 3.3 a 5 Voltios.
✓ 3 Líneas de Entrada y Salida E/S o GPIO (IO1, IO2, IO3) que se controlan directamente mediante comandos correspondientes.
✓ Salida PWM de audio que admite la conexión con un altavoz de 8 Ω.
✓ Capacidad de reproducción de hasta 9 minutos de sonido y/o voces

- **Especificaciones Técnicas**:

✓ Dimensiones Físicas:

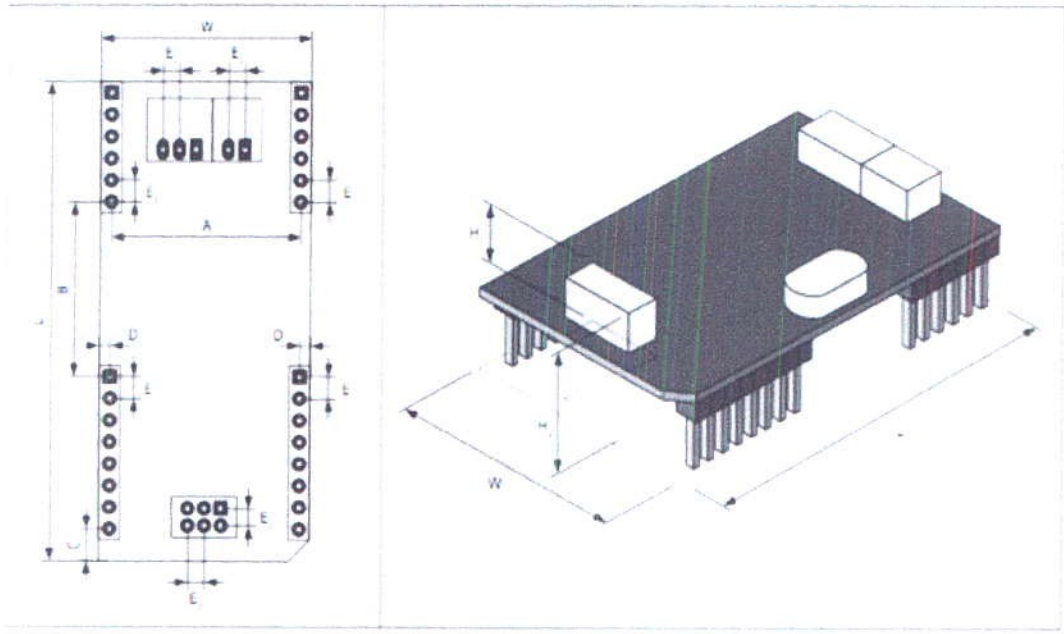


Figura 2. Modulo EasyVR. www.sparkfun.com/products

✓ Tabla 1. Asignación de Pines:

Group	Name	Number	Pin	Type	Description
MIKROBUS	J1	1	-	-	(Not connected)
		2	RST	I	Active low asynchronous reset (internal pull-up)
		3-6	-	-	(Not connected)
		7	3V3	I	3.3V DC power input
		8	GND	-	Ground
	J2	1	XM	I	Boot select (internal pull-down)
		2	DE	O	(Reserved)
		3	TX	O	Serial Data Transmit
		4	RX	I	Serial Data Receive
		5-6	-	-	(Not connected)
		7	5V	I	5.0V DC power input
		8	GND	-	Ground
GPIO	J3	1	IO1	I/O	General purpose I/O (VDD logic levels)
		2	IO2	I/O	General purpose I/O (VDD logic levels)
		3	IO3	I/O	General purpose I/O (VDD logic levels)
		4	IO4	I/O	General purpose I/O (VDD logic levels)
		5	IO5	I/O	General purpose I/O (VDD logic levels)
		6	IO6	I/O	General purpose I/O (VDD logic levels)
AUDIO	J4	1	SP+	O	Differential audio output (can directly drive 8Ω speaker)
		2	SP-	O	
		3	VM	O	Microphone power (to support custom microphones)
		4	MIC	I	Microphone audio input
		5	RET	-	Microphone return (analog ground)
		6	VDD	O	Internal logic voltage (for reference only)
CABLES	J5	1	SP-	O	Differential audio output (can directly drive 8Ω speaker)
		3	SP+	O	
		2	-	-	(Not connected)
	J6	1	MIC	I	Microphone audio input
		2	RET	-	Microphone return (analog ground)
ADAPTER	J7	1	RX_P	O	Programming cable serial data receive
		2	RTS_P	I	Programming cable request to send (reset/boot control)
		3	GND	-	Programming cable ground
		4	5V_P	I	Programming cable 5V DC power output
		5	TX_P	I	Programming cable serial data transmit
		6	CTS_P	O	Programming cable clear to send (tied to ground)

✓ Condiciones de Operación Recomendada:

Symbol	Parameter	Min	Typ	Max	Unit
5V	DC Power Input (Host) = V_{SEL}	3.15	5.0	5.5	V
3V3		3.15	3.3	5.5	V
5V_P	DC Power Input (Programming cable)	4.0	5.0	5.5	V
Ta	Ambient Operating Temperature Range	0	25	70	°C

✓ Requerimientos fuente de alimentación:

Symbol	Parameter	Min	Typ	Max	Unit
I_{SLEEP}	Sleep current (V_{SEL} = 5.0V)		6		mA
I_{OPER}	Operating current (V_{SEL} = 5.0V)		25	35	mA
I_{AUDIO}	Audio playback current (with 8Ω speaker)		175	250	$mA_{(RMS)}$
I_{TOT}	Total current consumption (excluding I/O)		25	285	$mA_{(RMS)}$
I_{PEAK}	Peak supply current (excluding I/O)		400		mA

✓ Características Eléctricas:

Symbol	Parameter	Min	Typ	Max	Unit
V_{IH}	Input High Voltage	2.1		5.5	V
V_{IL}	Input Low Voltage	0.0		0.9	V
I_{IL}	Input Leakage Current (0 < V_I < 5.5V)		-65		µA

These are applicable to pins TX, DE.

Symbol	Parameter	Min	Typ	Max	Unit
V_{OH}	Output High Voltage (I_{OH} = -0.3 mA, V_{SEL} = 3.3V)	2.6		3.3	V
	Output High Voltage (I_{OH} = -0.3 mA, V_{SEL} = 5.0V)	4.3		5.0	V
V_{OL}	Output Low Voltage (I_{OL} = 5 mA)	0.0		0.2	V

These are applicable to pin XM.

Symbol	Parameter	Min	Typ	Max	Unit
V_{IH}	Input High Voltage	1.4	(0.8)	5.5	V
V_{IL}	Input Low Voltage	0.0	(0.7)	0.5	V
I_{IN}	Input Current (0 < V_I < 3.3V)	0	0.2	0.4	mA
	Input Current (0 < V_I < 5.5V)	0	0.5	0.7	mA

These are applicable to pin RST.

Symbol	Parameter	Min	Typ	Max	Unit
V_{IH}	Input High Voltage	2.1		5.5	V
V_{IL}	Input Low Voltage	0.0		0.6	V
I_{IL}	Input Leakage Current (0 < V_I < 5.5V)		-85		µA

- **Microfono:**

Junto con el módulo EasyVR se suministra un micrófono unidireccional de tipo electret (Horn EM9745P-382):

- ✓ Sensibilidad de -38Db.
- ✓ Impedancia 2.2K
- ✓ Tensión de trabajo 3V.
- ✓ Respuesta a un amplio rango de frecuencias de 100Hz hasta 20Hz.

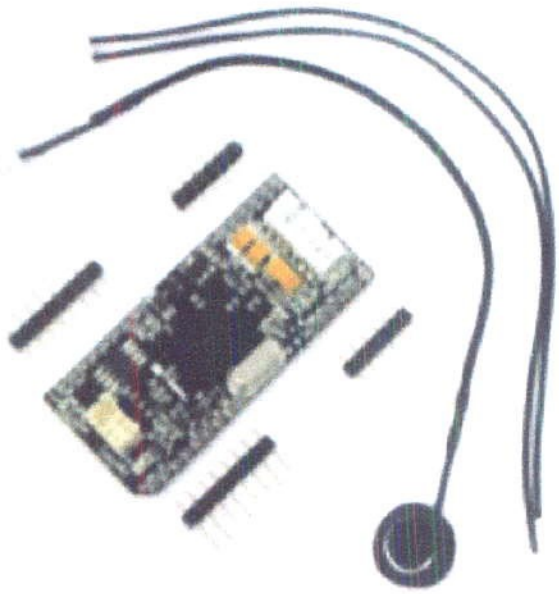


Figura 3. Micrófono del Modulo EasyVR. www.sparkfun.com/products

Una colocación acústica inadecuada del micrófono reducirá la precisión en el reconocimiento de las voces. Es posible emplear múltiples soluciones para fijar mecánicamente el micrófono en nuestra aplicación. Algunas mejorarán la precisión más que otras. En general debemos tener en cuenta las siguientes recomendaciones:

- ➢ Montaje en pared: El micrófono debe colocarse lo más cerca posible de la superficie y fijado en una carcasa plástica. No debe haber espacios vacíos entre el micrófono y la carcasa. Dichos espacios pueden modificar la resonancia acústica y reducir la precisión del reconocimiento, como se muestra en la siguiente figura.

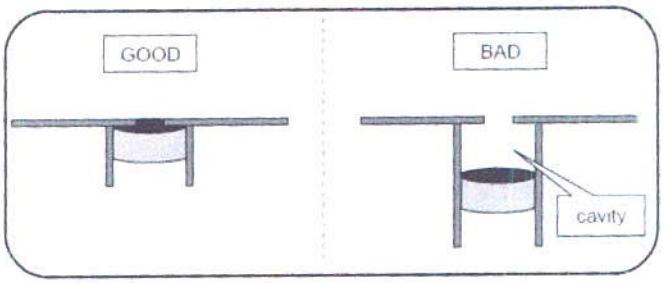


Figura 4. Micrófono del Modulo EasyVR. www.sparkfun.com/products

- No obstruir el orificio: La parte frontal del micrófono debe quedar totalmente libre de objetos que obstruyan el sonido e interfieran por tanto en el reconocimiento. El diámetro del orificio frente al micrófono debe ser de al menos 5mm. Si hubiera que proteger el orificio mediante algún tipo de plástico, éste no debería tener un grosor superior a 0,7mm. Como se muestra en la siguiente figura.

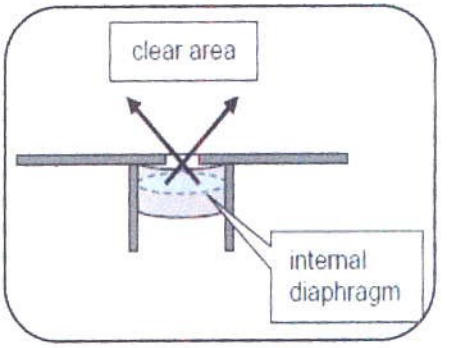


Figura 5. Micrófono del Modulo EasyVR. www.sparkfun.com/products

- Aislamiento: El micrófono debe quedar aislado respecto a la carcasa. Esto se puede conseguir envolviendo al micro con un material esponjoso para evitar ruidos producidos al manipular el módulo EasyVR o al propio micrófono, que reducirán la precisión del reconocimiento. Tal como se muestra en la siguiente figura.

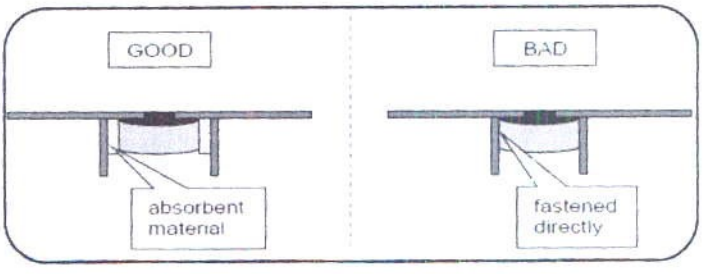


Figura 6. Micrófono del Modulo EasyVR. www.sparkfun.com/products

- Distancia: Los comandos vocales deben provenir de una distancia aproximada de ≤ 60 cm del micrófono, para distancias mayores se debe contar con amplificadores.

  Aumentar la distancia de 15 a 30 cm puede reducir el valor de la señal que proporciona el micrófono en un factor de cuatro, con lo que el reconocimiento de una voz puede ser inviable. El mismo factor y los mismos problemas se pueden dar si quien dicta la voz emplea un volumen fuerte o suave.

- Salida de Audio:

La salida de audio del módulo EasyVR puede actuar directamente sobre un altavoz de 8Ω. También es posible conectar esta salida con la entrada de un amplificador externo de audio.

- ✓ Conectar directamente unos altavoces de menor impedancia, puede dañar la salida de audio o al módulo EasyVR en su totalidad.
- ✓ Es posible conectar cargas de mayor impedancia como por ejemplo unos auriculares. En Este caso la potencia de salida habrá que reducirla conectando una resistencia en serie con ellos. El valor de esta resistencia depende de la potencia de los auriculares y del volumen de salida. Suele ser del orden de los 10 KΩ.
- ✓ La conexión directa de los auriculares con la salida de audio, los puede dañar.

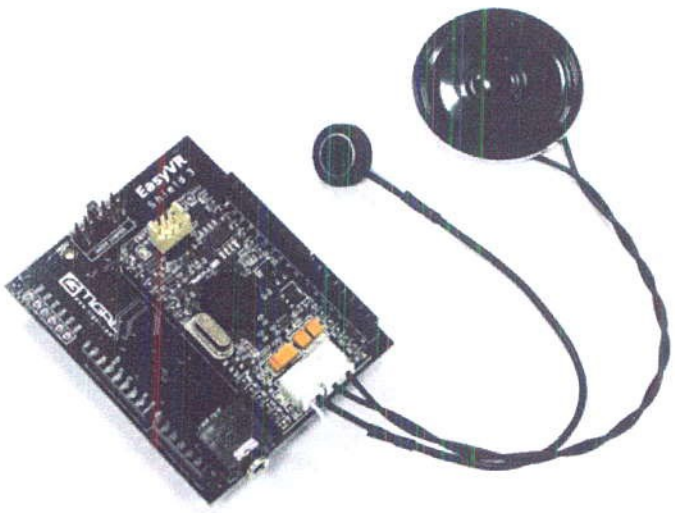


Figura 7. Micrófono del Modulo EasyVR. www.sparkfun.com/products

2. EasyVR Shield for Arduino

Es un escudo electrónico que integra el EasyVR al módulo Arduino, diseñado para simplificar la gestión en las aplicaciones Arduino.

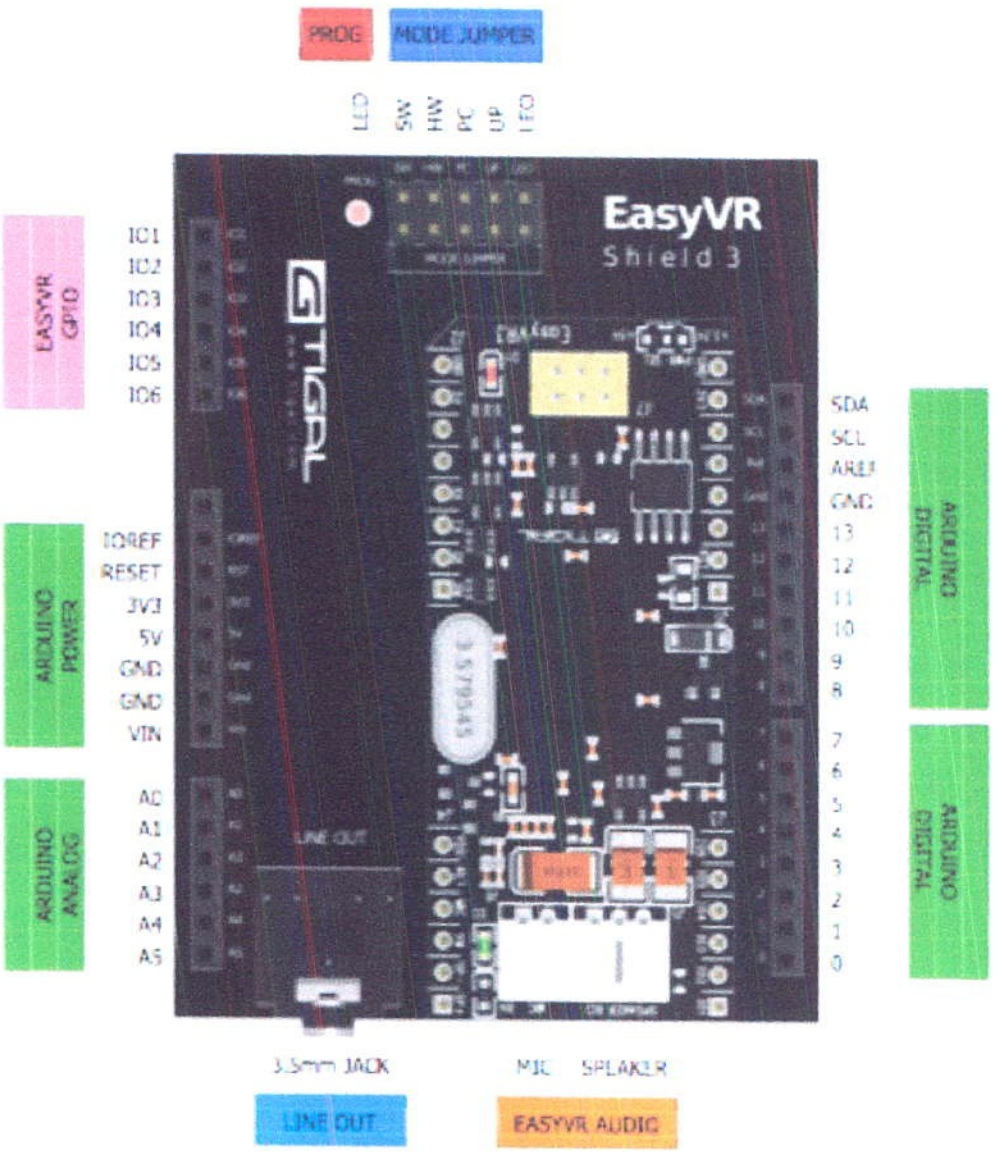


Figura 8. Escudo EasyVR. www.sparkfun.com/products

✓ Asignación de Pines:

Group	Pin	Description
ARDUINO HEADERS	-	Arduino UNO-R3 Shield interface, pass-through connectors (Pins 0-1 are in use when J12 is set to UP, PC, HW or LEO) (Pins 12-13 or 8-9 are in use when J12 is set to SW)
EASYVR AUDIO	-	Audio cables connectors of the EasyVR 3 module (microphone and speaker)
LINE OUT	-	3.5mm stereo/mono jack (16Ω - 32Ω headphones or line-level output)
MODE JUMPER	**SW**	Arduino Software Serial (connected to pins 12-13 or 8-9)
	HW	Arduino Hardware Serial (connected to pins 0-1)
	PC	PC Mode (Arduino disabled, EasyVR in command mode)
	UP	Update Mode (Arduino disabled, EasyVR in boot mode)
	LEO	Leonardo Update (Arduino enabled, EasyVR in boot mode)
PROG	-	Red light indicator for Flash programming modes (UP and LEO)
SW SERIAL PINS	**RX**	Use resistor to select Software Serial RX pin: 12 or 8
	TX	Use resistor to select Software Serial TX pin: 13 or 9
EASYVR GPIO	**IO1**	General purpose I/O as found on the embedded EasyVR 3 module (referenced at the internal VDD logic level – see note below)
	IO2	
	IO3	
	IO4	
	IO5	
	IO6	

Figura 9. Control pin /XM. www.sparkfun.com/products

- **Condiciones de Operación:**

✓ Los pines GPIO (J13.2, J13.3 y J13.4) tienen un voltaje nominal de 3.0V, No conecte estos pines directamente a 5.0V.

✓ El EasyVR shield versión 2.0 es compatilble solo con Arduino boards a niveles de voltaje de 5V I/O.

✓ Precaución: No conecte el EasyVR Shield a sistemas con un nivel de voltaje de 3.3V I/O.

- **Ajuste de los Jumper**

  J12 – Modo de Operación:

  - **UP – Flash update mode**: Se usa para actualizaciones de firmware o descargar datos de la tabla de sonido en la memoria flash a bordo del **Comandante EasyVR**. En este modo, el controlador Arduino se mantiene en reset y sólo se usa el adaptador USB/Serial incrustado. El módulo EasyVR se encuentra en modo de arranque.

  - **PC – PC conection mode:** Se usa para la conexión directa con el **Comandante EasyVR**. En este modo, el controlador Arduino se mantiene en reset y sólo se usa el adaptador USB/Serial incrustado.

  - **HW – Hardware serial mode:** Se usa para controlar el módulo EasyVR de la tabla Arduino a través del puerto serie del hardware, (usando los pines 0-1).

  - **SW – Software serial mode:** Se usa para controlar por software el módulo EasyVR de la tabla Arduino a través de un puerto serie (usando los pines 12-13). También puede conectar el Comandante EasyVR en este modo, siempre que se encuentre en bridge mode (ver bibliotecas).

- **LEDs:**

  Un LED verde (D6) está conectado al pin IO1 y puede ser controlado por el programa del usuario para mostrar comentarios durante las tareas de reconocimiento. Este LED está activado por defecto después de un reinicio o encendido.

  El LED rojo (D5) se enciende cuando se inicia el escudo, parpadea en modo de actualización (ver configuración de los jumpers).

- **Como usar el escudo (EasyVR Shield)**

Siga estos pasos para empezar a practicar con su EasyVR Shield y Arduino Board:

- ✓ Inserte el Escudo EasyVR en la parte superior de la placa Arduino.
- ✓ Si desea utilizar la salida de audio, conecte los cables de un altavoz 8Ω en los terminales de tornillo (J10) o conecte auriculares al jack de salida de 3,5 mm (J9).
- ✓ Conecte el micrófono suministrado a la (J11) conector MIC IN.
- ✓ Instale las bibliotecas EasyVR Arduino en el CP4 (detalles en la página de Arduino (http://arduino.cc/en/Guide/Libraries).
- ✓ Conecte la placa Arduino a su PC a través de USB.

- **EasyVR con Arduino Uno**

Para comprobar que todo funciona bien:

- ✓ Asegúrese de que el puente (J12) está en la posición PC.
- ✓ Abra el Comandante EasyVR y conéctese al puerto serie de Arduino.

Para descargar un nuevo sonido de la tabla:

- ✓ Asegúrese de que el puente (J12) está en la posición UP.
- ✓ Abra el Comandante EasyVR y seleccione el puerto serie de Arduino.
- ✓ Mientras desconectado elegir "Actualizar datos personalizados " en el menú "Archivo" (consulte Uso de datos personalizados).

Para probar el escudo con su IDE de programación Arduino:

- ✓ Coloque el puente (J12) en la posición de SW.
- ✓ Abra el ejemplo boceto TestEasyVR de su menú IDE "Archivo">"Ejemplos">"EasyVR".
- ✓ Sube el boceto y abra la ventana "Serial Monitor".
- ✓ Véanse los comentarios sobre la parte superior del boceto de detalles de uso.

Nota: Para mayor información se debe consultar directamente en el Manual de Usuario EasyVR versión 3.6.6.

3. Arduino Mega 2560

La siguiente información es un resumen de las guías y manuales del fabricante del producto Arduino en la página de internet http://www.arduino.cc

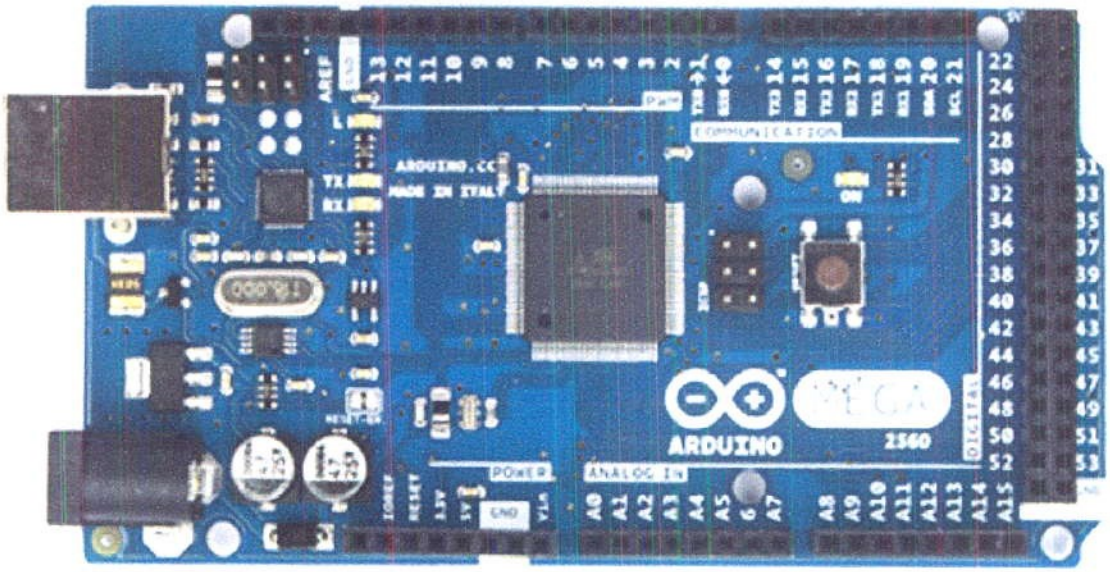


Figura 10. Arduino Mega 2560. www.arduino.cc.

El Arduino Mega 2560 es una placa electrónica basada en el Atmega 2560. Cuenta con 54 pines digitales de entrada / salida (de los cuales 15 pueden utilizarse para salidas PWM), 16 entradas analógicas, 4 UARTs (hardware puertos serie), un oscilador de crustak de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP, y un botón de reinicio.

Contiene todo lo necesario para apoyar el microcontrolador; basta con conectarlo a un ordenador con un cable USB o el poder con un adaptador de CA o la batería a CC para empezar.

- **Características del Arduino Mega**

✓ **Pinout**: SDA añadido y pines SCL que están cerca al pin AREF y otros dos nuevos pasadores colocados cerca del pin de RESET, la instrucción IOREF que permiten a los escudos para adaptarse a la tensión proporcionada.
✓ Circuito de RESET fuerte.
✓ ATmega 16U2 que sustituye el 8U2.

- **Esquema, Diseño y Mapa de Referencia de Pines:**

Microcontroladores	Atmega2560
Tensión De Funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límites)	6-20V
Digital pines I / O	54 (de las cuales 15 proporcionan salida PWM)
Botones de entrada analógica	16
Corriente DC por E / S Pin	40 mA
Corriente DC de 3.3V Pin	50 mA
Memoria Flash	256 KB de los cuales 8 KB utilizado por el gestor de arranque
SRAM	8 KB
EEPROM	4 KB
Velocidad De Reloj	16 MHz

- **Poder**

El Mega Arduino puede ser alimentado a través de la conexión USB o con una fuente de alimentación externa. La fuente de alimentación se selecciona automáticamente. El tablero puede funcionar con un suministro externo de 6 a 20 voltios. Si se suministra con menos de 7V, sin embargo, el pin de 5V puede suministrar menos de cinco voltios y la junta puede ser inestable. Si se utiliza más de 12V, el regulador de voltaje se puede sobrecalentar y dañar la placa. El rango recomendado es de 7 a 12 voltios.

- **Memoria**

El Atmega2560 tiene 256 KB de memoria flash para almacenar código (de los cuales 8 KB se utiliza para el cargador de arranque), 8 KB de SRAM y 4 KB de EEPROM (que puede ser leído y escrito con la biblioteca EEPROM).

- **Entradas y Salidas**

Cada uno de los 54 pines digitales en el Mega se puede utilizar como una entrada o salida, utilizando pinMode () ,digitalWrite (), y digitalRead () funciones. Operan en 5 voltios. Cada pin puede proporcionar o recibir un máximo de 40 mA y tiene una resistencia de pull-up (desconectado por defecto) de 20 a 50 KΩ. Además, algunos pines tienen funciones especializadas:

- ✓ Serial: 0 (RX) y 1 (TX); Serial 1: 19 (RX) y 18 (TX); Serial 2: 17 (RX) y 16 (TX); Serial 3: 15 (RX) y 14 (TX). Se utiliza para recibir (RX) y transmitir datos en serie (TX) TTL.Pernos 0 y 1 también están conectados a los pines correspondientes del ATmega16U2 USB a TTL chip de serie.

- ✓ Interrupciones externas:. 2 (interrumpir 0), 3 (alarma 1), 18 (interrumpe 5), 19 (interrupción 4), 20 (interrumpir 3), y 21 (2) de interrupción Estos pines pueden configurarse para activar una interrupción en un valor bajo, un flanco ascendente o descendente, o un cambio en el valor. Ver el attachInterrupt () función para más detalles.

- ✓ PWM: 2 a 13 y 44 a 46. proporcionar una salida PWM de 8 bits con el analogWrite () función.

- ✓ SPI:. 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS)Estos pines admite la comunicación SPI utilizando labiblioteca de SPI.Los pines SPI también se desglosan en la cabecera ICSP, que es físicamente compatible con el Uno, Duemilanove y Diecimila.

- ✓ LED: 13. Hay un LED incorporado conectado al pin digital 13. Cuando el pasador es de alto valor, el LED está encendido, cuando el pasador es bajo, es apagado.

- ✓ TWI: 20 (SDA) y 21 (SCL). Apoyar la comunicación TWI utilizando la biblioteca de alambre. Tenga en cuenta que estos pines no están en la misma ubicación que los pines TWI en el Duemilanove o Diecimila.

El Mega2560 tiene 16 entradas analógicas, cada una de las cuales proporcionan 10 bits de resolución (es decir, 1.024 valores diferentes). Por defecto se miden desde el suelo a 5 voltios, aunque es posible cambiar el extremo superior de su rango usando el pin AREF y función analogReference ().

Hay un par de patas de la placa:

- AREF. Voltaje de referencia para las entradas analógicas. Se utiliza con analogReference ().

- Restablecer. Traiga esta línea LOW para reiniciar el microcontrolador. Normalmente se utiliza para añadir un botón de reinicio para escudos que bloquean el unc en el tablero.

- **Programación:**

El Arduino Mega se puede programar con el software de Arduino. Los Atmega2560 en la Mega Arduino viene precargado con un gestor de arranque que le permite cargar nuevo código a él sin el uso de un programador de hardware externo. Se comunica usando el original STK500 protocolo (referencia, archivos de cabecera C).

También puede pasar por alto el gestor de arranque y programar el microcontrolador a través del ICSP (In-Circuit Serial Programming) cabecea usando Arduino ISP o similar; consulte estas instrucciones para más detalles.

El ATmega16U2 (o 8U2 en el rev1 y tableros Rev2) código fuente del firmware está disponible en el repositorio de Arduino. El ATmega16U2 / 8U2 se carga con un cargador de arranque DFU, que puede ser activado por:

El Mega2560 contiene un rastro que se puede cortar para deshabilitar el reinicio automático. Las almohadillas a ambos lados de la traza se pueden soldar juntos para volver a habilitarlo. Ha marcado "RESET-ES". También puede ser capaz de desactivar el reinicio automático mediante la conexión de una resistencia de 110 ohmios de 5V a la línea de reposición.

4. Software de Arduino

El código abierto Arduino Software (IDE) hace que sea muy fácil escribir código de programación y subirlo a la placa de Arduino Uno R3. Se ejecuta en Windows, Mac OS X y Linux. El entorno está escrito en Java y está basado en el procesamiento y software de código abierto. Este software se puede utilizar con cualquier placa Arduino. Se descarga gratuitamente de la página del fabricante: http://www.arduino.cc

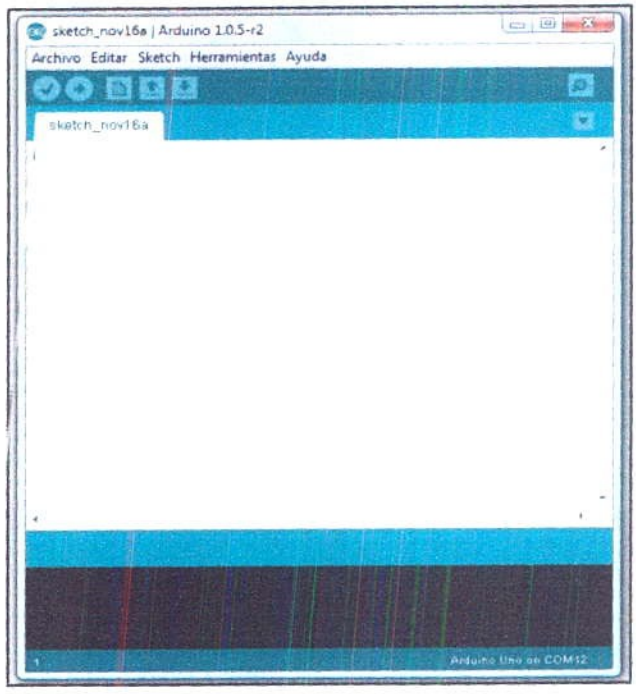


Figura 11. Sketch para programación en Arduino

Hay muchos otros microcontroladores y plataformas microcontroladas disponibles para computación física. Parallax Basic Stamp, Netmedia´s BX-24, Phidgets, MIT´s Handyboard, entre otras. Todas estas herramientas toman los desordenados detalles de la programación de microcontrolador y la encierran en un paquete fácil de usar. Sin embargo, Arduino ofrece algunas ventajas para profesores, estudiantes y makers.

- Económico:
- Multiplataforma: Se ejecuta en sistemas operativos Windows, Mac, Linux.
- Entorno de programación simple y claro: Fácil y flexible.
- Código Abierto y Software Extensible: El Arduino está basado en microcontroladores ATMEGA de Atmel, Los planos para los módulos están publicados bajo licencia Creative Commons, por lo que diseñadores experimentados pueden realizar su propia versión del módulo y mejorarlo.

5. Software Audacity

Audacity es un programa de grabación y edición de sonidos fácil de usar, multiplataforma, de libre uso, y de código abierto. Puede grabar sonidos, reproducir sonidos, importar y exportar archivos WAV, AIFF, y MP3, y más. De hecho, es uno de los programas libres de edición de sonido más fiable y avanzado que existe actualmente. No necesita siquiera instalación. Se puede bajar directamente de la página: http://audacity.sourceforge.net

El programa ofrece las posibilidades básicas de cortar, copiar y pegar, pero además se puede deshacer la última acción de forma ilimitada, trabajar con varias pistas a la vez, mezclarlas o aplicar diversos efectos a los sonidos.

Características:

- Grabar:
- ✓ Grabar sonidos en directo usando un micrófono o un mezclador.
- ✓ Digitalizar grabaciones ya realizadas.

- Importar y Exportar
- ✓ Importa y exporta archivos WAV, AAIFF, AU, Ogg Vorbis.
- ✓ No trabaja con formatos WMA, AAC, o formatos restringdos por sus propietarios.

- Edición:
- ✓ Edición Sencilla mediante cortar, copiar, pegar y borrar.
- ✓ Rápida edición de archivs grandes.
- ✓ Edita y mezcla un número ilimitado de pistas
- ✓ Utiliza herramienta de dibujo para alterar muestras individuales
- ✓ Disuelve el sonido suavemente con la herramienta "envolvente"

- Efectos:
- ✓ Cambia el tono sin alterar el tiempo y visceversa
- ✓ Elimina ruidos estáticos, silbidos, etc.
- ✓ Altera las frecuencias con la ecualización, filtros FFT y amplicación de los bajos.
- ✓ Ajusta los volúmenes con el compresor, amplificar y normalizar los efectos
- ✓ Otros efectos incluidos: Eco, Fade, Wahwah, Inversión.

- Complementos:
- ✓ Añade nuevos efectos con los complementos LADSPA
- ✓ Carga complementos VST para Windows y Mac, con el opcional Activador VST.
- ✓ Escribe nuevos efectos en lenguaje de programación Nyquist incluido.

- Análisis:
- ✓ Modo espectógrafo para visualizar frecuencias.
- ✓ Comando "Dibujar espectro" para obtener un análisis detallado de frecuencias.

- Libre y Multi-plataforma
- ✓ Con una Licencia Pública General de GNU (GNU GPL).
- ✓ Corre bajo Mac OS X, Windows y GNU/Linux

➢ **Como grabar un sonido (voces)**

1. Es necesario tener instalado un micrófono y tenerlo listo para grabar.

2. Abrimos el programa Audacity, vemos una serie de íconos en la parte superior, similares a los de cualquier reproductor o grabadora.

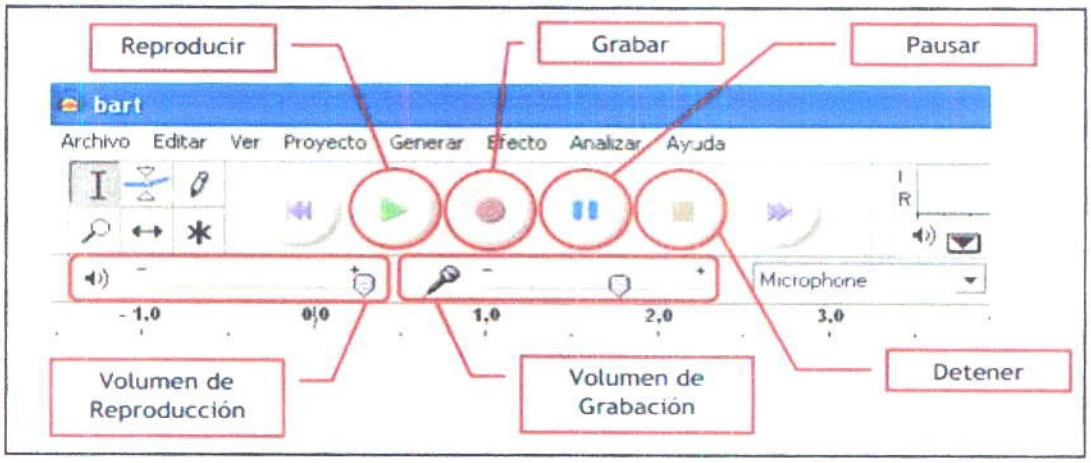


Figura 12. Controladores Software Audacity. http//audacity.sourceforge.net

3. Antes de grabar, es necesario que determinemos la calidad del sonido que deseamos obtener (si queremos que se escuche bien, lo que generará un archivo más grande, o si podemos perder algo de calidad obteniendo un archivo más pequeño). Con ello determinamos la frecuencia de muestreo predeterminada:

- ✓ 11025Hz: Sonido de calidad inferior y archivos de poco tamaño.
- ✓ 16000Hz: Sonido de calidad media y archivos de poco tamaño.
- ✓ 22050Hz:Sonido de calidad normal y archivos de tamaño normal.
- ✓ 44100Hz: Sonido de alta calidad y archivos de gran tamaño

4. Para grabar la voz, pulsamos sobre el ícono de grabar y comenzamos a hablar, Cuando terminemos de hablar pulsamos sobre el ícono parar.

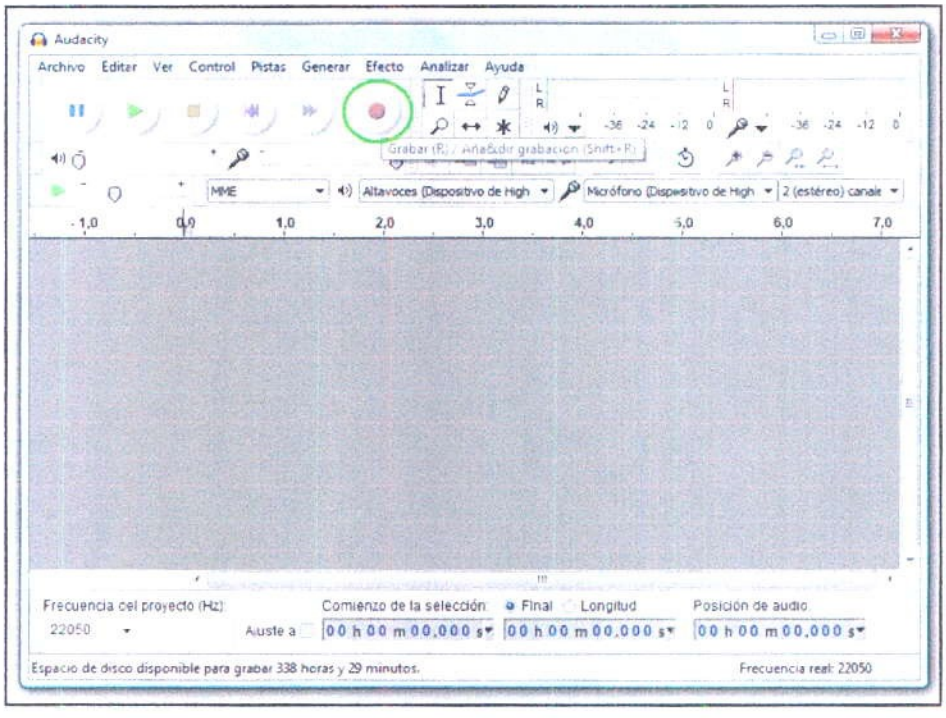


Figura 13. Icono Grabación en Software Audacity. http//audacity.sourceforge.net

5. Finalmente damos guardar el archivo grabado y posteriormente editarlo para mejorar algunos aspectos de sonido. Es necesario exportar como wav y asignar un nombre al archivo para guardarlo.

➢ **Como editar un sonido:**

1. Comprobamos que el micrófono está encendido y que funcionan los altavoces o auriculares de nuestro ordenador.
2. Grabamos un archivo en Audacity con el nombre PRUEBA.WAV.
3. Abrimos el archivo PRUEBA.WAV

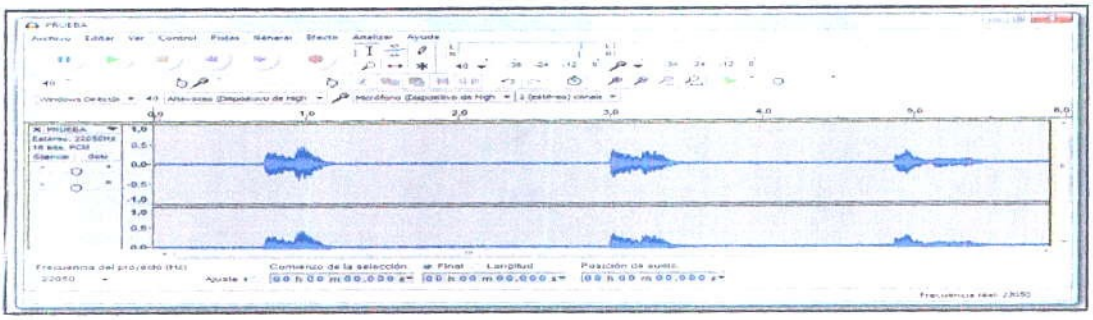


Figura 14. Grabación de voz. http//audacity.sourceforge.net

El dibujo generado muestra el patrón de sonido de nuestra grabación, Este patrón es como un gráfico espacio-temporal de nuestras palabras. Está compuesto de 'bolas" o "triángulos" con espacios o líneas en medio. Cada bola o triángulo es una palabra y cada espacio una pausa de nuestra locución.

4. Seleccionamos un fragmento, utilizando la herramienta selección. Hacemos clic en un punto cualquiera del patrón de sonido, arrastramos el puntero hasta el sitio que deseamos, así se mostrará un espacio sombreado del fragmento que hemos seleccionado.

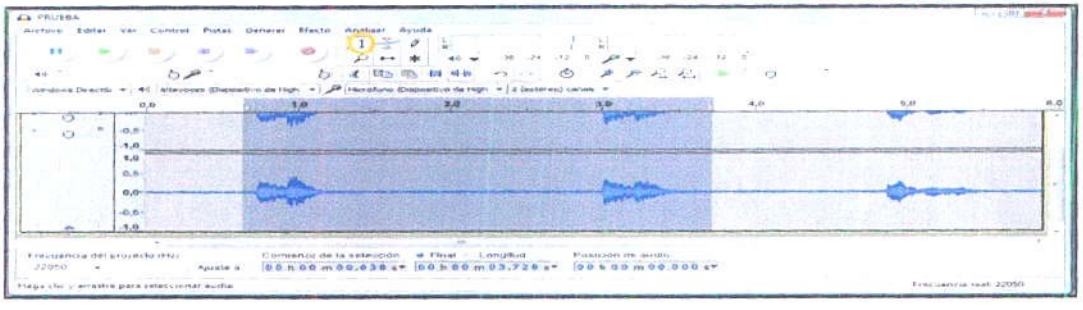


Figura 15. Edición de Sonido. http//audacity.sourceforge.net

5. Sobre el fragmento de sonido seleccionado podemos realizar diferentes operaciones de edición: Cortar el fragmento del sonido seleccionado. Copiar y pegar este fragmento en otro sonido.

6. Software Sensory QuickSynthesis 5



El módulo EasyVR puede también reproducir sonidos de cualquier tipo (música, voces, señales acústicas, etc.) previamente almacenados en su memoria flash interna. El módulo incorpora el sonido "beep".

Los sonidos se organizan en una "Tabla de sonidos" que el usuario debe crear mediante la aplicación QuickSyntesis 5 (QS5) de la firma Sensory. Esta aplicación se instala cuando instalamos el programa EasyVR Commander.

Preparamos los archivos de audio que queremos reproducir mediante el módulo EasyVR. Estos archivos deben tener formato WAV, con 16 bits sin compresión y mono de 22050Hz. Para crear este tipo de archivos existen gran variedad de programas y aplicaciones.

- ✓ Es muy importante abrir el software con click izquierdo en el mouse, eligiendo la opción **ejecutar como administrador**, para que el software funcione correctamente.

- ✓ Abrimos la aplicación QuickSyntesys 5 y creamos un nuevo proyecto especificando la familia RSC4. A continuación asignamos un nombre al proyecto (ejemplo "Sonidos MSE").

- ✓ Adicionamos cada uno de los archivos WAV con los que queremos construir la "tabla de sonidos" para su posterior grabación y reproducción en el módulo EasyVR. Con cada archivo añadido se debe indicar el tipo de compresión que se le va a aplicar.

Los archivos WAV con los que vamos a construir la "Tabla de sonidos", y la compresión asignada a cada uno, se pueden identificar perfectamente en la lista de archivos que hay en nuestra carpeta previamente elaborada.

La compresión para cada archivo tiene que ver con la calidad de reproducción y el tiempo total de reproducción disponible en el módulo EasyVR. En general, a menor compresión más calidad, aunque también ocupa mayor espacio en la memoria flash del módulo, por lo que el tiempo total de reproducción será menor.

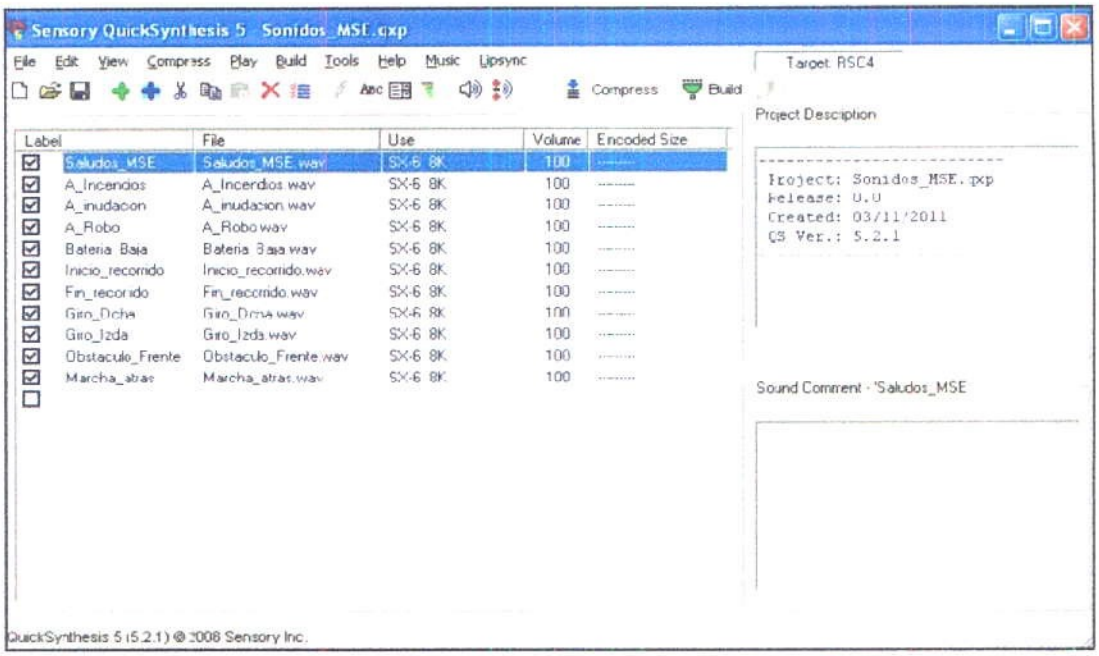


Figura 16. Creación de tablas de sonido. Manual EasyVR Commander.

La siguiente tabla nos puede dar una idea aproximada de la relación entre calidad y tiempo disponible.

Compression Scheme	Available Time (8kHz 15% silence)	Available Time (9.3kHz 15% silence)
SX-2	8.7 minutes	7.5 minutes
SX-3	7.6 minutes	6.6 minutes
SX-4	6.8 minutes	5.9 minutes
SX-5	6.1 minutes	5.2 minutes
SX-6	5.6 minutes	4.8 minutes
4-bit ADPCM	87 seconds	N/A
8-bit PCM	45 seconds	38 seconds

Figura 17. Calidad vs Tiempo. Manual EasyVR Commander

- ✓ Seleccionamos todos los archivos de la lista y procedemos a comprimirlos mediante el ícono "Compress".

- ✓ Mediante el ícono "Build" o Ctrl+B construimos el proyecto par obtenera la "Tabla de sonidos". Debemos usar los ajustes por defecto: "Build linkable module", "Load in CONST space" y "Load abobe or at:0".

- ✓ Debemos guardar el proyecto y, si procede, volverlo a construir para tener la versión más actualizada posible. En nuestro ejemplo el proyecto se llama Sonidos_MSE.qxp".

Una vez creada la tabla de sonidos, debemos ahora descargarla sobre el módulo EasyVR tal y como se comentó previamente en la actualización de la memoria flash interna.

- ✓ Desconectar la comunicación entre la aplicación EasyVR Commander y el módulo ("Disconnect").
- ✓ Desconectar la alimentación del móculo EasyVR y cerrar el jumper. Con esto conectamos la patilla / XM a Vcc a través de la resistenca. Volvemos a conectar la alimentación con lo que el módulo está en el modo de actualización de firmware o de descarga de la "Tabla de sonidos".
- ✓ Reestablecemos la comunicación entre el programa EasyVR Commander y el módulo ("Connect"). Observamos que el ícono "Update Sound Table" queda habilitado.
- ✓ Al dar clíck en el ícono "Update Sound Table" se abre una ventana de dialogo cue nos permite vuscar el archivo con la "Tabla de sonidos" que hemos construido en la aplicación Quick Sinthesis 5.
- ✓ Aparece una ventana "Build Sound Table", en ella se aprecia la lista de los sonidos que compilamos en el proyecto anterior. Activamos la casilla "Slow transfer" y pulsamos "Download". Debemos esperar un cierto tiempo para que dicha tabla de sonidos se grabe sobre la memoria flash del módulo EasyVR.

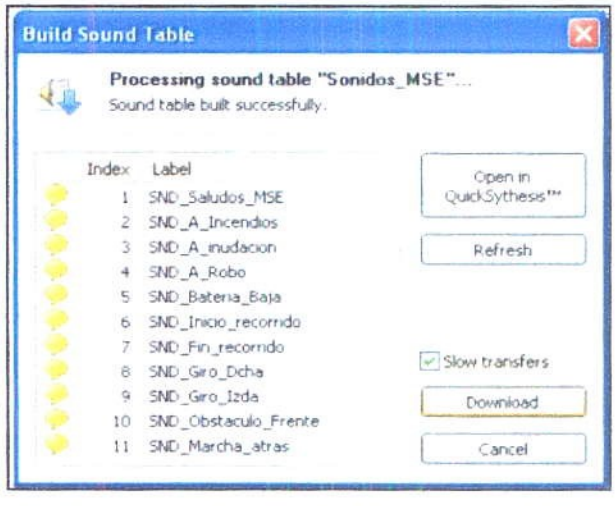


Figura 18. Cargar tabla de sonidos. Manual EasyVR Commander

- ✓ Desconectamos la alimentación del módulo y retiramos el jumper. La patilla /XM queda ahora a nivel "0" gracias a la resistencia pull-down interna. Volvemos a conectar la alimentación y reestablecemos la comunicación entre EasyVR Commander y el módulo.

7. EasyVR Commander (Software Interface Gráfico de Usuario)

Se trata de una aplicación para Windows que permite usar y configurar el módulo EasyVR de una forma sencilla e intuitiva.

- ✓ Instalamos el programa EasyVR Commander, la aplicación es gratuita y aparece en las páginas del proveedor del módulo EasyVR.
- ✓ Conectamos el módulo EasyVR al PC a través del puerto USB.
- ✓ **Ejecutamos la aplicación como administrador.**
- ✓ Aparecerá la ventana de trabajo para programar el módulo EasyVR

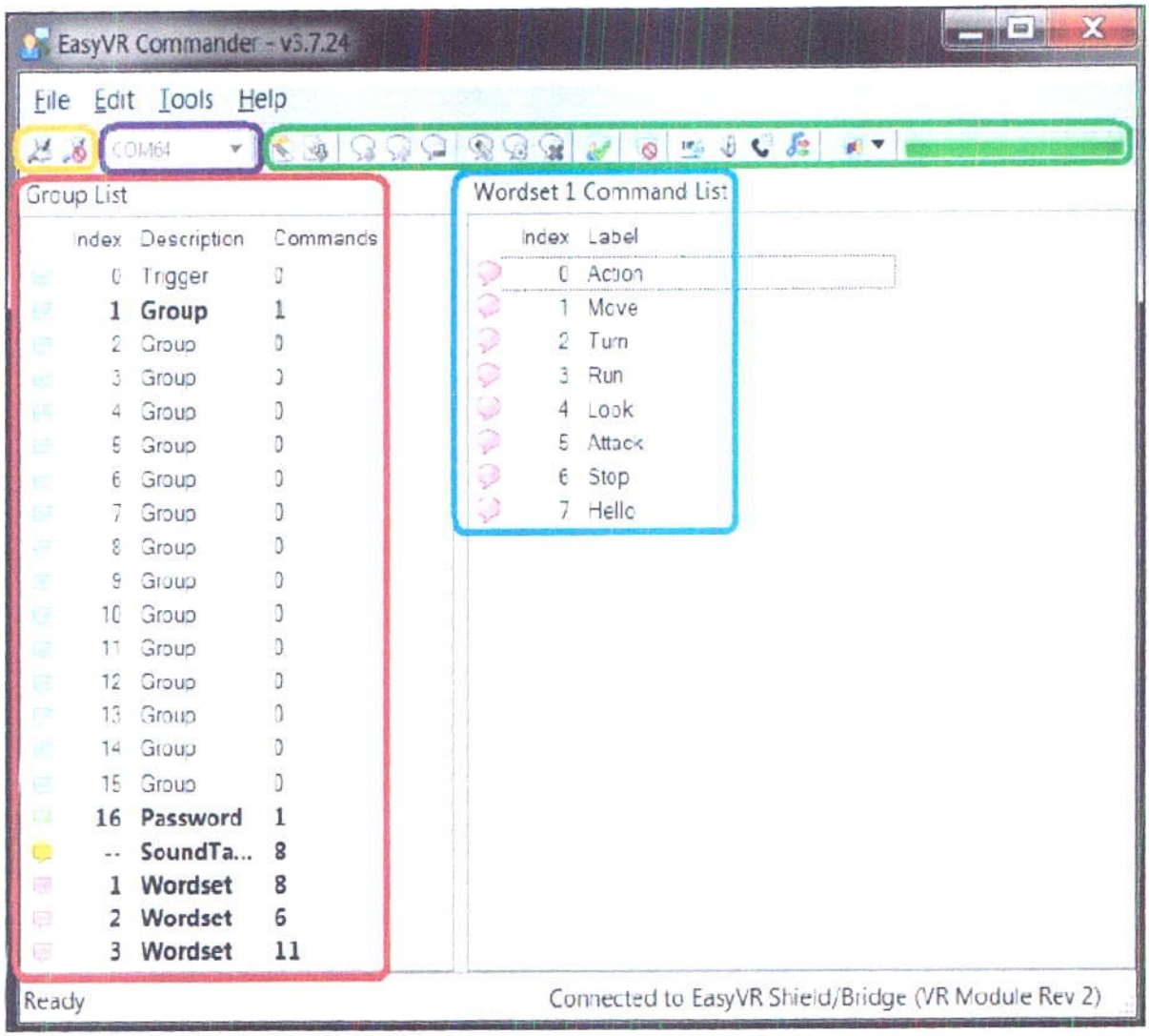


Figura 19. Software EasyVR Commander. Manual EasyVR Commander

- En el **recuadro naranja** tenemos dos iconos para la conexión y desconexión del módulo EasyVR al EasyVR Commander.

- En el **recuadro morado** tenemos un desplegable para seleccionar el puerto COM al que se conectó el módulo al PC.

- En el **recuadro verde** tenemos iconos para las diferentes tareas que podemos realizar con el módulo. añadir/editar/borrar comandos, enseñar voces asociadas a comandos, reconocimiento de voces, reproducción de sonidos, etc.
- En el **recuadro rojo** tenemos la Lista de Grupos, aparecen los grupos 0 - 16 disponible para almacenar las voces definidas por el usuario (SD) y otros tres grupos llamados "Wordset 1-3" donde están almacenadas las voces pregrabadas (SI).

  Al lado de cada grupo aparece también el número de voces que continene el mismo. Recuerde que solo pueden grabarse 32 voces definidas por el usuario (SD) en el módulo EasyVR.

  Al finalizar los grupos existe el SoundT, donde se puede almacenar diferentes sonidos digitaliados para su posterior reproducción.

- En el **recuadro verde** tenemos el área de comandos, donde se editan y visualizan las etiquetas o comandos asociados a las voces y qué posición ocupan dentro de un determinado grupo. Un comando o etiqueta es una cadena ASCII de hasta 32 caracteres. Toda etiqueta o comando debe estar asociada a una voz durante la fase de enseñanza. De esta forma cuando el módulo reconozca una voz, devuelve el comando o etiqueta asociada a la misma. Solo se admiten etiquetas formadas por caracteres ASCII en letras mayúsculas desde la A hasta la Z.

Las Voces SI predefinidas vienen grabadas de fábrica y no pueden ser modificadas. Pueden ser reconocidas en 6 idiomas diferentes: inglés, italiano, japonés, alemán, español y francés. Están almacenadas en los grupos llamados Trigger, Wrdset 1, Wordset 2, Wordset 3.

La siguiente tabla muestra según el idioma elegido, los grupos con las etiquetas y voces que contienen.

		Language						
		0	1	2		3	4	5
Trigger Word set	Command Index	English (US)	Italian	Japanese	(Rōmaji)	German	Spanish	French
0	0	robot	robot	ロボット	*robotto*	roboter	robot	robot
1	0	action	azione	アクション	*acution*	aktion	acción	action
	1	move	vai	進め	*susu-me*	gehe	muévete	bouge
	2	turn	gira	曲がれ	*magare*	wende	gira	tourne
	3	run	corri	走れ	*hashire*	lauf	corre	cours
	4	look	guarda	見ろ	*miro*	schau	mira	regarde
	5	attack	attacca	攻撃	*kougeki*	attacke	ataca	attaque
	6	stop	fermo	止まれ	*tomare*	halt	para	arrête
	7	hello	ciao	こんにちは	*konnichiwa*	hallo	hola	salut
2	0	left	a sinistra	左	*hidari*	nach links	a la izquierda	à gauche
	1	right	a destra	右	*migi*	nach rechts	a la derecha	à droite
	2	up	in alto	上	*ue*	hinauf	arriba	vers le haut
	3	down	in basso	下	*chita*	hinunter	abajo	vers le bas
	4	forward	avanti	前	*mae*	vorwärts	adelante	en avant
	5	backward	indietro	後ろ	*ushiro*	rückwärts	atrás	en arrière
3	0	zero	zero	ゼロ	*zero*	null	cero	zéro
	1	one	uno	一	*ichi*	eins	uno	un
	2	two	due	二	*ni*	zwei	dos	deux
	3	three	tre	三	*san*	drei	tres	trois
	4	four	quattro	四	*yon*	vier	cuatro	quatre
	5	five	cinque	五	*go*	fünf	cinco	cinq
	6	six	sei	六	*roku*	sechs	seis	six
	7	seven	sette	七	*nana*	sieben	siete	sept
	8	eight	otto	八	*hachi*	acht	ocho	huit
	9	nine	nove	九	*kyu*	neun	nueve	neuf
	10	ten	dieci	十	*jyuu*	zehn	diez	dix

Figura 20. Tabla de etiquetas y lenguaje del EasyVR Commander. Manual EasyVR Commander

- **Reconocimiento de voces predefinidas (SI)**

✓ Selección del idioma español con el icono "Set Language".
✓ Selección del grupo en que se encuentran las voces SI.
✓ Selección del icono "Test Group" para iniciar el reconocimiento.
✓ Pronunciamos cualquiera de las voces disponibles en el grupo seleccionado.
✓ Si la voz es reconocida, la etiqueta o comando quedará resalgada en color verde.
✓ Repetimos el proceso con diferentes voces del grupo y otros grupos.
✓ Si una vez iniciado el proceso de reconocimiento, se tarda más tiempo en dictar la voz, aparecerá un mensaje de rebasamiento del tiempo "Timerout expired".

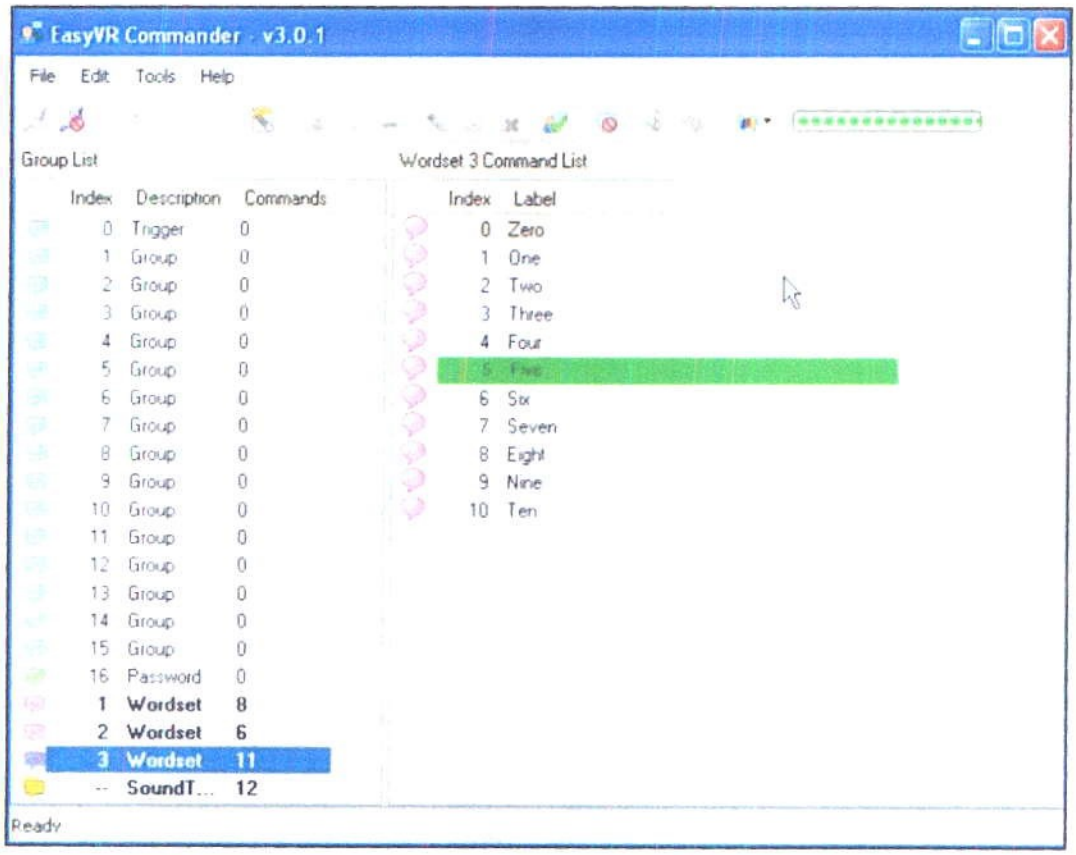
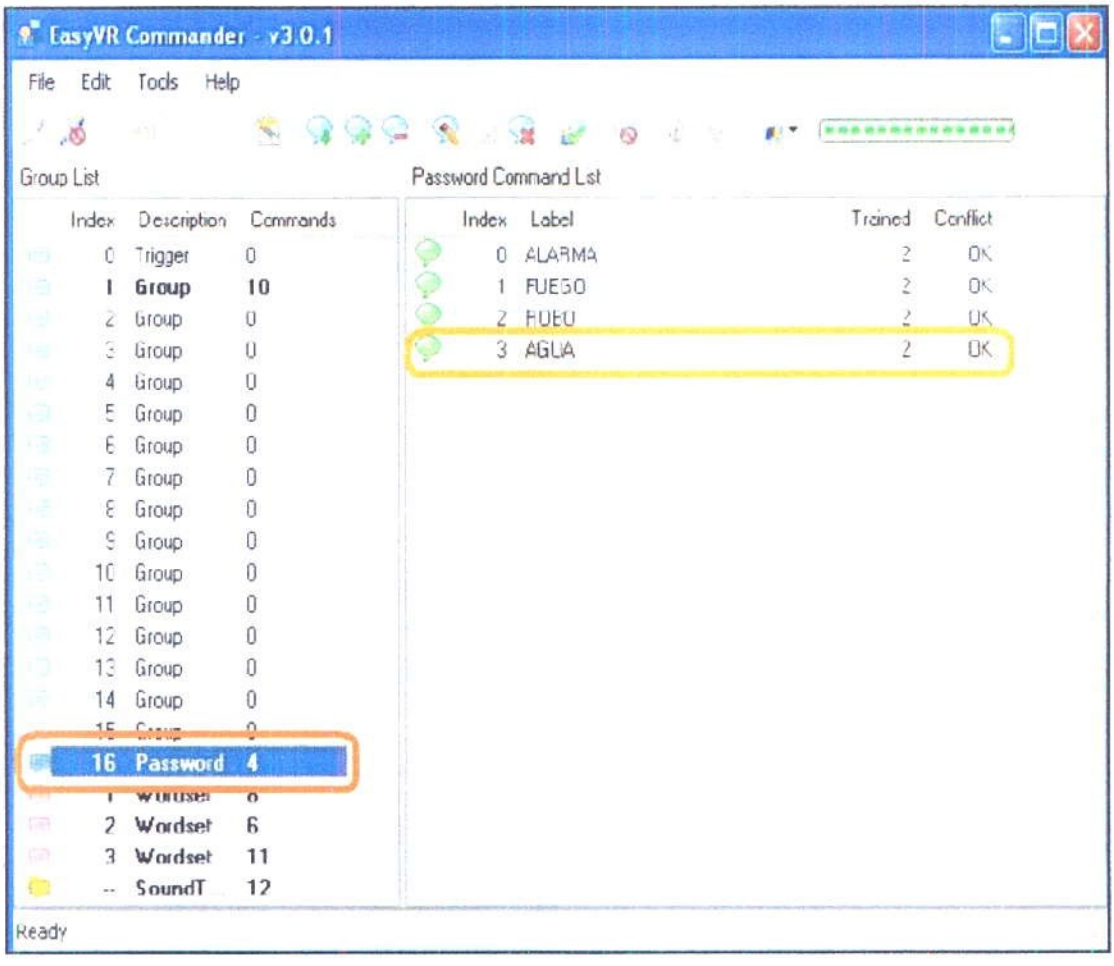


Figura 21. Reconocimiento de Voz. Manual EasyVR Commander

- **Edición de etiquetas y comandos:**

✓ Selección del grupo deseado (Grupo 1).
✓ Con el icono "Add Command" y/o "Insert Command" vamos añadiendo todas las etiquetas o comandos deseados.
✓ Con los iconos "Remove Command" y/o "Rename Coomand" podemos borrar y/o renombrar un comando.
✓ Para crear un comando solo se admiten caracteres ASCII de la A hasta la Z en mayúsculas.
✓ A cada comando se le asigna un número que se corresponde con la posición dentro del grupo.
✓ En el área de grupos, el grupo tendrá en frente el número de comandos que contiene.



F Figura 22. Creación de comandos. Manual EasyVR Commander

- **Ciclo de Enseñanza:**

  Consiste en grabar las voces que se asociarán a los comandos editados anteriormente y que se usarán durante el reconocimiento:

- ✓ Seleccionamos el grupo al que pertenece el comando.
- ✓ Seleccionamos el comando
- ✓ Mediante el ícono "Train Command" se inicia el ciclo de enseñanza.
- ✓ Disponemos de un cierto tempo para dictar la voz que queramos.
- ✓ Esa voz hay que volver a dictarla por segunda vez para tener una mayor garantía en los posteriores reconocimientos.
- ✓ Observaremos que a la derecha del comando aparece un valor que expresa el número de veces que la voz ha sido enseñada, seguido de OK cuando finaliza el ciclo
- ✓ Si rebasamos el tiempo o si la repetición de la voz por segunda vez no corresponde a la primera, aparecerá un mensaje de error.
- ✓ Repetimos el proceso con todos los comandos disponibles en todos los grupos.

- **Reconocimiento de Voz:**

- ✓ Seleccionamos el grupo en que se encuentran las voces a reconocer.
- ✓ Mediante el ícono "Test Group" iniciamos el reconocimiento.
- ✓ Dictamos cualquiera de las voces disponibles en el grupo seleccionado.
- ✓ Si la voz es reconocida, el comando asignado a ella quedará resaltado en verde.
- ✓ Repetms el proceso con diferentes voces para el reconocimiento de voz.
- ✓ Si una vez se inicia el proceso de reconocimiento, se tarda más de 5 segundos en dictar la voz, aparecerá un mensaje de rebasamiento del tiempo "Timerout expired"

Una vez tenemos una serie de comandos creados en el módulo EasyVR, procedemos al reconocimiento de las mismas utilizando los mismos pasos mencionados para el reconocimiento de voces predefinidas.

8. DISEÑO DEL MODULO DE CONTROL ACTIVADO POR LA VOZ

A continuación describiremos las diferentes fases del desarrollo del modulo de control activado por la vcz para el prototipo EMMRA-1

8.1 Implementación del hardware EasyVR Shield 3.0

Las piezas del la tarjeta electrónica vienen separadas para facilitar la implementación por separado del módulo de reconocimiento de voz y el escudo para conectividad con la tarjeta electrónica Arduino, como se muestra en la siguiente figura:

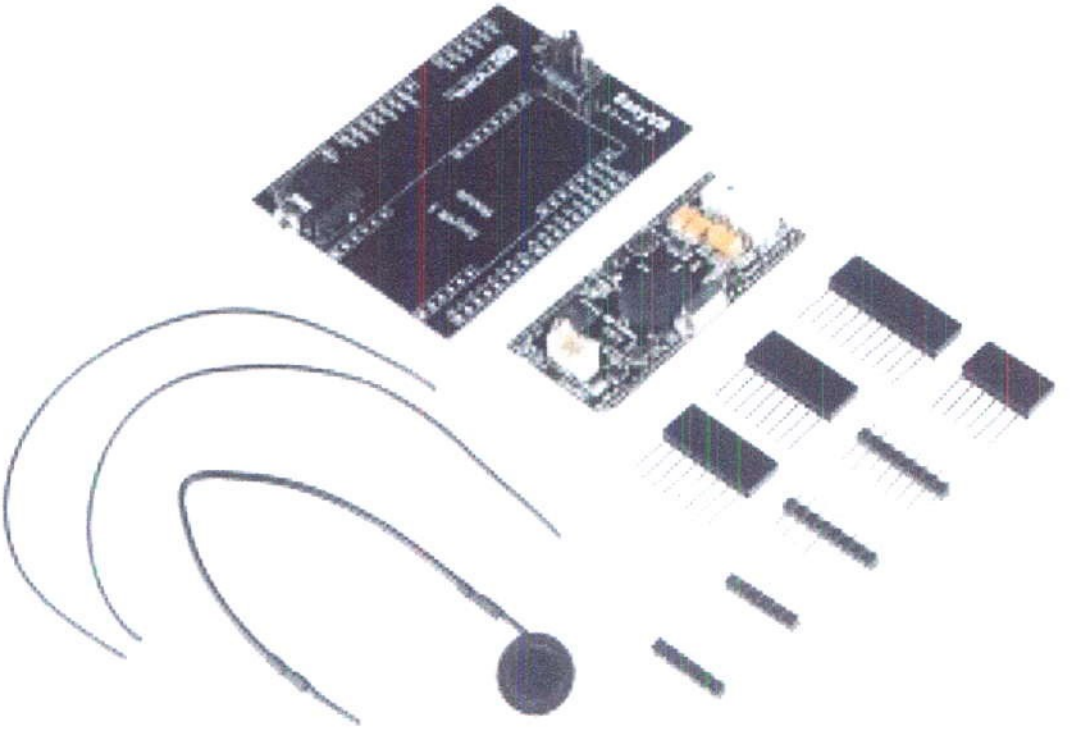


Figura 23. Piezas del Módulo de Voz EasyVR. www.sparkfun.com/products

Unimos las piezas con puntos de soldadura para crear todo el conjunto de la EasyVR Shield 3.0. Así, el módulo de reconocimiento de voz, queda en la parte superior del la tarjeta electrónica escudo, compatible con las tarjetas electrónicas Arduino.

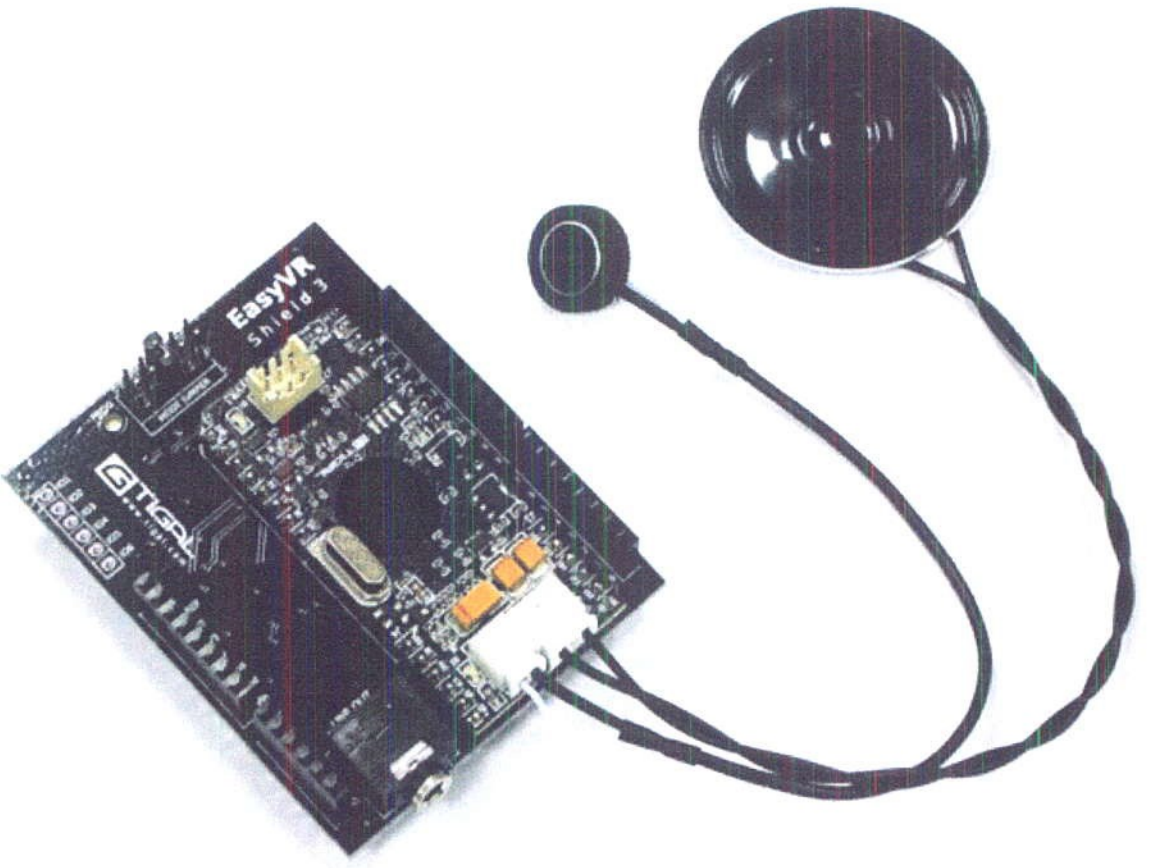


Figura 24. EasyVR Shield 3.0. Manual Usuario EasyVR 3.0

El escudo **EasyVR Shield 3.0** es una tabla electrónica diseñada para las tablas electrónicas de Arduino. Los pines de la cara inferior se ensamblan con precisión en la board **Arduino Uno R3**, **Arduino Mega, y Arduino Leonardo.**

La board EasyVR Shield 3.0 requiere de varios accesorios para su implementación: Micrófono, Audífono, Cable USB tipo A con USB tipo B, Amplificador.

- cable USB para la programación del módulo a través del PC.



Figura 25. Cable USB para interconexión del Arduino al PC

- Microfono original del EasyVR Shield.



Figura 26. Microfono original del EasyVR Shield 3.0

- Bafles internos de un amplificador de audio Genius



Figura 27. Parlantes Genius adaptados

8.2 Implementación del hardware Arduino Mega 2560

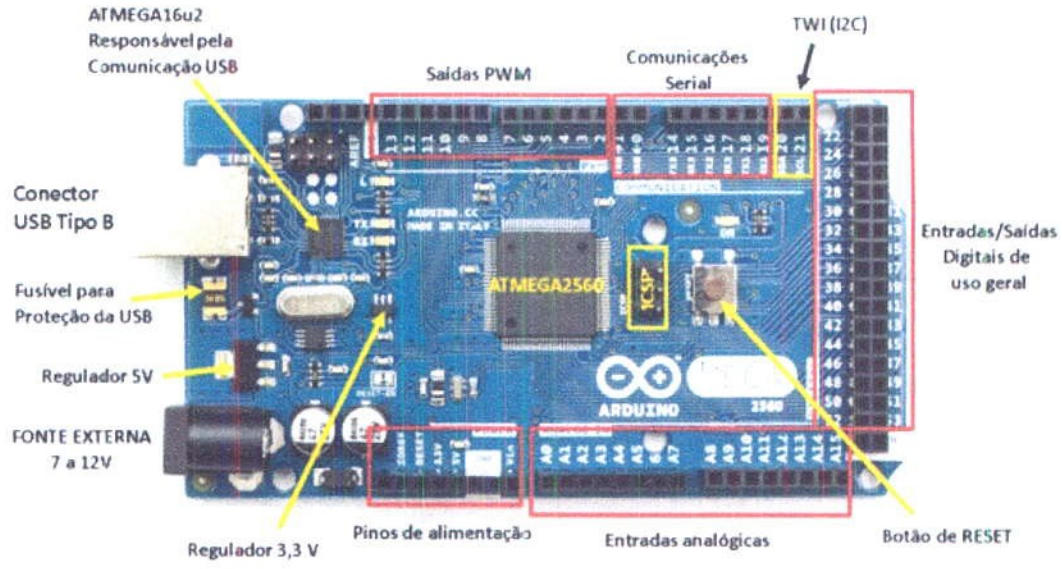


Figura 28. Tarjeta Arduino MEGA 2560. http://www.embarcados.com.br/arduino

La tabla electrónica de **Arduino Mega** cuenta con los siguientes componentes:

- ✓ **Microcontrolador ATmega2560** que programaremos con el **Software Arduino** de una manera flexible y sencilla.
- ✓ **Conexión USB, Microcontrolador ATmega16U2, Fusible para protección de la conexión USB**, componentes para permitir una adecuada conexión de la tarjeta con el PC.
- ✓ **Fuente externa de 7 a 12V**, la cual no está implementada en nuestro prototipo.
- ✓ **Regulador de 3.3V**, utilizado para la salida de energía a los indicadores Led.
- ✓ **Salidas PWM** Se utilizan las salidas 3, 5 y 11 para la comunicación con el módulo de control del equipo de transmisión de movimiento a la silla giratoria.
- ✓ **Comunicación Serial** en el pin 0 (RX) y pin 1 (TX) para la comunicación entre la tarjeta Arduino Mega 2560 y la EasyVR Shield 3.0.
- ✓ **Comunicación I2C** no implementada en nuestro prototipo.
- ✓ **Entradas y Salidas Digitales** implementadas para el envío de señales a los servomotores, la comunicación y los indicadores led de alta luminosidad,
- ✓ **Entradas Analógicas** utilizadas para la recepción de los datos de los SFR y los datos de los sensores de señal muscular.
- ✓ **Pines de Alimentación** utilizadas para la alimentación del módulo y los parlantes genius.

- **Detalle de pines del Arduino Mega 2560 utilizados en el prototipo:**

✓ **Comunicación pines 0 – 1:** (circulo violeta) Pines Rx y Tx para la transferencia de datos entre el EasyVR y la tarjeta Arduino Mega 2560
✓ **Salidas pines 5 y 11:** (**círculo amarillo**) Pines de comunicación con el módulo de control del equipo de transmisión de movimiento.

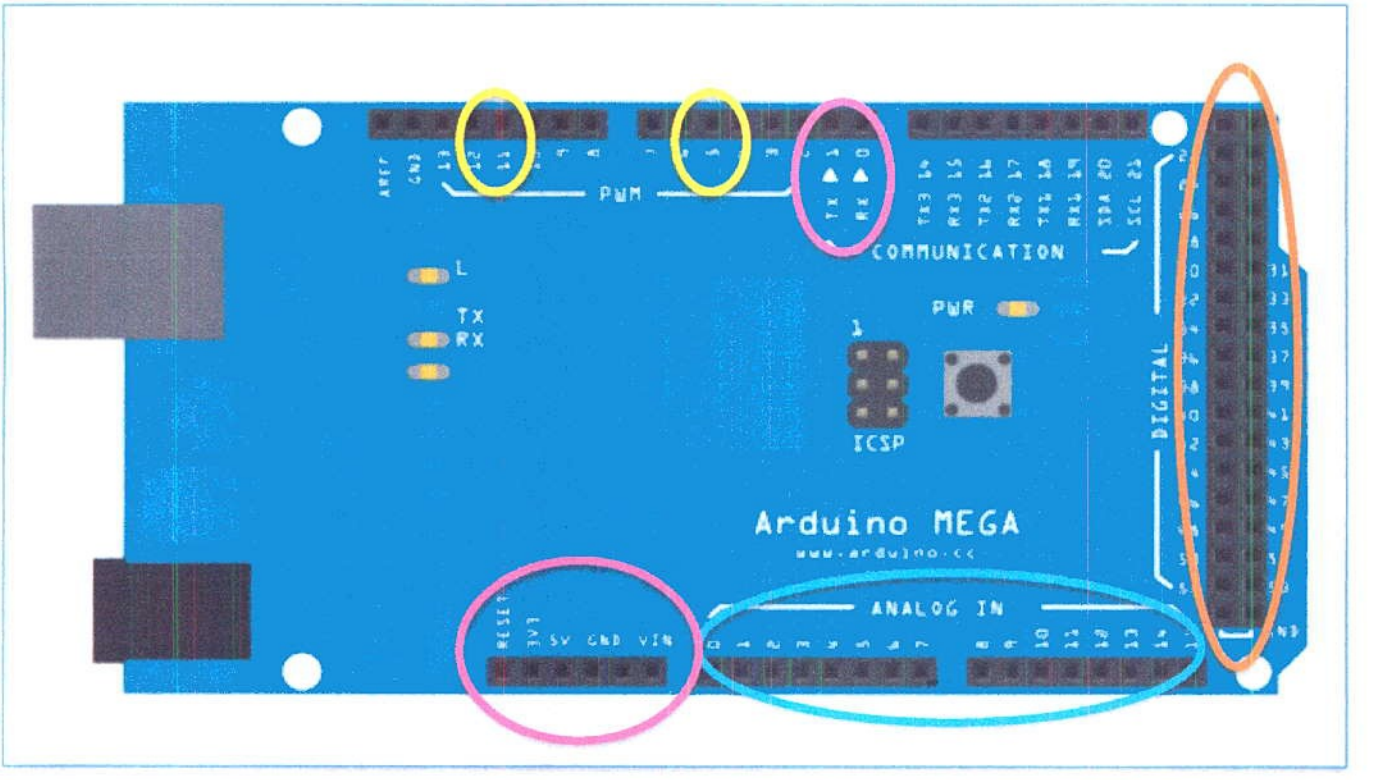


Figura 29. Tarjeta Arduino MEGA 2560. Pin 5 y 11

- **Entradas Analógicas** (**circulo azul**)**: A0 – A5:** Entrada de la señal de los Sensores de Fuerza Resistivos Mano izquierda. **A11 – A15:** Entrada de las señales de los Sensores de Fuerza Resistivos Mano Derecha.
- **E/S Digitales**: (circulo naranja)**: 52 - 54:** Pines para el control de los indicadores LED rojo y LED verde para comunicación con el paciente. **24 - 32:** Pines para la salida de la señal digital que control a los servomotores de la mano izquierda. **40 – 48:** Pines para la salida de la señal digital que control a los servomotores de la mano derecha.
- **Power: Vin:** Entrada Positiva de Energía de la fuente de alimentación.
- **GND:** Entrada negativa de Energía de la fuente de alimentación.
- **GND:** Conexión a tierra de los sensores de fuerza digitales.
- **5V:** Conexión de voltaje positivo de los sensores de fuerza digitales.
- **3.3V:** No se implementa.
- **Reset:** Para reiniciar el MRV.

8.3 Implementación del software Audacity

El Software Audacity es un software libre, se puede descargar gratuitamente de internet. Permite grabar, reproducir, modificar, mezclar y almacenar sonidos en diferentes tipos de archivos, para nuestro proyecto implementamos archivos wav requeridos por el EasyVR Shield 3.0.

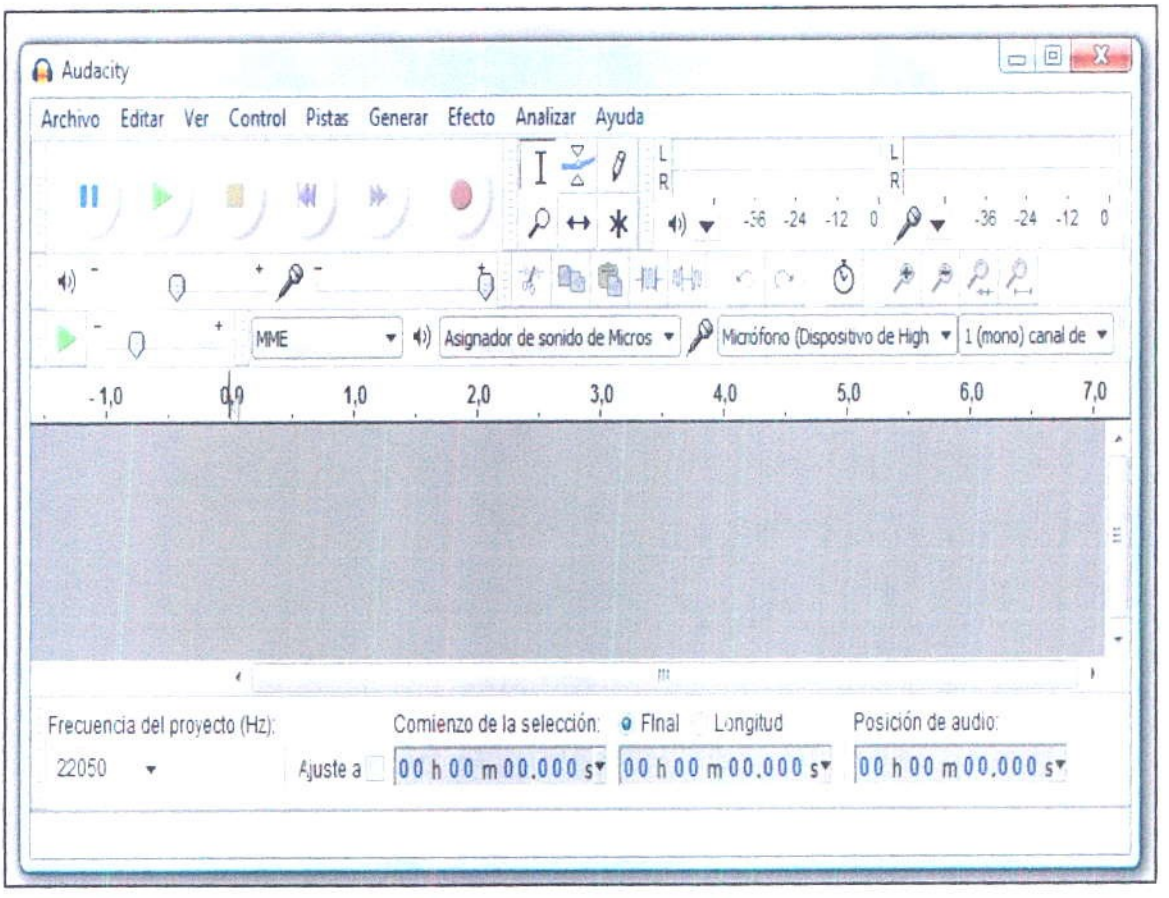


Figura 30. Software Audacity

Con el software Audacity crearemos archivos de sonido para la elaboración de Tablas de Sonido, según las especificaciones requeridas por la board EasyVR Shield 3.0.

✓ Abrimos el Software Audacity y damos clic en Editar/Preferencias.

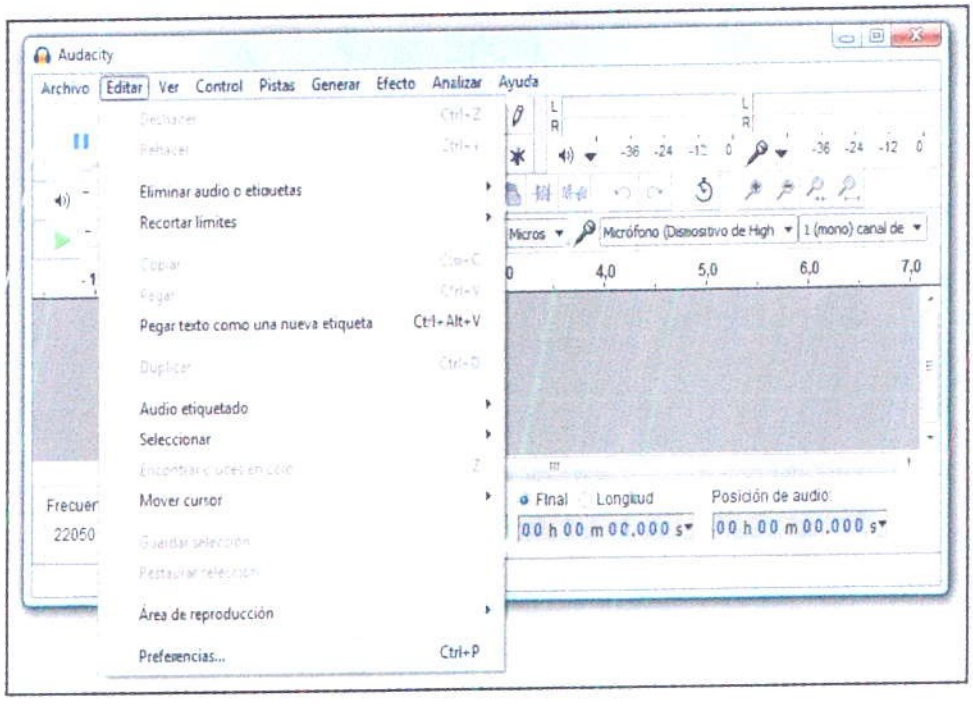


Figura 31. Editar preferencias Software Audacity

✓ Establecemos una Frecuencia de Muestreo: 22050 Hz, Formato de Muestra: 16 bits, y Monocanal. Requeridos por la EasyVR Shield 3.0

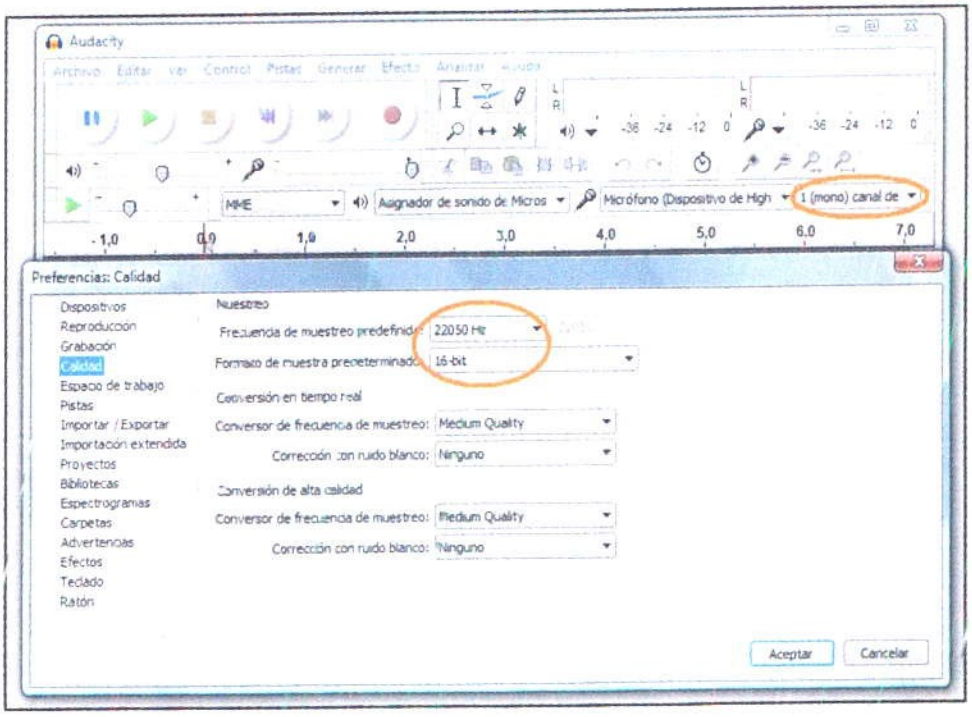


Figura 32. Selección de preferencias para EasyVR Shield 3.0

- ✓ Regresamos al Software Audacity y grabamos el comando o instrucción que deseamos escuche el usuario del EMMRA-1.

  Primer Instrucción: "***Indique código de inicio***".

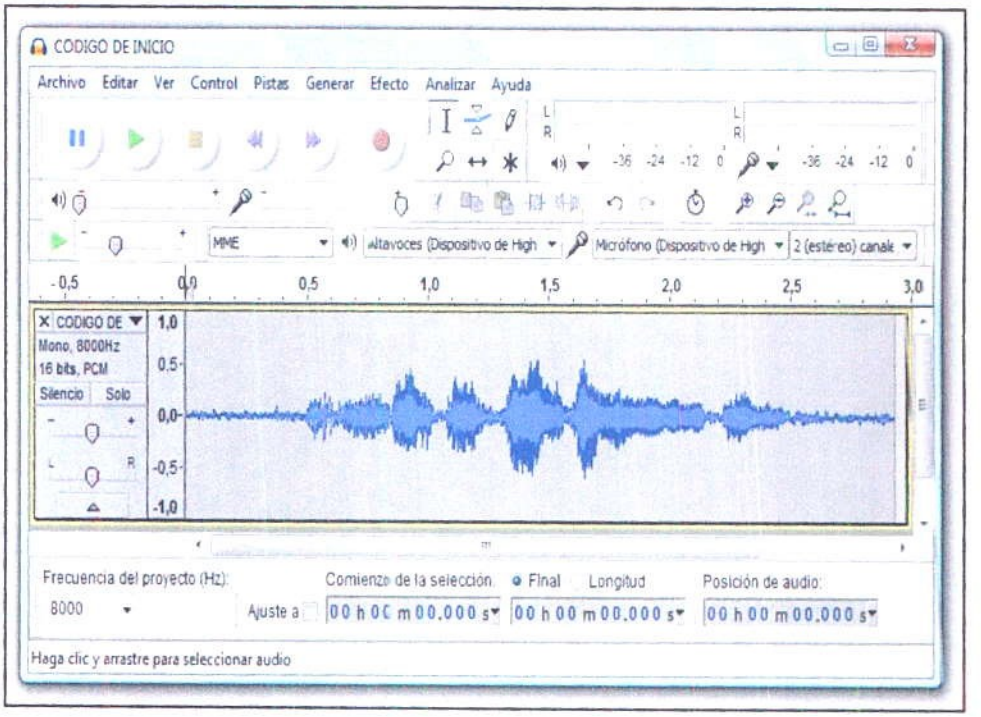


Figura 33. Grabación comando Código de Inicio

- ✓ Damos clic en Archivo/Exportar para crear nuestra tabla de sonidos

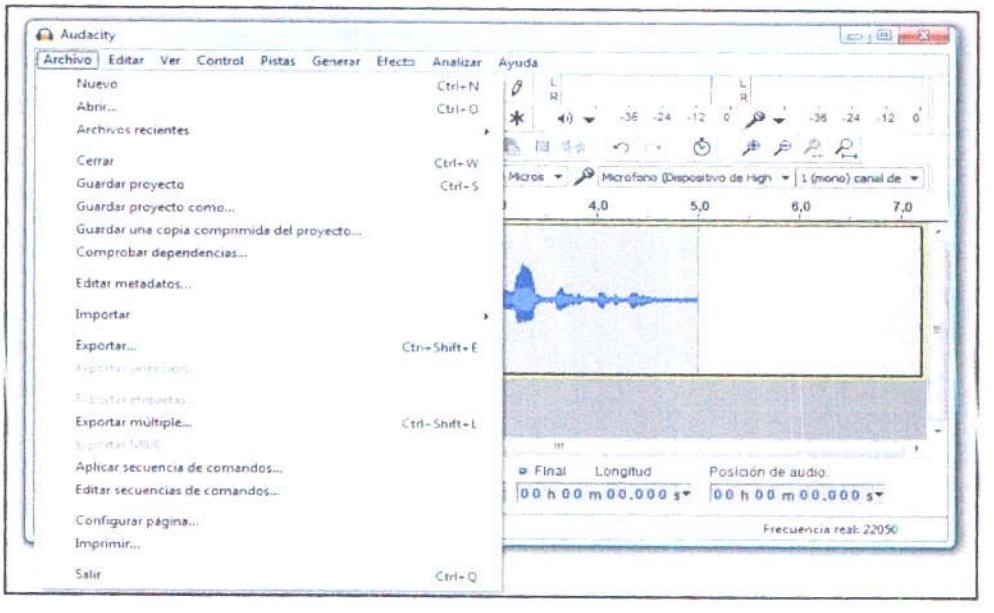


Figura 34. Exportar archivos en Audacity

- ✓ Almacenamos el archivo con el nombre de CODIGO INICIAR, quedará en formato WAV, al interior de una carpeta previamente creada en nuestro PC llamada "INSTRUCCIONES EMMRA_1"

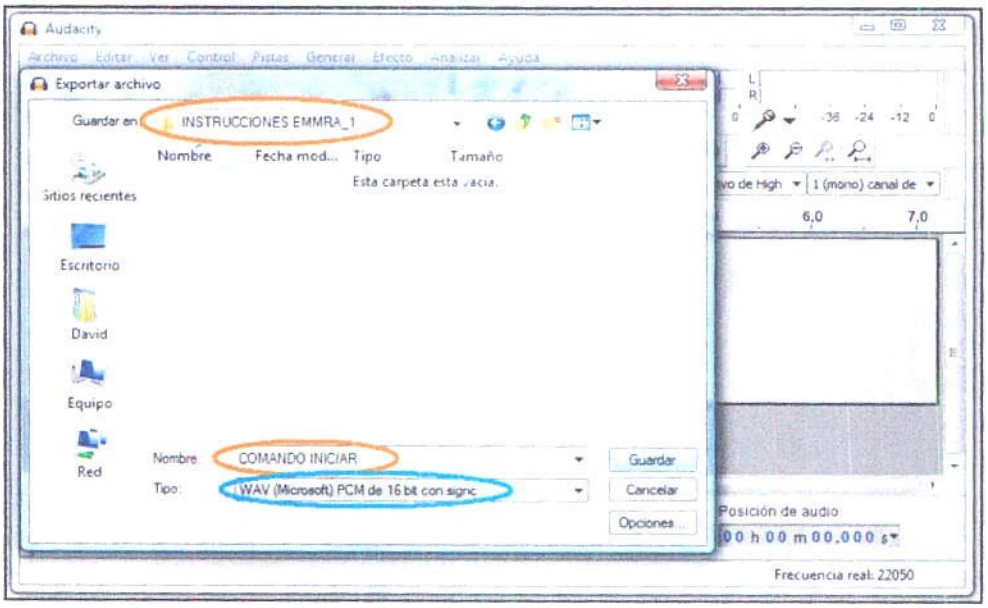


Figura 35. Creación de archivo para almacenar la tabla de sonidos

- ✓ Al guardar el archivo, se activará la ventana Editar Metadatos, con los cuales realizaremos una caracterización del archivo guardado.

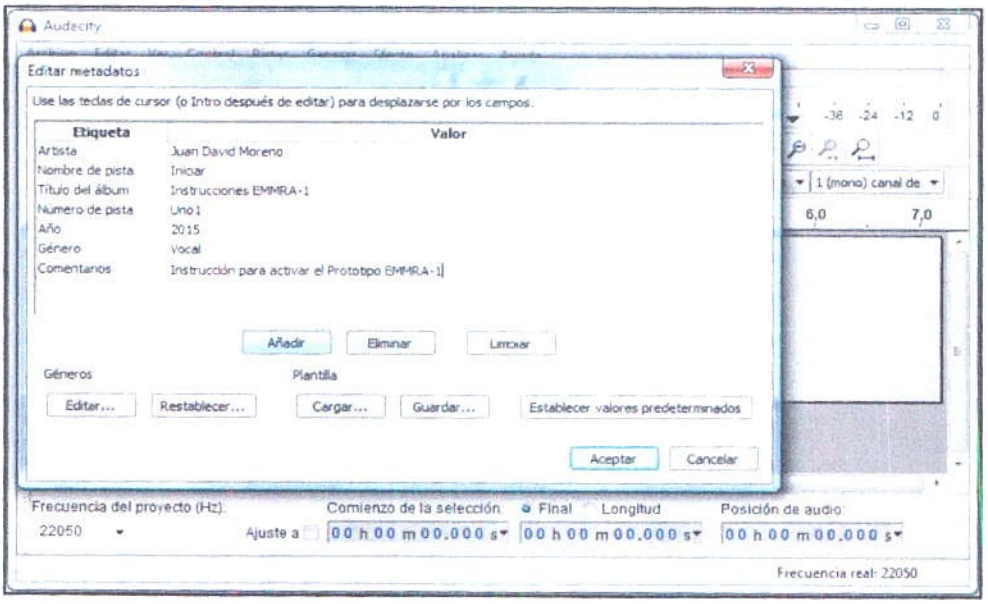


Figura 36. Caracterización del sonido grabado

- ✓ De esta forma tendremos nuestro primer archivo en formato WAV almacenado en la carpeta de INSTRUCIONES EMMRA-1.

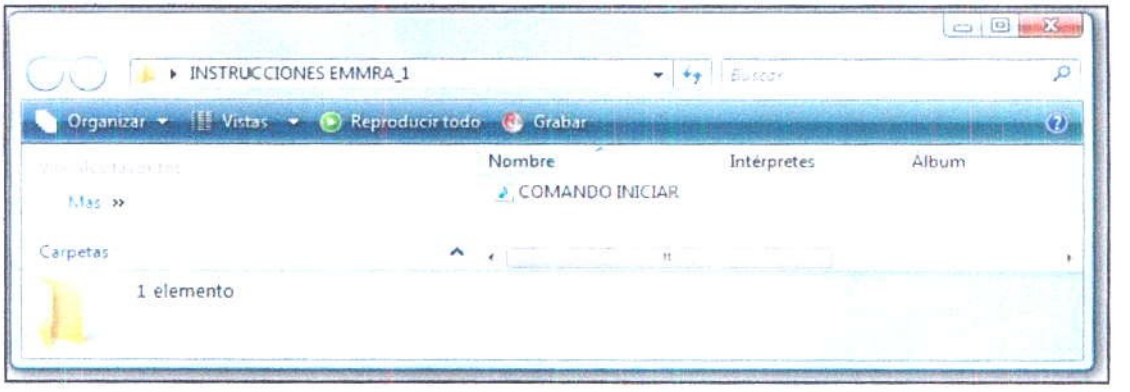


Figura 37. Archivo Código de Inicio almacenado

- ✓ Creamos todas las instrucciones requeridas para comunicarnos con el usuario del prototipo EMMRA-1

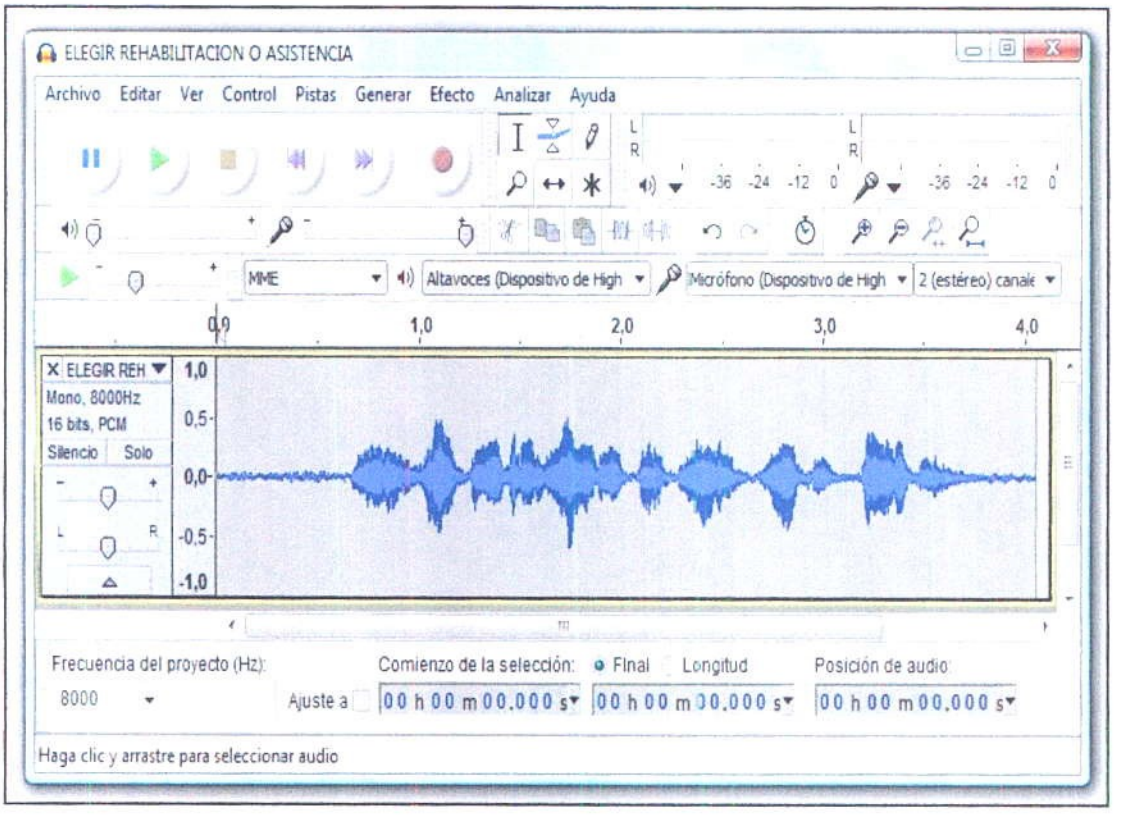


Figura 38. Grabación del comando Elija entre Rehabilitación o Asistencia

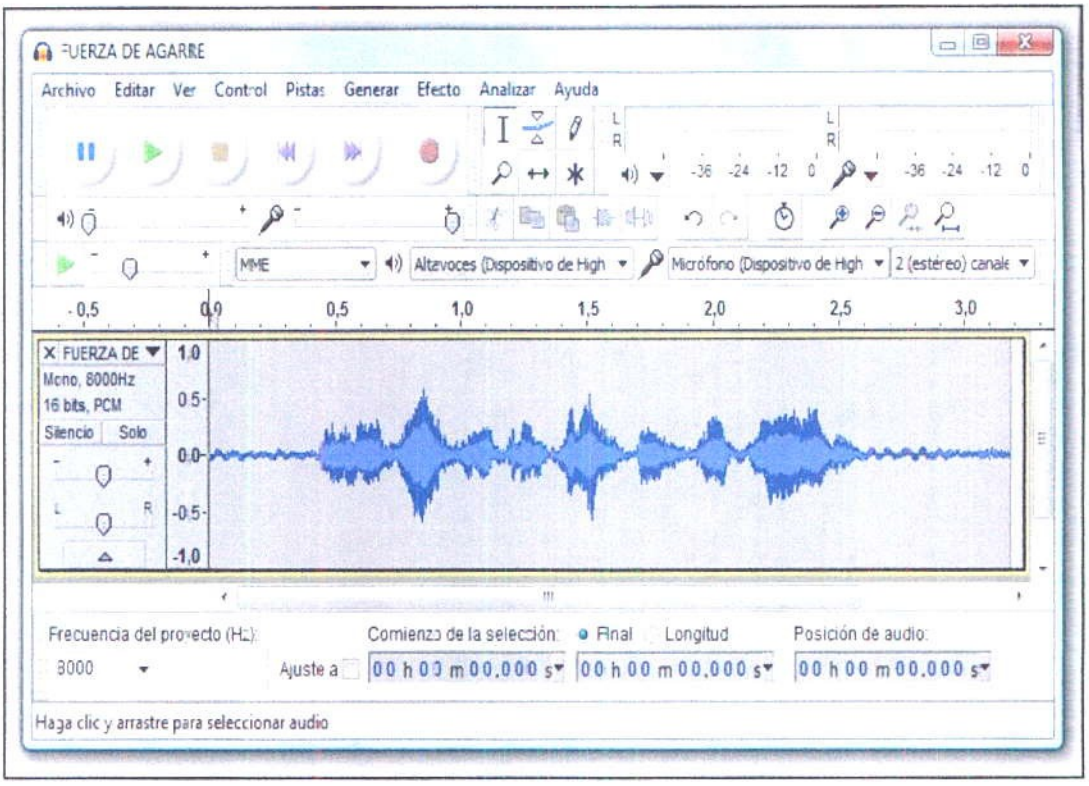


Figura 39. Grabación del comando Elija Fuerte o Suave

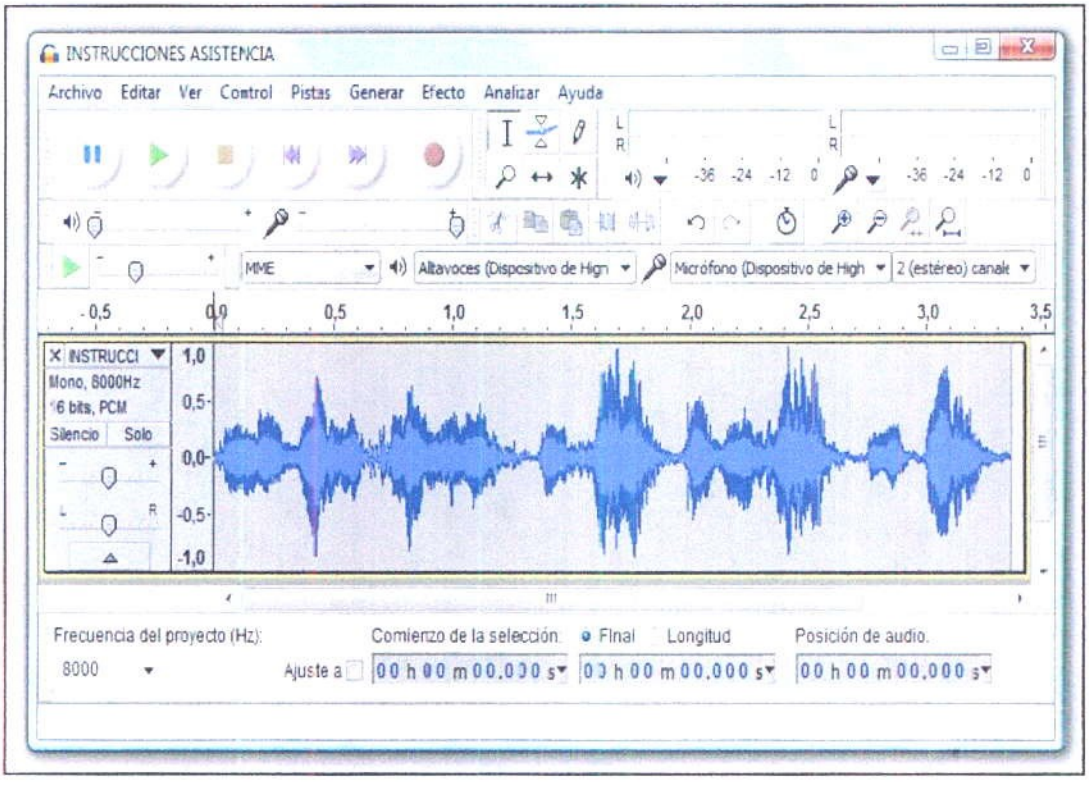


Figura 40. Grabación del comandos para la aplicación de asistencia

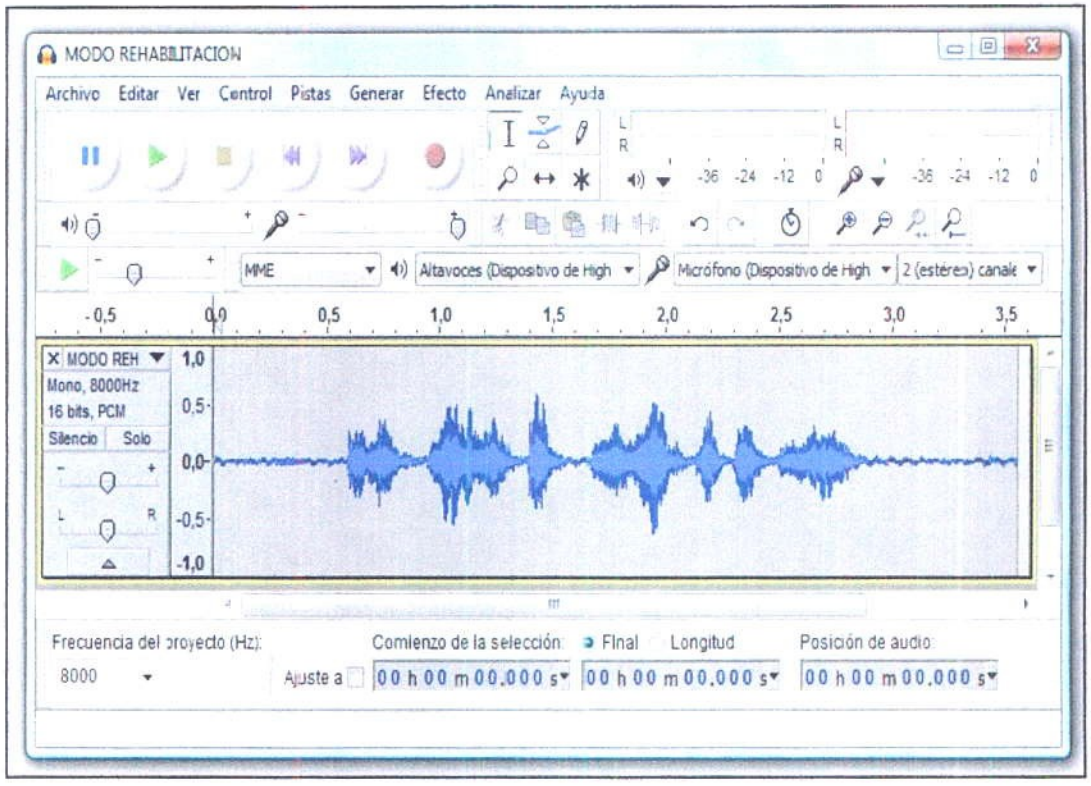


Figura 41. Grabación del comando para aplicación de rehabilitación

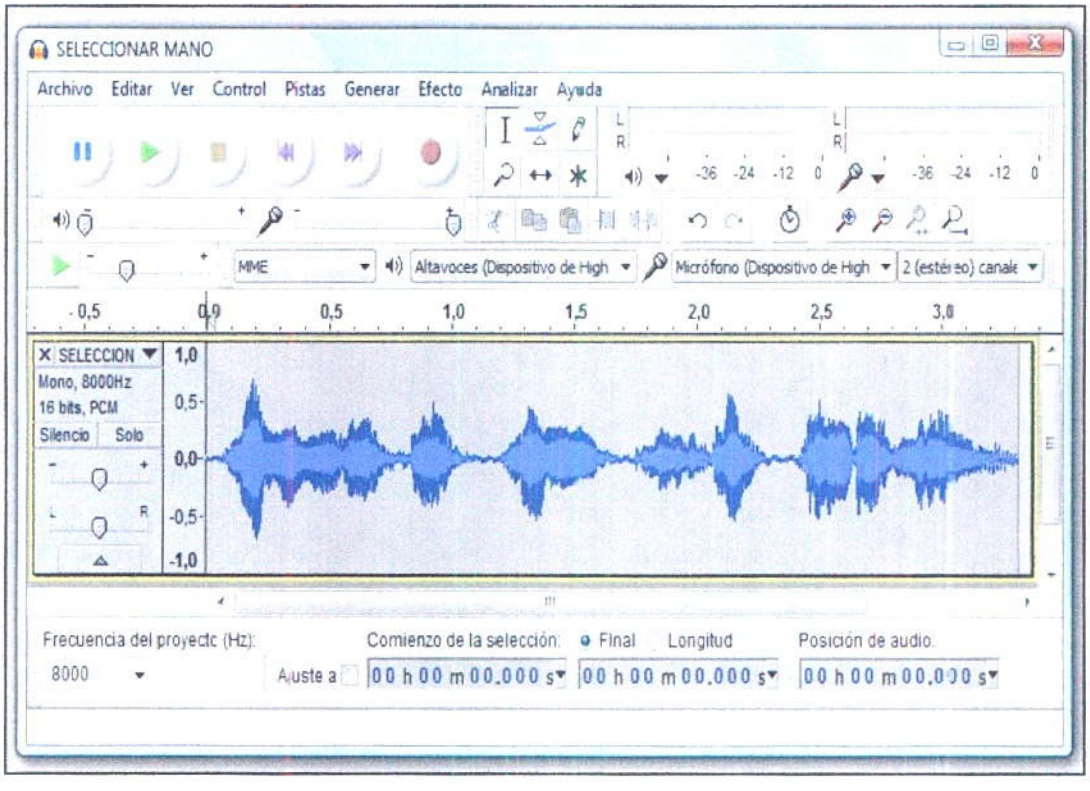


Figura 42. Grabación del comandos para seleccionar la mano a utilizar

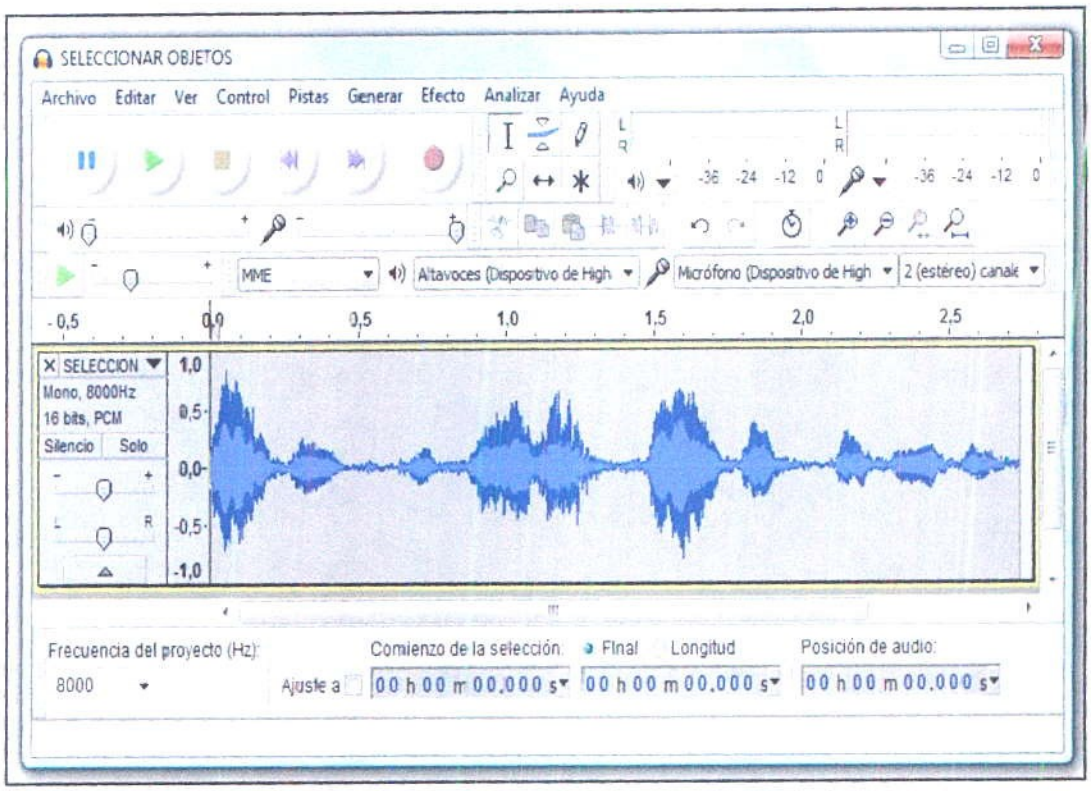


Figura 43. Grabación del comando para seleccionar objetos

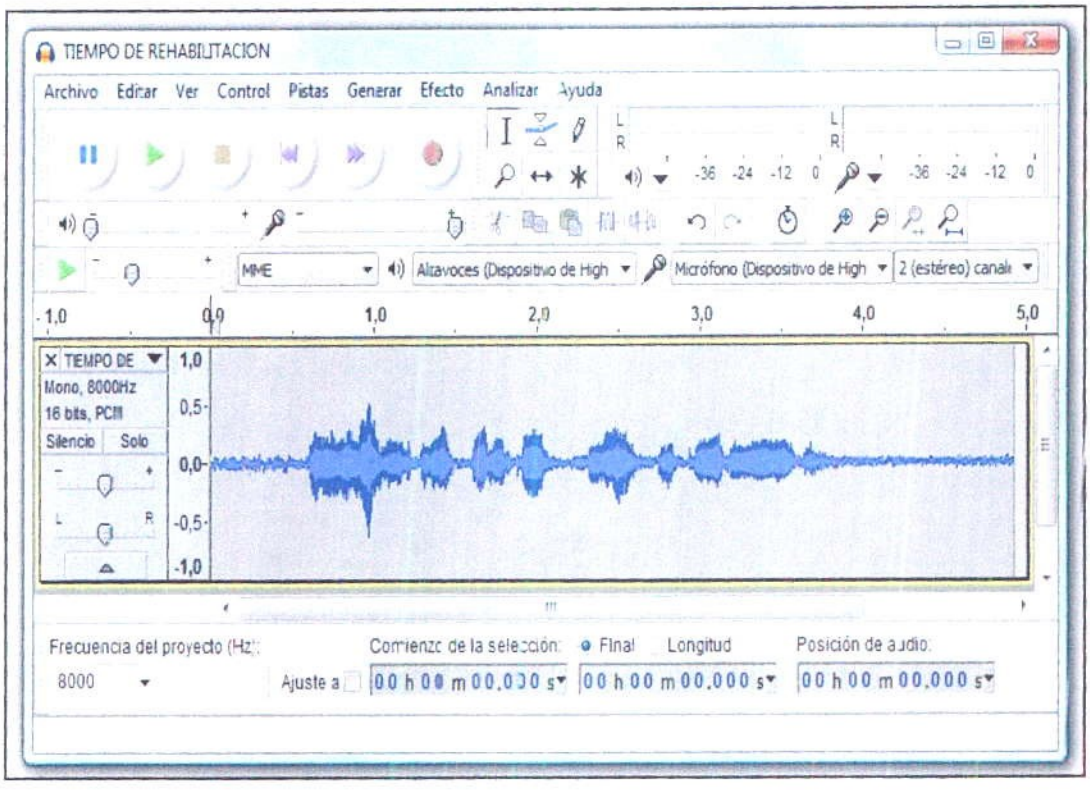


Figura 44. Grabación del comandos para el tiempo de rejabilitación

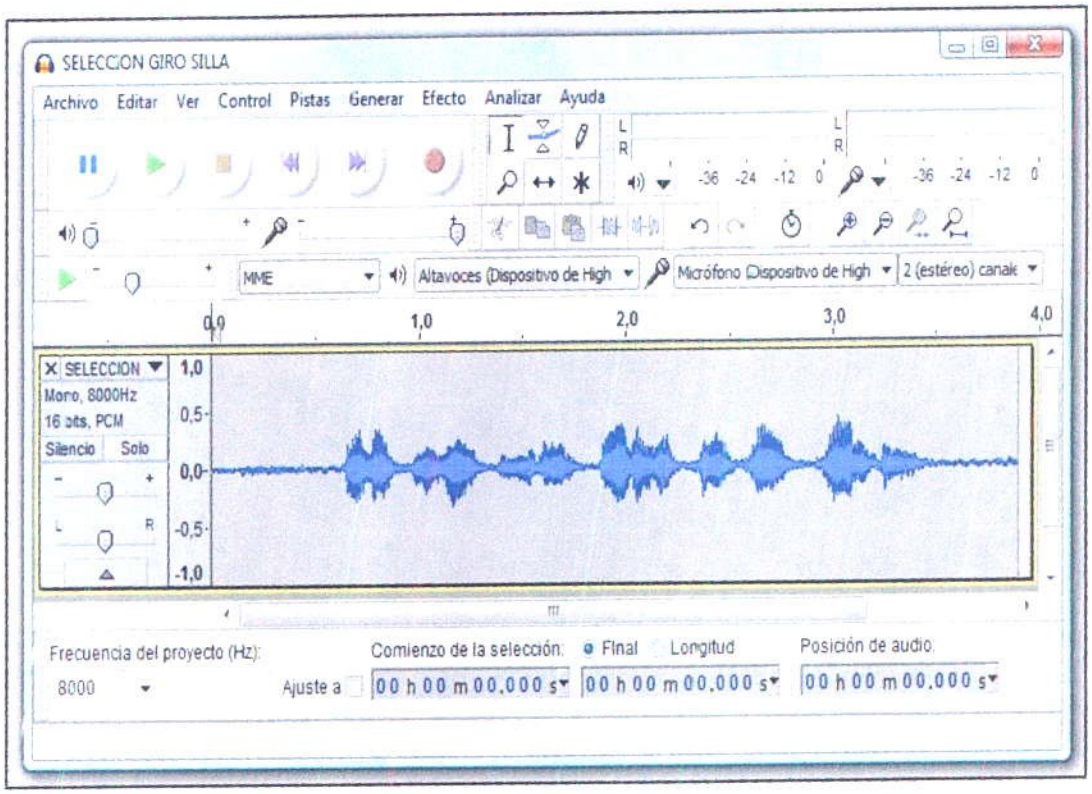


Figura 45. Grabación del comando para girar la silla

✓ Repetimos los pasos anteriores para crear todos los comandos o instrucciones para el usuario del EMMRA-1

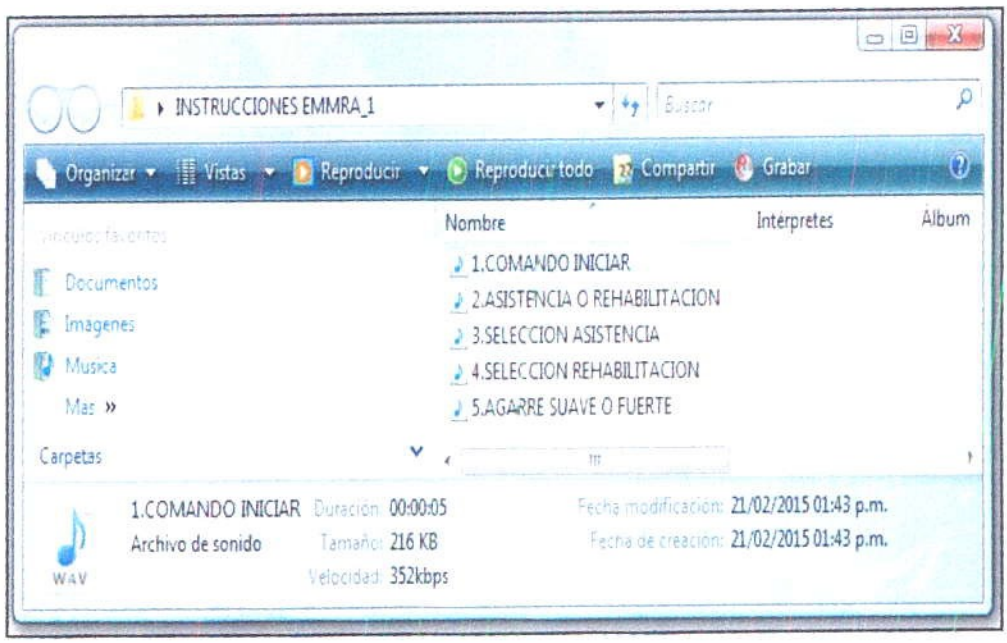


Figura 46. Creaciôm de Comandos de Voz

8.4 Implementación del software Sensory QuickSynthesis 5

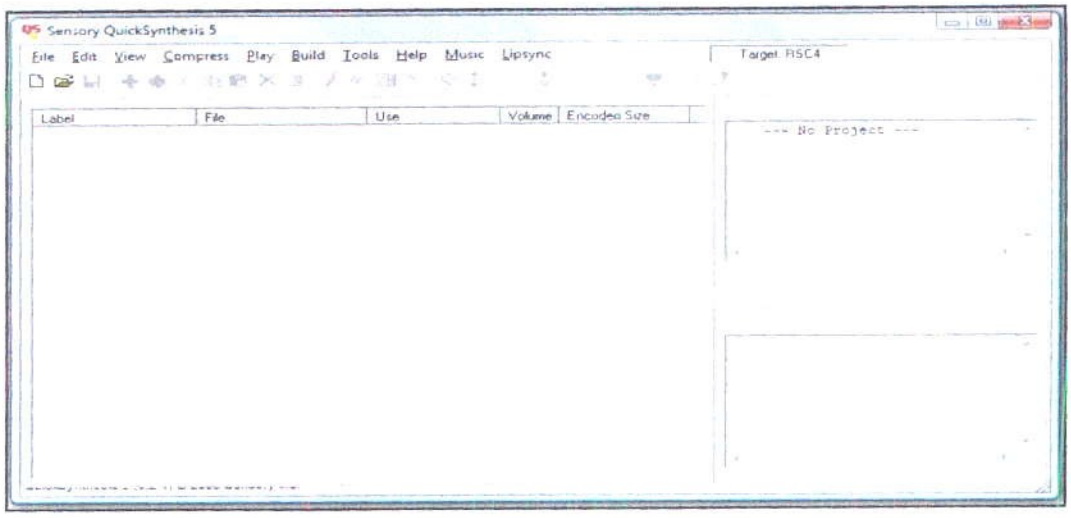


Figura 47. Software QuickSynthesis 5

El Software QuickSyntesis 5 (QS5) de la firma Sensory, es una aplicación para crear las tablas de sonido que grabaremos en el MRV. Esta aplicación se instala cuando instalamos el programa EasyVR Commander.

- ✓ Abrimos la aplicación dando clic derecho y eligiendo "**ejecutar como administrador**", esto es necesario para que permita realizar todos los pasos sin presentar errores.
- ✓ Damos clic en "**crear nuevo proyecto**".
- ✓ Seleccionamos la Tarjeta de la familia del dispositivo **RSC4 Family**
- ✓ Damos clic en OK

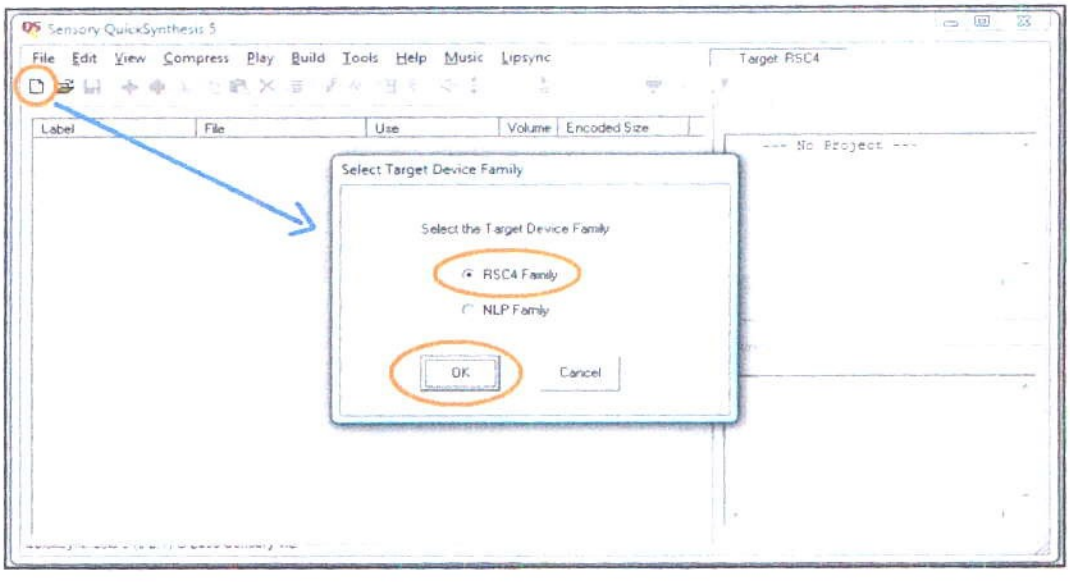


Figura 48. Creación de un nuevo archivo

- ✓ Seleccionamos el archivo con la tabla de sonidos "**INSTRUCCIONES EMMRA-1**", la cual creamos previamente con el software Audacity.
- ✓ Digitamos el nombre de la tabla de sonidos que estamos creando "**COMANDO DE INSTRUCCIONES**" para cargarla en el MRV

- ✓ Damos clic en Abrir

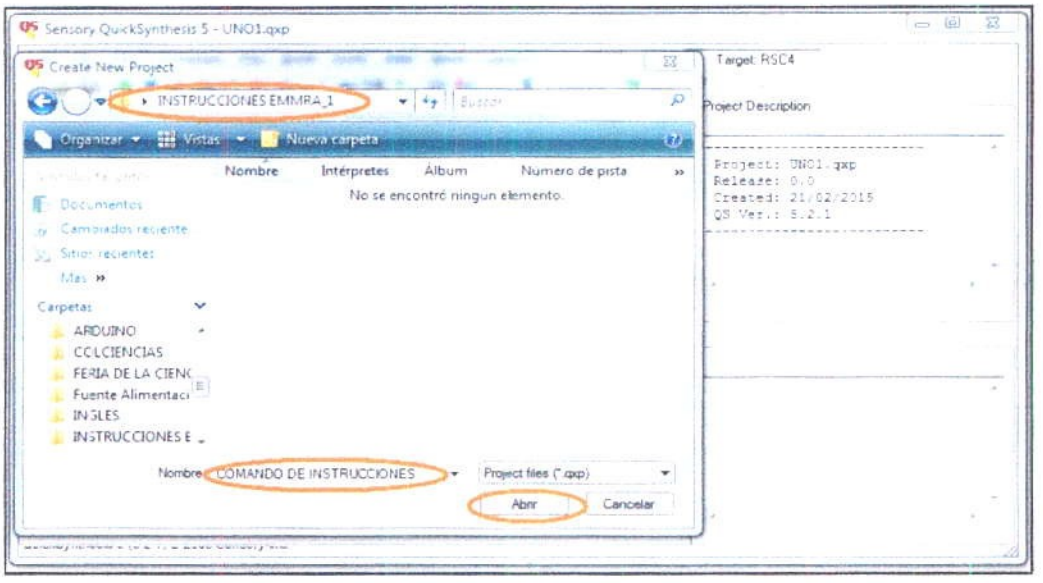


Figura 49. Importación de sonidos

- ✓ Damos clic en adicionar archivo.
- ✓ Seleccionamos el primer archivo en formato WAV.
- ✓ Damos clic en Abrir

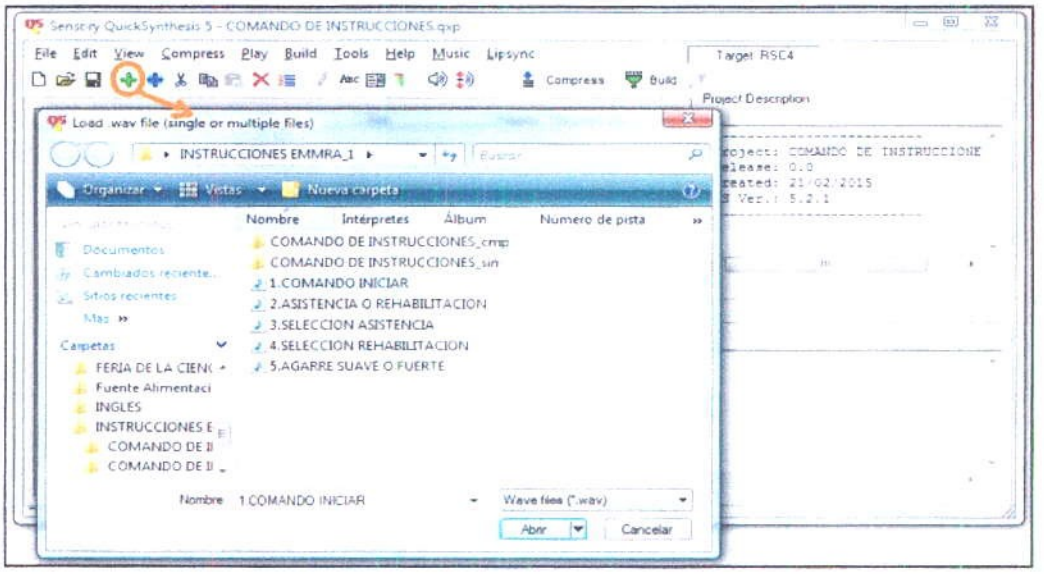


Figura 50. Selección de sonidos a importar

✓ Seleccionamos el tipo de compresión **SX- 8K (7800 bps)**

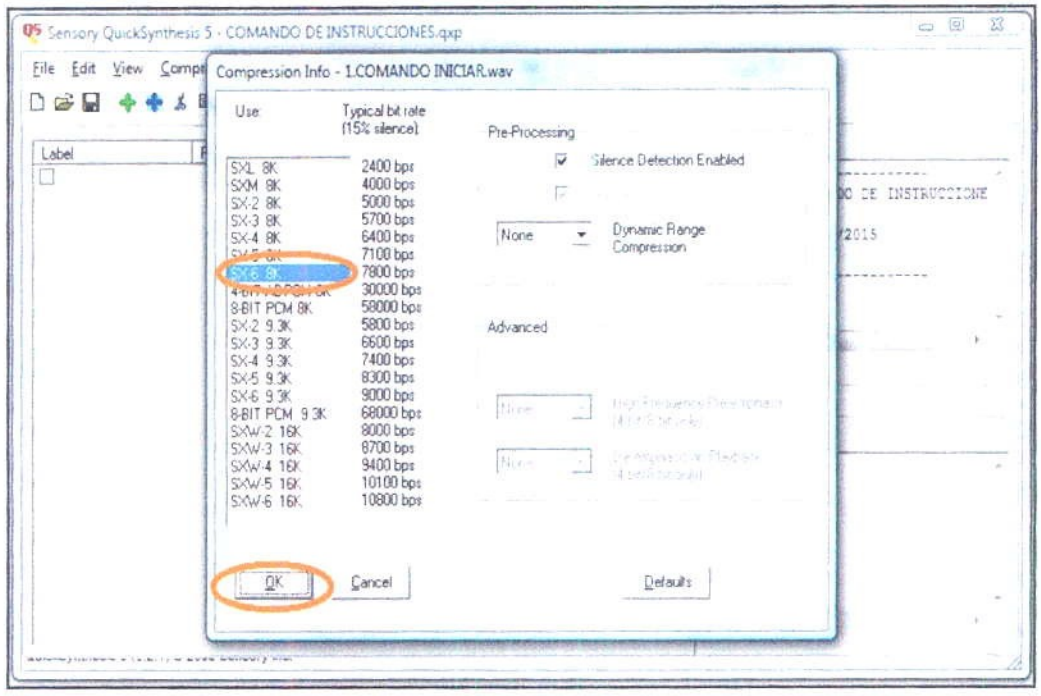


Figura 51. Selección de la compresión

✓ Ingresamos el archivo "**COMANDO INICIAR**".
✓ Damos clic en Abrir

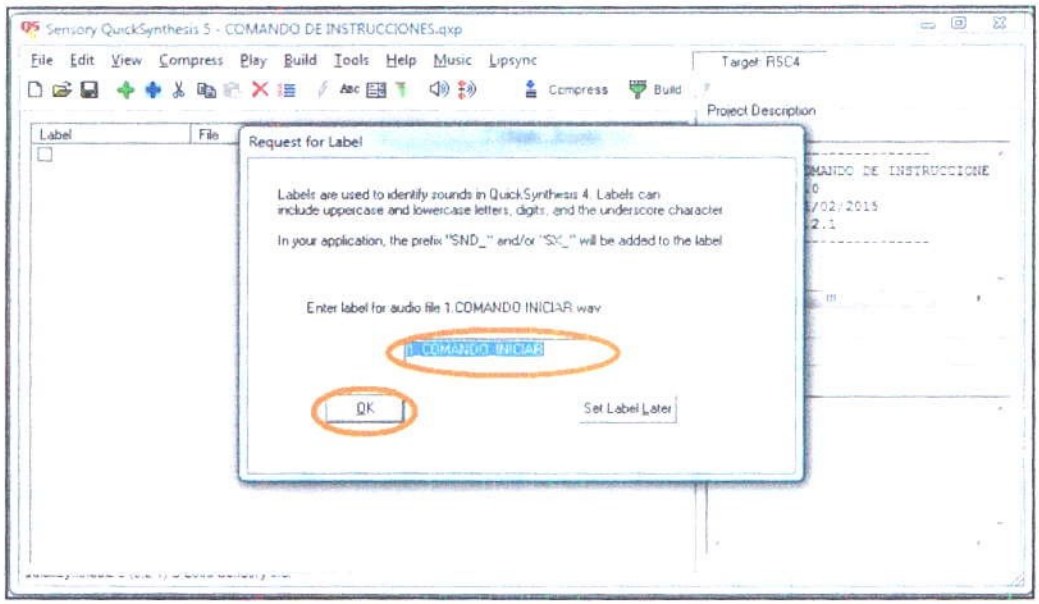


Figura 52. Selección del comando de voz

✓ Se presenta el archivo seleccionado en la tabla de sonidos que estamos creando.

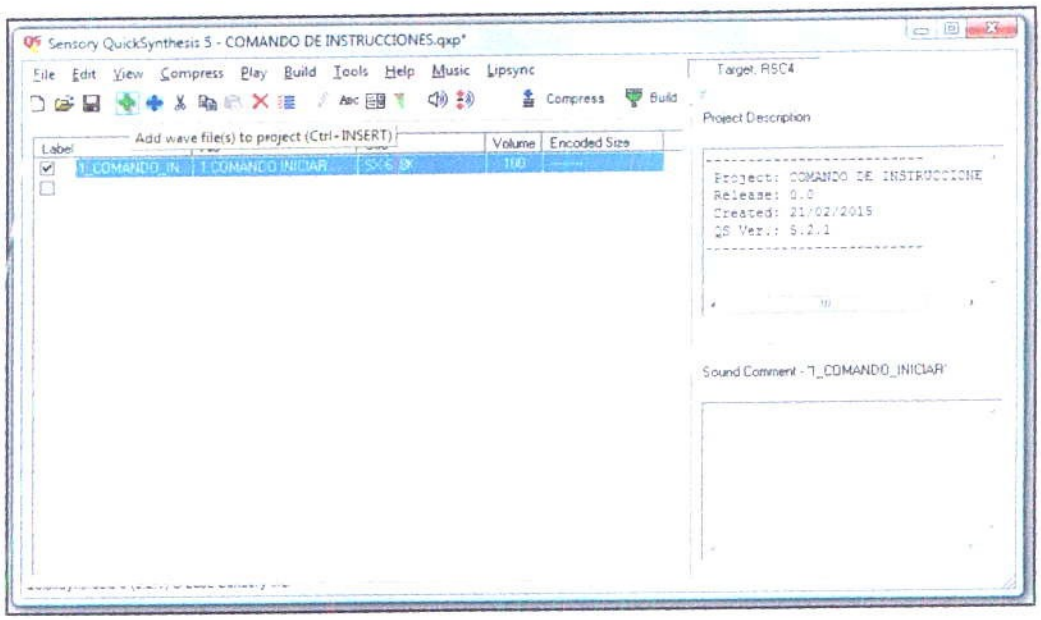


Figura 53. Sonido Almacenado

✓ Damos clic nuevamente en el ícono Adicionar archivo.
✓ Repetimos los pasos anteriores hasta adicionar todos los archivos que deseamos en la nueva tabla de sonidos.

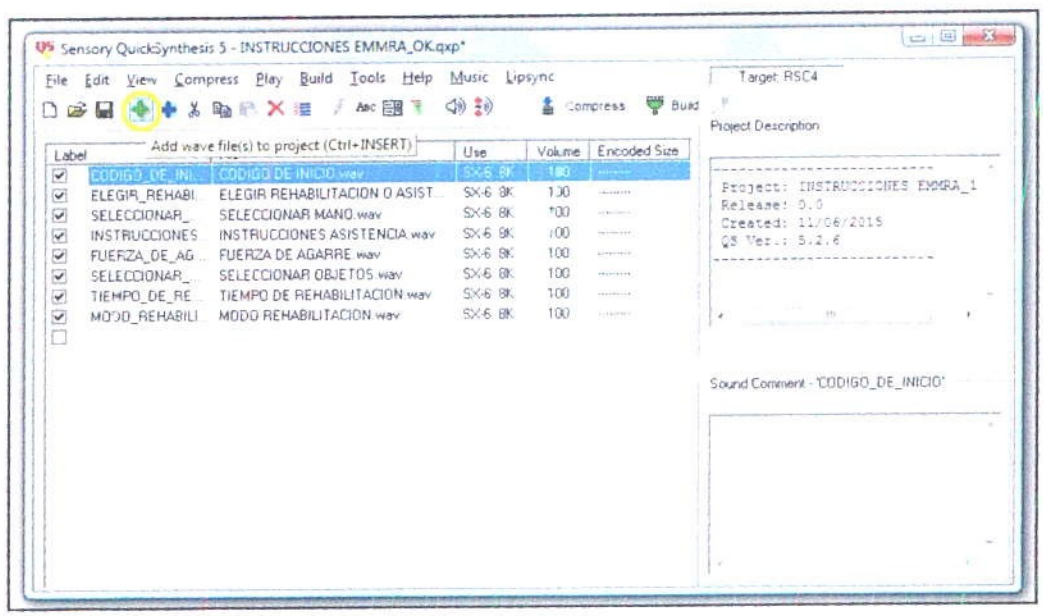


Figura 54. Construcción de tablas de sonidos

- ✓ Seleccionamos todos los archivos que hemos subido a la tabla.
- ✓ Damos clic en el ícono **Compress**

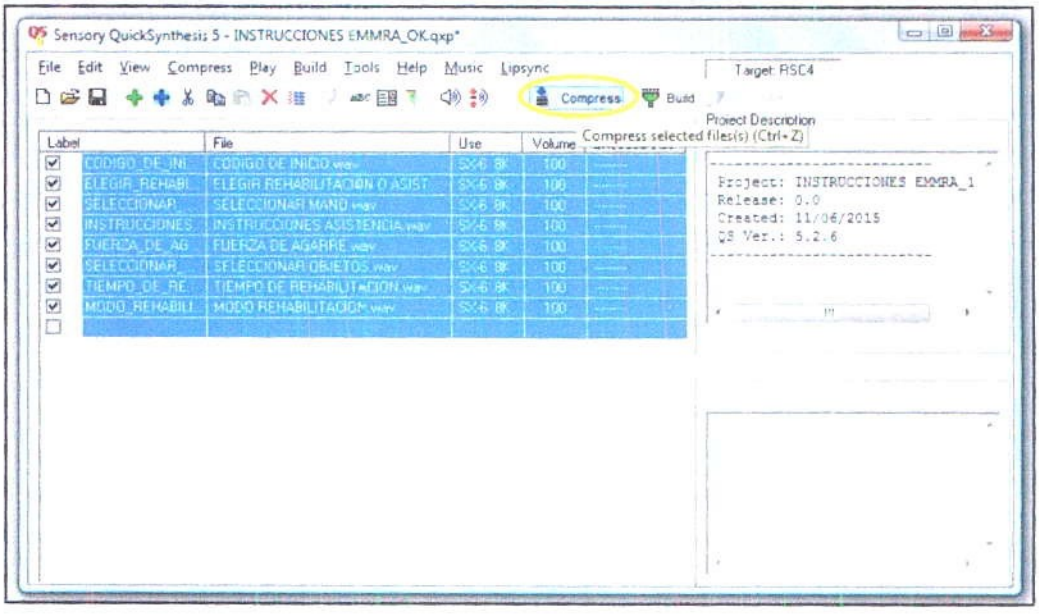


Figura 55. Compresión del archivo

- ✓ La compresión de cada archivo seleccionado se realiza automáticamente.
- ✓ Damos clic en el ícono **OK**

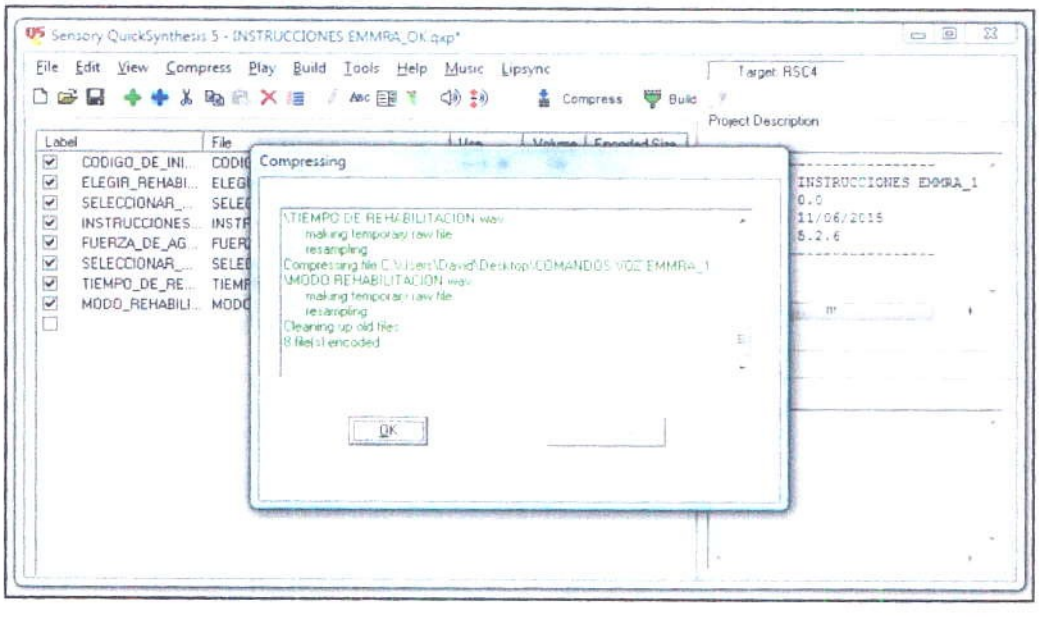


Figura 56. Compresión del archivo

- ✓ Damos clic en el ícono **Guardar**.
- ✓ Damos clic en el ícono **Buid**

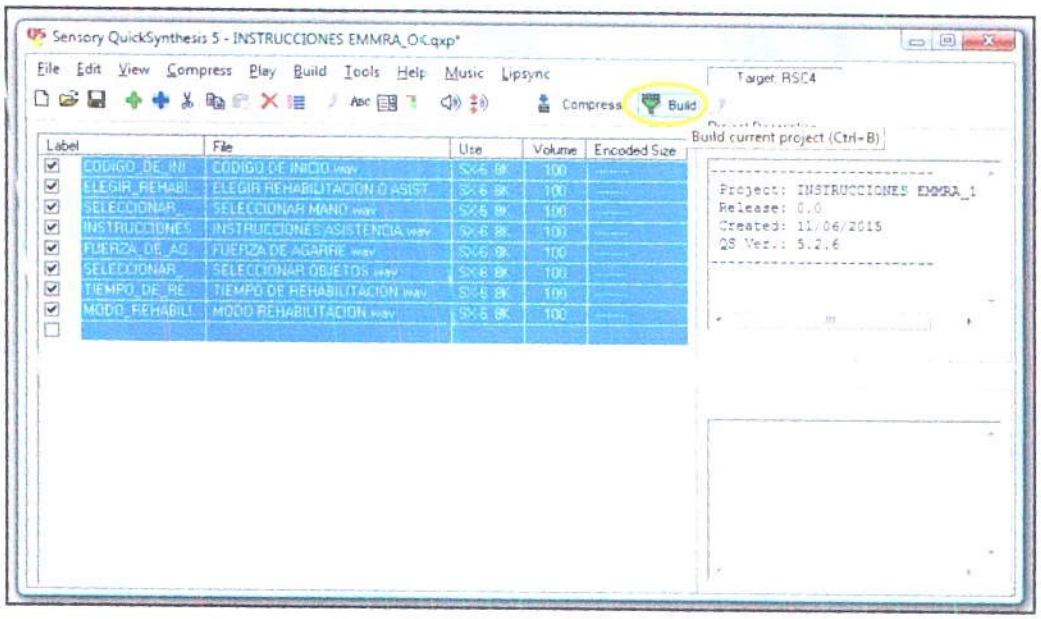


Figura 57. Crear la tabla de sonido con Built

- ✓ Seleccionamos Build Linkable Module.
- ✓ Seleccionamos Load in CONST space
- ✓ Verificamos que en Load above or at este un (0).
- ✓ Verificamos que en Load below or at este un (1ffffh)
- ✓ Seleccionamos Include Lipsync Data.
- ✓ Damos clic en **OK**

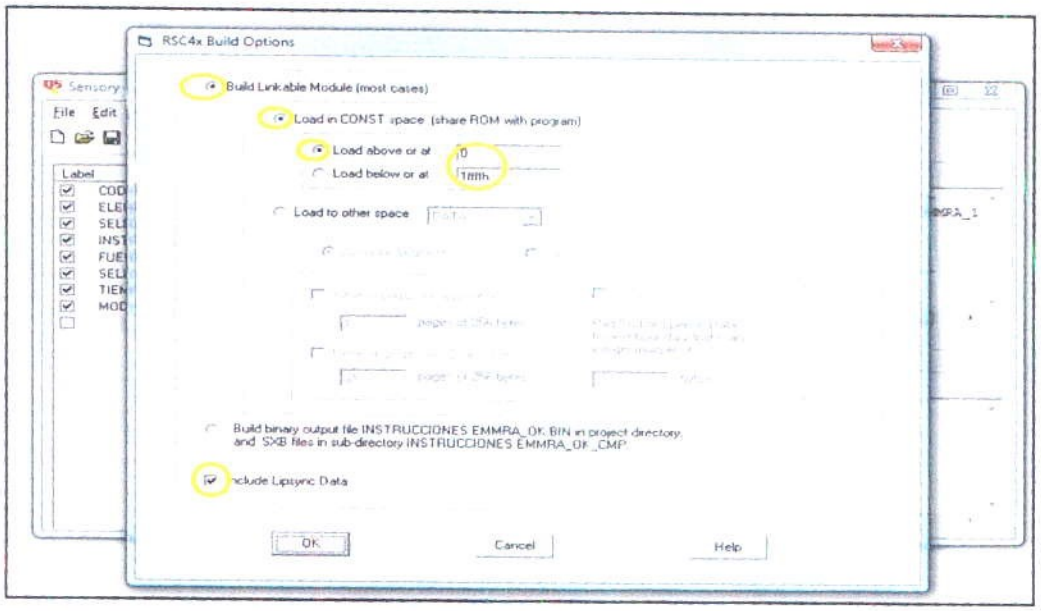


Figura 58. Especificaciones del Built

- ✓ Verificamos que la construcción de la tabla este hecha (**Done**).
- ✓ Damos clic en **OK**

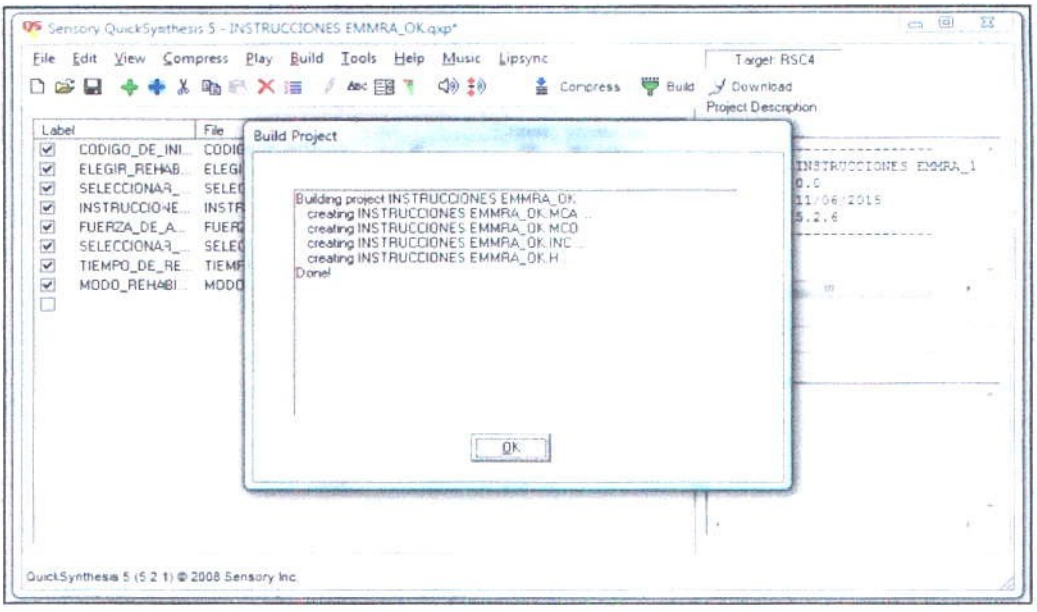


Figura 59. Finalización de la creación de la tabla de sonido

- ✓ Verificamos que se han creado los archivos que incluyen las tablas de sonido.

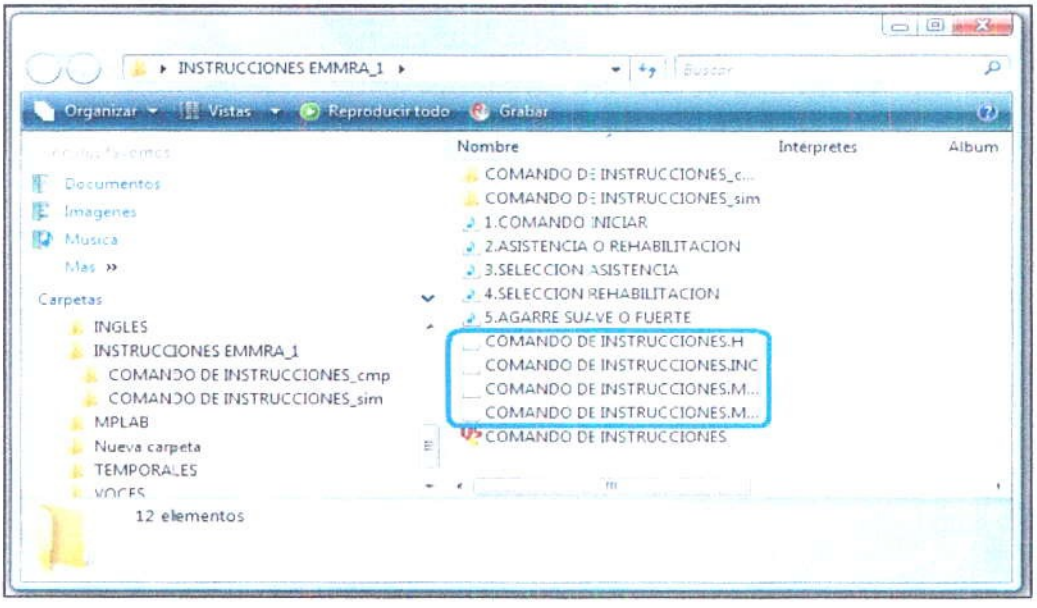


Figura 60. Archivo de la tabla de sonido

- ✓ El siguiente paso es subir la tabla de sonido al MRV, para el cual se requiere el Software EasyVR Commander. Este paso, se describirá más adelante.

8.5 Implementación del software EasyVR Commander

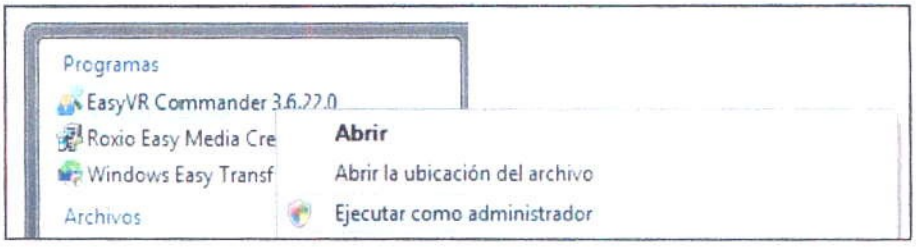


Figura 61. Inicializar el programa como administrador

Para abrir el Software EasyVR Commander debemos ejecutarlo como administrador, de esta forma evitará que se presenten errores durante la programación del MRV.

El Software EasyVR Commander es una aplicación para Windows que permite usar y configurar el módulo EasyVR de una forma sencilla e intuitiva.

La aplicación viene en diferentes, elegimos español en el ícono de elegir idioma.

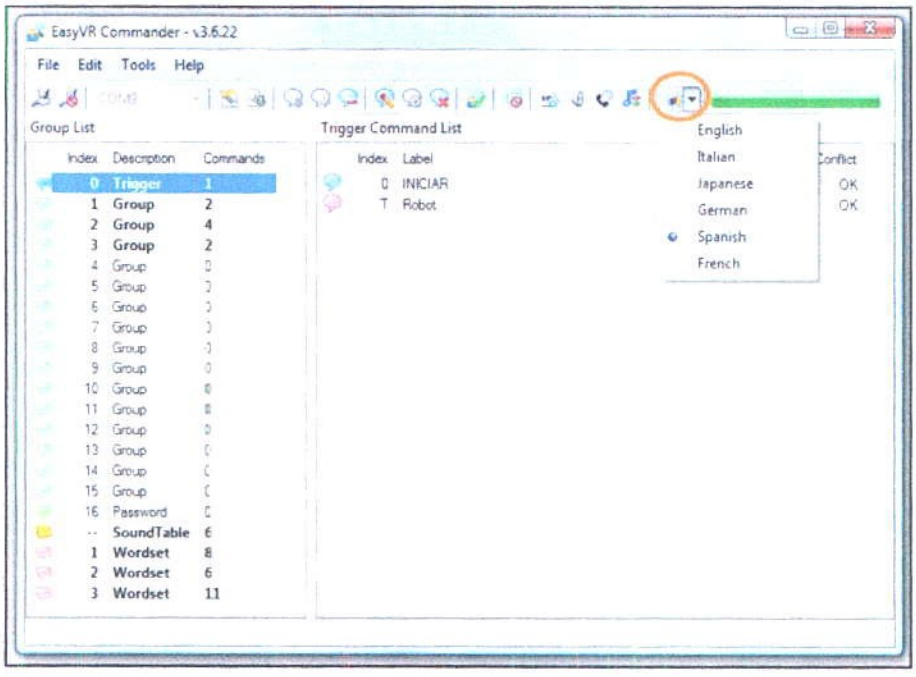


Figura 62. Esperar el tiempo de carga de la información

Al conectar el MRV al PC debe estar seleccionado el Jumper en PC. Esto permitirá que el EasyVR Commander pueda reconocer el puerto COM al que se asocia el nuevo hardware conectado al PC.

Para iniciar con el proceso de programación del MRV, debemos dar clic en el ícono de conectar del EasyVR Commander.

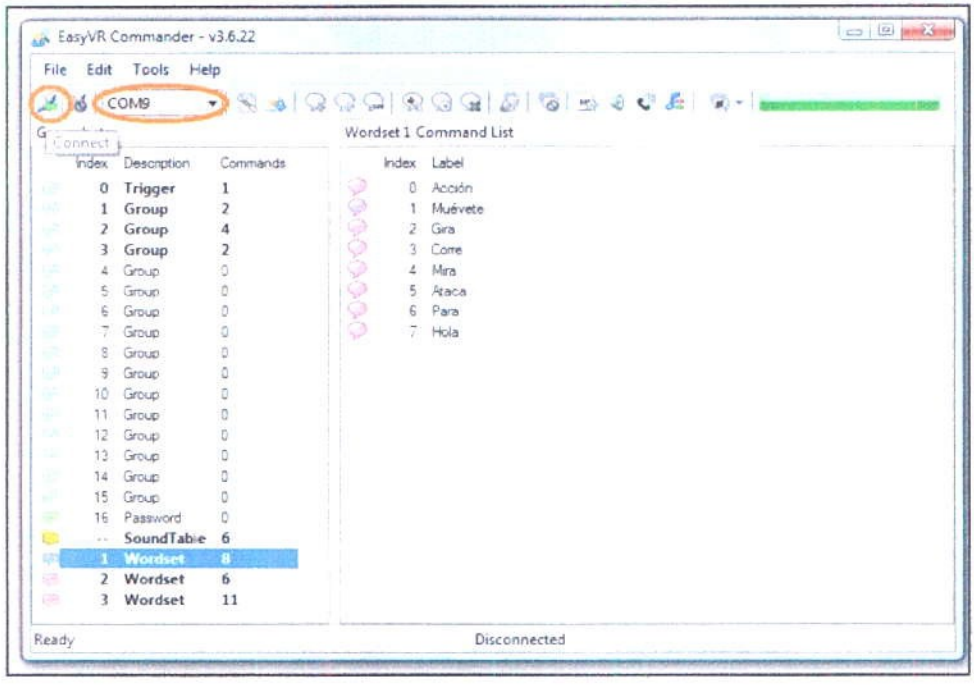


Figura 63. Verificación de archivos en el modulo de voz

Las funciones que realizaremos para el MRV corresponden a: Reconocimiento de comandos SI, Creación y Reconocimiento de comandos SD, Creación de tablas de sonido.

- **Reconocimiento de comandos SI.**
- **Creación y Reconocimiento de comandos SD**
- **Subir Tablas de Sonido al MRV**
- **Creación del Código Fuente para el MRV**

- **Reconocimiento de comandos SI.** Las voces pregrabadas SI, se encuentran al final del área de grupos:

  - 1 Wordset (8 comandos)

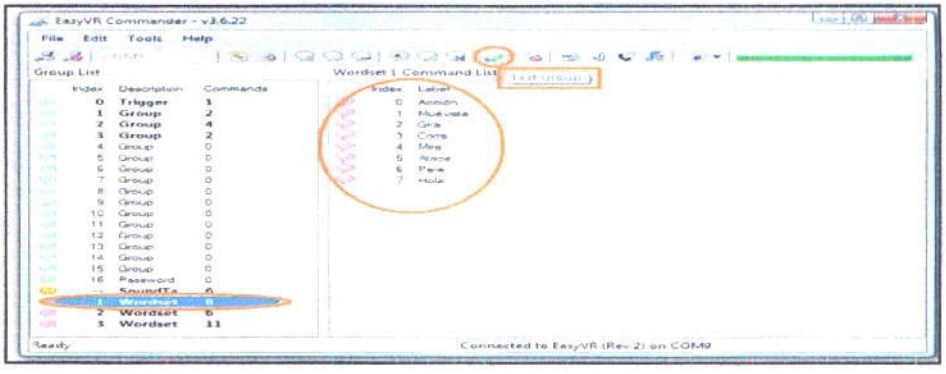


Figura 64. Identificación de wordset 1

  - 2 Wordset (6 comandos)

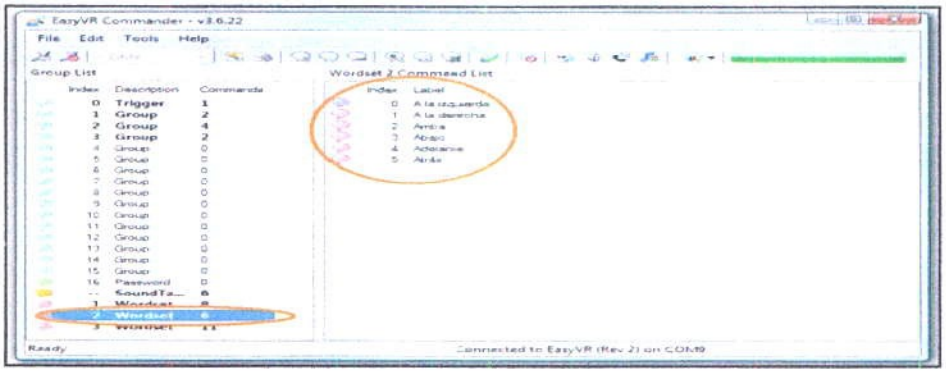


Figura 65. Identificación de wordset 2

  - 3 Wordset (11 comandos).

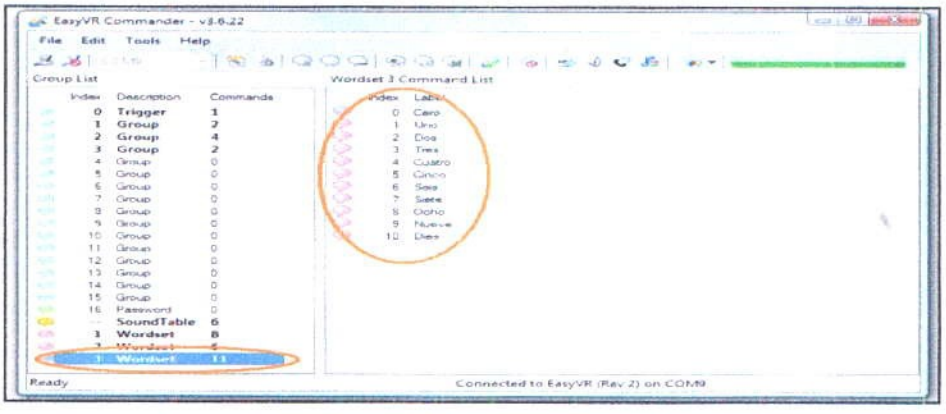


Figura 66. Identificación de wordset 3

Para validar el reconocimiento de voces SD, realizamos los siguientes pasos:

- ✓ Seleccionamos uno de los grupos Wordset.
- ✓ Damos clic en el ícono Test Group.

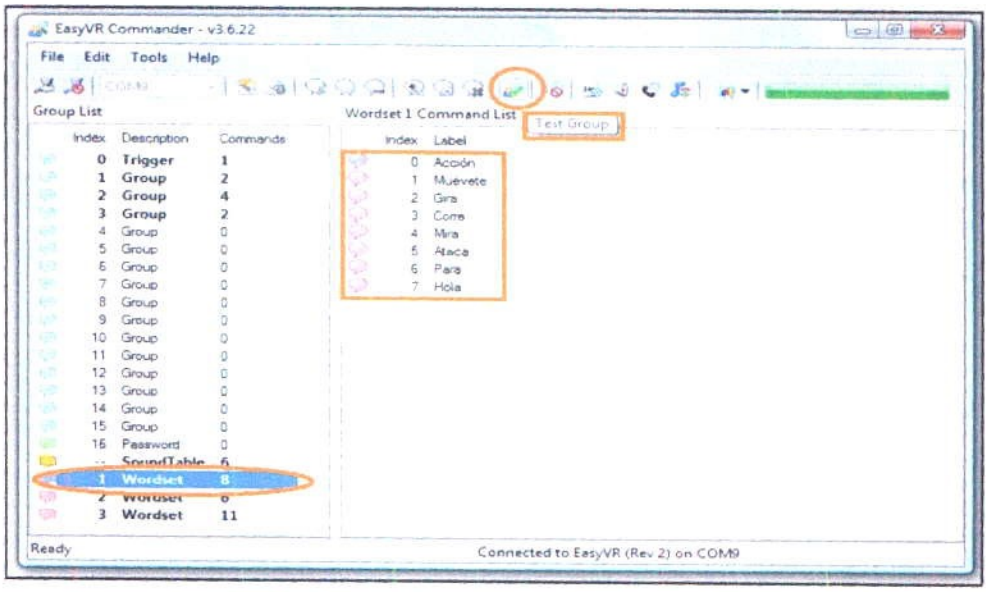


Figura 67. Reconocimiento de sonidos SI

- ✓ Pronunciamos uno de los comandos predefinidos SI.
- ✓ Cuando el MRV reconoce el comando, este queda sombreado de color verde.

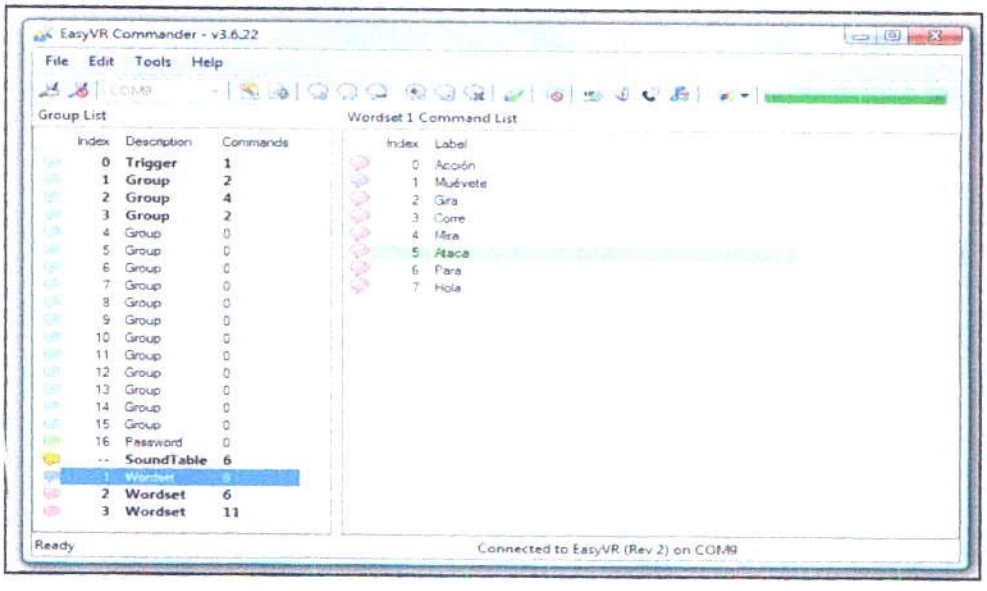


Figura 68. Reconocimiento de sonidos SI

- **Creación y reconocimiento de comandos SD**. Al conectar el EasyVR Coomander con una tarjeta de reconocimiento de voz nueva. Crearemos los grupos y comandos de una forma fácil e intuitiva.

  - ✓ Seleccionamos el Grupo **Trigger**.
  - ✓ Damos clic en el icono de **Insertar o Adicionar Comando**
  - ✓ Digitamos el comando que deseamos.

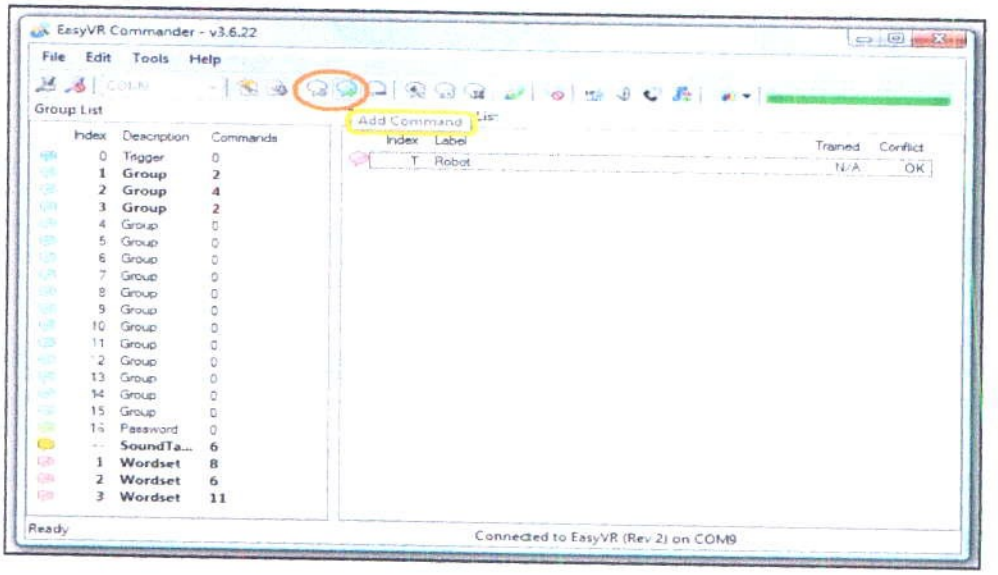


Figura 69. Creación de comandos de voz

  - ✓ Nos aparecerá un recuadro en el que debemos digitar el comando a crear.
  - ✓ Creamos el comando **INICIAR**

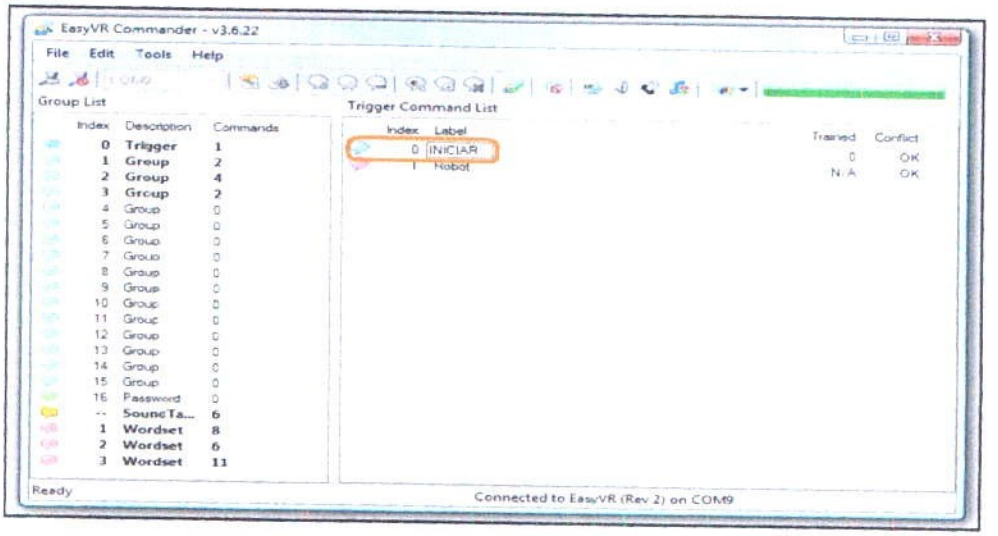


Figura 70. Almacenar el comando de voz

- ✓ Damos clic en **Train Command**, para asignar una palabra o frase al comando que estamos creando.

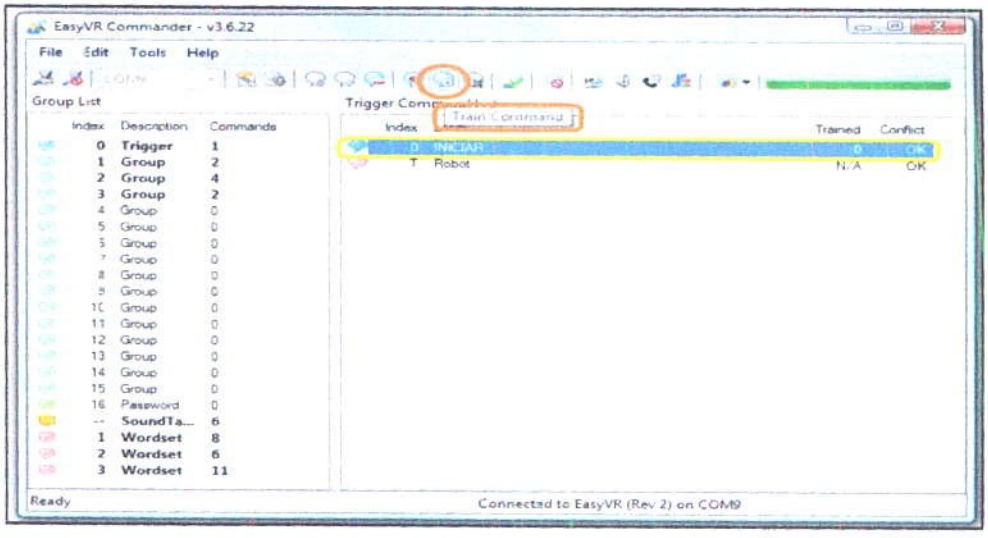


Figura 71. Grabar sinido en el comando de voz

- ✓ Aparecerá una ventana informando que solo tenemos **5 segundos** para pronunciar la palabra "**INICIAR**"
- ✓ Damos clic en **Phase 1**.
- ✓ Pronunciamos la palabra o frase que deseamos esté asociada al comando.
- ✓ Aparecerá una ventana informando que solo tenemos **5 segundos** para pronunciar la palabra "**INICIAR**"
- ✓ Damos clic en **Phase 2**.
- ✓ Pronunciamos nuevamente la palabra o frase que deseamos esté asociada al comando.

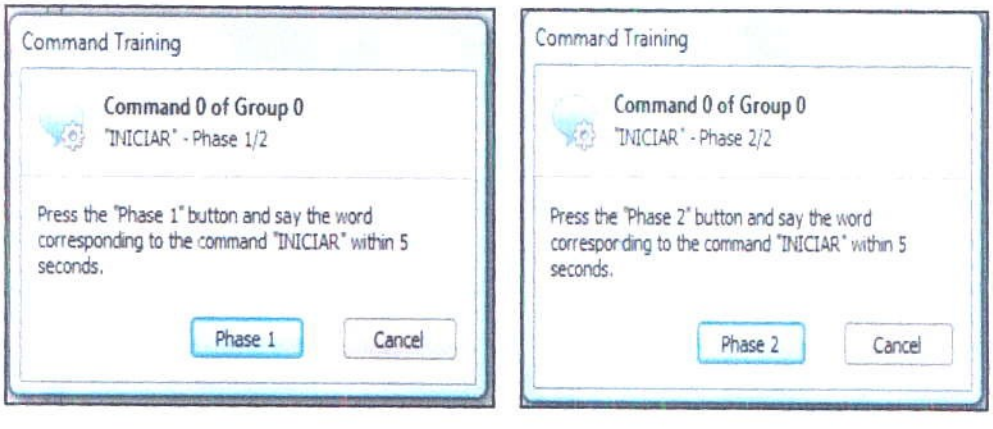


Figura 72. Grabar y confirnar sonido

- ✓ Si aparece algún error durante la pronunciación, debemos repetir la Phase 1 y la Phase 2.
- ✓ Si deseamos eliminar el comando que hemos creado, debemos seleccionar el comando y hacer clic en la opción **Remove Command.**
- ✓ Realizamos los pasos anteriores para crear comandos en los demás grupos.

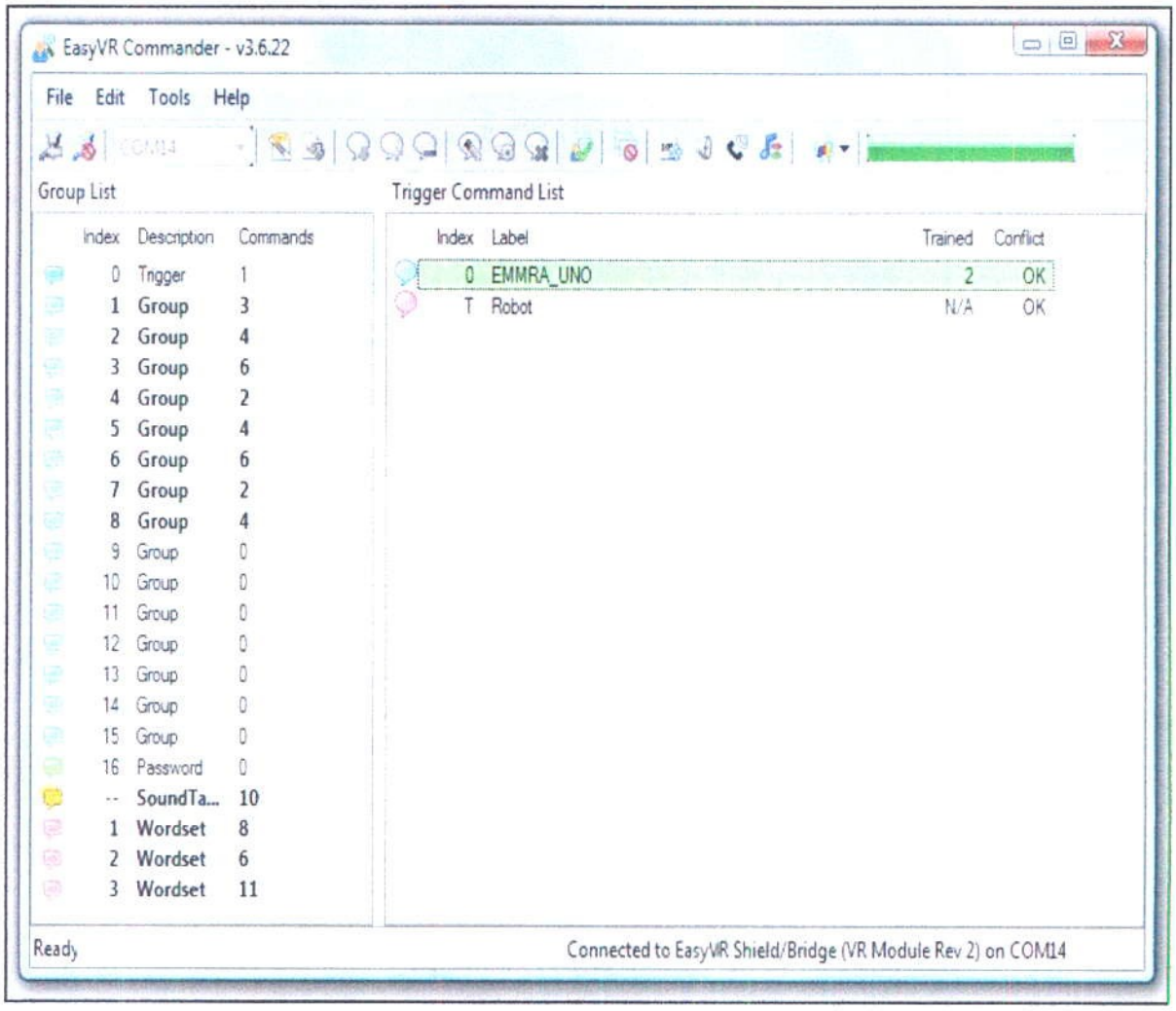


Figura 73. Creación de comandos en el grupo 0

- ✓ **En el Grupo 0, creamos el comando EMMRA_UNO para inicializar el programa de reconocimiento de voz**

- ✓ **En el Grupo 1, creamos los comandos para que el usuario seleccione entre Programa de Asistencia, Programa de Rehabilitación o Programa para el giro de la Silla.**

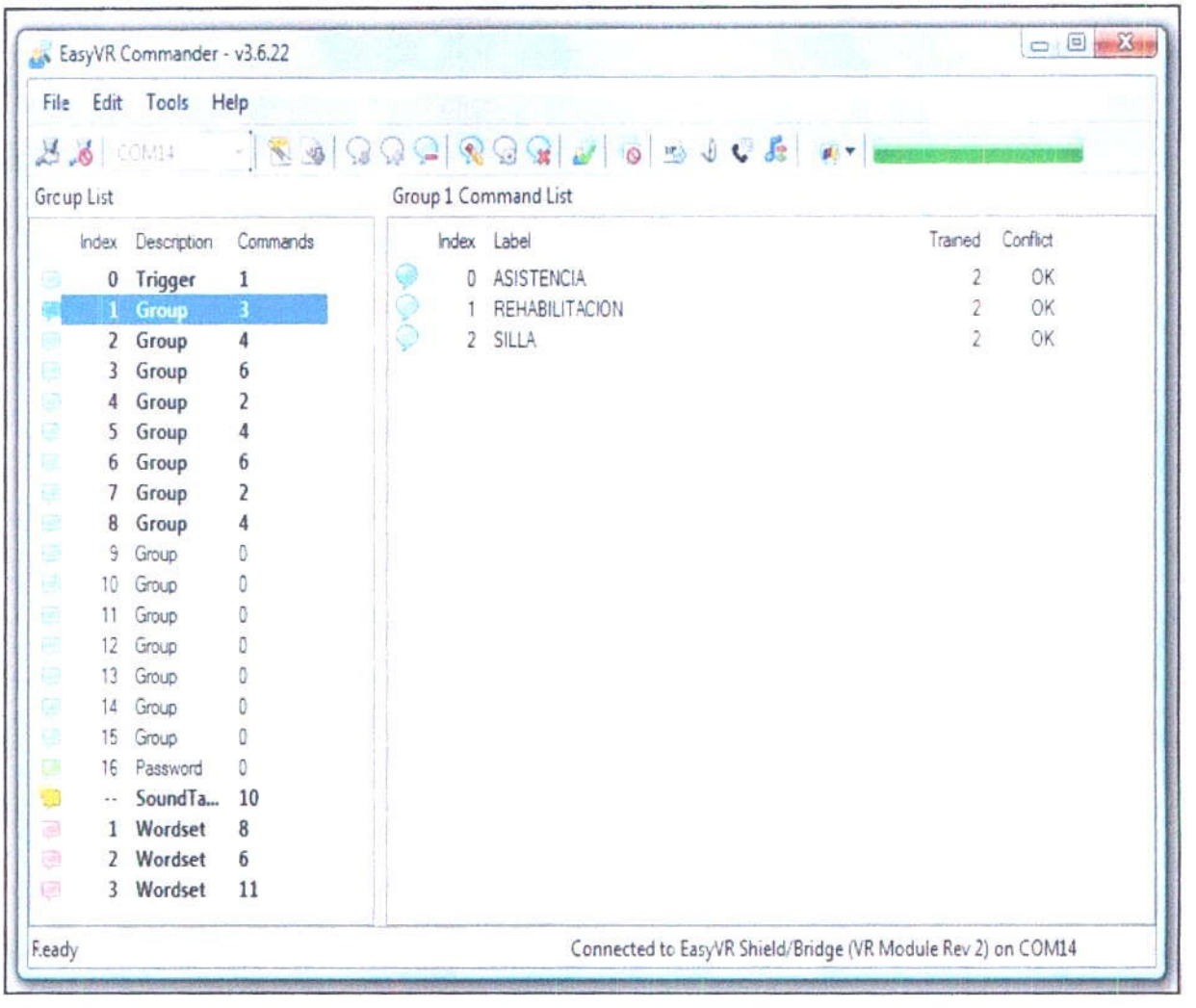


Figura 74. Creación de comandos en el grupo 1

- ✓ **En el Grupo 2, creamos los comandos para que el usuario seleccione entre Mano Derecha, Mano Izquierda, Ambas Manos o Reiniciar.**

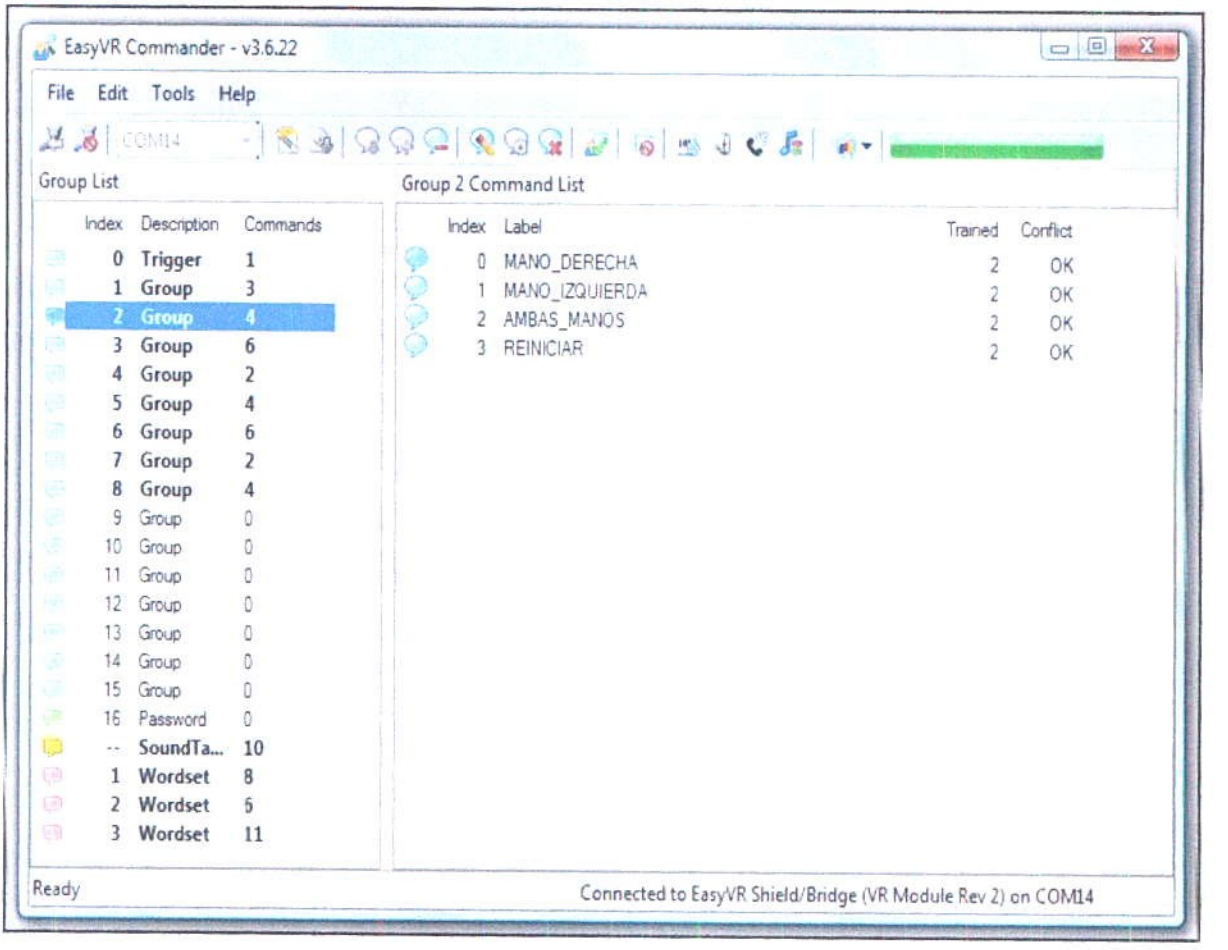


Figura 75. Creación de comandos en el grupo 2

Al seleccionar el comando Reiniciar, el programa se reinicia desde el Grupo 0.

✓ **En el Grupo 3, creamos los comandos para que el usuario seleccione el tipo de Asistencia que requiere.**

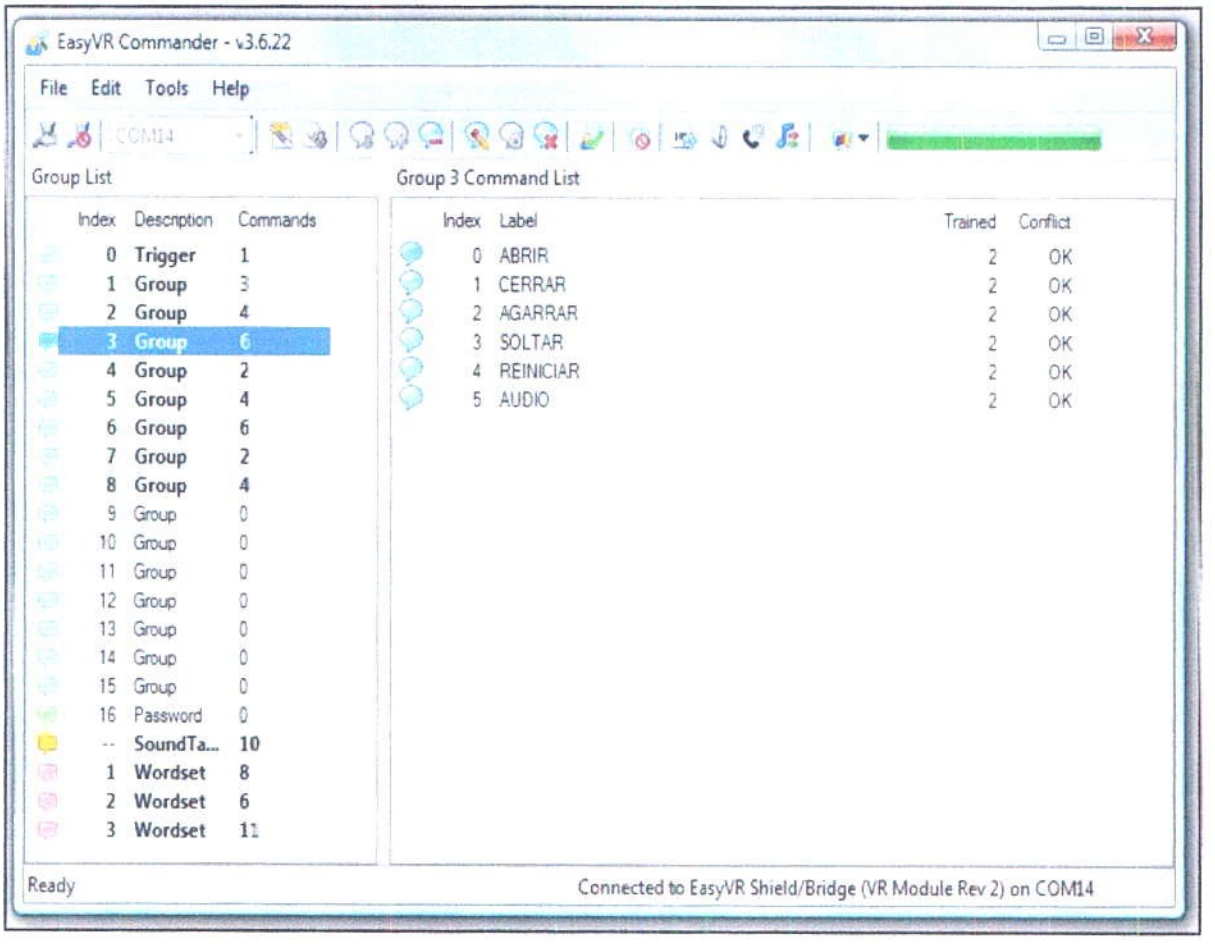


Figura 76. Creación de comandos en el grupo 3

Los comandos de asistencia son: Abrir, Cerrar, Agarrar, Soltar, Reiniciar y Audio.

Al pronunciar el comando Reiniciar, el programa regresará al grupo 2.

Al pronunciar el comando Audio, el programa reproducira las opciones de Abrir, Cerrrar, Agarrar o Soltar objetos.

- ✓ **En el Grupo 4, creamos los comandos para que el usuario seleccione entre Agarrar Fuerte o Suave algún objeto.**

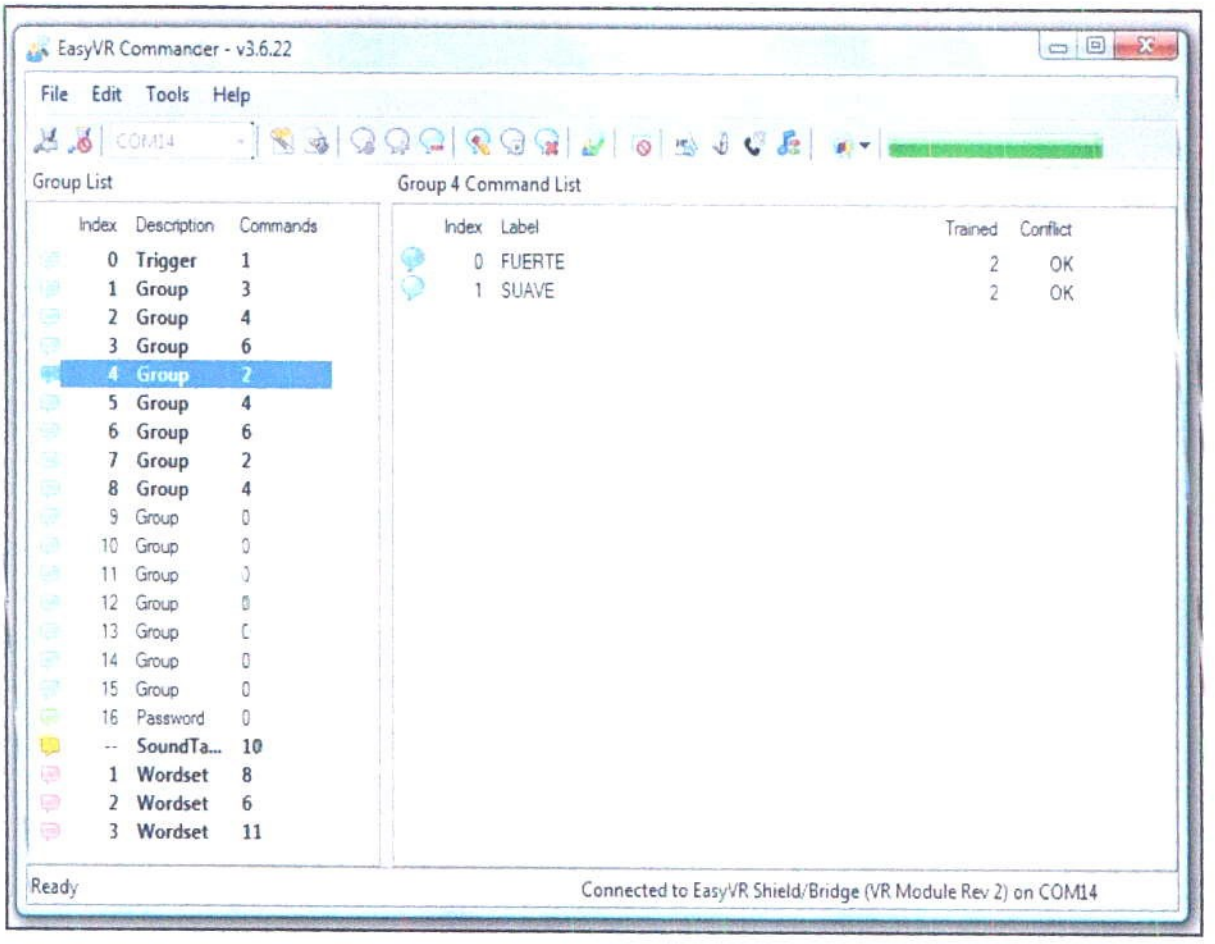


Figura 77. Creación de comandos en el grupo 4

- ✓ **En el Grupo 5, creamos los comandos para que el usuario seleccione el tiempo de rehabilitación entre CINCO, DIEZ o QUINCE minutos.**

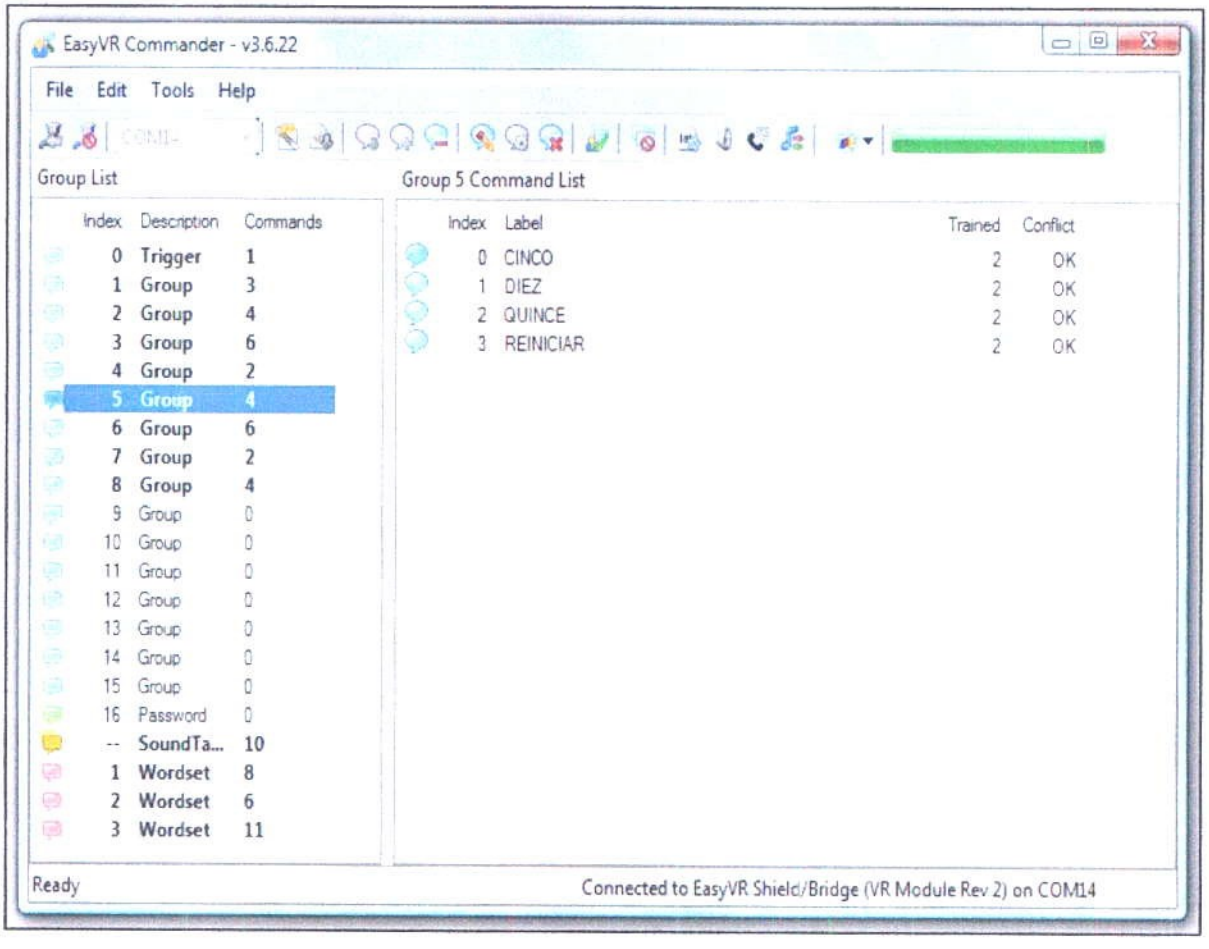


Figura 78. Creación de comandos en el grupo 5

Al seleccionar el tiempo de rehabilitación, el programa no se detendrá hasta que finalice el tiempo seleccionado.

- ✓ **En el Grupo 6, creamos los comandos para que el usuario seleccione el objeto que desea agarrar.**

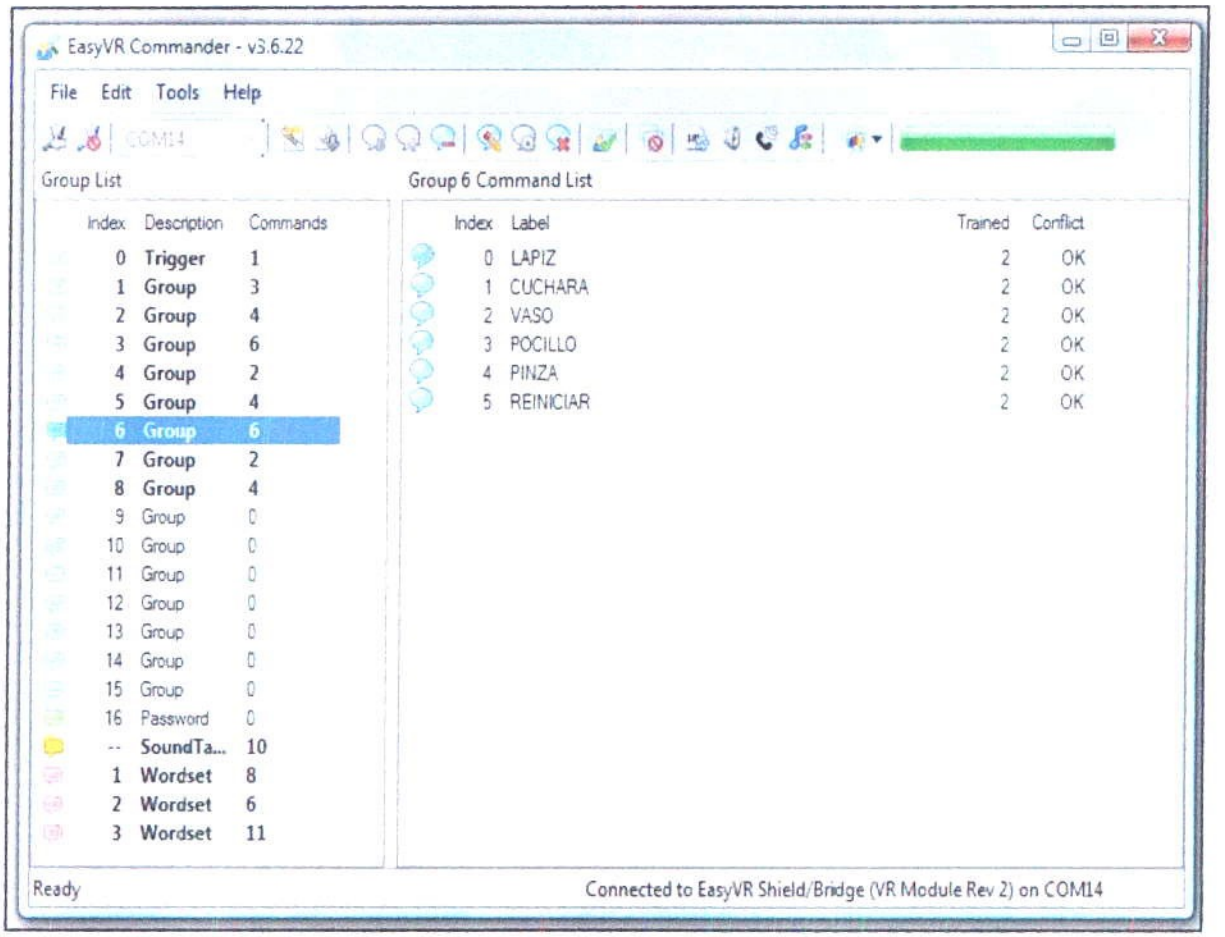


Figura 79. Creación de comandos en el grupo 6

El usuario puede seleccionar en agarrar un LAPIZ, VASO, CUCHARA, POCILLO, PINZA o RE'NICIAR.

Al pronunciar el comando reiniciar, el programa lo llevará nuevamente al grupo 2.

- ✓ **En el Grupo 7, creamos los comandos para que el usuario seleccione entre Rehabilitación Pasiva o Activa.**

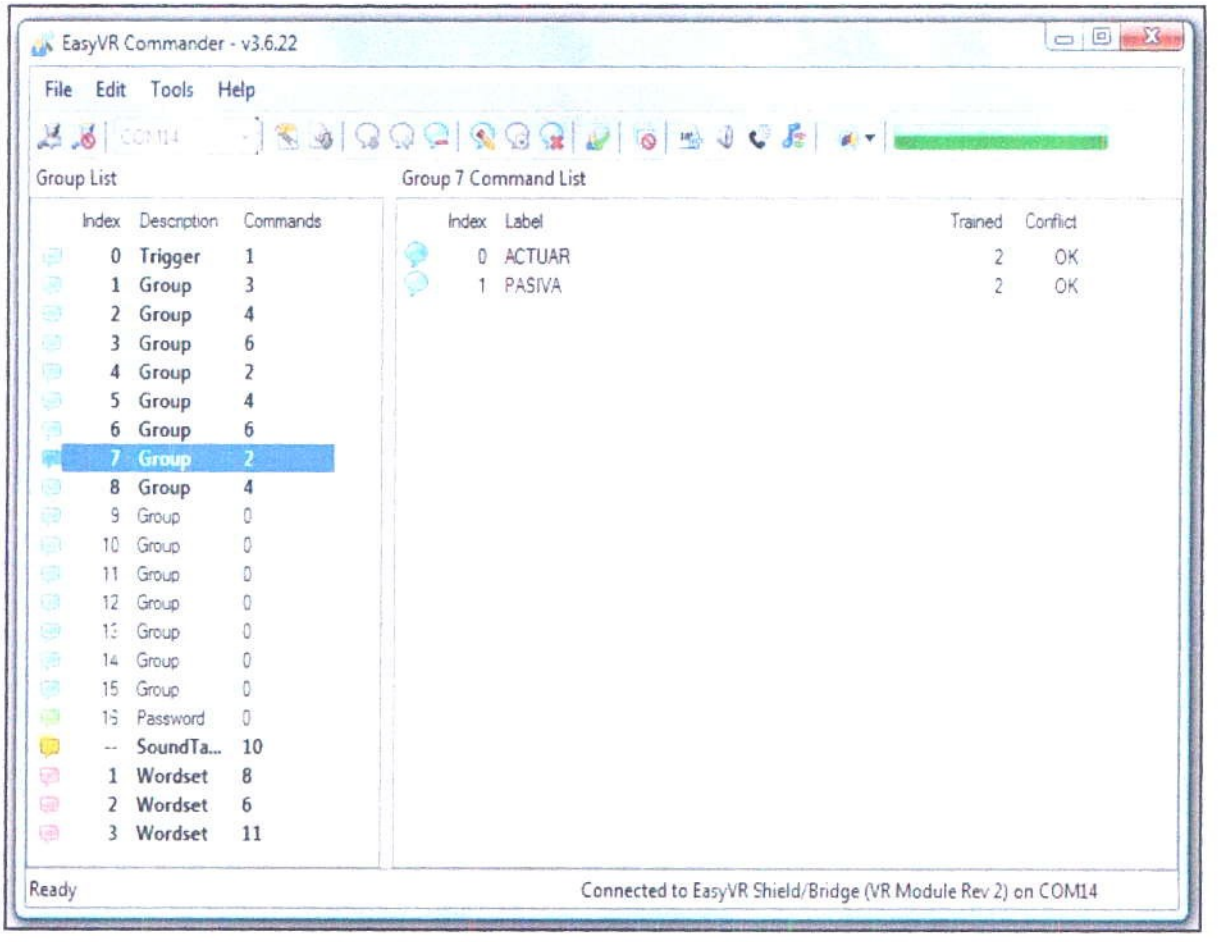


Figura 80. Creación de comandos en el grupo 7

Al seleccionar el comando Pasiva el Prototipo EMMRA-1 realizará movimientos de abrir y cerrar la mano, y flexionar o extender la muñeca automáticamente, durante un tiempo determinado entre (cinco, diez y quince minutos)

Al seleccionar el comando Actuar, el Prototipo EMMRA-1 realizará movimientos de abrir y cerrar la mano, y flexionar o extender la muñeca, solo cuando el sensor de electromiografía detecte un cambio en la tensión muscular del área sensada

- ✓ **En el Grupo 8, creamos los comandos para que el usuario seleccione entre Girar la Silla a la derecha o izquierda, también para detener la Silla o para continuar la ejecución del programa.**

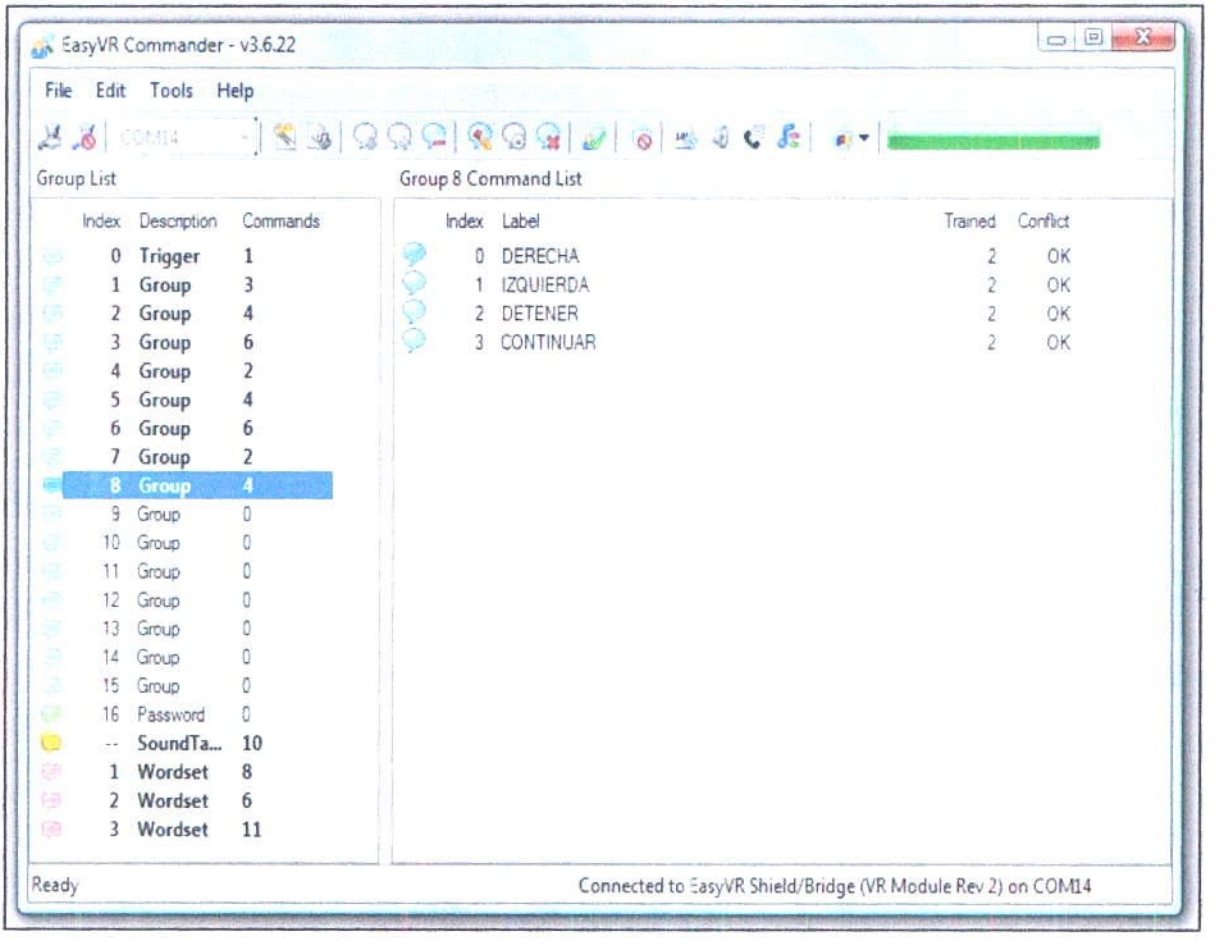


Figura 81. Creación de comandos en el grupo 8

Al seleccionar el comando Detener, la silla detendra el movimiento.

Al seleccionar el comando Continuar, el programa se reinicializará en el grupo 1

- ✓ Para eliminar todos los comandos creados, podemos hacer clic en el ícono **Reset All**.

8.6 Subir Tablas de sonido al MRV.

La board EasyVR Shield 3.0, permite también el ingreso de tablas de sonido construidas previamente con el Software QuickSynthesis 5.0 de Sensory.

A continuación describiremos con detalle el proceso de subir tablas de sonido, las cuales corresponcen a las instrucciones que se presentarán al usuario final del Prototipo EMMRA-1, para facilitar la operatividad del dispositivo.

- ✓ El Software EasyVR Commander está en modo ejecutar como administrador.

- ✓ Damos clic al ícono **Disconnect**

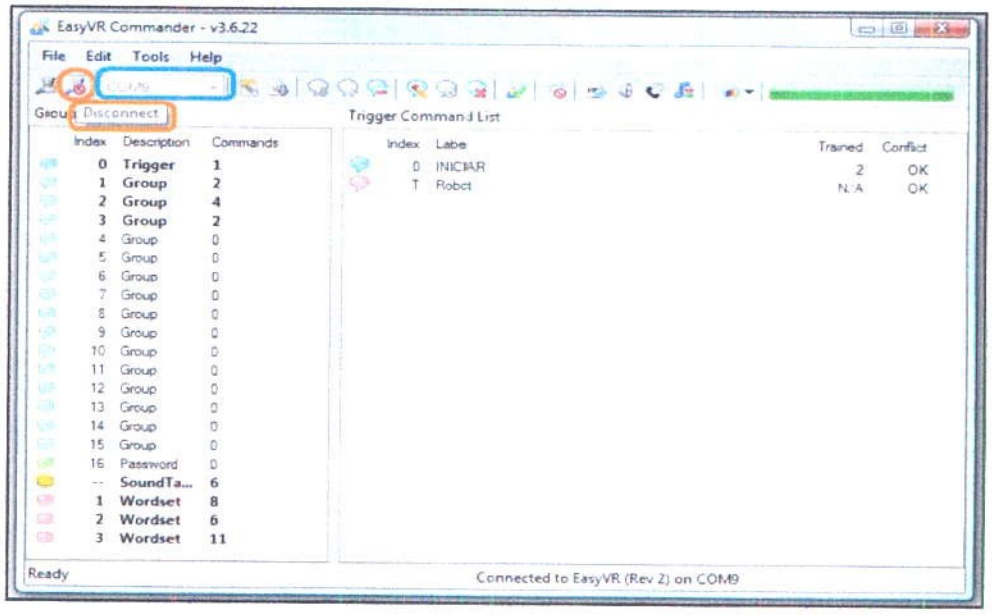


Figura 82. Desconexión de la Shield

- ✓ Desconectamos el cable USB del MRV al PC.
- ✓ Situamos el Jumper del EasyVR Shield 3.0 en posición **UP**.
- ✓ Reconectamos el cable USB del MRV al PC.
- ✓ Esperamos a que el EasyVR Commander reconozca el puerto COM que el PC asignó al nuevo hardware.
- ✓ Se habilita la el ícono **Update Custom Data**

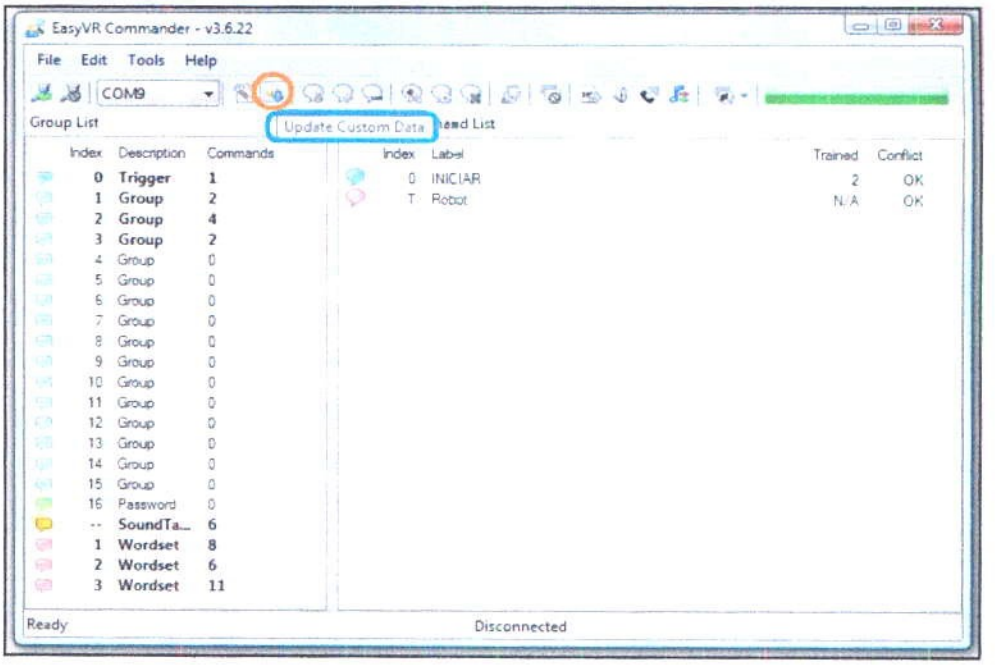


Figura 83. Selección Update Custom Data

- ✓ Damos clic en Update Custom Data.
- ✓ Seleccionamos Import en la ventana de diálogo, para subir el archivo de la tabla de sonido.

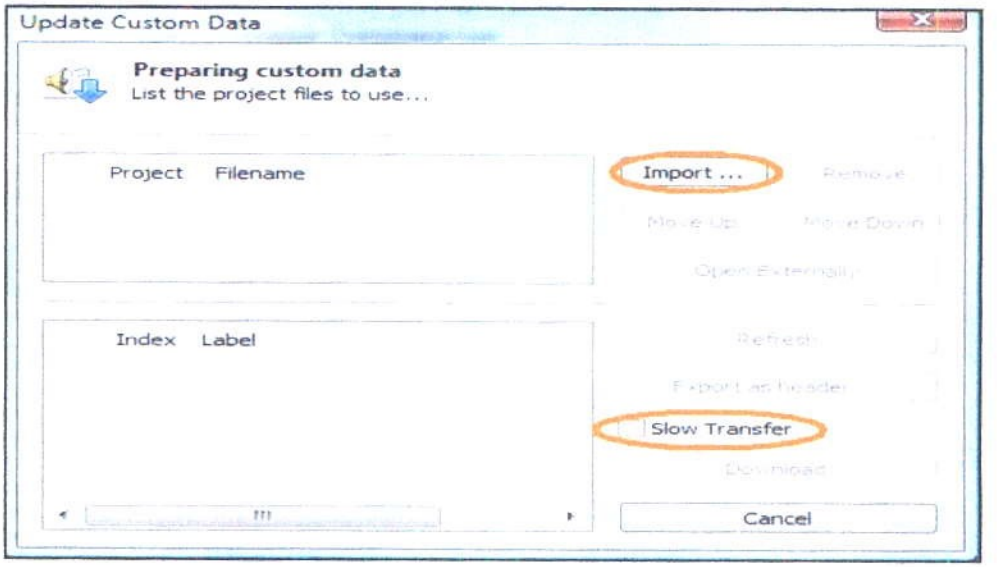


Figura 84. Selección modo lento para importar tablas de sonido

- ✓ Seleccionamos el archivo de nuestra tabla de sonidos creada en QuickSynthesis 5.0.
- ✓ Damos clic en Abrir.

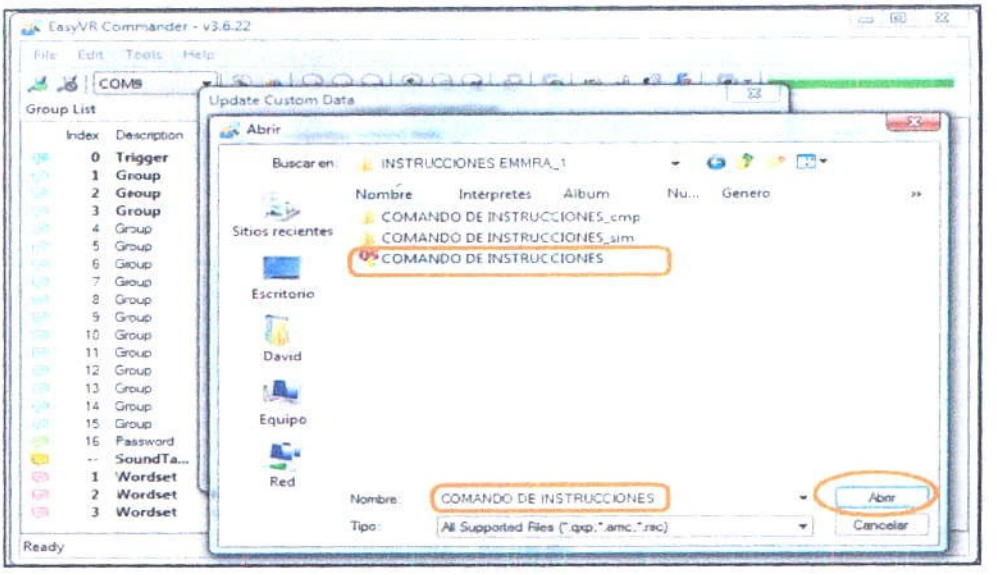


Figura 85. Selección de la tabla de sonidos a cargar

- ✓ Esperamos a que se importen todos los archivos.
- ✓ Seleccionamos **Slow Transfer**
- ✓ Damos clic en **Download**
- ✓ Esperamos a que se suban los archivos a la EasyVR Shield 2.0.

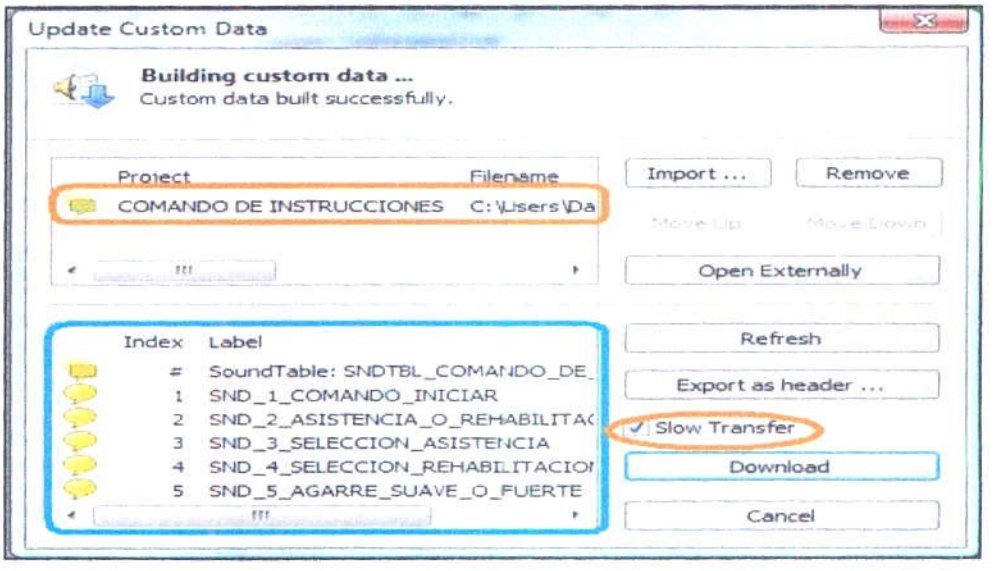


Figura 86. Transferencia de datos

✓ Desconectamos nuevamente el MRV del EasyVR Commander.

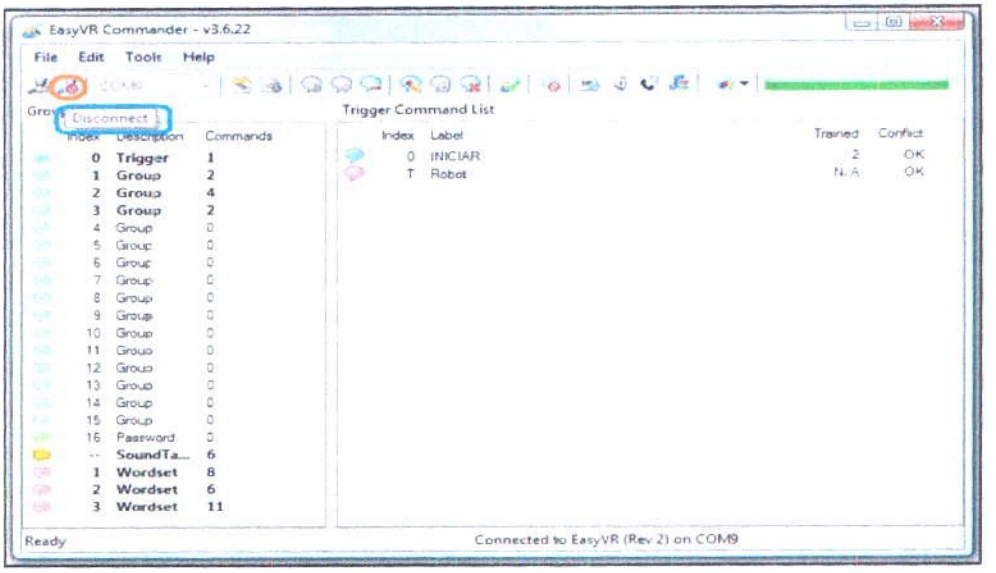


Figura 87. Desconectamos la Shield

✓ Desconectamos el cable USB del MRV al PC.
✓ Colocamos el Jumper del EasyVR Shield 2.0 en posición **PC**.
✓ Conectamos nuevamente el cable USB del MRV al PC.
✓ Esperamos a que el EasyVR Commander reconozca el puerto COM.
✓ Damos clic en **Conectar** del EasyVR Commander.

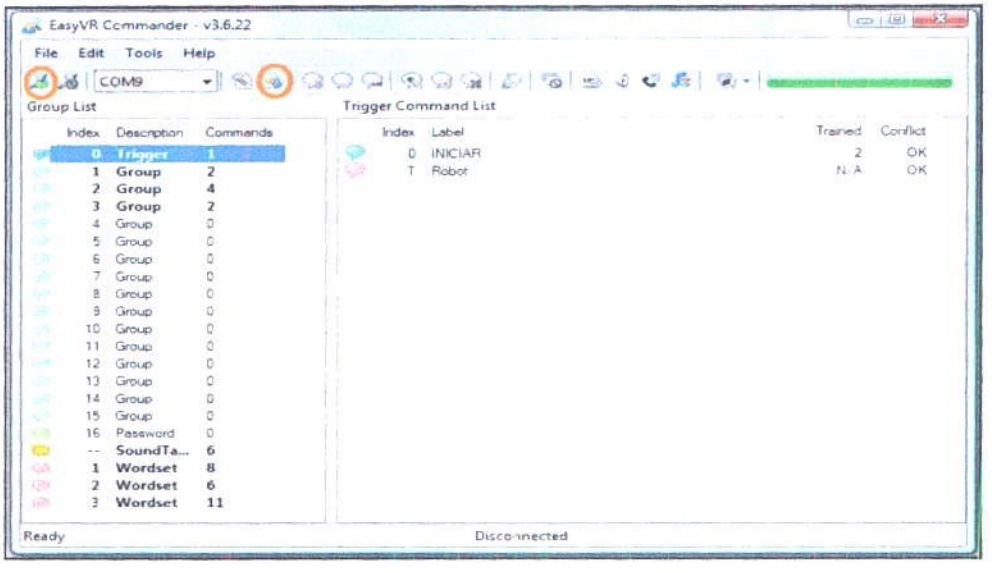


Figura 88. Conexión de la Shield

- ✓ Observamos nuestra tabla de sonidos insertada en el grupo Sound Table del MRV.
- ✓ Conectamos un amplificador de 8Ω al SPEAKER del MRV
- ✓ Seleccionamos un sonido.
- ✓ Damos clic en el icono Play Sound
- ✓ Escuchamos el comando o instrucción que hemos ingresado.

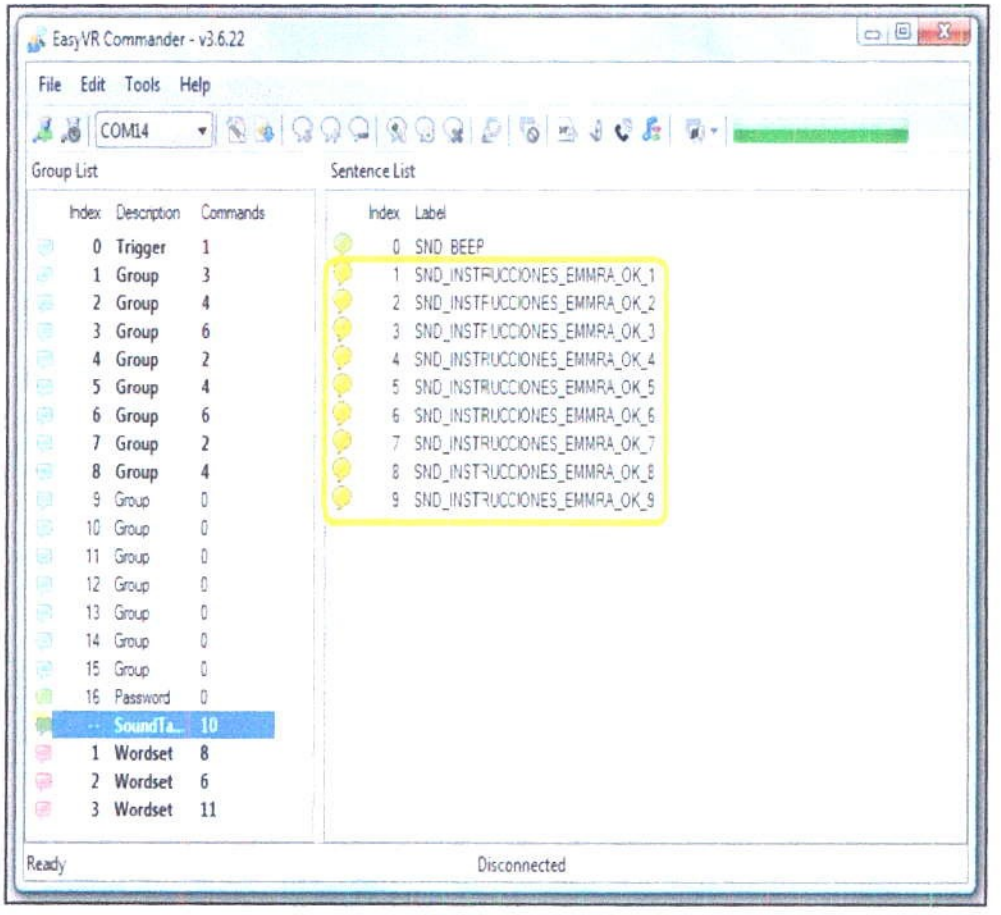


Figura 89. Verificación de los sonidos cargados

8.7 Creación del Código Fuente para el MRV

✓ Damos clic en el ícono **Generate Code**

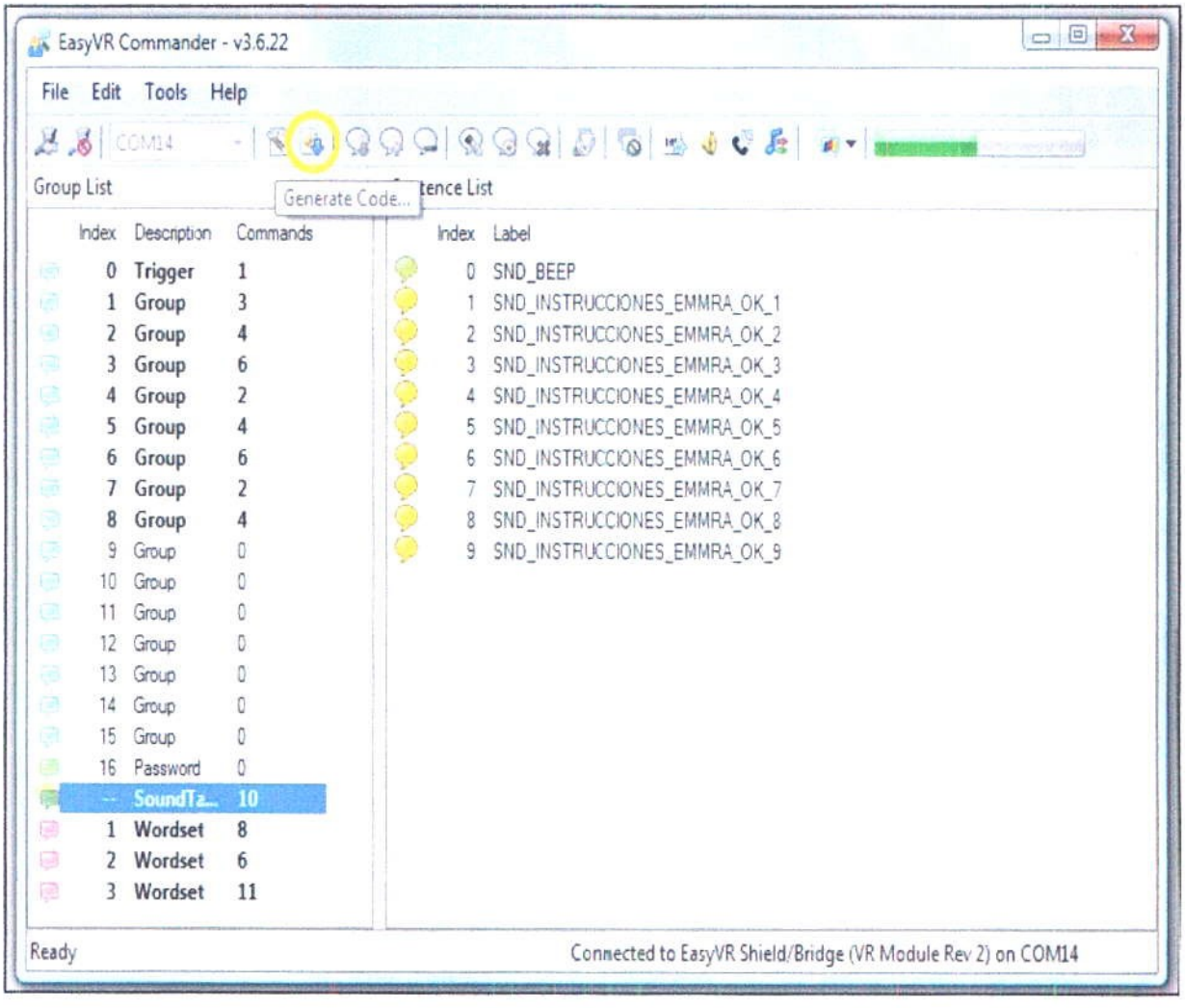


Figura 90. Generar código de programación

- ✓ Guardamos el código en una carpeta previamente creada en nuestro PC llamada "**CODIGO ARDUINO EMMRA_1**"
- ✓ Nombramos nuestro archivo como "**CODIGO MODULO DE VOZ EMMRA_1**"

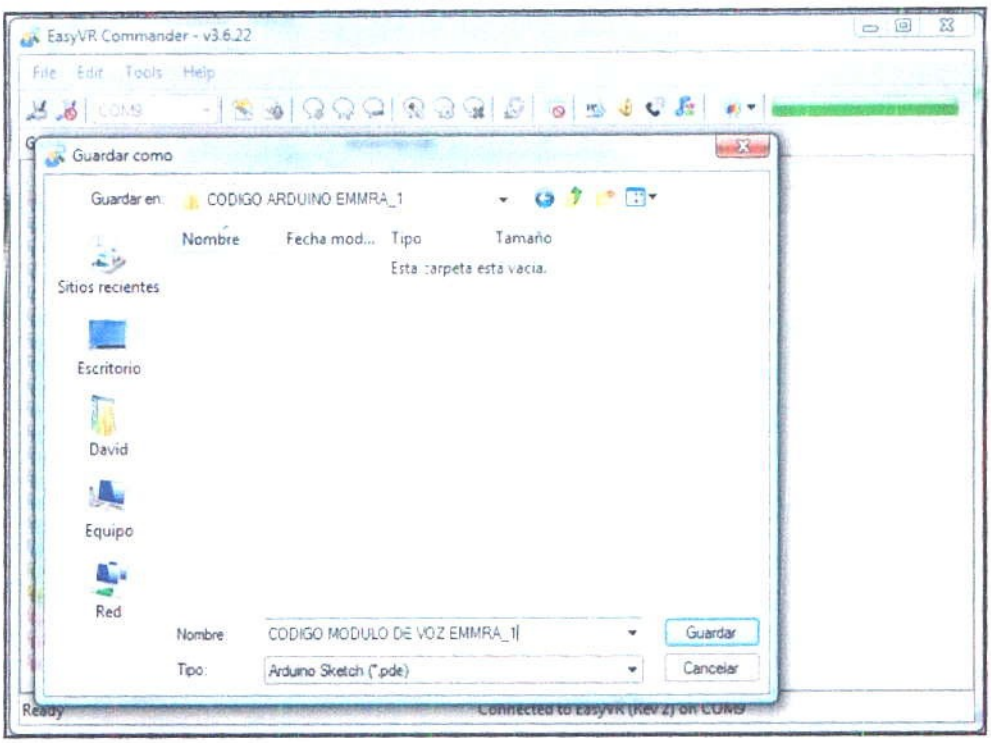


Figura 91. Control pin /XM. Tomado de Intern.et

- ✓ Nuestro archivo queda guardado en la carpeta seleccionada.

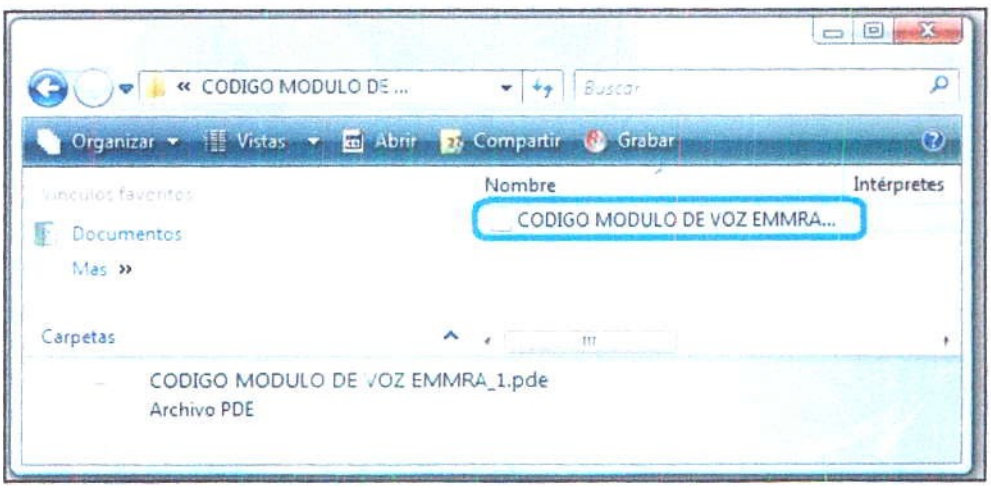


Figura 92. Control pin /XM. Tomado de Intern.et

- ✓ Abrimos el Software de Arduino.
- ✓ Damos clic en Archivo/Abrir/CODIGO MODULO DE VOZ EMMRA_1
- ✓ Observamos el código fuente generado en el Software Arduino.

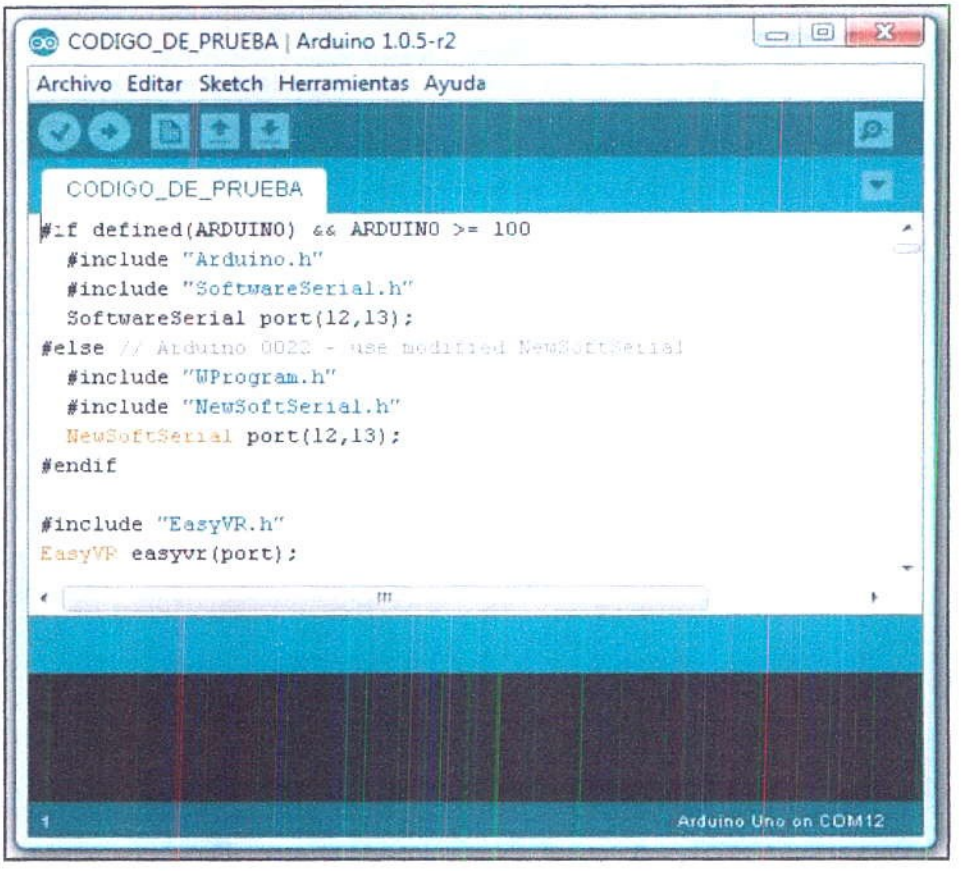


Figura 93. Librerías del código fuente

- ✓ La primera parte del código fuente contiene las librerías:

"Arduino.h".
"SoftwareSerual.h".
"Wprogram.h".
"NewSoftwareSerial.h".
"EasyVR.h".

- ✓ La segunda parte del código fuente contiene el listado de los grupos creados en el módulo de voz.

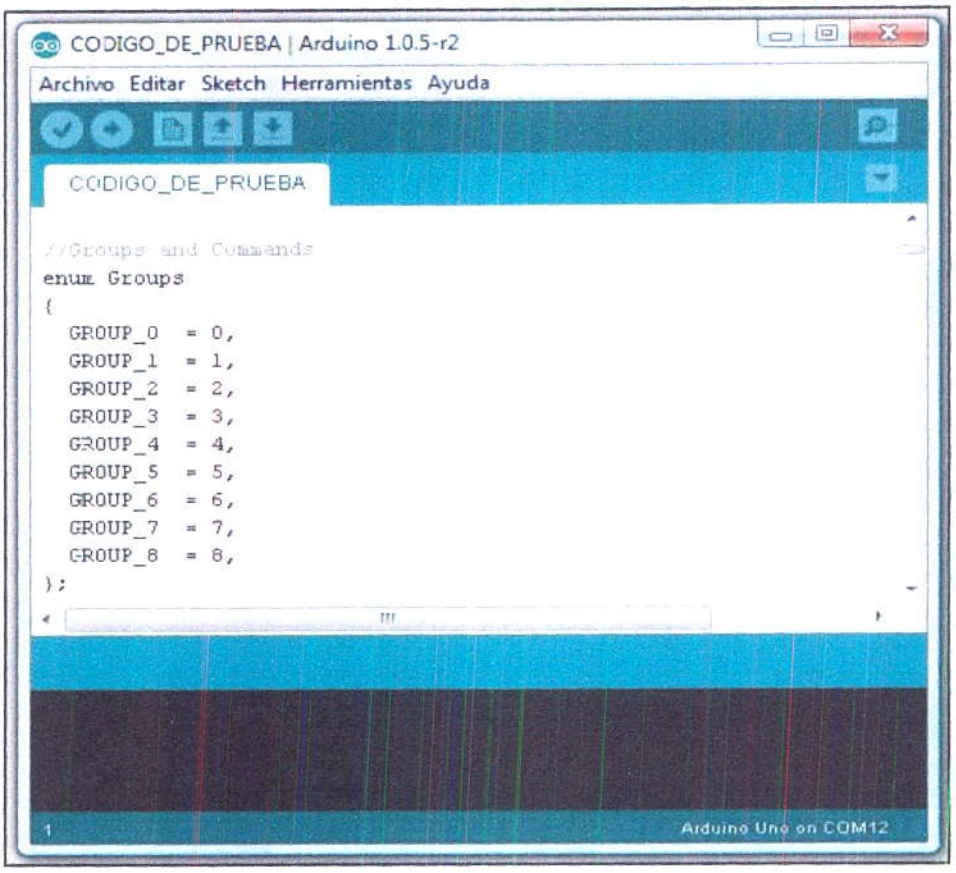


Figura 94. Listado de Grupos

- ✓ La tercera parte del código fuente contiene los comandos de voz creados en el modulo de voz.

- ✓ La cuarta parte del programa del codigo fuente contiene el void setup para la comunicación entre el Arduino Mega y el Easyvr.

- ✓ La quinta parte contiene el Void Loop para indicar la secuencia interna del programa.

- ✓ La sexta parte contiene el Main Void (programa principal) para ejecutar los ciclos de instrucciones que el programador desee.

✓ A continuación mostraremos todo el contenido del Código Fuente generado desde el EasyVR Commander:

```
// PRIMERA PARTE (LIBRERIAS)
// ************************************************************************
#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"
#include "SoftwareSerial.h"
SoftwareSerial port(12,13);
#else // Arduino 0022 - use modified NewSoftSerial
#include "WProgram.h"
#include "NewSoftSerial.h"
NewSoftSerial port(12,13);
#endif
#include "EasyVR.h"
EasyVR easyvr(port);
// ************************************************************************

// SEGUNDA PARTE (GRUPOS  DEL MODULO DE VOZ)
// ************************************************************************
//Groups and Commands
enum Groups
{
  GROUP_0  = 0,
  GROUP_1  = 1,
  GROUP_2  = 2,
  GROUP_3  = 3,
  GROUP_4  = 4,
  GROUP_5  = 5,
  GROUP_6  = 6,
  GROUP_7  = 7,
  GROUP_8  = 8,
};
// ************************************************************************

// TERCERA PARTE (COMANDOS MODULO DE VOZ)
// ************************************************************************
enum Group0
{
  G0_EMMRA_UNO = 0,
};
```

```
enum Group1
{
 G1_ASISTENCIA = 0,
 G1_REHABILITACION = 1,
 G1_SILLA = 2,
};
enum Group2
{
 G2_MANO_DERECHA = 0,
 G2_MANO_IZQUIERDA = 1,
 G2_AMBAS_MANOS = 2,
 G2_REINICIAR = 3,
};
enum Group3
{
 G3_ABRIR = 0,
 G3_CERRAR = 1,
 G3_AGARRAR = 2,
 G3_SOLTAR = 3,
 G3_REINICIAR = 4,
 G3_AUDIO = 5,
};
enum Group4
{
 G4_FUERTE = 0,
 G4_SUAVE = 1,
};
enum Group5
{
 G5_CINCO = 0,
 G5_DIEZ = 1,
 G5_QUINCE = 2,
 G5_REINICIAR = 3,
};

enum Group6
{
 G6_LAPIZ = 0,
 G6_CUCHARA = 1,
 G6_VASO = 2,
 G6_POCILLO = 3,
```

```
  G6_PINZA = 4,
  G6_REINICIAR = 5,
};
enum Group7
{
  G7_ACTUAR = 0,
  G7_PASIVA = 1,
};
enum Group8
{
  G8_DERECHA = 0,
  G8_IZQUIERDA = 1,
  G8_DETENER = 2,
  G8_CONTINUAR = 3,
};
// ************************************************************************

// CUARTA PARTE (VOID SETUP)
// ************************************************************************
EasyVRBridge bridge;
int8_t group, idx;
void setup()
{
  // bridge mode?
  if (bridge.check())
  {
    cli();
    bridge.loop(0, 1, 12, 13);
  }
  // run normally
  Serial.begin(9600);
  port.begin(9600);
  if (!easyvr.detect())
  {
    Serial.println("EasyVR not detected!");
    for (;;);
  }
  easyvr.setPinOutput(EasyVR::IO1, LOW);
  Serial.println("EasyVR detected!");
  easyvr.setTimeout(5);
  easyvr.setLanguage(4);
```

```
  group = EasyVR::TRIGGER; //<-- start group (customize)
}
// ************************************************************************

// QUINTA PARTE (VOID ACTION Y VOID LOOP)
// ************************************************************************
void action();
void loop()
{
  easyvr.setPinOutput(EasyVR::IO1, HIGH); // LED on (listening)
  Serial.print("Say a command in Group ");
  Serial.println(group);
  easyvr.recognizeCommand(group);
  do
  {
    // can do some processing while waiting for a spoken command
  }
  while (!easyvr.hasFinished());
  easyvr.setPinOutput(EasyVR::IO1, LOW); // LED off
  idx = easyvr.getWord();
  if (idx >= 0)
  {
    // built-in trigger (ROBOT)
    // group = GROUP_X; <-- jump to another group X
    return;
  }
  idx = easyvr.getCommand();
  if (idx >= 0)
  {
    // print debug message
    uint8_t train = 0;
    char name[32];
    Serial.print("Command: ");
    Serial.print(idx);
    if (easyvr.dumpCommand(group, idx, name, train))
    {
      Serial.print(" = ");
      Serial.println(name);
    }
    else
      Serial.println();
```

```
    easyvr.playSound(0, EasyVR::VOL_FULL);
    // perform some action
    action();

  }
  else // errors or timeout
  {
    if (easyvr.isTimeout())
      Serial.println("Timed out, try again...");
    int16_t err = easyvr.getError();
    if (err >= 0)
    {
      Serial.print("Error ");
      Serial.println(err, HEX);
    }
  }
}
// ************************************************************************

// SEXTA PARTE (VOID ACTION)
// ************************************************************************
void action()
{
  switch (group)
  {
  case GROUP_0:
    switch (idx)
    {
    case G0_EMMRA_UNO:
      // write your action code here
      // group = GROUP_X; <-- cr jump to another group X for
composite commands
      break;
    }
    break;
  case GROUP_1:
    switch (idx)
    {
    case G1_ASISTENCIA:
      // write your action code here
```

```
      // group = GROUP_X; <-- or jump to another group X for
composite commands
      break;
    case G1_REHABILITACION:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
      break;
    case G1_SILLA:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
      break;
    }
    break;
   case GROUP_2:
    switch (idx)
    {
    case G2_MANO_DERECHA:

      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
      break;
    case G2_MANO_IZQUIERDA:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
      break;
    case G2_AMBAS_MANOS:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
      break;
    case G2_REINICIAR:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
      break;
    }
    break;
```

```
  case GROUP_3:
   switch (idx)
   {
   case G3_ABRIR:
    // write your action code here
    // group = GROUP_X; <-- or jump to another group X for
composite commands
    break;
   case G3_CERRAR:
    // write your action code here
    // group = GROUP_X; <-- or jump to another group X for
composite commands
    break;
   case G3_AGARRAR:
    // write your action code here
    // group = GROUP_X; <-- or jump to another group X for
composite commands
    break;
   case G3_SOLTAR:

    // write your action code nere
    // group = GROUP_X; <-- or jump to another group X for
composite commands
    break;
   case G3_REINICIAR:
    // write your action code here
    // group = GROUP_X; <-- or jump to another group X for
composite commands
    break;
   case G3_AUDIO:
    // write your action code here
    // group = GROUP_X; <-- or jump to another group X for
composite commands
    break;
   }
   break;
  case GROUP_4:
   switch (idx)
   {
   case G4_FUERTE:
    // write your action code here
```

```
      // group = GROUP_X; <-- or jump to another group X for
composite commands
       break;
     case G4_SUAVE:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
       break;
     }
     break;
   case GROUP_5:
     switch (idx)
     {
     case G5_CINCO:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
       break;
     case G5_DIEZ:

      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
       break;
     case G5_QUINCE:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
       break;
     case G5_REINICIAR:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
       break;
     }
     break;
   case GROUP_6:
     switch (idx)
     {
     case G6_LAPIZ:
      // write your action code here
```

```
      // group = GROUP_X; <-- or jump to another group X for
composite commands
      break;
    case G6_CUCHARA:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
      break;
    case G6_VASO:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
      break;
    case G6_POCILLO:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
      break;
    case G6_PINZA:
      // write your action code here

      // group = GROUP_X; <-- or jump to another group X for
composite commands
      break;
    case G6_REINICIAR:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
      break;
    }
    break;
  case GROUP_7:
    switch (idx)
    {
    case G7_ACTUAR:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
      break;
    case G7_PASIVA:
      // write your action code here
```

```
      // group = GROUP_X; <-- or jump to another group X for
composite commands
       break;
     }
     break;
   case GROUP_8:
     switch (idx)
     {
     case G8_DERECHA:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
       break;
     case G8_IZQUIERDA:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
       break;


     case G8_DETENER:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
       break;
     case G8_CONTINUAR:
      // write your action code here
      // group = GROUP_X; <-- or jump to another group X for
composite commands
       break;
     }
     break;
   }
}
// ****************************************************************************
```

8.8 Implementación del software Arduino

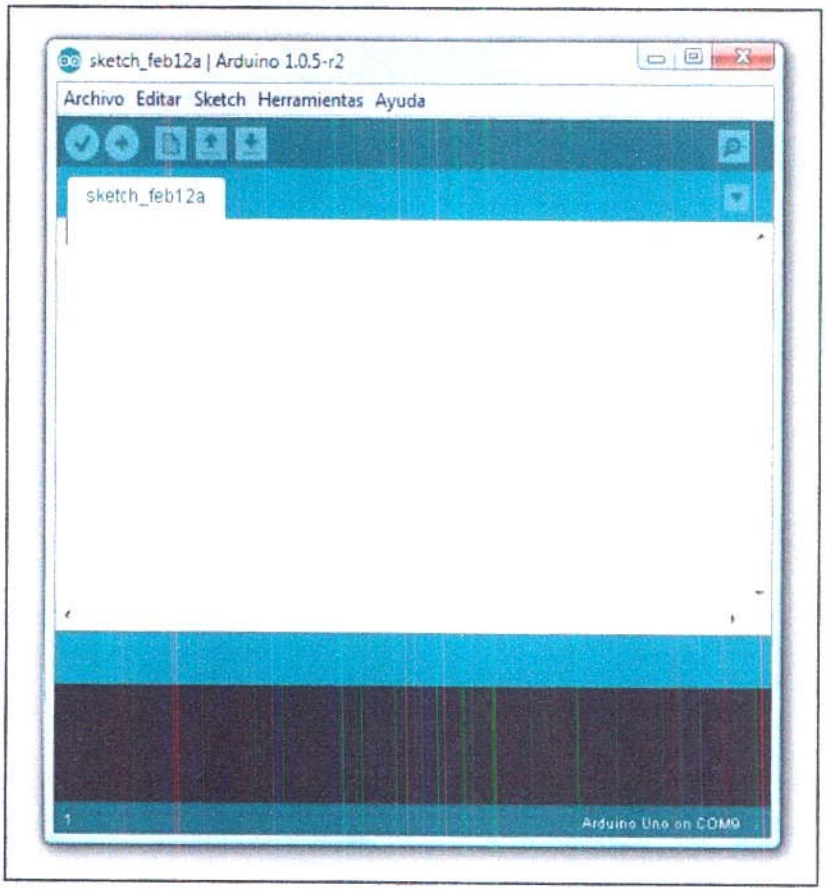


Figura 95. Sketch del Software Arduino.

Una vez ensamblada la **EasyVR Shield 3.0** en el **Arduino Mega 2560** procedemos a comprobar la funcionalidad de la tarjeta electrónica de reconocimiento de voz. Para ello utilizamos el **Software de Arduino**, que cuenta con dos programas para saber si el EasyVR está funcionando adecuadamente.

Los pasos para realizar este proceso son los siguientes:

- ✓ Ubicamos el jumper del MRV en PC.
- ✓ Conectamos el MRV con el PC a través de un cable USB.
- ✓ Abrimos el Software de Arduino.
- ✓ Damos clic en Archivo/Ejemplos/EasyVR/ EasyVRBridge o TestEasyVR

✓ Cargamos el código en la tarjeta Arduino Mega 2560.

✓ Damos clic en Herramientas/Monitor Serial

✓ Verificamos si la board de Arduino Mega 2560 detecta la board EasyVR

- **EasyVR Bridge:** Este programa se encuentra en los ejemplos del Software de Arduino, y permite validar la conexión entre el Puerto Serial del PC y el Módulo EasyVR.

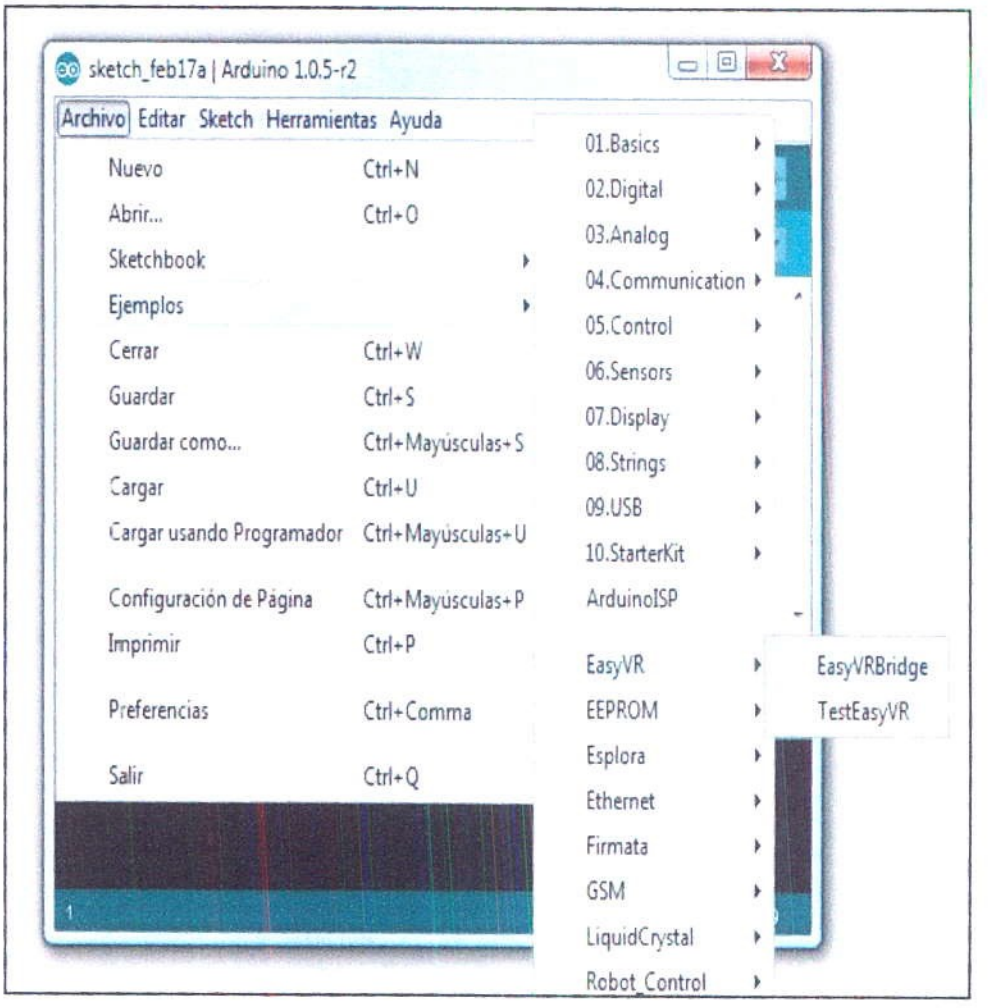


Figura 96. Selección del Programa Easyvr Bridge del Arduino

El código está elaborado para verificar el puente de conexión entre la boardArduino Mega 2560 y la board EasyVR Shield 3.0.

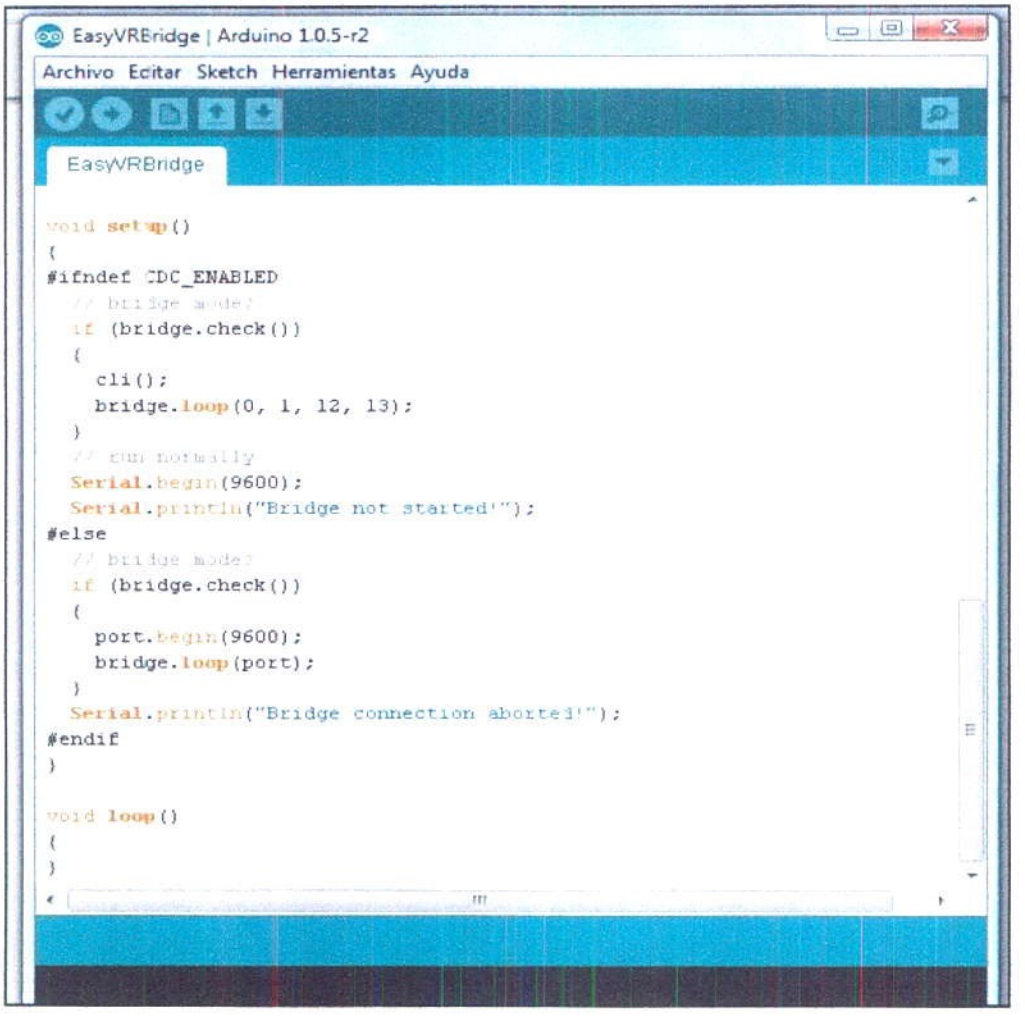


Figura 97. Programa Easyvr Bridge del Arduino

- **TestEasyVR:** Este programa se encuentra en los ejemplos del Software de Arduino, y permite validar si el microcontrolador ATmega16U2 del Arduino Mega 2560 detecta la EasyVR.

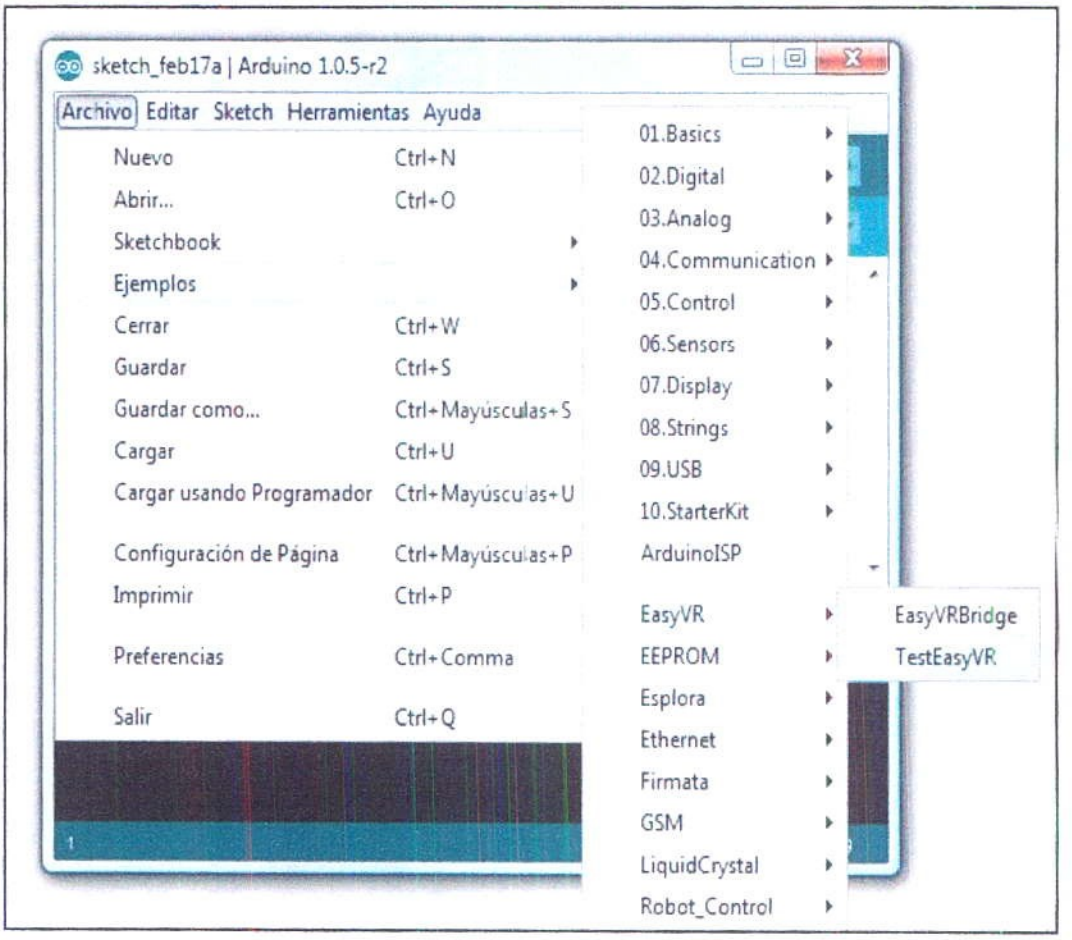


Figura 98. Selección del Programa TestEasyvr del Arduino

El código está elaborado para verificar que la board Arduino Mega 2560, ha reconocido correctamente la board EasyVR Shield 3.0

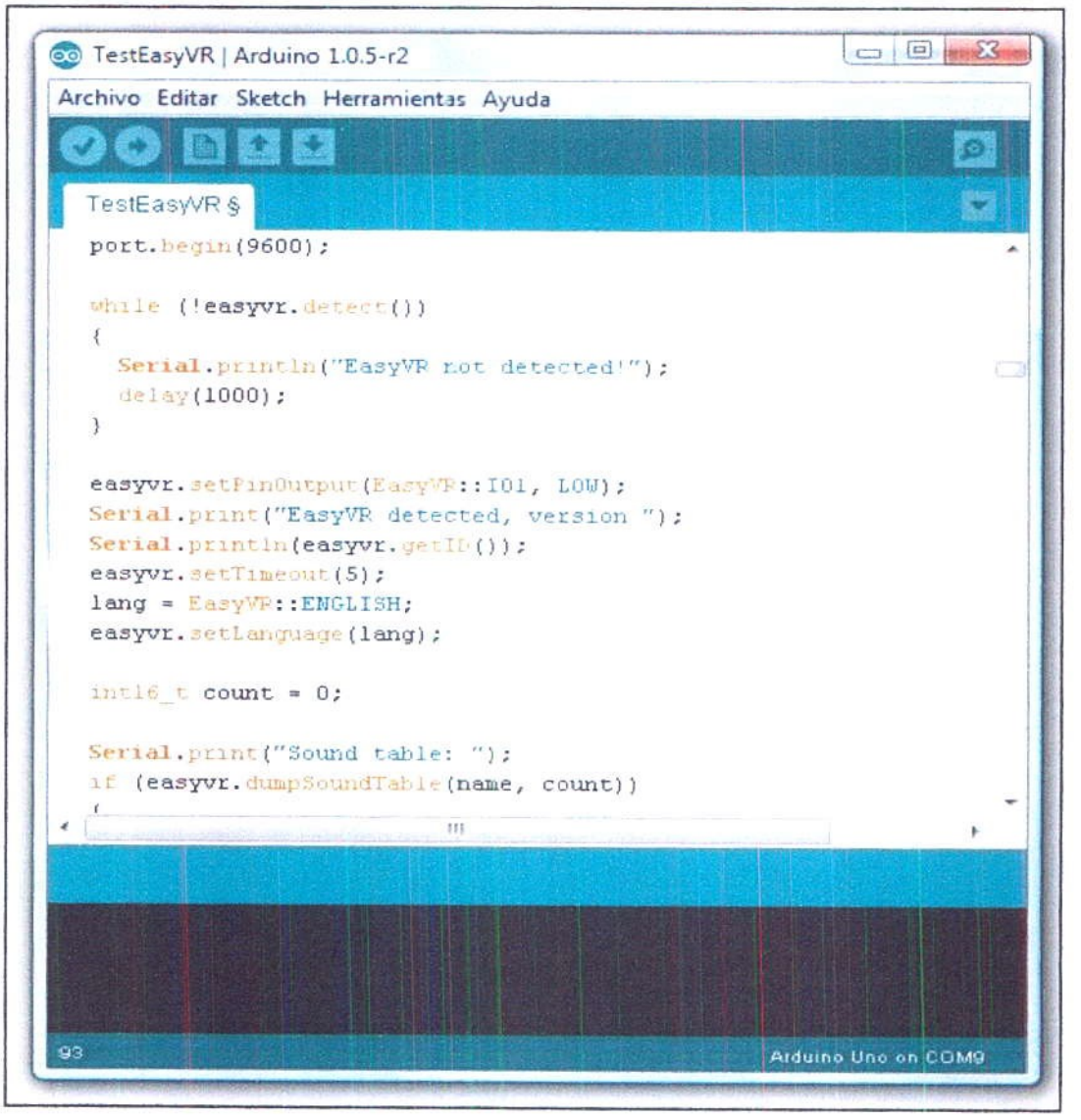


Figura 99. Programa TestEasyvr del Arduino

- **Código fuente para el MRV**

El EasyVR Commander genera un código de programación con los comandos SD creados en la lista de grupos, este código es un programa fuente estándar que debe ser abierto desde el Software Arduino, para asignar acciones específicas al microcontrolador ATMega 16 U2, una vez se reconozcan los comandos en la EasyVR Shield 2.0.

El código fuente para programar el MRV, tiene una estructura fundamentada en lenguaje de programación en C, con subprogramas propios del Software Arduino. A continuación describiremos las rutinas y los ciclos básicos del programa fuente:

- ✓ **Librerías Arduino:**

  Se incluyen las librerías para la interface del **Software Arduino** con las tarjetas electrónicas **Arduino Uno, Arduino Uno R3, Arduino Mega y Arduino Leonardo.**

- ✓ **Librería EasyVR:**

  Se incluye la librería para la interface del Software Arduino con la tarjeta electrónica **EasyVR Shield 3.0**

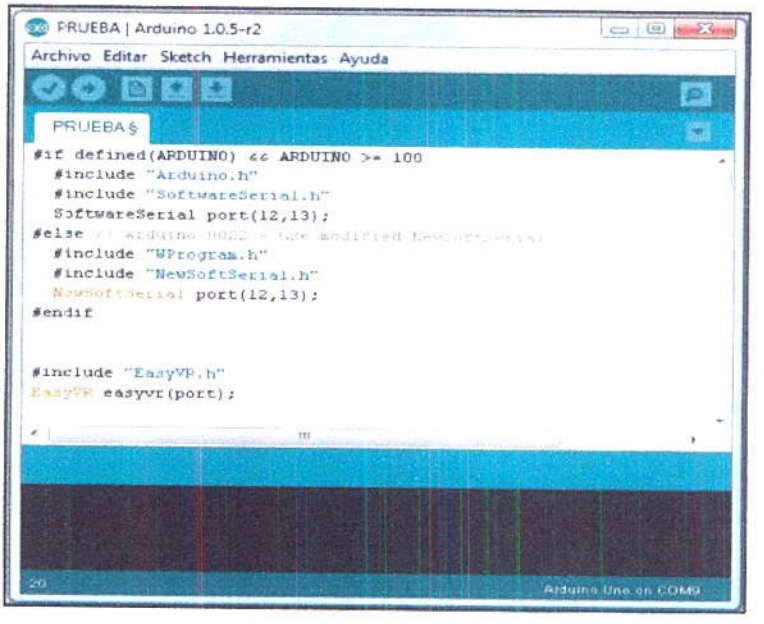


Figura 100. Librerias del código fuente

✓ **Presentación de Lista de Grupos:**

En esta rutina, se presentan los grupos creados en el Software EasyVR Commander y se enumeran de forma consecutiva los comandos e instrucciones SD.

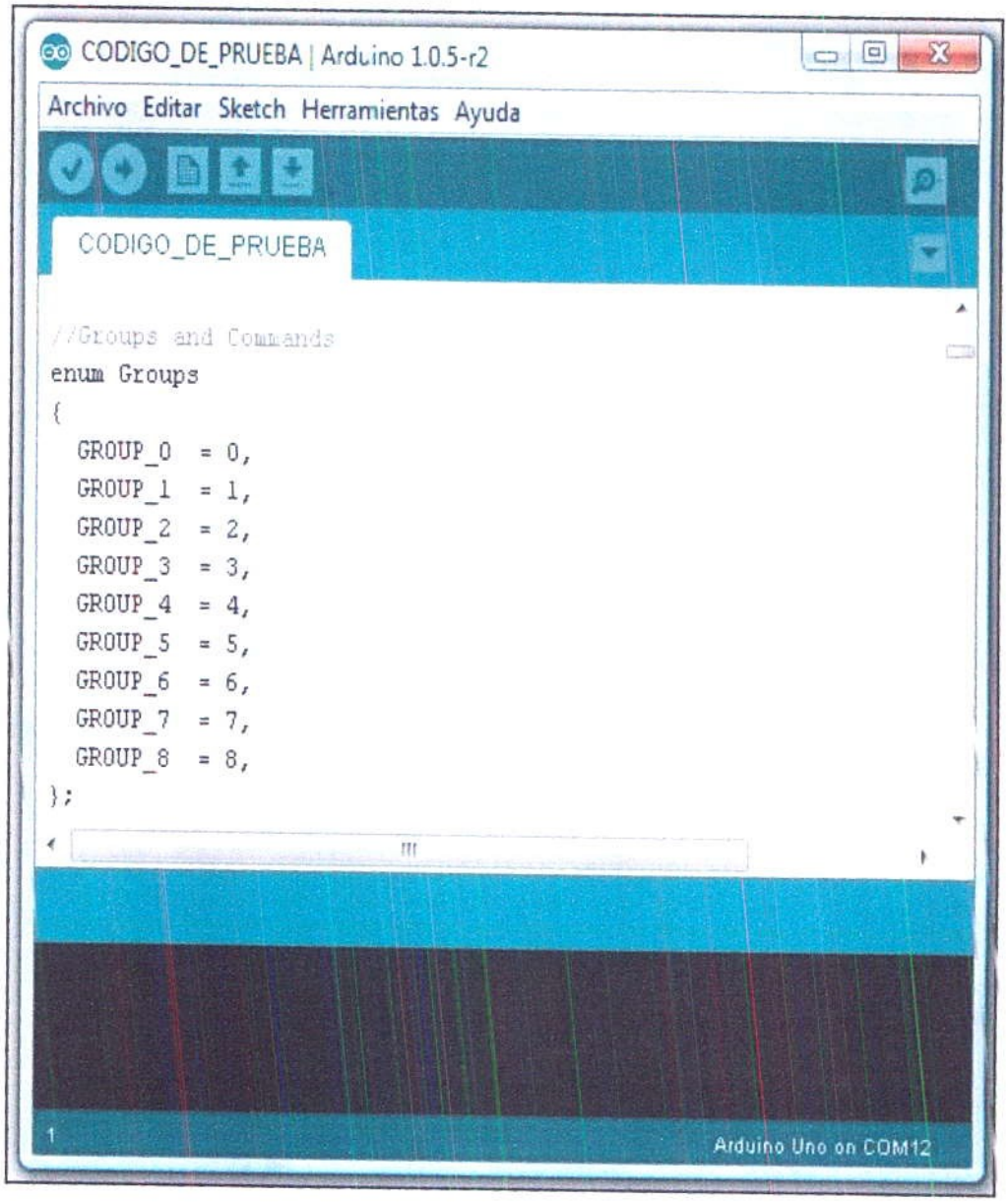


Figura 101. Grupos del código fuente

✓ **Subprograma EasyVRBridge:**

Subprograma para validar la adecuada conectividad entre la board EasyVR Shield 3.0 y la board Arduino Mega 2560.

Una vez detectada la EasyVR por el Microcontrolador ATMega16 U2, valida la conectivida entre el Puerto Serial del PC y el Módulo EasyVR.

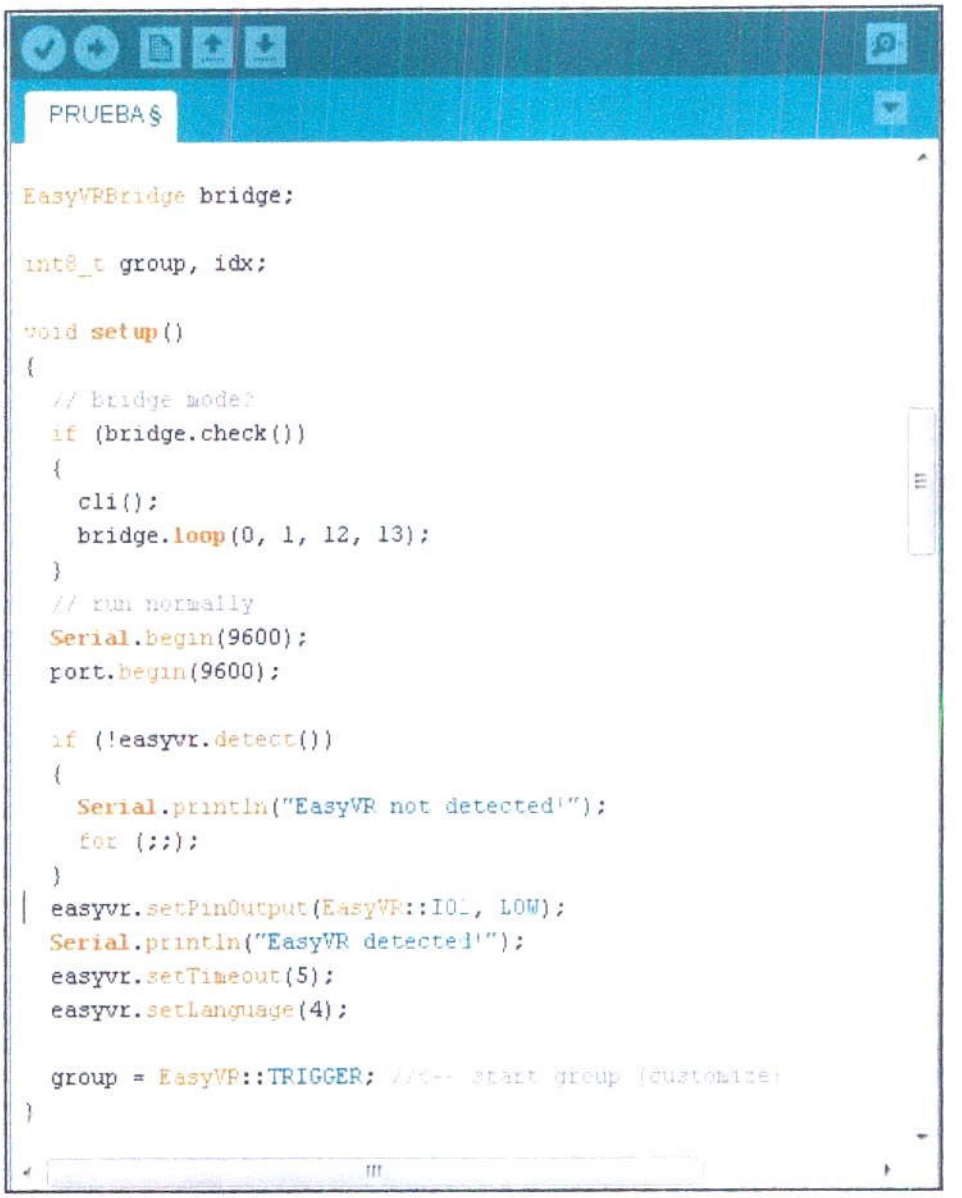


Figura 102. Subprograma EasyVR Bridge

✓ **Rutina de Comunicación EasyVR – Usuario:**

Es un subprograma que define una rutina de comunicación entre la board EasyVR y el Usuario.

Se genera una señal a través de un indicador (led y voz) para que el usuario pronuncie el comando, y establece el tiempo de recepción de la información para que dicho comando pueda ser reconocido por el módulo de reconocimiento de voz.

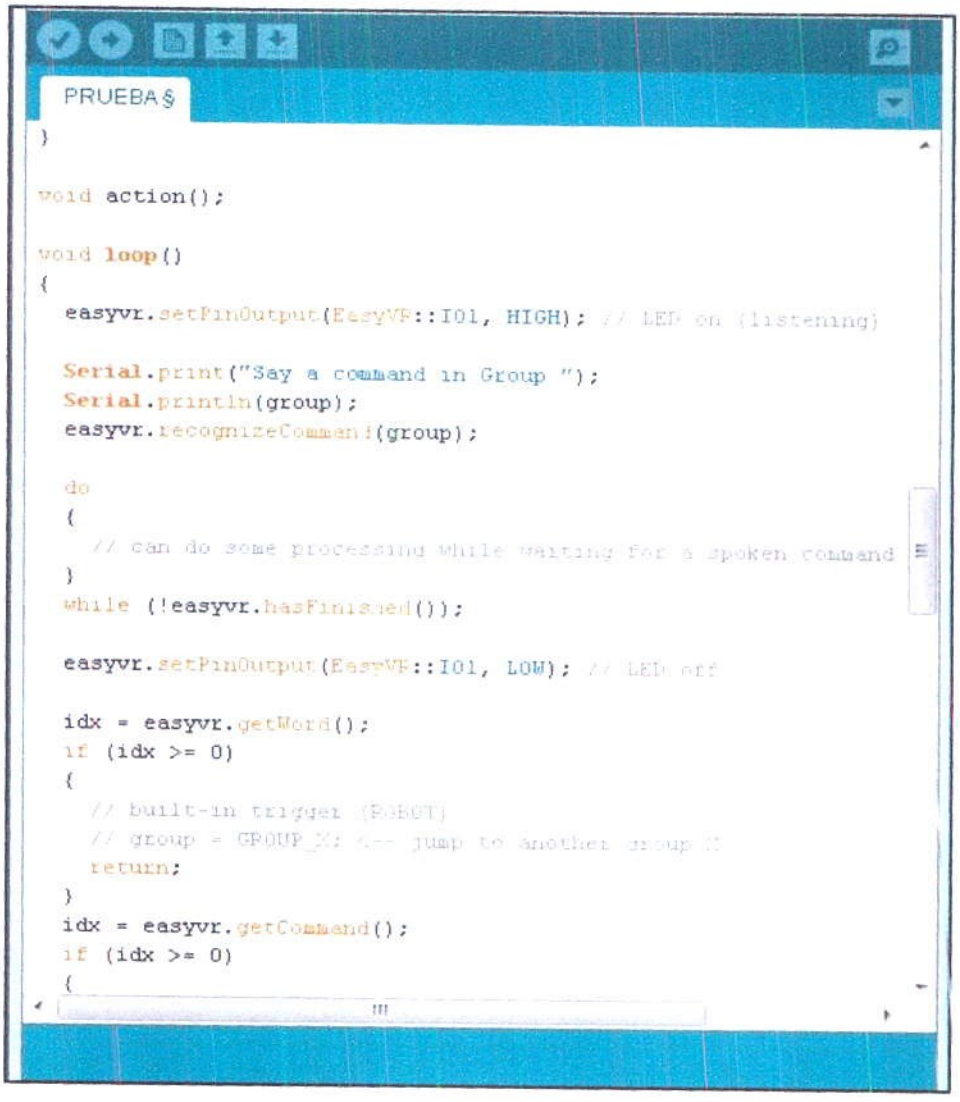


Figura 103. Comunicación Easyvr - Usuario

✓ **Rutina para Programar Acciones en el Microcontrolador:**

Una vez la EasyVR Shield 3.0 ha detectado el comando pronunciado por el usuario, se presenta la rutina para indicar acciones específicas al Microcontrolador ATMega 16 U2, según la instrucción o comando pronunciado.

El programa pregunta al usuario por una instrucción almacenada en uno de los grupos creados en el EasyVR Commander, Cuando la EasyVR Shield 3.0 detecta el comando pronunciado, permite crear una acción específica para ser realizada por el Microcontrolador ATMega 16U2, igualmente, permite direccionar al usuario hacia uno de los grupos, para que se pueda pronunciar una nueva instrucción.

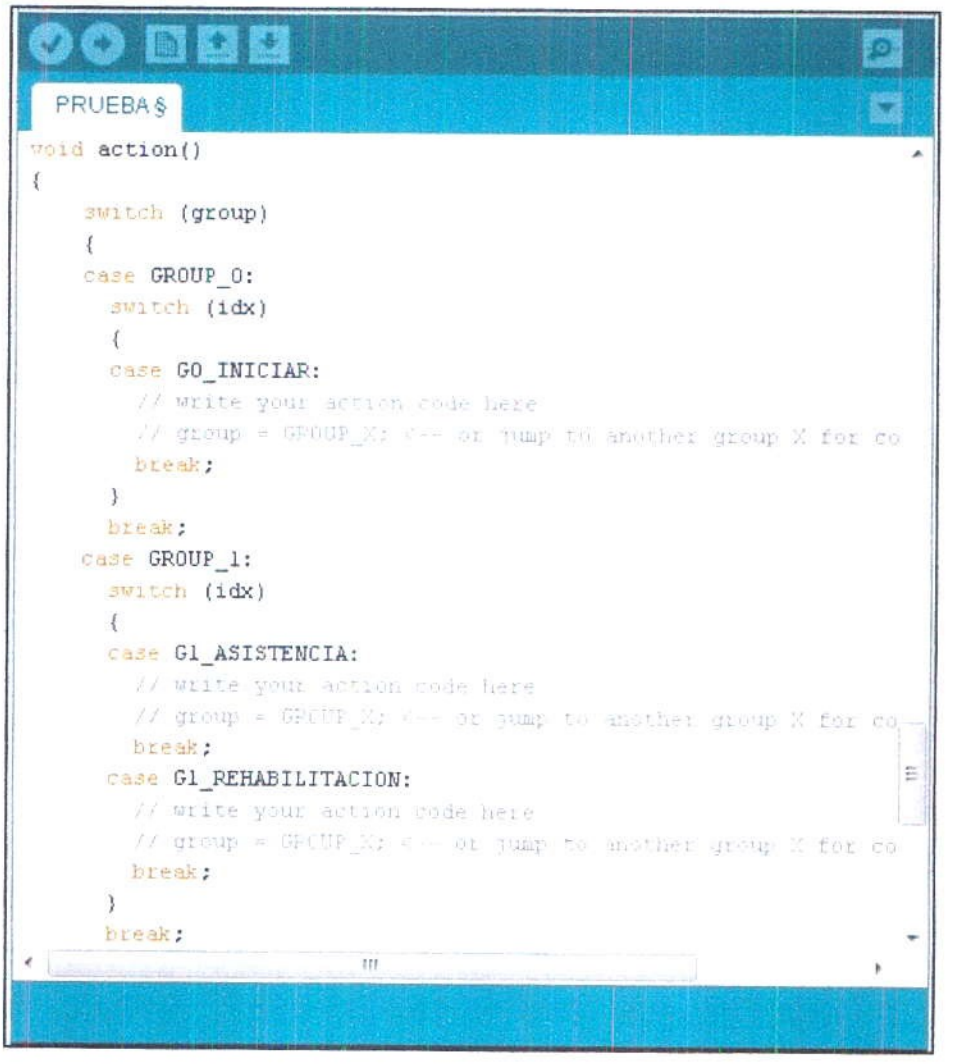


Figura 104. Rutina principal de acción

9. DISEÑO DEL MODULO DE CONTROL POR ELECTROMIOGRAFIA

Los pacientes con Lesión Medular Cervical Traumática, presentan diferentes grados de compromiso neurológico de acuerdo a la severidad y altura del trauma raquimedular. Los pacientes con lesiones medulares en C5 – C6 – C7, pueden conservar en menor o mayor grado, la actividad neuromuscular en Miembros Superiores, Los grupos musculares activos pueden ser rehabilitados, favoreciendo la plasticidad neuronal y la fuerza muscular en las zonas con interconexión neuronal central parcial.

Para estimular la rehabilitación activa, se implementa un sensor de señal muscular Modelo SEN_0661 compatible con Arduino y electrodos de electromiografía estandar. la intensidad de la contracción muscular es detectada y amplificada por el sensor muscular, el resultado es un voltaje diferencial entre los electrodos de electromiografía, señal filtrada y enviada a las entradas analógicas de la tarjeta electrónica Arduino Mega 2560. Según el Voltaje diferencial detectado se establecerán parámetros de respuesta del Exoesqueleto de Mano y Muñeca para la aplicación de rehabilitación.

La señal muscular detectada por los electrodos de electromiografía, pasa por el sensor de señal muscular, que realiza las siguientes operaciones electrónicas:

- Amplificación de la señal por un Amplificador de Instrumentación.
- Rectificador de Onda Completa.
- Amplificación
- Rectificador y Detector de picos.

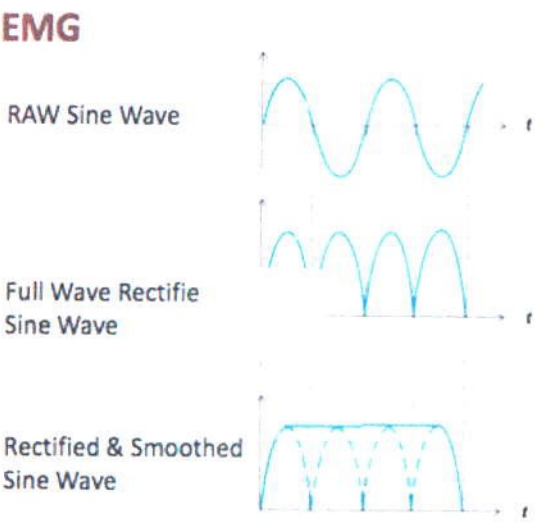


Figura 105. Procesamiento de la señal muscular. http://cdn.sparkfun.com

9.1. Sensor de Señal Muscular:

Los sensores de señal muscular se encuentran ubicados en el antebrazo del paciente, recogen la información de tres electrodos (Rojo, Azul y Negro). Mide la diferencia de presión entre los electrodos rojo y azul, los convierte en un voltaje y los envía al módulo de control electrónico (Tarjeta Arduino Mega 2650). El electrodo negro es la tierra del paciente al circuito electrónico.

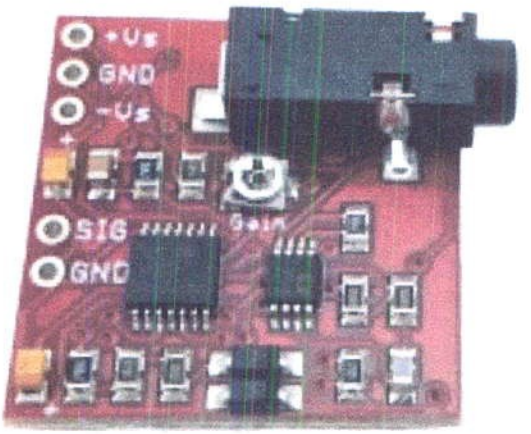


Figura 106. Sensor de Electromiografia SEN_0661. http://cdn.sparkfun.com

- **Pines de Voltaje**: Los sensores de electromiografía deben ser alimentados con un voltaje diferencial de -5 a -3 Voltios, con una tierra comun a la fuente de alimentación y a la tarjeta electrónica Arduino Mega 2560.
  +Vs: Alimentación de + 5 Voltios.
  -Vs: Alimentación de – 3 Voltios.
  GND: Tierra.

- **Pines de Señal de Salida:** La señal detectada en los electrodos de electromiografía, es enviada a los módulos de control por las salidas:
  Output Signal: Señal de Salida
  GND: Tierra

- El Puerto de 3.5 mm, es un conector hembra de 3 entradas, que recoge las señales de los tres electrodos de electromiografía.

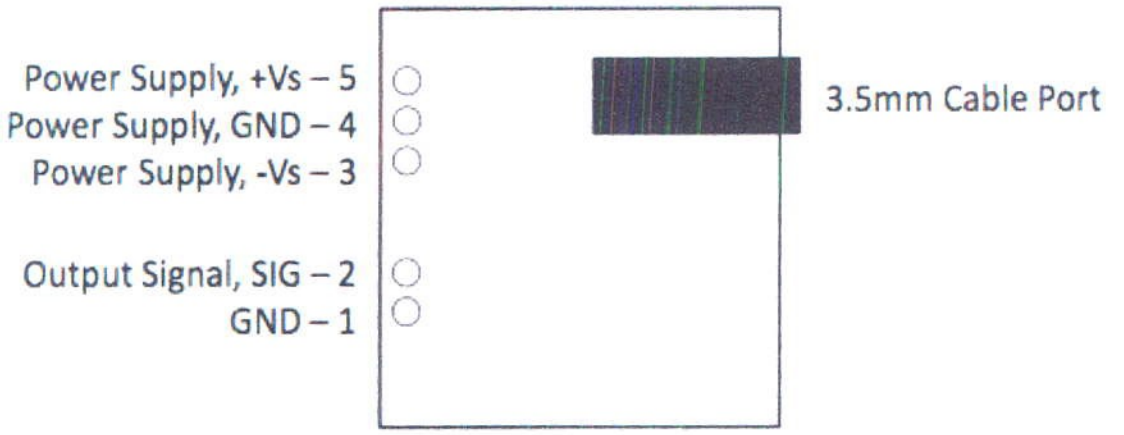


Figura 107. Entradas y Salidas del Sensor de Electromiografía. http://cdn.sparkfun.com

Idealmente nuestro proyecto implica la alimentación de todo el sistema por fuentes de alimentación conmutada para uso médico, estas fuentes se encuentran conectadas a la red eléctrica pública, lo cual introduce el ruido de 60Hz a la señal detectada por los electrodos de electromiografía, por tal motivo fue necesario implementar un sistema de alimentación con baterias recargables, para obtener un adecuado funcionamiento del dispositivo, como lo sugiere el fabricante del sensor muscular.

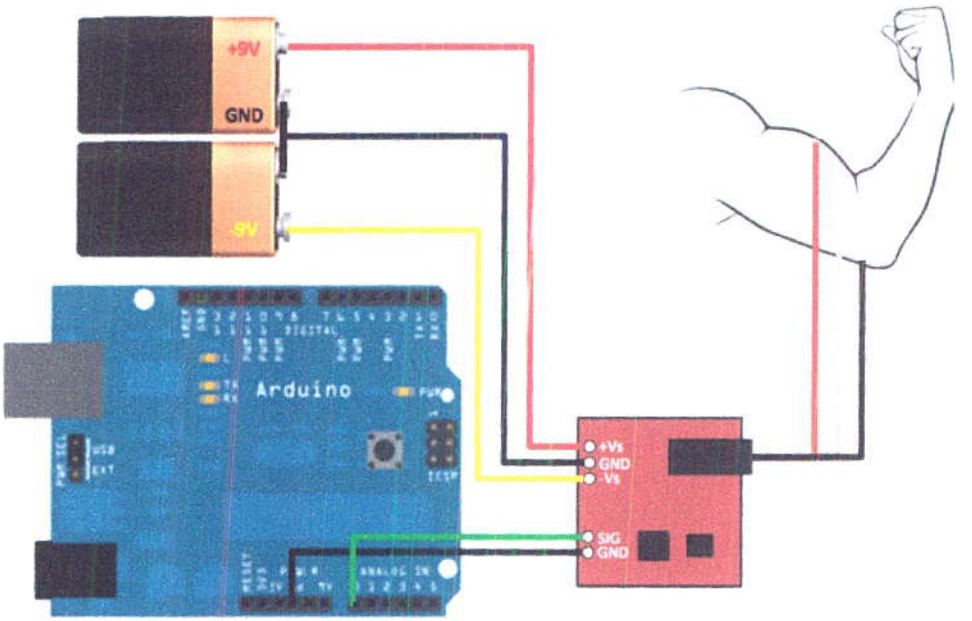


Figura 108. Conexiones del Sensor de Electromiografía. http://cdn.sparkfun.com

9.2 Electrodos de Electromiografía

La captación de las señales eléctricas producidas por los músculos durante una contracción muscular se conoce como electromiografía. Estas señales son generadas por el intercambio de iones a través de las membranas de las fibras musculares debido a una contracción muscular.

La electromiografía (EMG) consiste básicamente en la adquisición, registro y análisis de la actividad eléctrica generada en nervios y músculos a través de la utilización de electrodos (superficiales, de aguja, implantados). Las mediciones extraídas de EMG proporcionan una información valiosa acerca de la fisiología y los patrones de activación muscular.

Dicha información refleja las fuerzas que son generadas por los músculos y la temporización de los comandos motores, esta señal es detectada por los electrodos de electromiografía superficiales y tratada por el sensor de señal muscular, para ser enviada finalmente al módulo de control electrónico.



Figura 109. Cables y Electrodos de EMG. http://cdn.sparkfun.com

Se pueden obtener señales Electromiográficas para el estudio del movimiento utilizando electrodos de superficie. La amplitud y anchura de banda de la señal EMG no están determinada únicamente por las fuentes electrofisiológicas y sus distancias hasta los electrodos, sino también por los tipos y tamaños de electrodos utilizados y por el espaciamiento entre electrodos. Los electrodos de superficie van unidos a la piel sobre el segmento muscular que se está detectando.

Los electrodos de superficie se utilizan para estudiar la actividad de todo el músculo superficial. El espaciamiento entre electrodos determina el volumen de registro o recepción del tejido, resultando los espaciamientos más pequeños en registros más selectivos. Los electrodos de superficie suelen ser de ranura, con pasta de electrodo llenando la cavidad para conseguir más contacto con la piel y reducir la impedancia de los electrodos.

Los electrodos comercializados pueden ser desechables, como los electrodos Electrocardiográficos (ECG), o reutilizables con una protección de plástico y un cuello adhesivo por ambos lados. Su diámetro va de 2 a 10 mm para la parte activa del electrodo. Los electrodos de cloruro de plata-plata (Ag-Ag Cl) con pasta de cloruro se utilizan invariablemente debido a sus propiedades de estabilidad y reducción del ruido.

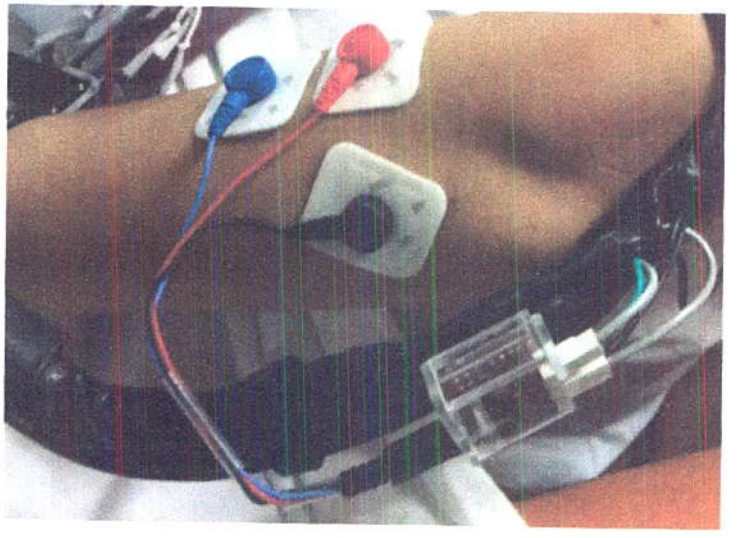


Figura 110. Cables y Electrodos de EMG en antebrazo

Idealmente los electrodos que miden el voltaje diferencial deben estar conectados de 1 a 2 cm de distancia, mientras qie el electródo que va a tierra, debe estar cerca de las superficies oseas.

8.3 Implementación del Software Arduino

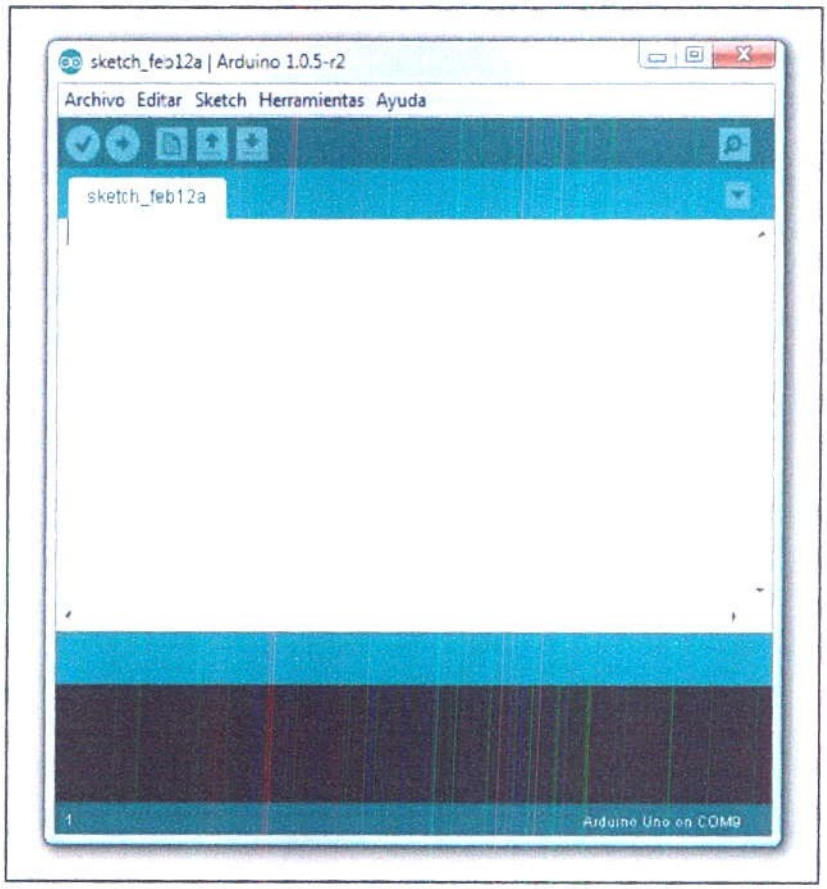


Figura 111. Sketch del Software Arduino.

La señal de electromiografía detectada es un voltaje diferencial de 0 a 10 milivoltics, esta señal es amplificada en el sensor muscular, rectificada, filtrada y procesada hasta convertirse en un voltaje de 0 a 5 Voltios, para ser llevado a las entradas analógicas del Arduino Mega 2560, el conversor análogo digital convierte estos valores analógos en valores digitales.

La lectura del valor digital medido, estará en un rango de 0 a 1023 bits, de acuerdo a esta intensidad del voltaje medido, se pueden establecer respuestas de abrir o cerrar la mano del paciente discapacitado.

Se utiliza la subrutina de conversión análoga a digital del Software Arduino, modificandose su estructura a las características del dispositivo biomédico.

Todos los detalles de la estructura del software se tratan en el capítulo 4 del informe final del proyecto Prototipo EMMRA-1.

9. Plataforma electrónica del módulo de control

El módulo de reconocimiento de voz EasyVR Shield 3.0 esta superpuesto a la tarjeta Arduino Mega 2560, la cual requiere conectividad con la fuente de alimentación conmutada, 2 conexiones por cada sensor de fuerza resistivo (10 SFR), 3 conexiones por cada servomotor (16 servomotores), una entrada por cada sensor de electromiografía (2 sensores), 2 conexiones por cada indicador led de alta luminosidad (4 indicadores), 2 conexiones con el equipo de transmisiόm de movimiento a la silla giratoria, conexiones con el micrófono y el amplificador de audio, conexiones con el PC, etc.

Lo anterior, implica que algunos cables tienen que recorrer una distancia de 1 a 3 metros, para interconectarse con el exoesqueleto de mano y muñeca, los sensores, los servomotores, los indicadores led, las fuente de alimentación y el equipo de transmisión de movimiento a la silla giratoria.

Se diseño y fabricó una plataforma de interconectividad entre la tarjeta Arduino Mega 2560 y todos los componentes electrónicos del sistema, esta plataforma se diseña en software ISIS-ARES versión estudiantil y se fabricó en baquelas de fibra de vidrio.

La plataforma electrónica contiene dos baquelas de fibra de vidrio, que permiten la interconectividad con todos los elementos del sistema descritos anteriormente.

En el plano inferior se conectan las lineas de alimentación de las fuentes de alimentación conmutada y se crea un divisor de corriente de alta impedancia para proteger la tarjeta Arduino Mega 2560 del Amperaje entregado por las Switching Power Supply. Además se conecta las lineas de tierra del circuito electrónico y la señal de los Sensores de Fuerza Resistivos de la mano izquierda y los sensores de Electromiografía de ambas manos.

En el plano superior se conectan las señales de los Sensores de Fuerza Resistivos de la mano derecha y todos los cables de los servomotores de ambas manos.

- El diseño de ambas placas se realiza en el Software ISIS-ARES versión libre para estudiantes.

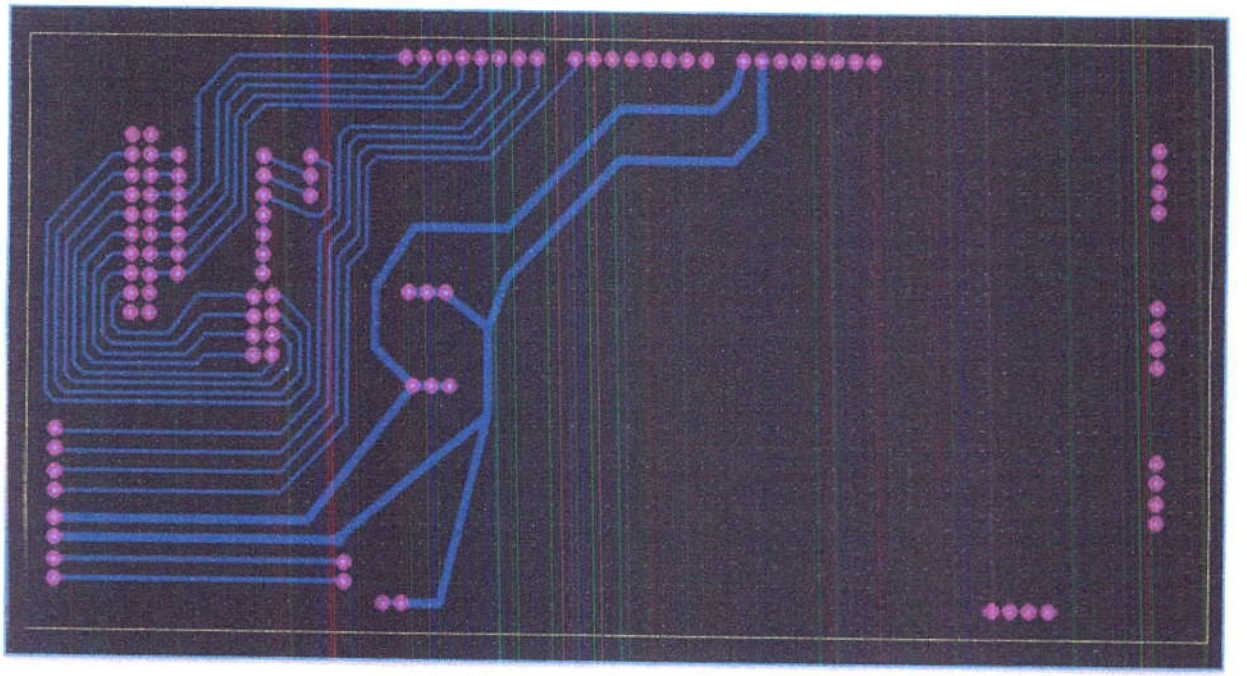


Figura 112. Diseño en Software ARES de la placa inferior

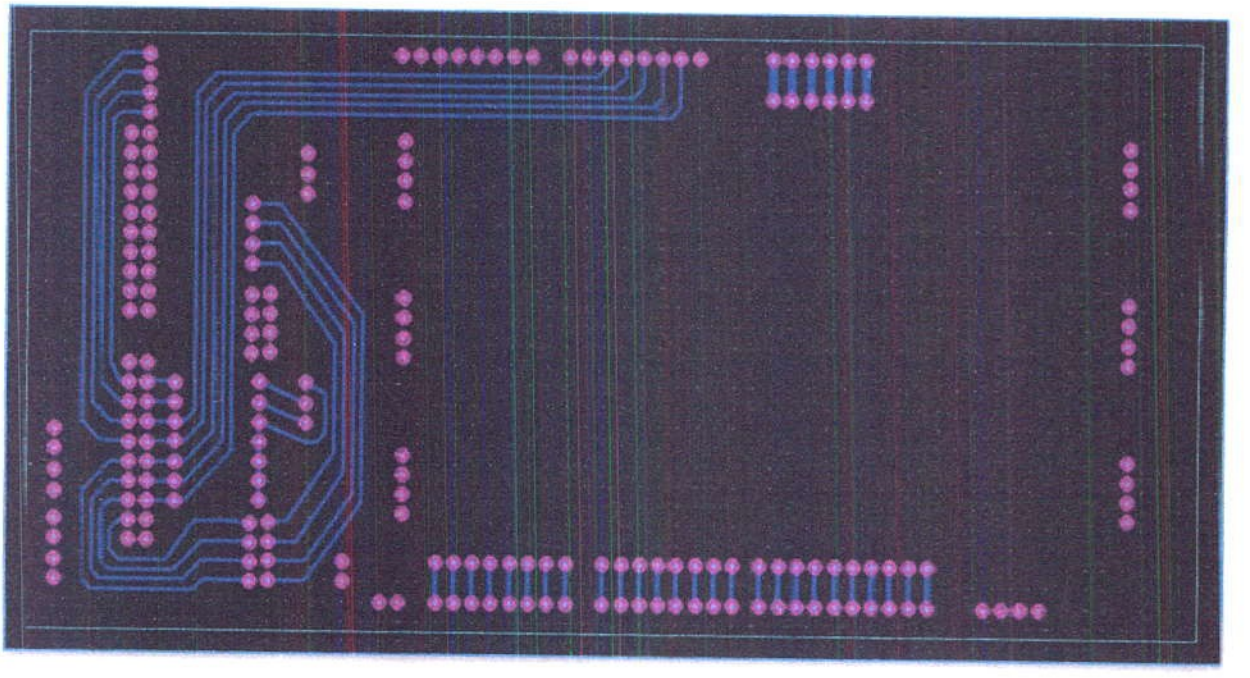


Figura 113. Diseño en Software ARES de la placa superior

- El diseño se archiva en formato PDF para ser impreso en una impresora laser.

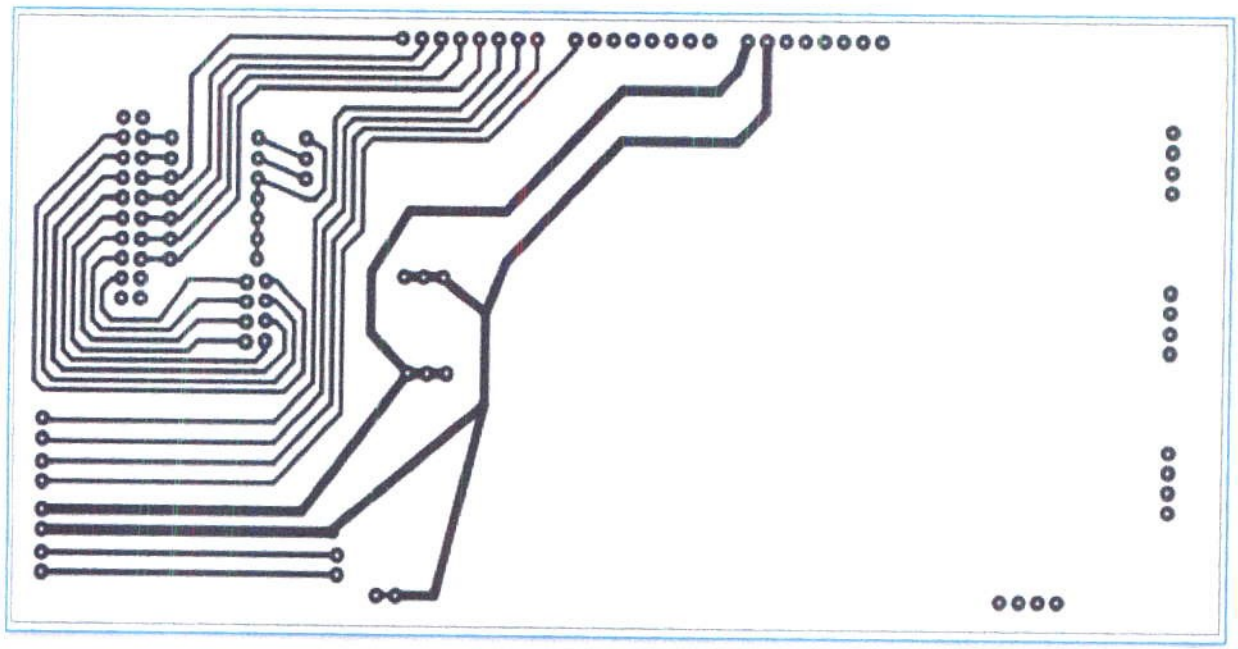


Figura 114. Impresión PDF de la placa inferior

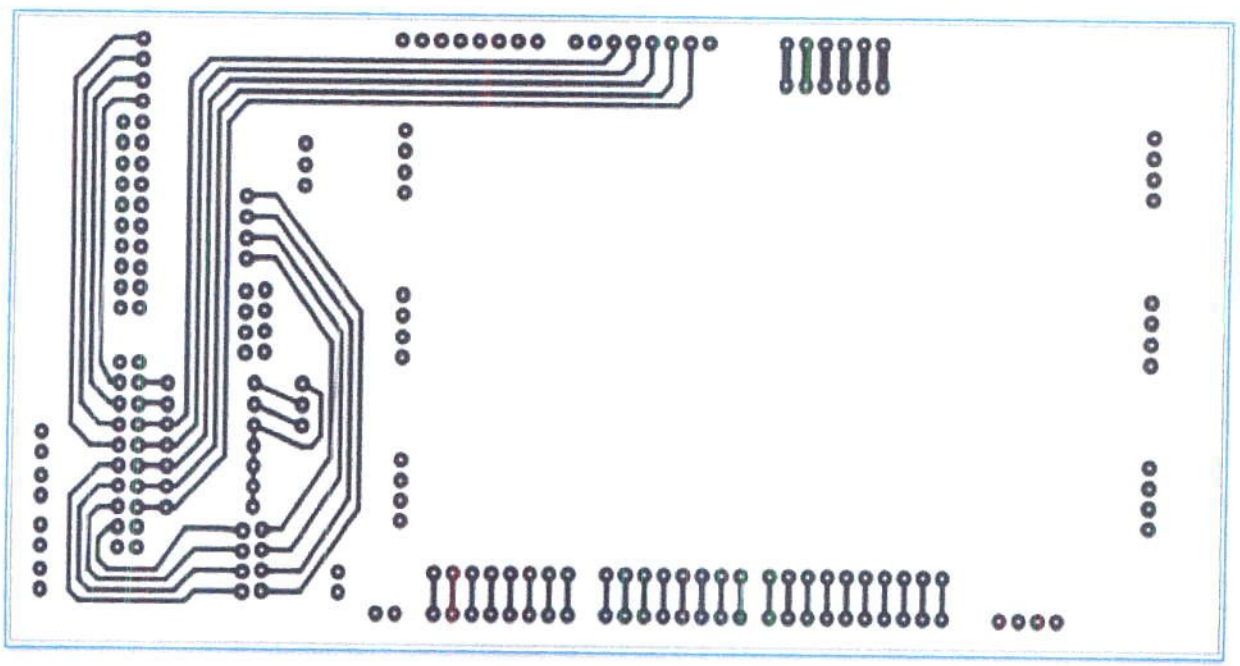


Figura 115. Impresión PDF de la placa superior

- Se calca el circuito a la baquela de fibra de vidrio con calor.

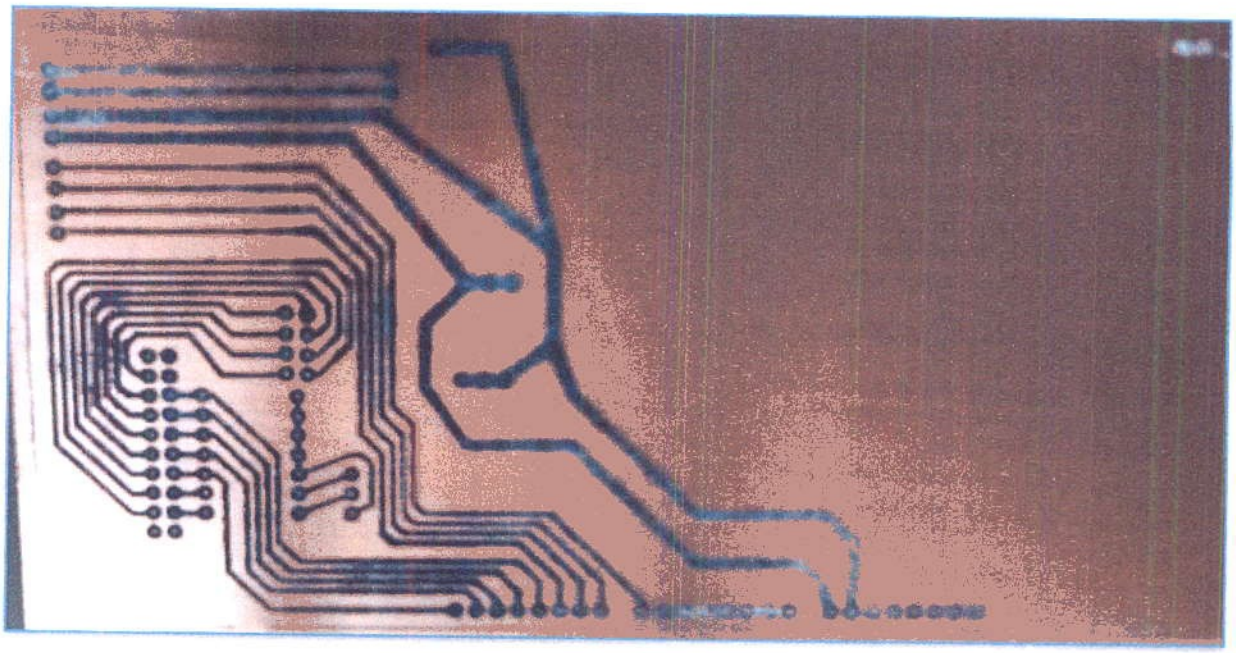


Figura 116. IPlaca inferior calcada en la baquela

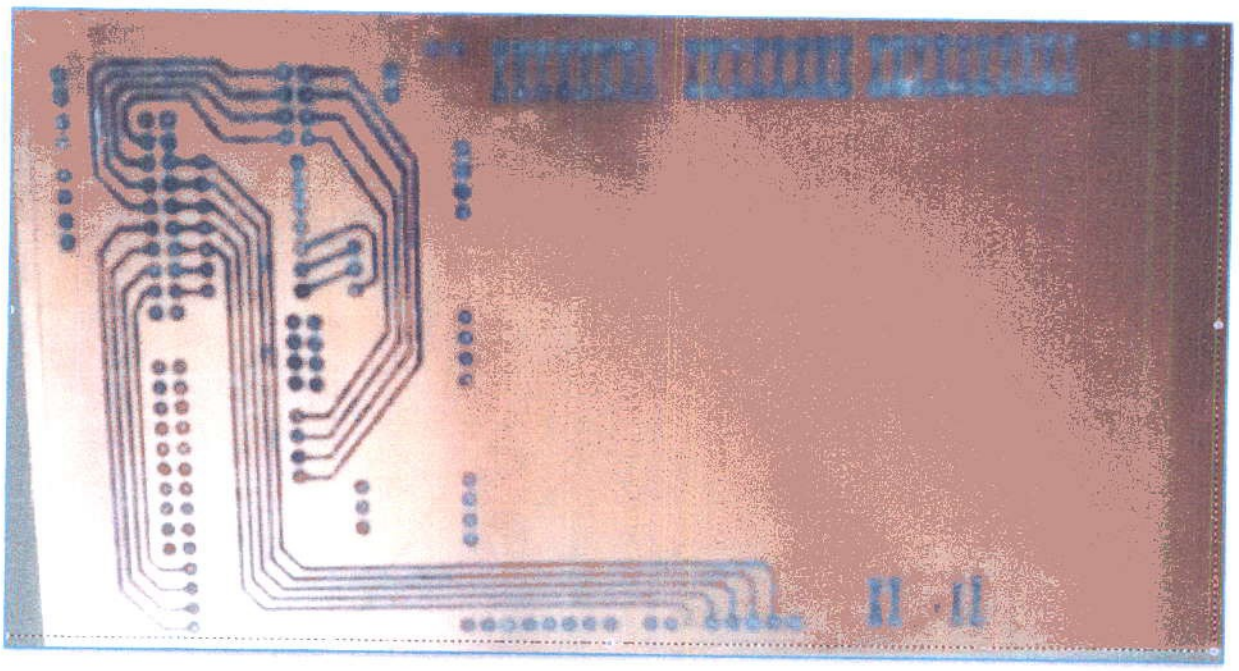


Figura 117. Placa superior calcada en la baquela

- Se retira el cobre sobrante y se ensamblan los componentes electrónicos.

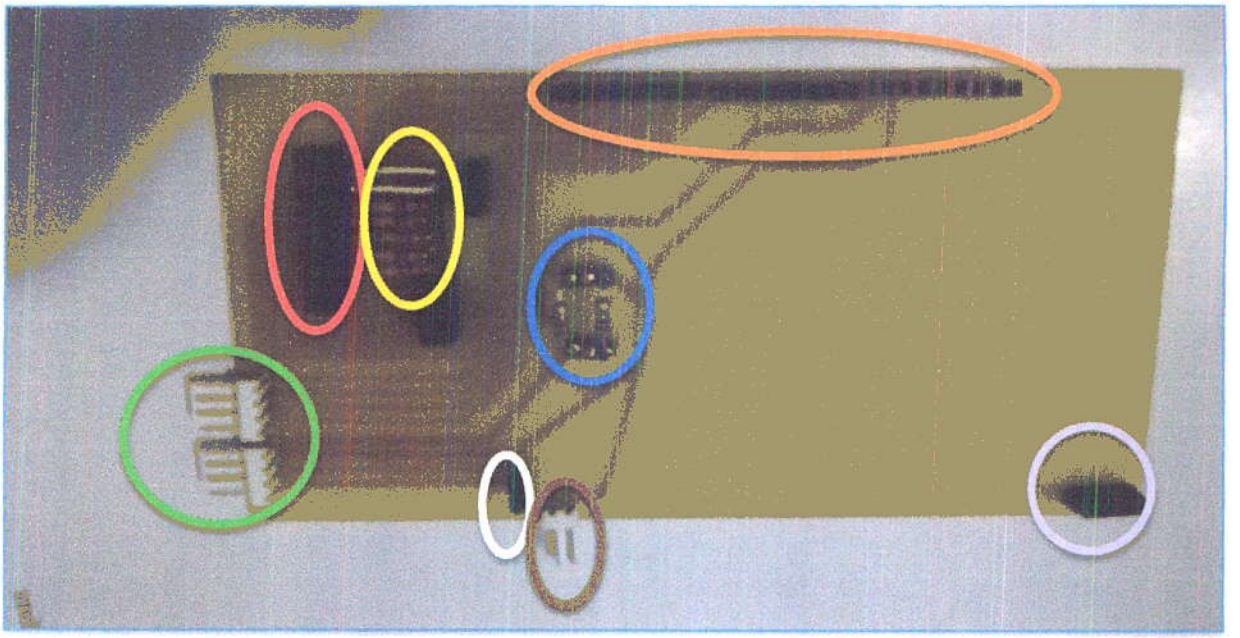


Figura 118. Tarjeta Electrónica de la placa inferior

- La plataforma inferior contiene las conexiiones representadas en la gráfica:

  - **Verde:** Conectores GP de 4 pines. Para la entrada de voltaje positivo y negativo al sistema. Para la salida de los voltajes a los SFR, Para la entrada de la señal de electromiografía.
  - **Roja:** Conectores para la entrada de la señal de los Servomotores de la mano derecha.
  - **Amarilla:** Resistencias para la conexión a tierra de los SFR.
  - **Naranja:** Conexión entre la p ataforma inferior y la superior
  - **Azul:** Resistencias de alta impedancia (1 MΩ) para atenuar el amperaje que ingresa a la tarjeta arduino.
  - **Blanco:** Conexión entre la plataforma inferior y la superior para el Voltaje de los SFR.
  - **Café:** Conexión a tierra de la fuente de alimentación y la plataforma.
  - **Lila:** Conexión entre la plataforma inferior y la superior

10. Fotos del producto terminado

- Montaje de la tarjeta Arduino Mega 2560 con el EasyVR Shield 3.0

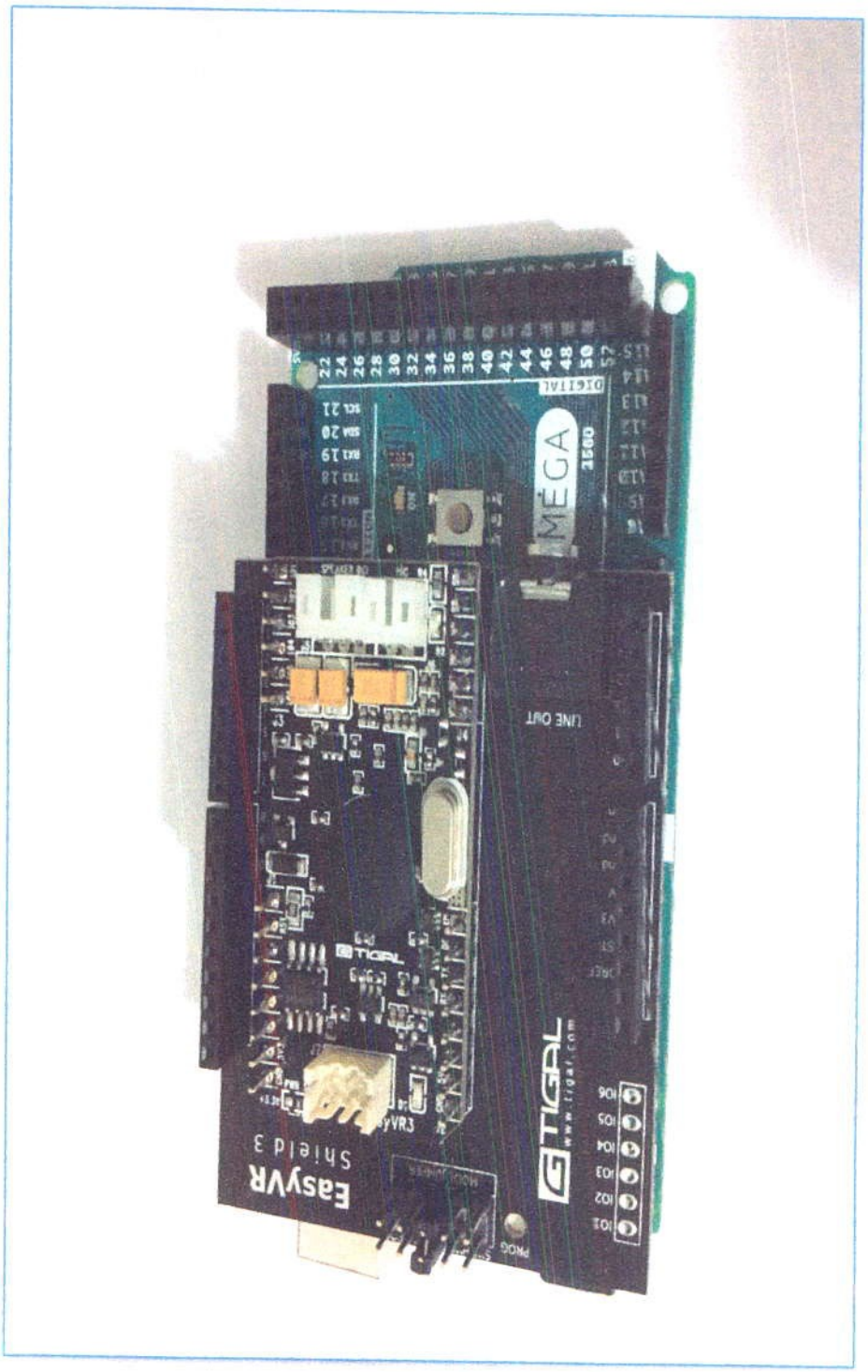


Figura 119. Tarjeta Arduino Mega acoplada a la EasyVR Shield 3.0

- Montaje de las tarjetas Arduino Mega 2560 y EasyVR Shield 3.0, sobre la plataforma del módulo de control electrónico.

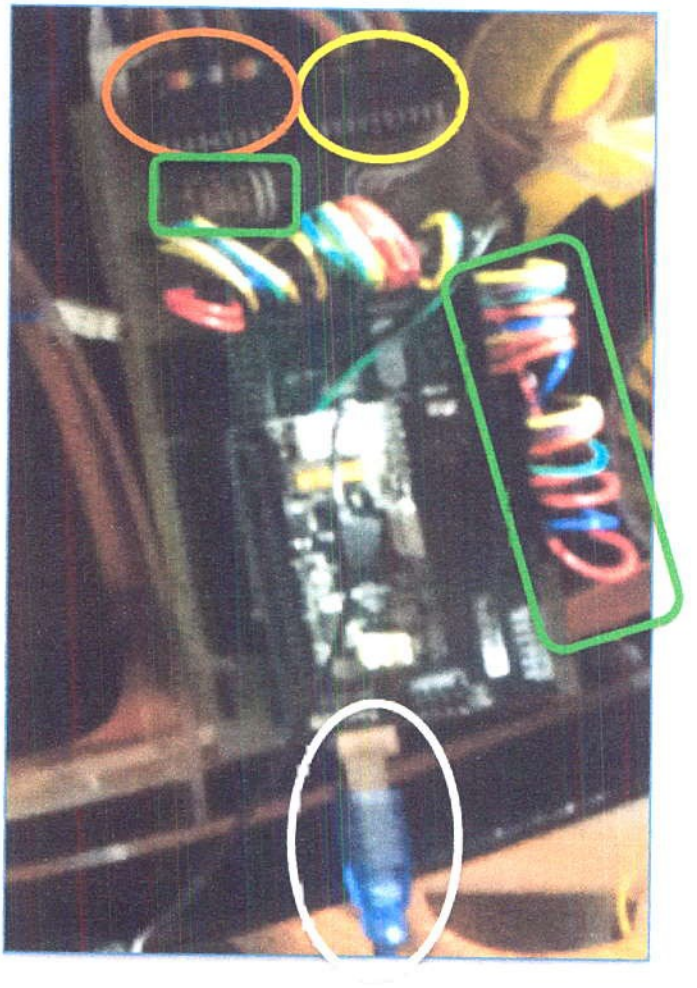


Figura 120. Módulo de control ensamblado en la plataforma superior

- ✓ **Conectores de seguridad:** Conectan los cables de los servomotores de la mano derecha (círculo naranja), y los de la mano izquierda (círculo amarillo).
- ✓ **Resistencias a tierra de los SFR:** Las resistencias de 10KΩ conectan la salida de la señal a la tierra del módulo de control (recuadro verde).
- ✓ **Conectores de circuitos:** Conectan las salidas de la plataforma al módulo de control del dispositivo. Se realiza de esta forma con propósitos educativos (recuadro rojo).
- ✓ **Conector USB:** Cable USB macho hembra conecta el módulo de control con el computador.

- Modulo de control del dispositivo ensamblado en las caras laterales de la silla giratoria del paciemte discapacitado.

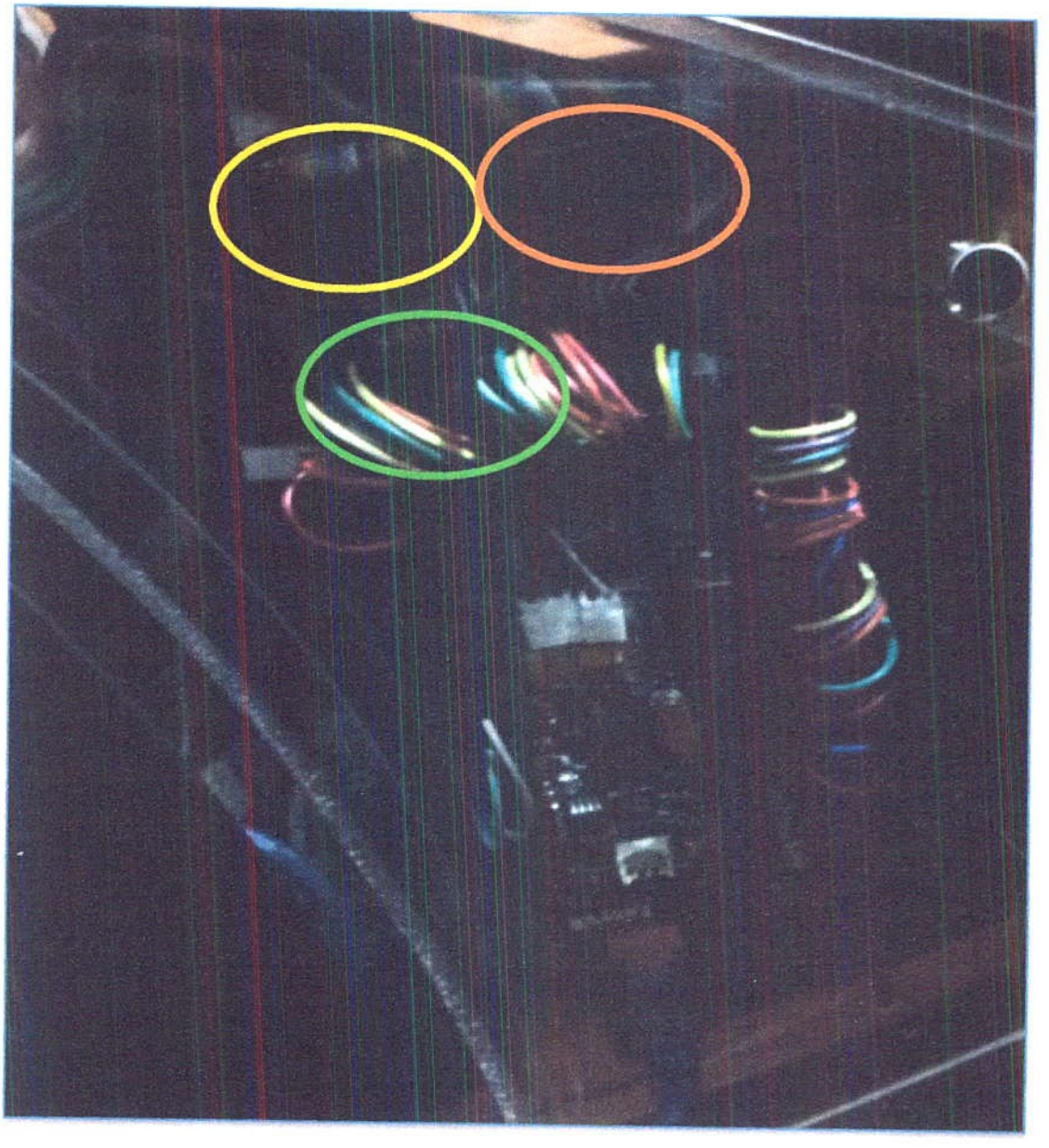


Figura 121. Modulo de contro del dispositivo

- **Amarillo:** Salida de las señales de los Servomotores de la mano izquierda
- **Naranja:** Salida de las señales de los Servomotores de la mano derecha
- **Verde:** Resistencias de la conexión a tierra de los SFR de mano izquierda

- Modulo de control del dispositivo ensamblado en las caras late-ales de la silla gi-atoria del paciemte discapacitado.

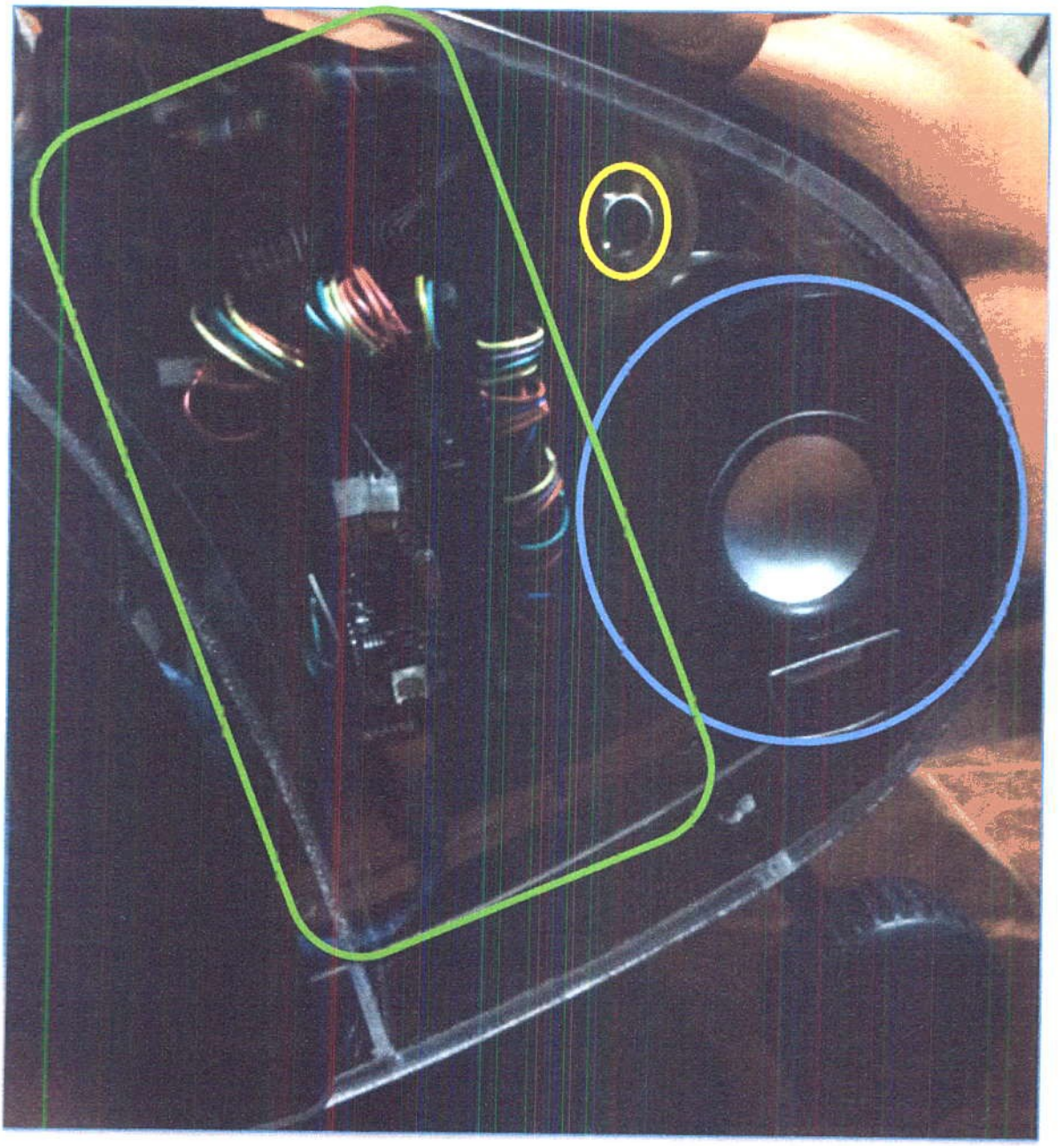


Figura 122. Modulo de control del dispositivo

- Verde: Tarjetas Electrónicas
- Amarillo: Microfono
- Azul: Parlante

INFORME FINAL DEL PROYECTO
EXOESQUELETO DE MANO Y MUÑECA CON APLICACIONES DE REHABILITACION Y ASISTENCIA (PROTOTIPO EMMRA-I)

CAPITULO IV
ADAPTACIONES A LA SILLA RECLINABLE

INVESTIGADORES:

JUAN DAVID MORENO ARANGO
JULIAN ALBERTO MORENO ARANGO

ASESORES:

VICTOR MAURICIO ACEVEDO CORREA

Manizales, 01 de Noviembre de 2015

INTRODUCCIÓN

Los pacientes discapacitados por Lesión Cervical Medular Traumática **LCMT** según el nivel del daño neurológico, deben permanecen la mayor parte del tiempo acostados, sentados en una silla reclinable, o sentados en una silla de ruedas. Algunos pacientes luego de varios meses del trauma, presentan lesiones en los sitios de presión o contacto de la piel con las superficies de la cama o sillas, estas lesiones progresan a **escaras** que pueden necrosar la piel y formar grandes zonas de infección, inflamación o necrosis.

Los materiales del exoesqueleto mecánico (guante de cuero y neopreno, eslabones en acrílico, ejes tipo remache, servomotores, conexiones y cables), son de materiales suaves y de bajo peso, sin embargo para un paciente discapacitado con pérdida de la fuerza en los miembros superiores, utilizar el dispositivo puede resultar en un reto importante. Por tal motivo, se retiraron las estructuras de control del dispositivo (tarjetas electrónicas, módulo de reconocimiento de voz y módulo de control por señal muscular. Se ubicaron todos como un solo módulo de control electrónico en las caras laterales de una silla reclinable, adaptada para cumplir muchas funciones como parte integral del dispositivo robótico biomédico.

La silla reclinable no solo contiene los módulos de control electrónico en su cara lateral, sino también las fuentes de alimentación para los exoesqueletos de las manos derecha e izquierda a cada lado. Igualmente, se realizaron adaptaciones para dar confort, seguridad y protección al paciente. Una espuma antiescara cubre toda la superficie de contacto en la base, el espaldar y los apoya brazos. La capa superficial de la silla es una cubierta de tela prana de fácil limpieza y sin relieve, para garantizar una superficie lisa en contacto con la piel.

Otro de los grandes retos para el paciente con LCMT, es alcanzar una ubicación espacial en su entorno doméstico, ya que su discapacidad no le permite girar su cuerpo, y depende de la asistencia de su familia para alcanzar una posición espacial diferente. Por tal motivo, se aprovecha la estructura de la silla reclinable para que gire gracias al diseño de un equipo de transmisión de movimiento a la silla giratoria. Se incorpora el control de este dispositivo al módulo de control electrónico del dispositivo exoesqueleto prototipo EMMRA-1, logrando asistencia al paciente en la ubicación espacial controlada por comandos de voz.

TABLA DE CONTENIDO

1. ADAPTACIONES PROTECCION ANTIESCARA..... 04
1.1 Descripción de la silla giratoria implementada..... 04
1.1.1 Figura 1. Silla de descanso filadelfia original..... 04
1.2 Características de la espuma antiescara..... 05
1.2.1 Figura 2. Vista superior de la espuma antiescara..... 05
1.2.2 Figura 3. Vista lateral de la espuma antiescara..... 05
1.3 Características de la tela protectora..... 06
1.3.3 Figura 4. Silla reclinable adaptada cara anterior..... 06
1.3.4 Figura 5. Silla reclinable adaptada cara lateral..... 07
2 ADAPTACIÓN DEL ESPACIO LATERAL..... 08
2.4.1 Figura 6. Espacio lateral de la silla filadelfia..... 08
2.4.2 Figura 7. Cajas en acrílico protectoras del módulo de control..... 09
2.4.3 Figura 8. Cubierta de acrílico caja lateral interna..... 09
2.4.4 Figura 9. Cubierta de acrílico caja lateral externa..... 10
2.4.5 Figura 10. Espacio aislado para la fuente de alimentación..... 10
3 Adaptaciones para las conexiones..... 11
3.1 Figura 11. Conectividad entre plataforma y Módulo de control..... 11
3.2 Figura 12. Conectividad interna en las cajas de acrílico..... 12
3.3 Figura 13. Conectividad cara lateral de la silla..... 12
3.4.1 Figura 14a. Conectividad de la silla al exoesqueleto..... 13
3.4.2 Figura 14b. Vaina en cuero textil..... 13
3.5 Figura 15. Conectividad entre el módulo de control y el computador..... 14
4. EQUIPO DE TRANSMISION DE MOVIMIENTO..... 15
4.1 Figura 16a. Diseño virtual equipo transmisión de movimiento superior..... 15
4.2 Figura 16b. Diseño virtual equipo transmisión de movimiento lateral..... 16
4.3 Figura 17. Equipo transmisión de movimiento Solido Base 16
4.4 Figura 18. Equipo transmisión de movimiento Solido Silla..... 17
4.5 Figura 19. Cremallera del equipo de transmisión..... 17
4.6 Figura 20. Plataforma del equipo de transmisión..... 18
4.7 Figura 21. Actuador Lineal del equipo de transmisión..... 18
4.8.1 Figura 22a. Módulo de control del equipo de transmisión de movimiento. 19
4.8.2 Figura 22a. Módulo de control del equipo de transmisión de movimiento. 19
4.9.1 Figura 23a. Platinas en Aluminio para descansar las piernas..... 20
4.9.2 Figura 23b. Platinas en Aluminio cubiertas..... 20
5. FOTOS DEL PRODUCTO TERMINADO. 21
5.1 Figura 24a. Adaptaciones silla reclinable vista anterior..... 21
5.2 Figura 24b. Adaptaciones silla reclinable vista lateral..... 22
5.3 Figura 25. Giro derecho de la silla con el paciente..... 23
5.3 Figura 25. Giro izquierdo de la silla con el paciente..... 24

1. ADAPTACIONES PROTECCIÓN ANTIESCARA

Los pacientes con discapacidad motriz asociada a LCMT, presentan dependiendo del nivel de lesión y el grado de compromiso medular, diferentes cuadros de cuadriplejia, lo cual implica que deban permanecer en cama o sentados todo el tiempo, pues el no tener control de los músculos de la columna les imposibilita ponerse de pie o permanecer sentados sin apoyo.

Para evitar que la piel del paciente se esfacele o se ulcere, es necesario realizar cambios de posición frecuentes cada 30 a 60 minutos mientras están acostados, además, se debe controlar el tiempo en que permanecen sentados. Por otra parte, se debe tener protección antiescara en las superficies de contacto de la cama y la silla con la piel del paciente, para ello, existen colchones y espumas antiescara, que evitan la formación de úlceras de presión.

1.1 Descripción de la silla giratoria implementada:

Para nuestro proyecto, utilizamos una silla giratoria reclinable modelo filadelfia, ya que presenta importantes ventajas: liviana, económica, giratoria, y tiene caras laterales libres.

1.1.1 En la gráfica siguiente observamos la silla de descanso filadelfia original



Figura 1. Silla de descanso filadelfia original

1.2 Características de la espuma antiescara:

Existen diversas formas de proteger la piel del paciente discapacitado, para ello en el mercado se consiguen materiales como espumas antiescara y protectores neumáticos. Se elige la espuma antiescara por ser más fácil de adaptar y ser mucho más económica que el mecanismo neumático.

1.2.1 En la gráfica siguiente se observa la espuma antiescara vista superior:

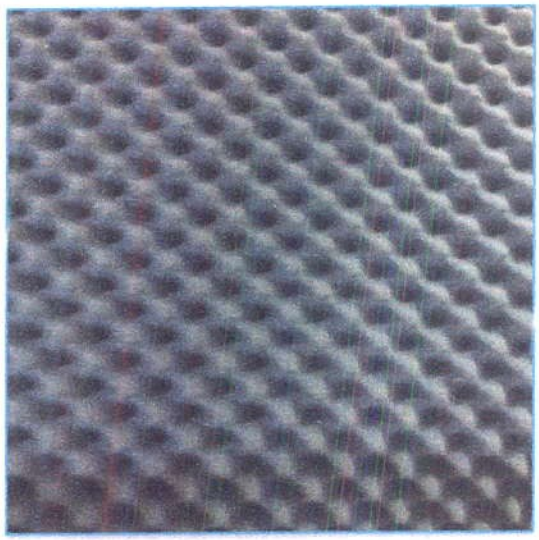


Figura 2. Espuma antiescara vista superior

1.2.2 En la gráfica siguiente se observa la espuma antiescara vista lateral:

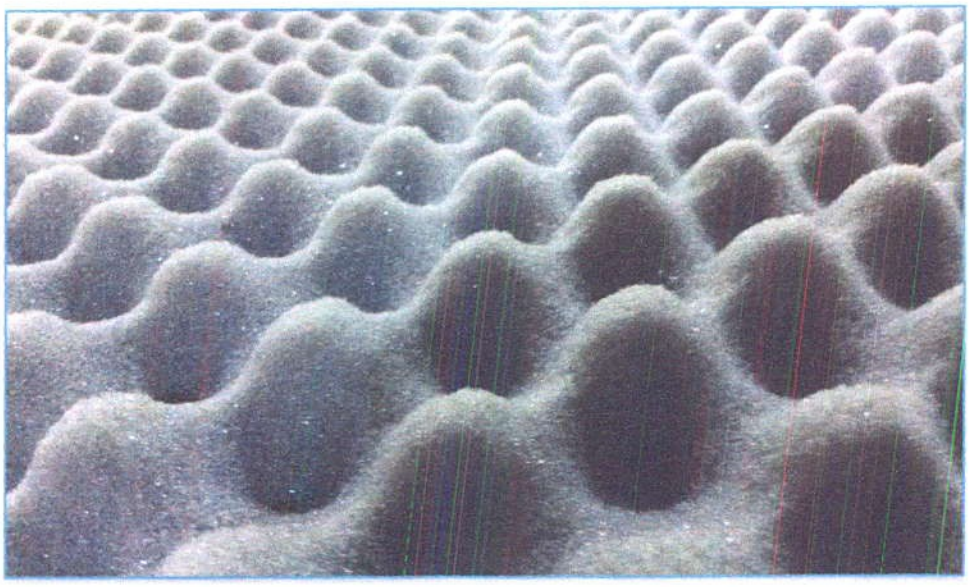


Figura 3. Espuma antiescara vista lateral

1.3 Características de la tela protectora:

Para dar protección a la piel del paciente, se cubre con tela prana color naranja en todas las superficies de contacto de la silla (espaldar, asiento, descasa brazos y descansa pies). Las ventajas de la tela prana son: ecorómica, lisa y facilmente lavable.

1.3.1 En la figura siguiente se observa la silla de descanso adaptada en su superficie por la espuma antiescara y la tela prana naranja.

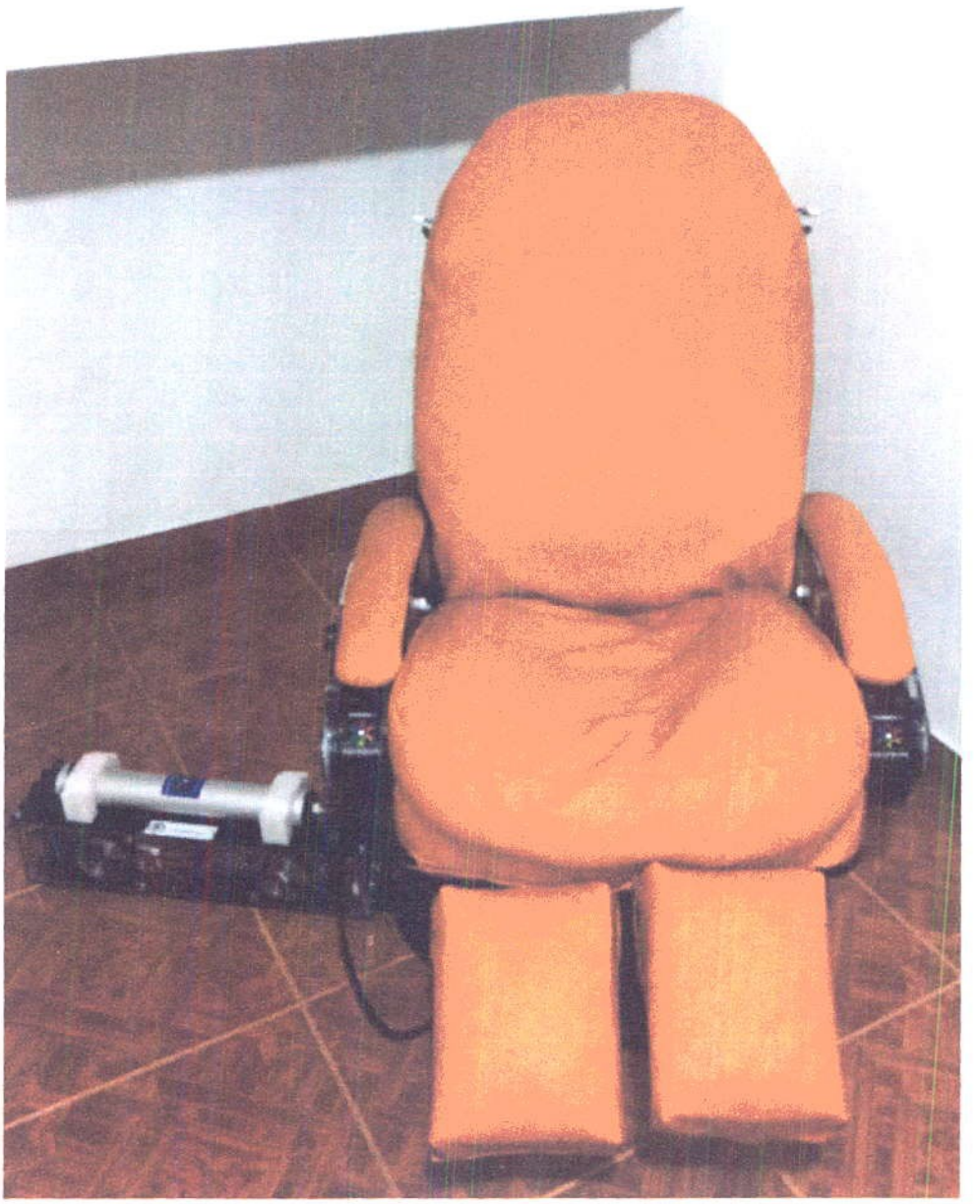


Figura 4. Silla reclinable adaptada cara anterior

1.3.2 En la figura siguiente se observa las caras laterales de la silla de descanso adaptada para el paciente discapacitado:



Figura 5. Silla reclinable adaptada cara anterior

2 ADAPTACIONES DE LA CARA LATERAL DE LA SILLA

El módulo de control del dispositivo, corresponde al montaje de una tarjeta electrónica EasyVR Shield 3.0 sobre una tarjeta electrónica Arduino Mega 2560. A partir de allí, salen las conexiones para el exoesqueleto mecatrónico, el equipo de transmisión de movimiento, los sensores de fuerza resistivos, los sensores de electromiografía y los indicadores led de alta luminosidad. Este módulo puede estar contenido en una caja de diferentes materiales con una dimensión de 15 x 10 cm.

Las fuentes de alimentación conmutadas con certificación para uso médico (switching power supply), también ocupan un espacio importante, pueden estar contenidas en una caja que cuente con ventilador de 12 V, cuyas dimensiones puedan ser 30 x 25 cm cada una.

Optimizando el espacio de las caras laterales de la silla filadelfia, se adaptaron unas cajas de acrìlico para contener las fuentes de alimentación conmutada, el módulo de control del dispositivo, los circuitos de interconexión, los parlantes y el micrófono del módulo de voz.

Cada área lateral de la silla giratoria, contiene una fuente conmutada MPT200A, este espacio esta protegido por aislamiento de la carcasa de la fuente a la silla, tiene unas compuertas laterales que permiten abrirse para hacer mantenimiento a la fuente de alimentación, en estos mismos espacios se encuentra un ventilador para evitar el sobrecalentamiento de las fuentes de alimentación.

En su parte anterior interna, se tiene un espacio cerrado que contiene el módulo de control del dispositivo, el cual se encuentra sobre una plataforma que facilita la interconexión de todos los elementos del sistema con el módulo de control electrónico (tarjetas electrónicas en fibra de vidrio).

2.1 En la siguiente figura, se observa el espacio lateral de la silla de descanso filadelfia que será adaptado al dispositivo.

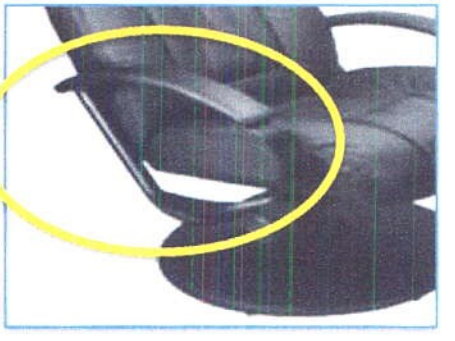


Figura 6. Área lateral de la silla filadelfia

2.2 En las siguiente figura se observa el diseño de las cajas protectoras del módulo de control del dispositivo, las cuales están fabricadas en acrílico 5 mm, se adaptan perfectamente a los espacios laterales de la silla filadelfia.

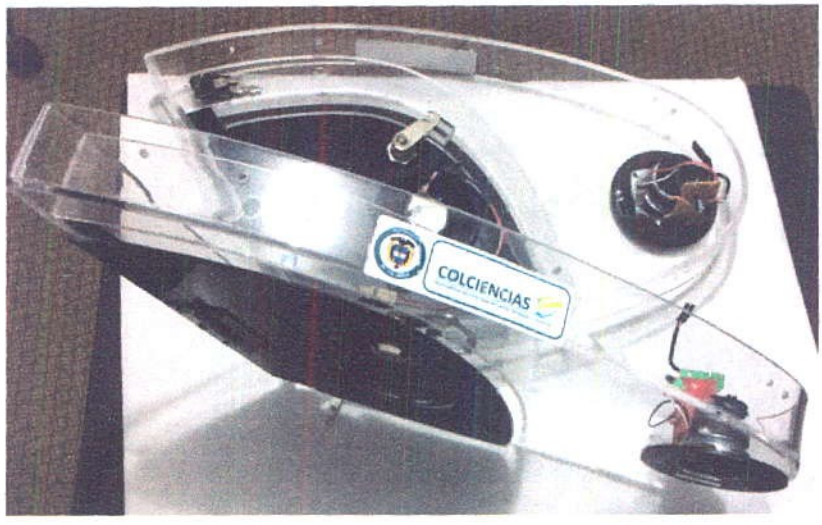


Figura 7. Cajas en acrílico protectoras del módulo de control

2.3 En la siguiente figura se observa la caja interna de acrílico insertada en el espacio lateral de la silla, con el espacio aislado para la fuente de alimentación (recuadro verde) y el espacio para el módulo de control (recuadro rojo).

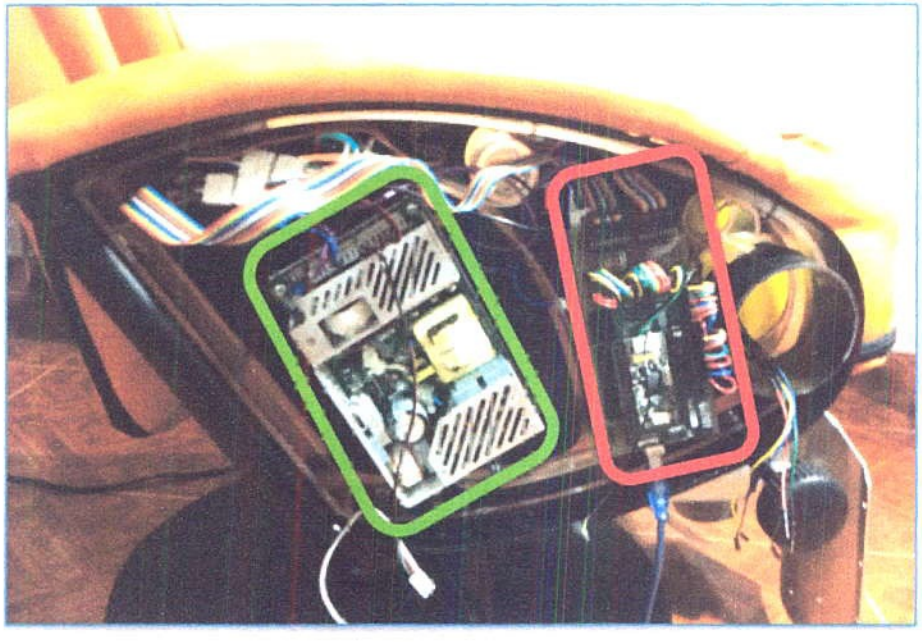


Figura 8. Cubierta de acrílico caja lateral interna

2.4 En la siguiente figura se observa la caja externa de acrílico insertada en el espacio lateral de la silla, con la tapa lateral (círculo verde), el micrófono (círculo blanco), el parlante (círculo amarillo) y el módulo de control del dispositivo asegurado internamente (recuadro rojo).

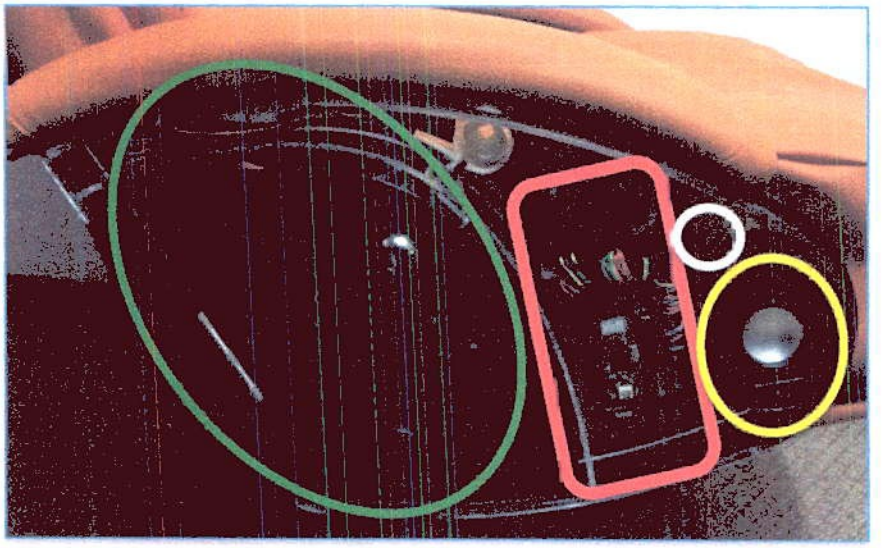


Figura 9. Cubierta de acrílico caja lateral externa

2.5 En la siguiente figura se observa el espacio aislado para la fuente de alimentación con tapa y seguridad (circulo verde), el ventilador de 12 Voltios (circulo amarillo) y la fuente de alimentación (recuadro rojo)

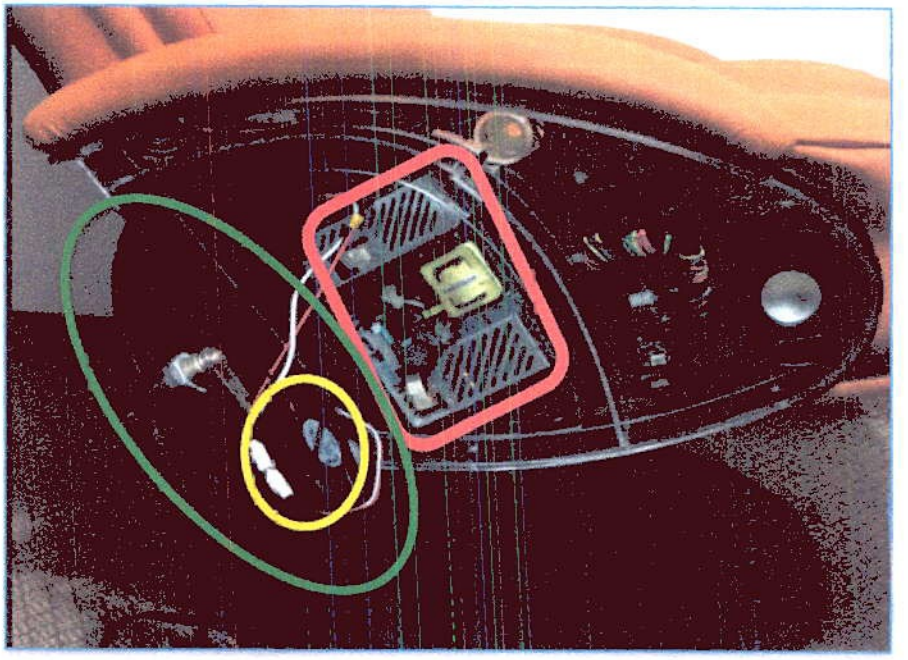


Figura 10. Espacio aislado para la fuente de alimentación

3 Adaptación para las conexiones:

La conectividad entre el módulo de contro del dispositivo y el exoesqueléto Prototipo EMMRA-1, involucra múltiples cables tipo ribon, vehicular y conectores electrónicos. Todos estos cables deben cumplir una trayectoria específica: primero internamente en la conectividad entre el módulo de control y la plataforma electrónica; segundo entre la plataforma y la silla giratoria; tercero entre la silla giratoria y la salida hacia los hombros del paciente; cuarto desde la salida de la silla hasta el exoesqueleto de mano y muñeca; quinto entre el módulo de control y el computador (para los ajustes de programación)

3.1 En la siguiente figura se observa la conectividad entre plataforma y módulo de control, se realiza con cables de conexión de circuitos macho hembra, el diseño está configurado de esta forma para ilustrar a los interesados (makers, estudiantes de ingeniería pre y posgrado, diseñadores, etc.).



Figura 11. Conectividad entre plataforma y Módulo de control

3.2 En la siguiente figura se observa la conectividad interna en las cajas de acrílico protectoras del módulo de control y la fuente de alimentación, esta conectividad incolucra las circuitería que se dirije desde el módulo de control hasta la silla (recuadro verce) y desde el módulo de control hacia la fuente de alimentación (recuadro rojo). Está conformada por cables ribon y cable vehicular.

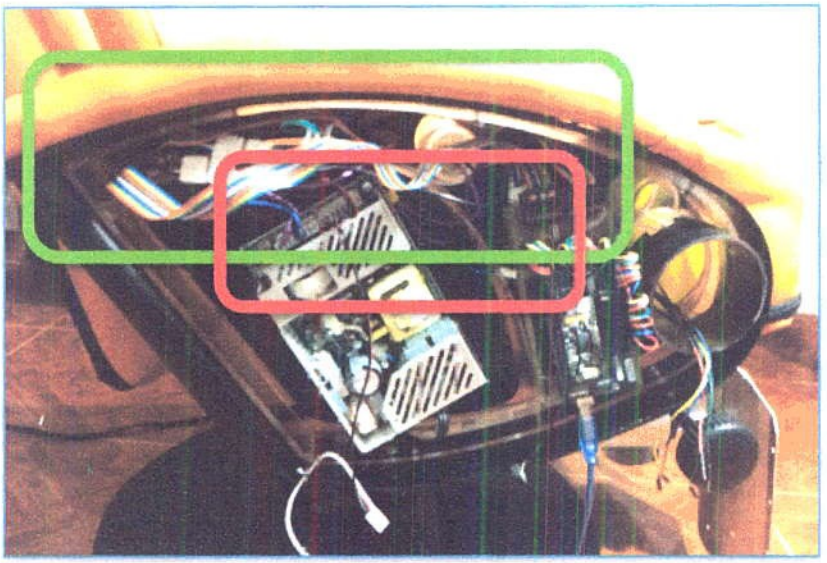


Figura 12. Conectividad interna en las cajas de acrílico

3.3 En la siguiente figura se observa la conectividad por la cara lateral de la silla reclinable (recuadro rojo), desde el módulo de control hasta la salida al nivel de la altura de los hombros del paciente (círculo amarillo).

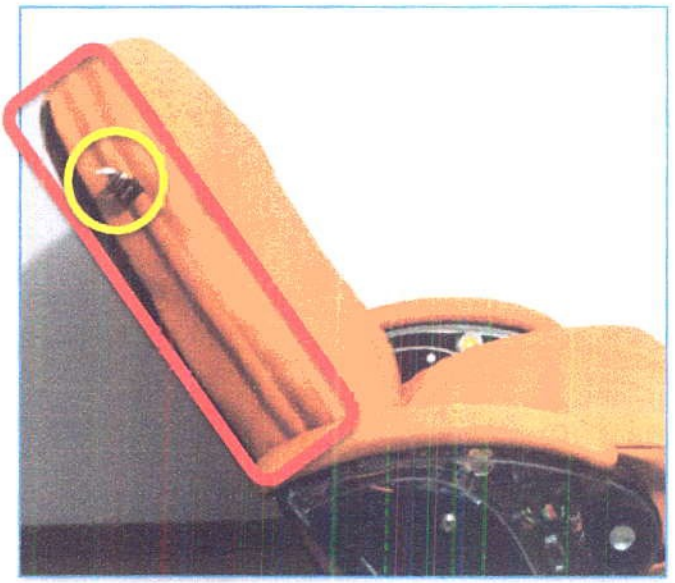


Figura 13. Conectividad cara lateral de la silla

3.4.1 En la siguiente figura observamos la conectividad entre la silla y el exoesqueleto mecatrónico prototipo EMMRA-1, todos los cables ribon y vehiculares, están cubiertos por una vaina en cuero textil, a la altura de los antebrazos se superpone el sensor de electromiografía (recuadro rojo) y los electrodos de electromiografía (círculo amarillo) la protección antiestática (círculo naranja). Además, la conexión entre la silla y la vaina de cuero, y entre el exoesqueleto prototipo y la vaina de cuero, corresponde a conectores GP y conectores de seguridad (círculo blanco)

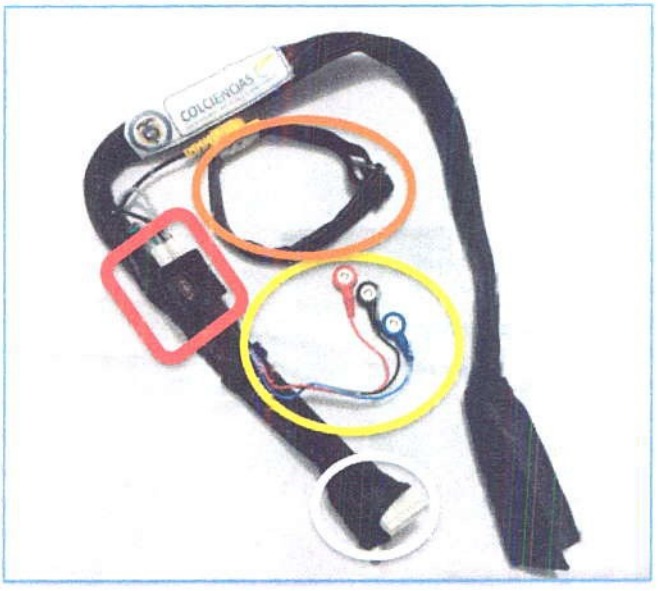


Figura 14a. Vaina de cuero conectora

3.4.2 En la siguiente figura observamos como se recresenta la conectividad entre la silla y el exoesqueleto, aplicado al paciente.

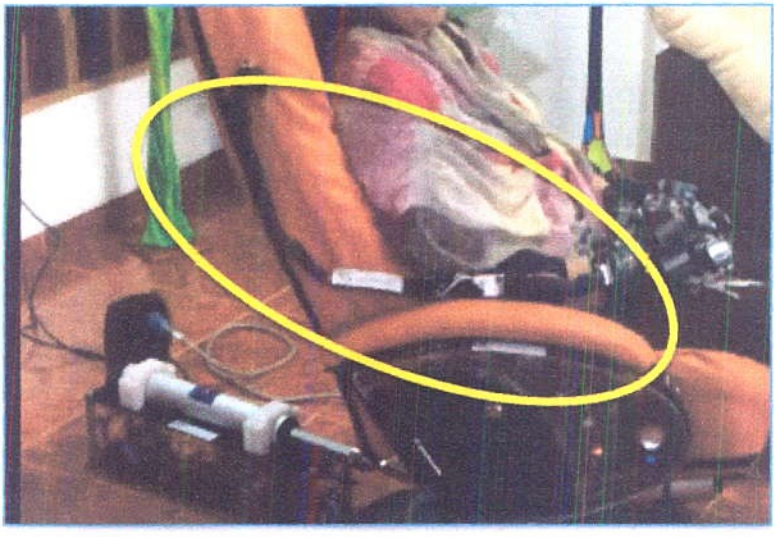


Figura 14b. Conectividad de la silla al exoesqueleto

3.5 En la siguiente figura observamos como es la conectividad entre el módulo de control y el computador. Esta conectividad corresponde a un cable USB que está conectado a la tarjeta electrónica Arduino Mega 2560, disponible en la caja de seguridad para ajustar la programación de sensores, servomotores y controladores.

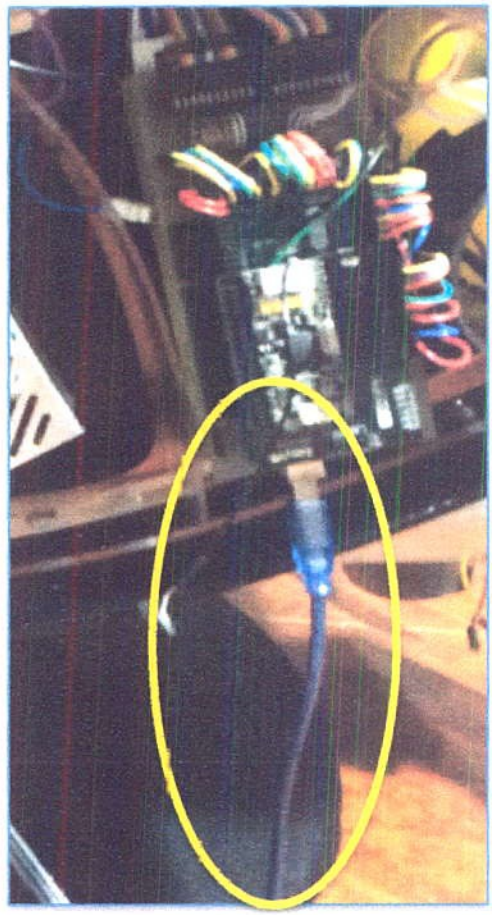


Figura 15. Conectividad entre el módulo de control y el computador

4. EQUIPO DE TRANSMISIÓN DE MOVIMIENTO

Para mejorar las prestaciones del prototipo EMMRA-1, se diseño y fabricó un equipo de transmisión de movimiento a la silla giratoria de descanso filadelfia. En la base de la silla se instaló una rueda dentada libre e integrada a una cremallera dentada, está última acoplada a un actuador lineal eléctrico con carrera de 20 cm y potencia de 100 Newton.

El equipo de transmisión de movimiento a la silla giratoria, permite al paciente con discapacidad motriz por LCMT, ubicarse en su ambiente doméstico a través de comandos de voz, que permiten un movimiento de la silla de izquierda a derecha y de derecha a izquierda en 180 grados.

En la base posterior de la silla se soldó una platina metálica como esqueleto al actuador lineal, en su base inferior se ubico la fuente de alimentación del actuador (NXT400 10-04), y el controlador del actuador lineal (Tarjeta electrónica Arduino Uno R3 acoplada a un driver de alta potencia para arduino Modelo VNH5019).

Igualmente se acoplaron a la base de la silla, dos actuadores lineales mecánicos para permitir la extensión y la flexión de las piernas, cuando el espaldar de la silla se lleve hacia atrás o hacia delante.

4.1 En la siguiente figura se observa el diseño del equipo de transmisión de movimiento a la silla giratoria, constituido por una rueda dentada y una cremallera metálica, además de los actuadores lineales ubicados para el propósito indicado.

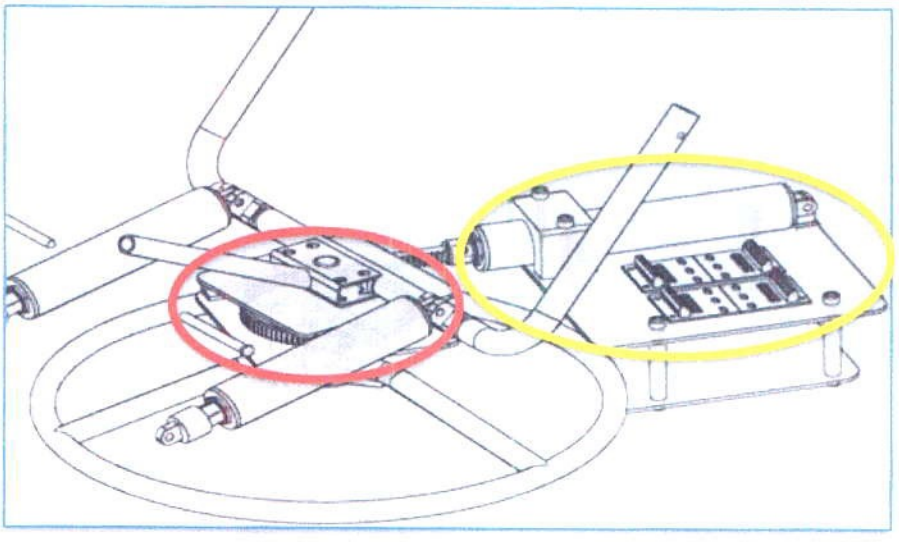


Figura 16a. Diseño del equipo de transmisión de movimiento plano superior

4.2 En la siguiente figura, observamos el diseño del equipo de transmisión de movimiento cara lateral, donde se detalla mejor la cremallera (círculo rojo) y rueda dentada metálica acoplada al actuador lineal eléctrico (círculo amarillo).

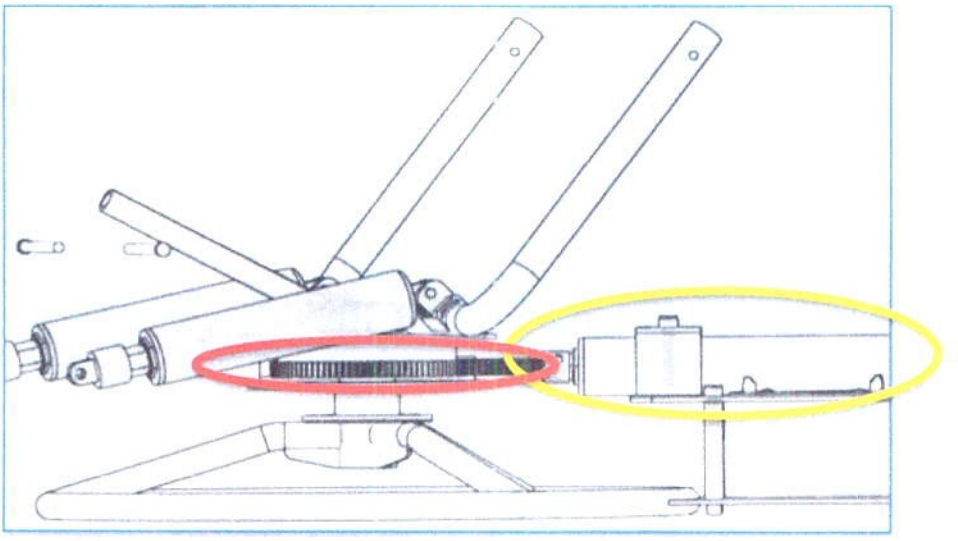


Figura 16b. Diseño del equipo de transmisión de movimiento plano superior

4.3 En la siguiente figura observamos el diseño del equipo de transmisión de movimiento en la vista superior.

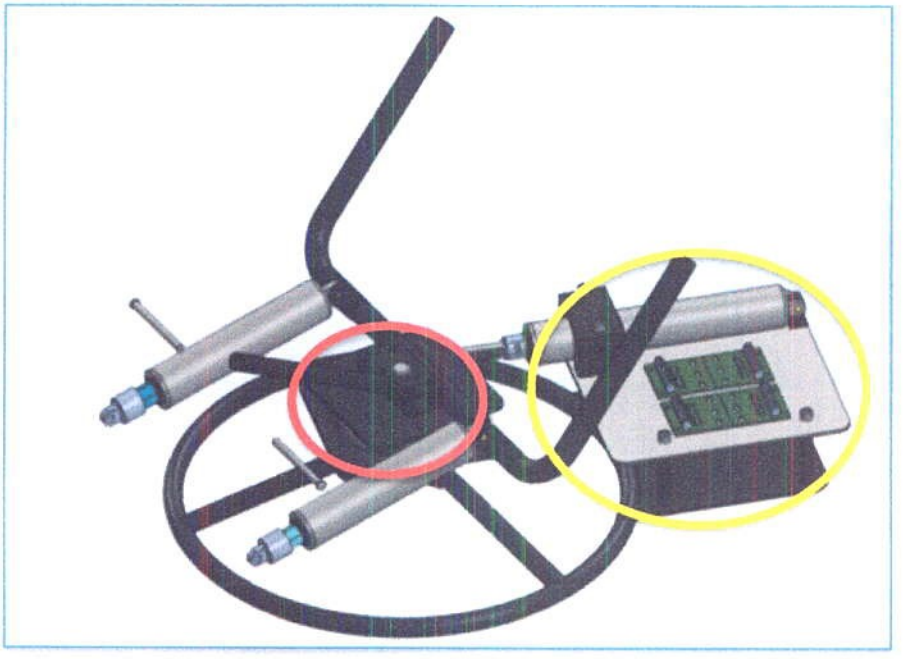


Figura 17. Diseño Solido del equipo de transmisión de movimiento plano superior

4.4 En la siguiente figura observamos el diseño del equipo de transmisión de movimiento acoplado a la silla giratoria. Los actuadores lineales eléctricos en la base de la silla permiten el desplazamiento mecánico de los descansapies.

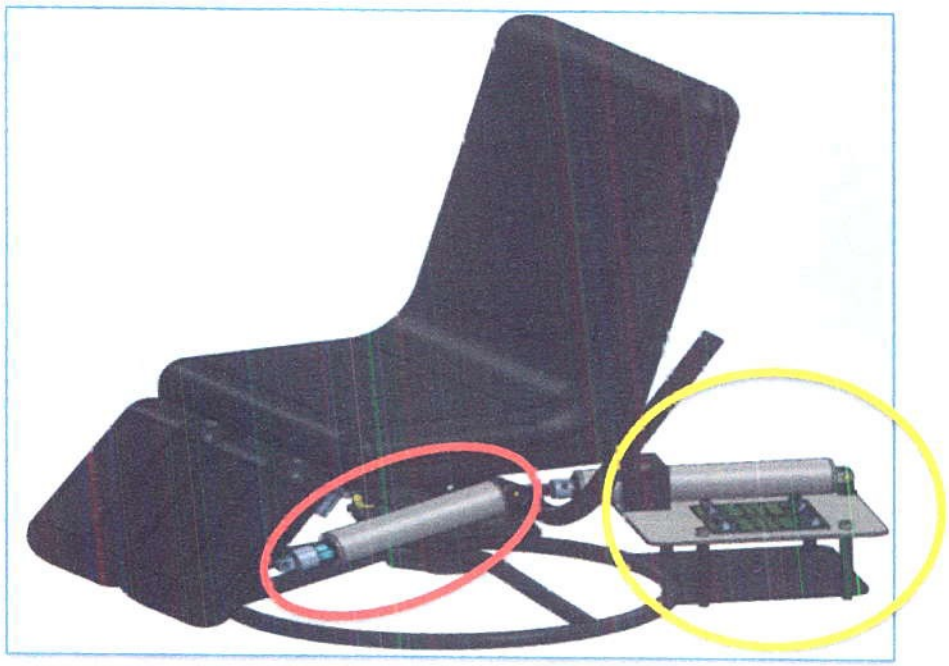


Figura 18. Diseño Solido del equipo de transmisión de movimiento acoplado a la silla

4.5 En la siguiente figura observamos la cremallera y rueda dentada ubicadas en la base de la silla giratoria.

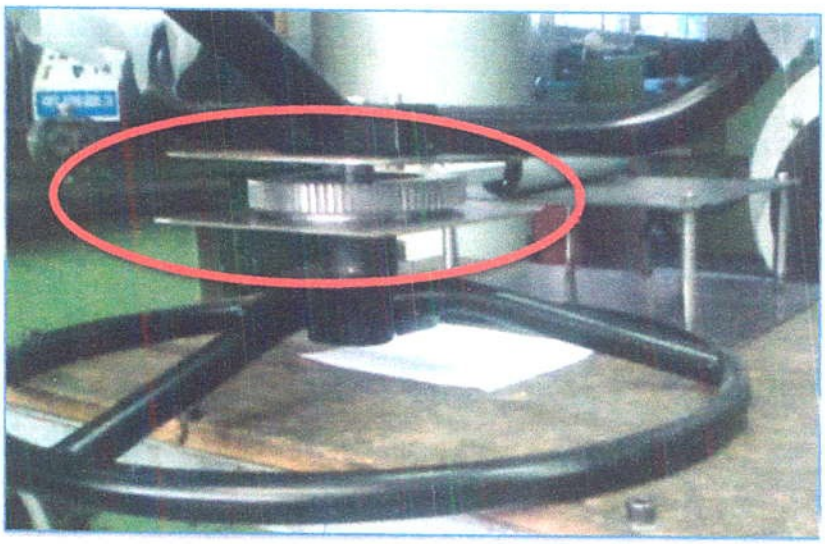


Figura 19. Cremallera metálica en la base de la silla

4.6 En la siguiente figura se observa la plataforma diseñada para el actuador lineal acoplado a la cremallera metálica que proporciona el giro a la silla.

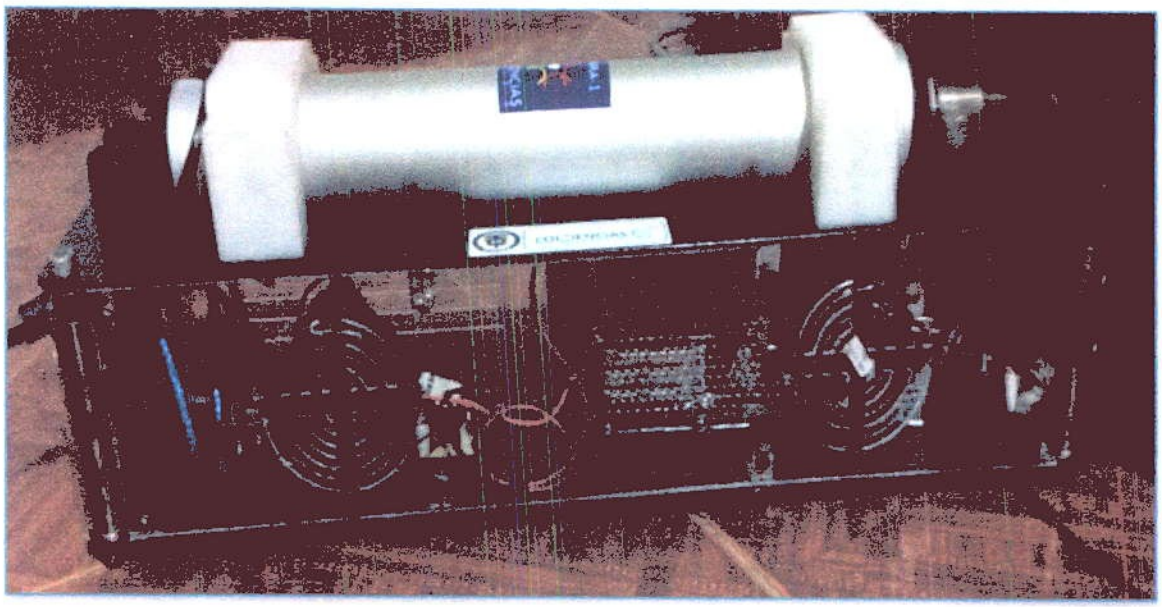


Figura 20. Plataforma metálica que soporta el actuador lineal

4.7 En la figura se observa el actuador lineal ajustado a la plataforma por abrazaderas plásticas.

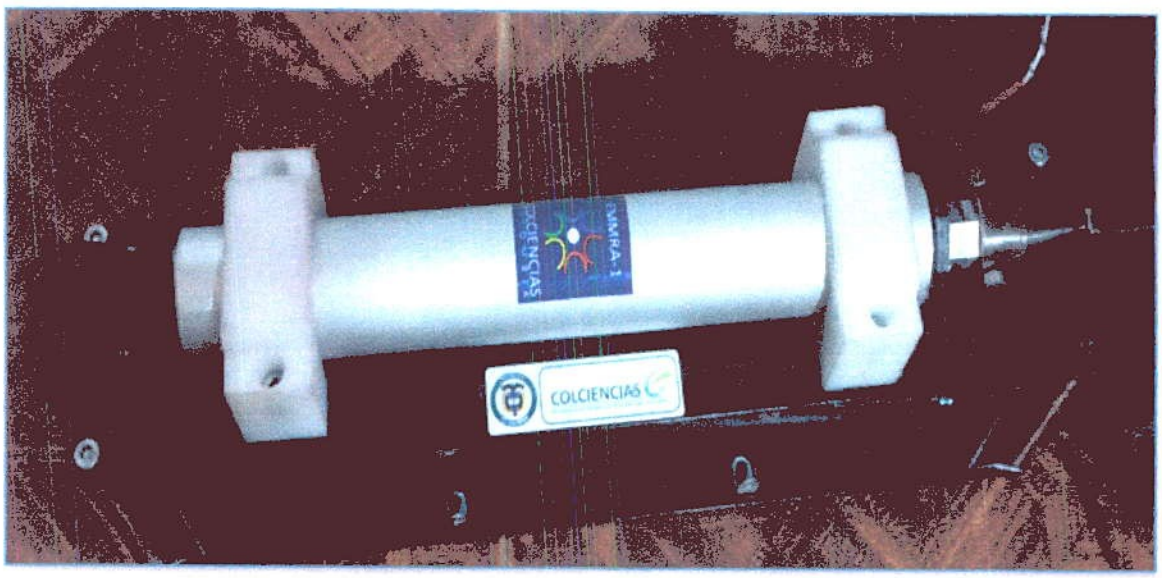


Figura 21. Actuador lineal acoplado a la plataforma metálica

4.8.1 En la siguiente figura observamos el módulo de control del equipo de transmisión de movimiento, constituido por una tarjeta electrónica tipo driver de alta potencia arduino VNH5019, insertado en una tarjeta electrónica Arduino Uno R3.

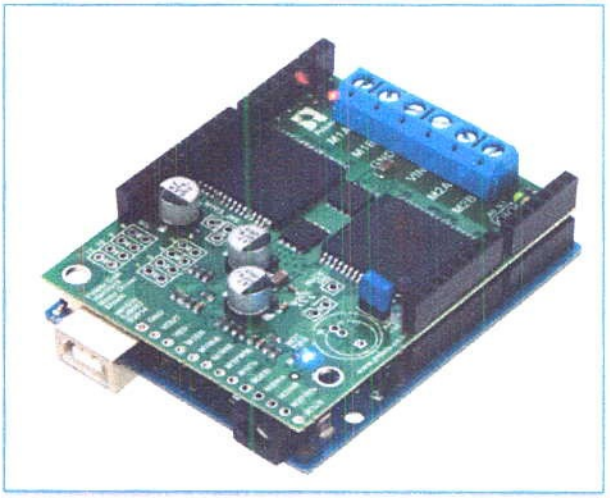


Figura 22a. Actuador lineal acoplado a la plataforma metálica

4.8.2 En la siguiente figura observamos el módulo de control integrado a la plataforma metálica y las tapas en acrílico que portegen la estructura, cuenta con fuente de alimentación NXT400 1004 (recuadro verde), Módulo de control (círculo rojo) y 2 ventiladores de 12V circulo amarillo.

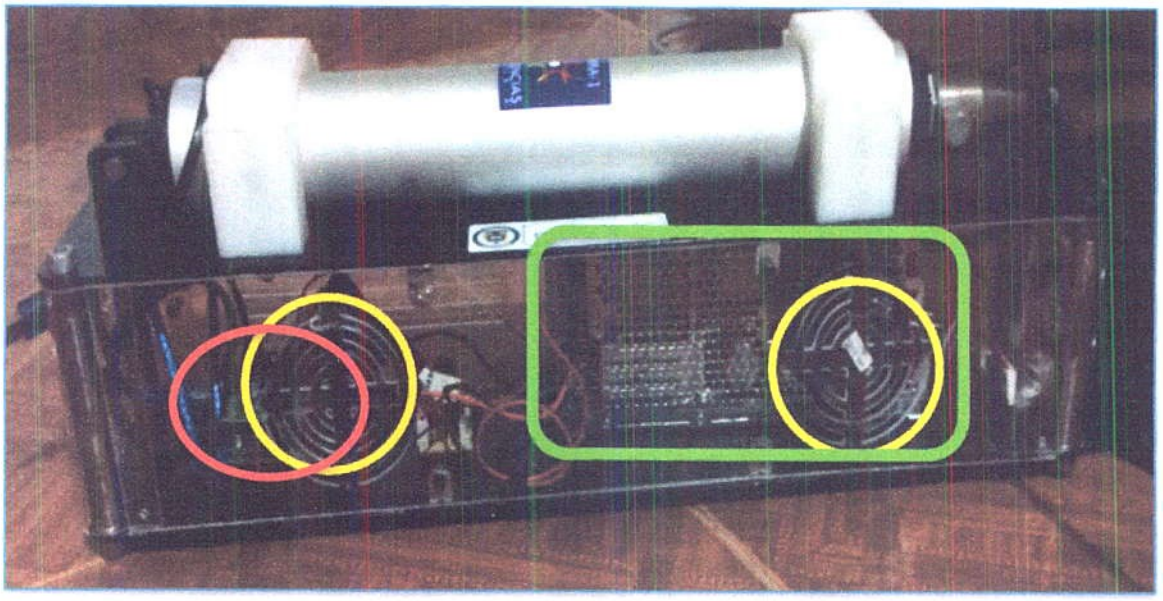


Figura 22b. Módulo de control del actuador lineal

4.9.1 En la siguiente figura observamos las Platinas en Aluminio que sirven de descansapiernas, permiten movimientos de flexión y estensión de las piernas de forma mecánica, cuando se inclina o reclina la silla g ratoria.

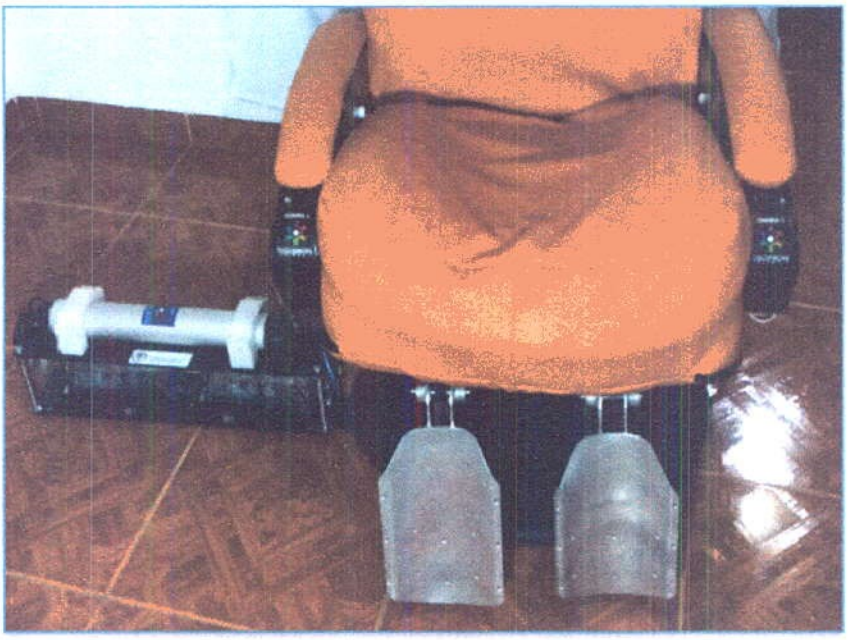


Figura 23a. Platinas en Aluminio para descansar las piernas

4.9.2 En la siguiente figura observamos las Platinas en Aluminio cubiertas por una espuma t pc cusa y protegidas por la tela de neopreno naranja.

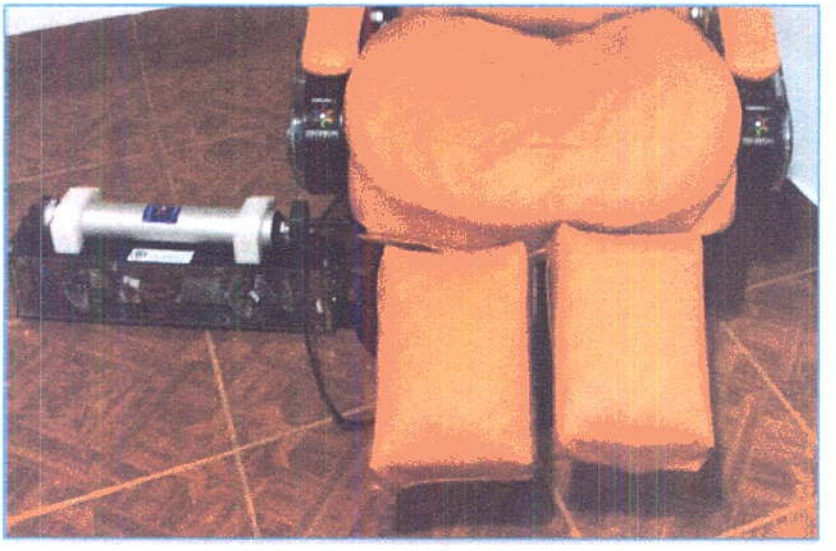


Figura 23b. Platinas en Aluminio cubiertas

5 FOTOS DEL PRODUCTO TERMINADO

5.1 En la siguiente figura observamos la silla giratoria adaptada al proyecto Prototipo EMMRA-1 (cara anterior)



Figura 24a. Acaptaciones silla reclinable vista anterior

5.2 En la siguiente figura observamos la silla giratoria adaptada al proyecto Prototipo EMMRA-1 (cara lateral)



Figura 24b Adaptaciones silla reclinable vista ateral

5.3 En la siguiente figura observamos la silla giratoria adaptada al proyecto Prototipo EMMRA-1 utilizada por la paciente (giro derecho)



Figura 25a. Giro derecho de la silla con el paciente

5.3 En la siguiente figura observamos la silla g ratoria adaptada al proyecto Prototipo EMMRA-1 utilizada por a paciente (giro izquierdo)



Figura 25b. Giro izquierdo de la silla con el paciente

INFORME FINAL DEL PROYECTO
EXOESQUELETO DE MANO Y MUÑECA CON APLICACIONES DE REHABILITACION Y ASISTENCIA (PROTOTIPO EMMRA-I)

CAPITULO V
CODIGO DE PROGRAMACIÓN DEL MODULO DE CONTROL DEL PROTOTIPO EMMRA-1

INVESTIGADORES:

JUAN DAVID MORENO ARANGO
JULIAN ALBERTO MORENO ARANGO

ASESORES:

VICTOR MAURICIO ACEVEDO CORREA
ALEXANDER RAMOS RODRIGUEZ

COLABORADORES

Manizales, 01 de Noviembre de 2015

INTRODUCCIÓN

El código abierto Arduino Software (IDE) hace que sea muy fácil escribir código de programación y subirlo a la placa de Arduino Uno R3. Se ejecuta en Windows, Mac OS X y Linux. El entorno está escrito en Java y está basado en el procesamiento y software de código abierto. Este software se puede utilizar con cualquier placa Arduino. Se descarga gratuitamente de la página del fabricante: http://www.arduino.cc

El EasyVR Commander genera un código de programación con los comandos SD creados en la lista de grupos, este código es un programa fuente estándar que debe ser abierto desde el Software Arduino, para asignar acciones específicas al microcontrolador ATMega 16 U2, una vez se reconozcan los comandos en la EasyVR Shield 2.0.

El código fuente para programar el MRV, tiene una estructura fundamentada en lenguaje de programación en C, con subprogramas propios del Software Arduino. Se incluyen las librerías para la interface del **Software Arduino** con las tarjetas electrónicas **Arduino Uno**, **Arduino Uno R3, Arduino Mega y Arduino Leonardo.**

Para elaborar el código de programación, se deben incluir las librerías para la interface del Software Arduino con la tarjeta electrónica **EasyVR Shield 3.0,** además los subprogramas para el manejo de Servomotores.

Las principal ventaja de programación en Software Arduino para el proyecto Prototipo EMMRA-1, se debe a que las plataformas de programación del módulo de reconocimmientc de voz son compatibles cor la plataforma arduino. Por esta razón, se utilizaron todos los equipos y sensores compatibles con la plataforma arduino, dado que permite la integralidad de programación del sistema.

El hardware adquirido que es compatible con Software Arduino es: Tarjeta electrónica Arduino Mega 2560, Tarjeta electrónica EasyVR Shield 3.0, Tarjeta Electrónica Arduino Uno R3, Controlador de Actuadores Lineales, Sensores de Señal Muscular.

El Módulo de Reconocimiento de Voz compuestc por la unión de las tarjetas electrónicas EasyVR Shield 3.0 y Arduino Mega 2560, requiere adaptaciones en el Software código fuente generado por el Software EasyVR Commander, para integrarlo al prototipo EMMRA-1.

TABLA DE CONTENIDO

1. Definición de Librerías..... 04
2. Definición de Controladores de Servomotores..... 04
3. Definición de Grupos..... 07
4. Void setup..... 08
5. Void loop..... 10
6. Inicialización de Variables..... 12
7. Void Action CICLO INFINITO..... 14
7.1 Subrutina para inicializar el exoesqueleto..... 14
7.2 Subrutina para direccionamiento entre grupos..... 15
7.3 Rutina para abrir las manos..... 16
7.4 Rutina para cerrar las manos..... 21
7.5 Rutina para agarrar objetos..... 30
7.6 Rutina para soltar objetos..... 34
7.7 Rutina para reiniciar el programa..... 39
7.8 Rutina para el módulo de EMG..... 40
7.9 Rutina para manipular objetos de forma específica..... 84
8. Controlador del Equipo de Transmisión de Movimiento..... 133
8. Flujogramas del código de programación..... 133
8.1 Flujograma para inicializar el programa..... 133
8.2 Flujograma para comandos del grupo 1..... 134
8.3 Flujograma para comandos del grupo 2..... 134
8.4 Flujograma para comandos del grupo 3..... 135
8.5 Flujograma para comandos del grupo 4..... 135
8.6 Flujograma para comandos del grupo 5..... 136
8.7 Flujograma para comandos del grupo 6..... 136
8.8 Flujograma para comandos del grupo 7..... 137
8.8 Flujograma para comandos del grupo 8..... 137
9. Código de programación del equipo de transmisión de movimiento..... 138

1. Definición de Librerias:

A continuación describiremos con detalle el código fuente generado por el EasyVR Commander **(letra negra)** y el nuevo software adicionado **(letra azul)**, los comentarios de explicación de software después del simbolo //

El software Arduino incluye la librería para la interface del Software Arduino con la tarjeta electrónica **EasyVR Shield 3.0.** Contiene las librerías **Arduino.h, SoftwareSerial.h y NewSoftSerial.h**, necesarias para el control de la tarjeta Arduino Mega 2560.

```
// PRIMERA PARTE (LIBRERIAS)
//*************************************************************************
#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"              // Librería específica de Arduino.
#include "SoftwareSerial.h"       // Nuevas librerias para Mega.
 SoftwareSerial port(12,13);
#else // Arduino 0022 - use modified NewSoftSerial
 #include "WProgram.h"
 #include "NewSoftSerial.h"
 NewSoftSerial port(12,13);
#endif

//_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
#include "EasyVR.h"               // Librerìas MÛdulo de Voz
EasyVR easyvr(port);
#include <Servo.h>                // Librería Control Servomotores
//_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

//*************************************************************************
```

NOTA: Requerimiento de Librerías: Adicionamos al código fuente la librería para el manejo de Servomotores, propia del Software Arduino

```
#include <Servo.h>
```

```
//*****************************************************************************
//CODIGO DE PROGRAMACIÓN NUEVO:
// Definición de controladores de servos
Servo servo1D;   //SERVO1= ROTACIÓN I DEDO
Servo servo2D;   //SERVO2= FLEXION Y EXTENSIÓN DE LA FALANGE
                 //            PROXIMAL DEL I DEDO MANO DER.
Servo servo3D;   //SERVO3= FLEXIÓN Y EXTENSIÓN FALANGE DISTAL
                 //            DEL I DEDO MANO DER.
Servo servo4D;   // SERVO4= FLEXIÓN Y EXTENSIÓN DE FALANGE
                 //            PROXIMAL DE II Y III DEDOS MANO DER
Servo servo5D;   //SERVO5= FLEXIÓN Y EXTENSIÓN DE FALANGES MEDIAS
                 //            DE II Y III DEDOS DE MANO DER.
Servo servo6D;   //SERVO6= FLEXIÓN Y EXTENSIÓN DE FALANGES
                 //            PROXIMALES DE IV Y V DEDOS MANO DER.
Servo servo7D;   //SERVO7= FLEXIÓN Y EXTENSIÓN DE FALANGES MEDIAS
                 //            DE IV Y V DEDOS MANO DER.
Servo servo8D;   //SERVO8= FLEXIÓN Y EXTENSIÓN DE MUÑECA DER.
Servo servo1I;   //SERVO1= IGUAL QUE SERVO8D LADO IZQ.
Servo servo2I;   //SERVO2= IGUAL QUE SERVO7D LADO IZQ.
Servo servo3I;   //SERVO3= IGUAL QUE SERVO6D LADO IZQ.
Servo servo4I;   //SERVO4= IGUAL QUE SERVO5D LADO IZQ.
Servo servo5I;   //SERVO5= IGUAL QUE SERVO4D LADO IZQ.
Servo servo6I;   //SERVO6= IGUAL QUE SERVO3D LADO IZQ.
Servo servo7I;   //SERVO7= IGUAL QUE SERVO2D LADO IZQ.
Servo servo8I;   //SERVO8= IGUAL QUE SERVO1D LADO IZQ.
//*****************************************************************************
```

NOTA: Identificación del número de servos a utilizar: Parametrizamos 8 servomotores para los movimientos del prototipo EMMRA-1, Cada servomotor definirá un movimiento en la mano y muñeca del paciente:

```
Servo servo1D;
```

```
// SEGUNDA PARTE (GRUPOS  DEL MODULO DE VOZ)
//*************************************************************************
//Groups and Commands
enum Groups
{
 GROUP_0  = 0,
 GROUP_1  = 1,
 GROUP_2  = 2,
 GROUP_3  = 3,
 GROUP_4  = 4,
 GROUP_5  = 5,
 GROUP_6  = 6,
 GROUP_7  = 7,
 GROUP_8  = 8,
 GROUP_9  = 9,
};
//*************************************************************************
```

NOTA: La estructura del programa no se modifica en esta parte del código fuente, en esta parte del programa se enumeran los grupos creados en el Software EasyVR Commander.

```
// TERCERA PARTE (COMANDOS MODULO DE VOZ)
//*************************************************************************
enum Group0
{
 G0_EMMRA_UNO = 0,
};
enum Group1
{
 G1_ASISTENCIA = 0,
 G1_REHABILITACION = 1,
 G1_SILLA = 2,
```

```
};
enum Group2
{
 G2_MANO_DERECHA = 0,
 G2_MANO_IZQUIERDA = 1,
 G2_AMBAS_MANOS = 2,
 G2_REINICIAR = 3,
};
enum Group3
{
 G3_ABRIR = 0,
 G3_CERRAR = 1,
 G3_AGARRAR = 2,
 G3_SOLTAR = 3,
 G3_REINICIAR = 4,
 G3_AUDIO = 5,
};
enum Group4
{
 G4_FUERTE = 0,
 G4_SUAVE = 1,
};
enum Group5
{
 G5_CINCO = 0,
 G5_DIEZ = 1,
 G5_QUINCE = 2,
 G5_REINICIAR = 3,
};
enum Group6
{
 G6_LAPIZ = 0,
 G6_CUCHARA = 1,
 G6_VASO = 2,
 G6_POCILLO = 3,
 G6_PINZA = 4,
 G6_REINICIAR = 5,
};
enum Group7
{
 G7_ACTUAR = 0,
```

```
  G7_PASIVA = 1,
};
enum Group8
{
  G8_DERECHA = 0,
  G8_IZQUIERDA = 1,
  G8_DETENER = 2,
  G8_CONTINUAR = 3,
};
enum Group9
{
  G9_ SOLTAR_OBJETO = 0,
};

//*************************************************************************
```

NOTA: La estructura del programa no se modifica en esta parte del código fuente, en esta parte del programa se describe con detalle los comandos de voz creados en el EasyVR Commander.

```
// CUARTA PARTE (VOID SETUP)
//*************************************************************************
EasyVRBridge bridge;
int8_t group, idx;
//*************************************************************************
//CODIGO DE PROGRAMACIÓN NUEVO:
// LED INDICADORES DEL PROCESO DE RECONOCIMIENTO DE VOZ
int led1 = 3;   // Led Giro de Silla, activado por el pin 3 del Arduino Mega
int led2 = 5;   // Led Giro de Silla, activado por el pin 5 del Arduino Mega
int led1D = 50;   // Led Superior Verde Derecho del MRV
int led2D = 52;   // Led Inferior Rojo Derecho del MRV
int led1I = 34;   // Led Superior Verde Izquierdo del MRV
int led2I = 36;   // Led Inferior Rojo Izquierdo del MRV
//*************************************************************************
// VARIABLE PARA ELEGIR LA VOZ EN LA TABLA DE SONIDOS
int voz = 1;
//*************************************************************************
void setup()
```

```
{
 // bridge mode?
 if (bridge.check())
 {
  cli();
  bridge.loop(0, 1, 12, 13);
 }
 // run normally
 Serial.begin(9600);
 port.begin(9600);
 if (!easyvr.detect())
 {

  Serial.println("EasyVR not detected!");
  for (;;);
 }
 easyvr.setPinOutput(EasyVR::IO1, LOW);
 //*************************************************************************
 //CODIGO DE PROGRAMACIÓN NUEVO:
 // Activamos indicadores del proceso de reconocimiento de voz
 digitalWrite(led1D, LOW);
 digitalWrite(led2D, HIGH);
 digitalWrite(led1I, LOW);
 digitalWrite(led2I, HIGH);
 pinMode (leg1, OUTPUT);
 pinMode (leg2, OUTPUT);
 //digitalWrite(leg1, LOW);
 //digitalWrite(leg2, LOW);
//*************************************************************************
 Serial.println("EasyVR detected!");
 easyvr.setTimeout(5);
 easyvr.setLanguage(4);
 group = EasyVR::TRIGGER; //<-- start group (customize)
 //*************************************************************************
 //CODIGO DE PROGRAMACIÓN NUEVO:
 //ASOCIACION DE SERVOS A LAS SALIDAS DE MICRO
 servo1D.attach(47); // Salida del Servo1 en el pin 47
 servo2D.attach(45); // Salida del Servo2 en el pin 45
 servo3D.attach(43); // Salida del Servo3 en el pin 43
 servo4D.attach(41); // Salida del Servo4 en el pin 41
 servo5D.attach(40); // Salida del Servo5 en el pin 40
```

```
 servo6D.attach(42); // Salida del Servo6 en el pin 42
 servo7D.attach(44); // Salida del Servo7 en el pin 44
 servo8D.attach(46); // Salida del Servo8 en el pin 46
 servo1I.attach(30); // Salida del Servo1 en el pin 30
 servo2I.attach(28); // Salida del Servo2 en el pin 28
 servo3I.attach(26); // Salida del Servo3 en el pin 26
 servo4I.attach(24); // Salida del Servo4 en el pin 24
 servo5I.attach(25); // Salida del Servo5 en el pin 25
 servo6I.attach(27); // Salida del Servo6 en el pin 27
 servo7I.attach(29); // Salida del Servo7 en el pin 29
 servo8I.attach(31); // Salida del Servo8 en el pin 31
}
//*************************************************************************
```

NOTA: La estructura del codigo fuente se modifica:

- Adicionando los indicadores Led para que el usuario pronuncie comandoz de voz cuando esta en verde y para que el módulo procese información cuando está en rojo.
- Adicionando la variable para intercambiar las voces de la tabla de sonido.
- Activando los indicadores Led.
- Asociando los Servomotores a las salidas digitales de la tarjeta Arduino Mega 2560.

```
// QUINTA PARTE (VOID LOOP)
// *************************************************************************
void action();
void loop()
{
 easyvr.setPinOutput(EasyVR::IO1, HIGH); // LED on (listening)
 Serial.print("Say a command in Group ");

// *************************************************************************
 // Iniciamos mÛdulo de conversaciÛn con usuario
 easyvr.playSound(voz, EasyVR::VOL_FULL);
 // *************************************************************************
 // Activamos indicadores del proceso de reconocimiento de voz
 digitalWrite(led1D, HIGH);
```

```
digitalWrite(led2D, LOW);
digitalWrite(led1I, HIGH);
digitalWrite(led2I, LOW);
// ************************************************************************

Serial.println(group);
easyvr.recognizeCommand(group);
do
{
 // can do some processing while waiting for a spoken command
}
while (!easyvr.hasFinished());
easyvr.setPinOutput(EasyVR::IO1, LOW); // LED off

//*********************************************************
// Activamos indicadores del proceso de reconocimiento de voz
digitalWrite(led1D, LOW);
digitalWrite(led2D, HIGH);
digitalWrite(led1I, LOW);
digitalWrite(led2I, HIGH);
//*********************************************************

idx = easyvr.getWord();
if (idx >= 0)
{
 // built-in trigger (ROBOT)
 // group = GROUP_X; <-- jump to another group X
 return;
}
idx = easyvr.getCommand();
if (idx >= 0)
{
 // print debug message
 uint8_t train = 0;
 char name[32];
 Serial.print("Command: ");
 Serial.print(idx);
 if (easyvr.dumpCommand(group, idx, name, train))
 {
  Serial.print(" = ");
  Serial.println(name);
```

```
    }
    else
      Serial.println();
    easyvr.playSound(0, EasyVR::VOL_FULL);
    // perform some action
    action();

  }
  else // errors or timeout
  {
    if (easyvr.isTimeout())
      Serial.println("Timed out, try again...");
    int16_t err = easyvr.getError();
    if (err >= 0)
    {
      Serial.print("Error ");
      Serial.println(err, HEX);
    }
  }
}
// ************************************************************************
```

NOTA: La estructura del codigo fuente se modifica:

- Activando la tabla de sonidos.
- Activando los indicadores Led

```
// ************************************************************************
//CODIGO DE PROGRAMACIÓN NUEVO:
// INICIALIZACION DE VARIABLES
int ManoD = 0;      // Define lateralidad Mano Derecha
int ManoI = 0;      // Define lateralidad Mano Izquierda
int pos = 0;        // Variable posición de servos
 int posi = 0;      // Posición Inicial de servos
int pos1D = 0;      // Posición Servo 1 Derecho
int pos1I = 0;      // Posición Servo 1 Izquierdo
int pos2D = 0;      // Posición Grupo Servos 2 Derechos
int pos2I = 0;      // Posición Grupo Servos 2 Izquierdos
int pos3D = 0;      // Posición Grupo Servos 3 Derechos
```

```
int pos3I = 0;        // Posición Grupo Servos 3 Izquierdos
int pos5D = 0;        // Posición Grupo Servos 5 Derechos
int pos5I = 0;        // Posición Grupo Servos 5 Izquierdos
int posact = 70;      // Posición Actual inició del programa
int posactS1D = 0; // Posición Actual Grupo Servos 1 Derechos
int posactS1I = 0;  // Posición Actual Grupo Servos 1 Izquierdos
int posactS2D = 0; // Posición Actual Grupo Servos 2 Derechos
int posactS2I = 0;  // Posición Actual Grupo Servos 2 Izquierdos
int posactS3D = 0;  // Posición Actual Grupo Servos 3 Derechos
int posactS3I = 0;   // Posición Actual Grupo Servos 3 Izquierdos
int posactS5D = 0;  // Posición Actual Grupo Servos 3 Derechos
int posactS5I = 0;   // Posición Actual Grupo Servos 5 Izquierdos
int posSFR2D = 0;  //Posición Según Sensor Fuerza Resistivo Servo2D
int posSFR3D = 0;  // Posición Según Sensor Fuerza Resistivo Servo3D
int posSFR4D = 0;  // Posición Según Sensor Fuerza Resistivo Servo4D
int posSFR5D = 0;  // Posición Según Sensor Fuerza Resistivo Servo5D
int posSFR6D = 0;  // Posición Según Sensor Fuerza Resistivo Servo6D
int posSFR7D = 0;  // Posición Según Sensor Fuerza Resistivo Servo7D
int posSFR2I = 0;   // Posición Según Sensor Fuerza Resistivo Servo2I
int posSFR3I = 0;   // Posición Según Sensor Fuerza Resistivo Servo2I
int posSFR4I = 0;   // Posición Según Sensor Fuerza Resistivo Servo2I
int posSFR5I = 0;   // Posición Según Sensor Fuerza Resistivo Servo2I
int posSFR6I = 0;   // Posición Según Sensor Fuerza Resistivo Servo2I
int posSFR7I = 0;   // Posición Según Sensor Fuerza Resistivo Servo2I
int sensorF1D = 0; // Valor digital del SFR 1 Derecho
int sensorF2D = 0; // Valor digital del SFR 2 Derecho
int sensorF3D = 0; // Valor digital del SFR 3 Derecho
int sensorF4D = 0; // Valor digital del SFR 4 Derecho
int sensorF5D = 0; // Valor digital del SFR 5 Derecho
int sensorF1I = 0;  // Valor digital del SFR 1 Izquierdo
int sensorF2I = 0;  // Valor digital del SFR 2 Izquierdo
int sensorF3I = 0;  // Valor digital del SFR 3 Izquierdo
int sensorF4I = 0;  // Valor digital del SFR 4 Izquierdo
int sensorF5I = 0;  // Valor digital del SFR 5 Izquierdo
float voltajeSF1D = 0; // Voltaje analogo en SFR 1 Derecho
float voltajeSF2D = 0; // Voltaje analogo en SFR 2 Derecho
float voltajeSF3D = 0; // Voltaje analogo en SFR 3 Derecho
float voltajeSF4D = 0; // Voltaje analogo en SFR 4 Derecho
float voltajeSF5D = 0; // Voltaje analogo en SFR 5 Derecho
float voltajeSF1I = 0;  // Voltaje analogo en SFR 1 Izquierdo
float voltajeSF2I = 0;  // Voltaje analogo en SFR 2 Izquierdo
```

```
float voltajeSF3I = 0;  // Voltaje analogo en SFR 3 Izquierdo
float voltajeSF4I = 0;  // Voltaje analogo en SFR 4 Izquierdo
float voltajeSF5I = 0;  // Voltaje analogo en SFR 5 Izquierdo
long fuerza = 3.5;      // Parámetro de fuerza determinante
int EMGactivo = 0;      // Activación del sensor de EMG
int EMGinactivo = 0;    // Inactivación del sensor de EMG
float sensorEMG1D = 0; // Valor análogo del sensor de EMG Derecho
float sensorEMG1I = 0;  // Valor análogo del sensor de EMG Izquierdo
float voltajeSEMG1D = 0; // Valor análogo del sensor de EMG Derecho
float voltajeSEMG1I = 0;  // Valor análogo del sensor de EMG Derecho
float segs = (segs * millis()); // Establece tiempo en segundos
float mins = (segs * 60);       // Establece tiempo en minutos
int time = 0;                   // Variable tiempo
int activa = 0;                 // Rehabilitación Activa
int pasiva = 0;                 // Rehabilitación Pasiva
// ************************************************************************
```

NOTA: La estructura del codigo fuente se modifica, ya que se crean todas las variables requeridas por el programa.

```
// ********************* CICLO INFINITO ******************************

void action()
{
  switch (group)
  {
  case GROUP_0:
   switch (idx)
   {
   case G0_EMMRA:
    for (posi = posact; pos <= 90; pos += 1)
    {
      servo1D.write(pos);   // SE INICIAN LOS SERVOS EN LA POSICION 90
      servo1I.write(pos);
      servo2D.write(pos);
      servo2I.write(pos);
      servo3D.write(pos);
```

```
      servo3I.write(pos);
      servo4D.write(pos);
      servo4I.write(pos);
      servo5D.write(pos);
      servo5I.write(pos);
      servo6D.write(pos);
      servo6I.write(pos);
      servo7D.write(pos);
      servo7I.write(pos);
      servo8D.write(pos);
      servo8I.write(pos);
      delay(15);
    }
    posactS2D = pos;   // TODAS LAS VARIABLES TOMAN EL ULTIMO
                       // VALOR DE LA VARIABLE POSICION (90 GRADOS)
    posactS2I = pos;
    posactS3D = pos;
    posactS3I = pos;
    posactS5D = pos;
    posactS5I = pos;
    pos = 0;
    delay(15);
    voz = 2;                // AL SALIR DE ESTA RUTINA, DA LA OPCION 2
    group = GROUP_1;   // AL SALIR DE ESTA RUTINA VA AL GRUPO 1
    break;
  }
  break;

case GROUP_1:
  switch (idx)
  {
  case G1_ASISTENCIA:
    voz = 3;
    group = GROUP_2;
    break;
  case G1_REHABILITACION:
    voz = 8;
    group = GROUP_7;
    break;
  case G1_SILLA:
    voz = 9;
```

```
    group = GROUP_8;
    break;
   }
   break;

 case GROUP_2:
   switch (idx)
   {
   case G2_MANO_DERECHA:
    ManoD = 1;
    ManoI = 0;
    voz = 4;
    group = GROUP_3;
    break;
   case G2_MANO_IZQUIERDA:
    ManoD = 0;
    ManoI = 1;
    voz = 4;
    group = GROUP_3;
    break;
   case G2_AMBAS_MANOS:
    ManoD = 1;
    ManoI = 1;
    voz = 4;
    group = GROUP_3;
    break;
   case G2_REINICIAR:
    voz = 1;
    group = GROUP_0;
    break;
   }
   break;


 case GROUP_3:
   switch (idx)
   {

   //*****RUTINA PARA ABRIR LAS MANOS*****
   //*************************************************************
   case G3_ABRIR:
```

```
if (ManoD == 1 && Manol == 0)
{
  if (posactS1D >= 55)
  {
   for (pos1D = posactS1D; pos1D >=50; pos1D -=2)
    {
      servo1D.write(pos1D);
      delay (15);
    }
    posactS1D = pos1D;
    pos1D = 0;
  }
  pos3D = posactS5D;
  for (pos2D = posactS2D; pos2D >=10; pos2D -=2)
  {
    pos3D += 2;
    if (pos2D >= 50)
    {
      servo2D.write(pos2D);
      servo3D.write(pos2D);
      servo4D.write(pos2D);
      servo6D.write(pos2D);
      delay (5);
    }
    if (pos3D <= 120)
    {
      servo5D.write(pos3D);
      servo7D.write(pos3D);
      delay (5);
    }
    // ****** servo8.write(pos2);
  }
  posactS2D = 50;
  posactS5D = 120;
  pos2D = 0;
  pos3D = 0;
  delay (15);
}
```

```
else if (ManoD == 0 && ManoI == 1)
{
  if (posactS1I <= 145)
  {
    for (pos1I = posactS1I; pos1I <=145; pos1I +=2)
    {
      servo1I.write(pos1I);
      delay (15);
    }
  posactS1I = pos1I;
  pos1I = 0;
  }
  pos3I = posactS5I;
  for (pos2I = posactS2I; pos2I <=150; pos2I +=2)
  {
    pos3I -= 2;
    if (pos2I <=100)
    {
      servo4I.write(pos2I);
      servo2I.write(pos2I);
      delay (5);
    }
    if (pos2I <=110)
    {
      servo3I.write(pos2I);
      delay (5);
    }
    if (pos2I <=120)
    {
      servo6I.write(pos2I);
      delay (5);
    }
    if (pos3I >= 60)
    {
      servo7I.write(pos3I);
      delay (5);
    }
    if (pos3I >= 50)
    {
      servo5I.write(pos3I);
      delay (5);
```

```
    }
    // ****** servo8.write(pos2);
  }
  posactS2I = 100;
  posactS5I = 60;
  pos2I = 0;
  pos3I = 0;
  delay (5);
}


else if (ManoD == 1 && ManoI == 1)
{
  if (posactS1I <= 145)
  {
    for (pos1I = posactS1I; pos1I <=145; pos1I +=2)
    {
      servo1I.write(pos1I);
      delay (15);
    }
    posactS1I = pos1I;
    pos1I = 0;
  }
  if (posactS1D >= 55)
  {
    for (pos1D = posactS1D; pos1D >=50; pos1D -=2)
    {
      servo1D.write(pos1D);
      delay (15);
    }
    posactS1D = pos1D;
    pos1D = 0;
  }
  pos3I = posactS5I;
  for (pos2I = posactS2I; pos2I <=150; pos2I +=2)
  {
    pos3I -= 2;
    if (pos2I <=100)
    {
      servo4I.write(pos2I);
```

```
    servo2I.write(pos2I);
    delay (5);
  }
  if (pos2I <=110)
  {
    servo3I.write(pos2I);
    delay (5);
  }
  if (pos2I <=120)
  {
    servo6I.write(pos2I);
    delay (5);
  }
  if (pos3I >= 60)
  {
    servo7I.write(pos3I);
    delay (5);
  }
  if (pos3I >= 50)
  {
    servo5I.write(pos3I);
    delay (5);
  }
  // ****** servo8.write(pos2);
}
posactS2I = 100;
posactS5I = 60;
pos2I = 0;
pos3I = 0;
delay (5);

pos3D = posactS5D;
for (pos2D = posactS2D; pos2D >=10; pos2D -=2)
{
  pos3D += 2;
  if (pos2D >= 50)
  {
    servo2D.write(pos2D);
    servo3D.write(pos2D);
    servo4D.write(pos2D);
    servo6D.write(pos2D);
```

```
        delay (5);
      }
      if (pos3D <= 120)
      {
        servo5D.write(pos3D);
        servo7D.write(pos3D);
        delay (5);
      }
      // ****** servo8.write(pos2);
    }
    posactS2D = 50;
    posactS5D = 120;
    pos2D = 0;
    pos3D = 0;
    delay (15);
  }
  voz = 0;
  group = GROUP_3;
  break;

//*****RUTINA PARA CERRAR LAS MANOS*****
//*********************************************************************
case G3_CERRAR:

  if (ManoD == 1 && ManoI == 0)
  {
    if (posactS1D <= 145)
    {
      for (pos1D = posactS1D; pos1D <=145; pos1D +=2)
      {
        servo1D.write(pos1D);
        delay (15);
      }
      posactS1D = pos1D;
      pos1D = 0;
    }

    pos3D = posactS5D;
    for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
    {
      pos3D -=2;
```

```
if (pos2D <= 100)
{
  // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
  sensorF1D = analogRead (A15);
  voltajeSF1D = sensorF1D * (5.0/1023.0);
  Serial.print (voltajeSF1D);
  if (voltajeSF1D < fuerza)
  {
    servo2D.write(pos2D);
    posSFR2D = pos2D;
  }
  // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
  sensorF2D = analogRead (A14);
  voltajeSF2D = sensorF2D * (5.0/1023.0);
  Serial.print (voltajeSF2D);
  if (voltajeSF2D < fuerza)
  {
    servo3D.write(pos2D);
    posSFR3D = pos2D;
  }
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
  sensorF3D = analogRead (A13);
  voltajeSF3D = sensorF3D * (5.0/1023.0);
  Serial.print (voltajeSF3D);
  if (voltajeSF3D < fuerza)
  {
    if (posactS2D <=90)
    {
      servo4D.write(pos2D);
      posSFR4D = pos2D;
    }
  }
}

if (pos3D >= 50)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
  sensorF4D = analogRead (A12);
  voltajeSF4D = sensorF4D * (5.0/1023.0);
  Serial.print (voltajeSF4D);
  if (voltajeSF4D < fuerza)
```

```
      {
        servo5D.write(pos3D);
        posSFR5D = pos3D;
      }
    }
    if (pos2D <= 120)
    {
      // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
      sensorF5D = analogRead (A11);
      voltajeSF5D = sensorF5D * (5.0/1023.0);
      Serial.print (voltajeSF5D);
      if (voltajeSF5D < fuerza)
      {
        servo6D.write(pos2D);
        if (pos3D >= 70)
        {
          servo7D.write(pos3D);
        }
        posSFR6D = pos2D;
        posSFR7D = pos3D;
      }
    }
    // ****** servo8.write(pos2);
  }
  posactS2D = 100;
  posactS5D = 70;
  pos2D = 0;
  pos3D = 0;
  fuerza = 3.5;
  delay (15);
}


else if (ManoD == 0 && ManoI == 1)
{
  if (posactS1I >= 50)
  {
    for (pos1I = posactS1I; pos1I >=50; pos1I -=2)
    {
      servo1I.write(pos1I);
      delay (15);
```

```
  }
  posactS1I = pos1I;
  pos1I = 0;
}

pos3I = posactS5I;
for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
{
  pos3I +=2;
  if (pos2I >= 50)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
    sensorF1I = analogRead (A0);
    voltajeSF1I = sensorF1I * (5.0/1023.0);
    Serial.print (voltajeSF1I);
    if (voltajeSF1I < fuerza)
    {
      servo2I.write(pos2I);
      posSFR2I = pos2I;
    }
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
    sensorF2I = analogRead (A1);
    voltajeSF2I = sensorF2I * (5.0/1023.0);
    Serial.print (voltajeSF2I);
    if (voltajeSF2I < fuerza)
    {
      servo3I.write(pos2I);
      posSFR3I = pos2I;
    }
    // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
    sensorF3I = analogRead (A2);
    voltajeSF3I = sensorF3I * (5.0/1023.0);
    Serial.print (voltajeSF3I);
    if (voltajeSF3I < fuerza)
    {
      servo4I.write(pos2I);
      posSFR4I = pos2I;
    }
  }

  if (pos3I <= 120)
```

```
    {
      // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
      sensorF4I = analogRead (A3);
      voltajeSF4I = sensorF4I * (5.0/1023.0);
      Serial.print (voltajeSF4I);
      if (voltajeSF4I < fuerza)
      {
        servo5I.write(pos3I);
        posSFR5I = pos3I;
      }
    }

// ********************************************************************************
    if (pos2I >= 50)
    {
      // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
      sensorF5I = analogRead (A4);
      voltajeSF5I = sensorF5I * (5.0/1023.0);
      Serial.print (voltajeSF5I);
      if (voltajeSF5I < fuerza)
      {
        servo6I.write(pos2I);
        if (pos3I <= 100)
        {
          servo7I.write(pos3I);
        }
         posSFR6I = pos2I;
         posSFR7I = pos3I;
      }
    }
    // ****** servo8.write(pos2);
  }
  posactS2I = 50;
  posactS5I = 100;
  pos2I = 0;
  pos3I = 0;
  fuerza = 3.5;
  delay (15);
}

else if (ManoD == 1 && ManoI == 1)
```

```
{
  if (posactS1D <= 145)
  {
    for (pos1D = posactS1D; pos1D <=150; pos1D +=2)
    {
      servo1D.write(pos1D);
      delay (15);
    }
    posactS1D = pos1D;
    pos1D = 0;
  }

  pos3D = posactS5D;
  for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
  {
    pos3D -=2;
    if (pos2D <= 100)
    {
      // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
      sensorF1D = analogRead (A15);
      voltajeSF1D = sensorF1D * (5.0/1023.0);
      Serial.print (voltajeSF1D);
      if (voltajeSF1D < fuerza)
      {
        servo2D.write(pos2D);
        posSFR2D = pos2D;
      }
      // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
      sensorF2D = analogRead (A14);
      voltajeSF2D = sensorF2D * (5.0/1023.0);
      Serial.print (voltajeSF2D);
      if (voltajeSF2D < fuerza)
      {
        servo3D.write(pos2D);
        posSFR3D = pos2D;
      }
      // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
      sensorF3D = analogRead (A13);
      voltajeSF3D = sensorF3D * (5.0/1023.0);
      Serial.print (voltajeSF3D);
      if (voltajeSF3D < fuerza)
```

```
    {
      if (posactS2D <=90)
      {
        servo4D.write(pos2D);
        posSFR4D = pos2D;
      }
    }

  }

  if (pos3D >= 50)
  {
    // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
    sensorF4D = analogRead (A12);
    voltajeSF4D = sensorF4D * (5.0/1023.0);
    Serial.print (voltajeSF4D);
    if (voltajeSF4D < fuerza)
    {
      servo5D.write(pos3D);
      posSFR5D = pos3D;
    }
  }
  if (pos2D <= 120)
  {
    // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
    sensorF5D = analogRead (A11);
    voltajeSF5D = sensorF5D * (5.0/1023.0);
    Serial.print (voltajeSF5D);
    if (voltajeSF5D < fuerza)
    {
      servo6D.write(pos2D);
      if (pos3D >= 70)
      {
        servo7D.write(pos3D);
      }
      posSFR6D = pos2D;
      posSFR7D = pos3D;
    }
  }
  // ****** servo8.write(pos2);
}
```

```
posactS2D = 100;
posactS5D = 70;
pos2D = 0;
pos3D = 0;
fuerza = 3.5;
delay (15);

if (posactS1I >= 50)
{
  for (pos1I = posactS1I; pos1I >=50; pos1I -=2)
  {
    servo1I.write(pos1I);
    delay (15);
  }
  posactS1I = pos1I;
  pos1I = 0;
}

pos3I = posactS5I;
for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
{
  pos3I +=2;
  if (pos2I >= 50)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
    sensorF1I = analogRead (A0);
    voltajeSF1I = sensorF1I * (5.0/1023.0);
    Serial.print (voltajeSF1I);
    if (voltajeSF1I < fuerza)
    {
      servo2I.write(pos2I);
      posSFR2I = pos2I;
    }
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
    sensorF2I = analogRead (A1);
    voltajeSF2I = sensorF2I * (5.0/1023.0);
    Serial.print (voltajeSF2I);
    if (voltajeSF2I < fuerza)
    {
      servo3I.write(pos2I);
      posSFR3I = pos2I;
```

```
  }
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
  sensorF3I = analogRead (A2);
  voltajeSF3I = sensorF3I * (5.0/1023.0);
  Serial.print (voltajeSF3I);
  if (voltajeSF3I < fuerza)
  {
    servo4I.write(pos2I);
    posSFR4I = pos2I;
  }
}


if (pos3I <= 100)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
  sensorF4I = analogRead (A3);
  voltajeSF4I = sensorF4I * (5.0/1023.0);
  Serial.print (voltajeSF4I);
  if (voltajeSF4I < fuerza)
  {
    servo5I.write(pos3I);
    posSFR5I = pos3I;
  }
}

if (pos2I >= 50)
{
  // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
  sensorF5I = analogRead (A4);
  voltajeSF5I = sensorF5I * (5.0/1023.0);
  Serial.print (voltajeSF5I);
  if (voltajeSF5I < fuerza)
  {
    servo6I.write(pos2I);
    if (pos3I <= 100)
    {
      servo7I.write(pos3I);
    }
     posSFR6I = pos2I;
     posSFR7I = pos3I;
```

```
            }
          }
          // ****** servo8.write(pos2);
        }
        posactS2I = 50;
        posactS5I = 100;
        pos2I = 0;
        pos3I = 0;
        fuerza = 3.5;
        delay (15);
      }
      voz = 0;
      group = GROUP_3;
      break;

//*****RUTINA PARA AGARRAR UN OBJETO*****
//*********************************************************************

    case G3_AGARRAR:

      if (ManoD == 1 && ManoI == 0)
      {
        if (posactS1D <= 145)
        {
          for (pos1D = posactS1D; pos1D <=150; pos1D +=2)
          {
            servo1D.write(pos1D);
            delay (15);
          }
          posactS1D = pos1D;
          pos1D = 0;
        }

        pos3D = posactS5D;
        for (pos2D = posactS2D; pos2D >=10; pos2D -=2)
        {
          pos3D += 2;
          if (pos2D >= 50)
          {
            servo2D.write(pos2D);
            servo3D.write(pos2D);
```

```
        servo4D.write(pos2D);
        servo6D.write(pos2D);
        delay (5);
      }
      if (pos3D <= 120)
      {
        servo5D.write(pos3D);
        servo7D.write(pos3D);
        delay (5);
      }
      // ****** servo8.write(pos2);
    }
    posactS2D = 50;
    posactS5D = 120;
    pos2D = 0;
    pos3D = 0;
    delay (15);
}

else if (ManoD == 0 && ManoI == 1)
{
  if (posactS1I >= 50)
  {
    for (pos1I = posactS1I; pos1I >=50; pos1I -=2)
    {
      servo1I.write(pos1I);
      delay (15);
    }
    posactS1I = pos1I;
    pos1I = 0;
  }

  pos3I = posactS5I;
  for (pos2I = posactS2I; pos2I <=150; pos2I +=2)
  {
    pos3I -= 2;
    if (pos2I <=100)
    {
      servo4I.write(pos2I);
      servo2I.write(pos2I);
      delay (5);
```

```
    }
    if (pos2l <=110)
    {
      servo3l.write(pos2l);
      delay (5);
    }
    if (pos2l <=120)
    {
      servo6l.write(pos2l);
      delay (5);
    }
    if (pos3l >= 60)
    {
      servo7l.write(pos3l);
      delay (5);
    }
    if (pos3l >= 50)
    {
      servo5l.write(pos3l);
      delay (5);
    }
    // ****** servo8.write(pos2);
  }
  posactS2l = 100;
  posactS5l = 60;
  pos2l = 0;
  pos3l = 0;
  delay (5);
}

else if (ManoD == 1 && Manol == 1)
{
  if (posactS1D <= 145)
  {
    for (pos1D = posactS1D; pos1D <=150; pos1D +=2)
    {
      servo1D.write(pos1D);
      delay (15);
    }
    posactS1D = pos1D;
    pos1D = 0;
```

```
}

pos3D = posactS5D;
for (pos2D = posactS2D; pos2D >=10; pos2D -=2)
{
  pos3D += 2;
  if (pos2D >= 50)
  {
    servo2D.write(pos2D);
    servo3D.write(pos2D);
    servo4D.write(pos2D);
    servo6D.write(pos2D);
    delay (5);
  }
  if (pos3D <= 120)
  {
    servo5D.write(pos3D);
    servo7D.write(pos3D);
    delay (5);
  }
  // ****** servo8.write(pos2);
}
posactS2D = 50;
posactS5D = 120;
pos2D = 0;
pos3D = 0;
delay (15);

if (posactS1I >= 50)
{
  for (pos1I = posactS1I; pos1I >=50; pos1I -=2)
  {
    servo1I.write(pos1I);
    delay (15);
  }
  posactS1I = pos1I;
  pos1I = 0;
}

pos3I = posactS5I;
for (pos2I = posactS2I; pos2I <=150; pos2I +=2)
```

```
  {
    pos3l -= 2;
    if (pos2l <=100)
    {
      servo4l.write(pos2l);
      servo2l.write(pos2l);
      delay (5);
    }
    if (pos2l <=110)
    {
      servo3l.write(pos2l);
      delay (5);
    }
    if (pos2l <=120)
    {
      servo6l.write(pos2l);
      delay (5);
    }
    if (pos3l >= 60)
    {
      servo7l.write(pos3l);
      delay (5);
    }
    if (pos3l >= 50)
    {
      servo5l.write(pos3l);
      delay (5);
    }
    // ****** servo8.write(pos2);
  }
  posactS2l = 100;
  posactS5l = 60;
  pos2l = 0;
  pos3l = 0;
  delay (5);
}
voz = 5;
group = GROUP_4;
break;
```

```
//*****RUTINA PARA REGRESAR AL GRUPO ANTERIOR*****
// ************************************************************************

    case G3_REINICIAR:
     voz = 3;
     group = GROUP_2;
     break;

//*****RUTINA PARA ESCUCHAR LAS OPCIONES DE SONIDO*****
// ************************************************************************

    case G3_AUDIO:
     voz = 4;
     group = GROUP_3;
     break;
    }
    break;

//*****RUTINA PARA AGARRAR OBJETOS FUERTEMENTE*********
// ************************************************************************
   case GROUP_4:
    switch (idx)
    {
    case G4_FUERTE:
     fuerza = 4.35;
     voz = 6;
     group = GROUP_6;
     break;

    case G4_SUAVE:
     fuerza = 3.35;
     voz = 6;
     group = GROUP_6;
     break;
    }
    break;


//*****RUTINA PARA REHABILITACION POR CINCO MINUTOS******
// ************************************************************************
```

```
case GROUP_5:
 switch (idx)
 {
 case G5_CINCO:

 if (EMGactivo == 1 && EMGinactivo == 0)
 {
  for (time = 0; time < 5; time +=1)
  {
  for (mins = 0; mins < 59; mins +=1)
  {
  for (segs = 0; segs < 125; segs +=1)
  {
  sensorEMG1D = analogRead (A5);
  voltajeSEMG1D = sensorEMG1D * (5.0/1023.0);
  Serial.print (voltajeSEMG1D);
  sensorEMG1I = analogRead (A8);
  voltajeSEMG1I = sensorEMG1I * (5.0/1023.0);
  Serial.print (voltajeSEMG1I);

  if (voltajeSEMG1D > 2.5)
  {
    digitalWrite(led1D, LOW);
    digitalWrite(led2D, HIGH);
    if (posactS1D >= 55)
    {
      for (pos1D = posactS1D; pos1D >=50; pos1D -=2)
      {
        servo1D.write(pos1D);
        delay (15);
      }
      posactS1D = pos1D;
      pos1D = 0;
   }
    pos3D = posactS5D;
    for (pos2D = posactS2D; pos2D >=10; pos2D -=2)
    {
      pos3D += 2;
      if (pos2D >= 50)
      {
        servo2D.write(pos2D);
```

```
      servo3D.write(pos2D);
      servo4D.write(pos2D);
      servo6D.write(pos2D);
      delay (10);
    }
    if (pos3D <= 120)
    {
      servo5D.write(pos3D);
      servo7D.write(pos3D);
      delay (10);
    }
    // ****** servo8.write(pos2);
  }
  posactS2D = 50;
  posactS5D = 120;
  pos2D = 0;
  pos3D = 0;
  delay (15);
}

else if (voltajeSEMG1D >= 1.5 && voltajeSEMG1D <= 2.5)
{
  digitalWrite(led1D, LOW);
  digitalWrite(led2D, HIGH);
  if (posactS1D <= 145)
  {
    for (pos1D = posactS1D; pos1D <=150; pos1D +=2)
    {
      servo1D.write(pos1D);
      delay (15);
    }
    posactS1D = pos1D;
    pos1D = 0;
  }

  pos3D = posactS5D;
  for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
  {
    pos3D -=2;
    if (pos2D <= 100)
    {
```

```
  // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
  sensorF1D = analogRead (A15);
  voltajeSF1D = sensorF1D * (5.0/1023.0);
  Serial.print (voltajeSF1D);
  if (voltajeSF1D < fuerza)
  {
    servo2D.write(pos2D);
    posSFR2D = pos2D;
  }
  // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
  sensorF2D = analogRead (A14);
  voltajeSF2D = sensorF2D * (5.0/1023.0);
  Serial.print (voltajeSF2D);
  if (voltajeSF2D < fuerza)
  {
    servo3D.write(pos2D);
    posSFR3D = pos2D;
  }
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
  sensorF3D = analogRead (A13);
  voltajeSF3D = sensorF3D * (5.0/1023.0);
  Serial.print (voltajeSF3D);
  if (voltajeSF3D < fuerza)
  {
    if (posactS2D <=90)
    {
      servo4D.write(pos2D);
      posSFR4D = pos2D;
    }
  }
}

if (pos3D >= 50)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
  sensorF4D = analogRead (A12);
  voltajeSF4D = sensorF4D * (5.0/1023.0);
  Serial.print (voltajeSF4D);
  if (voltajeSF4D < fuerza)
  {
    servo5D.write(pos3D);
```

```
        posSFR5D = pos3D;
      }
    }
    if (pos2D <= 120)
    {
      // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
      sensorF5D = analogRead (A11);
      voltajeSF5D = sensorF5D * (5.0/1023.0);
      Serial.print (voltajeSF5D);
      if (voltajeSF5D < fuerza)
      {
        servo6D.write(pos2D);
        if (pos3D >= 70)
        {
          servo7D.write(pos3D);
        }
        posSFR6D = pos2D;
        posSFR7D = pos3D;
      }
    }
    // ****** servo8.write(pos2);
  }
  posactS2D = 100;
  posactS5D = 70;
  pos2D = 0;
  pos3D = 0;
  fuerza = 3.5;
  delay (15);
}

else if (voltajeSEMG1D < 1.5)
{
 digitalWrite(led1D, HIGH);
 digitalWrite(led2D, LOW);
 delay (5);
}

if (voltajeSEMG1I > 2.5)
{
  digitalWrite(led1I, LOW);
  digitalWrite(led2I, HIGH);
```

```
if (posactS1l <= 145)
{
  for (pos1l = posactS1l; pos1l <=150; pos1l +=2)
  {
    servo1l.write(pos1l);
    delay (15);
  }
posactS1l = pos1l;
pos1l = 0;
}
pos3l = posactS5l;
for (pos2l = posactS2l; pos2l <=150; pos2l +=2)
{
  pos3l -= 2;
  if (pos2l <=100)
  {
    servo4l.write(pos2l);
    servo2l.write(pos2l);
    delay (5);
  }
  if (pos2l <=110)
  {
    servo3l.write(pos2l);
    delay (5);
  }
  if (pos2l <=120)
  {
    servo6l.write(pos2l);
    delay (5);
  }
  if (pos3l >= 60)
  {
    servo7l.write(pos3l);
    delay (5);
  }
  if (pos3l >= 50)
  {
    servo5l.write(pos3l);
    delay (5);
  }
```

```
    // ****** servo8.write(pos2);
  }
  posactS2I = 100;
  posactS5I = 60;
  pos2I = 0;
  pos3I = 0;
}

else if (voltajeSEMG1I > 1.5 && voltajeSEMG1I <= 2.5)
{
  digitalWrite(led1I, LOW);
  digitalWrite(led2I, HIGH);

  if (posactS1I >= 50)
  {
    for (pos1I = posactS1I; pos1I >=50; pos1I -=2)
    {
      servo1I.write(pos1I);
      delay (15);
    }
    posactS1I = pos1I;
    pos1I = 0;
  }

  pos3I = posactS5I;
  for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
  {
    pos3I +=2;
    if (pos2I >= 50)
    {
      // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
      sensorF1I = analogRead (A0);
      voltajeSF1I = sensorF1I * (5.0/1023.0);
      Serial.print (voltajeSF1I);
      if (voltajeSF1I < fuerza)
      {
        servo2I.write(pos2I);
        posSFR2I = pos2I;
      }
      // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
      sensorF2I = analogRead (A1);
```

```
  voltajeSF2I = sensorF2I * (5.0/1023.0);
  Serial.print (voltajeSF2I);
  if (voltajeSF2I < fuerza)
  {
    servo3I.write(pos2I);
    posSFR3I = pos2I;
  }
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
  sensorF3I = analogRead (A2);
  voltajeSF3I = sensorF3I * (5.0/1023.0);
  Serial.print (voltajeSF3I);
  if (voltajeSF3I < fuerza)
  {
    servo4I.write(pos2I);
    posSFR4I = pos2I;
  }
}

if (pos3I <= 100)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
  sensorF4I = analogRead (A3);
  voltajeSF4I = sensorF4I * (5.0/1023.0);
  Serial.print (voltajeSF4I);
  if (voltajeSF4I < fuerza)
  {
    servo5I.write(pos3I);
    posSFR5I = pos3I;
  }
}

if (pos2I >= 50)
{
  // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
  sensorF5I = analogRead (A4);
  voltajeSF5I = sensorF5I * (5.0/1023.0);
  Serial.print (voltajeSF5I);
  if (voltajeSF5I < fuerza)
  {
    servo6I.write(pos2I);
    if (pos3I <= 100)
```

```
            {
              servo7I.write(pos3I);
            }
             posSFR6I = pos2I;
             posSFR7I = pos3I;
          }
        }
        // ****** servo8.write(pos2);
      }
      posactS2I = 50;
      posactS5I = 100;
      pos2I = 0;
      pos3I = 0;
      fuerza = 3.5;
      delay (15);
    }

    else if (voltajeSEMG1I < 1.5)
    {
     digitalWrite(led1I, HIGH);
     digitalWrite(led2I, LOW);
     delay (5);
    }
  delay (50);
    }
    }
    }
  }

  else if (EMGactivo == 0 && EMGinactivo == 1)
  {
    for (time = 0; time < 5; time +=1)
    {
    for (mins = 0; mins < 5; mins +=1)
    {
    for (segs = 0; segs <= 1; segs +=1)
    {
      // ABRIR MANO DERECHA ////////////////////////
      if (posactS1D >= 55)
      {
        for (pos1D = posactS1D; pos1D >=50; pos1D -=2)
```

```
  {
    servo1D.write(pos1D);
    delay (8);
  }
  posactS1D = pos1D;
  pos1D = 0;
}
pos3D = posactS5D;
for (pos2D = posactS2D; pos2D >=10; pos2D -=2)
{
  pos3D += 2;
  if (pos2D >= 50)
  {
    servo2D.write(pos2D);
    servo3D.write(pos2D);
    servo4D.write(pos2D);
    servo6D.write(pos2D);
    delay (5);
  }
  if (pos3D <= 120)
  {
    servo5D.write(pos3D);
    servo7D.write(pos3D);
    delay (5);
  }
  // ****** servo8.write(pos2);
}
posactS2D = 50;
posactS5D = 120;
pos2D = 0;
pos3D = 0;

// ********************************

// ABRIR MANO IZQUIERDA ////////////////////////
if (posactS1I <= 145)
{
  for (pos1I = posactS1I; pos1I <=150; pos1I +=2)
  {
    servo1I.write(pos1I);
    delay (8);
```

```
  }
posactS1l = pos1l;
pos1l = 0;
}
pos3l = posactS5l;
for (pos2l = posactS2l; pos2l <=150; pos2l +=2)
{
  pos3l -= 2;
  if (pos2l <=100)
  {
    servo4l.write(pos2l);
    servo2l.write(pos2l);
    delay (5);
  }
  if (pos2l <=110)
  {
    servo3l.write(pos2l);
    delay (5);
  }
  if (pos2l <=120)
  {
    servo6l.write(pos2l);
    delay (5);
  }
  if (pos3l >= 60)
  {
    servo7l.write(pos3l);
    delay (5);
  }
  if (pos3l >= 50)
  {
    servo5l.write(pos3l);
    delay (5);
  }
  // ****** servo8.write(pos2);
}
posactS2l = 100;
posactS5l = 60;
pos2l = 0;
pos3l = 0;
delay (1000);
```

```
// CERRAR MANO DERECHA ////////////////////////
if (posactS1D <= 145)
{
  for (pos1D = posactS1D; pos1D <=145; pos1D +=2)
  {
    servo1D.write(pos1D);
    delay (5);
  }
  posactS1D = pos1D;
  pos1D = 0;
}

pos3D = posactS5D;
for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
{
  pos3D -=2;
  if (pos2D <= 100)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
    sensorF1D = analogRead (A15);
    voltajeSF1D = sensorF1D * (5.0/1023.0);
    Serial.print (voltajeSF1D);
    if (voltajeSF1D < fuerza)
    {
      servo2D.write(pos2D);
      posSFR2D = pos2D;
    }
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
    sensorF2D = analogRead (A14);
    voltajeSF2D = sensorF2D * (5.0/1023.0);
    Serial.print (voltajeSF2D);
    if (voltajeSF2D < fuerza)
    {
      servo3D.write(pos2D);
      posSFR3D = pos2D;
    }
    // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
    sensorF3D = analogRead (A13);
    voltajeSF3D = sensorF3D * (5.0/1023.0);
    Serial.print (voltajeSF3D);
```

```
    if (voltajeSF3D < fuerza)
    {
      if (posactS2D <=90)
      {
        servo4D.write(pos2D);
        posSFR4D = pos2D;
      }
    }
  }

  if (pos3D >= 50)
  {
    // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
    sensorF4D = analogRead (A12);
    voltajeSF4D = sensorF4D * (5.0/1023.0);
    Serial.print (voltajeSF4D);
    if (voltajeSF4D < fuerza)
    {
      servo5D.write(pos3D);
      posSFR5D = pos3D;
    }
  }
  if (pos2D <= 120)
  {
    // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
    sensorF5D = analogRead (A11);
    voltajeSF5D = sensorF5D * (5.0/1023.0);
    Serial.print (voltajeSF5D);
    if (voltajeSF5D < fuerza)
    {
      servo6D.write(pos2D);
      if (pos3D >= 70)
      {
        servo7D.write(pos3D);
      }
      posSFR6D = pos2D;
      posSFR7D = pos3D;
    }
  }
  // ****** servo8.write(pos2);
}
```

```
posactS2D = 100;
posactS5D = 70;
pos2D = 0;
pos3D = 0;
fuerza = 3.5;

// CERRAR MANO IZQUIERDA ////////////////////////
if (posactS1I >= 50)
{
  for (pos1I = posactS1I; pos1I >=50; pos1I -=2)
  {
    servo1I.write(pos1I);
    delay (5);
  }
  posactS1I = pos1I;
  pos1I = 0;
}

pos3I = posactS5I;
for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
{
  pos3I +=2;
  if (pos2I >= 50)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
    sensorF1I = analogRead (A0);
    voltajeSF1I = sensorF1I * (5.0/1023.0);
    Serial.print (voltajeSF1I);
    if (voltajeSF1I < fuerza)
    {
      servo2I.write(pos2I);
      posSFR2I = pos2I;
    }
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
    sensorF2I = analogRead (A1);
    voltajeSF2I = sensorF2I * (5.0/1023.0);
    Serial.print (voltajeSF2I);

    if (voltajeSF2I < fuerza)
    {
      servo3I.write(pos2I);
```

```
    posSFR3I = pos2I;
  }
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
  sensorF3I = analogRead (A2);
  voltajeSF3I = sensorF3I * (5.0/1023.0);
  Serial.print (voltajeSF3I);
  if (voltajeSF3I < fuerza)
  {
    servo4I.write(pos2I);
    posSFR4I = pos2I;
  }
}

if (pos3I <= 100)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
  sensorF4I = analogRead (A3);
  voltajeSF4I = sensorF4I * (5.0/1023.0);
  Serial.print (voltajeSF4I);
  if (voltajeSF4I < fuerza)
  {
    servo5I.write(pos3I);
    posSFR5I = pos3I;
  }
}

if (pos2I >= 50)
{
  // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
  sensorF5I = analogRead (A4);
  voltajeSF5I = sensorF5I * (5.0/1023.0);
  Serial.print (voltajeSF5I);
  if (voltajeSF5I < fuerza)
  {
    servo6I.write(pos2I);
    if (pos3I <= 100)
    {
      servo7I.write(pos3I);
    }
     posSFR6I = pos2I;
     posSFR7I = pos3I;
```

```
          }
        }
        // ****** servo8.write(pos2);
      }
      posactS2I = 50;
      posactS5I = 100;
      pos2I = 0;
      pos3I = 0;
      fuerza = 3.5;
      delay (1000);
    }
    }
    }
  }

  EMGactivo = 0;
  EMGinactivo = 0;
  voz = 2;
  group = GROUP_1;
  break;




//*****RUTINA PARA REHABILITACION POR DIEZ MINUTOS********
// ********************************************************************************

case G5_DIEZ:

  if (EMGactivo == 1 && EMGinactivo == 0)
  {
   for (time = 0; time < 10; time +=1)
   {
   for (mins = 0; mins < 59; mins +=1)
   {
   for (segs = 0; segs < 125; segs +=1)
   {
   sensorEMG1D = analogRead (A5);
   voltajeSEMG1D = sensorEMG1D * (5.0/1023.0);
   Serial.print (voltajeSEMG1D);
```

```
sensorEMG1I = analogRead (A8);
voltajeSEMG1I = sensorEMG1I * (5.0/1023.0);
Serial.print (voltajeSEMG1I);

if (voltajeSEMG1D > 2.5)
{
  digitalWrite(led1D, LOW);
  digitalWrite(led2D, HIGH);
  if (posactS1D >= 55)
  {
    for (pos1D = posactS1D; pos1D >=50; pos1D -=2)
    {
      servo1D.write(pos1D);
      delay (15);
    }
    posactS1D = pos1D;
    pos1D = 0;
  }
  pos3D = posactS5D;
  for (pos2D = posactS2D; pos2D >=10; pos2D -=2)
  {
    pos3D += 2;
    if (pos2D >= 50)
    {
      servo2D.write(pos2D);
      servo3D.write(pos2D);
      servo4D.write(pos2D);
      servo6D.write(pos2D);
      delay (10);
    }
    if (pos3D <= 120)
    {
      servo5D.write(pos3D);
      servo7D.write(pos3D);
      delay (10);
    }
    // ****** servo8.write(pos2);
  }
  posactS2D = 50;
  posactS5D = 120;
  pos2D = 0;
```

```
  pos3D = 0;
  delay (15);
}

else if (voltajeSEMG1D >= 1.5 && voltajeSEMG1D <= 2.5)
{
  digitalWrite(led1D, LOW);
  digitalWrite(led2D, HIGH);
  if (posactS1D <= 145)
  {
    for (pos1D = posactS1D; pos1D <=150; pos1D +=2)
    {
      servo1D.write(pos1D);
      delay (15);
    }
    posactS1D = pos1D;
    pos1D = 0;
  }

  pos3D = posactS5D;
  for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
  {
    pos3D -=2;
    if (pos2D <= 100)
    {
      // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
      sensorF1D = analogRead (A15);
      voltajeSF1D = sensorF1D * (5.0/1023.0);
      Serial.print (voltajeSF1D);
      if (voltajeSF1D < fuerza)
      {
        servo2D.write(pos2D);
        posSFR2D = pos2D;
      }
      // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
      sensorF2D = analogRead (A14);
      voltajeSF2D = sensorF2D * (5.0/1023.0);
      Serial.print (voltajeSF2D);
      if (voltajeSF2D < fuerza)
      {
        servo3D.write(pos2D);
```

```
    posSFR3D = pos2D;
  }
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
  sensorF3D = analogRead (A13);
  voltajeSF3D = sensorF3D * (5.0/1023.0);
  Serial.print (voltajeSF3D);
  if (voltajeSF3D < fuerza)
  {
    if (posactS2D <=90)
    {
      servo4D.write(pos2D);
      posSFR4D = pos2D;
    }
  }
}

if (pos3D >= 50)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
  sensorF4D = analogRead (A12);
  voltajeSF4D = sensorF4D * (5.0/1023.0);
  Serial.print (voltajeSF4D);
  if (voltajeSF4D < fuerza)
  {
    servo5D.write(pos3D);
    posSFR5D = pos3D;
  }
}
if (pos2D <= 120)
{
  // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
  sensorF5D = analogRead (A11);
  voltajeSF5D = sensorF5D * (5.0/1023.0);
  Serial.print (voltajeSF5D);
  if (voltajeSF5D < fuerza)
  {
    servo6D.write(pos2D);
    if (pos3D >= 70)
    {
      servo7D.write(pos3D);
    }
```

```
        posSFR6D = pos2D;
        posSFR7D = pos3D;
      }
    }
    // ****** servo8.write(pos2);
  }
  posactS2D = 100;
  posactS5D = 70;
  pos2D = 0;
  pos3D = 0;
  fuerza = 3.5;
  delay (15);
}

else if (voltajeSEMG1D < 1.5)
{
 digitalWrite(led1D, HIGH);
 digitalWrite(led2D, LOW);
 delay (5);
}

if (voltajeSEMG1I > 2.5)
{
  digitalWrite(led1I, LOW);
  digitalWrite(led2I, HIGH);

  if (posactS1I <= 145)
  {
    for (pos1I = posactS1I; pos1I <=150; pos1I +=2)
    {
      servo1I.write(pos1I);
      delay (15);
    }
  posactS1I = pos1I;
  pos1I = 0;
  }
  pos3I = posactS5I;
  for (pos2I = posactS2I; pos2I <=150; pos2I +=2)
  {
    pos3I -= 2;
    if (pos2I <=100)
```

```
    {
      servo4l.write(pos2l);
      servo2l.write(pos2l);
      delay (5);
    }
    if (pos2l <=110)
    {
      servo3l.write(pos2l);
      delay (5);
    }
    if (pos2l <=120)
    {
      servo6l.write(pos2l);
      delay (5);
    }
    if (pos3l >= 60)
    {
      servo7l.write(pos3l);
      delay (5);
    }
    if (pos3l >= 50)
    {
      servo5l.write(pos3l);
      delay (5);
    }
    // ****** servo8.write(pos2);
  }
  posactS2l = 100;
  posactS5l = 60;
  pos2l = 0;
  pos3l = 0;
}

else if (voltajeSEMG1l > 1.5 && voltajeSEMG1l <= 2.5)
{
  digitalWrite(led1l, LOW);
  digitalWrite(led2l, HIGH);

  if (posactS1l >= 50)
  {
    for (pos1l = posactS1l; pos1l >=50; pos1l -=2)
```

```
  {
    servo1I.write(pos1I);
    delay (15);
  }
  posactS1I = pos1I;
  pos1I = 0;
}

pos3I = posactS5I;
for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
{
  pos3I +=2;
  if (pos2I >= 50)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
    sensorF1I = analogRead (A0);
    voltajeSF1I = sensorF1I * (5.0/1023.0);
    Serial.print (voltajeSF1I);
    if (voltajeSF1I < fuerza)
    {
      servo2I.write(pos2I);
      posSFR2I = pos2I;
    }
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
    sensorF2I = analogRead (A1);
    voltajeSF2I = sensorF2I * (5.0/1023.0);
    Serial.print (voltajeSF2I);
    if (voltajeSF2I < fuerza)
    {
      servo3I.write(pos2I);
      posSFR3I = pos2I;
    }
    // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
    sensorF3I = analogRead (A2);
    voltajeSF3I = sensorF3I * (5.0/1023.0);
    Serial.print (voltajeSF3I);
    if (voltajeSF3I < fuerza)
    {
      servo4I.write(pos2I);
      posSFR4I = pos2I;
    }
```

```
    }

    if (pos3I <= 100)
    {
      // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
      sensorF4I = analogRead (A3);
      voltajeSF4I = sensorF4I * (5.0/1023.0);
      Serial.print (voltajeSF4I);
      if (voltajeSF4I < fuerza)
      {
        servo5I.write(pos3I);
        posSFR5I = pos3I;
      }
    }

    if (pos2I >= 50)
    {
      // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
      sensorF5I = analogRead (A4);
      voltajeSF5I = sensorF5I * (5.0/1023.0);
      Serial.print (voltajeSF5I);
      if (voltajeSF5I < fuerza)
      {
        servo6I.write(pos2I);
        if (pos3I <= 100)
        {
          servo7I.write(pos3I);
        }
         posSFR6I = pos2I;
         posSFR7I = pos3I;
      }
    }
    // ****** servo8.write(pos2);
  }
  posactS2I = 50;
  posactS5I = 100;
  pos2I = 0;
  pos3I = 0;
  fuerza = 3.5;
  delay (15);
}
```

```
 else if (voltajeSEMG1I < 1.5)
 {
  digitalWrite(led1I, HIGH);
  digitalWrite(led2I, LOW);
  delay (5);
 }
delay (50);
 }
 }
 }
}

else if (EMGactivo == 0 && EMGinactivo == 1)
{
 for (time = 0; time < 5; time +=1)
 {
 for (mins = 0; mins < 5; mins +=1)
 {
 for (segs = 0; segs <= 1; segs +=1)
 {
  // ABRIR MANO DERECHA //////////////////////
  if (posactS1D >= 55)
  {
    for (pos1D = posactS1D; pos1D >=50; pos1D -=2)
    {
      servo1D.write(pos1D);
      delay (8);
    }
    posactS1D = pos1D;
    pos1D = 0;
  }
  pos3D = posactS5D;
  for (pos2D = posactS2D; pos2D >=10; pos2D -=2)
  {
    pos3D += 2;
    if (pos2D >= 50)
    {
      servo2D.write(pos2D);
      servo3D.write(pos2D);
      servo4D.write(pos2D);
```

```
    servo6D.write(pos2D);
    delay (5);
  }
  if (pos3D <= 120)
  {
    servo5D.write(pos3D);
    servo7D.write(pos3D);
    delay (5);
  }
  // ****** servo8.write(pos2);
}
posactS2D = 50;
posactS5D = 120;
pos2D = 0;
pos3D = 0;

// ********************************

// ABRIR MANO IZQUIERDA ////////////////////////
if (posactS1I <= 145)
{
  for (pos1I = posactS1I; pos1I <=150; pos1I +=2)
  {
    servo1I.write(pos1I);
    delay (8);
  }
posactS1I = pos1I;
pos1I = 0;
}
pos3I = posactS5I;
for (pos2I = posactS2I; pos2I <=150; pos2I +=2)
{
  pos3I -= 2;
  if (pos2I <=100)
  {
    servo4I.write(pos2I);
    servo2I.write(pos2I);
    delay (5);
  }
  if (pos2I <=110)
  {
```

```
    servo3I.write(pos2I);
    delay (5);
  }
  if (pos2I <=120)
  {
    servo6I.write(pos2I);
    delay (5);
  }
  if (pos3I >= 60)
  {
    servo7I.write(pos3I);
    delay (5);
  }
  if (pos3I >= 50)
  {
    servo5I.write(pos3I);
    delay (5);
  }
  // ****** servo8.write(pos2);
}
posactS2I = 100;
posactS5I = 60;
pos2I = 0;
pos3I = 0;
delay (1000);

// CERRAR MANO DERECHA //////////////////////
if (posactS1D <= 145)
{
  for (pos1D = posactS1D; pos1D <=145; pos1D +=2)
  {
    servo1D.write(pos1D);
    delay (5);
  }
  posactS1D = pos1D;
  pos1D = 0;
}

pos3D = posactS5D;
for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
{
```

```
pos3D -=2;
if (pos2D <= 100)
{
  // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
  sensorF1D = analogRead (A15);
  voltajeSF1D = sensorF1D * (5.0/1023.0);
  Serial.print (voltajeSF1D);
  if (voltajeSF1D < fuerza)
  {
    servo2D.write(pos2D);
    posSFR2D = pos2D;
  }
  // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
  sensorF2D = analogRead (A14);
  voltajeSF2D = sensorF2D * (5.0/1023.0);
  Serial.print (voltajeSF2D);
  if (voltajeSF2D < fuerza)
  {
    servo3D.write(pos2D);
    posSFR3D = pos2D;
  }
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
  sensorF3D = analogRead (A13);
  voltajeSF3D = sensorF3D * (5.0/1023.0);
  Serial.print (voltajeSF3D);
  if (voltajeSF3D < fuerza)
  {
    if (posactS2D <=90)
    {
      servo4D.write(pos2D);
      posSFR4D = pos2D;
    }
  }
}

if (pos3D >= 50)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
  sensorF4D = analogRead (A12);
  voltajeSF4D = sensorF4D * (5.0/1023.0);
  Serial.print (voltajeSF4D);
```

```
    if (voltajeSF4D < fuerza)
    {
      servo5D.write(pos3D);
      posSFR5D = pos3D;
    }
  }
  if (pos2D <= 120)
  {
    // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
    sensorF5D = analogRead (A11);
    voltajeSF5D = sensorF5D * (5.0/1023.0);
    Serial.print (voltajeSF5D);
    if (voltajeSF5D < fuerza)
    {
      servo6D.write(pos2D);
      if (pos3D >= 70)
      {
        servo7D.write(pos3D);
      }
      posSFR6D = pos2D;
      posSFR7D = pos3D;
    }
  }
  // ****** servo8.write(pos2);
}
posactS2D = 100;
posactS5D = 70;
pos2D = 0;
pos3D = 0;
fuerza = 3.5;

// CERRAR MANO IZQUIERDA ////////////////////////
if (posactS1I >= 50)
{
  for (pos1I = posactS1I; pos1I >=50; pos1I -=2)
  {
    servo1I.write(pos1I);
    delay (5);
  }
  posactS1I = pos1I;
  pos1I = 0;
```

```
}

pos3I = posactS5I;
for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
{
  pos3I +=2;
  if (pos2I >= 50)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
    sensorF1I = analogRead (A0);
    voltajeSF1I = sensorF1I * (5.0/1023.0);
    Serial.print (voltajeSF1I);
    if (voltajeSF1I < fuerza)
    {
      servo2I.write(pos2I);
      posSFR2I = pos2I;
    }
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
    sensorF2I = analogRead (A1);
    voltajeSF2I = sensorF2I * (5.0/1023.0);
    Serial.print (voltajeSF2I);

    if (voltajeSF2I < fuerza)
    {
      servo3I.write(pos2I);
      posSFR3I = pos2I;
    }
    // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
    sensorF3I = analogRead (A2);
    voltajeSF3I = sensorF3I * (5.0/1023.0);
    Serial.print (voltajeSF3I);
    if (voltajeSF3I < fuerza)
    {
      servo4I.write(pos2I);
      posSFR4I = pos2I;
    }
  }

  if (pos3I <= 100)
  {
    // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
```

```
      sensorF4I = analogRead (A3);
      voltajeSF4I = sensorF4I * (5.0/1023.0);
      Serial.print (voltajeSF4I);
      if (voltajeSF4I < fuerza)
      {
        servo5I.write(pos3I);
        posSFR5I = pos3I;
      }
    }

    if (pos2I >= 50)
    {
      // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
      sensorF5I = analogRead (A4);
      voltajeSF5I = sensorF5I * (5.0/1023.0);
      Serial.print (voltajeSF5I);
      if (voltajeSF5I < fuerza)
      {
        servo6I.write(pos2I);
        if (pos3I <= 100)
        {
          servo7I.write(pos3I);
        }
         posSFR6I = pos2I;
         posSFR7I = pos3I;
      }
    }
    // ****** servo8.write(pos2);
  }
  posactS2I = 50;
  posactS5I = 100;
  pos2I = 0;
  pos3I = 0;
  fuerza = 3.5;
  delay (1000);
 }
 }
 }
}

EMGactivo = 0;
```

```
    EMGinactivo = 0;
    voz = 2;
    group = GROUP_1;
    break;

//*****RUTINA PARA REHABILITACION POR QUINCE MINUTOS******
// ***************************************************************************************
    case G5_QUINCE:

    if (EMGactivo == 1 && EMGinactivo == 0)
    {
     for (time = 0; time < 15; time +=1)
     {
     for (mins = 0; mins < 59; mins +=1)
     {
     for (segs = 0; segs < 125; segs +=1)
     {
          sensorEMG1D = analogRead (A5);
     voltajeSEMG1D = sensorEMG1D * (5.0/1023.0);
     Serial.print (voltajeSEMG1D);
     sensorEMG1I = analogRead (A8);
     voltajeSEMG1I = sensorEMG1I * (5.0/1023.0);
     Serial.print (voltajeSEMG1I);

     if (voltajeSEMG1D > 2.5)
     {
       digitalWrite(led1D, LOW);
       digitalWrite(led2D, HIGH);
       if (posactS1D >= 55)
       {
         for (pos1D = posactS1D; pos1D >=50; pos1D -=2)
         {
           servo1D.write(pos1D);
           delay (15);
         }
         posactS1D = pos1D;
         pos1D = 0;
       }
       pos3D = posactS5D;
       for (pos2D = posactS2D; pos2D >=10; pos2D -=2)
       {
```

```
    pos3D += 2;
    if (pos2D >= 50)
    {
      servo2D.write(pos2D);
      servo3D.write(pos2D);
      servo4D.write(pos2D);
      servo6D.write(pos2D);
      delay (10);
    }
    if (pos3D <= 120)
    {
      servo5D.write(pos3D);
      servo7D.write(pos3D);
      delay (10);
    }
    // ****** servo8.write(pos2);
  }
  posactS2D = 50;
  posactS5D = 120;
  pos2D = 0;
  pos3D = 0;
  delay (15);
}

else if (voltajeSEMG1D >= 1.5 && voltajeSEMG1D <= 2.5)
{
  digitalWrite(led1D, LOW);
  digitalWrite(led2D, HIGH);
  if (posactS1D <= 145)
  {
    for (pos1D = posactS1D; pos1D <=150; pos1D +=2)
    {
      servo1D.write(pos1D);
      delay (15);
    }
    posactS1D = pos1D;
    pos1D = 0;
  }

  pos3D = posactS5D;
  for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
```

```
{
  pos3D -=2;
  if (pos2D <= 100)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
    sensorF1D = analogRead (A15);
    voltajeSF1D = sensorF1D * (5.0/1023.0);
    Serial.print (voltajeSF1D);
    if (voltajeSF1D < fuerza)
    {
      servo2D.write(pos2D);
      posSFR2D = pos2D;
    }
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
    sensorF2D = analogRead (A14);
    voltajeSF2D = sensorF2D * (5.0/1023.0);
    Serial.print (voltajeSF2D);
    if (voltajeSF2D < fuerza)
    {
      servo3D.write(pos2D);
      posSFR3D = pos2D;
    }
    // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
    sensorF3D = analogRead (A13);
    voltajeSF3D = sensorF3D * (5.0/1023.0);
    Serial.print (voltajeSF3D);
    if (voltajeSF3D < fuerza)
    {
      if (posactS2D <=90)
      {
        servo4D.write(pos2D);
        posSFR4D = pos2D;
      }
    }
  }

  if (pos3D >= 50)
  {
    // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
    sensorF4D = analogRead (A12);
    voltajeSF4D = sensorF4D * (5.0/1023.0);
```

```
      Serial.print (voltajeSF4D);
      if (voltajeSF4D < fuerza)
      {
        servo5D.write(pos3D);
        posSFR5D = pos3D;
      }
    }
    if (pos2D <= 120)
    {
      // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
      sensorF5D = analogRead (A11);
      voltajeSF5D = sensorF5D * (5.0/1023.0);
      Serial.print (voltajeSF5D);
      if (voltajeSF5D < fuerza)
      {
        servo6D.write(pos2D);
        if (pos3D >= 70)
        {
          servo7D.write(pos3D);
        }
        posSFR6D = pos2D;
        posSFR7D = pos3D;
      }
    }
    // ****** servo8.write(pos2);
  }
  posactS2D = 100;
  posactS5D = 70;
  pos2D = 0;
  pos3D = 0;
  fuerza = 3.5;
  delay (15);
}

else if (voltajeSEMG1D < 1.5)
{
 digitalWrite(led1D, HIGH);
 digitalWrite(led2D, LOW);
 delay (5);
}
```

```
if (voltajeSEMG1I > 2.5)
{
  digitalWrite(led1I, LOW);
  digitalWrite(led2I, HIGH);

  if (posactS1I <= 145)
  {
    for (pos1I = posactS1I; pos1I <=150; pos1I +=2)
    {
      servo1I.write(pos1I);
      delay (15);
    }
  posactS1I = pos1I;
  pos1I = 0;
  }
  pos3I = posactS5I;
  for (pos2I = posactS2I; pos2I <=150; pos2I +=2)
  {
    pos3I -= 2;
    if (pos2I <=100)
    {
      servo4I.write(pos2I);
      servo2I.write(pos2I);
      delay (5);
    }
    if (pos2I <=110)
    {
      servo3I.write(pos2I);
      delay (5);
    }
    if (pos2I <=120)
    {
      servo6I.write(pos2I);
      delay (5);
    }
    if (pos3I >= 60)
    {
      servo7I.write(pos3I);
      delay (5);
    }
    if (pos3I >= 50)
```

```
    {
      servo5I.write(pos3I);
      delay (5);
    }
    // ****** servo8.write(pos2);
  }
  posactS2I = 100;
  posactS5I = 60;
  pos2I = 0;
  pos3I = 0;
}

else if (voltajeSEMG1I > 1.5 && voltajeSEMG1I <= 2.5)
{
  digitalWrite(led1I, LOW);
  digitalWrite(led2I, HIGH);

  if (posactS1I >= 50)
  {
    for (pos1I = posactS1I; pos1I >=50; pos1I -=2)
    {
      servo1I.write(pos1I);
      delay (15);
    }
    posactS1I = pos1I;
    pos1I = 0;
  }

  pos3I = posactS5I;
  for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
  {
    pos3I +=2;
    if (pos2I >= 50)
    {
      // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
      sensorF1I = analogRead (A0);
      voltajeSF1I = sensorF1I * (5.0/1023.0);
      Serial.print (voltajeSF1I);
      if (voltajeSF1I < fuerza)
      {
        servo2I.write(pos2I);
```

```
    posSFR2I = pos2I;
  }
  // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
  sensorF2I = analogRead (A1);
  voltajeSF2I = sensorF2I * (5.0/1023.0);
  Serial.print (voltajeSF2I);
  if (voltajeSF2I < fuerza)
  {
    servo3I.write(pos2I);
    posSFR3I = pos2I;
  }
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
  sensorF3I = analogRead (A2);
  voltajeSF3I = sensorF3I * (5.0/1023.0);
  Serial.print (voltajeSF3I);
  if (voltajeSF3I < fuerza)
  {
    servo4I.write(pos2I);
    posSFR4I = pos2I;
  }
}

if (pos3I <= 100)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
  sensorF4I = analogRead (A3);
  voltajeSF4I = sensorF4I * (5.0/1023.0);
  Serial.print (voltajeSF4I);
  if (voltajeSF4I < fuerza)
  {
    servo5I.write(pos3I);
    posSFR5I = pos3I;
  }
}

if (pos2I >= 50)
{
  // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
  sensorF5I = analogRead (A4);
  voltajeSF5I = sensorF5I * (5.0/1023.0);
  Serial.print (voltajeSF5I);
```

```
        if (voltajeSF5I < fuerza)
        {
          servo6I.write(pos2I);
          if (pos3I <= 100)
          {
            servo7I.write(pos3I);
          }
           posSFR6I = pos2I;
           posSFR7I = pos3I;
        }
      }
      // ****** servo8.write(pos2);
    }
    posactS2I = 50;
    posactS5I = 100;
    pos2I = 0;
    pos3I = 0;
    fuerza = 3.5;
    delay (15);
  }

  else if (voltajeSEMG1I < 1.5)
  {
   digitalWrite(led1I, HIGH);
   digitalWrite(led2I, LOW);
   delay (5);
  }
delay (50);
  }
  }
  }
}

else if (EMGactivo == 0 && EMGinactivo == 1)
{
 for (time = 0; time < 5; time +=1)
 {
 for (mins = 0; mins < 5; mins +=1)
 {
 for (segs = 0; segs <= 1; segs +=1)
 {
```

```
// ABRIR MANO DERECHA ////////////////////////
if (posactS1D >= 55)
{
  for (pos1D = posactS1D; pos1D >=50; pos1D -=2)
  {
    servo1D.write(pos1D);
    delay (8);
  }
  posactS1D = pos1D;
  pos1D = 0;
}
pos3D = posactS5D;
for (pos2D = posactS2D; pos2D >=10; pos2D -=2)
{
  pos3D += 2;
  if (pos2D >= 50)
  {
    servo2D.write(pos2D);
    servo3D.write(pos2D);
    servo4D.write(pos2D);
    servo6D.write(pos2D);
    delay (5);
  }
  if (pos3D <= 120)
  {
    servo5D.write(pos3D);
    servo7D.write(pos3D);
    delay (5);
  }
  // ****** servo8.write(pos2);
}
posactS2D = 50;
posactS5D = 120;
pos2D = 0;
pos3D = 0;

// *********************************

// ABRIR MANO IZQUIERDA ////////////////////////
if (posactS1I <= 145)
{
```

```
  for (pos1l = posactS1l; pos1l <=150; pos1l +=2)
  {
    servo1l.write(pos1l);
    delay (8);
  }
posactS1l = pos1l;
pos1l = 0;
}
pos3l = posactS5l;
for (pos2l = posactS2l; pos2l <=150; pos2l +=2)
{
  pos3l -= 2;
  if (pos2l <=100)
  {
    servo4l.write(pos2l);
    servo2l.write(pos2l);
    delay (5);
  }
  if (pos2l <=110)
  {
    servo3l.write(pos2l);
    delay (5);
  }
  if (pos2l <=120)
  {
    servo6l.write(pos2l);
    delay (5);
  }
  if (pos3l >= 60)
  {
    servo7l.write(pos3l);
    delay (5);
  }
  if (pos3l >= 50)
  {
    servo5l.write(pos3l);
    delay (5);
  }
  // ****** servo8.write(pos2);
}
posactS2l = 100;
```

```
posactS5I = 60;
pos2I = 0;
pos3I = 0;
delay (1000);

// CERRAR MANO DERECHA !!!!!!!!!!!!!!!!!!!!!!
if (posactS1D <= 145)
{
  for (pos1D = posactS1D; pos1D <=145; pos1D +=2)
  {
    servo1D.write(pos1D);
    delay (5);
  }
  posactS1D = pos1D;
  pos1D = 0;
}

pos3D = posactS5D;
for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
{
  pos3D -=2;
  if (pos2D <= 100)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
    sensorF1D = analogRead (A15);
    voltajeSF1D = sensorF1D * (5.0/1023.0);
    Serial.print (voltajeSF1D);
    if (voltajeSF1D < fuerza)
    {
      servo2D.write(pos2D);
      posSFR2D = pos2D;
    }
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
    sensorF2D = analogRead (A14);
    voltajeSF2D = sensorF2D * (5.0/1023.0);
    Serial.print (voltajeSF2D);
    if (voltajeSF2D < fuerza)
    {
      servo3D.write(pos2D);
      posSFR3D = pos2D;
    }
```

```
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
  sensorF3D = analogRead (A13);
  voltajeSF3D = sensorF3D * (5.0/1023.0);
  Serial.print (voltajeSF3D);
  if (voltajeSF3D < fuerza)
  {
    if (posactS2D <=90)
    {
      servo4D.write(pos2D);
      posSFR4D = pos2D;
    }
  }
}

if (pos3D >= 50)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
  sensorF4D = analogRead (A12);
  voltajeSF4D = sensorF4D * (5.0/1023.0);
  Serial.print (voltajeSF4D);
  if (voltajeSF4D < fuerza)
  {
    servo5D.write(pos3D);
    posSFR5D = pos3D;
  }
}
if (pos2D <= 120)
{
  // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
  sensorF5D = analogRead (A11);
  voltajeSF5D = sensorF5D * (5.0/1023.0);
  Serial.print (voltajeSF5D);
  if (voltajeSF5D < fuerza)
  {
    servo6D.write(pos2D);
    if (pos3D >= 70)
    {
      servo7D.write(pos3D);
    }
    posSFR6D = pos2D;
    posSFR7D = pos3D;
```

```
    }
  }
  // ****** servo8.write(pos2);
}
posactS2D = 100;
posactS5D = 70;
pos2D = 0;
pos3D = 0;
fuerza = 3.5;

// CERRAR MANO IZQUIERDA ////////////////////////
if (posactS1I >= 50)
{
  for (pos1I = posactS1I; pos1I >=50; pos1I -=2)
  {
    servo1I.write(pos1I);
    delay (5);
  }
  posactS1I = pos1I;
  pos1I = 0;
}

pos3I = posactS5I;
for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
{
  pos3I +=2;
  if (pos2I >= 50)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
    sensorF1I = analogRead (A0);
    voltajeSF1I = sensorF1I * (5.0/1023.0);
    Serial.print (voltajeSF1I);
    if (voltajeSF1I < fuerza)
    {
      servo2I.write(pos2I);
      posSFR2I = pos2I;
    }
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
    sensorF2I = analogRead (A1);
    voltajeSF2I = sensorF2I * (5.0/1023.0);
    Serial.print (voltajeSF2I);
```

```
  if (voltajeSF2I < fuerza)
  {
    servo3I.write(pos2I);
    posSFR3I = pos2I;
  }
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
  sensorF3I = analogRead (A2);
  voltajeSF3I = sensorF3I * (5.0/1023.0);
  Serial.print (voltajeSF3I);
  if (voltajeSF3I < fuerza)
  {
    servo4I.write(pos2I);
    posSFR4I = pos2I;
  }
}

if (pos3I <= 100)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
  sensorF4I = analogRead (A3);
  voltajeSF4I = sensorF4I * (5.0/1023.0);
  Serial.print (voltajeSF4I);
  if (voltajeSF4I < fuerza)
  {
    servo5I.write(pos3I);
    posSFR5I = pos3I;
  }
}

if (pos2I >= 50)
{
  // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
  sensorF5I = analogRead (A4);
  voltajeSF5I = sensorF5I * (5.0/1023.0);
  Serial.print (voltajeSF5I);
  if (voltajeSF5I < fuerza)
  {
    servo6I.write(pos2I);
    if (pos3I <= 100)
    {
```

```
            servo7I.write(pos3I);
          }
           posSFR6I = pos2I;
           posSFR7I = pos3I;
        }
      }
      // ****** servo8.write(pos2);
    }
    posactS2I = 50;
    posactS5I = 100;
    pos2I = 0;
    pos3I = 0;
    fuerza = 3.5;
    delay (1000);
  }
  }
  }
}

EMGactivo = 0;
EMGinactivo = 0;
voz = 2;
group = GROUP_1;
break;

//*****RUTINA PARA REGRESAR AL GRUPO ANTERIOR*******
// ***********************************************************************
case G5_REINICIAR:
  voz = 1;
  group = GROUP_0;
  break;
}
break;

// ********************************************************

//*****RUTINA PARA SELECCIONAR OBJETO A AGARRAR*********
// ***********************************************************************
case GROUP_6:
  switch (idx)
  {
```

```
//*****RUTINA PARA AGARRAR UN LAPIZ*****************************
// ***************************************************************************
    case G6_LAPIZ:

     if (ManoD == 1 && ManoI == 0)
     {
       if (posactS1D >= 100)
       {
         for (pos1D = posactS1D; pos1D >=100; pos1D -=2)
         {
           servo1D.write(pos1D);
           delay (15);
         }
         posactS1D = pos1D;
         pos1D = 0;
       }

       pos3D = posactS5D;
       for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
       {
         pos3D -=2;
         if (pos2D <= 95)
         {
           // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
           sensorF1D = analogRead (A15);
           voltajeSF1D = sensorF1D * (5.0/1023.0);
           Serial.print (voltajeSF1D);
           if (voltajeSF1D < fuerza)
           {
             servo2D.write(pos2D);
             posSFR2D = pos2D;
           }
         }
         if (pos2D <= 80)
         {
           // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
           sensorF2D = analogRead (A14);
           voltajeSF2D = sensorF2D * (5.0/1023.0);
           Serial.print (voltajeSF2D);
           if (voltajeSF2D < fuerza)
```

```
  {
    servo3D.write(pos2D);
    posSFR3D = pos2D;
  }
}
if (pos2D <= 90
{
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
  sensorF3D = analogRead (A13);
  voltajeSF3D = sensorF3D * (5.0/1023.0);
  Serial.print (voltajeSF3D);
  if (voltajeSF3D < fuerza)
  {
    if (posactS2D <=90)
    {
      servo4D.write(pos2D);
      posSFR4D = pos2D;
    }
  }

}

if (pos3D >= 60)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
  sensorF4D = analogRead (A12);
  voltajeSF4D = sensorF4D * (5.0/1023.0);
  Serial.print (voltajeSF4D);
  if (voltajeSF4D < fuerza)
  {
    servo5D.write(pos3D);
    posSFR5D = pos3D;
  }
}
if (pos2D <= 120)
{
  // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
  sensorF5D = analogRead (A11);
  voltajeSF5D = sensorF5D * (5.0/1023.0);
  Serial.print (voltajeSF5D);
  if (voltajeSF5D < fuerza)
```

```
      {
        servo6D.write(pos2D);
        if (pos3D >= 90)
        {
          servo7D.write(pos3D);
        }
        posSFR6D = pos2D;
        posSFR7D = pos3D;
      }
    }
    // ****** servo8.write(pos2);
  }
  posactS2D = 90;
  posactS5D = 90;
  pos2D = 0;
  pos3D = 0;
  fuerza = 3.5;
  delay (15);
}

else if (ManoD == 0 && ManoI == 1)
{
  if (posactS1I <= 130)
  {
    for (pos1I = posactS1I; pos1I <=130; pos1I +=2)
    {
      servo1I.write(pos1I);
      delay (15);
    }
    posactS1I = pos1I;
    pos1I = 0;
  }

  pos3I = posactS5I;
  for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
  {
    pos3I +=2;
    if (pos2I >= 50)
    {
      // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
      sensorF1I = analogRead (A0);
```

```
  voltajeSF1I = sensorF1I * (5.0/1023.0);
  Serial.print (voltajeSF1I);
  if (voltajeSF1I < fuerza)
  {
    servo2I.write(pos2I);
    posSFR2I = pos2I;
  }
 }
 if (pos2I >= 60)
 {
  // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
  sensorF2I = analogRead (A1);
  voltajeSF2I = sensorF2I * (5.0/1023.0);
  Serial.print (voltajeSF2I);
  if (voltajeSF2I < fuerza)
  {
    servo3I.write(pos2I);
    posSFR3I = pos2I;
  }
}
if (pos2I >= 60)
{
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
  sensorF3I = analogRead (A2);
  voltajeSF3I = sensorF3I * (5.0/1023.0);
  Serial.print (voltajeSF3I);
  if (voltajeSF3I < fuerza)
  {
    servo4I.write(pos2I);
    posSFR4I = pos2I;
  }
}

if (pos3I <= 120)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
  sensorF4I = analogRead (A3);
  voltajeSF4I = sensorF4I * (5.0/1023.0);
  Serial.print (voltajeSF4I);
  if (voltajeSF4I < fuerza)
  {
```

```
        servo5I.write(pos3I);
        posSFR5I = pos3I;
      }
    }
    if (pos2I >= 60)
    {
      // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
      sensorF5I = analogRead (A4);
      voltajeSF5I = sensorF5I * (5.0/1023.0);
      Serial.print (voltajeSF5I);
      if (voltajeSF5I < fuerza)
      {
        servo6I.write(pos2I);
        if (pos3I <= 90)
        {
          servo7I.write(pos3I);
        }
         posSFR6I = pos2I;
         posSFR7I = pos3I;
      }
    }
    // ****** servo8.write(pos2);
  }
  posactS2I = 60;
  posactS5I = 90;
  pos2I = 0;
  pos3I = 0;
  fuerza = 3.5;
  delay (15);
}

else if (ManoD == 1 && ManoI == 1)
{
  if (posactS1D >= 100)
  {
    for (pos1D = posactS1D; pos1D >=100; pos1D -=2)
    {
      servo1D.write(pos1D);
      delay (15);
    }
    posactS1D = pos1D;
```

```
  pos1D = 0;
}

pos3D = posactS5D;
for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
{
  pos3D -=2;
  if (pos2D <= 90)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
    sensorF1D = analogRead (A15);
    voltajeSF1D = sensorF1D * (5.0/1023.0);
    Serial.print (voltajeSF1D);
    if (voltajeSF1D < fuerza)
    {
      servo2D.write(pos2D);
      posSFR2D = pos2D;
    }
  }
  if (pos2D <= 90)
  {
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
    sensorF2D = analogRead (A14);
    voltajeSF2D = sensorF2D * (5.0/1023.0);
    Serial.print (voltajeSF2D);
    if (voltajeSF2D < fuerza)
    {
      servo3D.write(pos2D);
      posSFR3D = pos2D;
    }
  }
  if (pos2D <= 90)
  {
    // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
    sensorF3D = analogRead (A13);
    voltajeSF3D = sensorF3D * (5.0/1023.0);
    Serial.print (voltajeSF3D);
    if (voltajeSF3D < fuerza)
    {
      if (posactS2D <=90)
      {
```

```
        servo4D.write(pos2D);
        posSFR4D = pos2D;
      }
    }
  }

  if (pos3D >= 60)
  {
    // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
    sensorF4D = analogRead (A12);
    voltajeSF4D = sensorF4D * (5.0/1023.0);
    Serial.print (voltajeSF4D);
    if (voltajeSF4D < fuerza)
    {
      servo5D.write(pos3D);
      posSFR5D = pos3D;
    }
  }
  if (pos2D <= 120)
  {
    // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
    sensorF5D = analogRead (A11);
    voltajeSF5D = sensorF5D * (5.0/1023.0);
    Serial.print (voltajeSF5D);
    if (voltajeSF5D < fuerza)
    {
      servo6D.write(pos2D);
      if (pos3D >= 90)
      {
        servo7D.write(pos3D);
      }
      posSFR6D = pos2D;
      posSFR7D = pos3D;
    }
  }
  // ****** servo8.write(pos2);
}
posactS2D = 90;
posactS5D = 90;
pos2D = 0;
pos3D = 0;
```

```
fuerza = 3.5;
delay (15);


if (posactS1I <= 100)
{
  for (pos1I = posactS1I; pos1I <=100; pos1I +=2)
  {
    servo1I.write(pos1I);
    delay (15);
  }
  posactS1I = pos1I;
  pos1I = 0;
}

pos3I = posactS5I;
for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
{
  pos3I +=2;
  if (pos2I >= 60)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
    sensorF1I = analogRead (A0);
    voltajeSF1I = sensorF1I * (5.0/1023.0);
    Serial.print (voltajeSF1I);
    if (voltajeSF1I < fuerza)
    {
      servo2I.write(pos2I);
      posSFR2I = pos2I;
    }
   }
   if (pos2I >= 60)
   {
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
    sensorF2I = analogRead (A1);
    voltajeSF2I = sensorF2I * (5.0/1023.0);
    Serial.print (voltajeSF2I);
    if (voltajeSF2I < fuerza)
    {
      servo3I.write(pos2I);
      posSFR3I = pos2I;
```

```
  }
}
if (pos2l >= 60)
{
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
  sensorF3l = analogRead (A2);
  voltajeSF3l = sensorF3l * (5.0/1023.0);
  Serial.print (voltajeSF3l);
  if (voltajeSF3l < fuerza)
  {
    servo4l.write(pos2l);
    posSFR4l = pos2l;
  }
}

if (pos3l <= 90)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
  sensorF4l = analogRead (A3);
  voltajeSF4l = sensorF4l * (5.0/1023.0);
  Serial.print (voltajeSF4l);
  if (voltajeSF4l < fuerza)
  {
    servo5l.write(pos3l);
    posSFR5l = pos3l;
  }
}

if (pos2l >= 60)
{
  // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
  sensorF5l = analogRead (A4);
  voltajeSF5l = sensorF5l * (5.0/1023.0);
  Serial.print (voltajeSF5l);
  if (voltajeSF5l < fuerza)
  {
    servo6l.write(pos2l);
    if (pos3l <= 90)
    {
      servo7l.write(pos3l);
    }
```

```
            posSFR6I = pos2I;
            posSFR7I = pos3I;
          }
        }
        // ****** servo8.write(pos2);
      }
      posactS2I = 60;
      posactS5I = 90;
      pos2I = 0;
      pos3I = 0;
      fuerza = 3.5;
      delay (15);
    }
    voz = 0;
    group = GROUP_3;
    break;

//*****RUTINA PARA AGARRAR UNA CUCHARA*********************
// ***************************************************************************
  case G6_CUCHARA:

    if (ManoD == 1 && ManoI == 0)
    {
      if (posactS1D >= 100)
      {
        for (pos1D = posactS1D; pos1D >=100; pos1D -=2)
        {
          servo1D.write(pos1D);
          delay (15);
        }
        posactS1D = pos1D;
        pos1D = 0;
      }

      pos3D = posactS5D;
      for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
      {
        pos3D -=2;
        if (pos2D <= 95)
        {
          // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
```

```
  sensorF1D = analogRead (A15);
  voltajeSF1D = sensorF1D * (5.0/1023.0);
  Serial.print (voltajeSF1D);
  if (voltajeSF1D < fuerza)
  {
    servo2D.write(pos2D);
    posSFR2D = pos2D;
  }
}
if (pos2D <= 80)
{
  // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
  sensorF2D = analogRead (A14);
  voltajeSF2D = sensorF2D * (5.0/1023.0);
  Serial.print (voltajeSF2D);
  if (voltajeSF2D < fuerza)
  {
    servo3D.write(pos2D);
    posSFR3D = pos2D;
  }
}
if (pos2D <= 90
{
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
  sensorF3D = analogRead (A13);
  voltajeSF3D = sensorF3D * (5.0/1023.0);
  Serial.print (voltajeSF3D);
  if (voltajeSF3D < fuerza)
  {
    if (posactS2D <=90)
    {
      servo4D.write(pos2D);
      posSFR4D = pos2D;
    }
  }

}

if (pos3D >= 60)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
```

```
      sensorF4D = analogRead (A12);
      voltajeSF4D = sensorF4D * (5.0/1023.0);
      Serial.print (voltajeSF4D);
      if (voltajeSF4D < fuerza)
      {
        servo5D.write(pos3D);
        posSFR5D = pos3D;
      }
    }
    if (pos2D <= 120)
    {
      // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
      sensorF5D = analogRead (A11);
      voltajeSF5D = sensorF5D * (5.0/1023.0);
      Serial.print (voltajeSF5D);
      if (voltajeSF5D < fuerza)
      {
        servo6D.write(pos2D);
        if (pos3D >= 90)
        {
          servo7D.write(pos3D);
        }
        posSFR6D = pos2D;
        posSFR7D = pos3D;
      }
    }
    // ****** servo8.write(pos2);
  }
  posactS2D = 90;
  posactS5D = 90;
  pos2D = 0;
  pos3D = 0;
  fuerza = 3.5;
  delay (15);
}

else if (ManoD == 0 && ManoI == 1)
{
  if (posactS1I <= 130)
  {
    for (pos1I = posactS1I; pos1I <=130; pos1I +=2)
```

```
  {
    servo1I.write(pos1I);
    delay (15);
  }
  posactS1I = pos1I;
  pos1I = 0;
}

pos3I = posactS5I;
for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
{
  pos3I +=2;
  if (pos2I >= 50)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
    sensorF1I = analogRead (A0);
    voltajeSF1I = sensorF1I * (5.0/1023.0);
    Serial.print (voltajeSF1I);
    if (voltajeSF1I < fuerza)
    {
      servo2I.write(pos2I);
      posSFR2I = pos2I;
    }
   }
   if (pos2I >= 60)
   {
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
    sensorF2I = analogRead (A1);
    voltajeSF2I = sensorF2I * (5.0/1023.0);
    Serial.print (voltajeSF2I);
    if (voltajeSF2I < fuerza)
    {
      servo3I.write(pos2I);
      posSFR3I = pos2I;
    }
  }
  if (pos2I >= 60)
  {
    // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
    sensorF3I = analogRead (A2);
    voltajeSF3I = sensorF3I * (5.0/1023.0);
```

```
    Serial.print (voltajeSF3I);
    if (voltajeSF3I < fuerza)
    {
      servo4I.write(pos2I);
      posSFR4I = pos2I;
    }
  }

  if (pos3I <= 120)
  {
    // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
    sensorF4I = analogRead (A3);
    voltajeSF4I = sensorF4I * (5.0/1023.0);
    Serial.print (voltajeSF4I);
    if (voltajeSF4I < fuerza)
    {
      servo5I.write(pos3I);
      posSFR5I = pos3I;
    }
  }
  if (pos2I >= 60)
  {
    // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
    sensorF5I = analogRead (A4);
    voltajeSF5I = sensorF5I * (5.0/1023.0);
    Serial.print (voltajeSF5I);
    if (voltajeSF5I < fuerza)
    {
      servo6I.write(pos2I);
      if (pos3I <= 90)
      {
        servo7I.write(pos3I);
      }
       posSFR6I = pos2I;
       posSFR7I = pos3I;
    }
  }
  // ****** servo8.write(pos2);
}
posactS2I = 60;
posactS5I = 90;
```

```
  pos2I = 0;
  pos3I = 0;
  fuerza = 3.5;
  delay (15);
}

else if (ManoD == 1 && ManoI == 1)
{
  if (posactS1D >= 100)
  {
    for (pos1D = posactS1D; pos1D >=100; pos1D -=2)
    {
      servo1D.write(pos1D);
      delay (15);
    }
    posactS1D = pos1D;
    pos1D = 0;
  }

  pos3D = posactS5D;
  for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
  {
    pos3D -=2;
    if (pos2D <= 90)
    {
      // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
      sensorF1D = analogRead (A15);
      voltajeSF1D = sensorF1D * (5.0/1023.0);
      Serial.print (voltajeSF1D);
      if (voltajeSF1D < fuerza)
      {
        servo2D.write(pos2D);
        posSFR2D = pos2D;
      }
    }
    if (pos2D <= 90)
    {
      // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
      sensorF2D = analogRead (A14);
      voltajeSF2D = sensorF2D * (5.0/1023.0);
      Serial.print (voltajeSF2D);
```

```
  if (voltajeSF2D < fuerza)
  {
    servo3D.write(pos2D);
    posSFR3D = pos2D;
  }
}
if (pos2D <= 90)
{
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
  sensorF3D = analogRead (A13);
  voltajeSF3D = sensorF3D * (5.0/1023.0);
  Serial.print (voltajeSF3D);
  if (voltajeSF3D < fuerza)
  {
    if (posactS2D <=90)
    {
      servo4D.write(pos2D);
      posSFR4D = pos2D;
    }
  }
}

if (pos3D >= 60)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
  sensorF4D = analogRead (A12);
  voltajeSF4D = sensorF4D * (5.0/1023.0);
  Serial.print (voltajeSF4D);
  if (voltajeSF4D < fuerza)
  {
    servo5D.write(pos3D);
    posSFR5D = pos3D;
  }
}
if (pos2D <= 120)
{
  // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
  sensorF5D = analogRead (A11);
  voltajeSF5D = sensorF5D * (5.0/1023.0);
  Serial.print (voltajeSF5D);
  if (voltajeSF5D < fuerza)
```

```
    {
      servo6D.write(pos2D);
      if (pos3D >= 90)
      {
        servo7D.write(pos3D);
      }
      posSFR6D = pos2D;
      posSFR7D = pos3D;
    }
  }
  // ****** servo8.write(pos2);
}
posactS2D = 90;
posactS5D = 90;
pos2D = 0;
pos3D = 0;
fuerza = 3.5;
delay (15);


if (posactS1I <= 100)
{
  for (pos1I = posactS1I; pos1I <=100; pos1I +=2)
  {
    servo1I.write(pos1I);
    delay (15);
  }
  posactS1I = pos1I;
  pos1I = 0;
}

pos3I = posactS5I;
for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
{
  pos3I +=2;
  if (pos2I >= 60)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
    sensorF1I = analogRead (A0);
    voltajeSF1I = sensorF1I * (5.0/1023.0);
    Serial.print (voltajeSF1I);
```

```
  if (voltajeSF1I < fuerza)
  {
    servo2I.write(pos2I);
    posSFR2I = pos2I;
  }
 }
 if (pos2I >= 60)
 {
  // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
  sensorF2I = analogRead (A1);
  voltajeSF2I = sensorF2I * (5.0/1023.0);
  Serial.print (voltajeSF2I);
  if (voltajeSF2I < fuerza)
  {
    servo3I.write(pos2I);
    posSFR3I = pos2I;
  }
 }
 if (pos2I >= 60)
 {
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
  sensorF3I = analogRead (A2);
  voltajeSF3I = sensorF3I * (5.0/1023.0);
  Serial.print (voltajeSF3I);
  if (voltajeSF3I < fuerza)
  {
    servo4I.write(pos2I);
    posSFR4I = pos2I;
  }
 }

 if (pos3I <= 90)
 {
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
  sensorF4I = analogRead (A3);
  voltajeSF4I = sensorF4I * (5.0/1023.0);
  Serial.print (voltajeSF4I);
  if (voltajeSF4I < fuerza)
  {
    servo5I.write(pos3I);
    posSFR5I = pos3I;
```

```
        }
      }

      if (pos2I >= 60)
      {
        // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
        sensorF5I = analogRead (A4);
        voltajeSF5I = sensorF5I * (5.0/1023.0);
        Serial.print (voltajeSF5I);
        if (voltajeSF5I < fuerza)
        {
          servo6I.write(pos2I);
          if (pos3I <= 90)
          {
            servo7I.write(pos3I);
          }
           posSFR6I = pos2I;
           posSFR7I = pos3I;
        }
      }
      // ****** servo8.write(pos2);
    }
    posactS2I = 60;
    posactS5I = 90;
    pos2I = 0;
    pos3I = 0;
    fuerza = 3.5;
    delay (15);
  }

  voz = 0;
  group = GROUP_3;
  break;

//*****RUTINA PARA AGARRAR UN VASO*****************************
// ************************************************************************
  case G6_VASO:


  if (ManoD == 1 && ManoI == 0)
  {
```

```
if (posactS1D >= 145)
{
  for (pos1D = posactS1D; pos1D <=150; pos1D +=2)
  {
    servo1D.write(pos1D);
    delay (15);
  }
  posactS1D = pos1D;
  pos1D = 0;
}

pos3D = posactS5D;
for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
{
  pos3D -=2;
  if (pos2D <= 100)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
    sensorF1D = analogRead (A15);
    voltajeSF1D = sensorF1D * (5.0/1023.0);
    Serial.print (voltajeSF1D);
    if (voltajeSF1D < fuerza)
    {
      servo2D.write(pos2D);
      posSFR2D = pos2D;
    }
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
    sensorF2D = analogRead (A14);
    voltajeSF2D = sensorF2D * (5.0/1023.0);
    Serial.print (voltajeSF2D);
    if (voltajeSF2D < fuerza)
    {
      servo3D.write(pos2D);
      posSFR3D = pos2D;
    }
    // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
    sensorF3D = analogRead (A13);
    voltajeSF3D = sensorF3D * (5.0/1023.0);
    Serial.print (voltajeSF3D);
    if (voltajeSF3D < fuerza)
    {
```

```
      if (posactS2D <=90)
      {
        servo4D.write(pos2D);
        posSFR4D = pos2D;
      }
    }

  }

  if (pos3D >= 50)
  {
    // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
    sensorF4D = analogRead (A12);
    voltajeSF4D = sensorF4D * (5.0/1023.0);
    Serial.print (voltajeSF4D);
    if (voltajeSF4D < fuerza)
    {
      servo5D.write(pos3D);
      posSFR5D = pos3D;
    }
  }
  if (pos2D <= 120)
  {
    // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
    sensorF5D = analogRead (A11);
    voltajeSF5D = sensorF5D * (5.0/1023.0);
    Serial.print (voltajeSF5D);
    if (voltajeSF5D < fuerza)
    {
      servo6D.write(pos2D);
      if (pos3D >= 70)
      {
        servo7D.write(pos3D);
      }
      posSFR6D = pos2D;
      posSFR7D = pos3D;
    }
  }
  // ****** servo8.write(pos2);
}
posactS2D = 100;
```

```
  posactS5D = 70;
  pos2D = 0
  pos3D = 0;
  fuerza = 3.5;
  delay (15);
}

else if (ManoD == 0 && ManoI == 1)
{
  if (posactS1I >= 50)
  {
    for (pos1I = posactS1I; pos1I >=50; pos1I -=2)
    {
      servo1I.write(pos1I);
      delay (15);
    }
    posactS1I = pos1I;
    pos1I = 0;
  }

  pos3I = posactS5I;
  for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
  {
    pos3I +=2;
    if (pos2I >= 50)
    {
      // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
      sensorF1I = analogRead (A0);
      voltajeSF1I = sensorF1I * (5.0/1023.0);
      Serial.print (voltajeSF1I);
      if (voltajeSF1I < fuerza)
      {
        servo2I.write(pos2I);
        posSFR2I = pos2I;
      }
      // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
      sensorF2I = analogRead (A1);
      voltajeSF2I = sensorF2I * (5.0/1023.0);
      Serial.print (voltajeSF2I);
      if (voltajeSF2I < fuerza)
      {
```

```
    servo3I.write(pos2I);
    posSFR3I = pos2I;
  }
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
  sensorF3I = analogRead (A2);
  voltajeSF3I = sensorF3I * (5.0/1023.0);
  Serial.print (voltajeSF3I);
  if (voltajeSF3I < fuerza)
  {
    servo4I.write(pos2I);
    posSFR4I = pos2I;
  }
}

if (pos3I <= 100)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
  sensorF4I = analogRead (A3);
  voltajeSF4I = sensorF4I * (5.0/1023.0);
  Serial.print (voltajeSF4I);
  if (voltajeSF4I < fuerza)
  {
    servo5I.write(pos3I);
    posSFR5I = pos3I;
  }
}

if (pos2I >= 50)
{
  // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
  sensorF5I = analogRead (A4);
  voltajeSF5I = sensorF5I * (5.0/1023.0);
  Serial.print (voltajeSF5I);
  if (voltajeSF5I < fuerza)
  {
    servo6I.write(pos2I);
    if (pos3I <= 100)
    {
      servo7I.write(pos3I);
    }
     posSFR6I = pos2I;
```

```
        posSFR7I = pos3I;

    }
    // ****** servo8.write(pos2);
  }
  posactS2I = 60;
  posactS5I = 100;
  pos2I = 0;
  pos3I = 0;
  fuerza = 3.5;
  delay (15);
}

else if (ManoD == 1 && ManoI == 1)
{
  if (posactS1D <= 145)
  {
    for (pos1D = posactS1D; pos1D <=150; pos1D +=2)
    {
      servo1D.write(pos1D);
      delay (15);
    }
    posactS1D = pos1D;
    pos1D = 0;
  }

  pos3D = posactS5D;
  for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
  {
    pos3D -=2;
    if (pos2D <= 100)
    {
      // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
      sensorF1D = analogRead (A15);
      voltajeSF1D = sensorF1D * (5.0/1023.0);
      Serial.print (voltajeSF1D);
      if (voltajeSF1D < fuerza)
      {
        servo2D.write(pos2D);
        posSFR2D = pos2D;
      }
```

```
  // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
  sensorF2D = analogRead (A14);
  voltajeSF2D = sensorF2D * (5.0/1023.0);
  Serial.print (voltajeSF2D);
  if (voltajeSF2D < fuerza)
  {
    servo3D.write(pos2D);
    posSFR3D = pos2D;
  }
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
  sensorF3D = analogRead (A13);
  voltajeSF3D = sensorF3D * (5.0/1023.0);
  Serial.print (voltajeSF3D);
  if (voltajeSF3D < fuerza)
  {
    if (posactS2D <=90)
    {
      servo4D.write(pos2D);
      posSFR4D = pos2D;
    }
  }
}

if (pos3D >= 50)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
  sensorF4D = analogRead (A12);
  voltajeSF4D = sensorF4D * (5.0/1023.0);
  Serial.print (voltajeSF4D);
  if (voltajeSF4D < fuerza)
  {
    servo5D.write(pos3D);
    posSFR5D = pos3D;
  }
}
if (pos2D >= 50)
{
  // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
  sensorF5D = analogRead (A11);
  voltajeSF5D = sensorF5D * (5.0/1023.0);
  Serial.print (voltajeSF5D);
```

```
    if (voltajeSF5D < fuerza)
    {
      servo6D.write(pos2D);
      if (pos3D >= 70)
      {
        servo7D.write(pos3D);
      }
      posSFR6D = pos2D;
      posSFR7D = pos3D;
    }
  }
  // ****** servo8.write(pos2);
}
posactS2D = 100;
posactS5D = 70;
pos2D = 0;
pos3D = 0;
fuerza = 3.5;
delay (15);

if (posactS1I >= 50)
{
  for (pos1I = posactS1I; pos1I >=50; pos1I -=2)
  {
    servo1I.write(pos1I);
    delay (15);
  }
  posactS1I = pos1I;
  pos1I = 0;
}

pos3I = posactS5I;
for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
{
  pos3I +=2;
  if (pos2I >= 50)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
    sensorF1I = analogRead (A0);
    voltajeSF1I = sensorF1I * (5.0/1023.0);
    Serial.print (voltajeSF1I);
```

```
  if (voltajeSF1I < fuerza)
  {
    servo2I.write(pos2I);
    posSFR2I = pos2I;
  }
  // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
  sensorF2I = analogRead (A1);
  voltajeSF2I = sensorF2I * (5.0/1023.0);
  Serial.print (voltajeSF2I);
  if (voltajeSF2I < fuerza)
  {
    servo3I.write(pos2I);
    posSFR3I = pos2I;
  }
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
  sensorF3I = analogRead (A2);
  voltajeSF3I = sensorF3I * (5.0/1023.0);
  Serial.print (voltajeSF3I);
  if (voltajeSF3I < fuerza)
  {
    servo4I.write(pos2I);
    posSFR4I = pos2I;
  }
}

if (pos3I <= 100)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
  sensorF4I = analogRead (A3);
  voltajeSF4I = sensorF4I * (5.0/1023.0);
  Serial.print (voltajeSF4I);
  if (voltajeSF4I < fuerza)
  {
    servo5I.write(pos3I);
    posSFR5I = pos3I;
  }
}

if (pos2I >= 50)
{
  // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
```

```
            sensorF5I = analogRead (A4);
            voltajeSF5I = sensorF5I * (5.0/1023.0);
            Serial.print (voltajeSF5I);
            if (voltajeSF5I < fuerza)
            {
              servo6I.write(pos2I);
              if (pos3I <= 100)
              {
                servo7I.write(pos3I);
              }
               posSFR6I = pos2I;
               posSFR7I = pos3I;
            }
          }
          // ****** servo8.write(pos2);
        }
        posactS2I = 50;
        posactS5I = 100;
        pos2I = 0;
        pos3I = 0;
        fuerza = 3.5;
        delay (15);
      }
      voz = 0;
      group = GROUP_3;
      break;

//*****RUTINA PARA AGARRAR UN POCILLO************************
// ***************************************************************************
    case G6_POCILLO:

      if (ManoD == 1 && ManoI == 0)
      {
        if (posactS1D >= 130)
        {
          for (pos1D = posactS1D; pos1D >=130; pos1D -=2)
          {
            servo1D.write(pos1D);
            delay (15);
          }
          posactS1D = pos1D;
```

```
  pos1D = 0;
}

pos3D = posactS5D;
for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
{
  pos3D -=2;
  if (pos2D <= 90)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
    sensorF1D = analogRead (A15);
    voltajeSF1D = sensorF1D * (5.0/1023.0);
    Serial.print (voltajeSF1D);
    if (voltajeSF1D < fuerza)
    {
      servo2D.write(pos2D);
      posSFR2D = pos2D;
    }
  }
  if (pos2D <= 90)
  {
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
    sensorF2D = analogRead (A14);
    voltajeSF2D = sensorF2D * (5.0/1023.0);
    Serial.print (voltajeSF2D);
    if (voltajeSF2D < fuerza)
    {
      servo3D.write(pos2D);
      posSFR3D = pos2D;
    }
  }
  if (pos2D <= 95)
  {
    // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
    sensorF3D = analogRead (A13);
    voltajeSF3D = sensorF3D * (5.0/1023.0);
    Serial.print (voltajeSF3D);
    if (voltajeSF3D < fuerza)
    {
      if (posactS2D <=90)
      {
```

```
        servo4D.write(pos2D);
        posSFR4D = pos2D;
      }
    }

  }

  if (pos3D >= 50)
  {
    // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
    sensorF4D = analogRead (A12);
    voltajeSF4D = sensorF4D * (5.0/1023.0);
    Serial.print (voltajeSF4D);
    if (voltajeSF4D < fuerza)
    {
      servo5D.write(pos3D);
      posSFR5D = pos3D;
    }
  }
  if (pos2D >= 50)
  {
    // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
    sensorF5D = analogRead (A11);
    voltajeSF5D = sensorF5D * (5.0/1023.0);
    Serial.print (voltajeSF5D);
    if (voltajeSF5D < fuerza)
    {
      servo6D.write(pos2D);
      if (pos3D >= 110)
      {
        servo7D.write(pos3D);
      }
      posSFR6D = 50;
      posSFR7D = 100;
    }
  }
  // ****** servo8.write(pos2);
}
posactS2D = pos2D;
posactS5D = pos3D;
pos2D = 0;
```

```
  pos3D = 0
  fuerza = 3.5;
  delay (15);
}

else if (ManoD == 0 && ManoI == 1)
{
  if (posactS1I <= 70)
  {
    for (pos1I = posactS1I; pos1I <=70; pos1I +=2)
    {
      servo1I.write(pos1I);
      delay (15);
    }
    posactS1I = pos1I;
    pos1I = 0;
  }

  pos3I = posactS5I;
  for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
  {
    pos3I +=2;
    if (pos2I >= 60)
    {
      // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
      sensorF1I = analogRead (A0);
      voltajeSF1I = sensorF1I * (5.0/1023.0);
      Serial.print (voltajeSF1I);
      if (voltajeSF1I < fuerza)
      {
        servo2I.write(pos2I);
        posSFR2I = pos2I;
      }
    }
    if (pos2I >= 60)
    {
      // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
      sensorF2I = analogRead (A1);
      voltajeSF2I = sensorF2I * (5.0/1023.0);
      Serial.print (voltajeSF2I);
      if (voltajeSF2I < fuerza)
```

```
    {
      servo3I.write(pos2I);
      posSFR3I = pos2I;
    }
  }
  if (pos2I >= 55)
  {
    // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
    sensorF3I = analogRead (A2);
    voltajeSF3I = sensorF3I * (5.0/1023.0);
    Serial.print (voltajeSF3I);
    if (voltajeSF3I < fuerza)
    {
      servo4I.write(pos2I);
      posSFR4I = pos2I;
    }
  }

  if (pos3I <= 100)
  {
    // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
    sensorF4I = analogRead (A3);
    voltajeSF4I = sensorF4I * (5.0/1023.0);
    Serial.print (voltajeSF4I);
    if (voltajeSF4I < fuerza)
    {
      servo5I.write(pos3I);
      posSFR5I = pos3I;
    }
  }

  if (pos2I >= 60)
  {
    // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
    sensorF5I = analogRead (A4);
    voltajeSF5I = sensorF5I * (5.0/1023.0);
    Serial.print (voltajeSF5I);
    if (voltajeSF5I < fuerza)
    {
      servo6I.write(pos2I);
      if (pos3I <= 90)
```

```
            servo7I.write(pos3I);
          }
           posSFR6I = pos2I;
           posSFR7I = pos3I;
        }
      }
      // ****** servo8.write(pos2);
    }
    posactS2I = 60;
    posactS5I = 100;
    pos2I = 0;
    pos3I = 0;
    fuerza = 3.5;
    delay (15);
}

else if (ManoD == 1 && ManoI == 1)
{
  if (posactS1D >= 130)
  {
    for (pos1D = posactS1D; pos1D >=130; pos1D -=2)
    {
      servo1D.write(pos1D);
      delay (15);
    }
    posactS1D = pos1D;
    pos1D = 0;
  }

  pos3D = posactS5D;
  for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
  {
    pos3D -=2;
    if (pos2D <= 90)
    {
      // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
      sensorF1D = analogRead (A15);
      voltajeSF1D = sensorF1D * (5.0/1023.0);
      Serial.print (voltajeSF1D);
      if (voltajeSF1D < fuerza)
```

```
    {
      servo2D.write(pos2D);
      posSFR2D = pos2D;
    }
  }
  if (pos2D <= 90)
  {
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
    sensorF2D = analogRead (A14);
    voltajeSF2D = sensorF2D * (5.0/1023.0);
    Serial.print (voltajeSF2D);
    if (voltajeSF2D < fuerza)
    {
      servo3D.write(pos2D);
      posSFR3D = pos2D;
    }
  }
  if (pos2D <= 95)
  {
    // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
    sensorF3D = analogRead (A13);
    voltajeSF3D = sensorF3D * (5.0/1023.0);
    Serial.print (voltajeSF3D);
    if (voltajeSF3D < fuerza)
    {
      if (posactS2D <=90)
      {
        servo4D.write(pos2D);
        posSFR4D = pos2D;
      }
    }

  }

  if (pos3D >= 50)
  {
    // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
    sensorF4D = analogRead (A12);
    voltajeSF4D = sensorF4D * (5.0/1023.0);
    Serial.print (voltajeSF4D);
    if (voltajeSF4D < fuerza)
```

```
    {
      servo5D.write(pos3D);
      posSFR5D = pos3D;
    }
  }
  if (pos2D <= 60)
  {
    // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
    sensorF5D = analogRead (A11);
    voltajeSF5D = sensorF5D * (5.0/1023.0);
    Serial.print (voltajeSF5D);
    if (voltajeSF5D < fuerza)
    {
      servo6D.write(pos2D);
      if (pos3D >= 110)
      {
        servo7D.write(pos3D);
      }
      posSFR6D = pos2D;
      posSFR7D = pos3D;
    }
  }
  // ****** servo8.write(pos2);
}
posactS2D = 90;
posactS5D = 110;
pos2D = 0;
pos3D = 0;
fuerza = 3.5;
delay (15);

if (posactS1I <= 70)
{
  for (pos1I = posactS1I; pos1I <=70; pos1I +=2)
  {
    servo1I.write(pos1I);
    delay (15);
  }
  posactS1I = pos1I;
  pos1I = 0;
}
```

```
pos3I = posactS5I;
for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
{
  pos3I +=2;
  if (pos2I >= 60)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
    sensorF1I = analogRead (A0);
    voltajeSF1I = sensorF1I * (5.0/1023.0);
    Serial.print (voltajeSF1I);
    if (voltajeSF1I < fuerza)
    {
      servo2I.write(pos2I);
      posSFR2I = pos2I;
    }
  }
  if (pos2I >= 60)
  {
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
    sensorF2I = analogRead (A1);
    voltajeSF2I = sensorF2I * (5.0/1023.0);
    Serial.print (voltajeSF2I);
    if (voltajeSF2I < fuerza)
    {
      servo3I.write(pos2I);
      posSFR3I = pos2I;
    }
  }
  if (pos2I >= 55)
  {
    // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
    sensorF3I = analogRead (A2);
    voltajeSF3I = sensorF3I * (5.0/1023.0);
    Serial.print (voltajeSF3I);
    if (voltajeSF3I < fuerza)
    {
      servo4I.write(pos2I);
      posSFR4I = pos2I;
    }
  }
```

```
    if (pos3l <= 100)
    {
      // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
      sensorF4l = analogRead (A3);
      voltajeSF4l = sensorF4l * (5.0/1023.0);
      Serial.print (voltajeSF4l);
      if (voltajeSF4l < fuerza)
      {
        servo5l.write(pos3l);
        posSFR5l = pos3l;
      }
    }

    if (pos2l >= 60)
    {
      // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
      sensorF5l = analogRead (A4);
      voltajeSF5l = sensorF5l * (5.0/1023.0);
      Serial.print (voltajeSF5l);
      if (voltajeSF5l < fuerza)
      {
        servo6l.write(pos2l);
        if (pos3l <= 90)
        {
          servo7l.write(pos3l);
        }
         posSFR6l = pos2l;
         posSFR7l = pos3l;
      }
    }
    // ****** servo8.write(pos2);
  }
  posactS2l = 60;
  posactS5l = 100;
  pos2l = 0;
  pos3l = 0;
  fuerza = 3.5;
  delay (15);
}
voz = 0;
```

```
      group = GROUP_3;
      break;

//*****RUTINA PARA AGARRAR EN FORMA DE PINZA***************
// ***************************************************************************

    case G6_PINZA:

      if (ManoD == 1 && ManoI == 0)
      {
        if (posactS1D >= 140)
        {
          for (pos1D = posactS1D; pos1D >=140; pos1D -=2)
          {
            servo1D.write(pos1D);
            delay (15);
          }
          posactS1D = pos1D;
          pos1D = 0;
        }

        pos3D = posactS5D;
        for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
        {
          pos3D -=2;
          if (pos2D <= 100)
          {
            // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
            sensorF1D = analogRead (A15);
            voltajeSF1D = sensorF1D * (5.0/1023.0);
            Serial.print (voltajeSF1D);
            if (voltajeSF1D < fuerza)
            {
              servo2D.write(pos2D);
              posSFR2D = pos2D;
            }
          }
          if (pos2D <= 70)
          {
            // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
            sensorF2D = analogRead (A14);
```

```
  voltajeSF2D = sensorF2D * (5.0/1023.0);
  Serial.print (voltajeSF2D);
  if (voltajeSF2D < fuerza)
  {
    servo3D.write(pos2D);
    posSFR3D = pos2D;
  }
}
if (pos2D <= 90)
{
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
  sensorF3D = analogRead (A13);
  voltajeSF3D = sensorF3D * (5.0/1023.0);
  Serial.print (voltajeSF3D);
  if (voltajeSF3D < fuerza)
  {
    if (posactS2D <=90)
    {
      servo4D.write(pos2D);
      posSFR4D = pos2D;
    }
  }
}

if (pos3D >= 50)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
  sensorF4D = analogRead (A12);
  voltajeSF4D = sensorF4D * (5.0/1023.0);
  Serial.print (voltajeSF4D);
  if (voltajeSF4D < fuerza)
  {
    servo5D.write(pos3D);
    posSFR5D = pos3D;
  }
}
if (pos2D >= 60)
{
  // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
  sensorF5D = analogRead (A11)
  voltajeSF5D = sensorF5D * (5.0/1023.0);
```

```
      Serial.print (voltajeSF5D);
      if (voltajeSF5D < fuerza)
      {
        servo6D.write(pos2D);
        if (pos3D >= 80)
        {
          servo7D.write(pos3D);
        }
        posSFR6D = pos2D;
        posSFR7D = pos3D;
      }
    }
    // ****** servo8.write(pos2);
  }
  posactS2D = 90;
  posactS5D = 90;
  pos2D = 0;
  pos3D = 0;
  fuerza = 3.5;
  delay (15);
}

else if (ManoD == 0 && ManoI == 1)
{
  if (posactS1I <= 60)
  {
    for (pos1I = posactS1I; pos1I <=60; pos1I +=2)
    {
      servo1I.write(pos1I);
      delay (15);
    }
    posactS1I = pos1I;
    pos1I = 0;
  }

  pos3I = posactS5I;
  for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
  {
    pos3I +=2;
    if (pos2I >= 50)
    {
```

```
  // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
  sensorF1I = analogRead (A0);
  voltajeSF1I = sensorF1I * (5.0/1023.0);
  Serial.print (voltajeSF1I);
  if (voltajeSF1I < fuerza)
  {
    servo2I.write(pos2I);
    posSFR2I = pos2I;
  }
 }
 if (pos2I >= 80)
 {
  // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
  sensorF2I = analogRead (A1);
  voltajeSF2I = sensorF2I * (5.0/1023.0);
  Serial.print (voltajeSF2I);
  if (voltajeSF2I < fuerza)
  {
    servo3I.write(pos2I);
    posSFR3I = pos2I;
  }
}
if (pos2I >= 60)
{
  // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
  sensorF3I = analogRead (A2);
  voltajeSF3I = sensorF3I * (5.0/1023.0);
  Serial.print (voltajeSF3I);
  if (voltajeSF3I < fuerza)
  {
    servo4I.write(pos2I);
    posSFR4I = pos2I;
  }
}

if (pos3I <= 100)
{
  // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
  sensorF4I = analogRead (A3);
  voltajeSF4I = sensorF4I * (5.0/1023.0)
  Serial.print (voltajeSF4I);
```

```
      if (voltajeSF4I < fuerza)
      {
        servo5I.write(pos3I);
        posSFR5I = pos3I;
      }
    }

    if (pos2I >= 60)
    {
      // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
      sensorF5I = analogRead (A4);
      voltajeSF5I = sensorF5I * (5.0/1023.0);
      Serial.print (voltajeSF5I);
      if (voltajeSF5I < fuerza)
      {
        servo6I.write(pos2I);
        if (pos3I <= 70)
        {
          servo7I.write(pos3I);
        }
         posSFR6I = pos2I;
         posSFR7I = pos3I;
      }
    }
    // ****** servo8.write(pos2);
  }
  posactS2I = 90;
  posactS5I = 70;
  pos2I = 0;
  pos3I = 0;
  fuerza = 3.5;
  delay (15);
}

else if (ManoD == 1 && ManoI == 1)
{
  if (posactS1D >= 140)
  {
    for (pos1D = posactS1D; pos1D >=140; pos1D -=2)
    {
      servo1D.write(pos1D);
```

```
    delay (15);
  }
  posactS1D = pos1D;
  pos1D = 0;
}

pos3D = posactS5D;
for (pos2D = posactS2D; pos2D <=120; pos2D +=2)
{
  pos3D -=2;
  if (pos2D <= 100)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A15
    sensorF1D = analogRead (A15);
    voltajeSF1D = sensorF1D * (5.0/1023.0);
    Serial.print (voltajeSF1D);
    if (voltajeSF1D < fuerza)
    {
      servo2D.write(pos2D);
      posSFR2D = pos2D;
    }
  }
  if (pos2D <= 70)
  {
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A14
    sensorF2D = analogRead (A14);
    voltajeSF2D = sensorF2D * (5.0/1023.0);
    Serial.print (voltajeSF2D);
    if (voltajeSF2D < fuerza)
    {
      servo3D.write(pos2D);
      posSFR3D = pos2D;
    }
  }
  if (pos2D <= 90)
  {
    // SERVO 4 - SFR 3 - ENTRADA ANALOGA A13
    sensorF3D = analogRead (A13);
    voltajeSF3D = sensorF3D * (5.0/1023.0);
    Serial.print (voltajeSF3D);
    if (voltajeSF3D < fuerza)
```

```
    {
      if (posactS2D <=90)
      {
        servo4D.write(pos2D);
        posSFR4D = pos2D;
      }
    }
  }

  if (pos3D >= 50)
  {
    // SERVO 5 - SFR 4 - ENTRADA ANALOGA A12
    sensorF4D = analogRead (A12);
    voltajeSF4D = sensorF4D * (5.0/1023.0);
    Serial.print (voltajeSF4D);
    if (voltajeSF4D < fuerza)
    {
      servo5D.write(pos3D);
      posSFR5D = pos3D;
    }
  }
  if (pos2D >= 60)
  {
    // SERVO 6 - SFR 5 - ENTRADA ANALOGA A11
    sensorF5D = analogRead (A11);
    voltajeSF5D = sensorF5D * (5.0/1023.0);
    Serial.print (voltajeSF5D);
    if (voltajeSF5D < fuerza)
    {
      servo6D.write(pos2D);
      if (pos3D >= 80)
      {
        servo7D.write(pos3D);
      }
      posSFR6D = pos2D;
      posSFR7D = pos3D;
    }
  }
  // ****** servo8.write(pos2);
}
posactS2D = 90;
```

```
posactS5D = 60;
pos2D = 0;
pos3D = 0;
fuerza = 3.5;
delay (15);

if (posactS1I <= 60)
{
  for (pos1I = posactS1I; pos1I <=60; pos1I +=2)
  {
    servo1I.write(pos1I);
    delay (15);
  }
  posactS1I = pos1I;
  pos1I = 0;
}

pos3I = posactS5I;
for (pos2I = posactS2I; pos2I >=50; pos2I -=2)
{
  pos3I +=2;
  if (pos2I >= 50)
  {
    // SERVO 2 - SFR 1 - ENTRADA ANALOGA A0
    sensorF1I = analogRead (A0);
    voltajeSF1I = sensorF1I * (5.0/1023.0);
    Serial.print (voltajeSF1I);
    if (voltajeSF1I < fuerza)
    {
      servo2I.write(pos2I);
      posSFR2I = pos2I;
    }
   }
   if (pos2I >= 80)
   {
    // SERVO 3 - SFR 2 - ENTRADA ANALOGA A1
    sensorF2I = analogRead (A1);
    voltajeSF2I = sensorF2I * (5.0/1023.0);
    Serial.print (voltajeSF2I);
    if (voltajeSF2I < fuerza)
    {
```

```
      servo3I.write(pos2I);
      posSFR3I = pos2I;
    }
  }
  if (pos2I >= 60)
  {
    // SERVO 4 - SFR 3 - ENTRADA ANALOGA A2
    sensorF3I = analogRead (A2);
    voltajeSF3I = sensorF3I * (5.0/1023.0);
    Serial.print (voltajeSF3I);
    if (voltajeSF3I < fuerza)
    {
      servo4I.write(pos2I);
      posSFR4I = pos2I;
    }
  }

  if (pos3I <= 100)
  {
    // SERVO 5 - SFR 4 - ENTRADA ANALOGA A3
    sensorF4I = analogRead (A3);
    voltajeSF4I = sensorF4I * (5.0/1023.0);
    Serial.print (voltajeSF4I);
    if (voltajeSF4I < fuerza)
    {
      servo5I.write(pos3I);
      posSFR5I = pos3I;
    }
  }

  if (pos2I >= 60)
  {
    // SERVO 6 - SFR 5 - ENTRADA ANALOGA A4
    sensorF5I = analogRead (A4);
    voltajeSF5I = sensorF5I * (5.0/1023.0);
    Serial.print (voltajeSF5I);
    if (voltajeSF5I < fuerza)
    {
      servo6I.write(pos2I);
      if (pos3I <= 70)
      {
```

```
            servo7I.write(pos3I);
          }
           posSFR6I = pos2I;
           posSFR7I = pos3I;
        }
      }
      // ****** servo8.write(pos2);
    }

    posactS2I = 60;
    posactS5I = 70;
    pos2I = 0;
    pos3I = 0;
    fuerza = 3.5;
    delay (15);
  }
  voz = 0;
  group = GROUP_3;
  break;
```

```
/*** RUTINA PARA REGRESAR AL GRUPO ANTERIOR****
//*********************************************************************

  case G6_REINICIAR:

   voz = 1;
   group = GROUP_0;
   break;
  }
  break;
```

```
/*** RUTINA PARA DEFINIR TIPO DE REHABILITACIÓN**
//*********************************************************************

 case GROUP_7:
```

```
    switch (idx)
    {
    case G7_ACTUAR:
     EMGactivo = 1;
     EMGinactivo = 0;
     voz = 7;
     group = GROUP_5;
     break;

    case G7_PASIVA:
     EMGactivo = 0;
     EMGinactivo = 1;
     voz = 7;
     group = GROUP_5;
     break;
    }
    break;

/*** RUTINA PARA MANEJAR EL GIRO DE LA SILLA******
//*****************************************************************

//*****************************************************************

   case GROUP_8:

    switch (idx)
    {
    case G8_DERECHA:
     digitalWrite(leg1, HIGH);
     digitalWrite(leg2, LOW);
     delay (2);
     voz = 0;
     group = GROUP_8;
     break;

    case G8_IZQUIERDA:
     digitalWrite(leg1, LOW);
     digitalWrite(leg2, HIGH);
     delay (2);
     voz = 0;
     group = GROUP_8;
```

```
      break;

    case G8_DETENER:
      digitalWrite(leg1, LOW);
      digitalWrite(leg2, LOW);
      delay (2);
      voz = 9;
      group = GROUP_8;
      break;


    case G8_CONTINUAR:
      digitalWrite(leg1, LOW);
      digitalWrite(leg2, LOW);
      delay (2);
      voz = 2;
      group = GROUP_1;
      break;
    }
    break;
  }


//*****RUTINA PARA SOLTAR OBJETOS*******************************
// ***************************************************************************

//***************************************************************************

   case GROUP_9:

    switch (idx)
    {

    case G3_SOLTAR:

      if (ManoD == 1 && ManoI == 0)
      {
        pos3D = posactS5D;
        for (pos2D = posactS2D; pos2D >=10; pos2D -=2)
        {
          pos3D += 2;
```

```
    if (pos2D >= 50)
    {
      servo2D.write(pos2D);
      servo3D.write(pos2D);
      servo4D.write(pos2D);
      servo6D.write(pos2D);
      delay (10);
    }
    if (pos3D <= 120)
    {
      servo5D.write(pos3D);
      servo7D.write(pos3D);
      delay (10);
    }
    // ****** servo8.write(pos2);
  }
  posactS2D = 50;
  posactS5D = 120;
  pos2D = 0;
  pos3D = 0;
  delay (15);

  if (posactS1D >= 55)
  {
    for (pos1D = posactS1D; pos1D >=50; pos1D -=2)
    {
      servo1D.write(pos1D);
      delay (15);
    }
    posactS1D = pos1D;
    pos1D = 0;
  }
}

else if (ManoD == 0 && ManoI == 1)
{
  pos3I = posactS5I;
  for (pos2I = posactS2I; pos2I <=150; pos2I +=2)
  {
    pos3I -= 2;
    if (pos2I <=100)
```

```
  {
    servo4l.write(pos2l);
    servo2l.write(pos2l);
    delay (10);
  }
  if (pos2l <=110)
  {
    servo3l.write(pos2l);
    delay (5);
  }
  if (pos2l <=120)
  {
    servo6l.write(pos2l);
    delay (10);
  }
  if (pos3l >= 60)
  {
    servo7l.write(pos3l);
    delay (10);
  }
  if (pos3l >= 50)
  {
    servo5l.write(pos3l);
    delay (10);
  }
  // ****** servo8.write(pos2);
}
posactS2l = 100;
posactS5l = 60;
pos2l = 0;
pos3l = 0;
delay (15);
if (posactS1l <= 145)
{
  for (pos1l = posactS1l; pos1l <=150; pos1l +=2)
  {
    servo1l.write(pos1l);
    delay (15);
  }
posactS1l = pos1l;
pos1l = 0;
```

```
  }
}

else if (ManoD == 1 && ManoI == 1)
{
  pos3I = posactS5I;
  for (pos2I = posactS2I; pos2I <=150; pos2I +=2)
  {
    pos3I -= 2;
    if (pos2I <=100)
    {
      servo4I.write(pos2I);
      servo2I.write(pos2I);
      delay (10);
    }
    if (pos2I <=110)
    {
      servo3I.write(pos2I);
      delay (10);
    }
    if (pos2I <=120)
    {
      servo6I.write(pos2I);
      delay (10);
    }
    if (pos3I >= 60)
    {
      servo7I.write(pos3I);
      delay (10);
    }
    if (pos3I >= 50)
    {
      servo5I.write(pos3I);
      delay (10);
    }
    // ****** servo8.write(pos2);
  }
  posactS2I = 100;
  posactS5I = 60;
  pos2I = 0;
  pos3I = 0;
```

```
delay (15);
if (posactS1I <= 145)
{
  for (pos1I = posactS1I; pos1I <=150; pos1I +=2)
  {
    servo1I.write(pos1I);
    delay (15);
  }
  posactS1I = pos1I;
  pos1I = 0;
}

pos3D = posactS5D;
for (pos2D = posactS2D; pos2D >=10; pos2D -=2)
{
  pos3D += 2;
  if (pos2D >= 50)
  {
    servo2D.write(pos2D);
    servo3D.write(pos2D);
    servo4D.write(pos2D);
    servo6D.write(pos2D);
    delay (10);
  }
  if (pos3D <= 120)
  {
    servo5D.write(pos3D);
    servo7D.write(pos3D);
    delay (10);
  }
  // ****** servo8.write(pos2);
}
posactS2D = 50;
posactS5D = 120;
pos2D = 0;
pos3D = 0;
delay (15);

if (posactS1D >= 55)
{
  for (pos1D = posactS1D; pos1D >=50; pos1D -=2)
```

```
        {
          servo1D.write(pos1D);
          delay (15);
        }
        posactS1D = pos1D;
        pos1D = 0;
      }
    }
    voz = 0;
    group = GROUP_3;
    break;
  }
  break;
 }
}
```

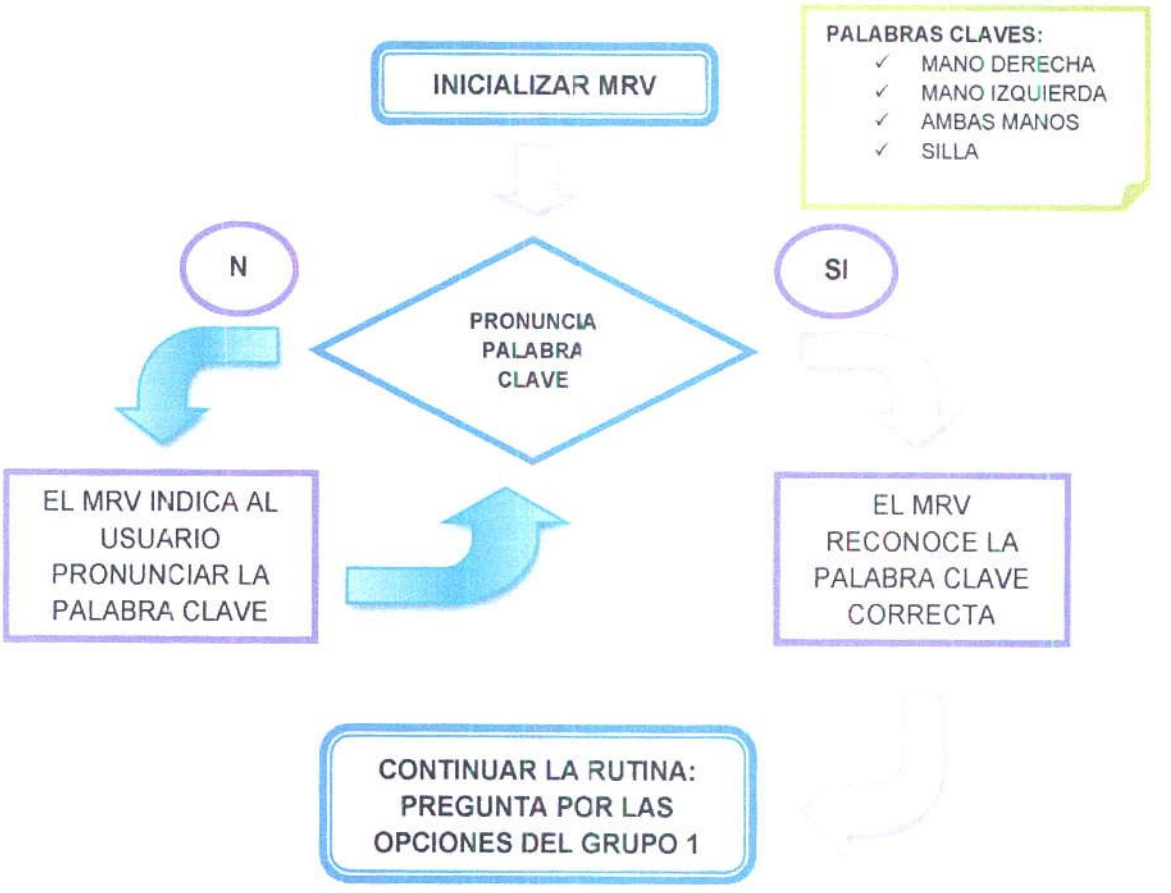
7 Flujogramas del código de programación para el control del MRV:

El Módulo de Reconocimiento de Voz (**MRV**) para el control del exoesqueleto prototipo EMMRA-1, ofrece una práctica programación de comunicación máquina – usuario, a través señales auditivas y visuales.

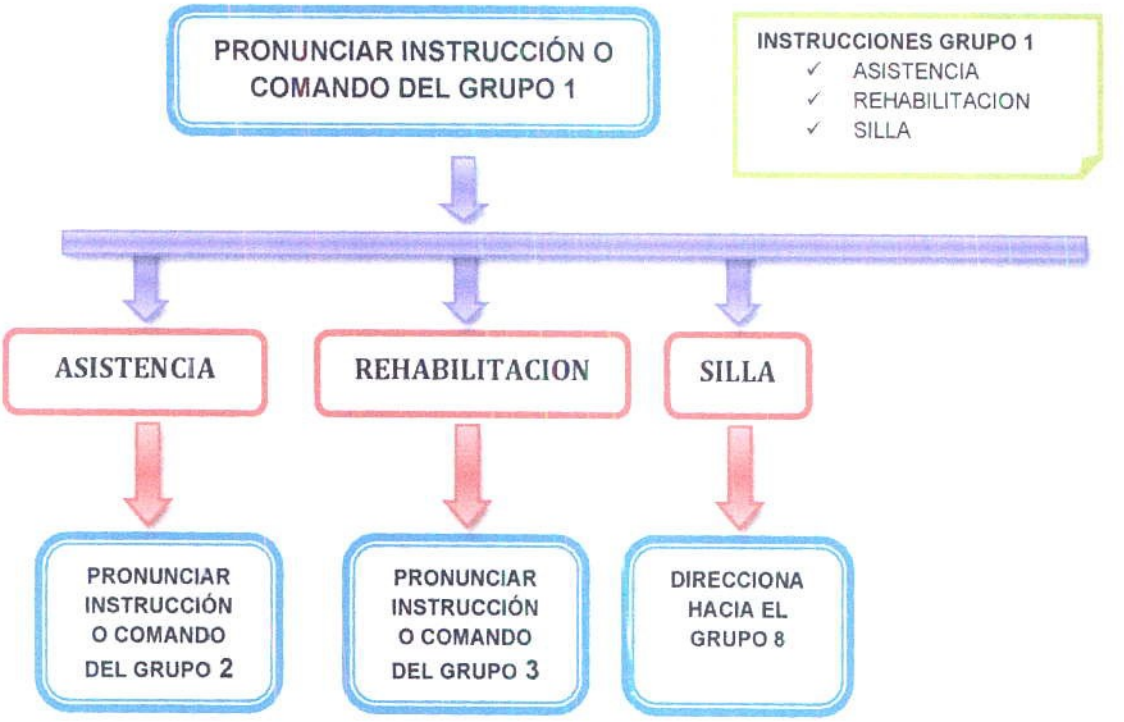
Las señalas auditivas corresponden a palabras en forma de preguntas (sonidos pregrabados) para indicarle al usuario que debe elegir una instrucción y pronunciarla en un tiempo determinado.

Las señales visuales corresponden a LED Verde para indicarle al usuario cuando pronunciar una instrucción y al LED Rojo para indicarle al usuario que el MRV está procesando información y no debe pronunciar ninguna instrucción.

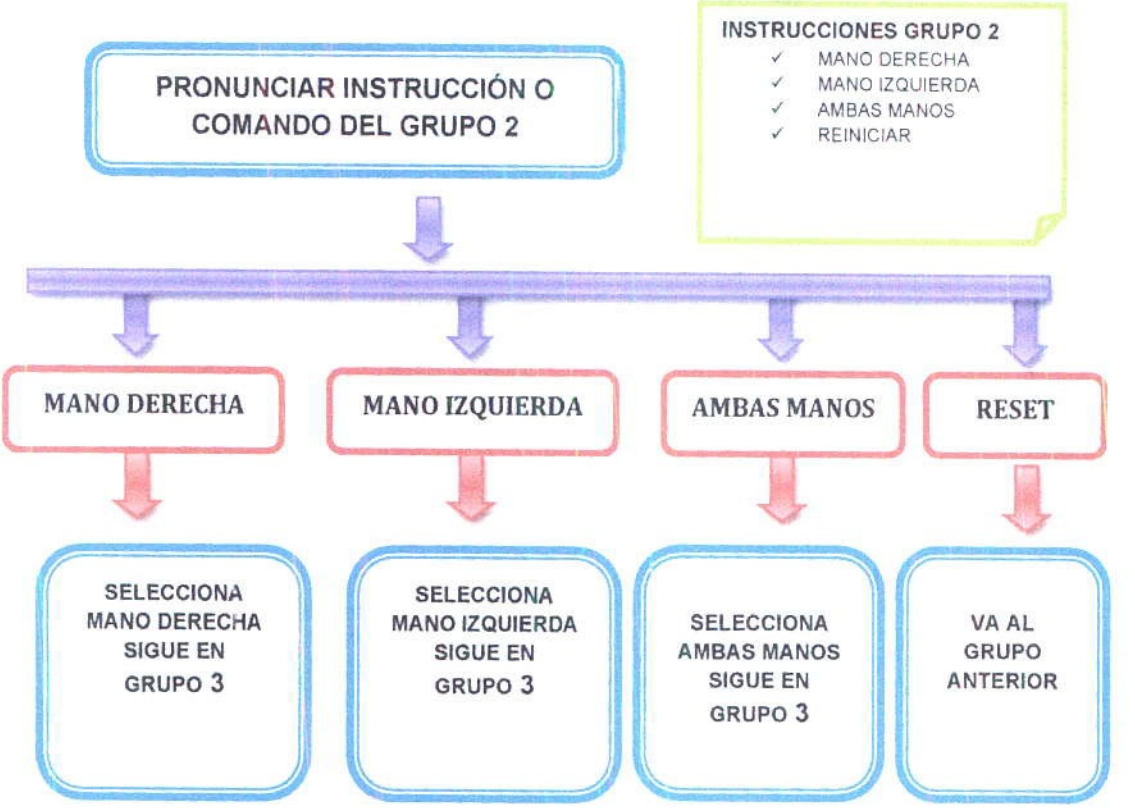
- **Flujograma para inicializar el programa y la selección de grupos**



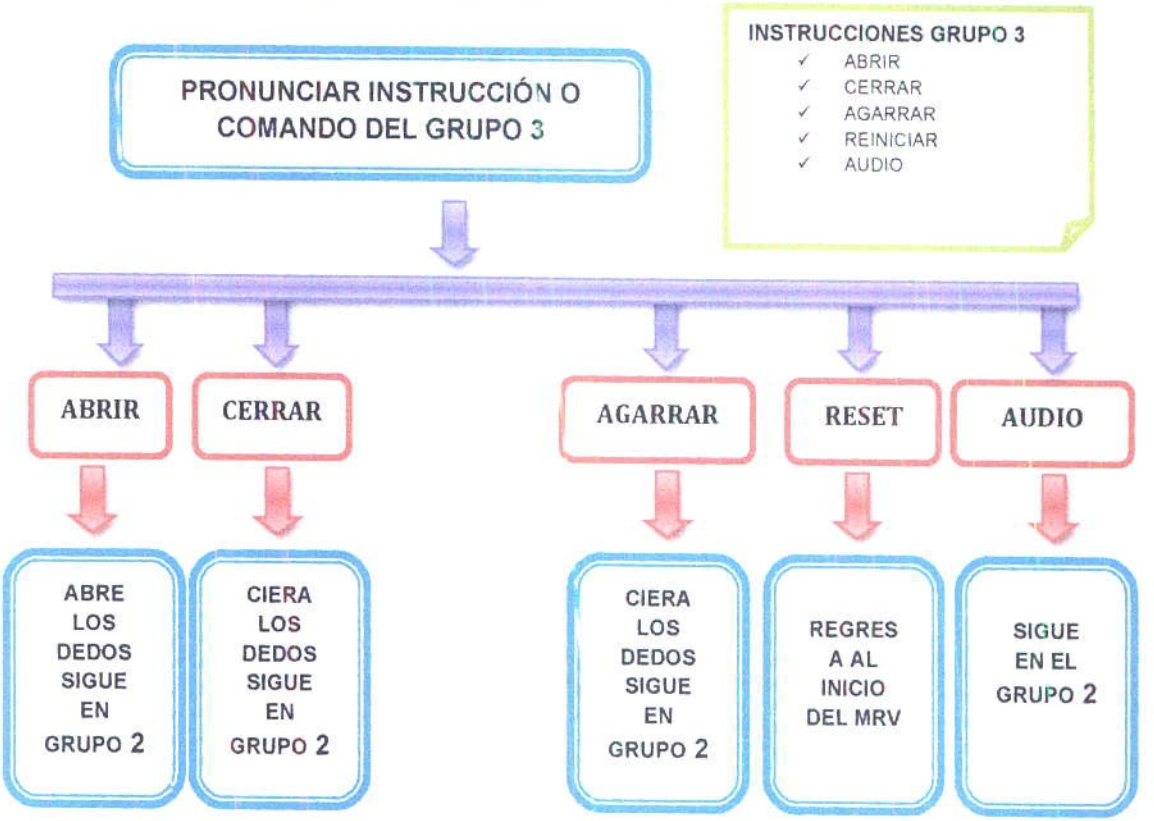
- Flujograma para la selección de comandos del GRUPO 1



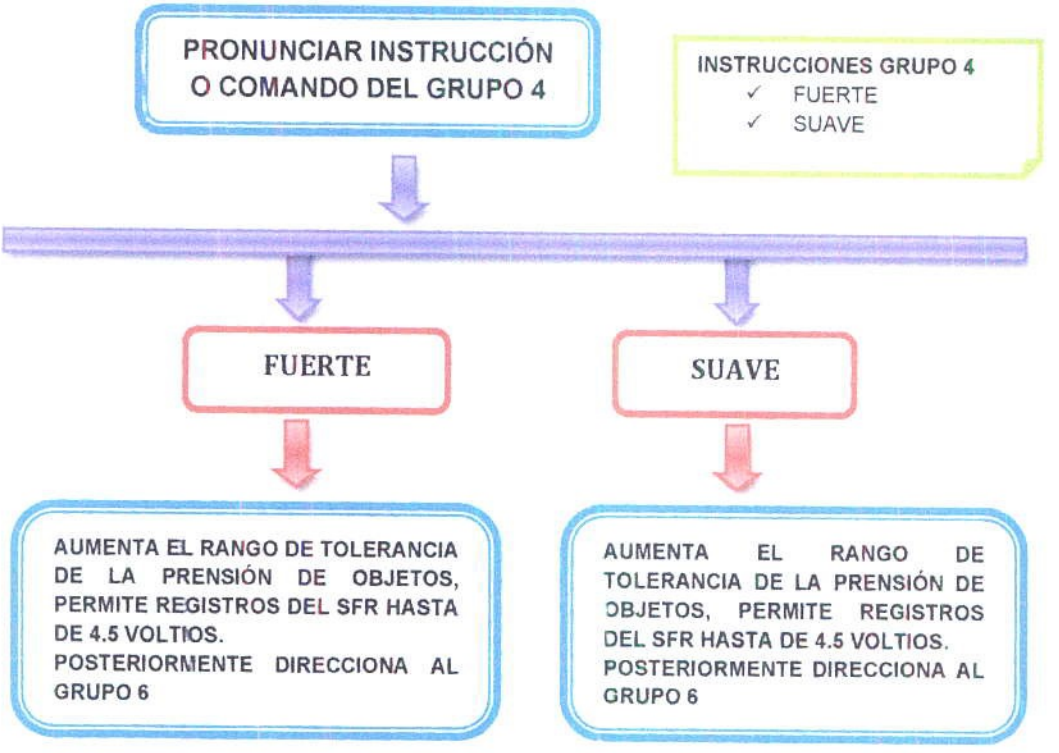
- Flujograma para la selección de comandos del GRUPO 2



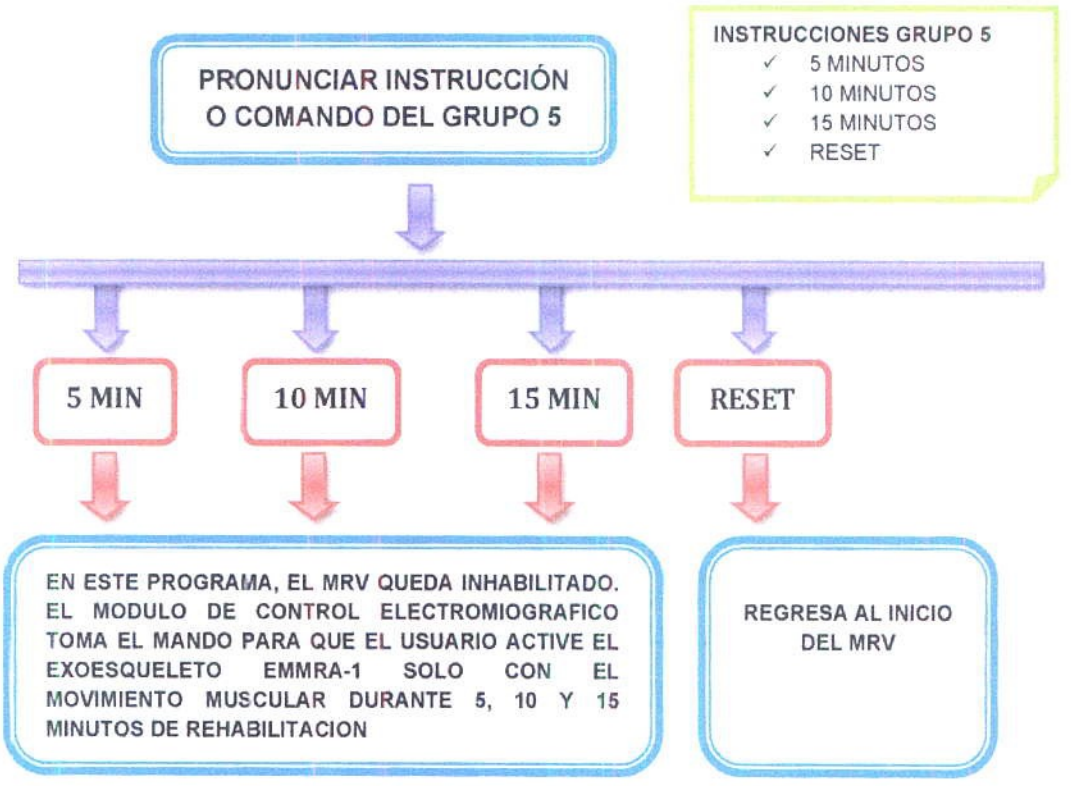
- Flujograma para la selección de comandos del GRUPO 3



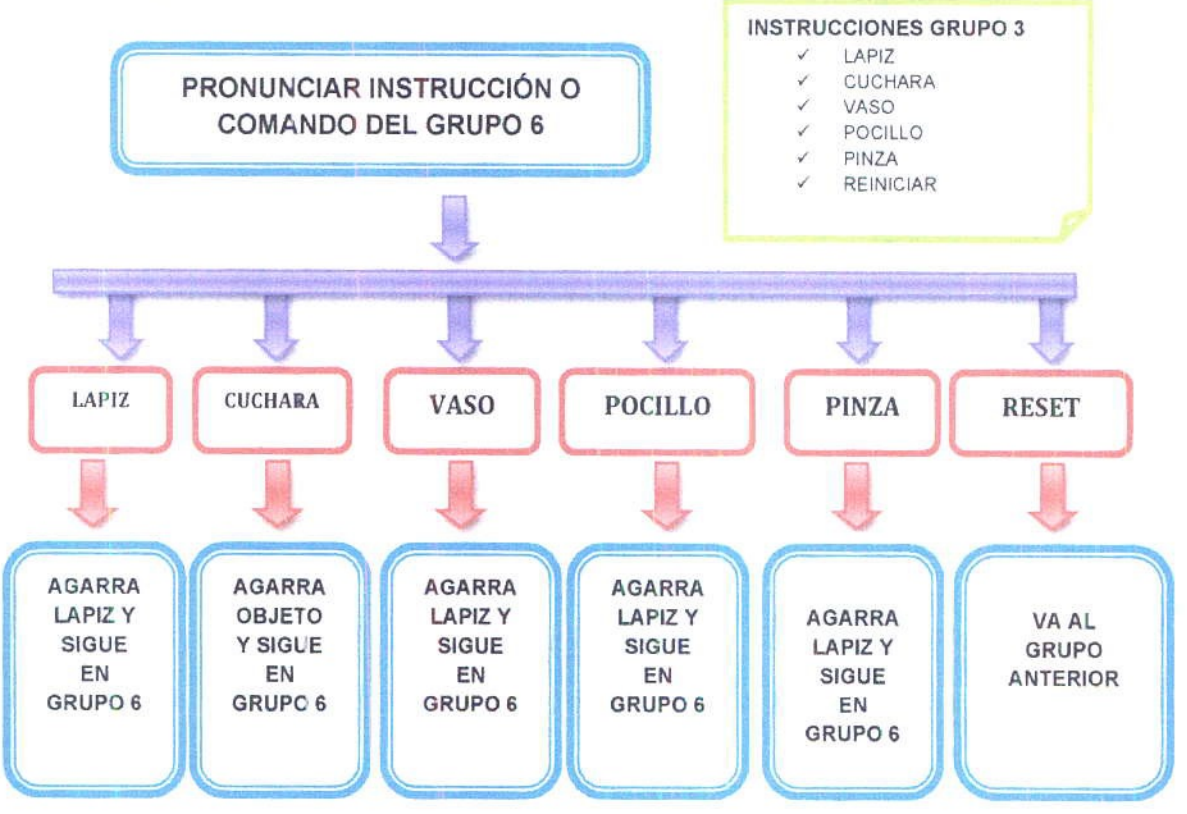
- Flujograma para la selección de comandos del GRUPO 4



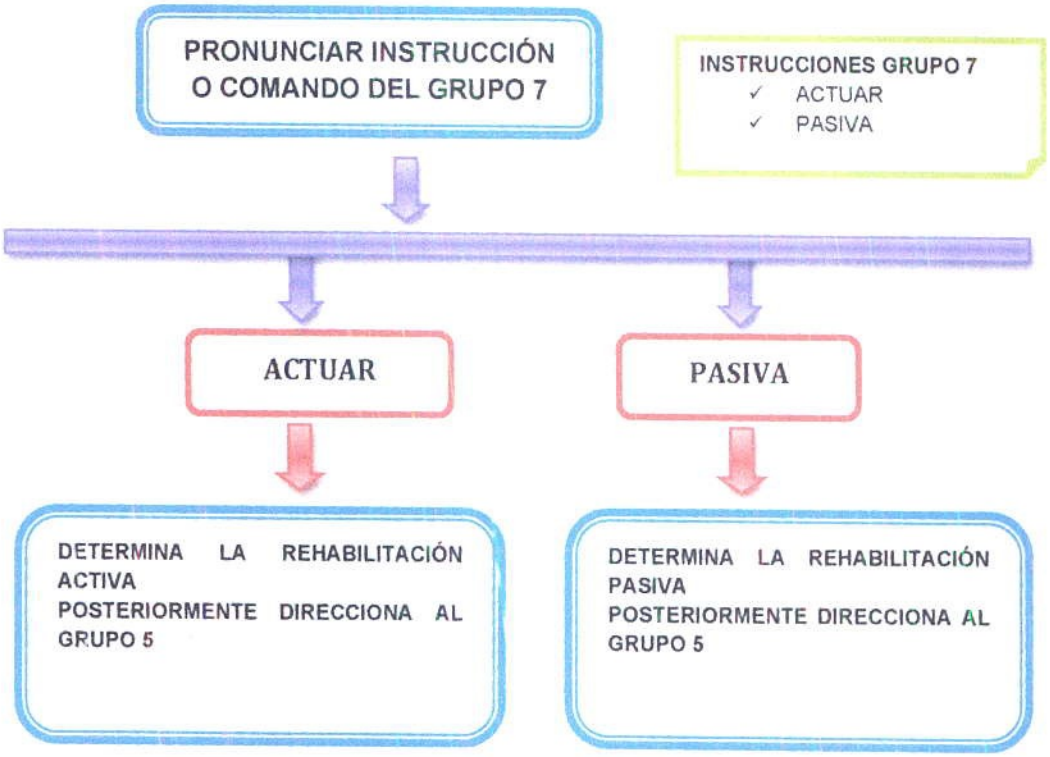
- Flujograma para la selección de comandos del GRUPO 5



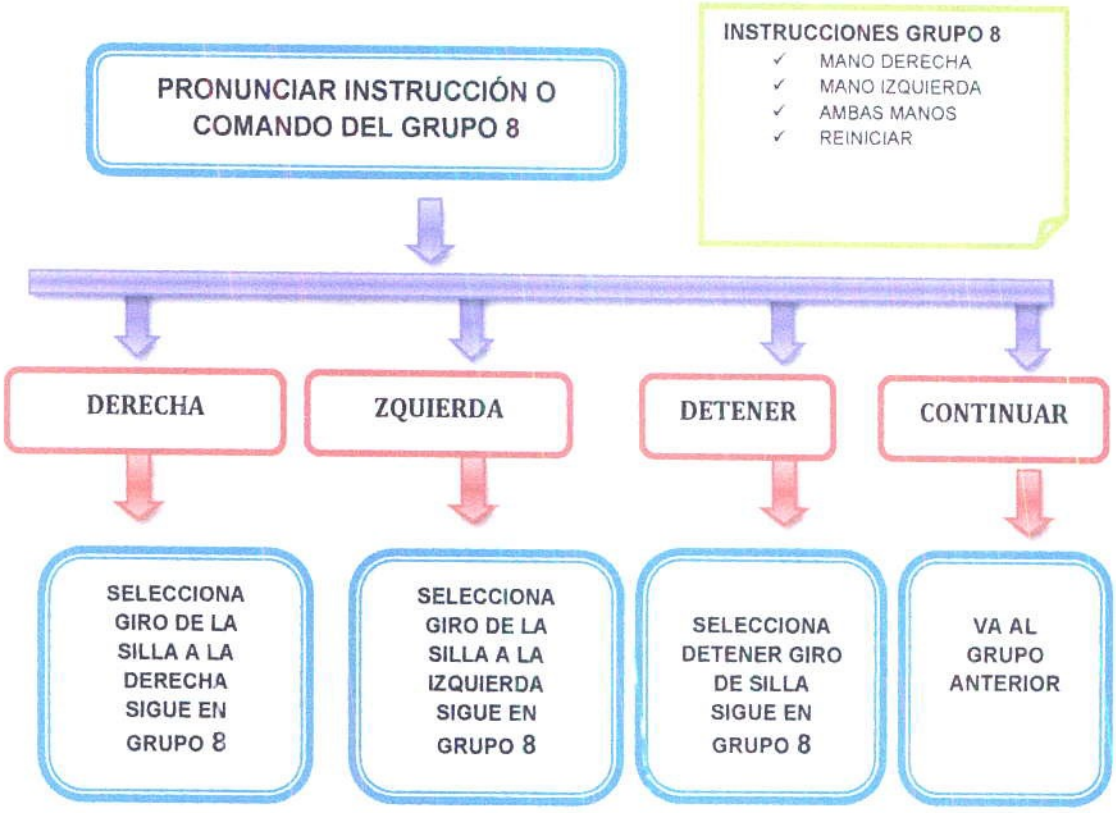
- Flujograma para la selección de comandos del GRUPO 6



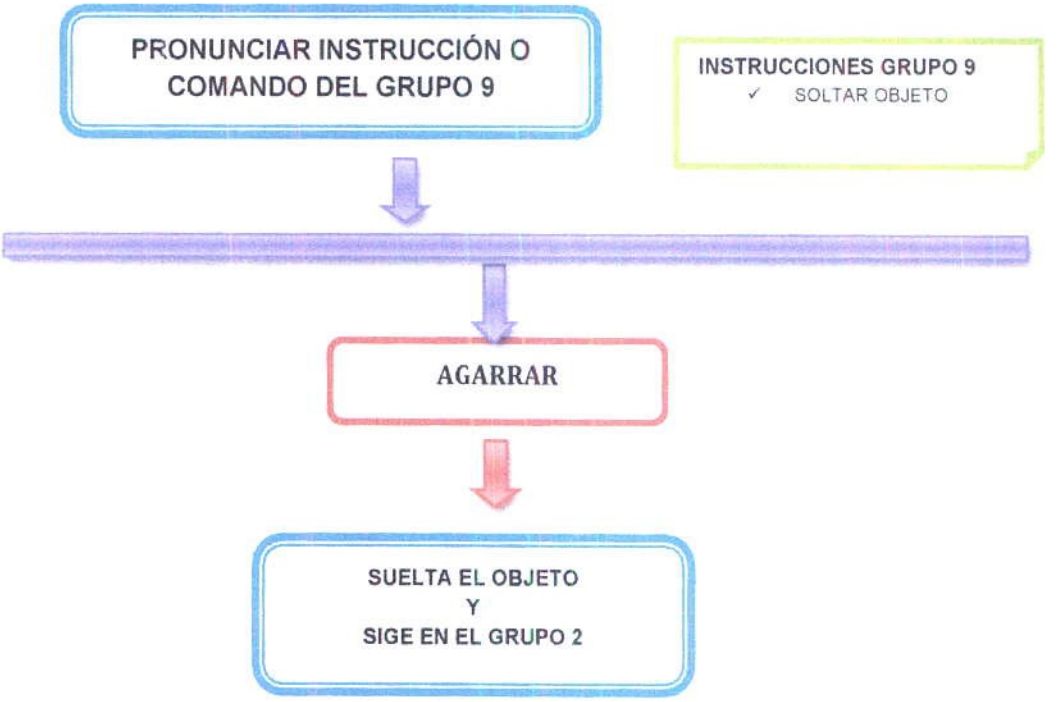
- Flujograma para la selección de comandos del GRUPO 7



- Flujograma para la selección de comandos del GRUPO 8



- Flujograma para la selección de comandos del GRUPO 9



9. CODIGO DE PROGRAMACIÓN DEL EQUIPO TRANSMISIÓN DE MOVIMIENTO

Uno de los grandes retos para el paciente con Lesión Cervical Medular Traumática es alcanzar una ubicación espacial en su entorno doméstico, ya que su discapacidad no le permite girar su cuerpo, y depende de la asistencia de su familia para alcanzar una posición espacial diferente. Por tal motivo, se aprovecha la estructura de la silla reclinable para que gire gracias al diseño de un equipo de transmisión de movimiento a la silla giratoria. Se incorpora el control de este dispcsitivo al módulo de control electrónico del dispositivo exoesqueleto prototipo EMMRA-1, logrando asistencia al paciente en la ubicación espacial controlada por comandos de voz.

El módulo de control del equipo de transmisión de movimiento, constituido por una tarjeta electrónica tipo driver de alta potencia arduino VNH5019, insertado en una tarjeta electrónica Arduino Uno R3.

El código de programación del equipo de transmisión de movimiento a la silla giratoria del paciente discapacitado, también se realizó en Software Arduino, modificando los modelos de programación del driver de alta potencia arduino VNH5019, como se ilustra a continuación.

- Librerías

```
#include "DualVNH5019MotorShield.h"
// conf gure library with pins as remapped for single-channel operation
// this lets the single motor be controlled as if it were 'motor 1"
DualVNH5019MotorShield md(2, 7, 6, A0, 2, 7, 12, A1);
```

- Void Setup

```
void setup()
{
 Serial.begin(115200);
 Serial.println("Dual VNH5019 Motor Shield");
 md.init();
 pinMode(5, INPUT);
 pinMode(11, INPUT);
```

```
  md.setM1Speed(-200);
  delay(2000);
}
```

- Void Loop

```
void loop()
{
  int derecha = digitalRead (5);
  Serial.print (derecha);
  int izquierda = digitalRead (11);
  Serial.print (izquierda);

  if (derecha == 1 && izquierda == 0)
  {
    md.setM1Speed(-140);
    delay(5);
  }

  if (derecha == 0 && izquierda == 1)
  {
    md.setM1Speed(140);
    delay(5);
  }

  else if (derecha == 0 && izquierda == 0)
  {
    md.setM1Speed(0);
    delay(500);
    md.setM1Brake(400);
  }

  else if (derecha == 1 && izquierda == 1)
  {
    md.setM1Speed(0);
    delay(500);
    md.setM1Brake(400);
  }
}
```

446

SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO
NIT : 800.176.089-2
- / -

Industria y Comercio
SUPERINTENDENCIA

RECIBO DE CAJA

No. 15 - 126317

Bogotá D.C., Noviembre 23 de 2015 - 08:12:29

RECIBIDO DE : JUAN DAVID MORENO ARANGO CC 75.076.197

*** Soporte del Pago ***

TIPO PAGO	BANCO	CUENTA	No. PAGO	FECHA PAGO	VR.PAGO
CONSIGNACION	BANCO DE BOGOTA	062754387	682850383	17/11/2015	530.000.00

*** Conceptos Pagados ***

CANT.	RENTISTICO	CONCEPTO	Vr.UNDITARIO	Vr.CONCEPTO
1	50005-01-01 SOLICITUDES	1 TRAMITES DE SOL. DE PATENTE DE INVENCION	530.000.00	530.000.00
				$530.000.00

SON: **QUINIENTOS TREINTA MIL PESOS MONEDA CORRIENTE***

Responsable: _____

Recibo de Caja Aplicado al Expediente No. _____

SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO
No. 15-273320- -00000-0000
Fecha: 2015-11-23 09:36:10 Dep: 2020 DIR NUEVASCF
Tra: 2 PATENTES Eve: 1 REGDEPOSITO
Act: 411 PRESENTACION Folios: 446 447

SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO
No. 15-278320- -00000 -0000
Fecha: 2015-11-23 09:36:10 Dep: 2020 DIR NUEVASCR
Tra: 2 PATENTES Lve: 1 REGDEPOSITO
Act: 411 PRESENTACION Folios: 446 447

447

Industria y Comercio
SUPERINTENDENCIA

LISTA DE CHEQUEO
ADMISIÓN A TRÁMITE – NUEVAS CREACIONES

PATENTE DE INVENCIÓN ☐ **MODELO DE UTILIDAD** ☐ **Art 33 Decisión 486/00**

- ☐ Indicación que se solicita una patente.
- ☐ Datos de identificación del solicitante o de la persona que presenta la solicitud
- ☐ Descripción de la invención
- ☐ Dibujos de ser estos pertinentes
- ☐ Comprobante de pago de las tasas establecidas (De ser el caso formato de descuento)

Completa ☐ Incompleta ☐

PATENTE DE INVENCIÓN PCT ☐ **MODELO DE UTILIDAD PCT** ☐ **Art.33 Decisión 486/00, Circular Única**

- ☐ Indicación que se solicita una PCT
- ☐ Copia de la solicitud en español, tal como fue presentada inicialmente (capítulo descriptivo, reivindicatorio, resumen)
- ☐ Dibujos de ser estos pertinentes
- ☐ Comprobante de pago de las tasas establecidas (de ser el caso formato de descuento)

Completa ☐ Incompleta ☐

DISEÑO INDUSTRIAL ☐ **(Art. 119 Decisión 486/00)**

- ☐ Indicación que se solicita Diseño industrial
- ☐ Datos de identificación del solicitante o de la persona que presenta la solicitud
- ☐ Representación gráfica y fotográfica del Diseño industrial o muestra del material que incorpora el diseño
- ☐ Comprobante de pago de las tasas establecidas

Completa ☐ Incompleta ☐

ESQUEMA DE TRAZADO ☐ **(Art. 92 Decisión 486/00)**

- ☐ Indicación que se solicita un esquema de trazado
- ☐ Datos de identificación del solicitante o de la persona que presenta la solicitud
- ☐ Representación gráfica de un esquema de trazado
- ☐ Comprobante de pago de las tasas establecidas

Completa ☐ Incompleta ☐