CONTROL DE UN BRAZO ROBÓTICO VIRTUAL USANDO UN EXOESQUELETO ROBOT DE MIEMBRO SUPERIOR

FRANCO MARTINEZ JONATAN JAYC

UNIVERSIDAD LIBRE

FACULTAD DE INGENIERIA

PROGRAMA DE INGENIERIA MECANICA

BOGOTÁ

2018

CONTROL DE UN BRAZO ROBÓTICO VIRTUAL USANDO UN EXOESQUELETO ROBOT DE MIEMBRO SUPERIOR

FRANCO MARTINEZ JONATAN JAYC

TRABAJO DE GRADO

Director de Proyecto
EDILBERTO CARLOS VIVAS GONZÁLEZ
Ingeniero Electrónico

UNIVERSIDAD LIBRE

FACULTAD DE INGENIERIA

PROGRAMA DE INGENIERIA MECANICA

BOGOTÁ

2018

Nota de aceptación:

_____

Jurado

_____

Jurado

Bogotá, D.C. 31 enero de 2018

AGRADECIMIENTOS

Agradezco en primer lugar a Dios y a mi familia, especialmente a mi Madre por su ejemplo y su constancia en la enseñanza de valores morales y éticos al igual que su gran e incondicional amor pasando noches y días enteros sin dormir para poder brindarme la gran bendición de poder culminar mis estudios. A mi Padre quien a pesar de las dificultades de la vida siempre me ha protegido con su cariño, enseñándome la importancia del respeto a los demás y la nobleza de un buen ser humano.

También quiero agradecer al Ingeniero Edilberto Carlos Vivas González por su participación durante el desarrollo de esta investigación, sus conocimientos y su constante motivación han sido de gran importancia para mí.

Agradezco especialmente al Ingeniero Julio Cesar Roncancio por su gran apoyo, por la fe que puso en mí desde el primer momento y por ser mi guía en mi vida laboral y personal fuera del hogar, brindándome cada día una nueva enseñanza.

RESUMEN

Se realizó el diseño de un exoesqueleto de miembro superior dentro del software CAD, SOLIDWORKS, identificando los parámetros físicos y cargas mecánicas mediante simulación numérica para posteriormente ser impreso en tres dimensiones (3D), se identificó los tipos de sensores a utilizar en cada articulación del exoesqueleto, para construir el sistema de adquisición y acondicionamiento de señales mediante el microcontrolador Arduino Mega 2560.

Se desarrolló la programación del microcontrolador Arduino para enviar las señales de los sensores al Software V-REP (Versión EDUCACIONAL sin límites. Software Libre). Se exporto el exoesqueleto digitalizado a este software de realidad virtual, se programó para recibir las señales provenientes del Arduino. Se realizó programación de una interfaz de usuario para observar los parámetros de la cinemática directa e inversa de cada mecanismo los cuales fueron el robot serial y el exoesqueleto virtual, Se realizó la simulación de la trayectoria dentro del software de realidad virtual comunicando el exoesqueleto real, con el exoesqueleto virtual y este último con el robot serial.

Se trazó con el exoesqueleto real la trayectoria sobre un eje cartesiano en X, Y, y Z y simultáneamente se registró los puntos coordenados del efector final del exoesqueleto virtual que simuló el mismo trayecto dentro del software V-REP. Las diferencias entre estas dos trayectorias fueron utilizadas para medir el error absoluto, el error porcentual y analizar el surgimiento del error entre la realidad contra la realidad virtual.

LISTA DE DIAGRAMAS

Diagrama 1. Tipo de investigación.....30
Diagrama 2. Diseño metodológico del proyecto.....31
Diagrama 3. Proceso general del proyecto.....32
Diagrama 4. Análisis de registro de trayectoria “eje z”.....62
Diagrama 5. Análisis de registro de trayectoria “eje y”.....63
Diagrama 6. Análisis de registro de trayectoria “eje x”.....64
Diagrama 7. Comparación de la media de errores antes y después 66

LISTA DE TABLAS

Tabla 1. Medidas del brazo del autor 33
Tabla 2 Dimensiones de cada eslabón 60
Tabla 3. Análisis de registro de trayectoria “eje z” 61
Tabla 4. Análisis de registro de trayectoria “eje y”. 62
Tabla 5. Análisis de registro de trayectoria “eje x”. 63

LISTA DE FIGURAS

Figura 1. Procesamiento de los movimientos del exoesqueleto a imagen virtual. .18
Figura 2. Exoesqueleto, mano robótica diseñada por FESTO 18
Figura 3. Robot YUMI ®. ABB's award winning collaborative robot YuMi 19
Figura 4. Puntos coordinados de cada eslabón robot IRB 140. 23
Figura 5. Descripción de los componentes de la placa Arduino "uno" 24
Figura 6. Robot ABB IRB 140 29
Figura 7. Rodamientos en exoesqueleto.....34
Figura 8. Simulación de un análisis estático. 35
Figura 9. Modelo impreso en 3D de exoesqueleto.....37
Figura 10. Ejemplo de conexión con el microcontrolador Arduino mega2560 38
Figura 11. Sensor tipo Flex dentro protector de polímero flexible en exoesqueleto. 38
Figura 12. Posicionador con punta de aguja para los potenciómetros. 39
Figura 13. Posicionador con punta de aguja, vista superior.....39
Figura 14. Gráfica de caracterización de Potenciómetro 41
Figura 15. Gráficas de caracterización de Sensor Flex 42
Figura 16. Circuito electrónico del exoesqueleto. 43
Figura 17. Scene Object Properties 51
Figura 18. Calculation Modules e IK Group 52
Figura 19. IK Group secuencia de procesado. 53
Figura 20. Interfaz de usuario del exoesqueleto y del robot virtual 54
Figura 21. Toggle OpenGL-based custom UI edit mode. 56
Figura 22. falta 66

LISTA DE ECUACIONES

Ecuación 1. Matriz de Transformación.....21
Ecuación 2. Matriz de transformación asociada a cada eje 22
Ecuación 3. Matriz de transformación resultante 22
Ecuación 4. Vector de traslación.....22
Ecuación 5. Sumatoria de fuerzas igual a cero.....24
Ecuación 6. Sumatoria de momentum igual a cero.....24
Ecuación 7. Factor de Seguridad.....26
Ecuación 8. Característica línea de palabra binaria a ángulos 40
Ecuación 9. Distancia entre puntos en el espacio.....59

PARTICIPACIÓN EN EVENTOS

Evento:	Emprendetronika - Bogotá Robótica
Fecha:	Del 2 al 5 de octubre de 2014
Lugar:	Plaza de los Artesanos
Tema expuesto:	Brazo Robótico.
En calidad:	De expositor
Certificado por:	Alta Consejería Distrital de TIC Secretaria General de la Alcaldía Mayor de Bogotá, D.C.

CONTENIDO

INTRODUCCIÓN 11

1.DEFINICIÓN DEL PROBLEMA 12

1.1 Descripción del problema 12
1.2 Antecedentes 13
1.3 Formulación 13

2. JUSTIFICACIÓN 14

3. OBJETIVOS 15
3.1 Objetivo General 15
3.2 Objetivos Específicos 15

4. DELIMITACIÓN 16

5. MARCO REFERENCIAL 17

5.1 ESTADO DEL ARTE 17
5.1.1 Sistemas Teleoperados 17
5.1.2 Robots seriales 17
5.1.3 Proyectos relacionados 17

6.2 MARCO TEÓRICO 20

6.2.1 Robótica 20
6.2.2 Cinemática 21
6.2.2.1 Cinemática directa 21
6.2.2.2 Transformaciones homogéneas 21
6.2.2.3 Convención Denavit – Hartenberg: 22
6.2.2.4 Cinemática inversa: 23
6.2.3 Estática: 24
6.2.4 Descripción de la placa Arduino 24
6.2.5 Factor de Seguridad 26
6.3 MARCO CONCEPTUAL 27

6.3.1 Exoesqueleto....27
6.3.2 Brazo robótico....27
6.3.3 Arduino....27
6.3.4 Sensores análogos....28
6.3.5 Robot ABB IRB 140....28
6.3.6 Software V-REP (Virtual Robot Experimentation Platform)....29
6.4 MARCO METODOLÓGICO....30
6.4.1 Descripción del diseño Metodológico....30
6.4.2 Desarrollo Objetivo 1....33
6.4.2.1 Construcción de exoesqueleto....33
6.4.2.2 Tipos de sensores y programación de sensores en Arduino y V-REP....37
6.4.3 Desarrollo de Objetivo 2....50
6.4.3.1 Cinemática directa e inversa de exoesqueleto y robot serial.....50
6.4.3.2 Registro de la trayectoria del exoesqueleto al brazo robot virtual.....53
6.4.4 Desarrollo de Objetivo 3....58
6.4.4.1 Cálculo de error de la trayectoria....58
7. RESULTADOS Y DISCUSIÓN....60
7.1 Análisis Estático....60
7.2 Análisis de registro de error de trayectoria....61
8. CONCLUSIONES....67
9. RECOMENDACIONES....69
REFERENCIAS BIBLIOGRAFICAS....70
Referencia Normas Complementarias....73
ANEXOS....74
Script de Exoesqueleto....74
Script de Brazo Robot IRB 140....86
Script Obtención de Señales – V-REP.....98
Script Obtención de Señales – ARDUINO....99
Planos....103

INTRODUCCIÓN

En la actualidad, el área de la ingeniería está avanzando de forma acelerada en todos los campos de investigación. Es por esto, que dentro de las grandes instituciones como las universidades se están dando investigaciones en torno a las nuevas áreas de la ingeniería, las cuales buscan mejorar la calidad de vida del ser humano, ampliar las posibilidades y tener un impacto más amplio en todas las comunidades.

En el área de la bioingeniería se están desarrollando diferentes prototipos para mejorar la calidad de vida, como es el caso de los exoesqueletos robóticos, los cuales están pensados en mejorar las capacidades humanas como la fuerza o dotar al sujeto de habilidades que por su propia cuenta no puede ejecutar, como es el caso de caminar para las personas con discapacidad física parcial o total, ya sea biológica o generada por algún factor externo en particular.

El presente proyecto está pensado en incursionar en el área de la seguridad industrial a partir de un prototipo de simulación virtual que permita el movimiento de un robot con mando a distancia usando un exoesqueleto robótico de miembro superior.

1.DEFINICIÓN DEL PROBLEMA

1.1 Descripción del problema

La humanidad desde sus inicios se ha visto expuesta en actividades que pueden perjudicar parcial o completamente su integridad y salud física. Existen trabajos que pueden dañar directamente al ser humano; variables como altas temperaturas, químicos tóxicos, peligros ambientales, riesgos biológicos, esfuerzos físicos, entre otros.

En la actualidad empresas y universidades han trabajado en el diseño de exoesqueletos para los procesos de teleoperación; (teleoperación es aquella actividad que permite ser ejecutada en tiempo real y a distancia por una persona, con la ayuda de un manipulador o robot). Con el objetivo de lograr reducir la intervención de las personas en trabajos directos que los lesionen y puedan llegar a causar daños de consideración a su salud.

Los sistemas de teleoperación mecánicos tipo exoesqueleto logran la adquisición de los movimientos que realizan las personas de forma natural; siendo uno de sus múltiples empleos el captar los movimientos del brazo humano y con estos dar instrucciones a un robot, logrando realizar un trabajo sin la necesidad de la intervención o manipulación directa del hombre.

Esta explicación permite ver la importancia del desarrollo de la robótica y del exoesqueleto, en pro de la humanidad y en la prevención de riesgos laborales. El exoesqueleto robótico, es una herramienta ingenieril que abarca varias áreas: entre ellas se encuentran la ingeniería aeroespacial, la bioingeniería, y demás áreas afines. Todas ellas buscan disminuir mediante su empleo, el riesgo al ser humano en ambientes hostiles y logran evitar su presencia directa con el trabajo a realizar.

Un ejemplo de estas investigaciones las realiza la NASA (National Aeronautics and Space Administration), donde un astronauta controla un robot humanoide mediante el uso de un exoesqueleto robótico para la obtención de muestras en otro planeta sin la necesidad de estar presente. [1]

Este proyecto se enfocó en el desarrollo del prototipo de un exoesqueleto de miembro superior, donde se enviaron las coordenadas espaciales de los movimientos efectuados en un espacio real por el exoesqueleto a un robot serial ubicado dentro de una realidad virtual; simulando un trabajo de teleoperación.

Con el resultado de este proyecto se aportó al avance de nuevas técnicas que permiten al ser humano salvaguardar su salud, aplicando los conocimientos de la ingeniería mecánica.

1.2 Antecedentes

La implementación de exoesqueletos de miembro superior se ha desarrollado por diferentes instituciones. En Europa la ESA (European Space Agency) ha desarrollado un complejo exoesqueleto (en conjunto con la University of Brussels, Space Application Services (SAS) y MicroMega), este le permite al usuario un movimiento de su brazo en 7 grados de libertad, tomar las señales del mismo y controlar un robot virtual, posee un feedback de fuerza al tener actuadores en el mismo exoesqueleto, que le permiten identificar si el robot virtual toca algún objeto externo a él." [2]

Al igual que la ESA, la NASA ha desarrollado un exoesqueleto con fines parecidos, los cuales permite controlar un robot a distancia reduciendo el índice de riesgo del astronauta en el espacio, SAM (Sensor Exoskeleton Arm-Máster for Robot Control) [11] el cual permite un control de cinco grados de libertad, y posee un feedback de fuerza.

Se sabe que los exoesqueletos de miembro superior además de situarse en todo el brazo pueden ser usados para leer las diferentes posiciones de los dedos de las manos. En este proyecto entra FESTO, una empresa alemana especializada en equipos robóticos, con su diseño de un exoesqueleto mano logrando la imitación del movimiento de la mano y transfiriendo estos movimientos a una mano robótica de gran precisión ambas partes son neumáticas, muy característico de FESTO (EXOHAND). [3]

Como se observa todas estas aplicaciones van enfocadas en buscar un mejoramiento en la calidad del ser humano, disminuyendo el riesgo a enfrentarse a ambientes que podrían ocasionar un daño a su integridad física.

1.3 Formulación

Es importante dar los primeros pasos a nivel de investigación dentro de la universidad sobre la creación de exoesqueletos robóticos, donde permita ver las utilidades que tienen para la industria, la seguridad industrial y el mejoramiento de la calidad de vida. En vista de que son pocas las investigaciones en este ámbito, es importante dar el primer paso en la investigación del campo de la robótica y la bioingeniería, así como el uso de herramientas de computación para la simulación de este tipo de estudios.

2. JUSTIFICACIÓN

El incluir la robótica en la mayor parte de procesos productivos ha dado como resultado grandes avances en diferentes tipos de industrias; áreas como la salud se han visto beneficiadas con estos, gran parte de ello se debe a que el hombre siempre ha estado en la búsqueda de mejorar la calidad de la vida humana. La creación de un exoesqueleto robótico puede aportar al mejoramiento de las condiciones de seguridad en la industria, ya que las personas se exponen a peligros que pueden comprometer su vida y su integridad física.

En un estudio realizado en el año 2014 llamado “Comportamiento de la accidentalidad en una empresa metalmecánica en Cartagena, Entre los agentes causales de los accidentes esta investigación se destacan las herramientas manuales con un 66.6% (24 de 36)” [4]. La investigación de los diferentes riesgos humanos existentes tanto en la industria como en otras áreas puede disminuir la probabilidad de accidentes laborales. Es por esto por lo que se ve necesario plantear alternativas que permitan disminuir los riesgos que puedan comprometer la integridad física de las personas. Para este fin se planteó el uso de un subgrupo de robots llamados *wearable robots* (robots de vestir) [5] los cuales contienen a los EXOESQUELETOS ROBÓTICOS.

Los miembros superiores son los más afectados en el área laboral, por tal motivo el diseño del exoesqueleto para miembro superior permitirá realizar trabajos complejos que el ser humano ha ejecutado durante años, reduciendo drásticamente la ejecución de trabajos perjudiciales para la salud en ambientes poco amigables.

3. OBJETIVOS

3.1 Objetivo General

Creación de un exoesqueleto robótico de miembro superior para la teleoperación de un brazo robot virtual.

3.2 Objetivos Específicos

- Diseñar el sistema de adquisición y acondicionamiento de señales en el exoesqueleto robótico impreso en 3D, mediante una metodología experimental.

- Registrar la trayectoria obtenida del exoesqueleto por medio de una herramienta de software de realidad virtual representada en el movimiento de un brazo robótico.

- Evaluar el error de seguimiento de las trayectorias obtenidas con el exoesqueleto.

4. DELIMITACIÓN

Este proyecto se enfocará únicamente a la simulación del movimiento de un robot virtual, mediante la adquisición y acondicionamiento de señales obtenidas por los movimientos sensados en el exoesqueleto, el cual será construido para ser utilizado por el brazo de una persona, dando parámetros óptimos del funcionamiento de este.

El exoesqueleto modelado y construido en su mayor parte por impresión 3D permitirá reducir costos y realizar una simulación de cómo un brazo exoesqueleto robótico puede controlar a distancia un robot industrial, en este caso de modo virtual únicamente.

5. MARCO REFERENCIAL

5.1 ESTADO DEL ARTE

5.1.1 Sistemas Teleoperados

Los sistemas teleoperados nacen como un requerimiento en la industria nuclear, donde se necesitaba manipular elementos en ambientes peligrosos para el ser humano. [6]

La teleoperación es el conjunto de técnicas que permiten a un operador manejar un dispositivo remoto mediante el uso de un dispositivo local. [7]

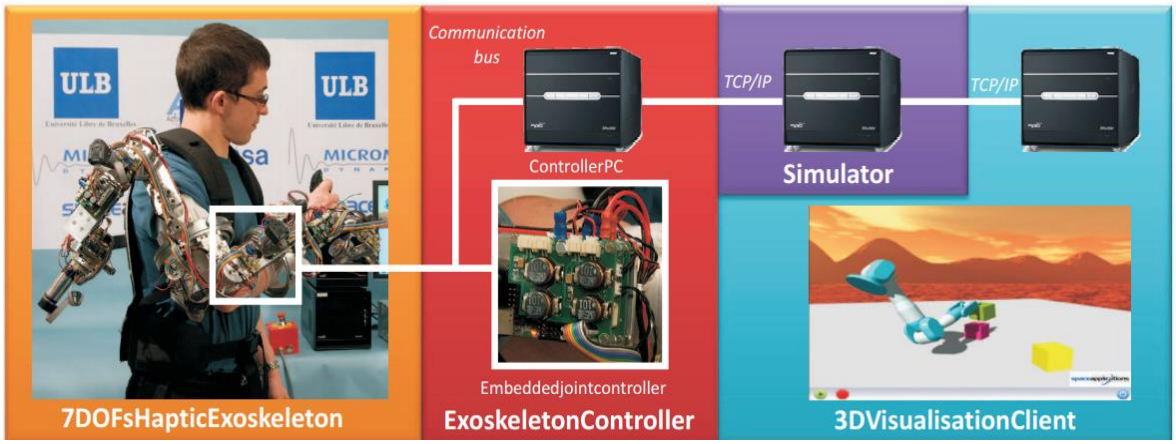
5.1.2 Robots seriales

Los robots pueden ser clasificados de acuerdo con varios criterios: por la tecnología de sus controladores y actuadores (eléctricos, mecánicos, hidráulicos o neumáticos), por la estructura cinemática del mecanismo, por la naturaleza de su movimiento o por el número de grados de libertad que posee, entre otros: de acuerdo con la estructura cinemática del mecanismo los robots se dividen en tres grandes categorías: robots seriales, paralelos e híbridos. Se dice que un robot es serial si su cadena cinemática es abierta; es decir, sus elementos se encuentran conectados uno a uno en serie. [8]

5.1.3 Proyectos relacionados

1. Exostation: haptic exoskeleton based control station (*Exostación: estación de control basada en exoesqueleto háptico*) es un proyecto, el cual se puede tomar como base para el diseño del proyecto de investigación, ya que ellos han diseñado un exoesqueleto que controla un robot esclavo por software. Con una ventaja en su gran precisión y feedback de fuerza. Aunque con un enorme tamaño en su diseño, por su variedad de funciones y sistemas embebidos. [9]

Figura 1. Procesamiento de los movimientos del exoesqueleto a imagen virtual.



Fuente 1. [http://esa-telerobotics.net/gallery/Research/Haptic-robots-design-development/SAM-Exoskeleton/16]

2.- "Exoskeleton hand by Festo (*Mano exoesqueleto por Festo*) exoesqueleto de mano que recolecta todos los movimientos del miembro superior, usa la realidad virtual como medio de visualización y de este modo mueve un robot serial construido con una mano robótica en su efector final, replicando todos los movimientos de la mano y brazo del operador. Su mecanismo es mediante un sistema hidráulico; permite tener una retroalimentación de fuerza. Este funciona a partir de un sistema hidráulico-mecánico. [3]

Figura 2. Exoesqueleto, mano robótica diseñada por FESTO

Fuente 2.FESTO. [https://www.festo.com/group/en/cms/10233.htm]

3.- ABB's award winning collaborative robot YUMI®: (*El galardonado robot colaborativo de ABB YUMI*®) La empresa ABB fabrica brazos robóticos industriales. Desarrollaron una herramienta de enseñanza donde el operario manualmente le indica al robot la posición donde ir, este mueve el robot a todos los puntos manualmente y después de procesar los puntos de trayectoria el robot YUMI® reproduce tales movimientos. [10] (ABB's award winning collaborative robot YuMi, Photo by ABB) [11]

Figura 3. Robot YUMI ®. ABB's award winning collaborative robot YuMi

Fuente: ABB [ABB's award winning collaborative robot YuMi Photo by ABB]

6.2 MARCO TEÓRICO

6.2.1 Robótica

La robótica es la ciencia que estudia el diseño y la implementación de robots, conjugando múltiples disciplinas, como la mecánica, la electrónica, la informática, la inteligencia artificial y la ingeniería de control, entre otras. [12] El estudio posee tanto la investigación teórica como la aplicada, partiendo su estudio en el diseño del robot con su funcionalidad, su mecánica, la planeación y control de trayectoria, su programación e inteligencia artificial [13].

Según la RIA (Robotics Industry Association) un robot es un manipulador programable y multifuncional, diseñado para mover materiales, piezas, herramientas o dispositivos especializados mediante movimientos programados y variables con objeto de realizar diversas tareas. Esta definición, aunque se ajusta perfectamente a los robots industriales no lo hace tanto a los denominados robots móviles autónomos y no manipuladores. [14]

En la práctica, las actuales y potenciales aplicaciones no industriales de los robots son tan variadas y diferentes, que es difícil encontrar una definición suficientemente amplia y concreta a la vez de un robot de servicio, más aún por cuanto la gran diversificación de estas aplicaciones y el bajo número de sistemas iguales (en muchos casos son sistemas o aplicaciones únicas), dificulta su conocimiento y catalogación.

Tratando no obstante de establecer una primera división en estas aplicaciones no industriales de los robots, la IFR (International Federation of Robotics) ha propuesto clasificarlas en:

- Aplicaciones de Servicio a humanos (personal, protección, limpieza, etc.)
- Aplicaciones Servicio a equipos (mantenimiento, reparación, limpieza, etc.)
- Otras funciones autónomas (vigilancia, transporte, adquisición de datos, inspección, etc.) [15]

Los robots pueden ser clasificados de acuerdo con varios criterios: por la tecnología de sus controladores y actuadores (eléctricos, mecánicos, hidráulicos o neumáticos), por la estructura cinemática del mecanismo, por la naturaleza de su movimiento o por el número de grados de libertad que posee, entre otros. Los robots se dividen en tres categorías: robots seriales, paralelos e híbridos.

Un robot es serial si su cadena cinemática es abierta; es decir, sus elementos se encuentran conectados uno a uno en serie. Un robot paralelo, por el contrario, posee una cadena cinemática cerrada en la cual se pueden identificar varios lazos. Si se conectan ambos tipos de cadena cinemática en un robot se dice que éste se vuelve híbrido. [16]

Dependiendo a su naturaleza de movimiento los robots pueden clasificarse en planares y espaciales, con respecto a si su movimiento actúa planos paralelos, en 2 dimensiones o en un espacio de 3 dimensiones. Tanto los robots paralelos, seriales e híbridos se les permite ser planares o espaciales. [17]

6.2.2 Cinemática

Esta conlleva un estudio analítico de la geometría del movimiento de un mecanismo robótico en el espacio con respecto a un sistema de coordenadas fijo, como una función del tiempo sin suponer las fuerzas y momentos que causaron el movimiento. [18]

6.2.2.1 Cinemática directa

La cinemática directa se resume a encontrar una matriz de transformación que corresponde al sistema de coordenadas unido al mecanismo del sistema. Se obtendrá la posición determinada de un punto cinemático final del manipulador, proporcionando los ángulos de las articulaciones y los parámetros geométricos. [19]

6.2.2.2 Transformaciones homogéneas

La transformación homogénea es una herramienta matemática que describe la traslación y rotación de un sistema geométrico en una matriz. Es decir, la posición y orientación relativa entre los distintos sistemas asociados a dos eslabones consecutivos del robot (se denomina i−1Ai).la matriz 0Ak, resultante del producto de las matrices i−1Ai con i desde 1 hasta k, es la que representa de forma total o parcial la cadena cinemática que forma el robot con respecto al sistema de referencia inercial asociado a la base. Cuando se consideran todos los grados de libertad, a la matriz 0An se le denomina T La matriz de transformación relaciona la posición y orientación del punto extremo final del robot respecto del sistema fijo situado en la base de este:

Ecuación 1. Matriz de Transformación

$$0_{A_n} = T$$

Siendo $S\theta = Sen\theta \; y \;\; C\theta = Cos\theta$

$$T = T\,(z,\theta)t\,(y,\phi)T(x,\alpha) = \begin{bmatrix} C\theta & -s\theta & 0 & 0 \\ s\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} C\phi & 0 & S\phi & 0 \\ 0 & 1 & 0 & 0 \\ -S\phi & 0 & C\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha & -S\alpha & 0 \\ 0 & S\alpha & C\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 2. Matriz de transformación asociada a cada eje

$$\mathrm{T}= \begin{bmatrix} C\phi C\theta & -S\theta C\alpha + C\theta S\phi S\alpha & 0S\theta S\alpha + C\theta S\phi C\alpha & 0 \\ S\theta C\phi & C\theta C\alpha + S\theta S\phi S\alpha & -C\theta S\alpha + S\theta S\phi C\alpha & 0 \\ -S\phi & C\phi S\alpha & C\alpha C\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ecuación 3. Matriz de transformación resultante

En donde se hace referencia al seno=S y al coseno del ángulo y para la representación de la rotación en los tres ejes se pueden multiplicar las matrices de rotaciones de formas individuales. El vector de traslación T, el cual indica la traslación existente con respecto a los ejes de un sistema de coordenadas fijo. [20]

$$T = \begin{bmatrix} Tx \\ Ty \\ Tx \end{bmatrix}$$

Ecuación 4. Vector de traslación

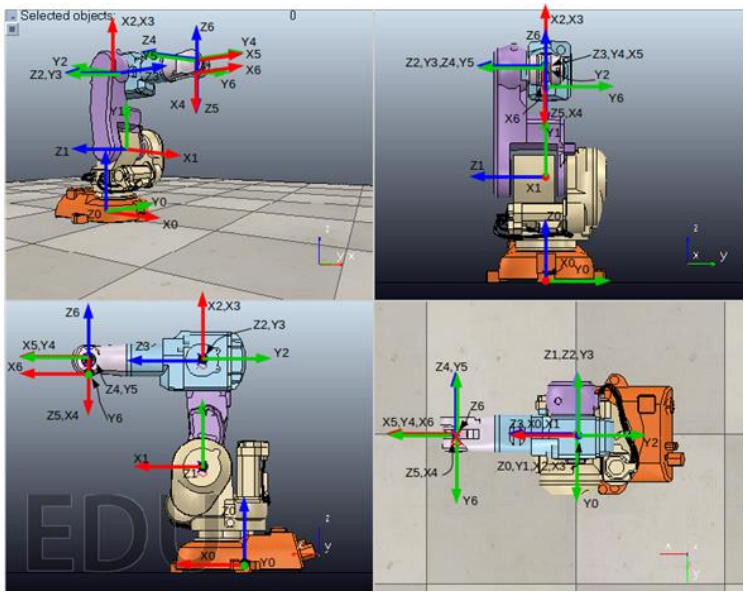
Con los dos componentes anteriores se obtiene la posición final del mecanismo.

6.2.2.3 Convención Denavit – Hartenberg:

La convención o metodología de Denavit-Hartenberg (DH, Jacques Denavit y Richard) permite establecer la ubicación de los sistemas de referencia de los eslabones en los sistemas robóticos articulados. Que permite tener una

representación de la orientación y traslación de un brazo robótico, esta representación, se aplica a robots seriales o de cadena cinemática abierta y consta de una serie de reglas para colocar los sistemas de referencia de cada eslabón. [21]

Figura 4. Puntos coordinados de cada eslabón robot IRB 140.



Fuente: V-REP.

6.2.2.4 Cinemática inversa:

La cinemática inversa se enfoca en hallar de forma analítica los parámetros (ángulos) que tomarán las articulaciones para que el punto final del mecanizado alcance la posición deseada. Por lo general se pueden encontrar configuraciones que no son posibles, es decir, que no son realizables por el robot, configuraciones particulares que requieren una dirección de movimiento no-holónoma (son aquellas restricciones que dependen de la velocidad. Además, se exige que sea integrable y no se pueden obtener derivando una restricción holónoma o incluso soluciones múltiples, típicamente soluciones orientación del codo final hacia arriba o codo hacia abajo en los brazos robots articulados en serie (Peñín, Barrientos et al. 2007). [22]

6.2.3 Estática:

La ESTÁTICA trata con el equilibrio de los cuerpos, esto es, aquellos que están en reposo o se mueven con velocidad constante, sometidos a la acción externa de cargas puntuales y distribuidas, así como de momentos. [23] En el caso de la robótica, al ser un sistema mecánico, está sometido por diferentes fuerzas y momentums interconectados, es por esto por lo que la fuerza que se ejerce en el primer eslabón tendrá un efecto en cadena en el siguiente eslabón; es de suma importancia conocer las magnitudes necesarias para mantener al mismo en un equilibrio estático para su diseño y posterior construcción.

Esto se puede calcular analíticamente a partir de las siguientes fórmulas:

$$\Sigma F = 0$$

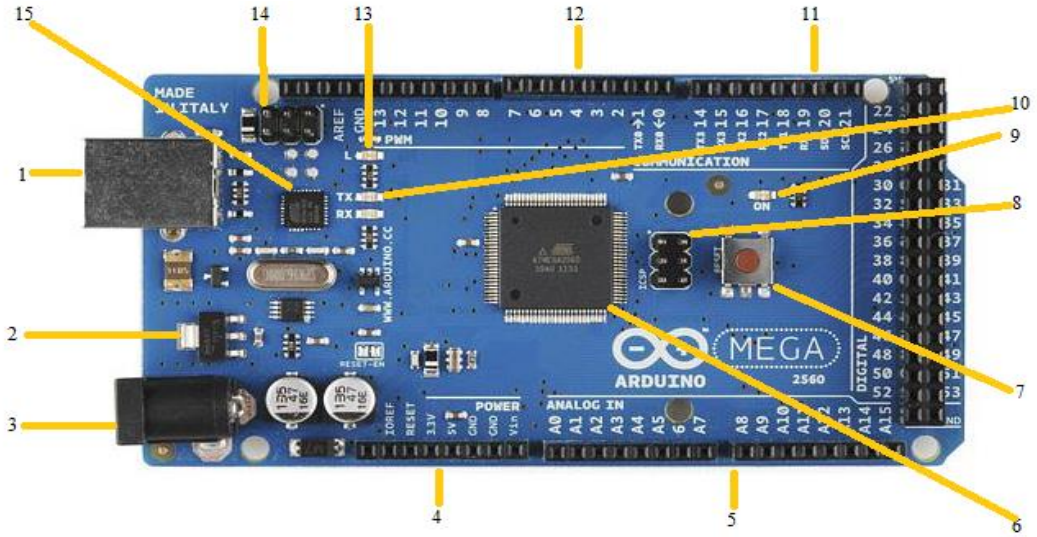
Ecuación 5. Sumatoria de fuerzas igual a cero

$$\Sigma M = 0$$

Ecuación 6. Sumatoria de momentum igual a cero

6.2.4 Descripción de la placa Arduino

Figura 5. Descripción de los componentes de la placa Arduino "uno"



Fuente: http//: www.Arduino.cc

1 Conector USB: Proporciona la comunicación para la programación y la toma de datos, también provee una fuente de 5vdc para alimentar al Arduino, pero de baja corriente por lo que no sirve para alimentar motores de gran potencia. Pero si sensores de alta impedancia.

2 Regulador de voltaje de 5v: Se encarga de convertir el voltaje ingresado por el plug 3, en un voltaje de 5v regulado necesario para el funcionamiento de la placa y para alimentar circuitos externos.

3 Plug de conexión: para fuente de alimentación externa: Es el voltaje que se suministra que debe ser directo y estar entre 6v y 18v ó hasta 20v, generalmente se debe de tener cuidado de que el terminal del centro del plug quede conectado a positivo ya que algunos adaptadores traen la opción de intercambiar la polaridad de los cables.

4 Puerto de conexiones: Es constituido por 6 pines de conexión con las funciones de reset que permite resetear el microcontrolador al enviarle un cero lógico. Pin 3.3v provee de una fuente de 3.3vdc para conectar dispositivos externos como en la protoboard, por ejemplo. Pin 5v es una fuente de 5vdc para conectar dispositivos externos. Dos pines GND que permite la salida de cero voltios para dispositivos externos. Pin VIN, este pin está conectado con el positivo del plug 3 por lo que se usa para conectar la alimentación de la placa con una fuente externa de entre 6 y 12 VDC en lugar del plug 3 o la alimentación por el puerto USB.

5 Puertos de entradas análogas: Lugar donde se conectan las salidas de los sensores análogos. Estos pines solo funcionan como entradas recibiendo voltajes entre cero y cinco voltios directos.

6 Microcontrolador atmega2560: en montaje superficial (SMD), ventajas son la reducción del peso y ganar un poco de espacio.

7 Botón reset: Permite resetear el microcontrolador haciendo que reinicie el programa.

8 Pines de programación ICSP: Son usados para programar microcontroladores en protoboard o sobre circuitos impresos sin tener que retirarlos de su sitio.

9 Led ON: Enciende cuando el Arduino está encendido.

10 Leds de recepción y transmisión: Se encienden cuando la tarjeta se comunica con el pc. El TX indica transmisión de datos y el RX recepción.

11 Puertos de conexiones de pines de entradas o salidas digitales: La configuración como entrada o salida debe ser incluida en el programa. Cuando se usa el terminal serial es conveniente no utilizar los pines cero (rx) y uno (tx). Los

pines 3, 5 y 6 están precedidos por el símbolo ~, lo que indica que permiten su uso como salidas controladas por ancho de pulso PWM.

12 Puerto de conexiones 5 entradas o salidas adicionales: Las salidas 9, 10 y 11 permiten control por ancho de pulso; la salida 13 es un poco diferente pues tiene conectada una resistencia en serie, lo que permite conectar un led directamente entre ella y tierra. Finalmente hay una salida a tierra GND y un pin aref que permite ser empleado como referencia para las entradas análogas.

13 Led pin 13: Indica el estado en que se encuentra.

14 Pines de programación ICSP: Son usados para programar microcontroladores en protoboard o sobre circuitos impresos sin tener que retirarlos de su sitio.

15 Chip de comunicación: Permite la conversión de serial a USB.

6.2.5 Factor de Seguridad

El factor de seguridad se puede calcular como la relación entre la tensión máxima permitida y la tensión equivalente (Von Mises = σVonMises) cuando se usa el límite de elasticidad. Debe ser superior a uno (1) para que el diseño sea aceptable. (Un valor inferior a 1 indica que existe una deformación permanente.) Cuando se usa la resistencia máxima, la tensión principal máxima se emplea para determinar los coeficientes de seguridad. [24]

Dentro del software de simulación, "los resultados del coeficiente de seguridad señalan inmediatamente áreas de elasticidad potencial. Los resultados de la tensión equivalente se muestran en rojo en las áreas de máxima tensión, con independencia de que el valor sea alto o bajo. Un coeficiente de seguridad de 1 significa que el material es esencialmente elástico. La mayoría de los diseñadores procuran obtener un coeficiente de seguridad entre 2 y 4 según el escenario de carga máxima prevista. Si algunas áreas del diseño van a elasticidad no significa siempre que haya un error en la pieza. a menos que la carga máxima prevista se repita con frecuencia". [24]

El límite elástico (σLimit) es una propiedad dependiente de la temperatura. Este valor especificado del límite elástico debe considerar la temperatura del componente. El factor de seguridad en una ubicación se calcula a partir de: [25]

$$\text{Factor de seguridad } (\eta) = \sigma_{Limit} / \sigma_{VonMises}$$

Ecuación 7. Factor de Seguridad

6.3 MARCO CONCEPTUAL

6.3.1 Exoesqueleto

Los seres humanos han construido desde la antigüedad armaduras artificiales para su protección y seguridad, especialmente en combate. Actualmente este tipo de armaduras, conocidas como exoesqueletos se usan para distintos campos; existen exoesqueletos que permiten la rehabilitación de pacientes con pérdida de movimiento, otros exoesqueletos son utilizados para multiplicar la fuerza de miembros tanto superiores como inferiores al igual que otros usados como manipuladores de robots a distancia. Las ortesis son una forma médica limitada de exoesqueleto. Una ortesis es un mecanismo que, acoplado a una pierna, o al torso, permite mejorar o corregir el comportamiento de esa pierna o de la espina dorsal.

Una prótesis de pierna es un dispositivo que sustituye la parte faltante de una pierna. Si la prótesis forma su propia cubierta, se considera exoesqueletal. Si la estructura y el mecanismo son usados de manera interna y está cubierto de un material blando y no estructural, se considera una prótesis endoesquelética.

El exoesqueleto mecánico, también llamado exoesqueleto de potencia o exoesqueleto robótico (posiblemente conocido como exomarco o exotraje), es una máquina móvil cuya estructura contiene un armazón con un sistema de potencia de motores/ hidráulicos, o simplemente sensores de movimiento. Proporcionando al menos parte de la energía para el movimiento de los miembros o recolectando información de los grados de rotación a su portador. [26]

6.3.2 Brazo robótico

La palabra robot deriva del checo robota, su principal significado es trabajo forzado.

Ahora su definición viene unida a la expresión manipulador multifuncional que puede ser reprogramable, el cual puede ser diseñado para mover materiales o piezas y hasta otros dispositivos especializados. Un robot puede hacer diferentes tareas a través de distintos movimientos. Un brazo robot, es del tipo cinemática en serie, es decir, sus articulaciones solo son unidas por un punto secuencial que no se repite. Se busca asimilar al brazo humano. Su trabajo se realiza dentro de un espacio específico de ejecución. Es usado en diferentes procesos. como sujetar, soldar, pintar entre otras varias opciones. [27]

6.3.3 Arduino

Es una plataforma de hardware de código abierto, basada en una sencilla placa de circuito impreso que contiene un microcontrolador de la marca “Atmel” que cuenta con entradas y salidas, analógicas y digitales, en un entorno de desarrollo que está

basado en el lenguaje de programación processing. El dispositivo conecta el mundo físico con el mundo virtual, o el mundo analógico con el digital controlando, sensores, alarmas, sistemas de luces, motores, sistemas comunicaciones y actuadores físicos. El software Arduino está publicado bajo una licencia libre y preparada para ser ampliado por programadores y desarrolladores experimentados.

El lenguaje puede ampliarse a través de librerías de C++ y modificarlo a través del lenguaje de programación AVR C en el que está diseñado. [28]

6.3.4 Sensores análogos

- **Potenciómetros:** Son ideales para aplicaciones de instrumentación debido a la variación lineal de la resistencia eléctrica en función de la posición angular.

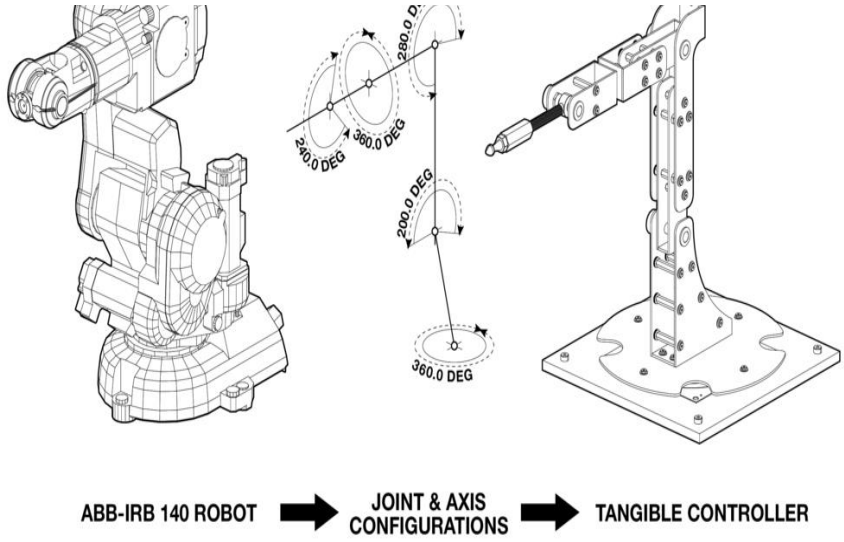
- **Deflexión:** Este sensor de deflexión puede detectar la flexión cuando su posición cambia de su estado de reposo cambiando la resistencia medida en sus salidas, con este efecto se puede adaptar este sensor a sistemas como guantes virtuales o rotación de muñecas. [29]

6.3.5 Robot ABB IRB 140

El robot industrial multiusos de 6 ejes IRB 140 soporta una carga de 6kg con un alcance de 810mm. Se puede instalar en el suelo, de manera invertida o en la pared desde cualquier ángulo. Disponible en las siguientes versiones: estándar, Foundry Plus, Clean Room y Hermético, en el que el robot tiene una completa protección IP67 que lo hace apropiado para una extensa gama de aplicaciones. [30]

Su diseño robusto, con cables completamente integrados, además de su completa flexibilidad y su función de detección de colisiones con completa retracción de la trayectoria, aseguran la fiabilidad y seguridad de este robot.

Figura 6. Robot ABB IRB 140



Fuente: http://www.liftarchitects.com/robotic-motion-controller/

6.3.6 Software V-REP (Virtual Robot Experimentation Platform)

El simulador de robot V-REP de la marca Coppelia Robotics, con entorno de desarrollo integrado, se basa en una arquitectura de control distribuido: en donde cada objeto / modelo puede controlarse individualmente mediante un script incorporado, un complemento, un nodo ROS, un cliente API remoto o una solución personalizada. Esto hace que V-REP sea muy versátil e ideal para aplicaciones multi-robot.

Entidades educativas (aficionados, estudiantes, maestros, profesores, escuelas y universidades) pueden utilizar V-REP PRO EDU gratis. El código fuente de todos los elementos está disponible.

6.4 MARCO METODOLÓGICO

6.4.1 Descripción del diseño Metodológico

El diseño metodológico es la estrategia que se utiliza para dar solución a la problemática planteada, por lo cual en esta sección se plasma cómo se desarrolló la investigación. Durante esta investigación se utilizaron dos métodos en sus diferentes fases; la investigación cuantitativa – experimental [31] y la investigación documental [32]. A partir de las mismas se obtuvieron diferentes resultados y se pudo dar una adecuada continuidad al proceso de investigación. En los siguientes diagramas se hace una breve explicación sobre cada método de investigación y como se les dio respuesta a los objetivos planteados:

TIPO DE INVESTIGACIÓN

Investigación cuantitativa –experimental:

Esta investigación procedió en un primer momento desde la creación de un exoesqueleto robótico que sensa la posición del miembro superior y reproduce la posición en un robot virtual. Es un proceso sistemático y una aproximación científica a la investigación en la cual el investigador manipula una o más variables; controla y mide cualquier cambio en otras variables.

Documental:

Como segundo momento, se tomó la investigación documental (Garza, 1988) es la técnica que se enfoca en la utilización de registros impresos, manuscritos, sonoros y gráficos, los cuales se emplean básicamente como fuentes de información. Este tipo de investigación sirve para aprender, analizar y aplicar como programar el software de realidad virtual y microcontrolador.

Diagrama 1. Tipo de investigación
Fuente: Autor

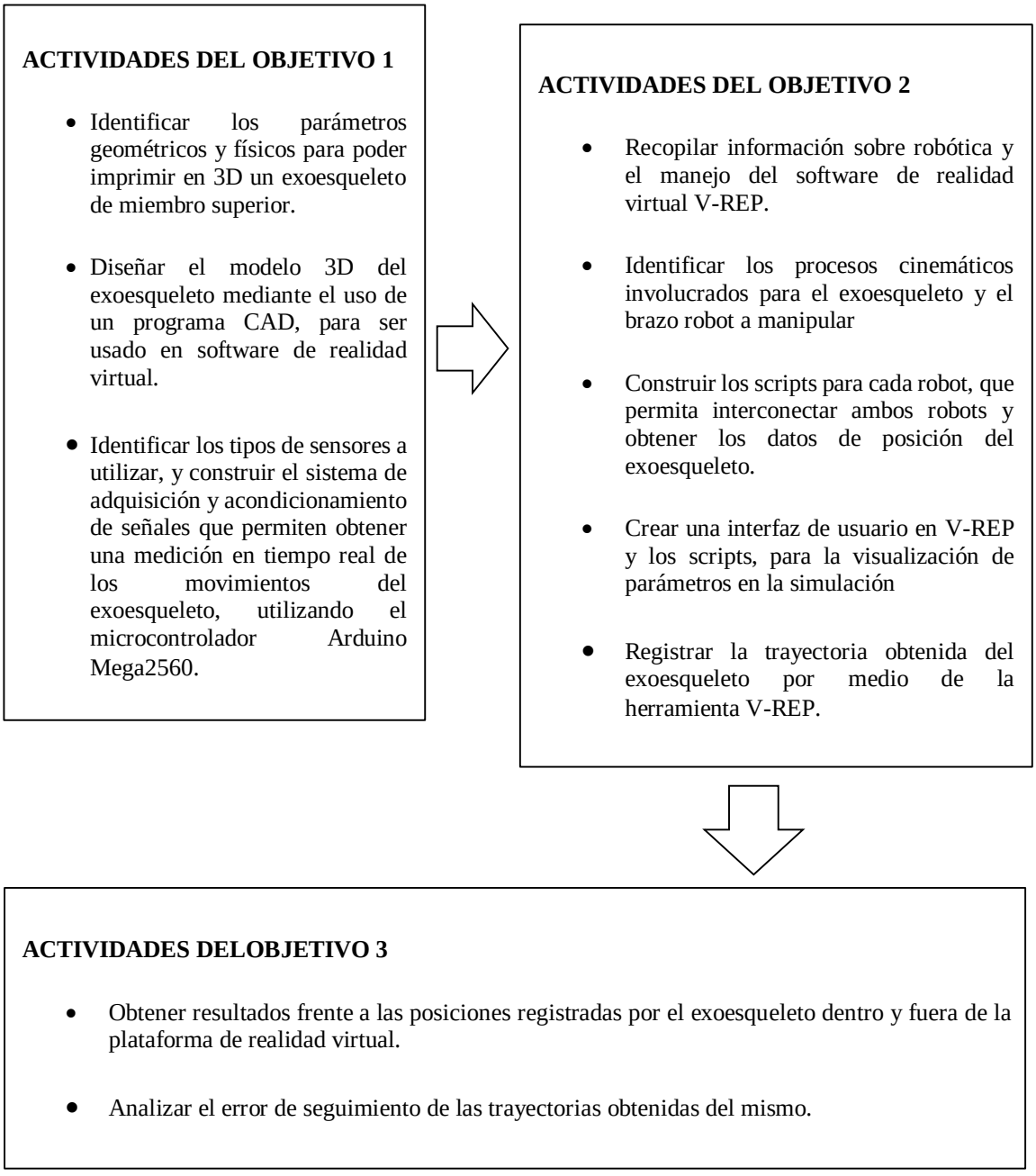


Diagrama 2. Diseño metodológico del proyecto.

Fuente: Autor

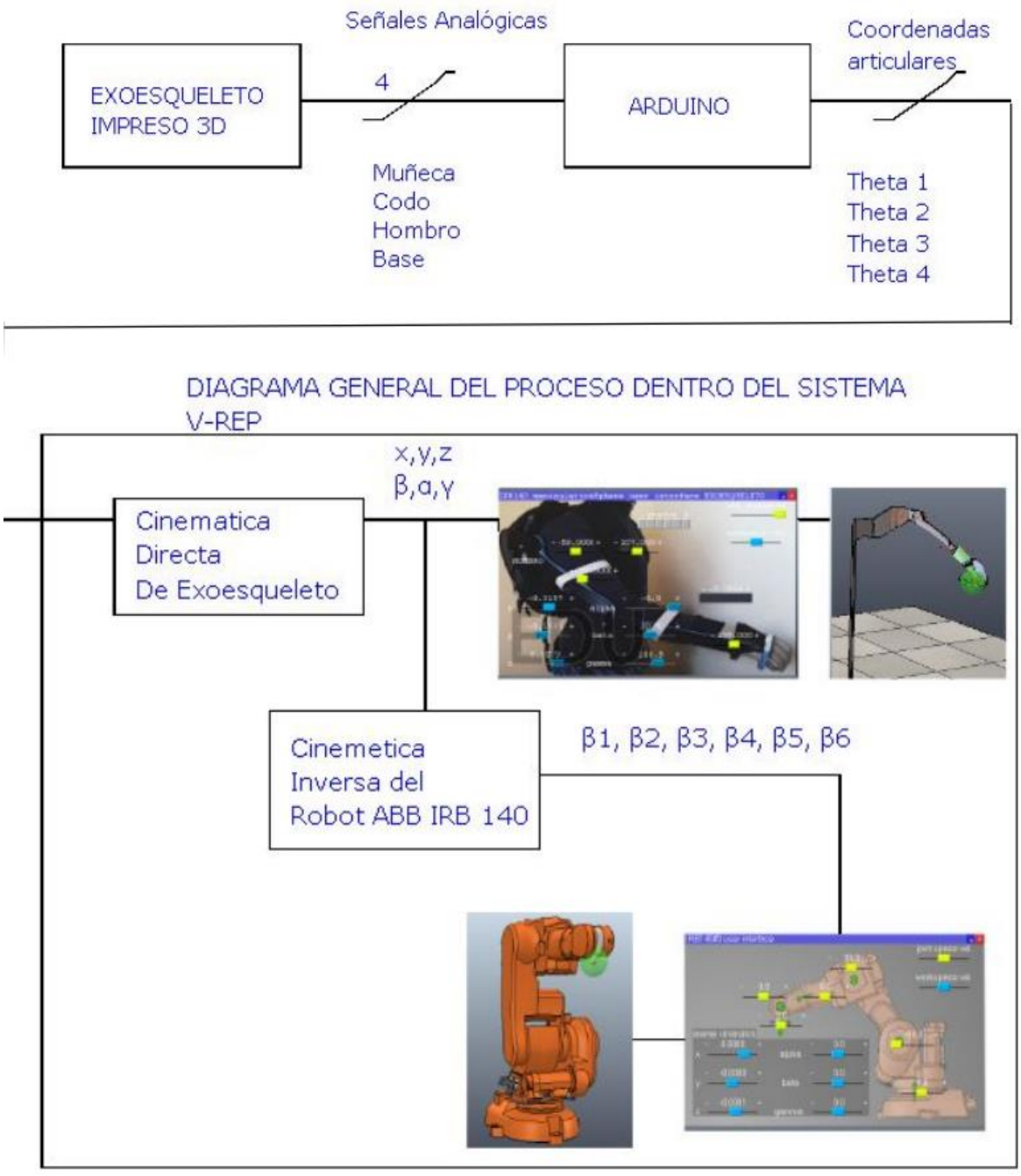


Diagrama 3. Proceso general del proyecto.

Fuente: Autor

6.4.2 Desarrollo Objetivo 1

6.4.2.1 Construcción de exoesqueleto

Se desarrolló la construcción del exoesqueleto en impresión 3D utilizando como material principal el polímero poli láctico (PLA). Este material al ser biodegradable contribuye con el cuidado y protección al medio ambiente y el costo de fabricación es bajo en comparación con otros materiales disponibles en el mercado actual; la impresión 3D ayudará a desarrollar nuevos prototipos a futuro; esta investigación permite ser mejorada a partir de nuevas ideas que complementen su función inicial.

Se tomó en cuenta como principal objetivo que la construcción de este exoesqueleto sea ergonómica, adaptable y de fácil transporte; las dimensiones que se tomaron como parámetros iniciales fueron adaptadas al cuerpo del autor de esta investigación; debiendo permitir movimientos en cuatro grados de libertad contando las siguientes articulaciones: dos grados de libertad en el hombro, uno en el codo y otro en la muñeca, permitiendo así su rotación y fácil manipulación. Es por ello por lo que se utilizaron ejes rotacionales y no del tipo translación.

En la siguiente tabla se explican las medidas tomadas del brazo del autor, las cuáles son los parámetros iniciales para la construcción del exoesqueleto:

Ancho de espalda	380 mm.
Distancia entre omoplato a hombro	120 mm.
Antebrazo	330 mm.
Brazo	300 mm.
Ancho de muñeca	60 mm.

Tabla 1. Medidas del brazo del autor

Fuente: Autor

Como se tomó en cuenta la forma del brazo humano para modelar la curvatura de los eslabones, permitió que el diseño fuera ergonómico al momento de colocarse, su forma se adapta a la del brazo mismo y su manipulación fue mucho más efectiva. Se diseñó el modelo CAD con estos parámetros, utilizando la implementación de rodamientos de bolas estriado (680 IRS) para la disminución de esfuerzos en las

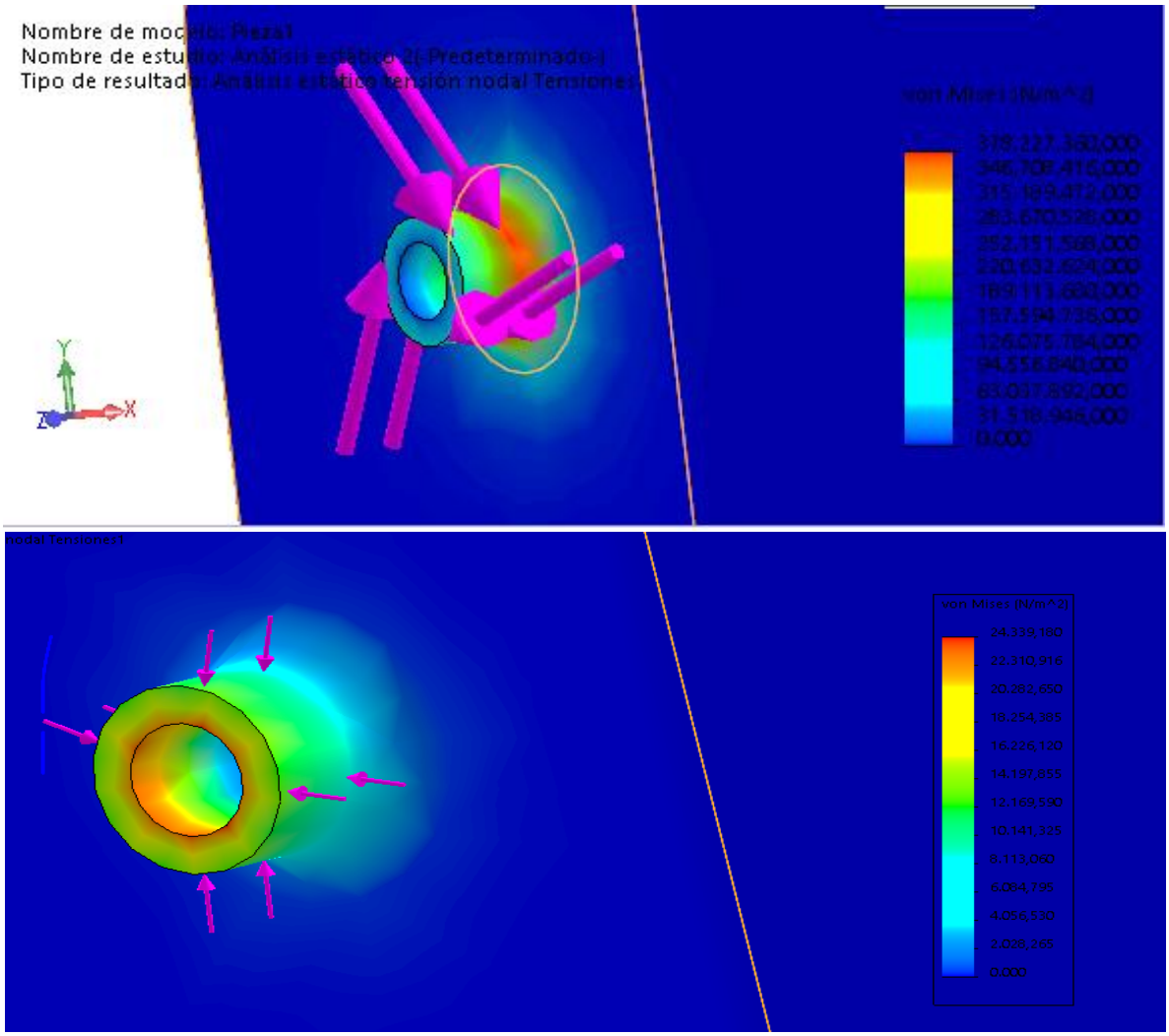
rotaciones de las articulaciones del exoesqueleto, así como el tipo de sensor que se describe en el siguiente numeral para censar cada grado de libertad:

Figura 7. Rodamientos en exoesqueleto.

Fuente: Autor

En la figura 7 se observan los rodamientos que tienen los soportes en cada articulación, los cuales disminuyen los esfuerzos al momento de ejecutar un movimiento. En primera medida, esto se realizó con un diámetro menor, por tanto, el esfuerzo se distribuyó en la mayor parte del soporte; se decidió aumentar el diámetro y redondez de la arista interna a 3 mm de radio, en la zona de unión con el eslabón para disminuir el hecho de una posible fractura en la parte más crítica (roja) de la pieza. Como se observa en la figura 8:

Figura 8. Simulación de un análisis estático.



Fuente: Autor

Se identificaron los puntos más críticos en un análisis estático por esfuerzos combinados Von Mises, por lo tanto, se usó la siguiente fórmula para identificar las fuerzas y momentos máximos aplicados en el exoesqueleto.

Calculo del factor de seguridad

$\Sigma F = 0$

$\Sigma M = 0$

Ecuaciones 5 y 6

Calculo de la carga máxima (Fb) de un brazo humano promedio sobre el exoesqueleto

Peso del brazo humano promedio 5 Kg.

Fb= -5Kg* 9.81 m/s2

Fb=-49.1 N

Cálculo del momentum (Mb) ejercido por la carga máxima en la distancia más lejana a la base.

Distancia máxima, brazo extendido= D = 0.75m

Mb= Fb*D

Mb=-49.1N*0.75m

Mb=-36.78 N.m

Esfuerzo de Von Mises máximo (σm) obtenido de simulación de Solidworks con los parametros calculados anteriormente,

$\sigma m = 378227360\ N/m\text{^}2 = 378.23\ MPa$

PROPIEDAD MECÁNICA DEL PLA

$Limite\ Elastico\ PLA = \sigma e = 2346.5\ MPa$

Calculo del factor de seguridad por Von mises (ηVm)

$\eta Vm = \sigma e/\sigma m$

ηVm= 6.2

Un Factor de seguridad ηVm=6, indicando que el elemento puede soportar 6 veces la carga de diseño máxima.

En la figura 9 se puede observar el modelo del exoesqueleto ya completo e impreso el cual se exporto en formato STL dentro del programa de realidad virtual V-REP, permitiendo ser exportado con las mismas medidas que el archivo impreso en 3D.

Figura 9. Modelo impreso en 3D de exoesqueleto

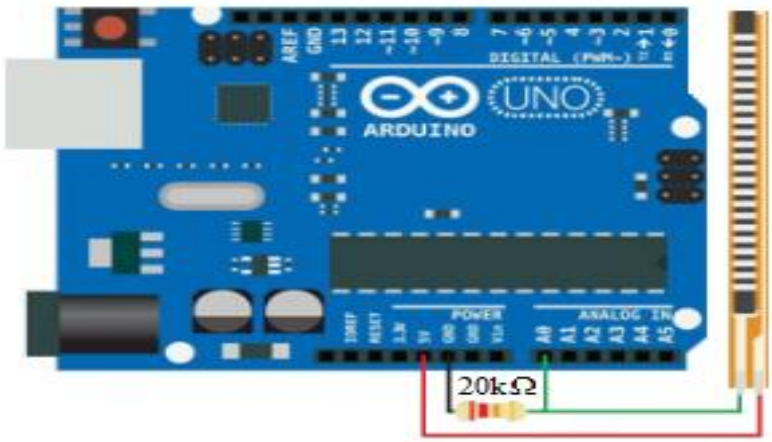
Fuente: Autor

6.4.2.2 Tipos de sensores y programación de sensores en Arduino y V-REP

Se utilizaron sensores de resistencia variable, ya que para la medición de las articulaciones de un movimiento rotacional es preciso utilizar potenciómetros lineales de precisión, los cuales no son afectados de forma significativa por el ruido externo y/o fallas de fabricación. La articulación de la muñeca por otra parte fue equipada con una resistencia variable en forma de cinta llamada sensor flexible (sensor Flex); este sensor permite disminuir la complejidad de la muñeca, ya que al ser un eje interno que no se puede invadir a diferencia de los ejes del codo y hombro es preciso tener más libertad al momento de ejecutar un movimiento de rotación.

Para el correcto funcionamiento de este sensor fue necesario diseñar un divisor de tensión utilizando una resistencia variable de 20 k como se evidencia en la siguiente figura:

Figura 10. Ejemplo de conexión con el microcontrolador Arduino mega2560

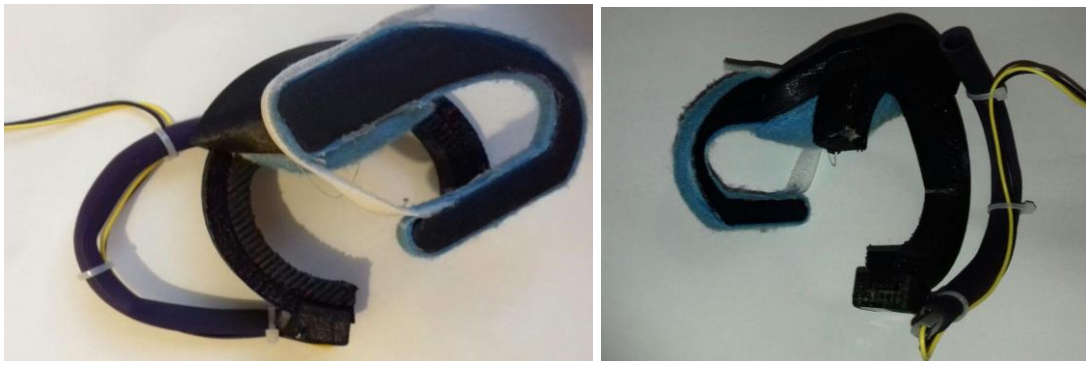


Fuente: Autor

El sensor Flex cambia su resistencia cuando se flexiona para que se pueda medir ese cambio usando uno de los pines analógicos de algún microcontrolador como Arduino. Pero para hacer eso, se necesita de una resistencia fija de 20 K Ohm. Esto se llama divisor de tensión y divide los 5v, entre el sensor de flexión y la resistencia.

Dentro de la flexión del sensor son elementos resistivos de carbono dentro de un sustrato flexible y delgado. (Más carbono significa menos resistencia). Cuando se dobla el sustrato del sensor produce una salida de resistencia en relación con el radio de curvatura. Con un sensor típico Flex, una flexión de 0° da la resistencia de 10K Ohm, para una flexión de 90° que da entre 30 a 40 K Ohm. En la figura 11 se observa el sensor tipo flex y su forma de funcionamiento en la muñeca del exoesqueleto:

Figura 11. Sensor tipo Flex dentro protector de polímero flexible en exoesqueleto.



Fuente: Autor

- **Caracterización de potenciómetros en el exoesqueleto**

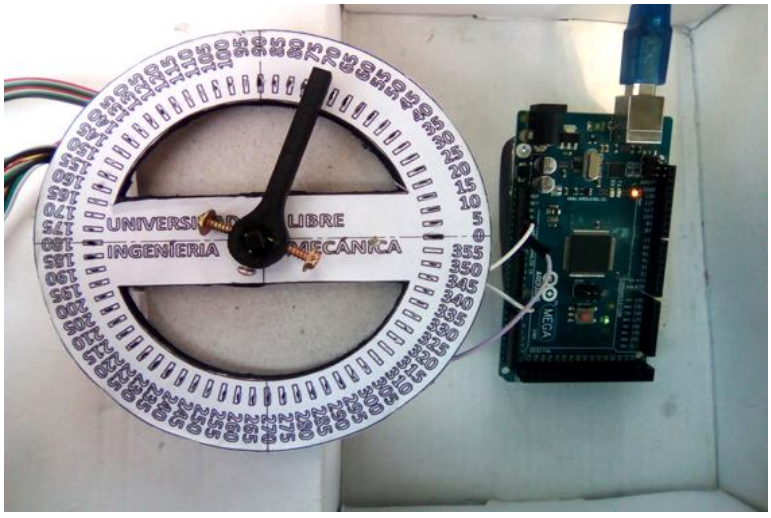
Se diseñó y construyó un posicionador de ángulos para potenciómetros, el cual permite ubicar el potenciómetro que se desea utilizar en el exoesqueleto y observar los ángulos de trabajo que está faculta utilizar.

Figura 12. Posicionador con punta de aguja para los potenciómetros.

Fuente: Autor

En la figura 12 y la figura 13, se observa el posicionador, el cual tiene una punta de aguja que permite encajar asertivamente en las ranuras previamente construidas con una impresora 3D.

Figura 13. Posicionador con punta de aguja, vista superior.



Fuente: Autor

Este diseño permite obtener un margen más bajo de incertidumbre en la medida tomada, el resultado se anexa en el siguiente código realizado en Matlab, siendo Theta el ángulo de giro, y X la palabra binaria extraída del Arduino. Potenciómetro:

```
Theta(θ) = [25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115 120 125 130 135 140
145 150 155 160 165 170 175 180 185 190 195 200 205 210 215 220 225 230 235 240 245 250 255
260 265 270 275 275.25]';
X = [0 12 26 46 65 85 105 123.5 142 159.5 178 193 212 230 248 269 290 310 332 353 374 395 420
442 467 489 511 533 566 579 604 627 647 670 692 714 734 753 776 797 820 840 858 879 899
918.5 941 958 980 1004 1018 1023]';
N = length(x);
polyfit (N, Theta,1)
```

Se realizó con la herramienta de Matlab POLYFIT, una regresión lineal, lo que permitió obtener las constantes de una ecuación lineal:

$$\theta = (m * X) + b$$

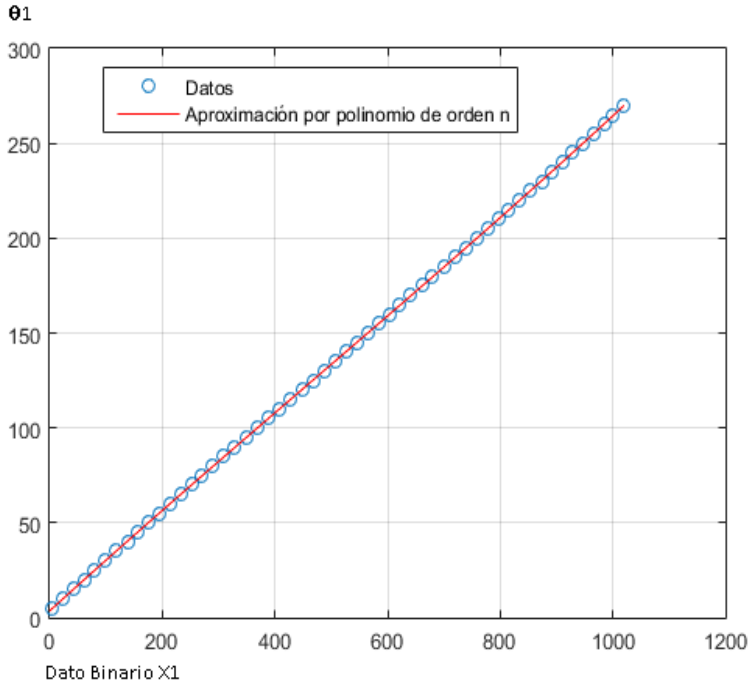
Ecuación 8. Característica línea de palabra binaria a ángulos

De este modo, se obtuvo los puntos de operación y caracterización de cada potenciómetro.

Figura 14. Gráfica de caracterización de Potenciómetro

La ecuación que relaciona Angulo vs dato binario

$$\theta_1 = Ángulo\ del\ potenciómetro$$
$$X_1 = Dato\ Binario\ del\ Potenciómetro$$



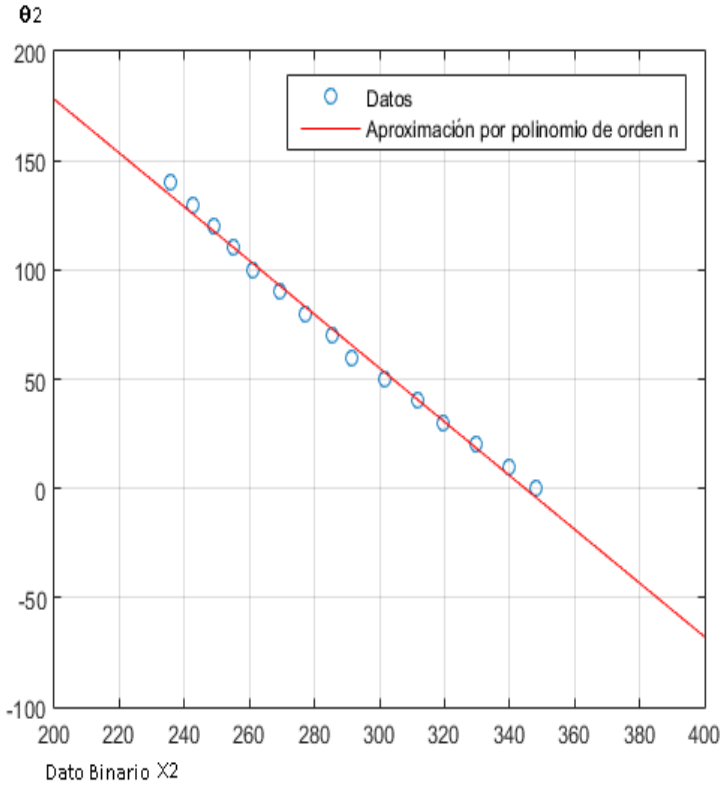
$$\theta_1 = 0.2589\ X_1 + 4.23$$

Error Máximo entre ecuación característica y puntos obtenidos= 1,27%

Fuente: Autor

Figura 15. Gráficas de caracterización de Sensor Flex

$$\theta_2 = Ángulo\ del\ Sensor\ Flex$$
$$X_2 = Dato\ Binario\ del\ Sensor\ Flex.$$

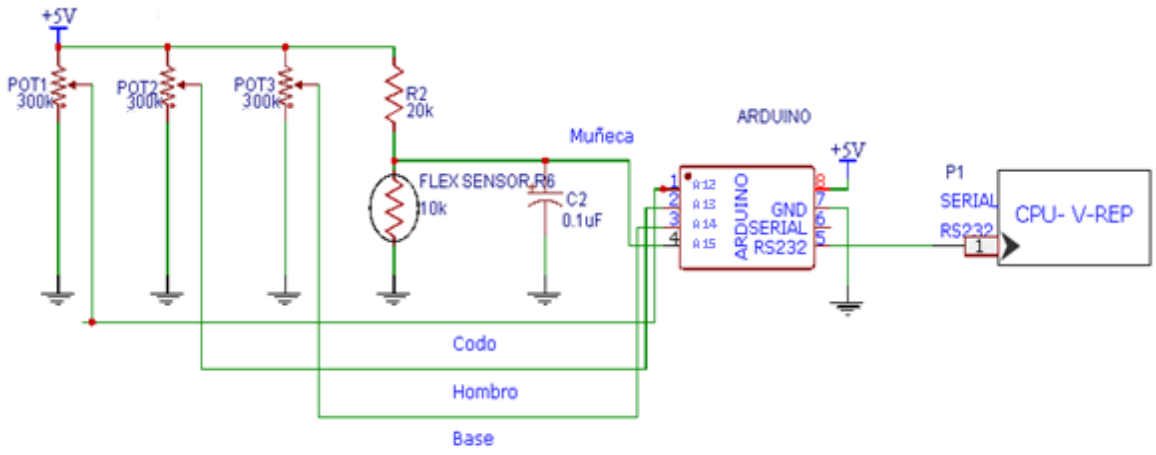


$$\theta_2 = -0.8084 * X_2 + 344.42$$

Error Máximo entre ecuación característica y puntos obtenidos = 1,95%

Fuente: Autor

Figura 16. Circuito electrónico del exoesqueleto.



Fuente: Autor, elaborado con FRITZING.

El rango de operación de las palabras binarias de cada sensor está entre 0 y 1023. Para evitar errores en la comunicación y determinar a cuál sensor corresponde las palabras binarias transmitidas, se le suma una constante diferente al dato binario correspondiente a cada potenciómetro antes de transmitirla por el puerto serial, con lo cual los datos de cada sensor quedan en un rango de operación. Para disminuir el ruido de medición se toman 10 datos y se envía el valor promedio vía puerto serial al programa V-REP el cual incluye la ecuación lineal de cada sensor para hacer la conversión a ángulo en grados. El programa embebido en la tarjeta Arduino es el siguiente:

```
//Código Arduino, obtención de la palabra binaria según el grado del sensor
int Lecturas1[10]; //Vector de lecturas.
int Lecturas2[10];
int Lecturas3[10];
int Lecturas4[10];
int Val1, i1 = 0, Total1 = 0, Promedio1 = 0;
int Val2,       Total2 = 0, Promedio2 = 0;
int Val3,       Total3 = 0, Promedio3 = 0;
```

```
int Val4,        Total4 = 0, Promedio4 = 0;
void setup()
{
  Serial.begin(9600);
{
for(i1=0; i1< 10; i1++) //Inicialización del vector.
Lecturas1[i1] = 0;
}
{
for(i1=0; i1< 10; i1++) //Inicialización del vector.
Lecturas2[i1] = 0;
}
{
for(i1=0; i1< 10; i1++) //Inicialización del vector.
Lecturas3[i1] = 0;
}
{
for(i1=0; i1< 10; i1++) //Inicialización del vector.
Lecturas4[i1] = 0;
}
}
void loop()
{

//tomar datos articulacion 1
 {
for(i1=0; i1< 10; i1++)
Lecturas1[i1] = analogRead(12);
}
```

```
Total1=0;
 {
for(i1=0; i1< 10; i1++) //Inicialización del vector.
Total1 = Total1+Lecturas1[i1];
}
Promedio1 = Total1 / 10;
Val1=Promedio1+5000;

//tomar datos articulacion 2
 {
for(i1=0; i1< 10; i1++)
Lecturas2[i1] = analogRead(13);
}
Total2=0;
 {
for(i1=0; i1< 10; i1++) //Inicialización del vector.
Total2 = Total2+Lecturas2[i1];
}
Promedio2 = Total2 / 10;
Val2=Promedio2+1025;

//tomar datos articulacion 3
 {
for(i1=0; i1< 10; i1++)
Lecturas3[i1] = analogRead(14);
}
Total3=0;
 {
for(i1=0; i1< 10; i1++) //Inicialización del vector.
Total3 = Total3+Lecturas3[i1];
```

```
}
Promedio3 = Total3 / 10;
Val3=Promedio3+2054;

//tomar datos articulacion 4
 {
for(i1=0; i1< 10; i1++)
Lecturas4[i1] = analogRead(15);
}
Total4=0;
 {
for(i1=0; i1< 10; i1++) //Inicialización del vector.
Total4 = Total4+Lecturas4[i1];
}
Promedio4 = Total4 / 10;
Val4=Promedio4+3075;

//enviar datos por puerto serial
 Serial.print(Val1);
 Serial.print(",");
 Serial.print(Val2);
 Serial.print(",");
 Serial.print(Val3);
 Serial.print(",");
 Serial.print(Val4);
 Serial.print(",");
delay(10);
  }
```

Para la obtención de los datos a través del software V-REP se creó un Dummy MENÚ - ADD – DUMMY; permitiendo la comunicarse con el COM del puerto serial del Arduino al software de realidad virtual. Como la tarjeta Arduino envía los datos separados por comas; para que el programa de V-REP reconozca la secuencia de datos se crearon unos condicionales por cada sensor con base en los rangos numéricos de cada articulación. Para el potenciómetro “dos” los valores fueron de 1025 hasta 2053, en el potenciómetro “tres” los valores fueron de 2054 hasta 3073, el potenciómetro “cuatro” fueron los valores mayores de 5000 hasta 6024. Y por último para el sensor Flex de la articulación “cuatro” los valores de la palabra binaria van de fueron de 3075 a 4998.

De este modo se enviaron los datos correctos hacia el software V-REP. El código por su parte, una vez ha obtenido los datos, le resta a cada valor la cantidad que en el código de Arduino fue sumado; así se obtuvo nuevamente el valor correcto en grados. Para el procedimiento anterior se utilizó el siguiente código:

```
-- Este codigo fue obtenido y modificado gracias a la pagina del programa,
www.forum.coppeliarobotics.com/

simSetThreadSwitchTiming(2) -- Default timing for automatic thread switching

simDelegateChildScriptExecution()

--defining the serial port number

portNumber="\\\\.\\COM2"

--could be defined as followed

--portNumber=[[\\.\COM12]]

baudrate=9600

serial=simSerialOpen(portNumber,baudrate)

--getting the cube handle and initial pose
```

```
while (simGetSimulationState()~=sim_simulation_advancing_abouttostop) do

    --read a full line
    str=simSerialRead(serial,32,true,'\n',10)
    if str ~= nil then
        --print(str)
        local token
        val={}
        cpt=0
        --extracting the two values in str separated by a ,
        for token in string.gmatch(str, "[^,]+") do
            cpt=cpt+1
            val[cpt]=token

              local
a1=tonumber(token)
        if a1>5000 and a1<6024 then
pot1=((a1-5000) * 0.2589)  + 4.2
pp1=((pot1+236)*math.pi/180)
simSetStringSignal("x1",pp1)
            end
local
a2=tonumber(token)
if  a2>=1025 and a2<2049 then
pot2=((a2-1025) * 0.2589)  + 4.2
        pp2=((pot2-25)*math.pi/180)
simSetStringSignal("x2",pp2)
```

```
end
local
a3=tonumber(token)
if  a3> 2050 and a3<3074 then
 pot3=((a3-2050) * 0.2589)  + 4.2
pp3=((pot3-80)*math.pi/180)
simSetStringSignal("x3",pp3)
   end
local
a4=tonumber(token)
if  a4> 3075 and a4<4999 then

pot4=((a4-3075) * -1.2292)  +423.81
pp4=((pot4-15)*math.pi/180)
simSetStringSignal("x4",pp4)
   end
end


  -- not wasting time here, we force the switching of the thread
   simSwitchThread()
end
end
```

Una vez separados los datos, se enviaron al robot virtual de tipo exoesqueleto, el cual, utiliza la cinemática directa, así se obtuvo la posición (x, y, z) y (alfa, beta, gamma) del efector final.

6.4.3 Desarrollo de Objetivo 2

6.4.3.1 Cinemática directa e inversa de exoesqueleto y robot serial.

Para lograr realizar una teleoperación entre el exoesqueleto y el brazo robot serial virtual fue necesario obtener la cinemática inversa del robot serial y la cinemática directa del exoesqueleto, así al efectuar un movimiento en el exoesqueleto los ángulos registrados en la programación dieron la posición final (x, y, z, Alfa, Gamma y Beta) y envío la posición final (x, y, z, Alfa, Gamma, y Beta), a la programación del robot serial el cual mediante el método de cinemática inversa entregó los ángulos necesarios para llegar a dicha posición.

El módulo de cálculo de cinemática inversa de V-REP es muy flexible y de gran alcance permite manipular varios tipos de mecanismos en modo ik (cinemática inversa) o en modo fk (cinemática directa). En la mayoría de casos el solucionar un problema cinemático inverso puede conllevar a más de una solución, teniendo diferentes posiciones y orientaciones del extremo del robot para el mismo. para la resolución de la cinemática inversa del manipulador se utilizó el módulo de cálculo IK, el cual, a partir de determinados algoritmos, resuelve el problema cinemático.

Para activarlo se deben realizar los **siguientes pasos:**

1 Se importa el ensamble en STL, se agregan articulaciones rotacionales a cada eslabón utilizando la opción ADD-JOINT.

2 Como sugerencia es necesario crear pasadores en el CAD Simulando la posición de las articulaciones rotacionales, de este modo cuando se ubiquen las articulaciones rotacionales, de un modo más fácil utilizando la herramienta Object / Item Translation / Position, se puede ubicar la articulación rotacional (Joint) en la misma posición del pasador automáticamente realizando los siguientes pasos:

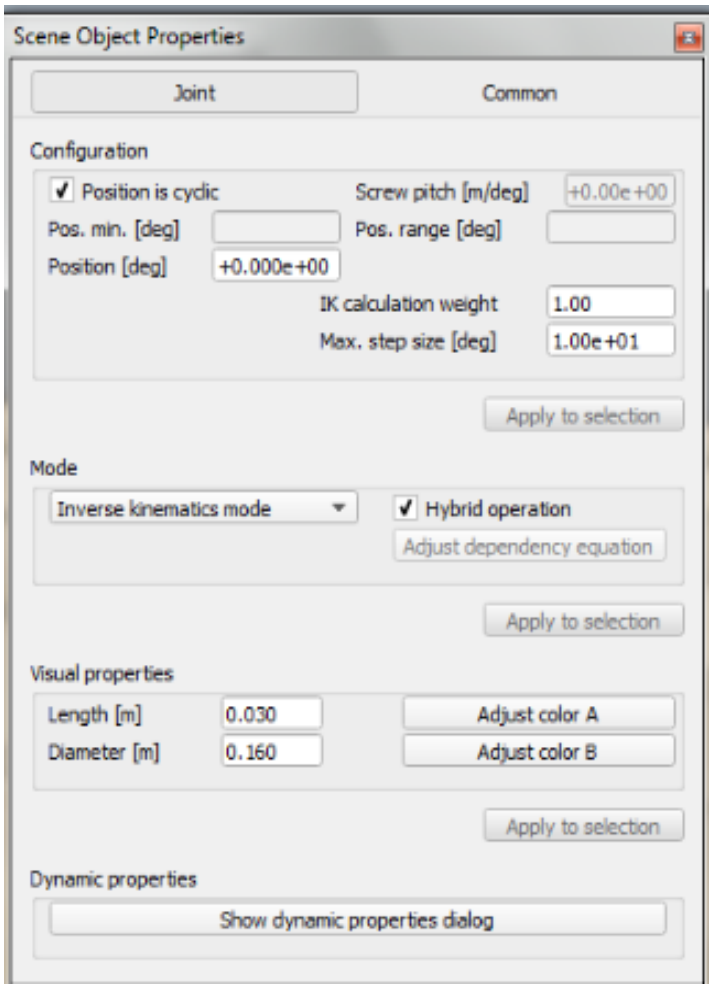
- Se selecciona con click izquierdo, la articulación rotacional (Joint) y con ayuda de la tecla CTRL mantenido, Seleccionar el pasador dónde se quiere ubicar el Joint, se da clic en APPLY TO SELECTION tanto en POSITION, como en TRANSLATION.

3 Establecer relaciones de parentesco entre los eslabones, articulaciones y los demás objetos que tenga el mecanismo, utilizando el árbol de elementos empezando desde el último elemento se selecciona y con la tecla ctrl + click izquierdo se selecciona el que está atrás posterior a este, se da clic derecho MENÚ-EDIT-MAKE LAST SELECTED OBJECT PARENT, Para que con este fin se tenga

la cadena cinemática desde la base hasta el último eslabón del robot repitiendo este proceso hacia atrás.

4 Entrar a cada articulación (Joint) y activar el modo Joint como Inverse kinematics mode, V-REP internamente será el que calcule para cada articulación el valor correspondiente para resolver el mecanismo del brazo robot.

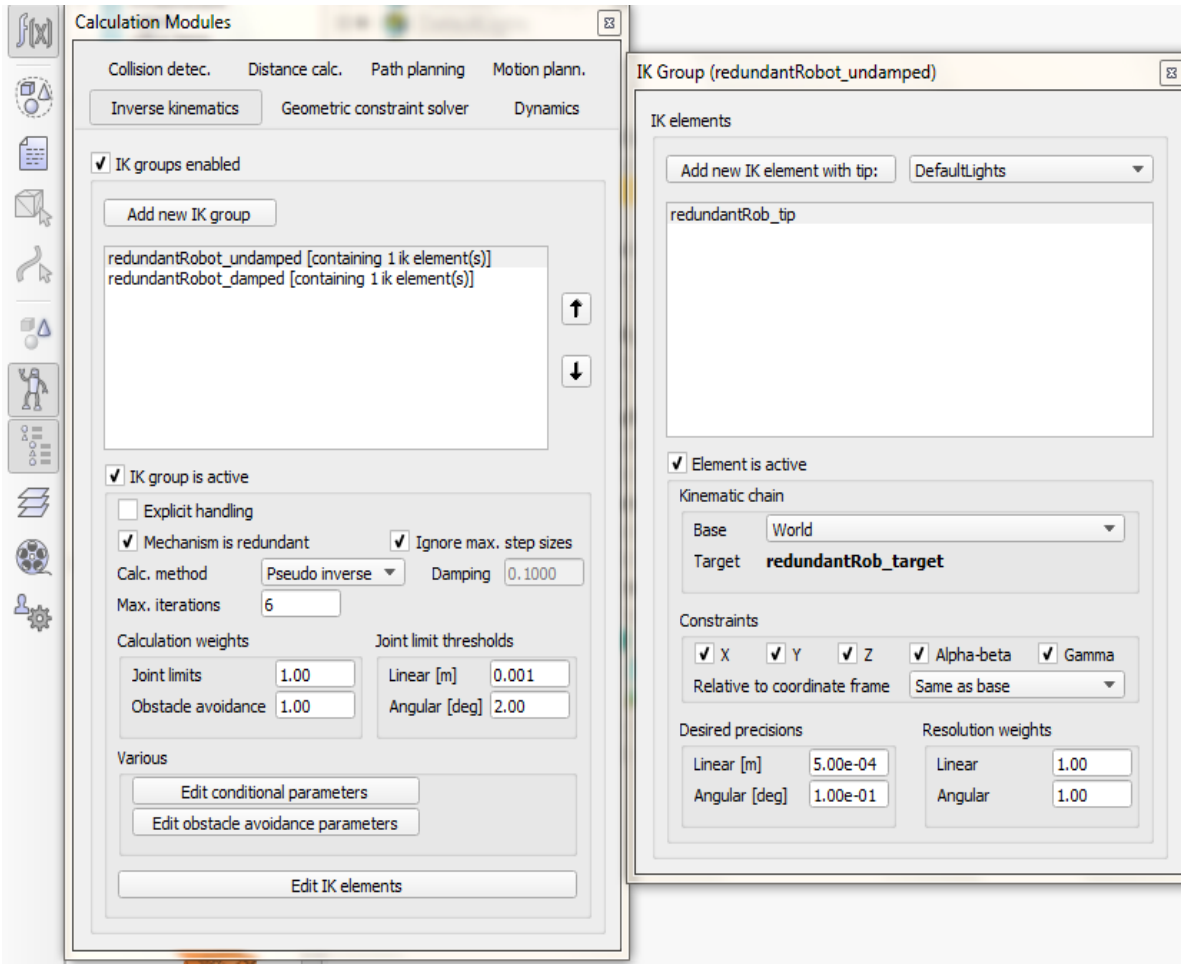
Figura 17. Scene Object Properties



Fuente: V-REP.

5 Ahora se da seleccionar la figura f(x) (Calculation modules) En el diálogo cinemática inversa dar clic en agregar un nuevo grupo IK, a este nuevo grupo nombrado “IK-Group”. se da click en Edit IK elements. Desplegando un nuevo menú IK Group [nombre del grupo]. En este punto se añade el dummy TIP, el cual fue creado para identificar la punta final del mecanismo. Se selecciona, como se ve en la imagen los mismos checks, y menús.

Figura 18. Calculation Modules e IK Group



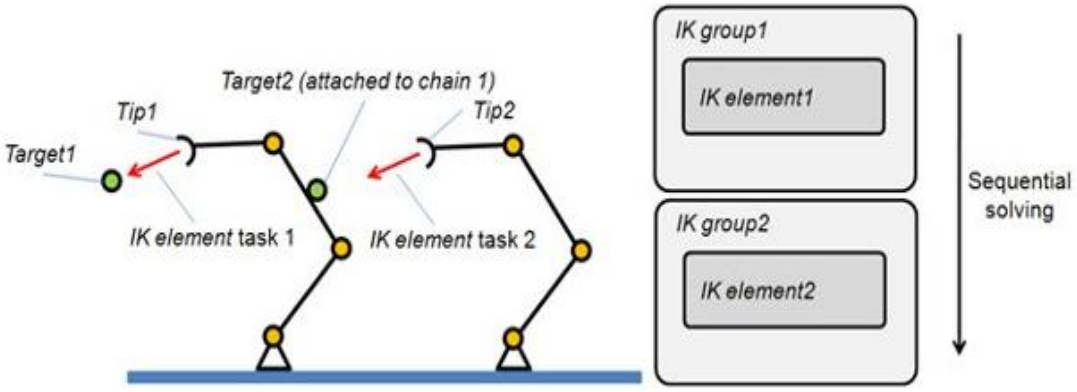
Fuente: V-REP

6 Se pueden seleccionar dos métodos de cálculo tipo pseudo inverso y cálculo DLS, Se habilita el estudio de cinemática inversa, en el Robot ABB, con el módulo de Inverse Kinematic, programando el uso de dos grupos IK, donde comenzará resolviendo la cinemática inversa por el método de pseudoinverse de Moore-Penrose y al momento de encontrar singularidades pasar a resolver la IK, por el método DLS (mínimos cuadrados con factor de amortiguamiento).

7 Cuando en la simulación se rompe la cadena cinemática se puede utilizar el método DLS, así de este modo las simulaciones de la cinemática en puntos singulares resultan más estables, de igual modo el damping o coeficiente de

amortiguación permite mejorar la resolución en la cinemática, más sin embargo se vuelve un poco más lenta a mayores iteraciones con el . Se puede para un mismo mecanismo tener dos grupos de IK el cual si no logra resolver el primero como se ve en la imagen, el segundo lo resuelve el modo secuencial.

Figura 19. IK Group secuencia de procesado.



Fuente: Manual V-REP V3.

6.4.3.2 Registro de la trayectoria del exoesqueleto al brazo robot virtual.

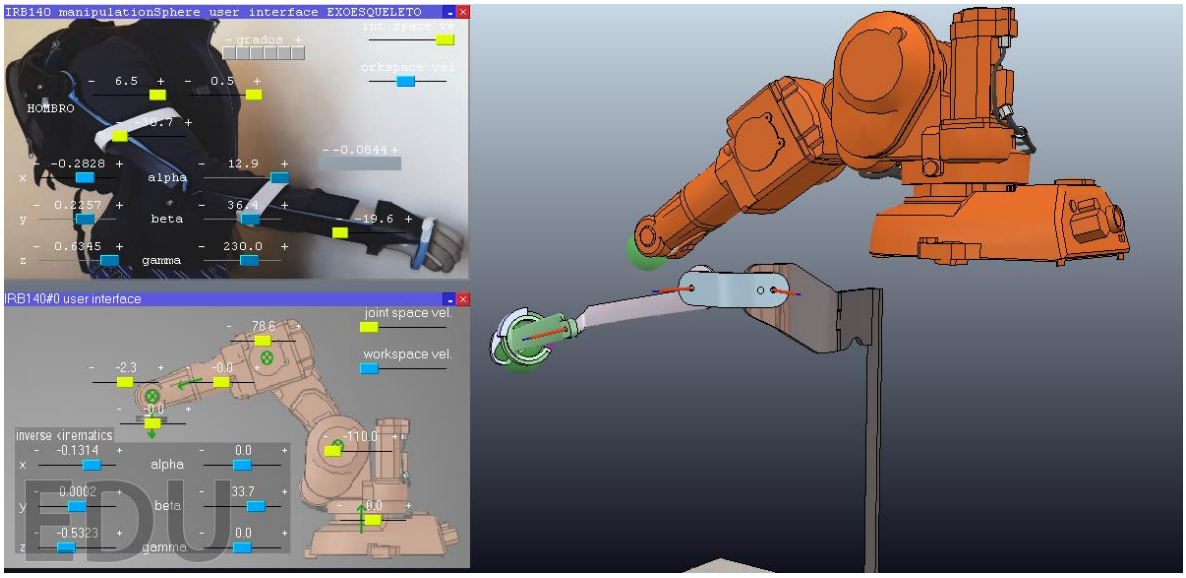
Para el registro de la trayectoria luego de haberse cumplido los pasos anteriores, el código del Arduino se programó para permitir obtener las señales analógicas de los sensores y cifrar dichas señales en una palabra binaria que se envía por puerto serial. De forma simultánea, el Script diseñado en el software de realidad virtual se comunica con el puerto serial para captar estos datos, organizarlos y descifrarlos, para conocer qué dato corresponde a cada articulación del exoesqueleto, convertirlos a grados y se incorpora al Exoesqueleto virtual.

Una vez se obtuvieron los grados en el exoesqueleto virtual, y mediante el uso de una interfaz de usuario interconectada en la programación de un Script con una cinemática directa adecuada se logró visualizar los puntos coordenados del efector final del exoesqueleto (x, y, z, alfa, beta y gamma).

Estos datos son enviados al brazo robot virtual, el cual, usando el modo de cinemática inversa, llegará a la posición solicitada. La interfaz de usuario se debe programar utilizando fácilmente un código ayuda del robot que se va a manipular

simulando un esclavo, en este caso el robot ABB IRB 140. Esta interfaz consta de dos partes: una para la cinemática inversa, la cual como se observa en la imagen con sliders de color azul y la segunda es de la cinemática directa con sliders de color amarillo.

Figura 20. Interfaz de usuario del exoesqueleto y del robot virtual



Fuente: Autor

El funcionamiento de esta IU (Interfaz de usuario) permite al brazo robótico ser programado rápidamente en IK (Cinemática inversa) y FK (Cinemática Directa). Con solo manipular las sliders; al igual que la velocidad a la cual se desea mover. Se usa esta misma interfaz con el exoesqueleto en este proyecto, realizando la programación adecuada en el código del script, el cual es el encargado de manipular el exoesqueleto.

- **Programación del Script para enviar datos:**

Para enviar los datos de un Script a otro se utiliza el siguiente código (Ejemplo):

```
pp1=pot1-100
simSetStringSignal("x1",pp1)
```

I. Para recibirlo se usa el siguiente código en el Script final:

```
local B0=simGetStringSignal("x1")
if B0 then
B0=B0*math.pi/180
 simAddStatusbarMessage("robot1: "..B0)
```

II. Como se observa en el primer código el x1, representa el nombre de la variable que se va a enviar, mientras el "pp1" corresponde a la variable en sí. Este código se utiliza para enviar los datos del Script que se conecta con el puerto serial y envía los datos con esta programación al exoesqueleto robótico virtual, posterior a esto la posición del efector del exoesqueleto robótico virtual se comunica con ayuda de este mismo código para enviar los datos al robot ABB IRB 140.

- **Programación del Script del exoesqueleto:**

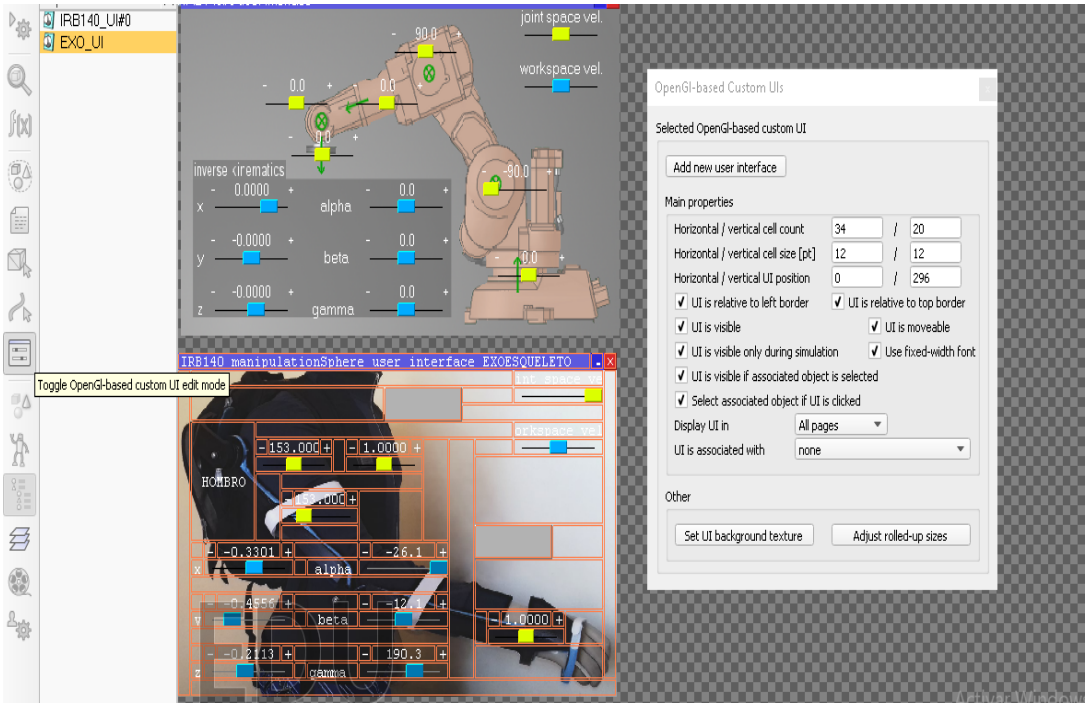
(Observar ANEXO A - SCRIPT DE EXOESQUELETO)

Se nombra en el Script del exoesqueleto las variables identificadas a continuación:

1. Se da como variable la identificación del TIP, el TARGET, el TARGET SPHERE y TARGET SPHERE BASE (esfera verde). Posterior a esto se nombran los Joints o articulaciones. (Es recomendable que tengan un nombre en común y una secuencia de números, de esta forma es simple de identificarlos y acorta el código)

2. Seguido a esto se crea la interfaz de usuario, para ello hay que programarla ingresando al menú Toggle OpenGL-based custom UI edit mode:

Figura 21. Toggle OpenGL-based custom UI edit mode.



Fuente: Autor

3. Al entrar al menú Toggle OpenGL-based custom UI edit mode, Se observa la interfaz de usuario del robot IRB 140 la cual se puede copiar, pegar y cambiar el nombre a EXO_UI. Entrando al menú set UI background Texture Se puede incluir una imagen de fondo, en este caso se observa el exoesqueleto.

4. Cada botón y Slider tienen un número propio; este número es utilizado en el Script para asignar una función a dicho elemento de la interfaz de usuario.

5. Una vez posicionados los Botones, Sliders y Labels, cómo se observa en la imagen, se puede volver al Script seleccionando nuevamente el botón Toggle OpenGL-based custom UI edit mode.

6. Se programa dos tipos de IK; por un lado, se programan tal como se explica en el numeral 6.4.3.1: En el primero, usar el modo Pseudo, y en el segundo usar el modo DLS (si tiene Damping = 1.0000)

```
ik1=simGetIkGroupHandle('EXO_undamped')
ik2=simGetIkGroupHandle('EXO_damped')
```

7. Por último, se debe modificar en el código la posición de los ángulos del exoesqueleto, los cuales serán modificados directamente por los datos obtenidos del Arduino, para ello se utiliza el siguiente código en la línea 261:

```
simSetJointPosition(armJoints[1],(B1))
simSetJointPosition(armJoints[2],(B2))
simSetJointPosition(armJoints[3],(B3))
simSetJointPosition(armJoints[4],(B4))
```

8. La otra parte del código se deja igual al código muestra del IRB 140. Se finaliza el código utilizando la función para enviar los puntos de coordenadas del efector final del exoesqueleto al robot IRB 140.

```
pp1=desiredConf[1]
simSetStringSignal("EXO1",pp1)
pp2=desiredConf[2]
simSetStringSignal("EXO2",pp2)
pp3=desiredConf[3]
simSetStringSignal("EXO3",pp3)
pp4=desiredConf[2]
simSetStringSignal("EXO4",pp4)
pp5=desiredConf[3]
simSetStringSignal("EXO5",pp5)
pp6=desiredConf[4]
simSetStringSignal("EXO6",pp6)
   end
```

- **Programación del Script Robot IRB 140:**

(Observar ANEXO B - SCRIPT DE ROBOT IRB 140)

I. Mirar la programación inicial recibiendo las coordenadas de posición del exoesqueleto utilizando el mismo código anteriormente enunciado.

II. En este código se cambió la posición final dando al robot las coordenadas que le permiten llegar a dicha posición. Como la cinemática directa ya está contemplada, utilizando el algoritmo de V-REP, al realizar el cálculo, el efector final del robot llega a dicha posición en su IK, línea 177.

```
currentConf[1]=w1
currentConf[2]=w2
currentConf[3]=w3
currentConf[4]=w4
currentConf[5]=w5
currentConf[6]=w6
end
```

6.4.4 Desarrollo de Objetivo 3

6.4.4.1 Cálculo de error de la trayectoria.

Con el objetivo de recrear una zona de trabajo se construyó un plano cartesiano con aristas de 30x30x30 cm. donde por ejemplo una persona emplea el efector final del brazo robot serial portando una cámara exploratoria o un efector para manipular un objeto de una posición a otra.

Se obtuvo los mesurandos que permitió medir el error dentro de esta zona, de los cuales se tomaron medidas a lo largo de cada eje (x, y, z). Con el programa V-REP se obtuvo la posición inicial en coordenadas (x, y, z) hasta el punto final simuladas dentro de la realidad virtual. La distancia absoluta se obtuvo con la siguiente ecuación:

$$d_{EXO} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Ecuación 9. Distancia entre puntos en el espacio

Se realizó la misma prueba sobre cada eje de 30 cm subdividido en una escala cada 5 cm para un total de 7 medidas por eje, se calculó el error entre el valor real y el valor obtenido en el software.

7. RESULTADOS Y DISCUSIÓN

7.1 Análisis Estático

Mediante el análisis estático se aplicó el peso promedio de un brazo como carga máxima siendo de 5 Kg, dando como fuerza ejercida del mismo de 49.1 N, y la distancia máxima entre base y efector final fue de 0,75m, es su totalidad se obtuvo un momentum en la articulación de la base de 36,78 N.m. Con el uso de estos datos se obtuvo de la simulación por análisis estático dentro del software SolidWorks, el esfuerzo de Von Mises Máximo de 378.23 MPa. El material para la impresión 3D utilizado en este diseño fue el PLA (polímero poli láctico) un polímero biodegradable, no tóxico y versátil que proviene de residuos renovables. Dando un factor de seguridad de 6, por tanto, este diseño soporta 6 veces las cargas a las que va a trabajar, siendo una construcción segura.

Para el diseño, construcción e impresión del exoesqueleto se tuvieron en cuenta las siguientes dimensiones y grados de giro, para una persona de 1,85 m de alto de compostura delgada:

Dimensiones de cada eslabón		Angulos de giro
Ancho de espalda	380 mm.	0
Distancia entre omoplato a hombro	120 mm.	180
Antebrazo	330 mm.	269
Brazo	300 mm.	110
Ancho de muñeca	60 mm.	148

Tabla 2 Dimensiones de cada eslabón

Fuente: Autor

Para el uso en sujetos con diferentes fisonomías, es recomendable diseñar un exoesqueleto ajustable y ergonómico, que permita aumentar el cubrimiento en una población de características antropométricas diferentes.

Durante la realización de la impresión 3D, se tuvo en cuenta la calidad de impresión en 0.1mm por capa debido a que la tolerancia puede afectar la calidad de impresión,

como son aberturas para pasadores y demás superficies que requieren detalle. Para este diseño únicamente se tuvo en cuenta 4 grados de libertad, pero para próximas investigaciones se propone incluir más grados de libertad al exoesqueleto, aunque conlleva una dificultad mayor; este documento será un punto de partida para próximas investigaciones, facilitando el trabajo para el diseño, la obtención de la cinemática inversa y directa de los mecanismos, y la simulación de este tanto en exoesqueletos como robóticos.

7.2 Análisis de registro de error de trayectoria

Se trazaron sobre un plano cartesiano físico de 3 ejes coordenados una trayectoria lineal por cada eje (x,y,z); cada eje con una longitud de inicio a fin de 30 cm, de los cuales se tomaron medidas cada 5 cm, para un total de 7 medidas por eje, y un total final de 21 mesurados; de cada trayectoria se registró la distancia absoluta entre el punto inicial (X1,Y1,Z1) y el punto final (X2,Y2,Z2) tanto en la realidad virtual (d_{exo}) como en la realidad física (d_{plano}), comparando dichas trayectorias se tomó la distancia de la realidad física como la distancia del plano cartesiano, el cual fue medido con un flexómetro. El error en la trayectoria puede ser observado en la tabla de abajo y en las gráficas presentadas:

$$Error = d_{plano} - d_{exo}$$

$$\%Error\ Absoluto = \frac{\left[d_{plano} - d_{exo}\right]}{d_{teorica}} * 100$$

$$d_{exo} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

	x_1	y_1	z_1	x_2	y_2	z_2	d_{exo}	d_{plano}	*% Error Absoluto*	*Error*
1	-0,3510	0,4107	0,0945	-0,3515	0,4115	0,0914	0,0032	0,0000	0,00	0,0032
2	-0,3515	0,4115	0,0914	-0,3403	0,4084	0,1368	0,0469	0,0500	6,27	0,0031
3	-0,3515	0,4115	0,0914	-0,3309	0,4002	0,2040	0,1150	0,1000	15,03	0,0150
4	-0,3515	0,4115	0,0914	-0,3257	0,4017	0,2613	0,1721	0,1500	14,75	0,0221
5	-0,3515	0,4115	0,0914	-0,3128	0,3899	0,3073	0,2204	0,2000	10,20	0,0204
6	-0,3515	0,4115	0,0914	-0,2871	0,3837	0,3501	0,2680	0,2500	7,22	0,0180
7	-0,3515	0,4115	0,0914	-0,2657	0,3832	0,3689	0,2918	0,3000	2,72	0,0082
EJE COORDENADO "z" (metros)						Sumatoria	0,2918	0.3000	2,72	0,0082

Tabla 3. Análisis de registro de trayectoria "eje z"

Fuente: Auto

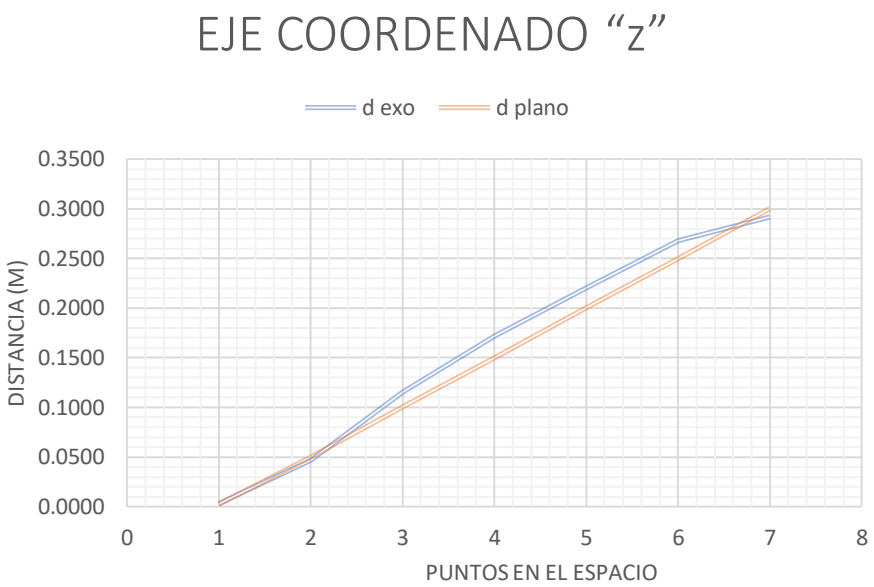


Diagrama 4. Análisis de registro de trayectoria “eje z”

Fuente: Autor

	x_1	y_1	z_1	x_2	y_2	z_2	d_{exo}	d_{plano}	**% Error Absoluto**	**Error**
1	-0,3646	0,3077	-0,0974	-0,3646	0,3077	-0,0954	0,0020	0,0020	0,00	0,0000
2	-0,3646	0,3077	-0,0974	-0,3550	0,2695	-0,1356	0,0549	0,0500	9,72	0,0049
3	-0,3646	0,3077	-0,0974	-0,3426	0,2325	-0,1738	0,1094	0,1000	9,41	0,0094
4	-0,3646	0,3077	-0,0974	-0,3352	0,2130	-0,2120	0,1515	0,1500	1,01	0,0015
5	-0,3646	0,3077	-0,0974	-0,3236	0,1620	-0,2501	0,2150	0,2000	7,52	0,0150
6	-0,3646	0,3077	-0,0974	-0,3130	0,1360	-0,2883	0,2619	0,2500	4,76	0,0119
7	-0,3646	0,3077	-0,0974	-0,2946	0,0840	0,0680	0,2869	0,3000	4,37	0,0131
EJE COORDENADO “y” (metros)						Sumatoria	0,2869	0.3000	4.37	0,0131

Tabla 4. Análisis de registro de trayectoria “eje y”.

Fuente: Autor

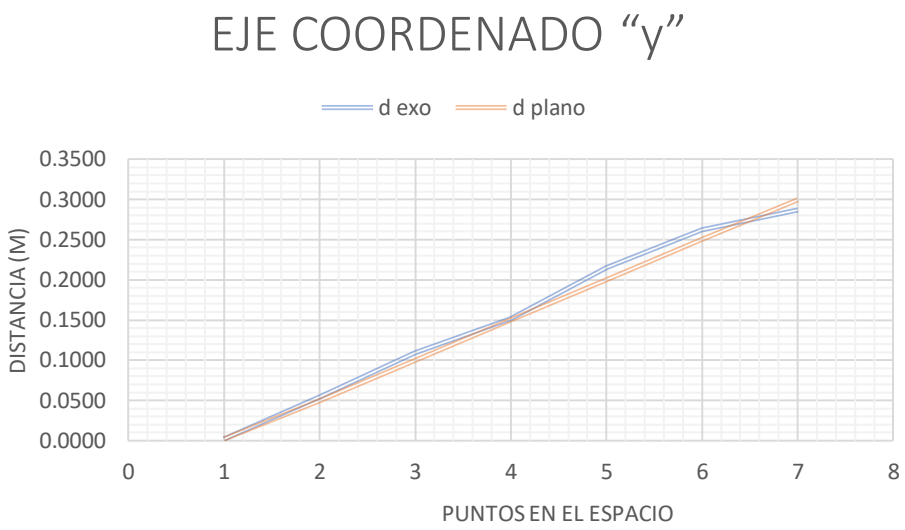


Diagrama 5. Análisis de registro de trayectoria “eje y”.

Fuente: Autor

	x_1	y_1	z_1	x_2	y_2	z_2	d_{exo}	d_{plano}	*% Error Absoluto*	*Error*
1	-0,2726	0,3739	-0,0678	-0,2726	0,3739	-0,0678	0,0000	0,0000	0	0,0000
2	-0,2726	0,3739	-0,0678	-0,2411	0,3302	-0,0728	0,0541	0,0500	8,20	0,0041
3	-0,2726	0,3739	-0,0678	-0,2274	0,2923	-0,0799	0,0941	0,1000	5,94	0,0059
4	-0,2726	0,3739	-0,0678	-0,1964	0,2487	-0,0834	0,1474	0,1500	1,74	0,0026
5	-0,2726	0,3739	-0,0678	-0,1781	0,2270	-0,0747	0,1748	0,2000	12,60	0,0252
6	-0,2726	0,3739	-0,0678	-0,1475	0,1850	-0,0690	0,2266	0,2500	9,37	0,0234
7	-0,2726	0,3739	-0,0678	-0,1067	0,1403	-0,0492	0,2871	0,3000	4,29	0,0129
EJE COORDENADO “x” (metros)						Sumatoria	0,2871	0,3000	4,29	0,0129

Tabla 5. Análisis de registro de trayectoria “eje x”.

Fuente: Autor

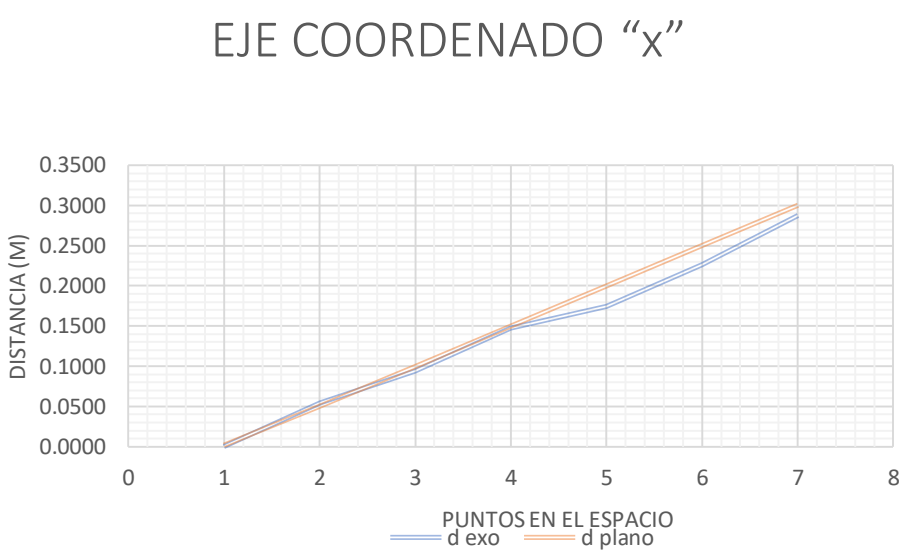


Diagrama 6. Análisis de registro de trayectoria “eje x”.

Fuente: Autor

Se extrae de los resultados mencionados anteriormente que el error máximo entre el inicio y final del eje al completar los 30 cm, es de 4.37%, que corresponde a un error de 0.0131 m.

Antes de llegar a este porcentaje; el error obtenido oscilaba entre un 14.2% y 17.9% lo que representaba una variación de más de 9 cm (0.09m). Al analizar este error se observó que provenía de la acumulación de errores, lo cual como se evidencio se generaba por demasiada vibración en el movimiento, llevando a un resultado de una desviación alta. Se analizó si el ruido provenía de las lecturas analógicas, aunque el voltaje tiene una variación de 0,001 V no fue significante ya que cada grado tiene una resolución de cambio de 0,0048 V. Se decidió quitar el regulador de voltaje al utilizar una fuente externa de alimentación para descartar la influencia del ruido de alterna (60Hz), y alimentar directamente la tarjeta con la fuente del computador a 5 VDC, Como se utilizó resistencias del orden de 300 KΩ, el consumo de corriente no fue muy alta.

Aparte del cambio de fuente de alimentación se realizó un cambio en el código, y este permitió comprender que la transmisión por puerto serial de varios datos para diferentes procesos conlleva a la necesidad de cifrarlos, para que el receptor entienda de donde proviene dicho dato. Este tipo de cifrado se afectó al no poder usar todos los decimales posibles en una operación matemática. De lo cual se deduce la existencia de un error acumulado. Es por ello que la mejor opción, es enviar los datos enteros por puerto serial y que el programa receptor efectúe los cálculos matemáticos faltantes, ya que realiza mejor procesamiento que el microcontrolador Arduino Mega 2560 y también está dotado de un mejor

almacenamiento de datos, permitiendo una mayor fluidez en el comportamiento del robot.

La opción escogida para reducir el ruido en la señal y de este modo mejorar el registro en la trayectoria del exoesqueleto disminuyendo el error fue realizar un cambio en el código, delegando las órdenes del código de Arduino al software V-REP, del siguiente modo:

- El Arduino únicamente debe entregar la palabra digital promediada a 10 lecturas que corresponde a dicho grado, y enviar el dato cifrado por puerto Serial, al software V-REP. Al realizar dicha promediación logra reducir parte del ruido.

- Por otro lado, el software V-REP, debe tomar la palabra digital, con un aumento en la lectura serial de 32 bits como mínimo tamaño de palabra, organizarla para saber a qué articulación corresponde, convertirla a radianes y enviarla al exoesqueleto para su posterior procesamiento.

Con el procedimiento anterior se logró mejorar el error presentado, obteniendo una disminución de este error al 4,72 % como se observa en la tabla 4, se obtuvo con esto una tolerancia de error de 0,6 cm a 1 cm (0.006 - 0.01 m), con una incertidumbre en la medida de 0.11cm (0.0011m) causante de una resolución de 0,001m entregado por el software; Debido a que se obtuvo un máximo error de 1,95% en la caracterización del sensor flex y 1,27% en los potenciómetros la acumulación de estos errores con lleva a presentar un error máximo en la trayectoria del 4,27%,
Con la programación efectuada se entrega como resultado que la visualización y procesamiento de los grados en el exoesqueleto resultaron fluidos, y con mayor precisión. En el siguiente diagrama se muestra una comparación de la disminución del error en porcentajes (%) antes y después del ajuste:

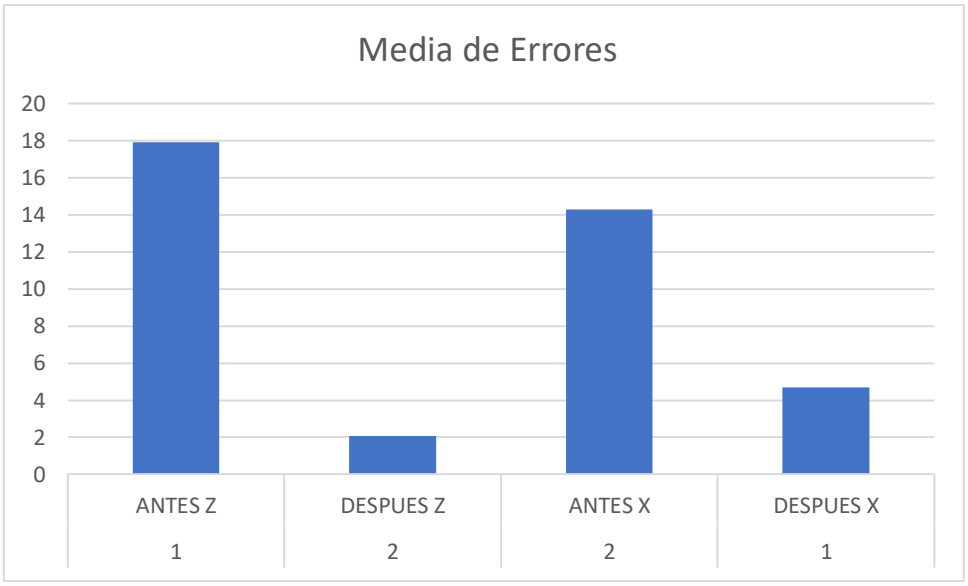
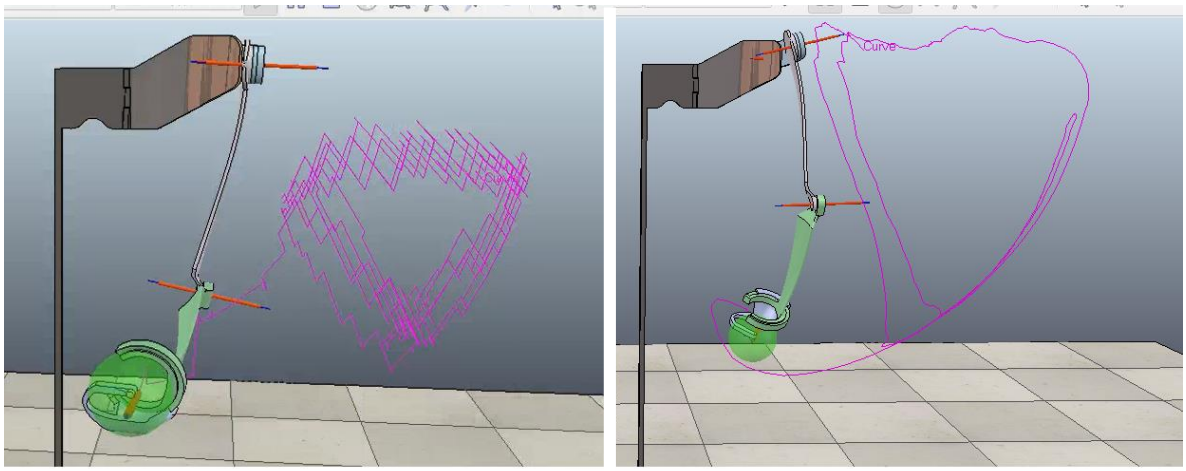


Diagrama 7. Comparación de la media de errores antes y después

Fuente: Autor

En la siguiente figura se muestra una comparación entre un antes y un después del error y la mejora que este obtuvo:

Figura 22. Comparación antes y después del error



Fuente: Autor

8. CONCLUSIONES

- Se diseñó y construyó el exoesqueleto en impresora 3D con el material polímero poli láctico (PLA), el cual fue fácil de adquirir por tener una amplia oferta en el mercado. Este material al no ser tóxico permitió el desarrollo de la construcción en ambientes no controlados. Así se concluye que este proyecto sirve de ejemplo para nuevos proyectos que tengan entre sus finalidades cuidar el medio ambiente y a su vez usar tecnología de punta con costos racionales.

- La construcción del exoesqueleto con 4 grados de libertad permitió un modelo de 1,5 Kg con un volumen aproximado de 135 cm^2 siendo ergonómico, sencillo, compacto y ligero por el material empleado; además de permitir fluidez de movimiento. Se ubicaron sensores para detectar la rotación de los 2 ejes del hombro, uno del codo, y por último la rotación de la muñeca.

- Mediante la simulación de las cargas dentro de un análisis estático, se logró identificar los máximos esfuerzos para realizar el diseño. Se priorizo el análisis en las articulaciones y se reforzaron los ejes de rotación con rodamientos de bolas estriados; mediante esto se logró un diseño resistente y de fácil rotación, obteniendo un factor de seguridad igual a 6, concluyendo que el material en este punto es seguro.

- la caracterización de los sensores (potenciómetros) utilizados proporcionaron una linealidad con un error máximo del 1,27% y para el sensor Flex del 1,95%, dato que influye en el error de la trayectoria entre la realidad física y virtual que fue de 4,72%. Al tener varias acumulaciones de error se concluye la importancia de la calidad de los sensores para obtener una mejor aproximación entre la trayectoria real y virtual.

- En el proceso de adquisición y acondicionamiento de señales, se programó el microcontrolador para promediar 10 lecturas adquiridas y posteriormente enviar la palabra binaria que representa la señal del sensor. Esto permitió obtener un trazado fluido de la trayectoria visualizada dentro del software V-REP, obteniendo un error mínimo de 2,72% y un error máximo de 4,37% sobre una trayectoria trazada en la realidad física; datos que son influidos por la incertidumbre en la medición, la resolución y linealidad en la lectura de los sensores.

- El desarrollo de la cinemática directa para el exoesqueleto y la cinemática inversa para el robot serial virtual a operar, se desarrolló mediante los métodos de Pseudoinversa y DLS que el software V-REP resuelve mediante algoritmos incluidos en el mismo. Se logró con ello el control del brazo robot serial, el cual fue programado para alcanzar los puntos coordenados que el exoesqueleto real

le suministro. Se comprobó mediante simulación que la teleoperación del robot serial mediante el exoesqueleto fue exitosa.

9. RECOMENDACIONES

- Se puede continuar el estudio de la creación de una nueva articulación en la muñeca del exoesqueleto, la cual puede ser utilizada para manipular con ayuda de una mano robótica en el robot serial objetos más complejos, sensando con el exoesqueleto todas las articulaciones de los dedos y de este modo se diseñaría un exoesqueleto más robusto y con mayores aplicaciones.

- Incursionar dentro de la realidad virtual y en las potencialidades del software V-REP con aplicaciones relacionadas a la robótica y a los exoesqueletos cuyo objetivo sea aumentar el conocimiento iniciado en este proyecto.

- Trabajar con el diseño actual en el estudio de aplicaciones más avanzadas como lo son exoesqueletos de fuerza incluyendo en el mecanismo actuadores en cada articulación o incluyendo Feedbacks de fuerza que interrelacionan la realidad física con la realidad virtual.

- Partiendo del diseño del exoesqueleto actual es posible implementar un mecanismo ajustable según la antropometría del sujeto.

REFERENCIAS BIBLIOGRAFICAS

1 NATIONAL AERONAUTICS AND SPACE ADMINISTRATION. “Sensor Exoskeleton ArmMaster for Robot Control”. Johnson Space Center.Estados Unidos. NASA, 1990. {En línea}. {9 marzo de 2016} disponible en: (https://sbir.gsfc.nasa.gov/SBIR/successes/ss/035text.html.).

2 “EUROPEAN SPACE AGENCY. SAM (Sensory Arm Master) Exoskeleton”. telerobotics & haptics lab. España. ESA, 2012. {En línea}. {10 julio de 2016} disponible en: (http://esa-telerobotics.net/gallery/Research/Haptic-robots-design-development/SAMExoskeleton/16).

3 FESTO AG & Co. KG. “ExoHand: New areas for action for man and machine. Alemania. FESTO”, 2012. {En línea}. {11 enero de 2017} disponible en: (https://www.festo.com/group/en/cms/10233.htm).

4 UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA. “Comportamiento de accidentalidad de una empresa metalúrgica en Cartagena”. Vol.17. Bogotá, Colombia. NOVA, 2014. {En línea}. {16 agosto de 2017} disponible en: (https://www.dane.gov.co/flies/investigaciones/discapacidad/inform_estad.pdf.).

5 PONS. José. “Wearable robots Biomechatronic Exoskeletons”. Inglaterra: Wikey, 2008. 5p

6 GUTIERREZ. R. “Control de manipuladores teleoperados”. En: Ciencia e ingeniería neogranadina. Vol.; 19. No 1 (Ago. 2006); p. 35.

7 VERTUT. Jean. & COIFFET. Phillip. “Teleoperation and Robotics: Applications and Technology”. Francia: Springer Science & Business Media, 1985. 19p.

8 MERLET. JEAN-PIERE. “Parallel Robots”. Second Edition. Springer Science & Business Media. 2005. 402 p.

9 GOVINDRAJ. Shashank. CHINTAMANI. Keshav, GANCET. Jeremi. LEITER. Pierre, VAN LIERDE. Boris, NEVATIA. Yashodhan, DE CUBBER. Geert, SERRANO. Daniel, PALOMARES. Miguel, BEDKOWSKI. Janusz, AMBRUST Christopher, SANCHEZ. Jose, COEHLO. Antonio, ORBE. Iratxe. “The ICARUS project - Command Control and Intelligence (C2I)”. Safety Security and Rescue Robotics (SSRR). IEEE International Symposium 2013. 1-4p. {En línea}. {10 julio de 2017} disponible en: (http://ieeexplore.ieee.org/document/5509709/).

10 OLLERO, Anibal “Robótica: Manipuladores y robots móviles”. Barcelona, Marcombo, 2001.

11 ABB. "ABB’s YuMi collaborative robot named *2016 Best Industrial Robot*”. {En línea}. {17agosto de 2017} disponible en: (http://www.abb.com/cawp/seitp202/c9aa2ac92a152904c125801200537df0.aspx).

12 RICILLIO, Marcela. “Robótica: Entrá al mundo de la inteligencia artifical”. ¿Qué es la robótica?" 3p. {En línea}. {7 eneroo de 2018} disponible en: (www.mty.itsem.mx/externos/alaic/texto1html).

13 CRAIG. John. “Introduction to Robotics: Mechanics & amp”. Control. Boston: Addison-Wesley Publishing Company, 1986.

14 ANÓNIMO. “Robots industriales”. {En línea}. {22 noviembre de 2017} disponible en:(http://platea.pntic.mec.es/vgonzale/cyr_0708/archivos/_15/Tema_5.4.htm).

15 SOMOLINOS SANCHEZ. José. Avances en robótica y visión por computador. La Mancha: Universidad de Castilla, 2002. 234p.

16 LUNG-WENG. Tsai. "Robot Analysis: The Mechanics of Serial and Parallel Manipulators". Maryland,
John Wiley & Sons, Inc., 1999.

17 VIVAS. Andrés. “Robótica paralela: Aplicaciones industriales, modelado y control”. {En línea}. {22 enero de 2018} disponible en: (http://www.unicauca.edu.co/ai/publicaciones/ISAShow_Vivas.pdf).

18 CRAIG. John. “Robótica”. México: Pearson Educación, 2006. 62p.

19 VILLARROEL GRATEROL, Jesús. Diseño e implementación de un controlador de posición para una plataforma de stewart con acomodación activa. Catalunya, 2011, 21p. Trabajo de investigación (Ingeniería de Telecomunicación). Universidad politécnica de Catalunya. Escuela Técnica Superior de Ingeniería de telecomunicaciones de Barcelona.

20 ANÓNIMO. “Cinemática Directa. Concepto teórico ”. {En línea}. {6 enero de 2018} disponible en: (http://www.aurova.ua.es/robolab/EJS3/RRP_Intro_2.html).

[21 BARRIENTOS. Antonio. PEÑIN. Luis. BALAGUER. Carlos. ARACIL. Rafael. “Fundamentos de robótica”. Bogotá: McGraw Hill, 1997. 97p.

22 BARRIENTOS. Antonio. PEÑIN. Luis. BALAGUER. Carlos. ARACIL. Rafael. “Fundamentos de robótica”. Bogotá: McGraw Hill, 1997. 108p.

23 BONILLA. William. TERAN. Hector. REINOSO. Hector. "Mecánica para ingeniería estática teoría y problemas resueltos". Universidad de las fuerzas armadas. Ecuador. 2016. 16p.
24 AUTODESK KNOWLEDGE NETWORK. “Coeficiente de seguridad”. Autodesk inventor 2015. AUTODESK, 2015.

25 DASSALAUT SYSTEMS. "Criterio de máxima tensión de von Mises". Ayuda de solidworks. SOLIDWORKS, 2014.

26 SEDELUL. Exoesqueleto apoyará la movilidad de personas parapléjicas. (febrero 2015).

27 ANÓNIMO “¿De dónde viene la palabra robot?”. {En línea}. {25 enero 2018} (http://www.robotica.es/de-donde-viene-la-palabra-robot/).

28 VASQUEZ. Santiago “Diseño e implementación de un sistema electrónico para el registro de acceso y envío de información mediante tecnología NFC al personal administrativo y de soporte técnico de la eempresa WISP AIRMAXTELECOM soluciones tecnológicas S.A”. {En línea}. {25 enero 2018} (http://repositorio.utn.edu.ec/bitstream/123456789/7194/2/04%20RED%20102%20ARTICULO.pdf/).

29 RAMBAL LTDA. "Flex Sensor ”. {En línea}. {13 agosto de 2017} disponible en: (http://rambal.com/index.php?controller=attachment&id_attachment=408).

30 LIFT ARCHITECTS. "Robotic motion controller" {En línea}. {13 agosto de 2017} disponible en: (http://www.liftarchitects.com/robotic-motion-controller/)

31 EXPLORABLE. "Investigación experimental”. {En línea}. {2 octubre de 2017} disponible en: (https://explorable.com/es/investigacion-experimental).

32 GARZA. Enrique. "La investigación documental" México. 1988.

Referencia Normas Complementarias

- Norma ICONTEC NTC 1486. Bogotá: Instituto Colombiano de Normas Técnicas y Certificación. Esta establece la normativa para la presentación de proyectos (tablas, cuadros, figuras, complementos, citas y contenido del mismo).

- Norma ICONTEC NTC 1914 - NTC 1594. Bogotá: Instituto Colombiano de Normas Técnicas y Certificación. Es la normatividad referente al dibujo técnico; las cuales contienen especificaciones para la realización de planos en ingeniería.

- Norma ICONTEC NTC 5613. En esta se aborda la normativa para la correcta elaboración de citas bibliográficas contenidas en un proyecto.

- LEY 23 DE 1982. Esta trata de los derechos y condiciones sobre la propiedad intelectual y los derechos de autor en Colombia. En esta se abordan especificaciones referentes a la creación de software e implementaciones científicas y tecnológicas.

ANEXOS

Script de Exoesqueleto

ANEXO A:
Fuente: Autor

```
-- This example script is non-threaded (executed at each simulation pass)
-- The functionality of this script (or parts of it) could be implemented
-- in an extension module (plugin) and be hidden. The extension module could
-- also allow connecting to and controlling the real robot.- Este script de ejemplo no está enhebrado (se ejecuta en cada pase de simulación)
-- La funcionalidad de este script (o partes del mismo) podría implementarse
-- en un módulo de extensión (complemento) y estar oculto. El módulo de extensión podría
-- también permite la conexión y el control del robot real.

local B0=simGetStringSignal("x1")
if B0 then
B0=B0*math.pi/180
 simAddStatusbarMessage("robot1: "..B0)
end
local B2=simGetStringSignal("x2")
if B2 then
B2=-B2*math.pi/180
B1=B2
 simAddStatusbarMessage("robot2: "..B2)

end
local B3=simGetStringSignal("x3")
if B3 then
B3=B3*math.pi/180
B4=B3
```

```
 simAddStatusbarMessage("robot3: "..B3)


end
local B4=simGetStringSignal("x4")
if B4 then
B4=B4*math.pi/180
 simAddStatusbarMessage("robot3: "..B4)
end
if (sim_call_type==sim_childscriptcall_initialization) then
    --Prepare initial values and retrieve handles-Prepare los valores iniciales y recupera los controles::
    Exoesqueleto=simGetObjectHandle('EXO')
    tip=simGetObjectHandle('TIP')
    targetSphere=simGetObjectHandle('IRB140_manipulationSphere')
    targetSphereBase=simGetObjectHandle('manipulationSphereBase')
    armJoints={-1,-1,-1,-1}
    for i=1,4,1 do
        armJoints[i]=simGetObjectHandle('EJE'..i)
    end
    ui=simGetUIHandle('EXO_UI')
    simSetUIButtonLabel(ui,0,simGetObjectName(targetSphere)..' user interface EXOESQUELETO') -- Set the UI INTERFAZ DE USUARIO title (with the name of the current robot)
    ik1=simGetIkGroupHandle('EXO_undamped')
    ik2=simGetIkGroupHandle('EXO_damped')
    ikFailedReportHandle=-1

    initSizeFactor=simGetObjectSizeFactor(Exoesqueleto) -- only needed if we scale the robot up/down
    -- desired joint positions, and desired cartesian positions-90*math.pi/180:
    desiredJ={0,-90*math.pi/180,0,0} -- when in FK mode
```

```
    for i=1,4,1 do
        simSetJointPosition(armJoints[i],desiredJ[i])
    end
    desiredConf={0,0,0,0,0,0} -- when in IK mode
    currentConf={0,0,0,0,0,0} -- when in IK mode
    ikMinPos={-1.5*initSizeFactor,-1*initSizeFactor,-1*initSizeFactor}
    ikRange={2*initSizeFactor,2*initSizeFactor,1.75*initSizeFactor}
    -- We compute the initial position and orientation of the tip RELATIVE to the robot base (because the base is moving)
    initialTipPosRelative=simGetObjectPosition(tip,Exoesqueleto)
    -- Before V-REP V2.5.1, "simGetObjectOrientation" contained a bug when Euler angles were queried not absolutely
    -- So instead of using "initialTipOrientRelative=simGetObjectOrientation(tip,Exoesqueleto)", we make it a little bit more complicated:
    m=simGetObjectMatrix(tip,Exoesqueleto)
    initialTipOrientRelative=simGetEulerAnglesFromMatrix(m)
    movementMode=0 -- 0=FK, 1=IK through dialog, 2=IK through manipulation sphere

    maxJointVelocity=180*math.pi/180
    maxPosVelocity=1.0*initSizeFactor
    maxOrientVelocity=45*math.pi/180
    previousS=initSizeFactor
end
if (sim_call_type==sim_childscriptcall_cleanup) then


end
if (sim_call_type==sim_childscriptcall_actuation) then
    -- s will scale a few values hereafter (has only an effect if the robot is scaled down/up)
    s=simGetObjectSizeFactor(Exoesqueleto)
    if (s~=previousS) then
```

```
        f=s/previousS
        for i=1,6,1 do
            desiredConf[i]=desiredConf[i]*f
            currentConf[i]=currentConf[i]*f
            ikMinPos[i]=ikMinPos[i]*f
            ikRange[i]=ikRange[i]*f
            initialTipPosRelative[i]=initialTipPosRelative[i]*f
        end
        maxPosVelocity=maxPosVelocity*f
        previousS=s
    end
    maxJointVelocity=simGetUISlider(ui,9)*0.001*math.pi
    maxPosVelocity=simGetUISlider(ui,10)*0.001*initSizeFactor
    maxOrientVelocity=simGetUISlider(ui,10)*0.001*math.pi/2


    -- Now we check if we wanna move the robot through the green manipulation sphere:
    pos=simGetObjectPosition(targetSphere,sim_handle_parent)
    -- Before V-REP V2.5.1, "simGetObjectOrientation" contained a bug when Euler angles were queried not absolutely
    -- So instead of using "orient=simGetObjectOrientation(targetSphere,sim_handle_parent)", we make it a little bit more complicated:
    m=simGetObjectMatrix(targetSphere,sim_handle_parent)
    euler=simGetEulerAnglesFromMatrix(m)
    if (math.abs(pos[1])>0.0001)or(math.abs(pos[2])>0.0001)or(math.abs(pos[3])>0.0001)or
        (math.abs(euler[1])>0.0005)or(math.abs(euler[2])>0.0005)or(math.abs(euler[3])>0.0005) then
            movementMode=2 -- IK by sphere manipulation
            m=simGetObjectMatrix(targetSphere,-1)
            simSetObjectMatrix(targetSphereBase,-1,m)
            simSetObjectPosition(targetSphere,sim_handle_parent,{0,0,0})
```

```
        simSetObjectOrientation(targetSphere,sim_handle_parent,{0,0,0})


        m=simGetObjectMatrix(targetSphere,Exoesqueleto)

        euler=simGetEulerAnglesFromMatrix(m)

        desiredConf={m[4]-initialTipPosRelative[1],m[8]-initialTipPosRelative[2],m[12]-
initialTipPosRelative[3],

                euler[1]-initialTipOrientRelative[1],euler[2]-initialTipOrientRelative[2],euler[3]-
initialTipOrientRelative[3]}

        for i=1,6,1 do

           currentConf[i]=desiredConf[i]

        end

    end

    buttonID=simGetUIEventButton(ui)


    if (buttonID>=1001)and(buttonID<=1006) then -- we want to control the arm in FK mode!

        cyclic,interval=simGetJointInterval(armJoints[buttonID-1000])

        desiredJ[buttonID-1000]=interval[1]+simGetUISlider(ui,buttonID)*0.001*interval[2]

        movementMode=0

    end


    if ((buttonID>=2001)and(buttonID<=2003)) then -- we want to control the arm in IK mode!
(position only is handled here)

        desiredConf[buttonID-2000]=ikMinPos[buttonID-2000]+ikRange[buttonID-
2000]*simGetUISlider(ui,buttonID)/1000

        movementMode=1

    end

    if ((buttonID==2004)or(buttonID==2006)) then -- we want to control the arm in IK mode!
(orientation 1&3 only is handled here)

        desiredConf[buttonID-2000]=math.pi*(-1+2*simGetUISlider(ui,buttonID)/1000)

        movementMode=1

    end
```

```
    if (buttonID==2005) then -- we want to control the arm in IK mode! (orientation 2 only is handled here)
        desiredConf[buttonID-2000]=0.5*math.pi*(-1+2*simGetUISlider(ui,buttonID)/1000)
        movementMode=1
    end

    if movementMode==1 then
        -- We are in IK mode
        maxLinVariationAllowed=maxPosVelocity*simGetSimulationTimeStep()
        maxAngVariationAllowed=maxOrientVelocity*simGetSimulationTimeStep()
        deltaX={0,0,0,0,0,0}
        -- position:
        for i=1,3,1 do
            deltaX[i]=desiredConf[i]-currentConf[i]
            if (math.abs(deltaX[i])>maxLinVariationAllowed) then
                deltaX[i]=maxLinVariationAllowed*deltaX[i]/math.abs(deltaX[i]) -- we limit the variation to the maximum allowed
            end
        end
        -- orientation:
        for i=1,3,1 do
            deltaX[3+i]=desiredConf[3+i]-currentConf[3+i]
            -- Normalize delta to be between -pi and +pi (or -pi/2 and +pi/2 for beta):
            if (i==2) then
                rnge=math.pi
            else
                rnge=math.pi*2
            end
            deltaX[3+i]=math.fmod(deltaX[3+i],rnge)
            if (deltaX[3+i]<-rnge*0.5) then
```

```
            deltaX[3+i]=deltaX[3+i]+rnge
         else
            if (deltaX[3+i]>rnge*0.5) then
               deltaX[3+i]=deltaX[3+i]-rnge
            end
         end
         if (math.abs(deltaX[3+i])>maxAngVariationAllowed) then
            deltaX[3+i]=maxAngVariationAllowed*deltaX[3+i]/math.abs(deltaX[3+i]) -- we limit
the variation to the maximum allowed
         end
      end

      for i=1,6,1 do
         currentConf[i]=currentConf[i]+deltaX[i]
      end

      -- Normalize the orientation part to display normalized values:
      for i=1,3,1 do
         f=1
         if i==2 then f=0.5 end
         currentConf[3+i]=math.fmod(currentConf[3+i],math.pi*2*f)
         if (currentConf[3+i]<-math.pi*f) then
            currentConf[3+i]=currentConf[3+i]+math.pi*2*f
         else
            if (currentConf[3+i]>math.pi*f) then
               currentConf[3+i]=currentConf[3+i]-math.pi*2*f
            end
         end
      end
```

```
        pos={0,0,0}
        orient={0,0,0}
        for i=1,3,1 do
            pos[i]=initialTipPosRelative[i]+currentConf[i]
            orient[i]=initialTipOrientRelative[i]+currentConf[3+i]
        end
        -- We set the desired position and orientation
        simSetObjectPosition(targetSphereBase,Exoesqueleto,pos)
        simSetObjectOrientation(targetSphereBase,Exoesqueleto,orient)
    end


    if (movementMode==1) or (movementMode==2) then
        if (simHandleIkGroup(ik1)==sim_ikresult_fail) then
            -- the position/orientation could not be reached.
            simHandleIkGroup(ik2) -- Apply a damped resolution method
            if (ikFailedReportHandle==-1) then -- We display a IK failure report message
                ikFailedReportHandle=simDisplayDialog("IK Reporte de Fallo","IK no se
resolvio.",sim_dlgstyle_message,false,"",nil,{1,0.7,0,0,0,0})
            end
        else
            if (ikFailedReportHandle>=0) then
                simEndDialog(ikFailedReportHandle) -- We close any report message about IK failure
                ikFailedReportHandle=-1
            end
        end
        -- Now update the desiredJ in case we switch back to FK mode:
        for i=1,4,1 do
            desiredJ[i]=simGetJointPosition(armJoints[i])
        end
```

```
    end


    if movementMode==0 then
        -- We are in FK mode

        currentJ={0,0,0,0}
        for i=1,4,1 do
            currentJ[i]=simGetJointPosition(armJoints[i])
        end
        maxVariationAllowed=maxJointVelocity*simGetSimulationTimeStep()
        for i=1,4,1 do
            delta=desiredJ[i]-currentJ[i]
            if (simGetJointInterval(armJoints[i])) then
                -- Joint is cyclic, we go the fastest direction:
                if (delta>math.pi) then
                    delta=delta-math.pi*2
                end
                if (delta<-math.pi) then
                    delta=delta+math.pi*2
                end
            end
            if (math.abs(delta)>maxVariationAllowed) then
                delta=maxVariationAllowed*delta/math.abs(delta) -- we limit the variation to the
maximum allowed
            end
simSetJointPosition(armJoints[1],(B1))
simSetJointPosition(armJoints[2],(B2))
simSetJointPosition(armJoints[3],(B3))
simSetJointPosition(armJoints[4],(B4))
```

```
        end
        -- Now make sure that everything is ok if we switch to IK mode:
        simSetObjectPosition(targetSphereBase,-1,simGetObjectPosition(tip,-1))
        simSetObjectOrientation(targetSphereBase,-1,simGetObjectOrientation(tip,-1))
        tipPosRel=simGetObjectPosition(tip,Exoesqueleto)
        -- Before V-REP V2.5.1, "simGetObjectOrientation" contained a bug when Euler angles were queried not absolutely
        -- So instead of using "tipOrientRel=simGetObjectOrientation(tip,Exoesqueleto)", we make it a little bit more complicated:
        m=simGetObjectMatrix(tip,Exoesqueleto)
        tipOrientRel=simGetEulerAnglesFromMatrix(m)

        desiredConf={tipPosRel[1]-initialTipPosRelative[1],tipPosRel[2]-initialTipPosRelative[2],tipPosRel[3]-initialTipPosRelative[3],
                    tipOrientRel[1]-initialTipOrientRelative[1],tipOrientRel[2]-initialTipOrientRelative[2],tipOrientRel[3]-initialTipOrientRelative[3]}
        for i=1,6,1 do
            currentConf[i]=desiredConf[i]
        end
        -- Close any IK warning dialogs:
        if (ikFailedReportHandle>=0) then
            simEndDialog(ikFailedReportHandle) -- We close any report message about IK failure
            ikFailedReportHandle=-1
        end
    end

    -- Now update the user interface:
    -- First the FK part, text boxes:
    for i=1,4,1 do
        simSetUIButtonLabel(ui,1010+i,string.format("%.4f",simGetJointPosition(armJoints[i])*180/math.pi))
```

```
    end
    -- Then the FK part, sliders, based on the target joint position if in FK mode, or based on the current joint position if in IK mode:
    for i=1,4,1 do
        cyclic,interval=simGetJointInterval(armJoints[i])
        if (movementMode~=0) then
            simSetUISlider(ui,1000+i,1000*(simGetJointPosition(armJoints[i])-interval[1])/interval[2])
        else
            simSetUISlider(ui,1000+i,1000*(desiredJ[i]-interval[1])/interval[2])
        end
    end
    -- Now the IK part:
    -- First the text boxes:
    -- Linear:
    for i=1,3,1 do
        str=string.format("%.4f",currentConf[i])
        if (str=='-0.000000') then
            str='0.00000' -- avoid having the - sign appearing and disappearing when 0
        end
        simSetUIButtonLabel(ui,2010+i,str)
    end
    -- Angular:
    for i=1,3,1 do
        str=string.format("%.1f",currentConf[1+i]*180/math.pi)
        if (str=='-0.0') then
            str='0.0' -- avoid having the - sign appearing and disappearing when 0
        end
        simSetUIButtonLabel(ui,2013+i,str)
simSetUIButtonLabel(ui,1015,str2)
```

```
    end
    -- Now the sliders, based on the desired configuration if in IK mode, or based on the current tip configuration if in FK mode:
    -- Linear:
    for i=1,3,1 do
        if (movementMode~=0) then
            simSetUISlider(ui,2000+i,1000*(desiredConf[i]-ikMinPos[i])/ikRange[i])
        else
            simSetUISlider(ui,2000+i,1000*(currentConf[i]-ikMinPos[i])/ikRange[i])
        end
pp1=desiredConf[1]
simSetStringSignal("EXO1",pp1)

pp2=desiredConf[2]
simSetStringSignal("EXO2",pp2)
pp3=desiredConf[3]
simSetStringSignal("EXO3",pp3)
pp4=desiredConf[2]
simSetStringSignal("EXO4",pp4)
pp5=desiredConf[3]
simSetStringSignal("EXO5",pp5)
pp6=desiredConf[4]
simSetStringSignal("EXO6",pp6)

    end
    -- Angular:
    if (movementMode~=0) then
        simSetUISlider(ui,2004,1000*(desiredConf[4]+math.pi)/(2*math.pi))
        simSetUISlider(ui,2005,1000*(desiredConf[5]+math.pi*0.5)/math.pi)
        simSetUISlider(ui,2006,1000*(desiredConf[6]+math.pi)/(2*math.pi))
```

```
    else
        simSetUISlider(ui,2004,1000*(currentConf[4]+math.pi)/(2*math.pi))



    end
end
```

ANEXO B:

Script de Brazo Robot IRB 140

Fuente: Autor

```
-- This example script is non-threaded (executed at each simulation pass)
-- The functionality of this script (or parts of it) could be implemented
-- in an extension module (plugin) and be hidden. The extension module could
-- also allow connecting to and controlling the real robot.
local w1=simGetStringSignal("EXO1")

    local w2=simGetStringSignal("EXO2")

    local w3=simGetStringSignal("EXO3")

    local w4=simGetStringSignal("EXO4")

    local w5=simGetStringSignal("EXO5")


    local w6=simGetStringSignal("EXO6")
```

```
    for i=1,6,1 do
w={0,0,0,0,0,0}
w[i]=w1,w2,w3,w4,w5,w6


end
if (sim_call_type==sim_childscriptcall_initialization) then
    --Prepare initial values and retrieve handles:
    irb140=simGetObjectHandle('IRB140')
    tip=simGetObjectHandle('IRB140_tip')
    targetSphere=simGetObjectHandle('IRB140_manipulationSphere')
    targetSphereBase=simGetObjectHandle('IRB140_manipulationSphereBase')
    armJoints={-1,-1,-1,-1,-1,-1}
    for i=1,6,1 do
        armJoints[i]=simGetObjectHandle('IRB140_joint'..i)
    end
    ui=simGetUIHandle('IRB140_UI')
    simSetUIButtonLabel(ui,0,simGetObjectName(irb140)..' user interface') -- Set the UI title
(with the name of the current robot)
    ik1=simGetIkGroupHandle('IRB140_undamped')
    ik2=simGetIkGroupHandle('IRB140_damped')
    ikFailedReportHandle=-1

    initSizeFactor=simGetObjectSizeFactor(irb140) -- only needed if we scale the robot up/down

    -- desired joint positions, and desired cartesian positions:
    desiredJ={0,-90*math.pi/180,90*math.pi/180,0,0,0} -- -90*math.pi/180when in FK mode
    for i=1,6,1 do
        simSetJointPosition(armJoints[i],desiredJ[i])
    end
```

```
    desiredConf={0,0,0,0,0,0} -- when in IK mode
    currentConf={0,0,0,0,0,0} -- when in IK mode
    ikMinPos={-1.5*initSizeFactor,-1*initSizeFactor,-1*initSizeFactor}
    ikRange={2*initSizeFactor,2*initSizeFactor,1.75*initSizeFactor}

    -- We compute the initial position and orientation of the tip RELATIVE to the base (because the base is moving)
    initialTipPosRelative=simGetObjectPosition(tip,irb140)
    -- Before V-REP V2.5.1, "simGetObjectOrientation" contained a bug when Euler angles were queried not absolutely
    -- So instead of using "initialTipOrientRelative=simGetObjectOrientation(tip,irb140)", we make it a little bit more complicated:
    m=simGetObjectMatrix(tip,irb140)
    initialTipOrientRelative=simGetEulerAnglesFromMatrix(m)

    movementMode=0 -- 0=FK, 1=IK through dialog, 2=IK through manipulation sphere


    maxJointVelocity=180*math.pi/180
    maxPosVelocity=1.0*initSizeFactor
    maxOrientVelocity=45*math.pi/180
    previousS=initSizeFactor

end
if (sim_call_type==sim_childscriptcall_cleanup) then


end

if (sim_call_type==sim_childscriptcall_actuation) then
    -- s will scale a few values hereafter (has only an effect if the robot is scaled down/up)
    s=simGetObjectSizeFactor(irb140)
    if (s~=previousS) then
```

```
        f=s/previousS
        for i=1,3,1 do
            desiredConf[i]=desiredConf[i]*f
            currentConf[i]=currentConf[i]*f
            ikMinPos[i]=ikMinPos[i]*f
            ikRange[i]=ikRange[i]*f
            initialTipPosRelative[i]=initialTipPosRelative[i]*f
        end
        maxPosVelocity=maxPosVelocity*f
        previousS=s
    end

    maxJointVelocity=simGetUISlider(ui,9)*0.001*math.pi
    maxPosVelocity=simGetUISlider(ui,10)*0.001*initSizeFactor
    maxOrientVelocity=simGetUISlider(ui,10)*0.001*math.pi/2

    -- Now we check if we wanna move the robot through the green manipulation sphere:
    pos=simGetObjectPosition(targetSphere,sim_handle_parent)
    -- Before V-REP V2.5.1, "simGetObjectOrientation" contained a bug when Euler angles were queried not absolutely
    -- So instead of using "orient=simGetObjectOrientation(targetSphere,sim_handle_parent)", we make it a little bit more complicated:
    m=simGetObjectMatrix(targetSphere,sim_handle_parent)
    euler=simGetEulerAnglesFromMatrix(m)
    if (math.abs(pos[1])>0.0001)or(math.abs(pos[2])>0.0001)or(math.abs(pos[3])>0.0001)or
        (math.abs(euler[1])>0.0005)or(math.abs(euler[2])>0.0005)or(math.abs(euler[3])>0.0005) then
            movementMode=2 -- IK by sphere manipulation
            m=simGetObjectMatrix(targetSphere,-1)
            simSetObjectMatrix(targetSphereBase,-1,m)
```

```
        simSetObjectPosition(targetSphere,sim_handle_parent,{0,0,0})
        simSetObjectOrientation(targetSphere,sim_handle_parent,{0,0,0})


        m=simGetObjectMatrix(targetSphere,irb140)
        euler=simGetEulerAnglesFromMatrix(m)
        desiredConf={m[4]-initialTipPosRelative[1],m[8]-initialTipPosRelative[2],m[12]-
initialTipPosRelative[3],
                euler[1]-initialTipOrientRelative[1],euler[2]-initialTipOrientRelative[2],euler[3]-
initialTipOrientRelative[3]}
        for i=1,6,1 do
           currentConf[i]=desiredConf[i]
        end
    end



    buttonID=simGetUIEventButton(ui)


    if (buttonID>=1001)and(buttonID<=1006) then -- we want to control the arm in FK mode!
        cyclic,interval=simGetJointInterval(armJoints[buttonID-1000])
        desiredJ[buttonID-1000]=interval[1]+simGetUISlider(ui,buttonID)*0.001*interval[2]
        movementMode=0
    end


    if ((buttonID>=2001)and(buttonID<=2003)) then -- we want to control the arm in IK mode!
(position only is handled here)
        desiredConf[buttonID-2000]=ikMinPos[buttonID-2000]+ikRange[buttonID-
2000]*simGetUISlider(ui,buttonID)/1000
        movementMode=1
    end
    if ((buttonID==2004)or(buttonID==2006)) then -- we want to control the arm in IK mode!
(orientation 1&3 only is handled here)
```

```
        desiredConf[buttonID-2000]=math.pi*(-1+2*simGetUISlider(ui,buttonID)/1000)
        movementMode=1
    end
    if (buttonID==2005) then -- we want to control the arm in IK mode! (orientation 2 only is handled here)
        desiredConf[buttonID-2000]=0.5*math.pi*(-1+2*simGetUISlider(ui,buttonID)/1000)
        movementMode=1
    end


    if movementMode==1 then
        -- We are in IK mode
        maxLinVariationAllowed=maxPosVelocity*simGetSimulationTimeStep()
        maxAngVariationAllowed=maxOrientVelocity*simGetSimulationTimeStep()
        deltaX={0,0,0,0,0,0}
        -- position:
        for i=1,3,1 do
            deltaX[i]=desiredConf[i]-currentConf[i]
            if (math.abs(deltaX[i])>maxLinVariationAllowed) then
                deltaX[i]=maxLinVariationAllowed*deltaX[i]/math.abs(deltaX[i]) -- we limit the variation to the maximum allowed
            end
        end
        -- orientation:
        for i=1,3,1 do
            deltaX[3+i]=desiredConf[3+i]-currentConf[3+i]
            -- Normalize delta to be between -pi and +pi (or -pi/2 and +pi/2 for beta):
            if (i==2) then
                rnge=math.pi
            else
                rnge=math.pi*2
```

```
        end
        deltaX[3+i]=math.fmod(deltaX[3+i],rnge)
        if (deltaX[3+i]<-rnge*0.5) then
            deltaX[3+i]=deltaX[3+i]+rnge
        else
            if (deltaX[3+i]>rnge*0.5) then
                deltaX[3+i]=deltaX[3+i]-rnge
            end
        end
        if (math.abs(deltaX[3+i])>maxAngVariationAllowed) then
            deltaX[3+i]=maxAngVariationAllowed*deltaX[3+i]/math.abs(deltaX[3+i]) -- we limit
the variation to the maximum allowed
        end
    end



    for i=1,6,1 do
        currentConf[1]=w1
        currentConf[2]=w2
        currentConf[3]=w3
        currentConf[4]=w4
        currentConf[5]=w5
        currentConf[6]=w6

    end


    -- Normalize the orientation part to display normalized values:
    for i=1,3,1 do
        f=1
        if i==2 then f=0.5 end
```

```
        currentConf[3+i]=math.fmod(currentConf[3+i],math.pi*2*f)
        if (currentConf[3+i]<-math.pi*f) then
            currentConf[3+i]=currentConf[3+i]+math.pi*2*f
        else
            if (currentConf[3+i]>math.pi*f) then
                currentConf[3+i]=currentConf[3+i]-math.pi*2*f
            end
        end
    end



    pos={0,0,0}
    orient={0,0,0}
    for i=1,3,1 do
        pos[i]=initialTipPosRelative[i]+currentConf[i]
        orient[i]=initialTipOrientRelative[i]+currentConf[3+i]
    end
    -- We set the desired position and orientation
    simSetObjectPosition(targetSphereBase,irb140,pos)
    simSetObjectOrientation(targetSphereBase,irb140,orient)
end

if (movementMode==1) or (movementMode==2) then
    if (simHandleIkGroup(ik1)==sim_ikresult_fail) then
        -- the position/orientation could not be reached.
        simHandleIkGroup(ik2) -- Apply a damped resolution method
        if (ikFailedReportHandle==-1) then -- We display a IK failure report message
```

```
            ikFailedReportHandle=simDisplayDialog("IK ROBOT IRB140","IK No pudo
resolverla.",sim_dlgstyle_message,false,"",nil,{1,0.7,0,0,0,0})
        end
    else
        if (ikFailedReportHandle>=0) then
            simEndDialog(ikFailedReportHandle) -- We close any report message about IK failure
            ikFailedReportHandle=-1
        end
    end
    -- Now update the desiredJ in case we switch back to FK mode:
    for i=1,6,1 do
        desiredJ[i]=simGetJointPosition(armJoints[i])
    end
end

if movementMode==0 then
    -- We are in FK mode
    currentJ={0,0,0,0,0,0}
    for i=1,6,1 do
        currentJ[i]=simGetJointPosition(armJoints[i])
    end
    maxVariationAllowed=maxJointVelocity*simGetSimulationTimeStep()
    for i=1,6,1 do
        delta=desiredJ[i]-currentJ[i]
        if (simGetJointInterval(armJoints[i])) then
            -- Joint is cyclic, we go the fastest direction:
            if (delta>math.pi) then
                delta=delta-math.pi*2
            end
```

```
            if (delta<-math.pi) then
                delta=delta+math.pi*2
            end
        end
        if (math.abs(delta)>maxVariationAllowed) then
            delta=maxVariationAllowed*delta/math.abs(delta) -- we limit the variation to the maximum allowed
        end
        simSetJointPosition(armJoints[i],currentJ[i]+delta)
    end
    -- Now make sure that everything is ok if we switch to IK mode:
    simSetObjectPosition(targetSphereBase,-1,simGetObjectPosition(tip,-1))
    simSetObjectOrientation(targetSphereBase,-1,simGetObjectOrientation(tip,-1))
    tipPosRel=simGetObjectPosition(tip,irb140)
    -- Before V-REP V2.5.1, "simGetObjectOrientation" contained a bug when Euler angles were queried not absolutely
    -- So instead of using "tipOrientRel=simGetObjectOrientation(tip,irb140)", we make it a little bit more complicated:
    m=simGetObjectMatrix(tip,irb140)
    tipOrientRel=simGetEulerAnglesFromMatrix(m)

    desiredConf={tipPosRel[1]-initialTipPosRelative[1],tipPosRel[2]-initialTipPosRelative[2],tipPosRel[3]-initialTipPosRelative[3],
                 tipOrientRel[1]-initialTipOrientRelative[1],tipOrientRel[2]-initialTipOrientRelative[2],tipOrientRel[3]-initialTipOrientRelative[3]}
    for i=1,6,1 do
        currentConf[i]=desiredConf[i]
    end
    -- Close any IK warning dialogs:
    if (ikFailedReportHandle>=0) then
        simEndDialog(ikFailedReportHandle) -- We close any report message about IK failure
```

```
            ikFailedReportHandle=-1
        end
    end


    -- Now update the user interface:
    -- First the FK part, text boxes:
    for i=1,6,1 do
        simSetUIButtonLabel(ui,1010+i,string.format("%.1f",simGetJointPosition(armJoints[i])*18
0/math.pi))
    end
    -- Then the FK part, sliders, based on the target joint position if in FK mode, or based on the
current joint position if in IK mode:
    for i=1,6,1 do
        cyclic,interval=simGetJointInterval(armJoints[i])
        if (movementMode~=0) then
            simSetUISlider(ui,1000+i,1000*(simGetJointPosition(armJoints[i])-
interval[1])/interval[2])
        else
            simSetUISlider(ui,1000+i,1000*(desiredJ[i]-interval[1])/interval[2])
        end
    end


    -- Now the IK part:
    -- First the text boxes:
    -- Linear:
    for i=1,3,1 do
        str=string.format("%.4f",currentConf[i])
        if (str=='-0.000') then
            str='0.000' -- avoid having the - sign appearing and disappearing when 0
        end
```

```
        simSetUIButtonLabel(ui,2010+i,str)
    end
    -- Angular:
    for i=1,3,1 do
        str=string.format("%.1f",currentConf[3+i]*180/math.pi)
        if (str=='-0.0') then
            str='0.0' -- avoid having the - sign appearing and disappearing when 0
        end
        simSetUIButtonLabel(ui,2013+i,str)
    end


    -- Now the sliders, based on the desired configuration if in IK mode, or based on the current tip configuration if in FK mode:
    -- Linear:
    for i=1,3,1 do
        if (movementMode~=0) then
            simSetUISlider(ui,2000+i,900*(desiredConf[i]-ikMinPos[i])/ikRange[i])
        else
            simSetUISlider(ui,2000+i,1000*(currentConf[i]-ikMinPos[i])/ikRange[i])
        end
    end
    -- Angular:

end
    if (movementMode~=0) then
        simSetUISlider(ui,2004,1000*(desiredConf[4]+math.pi)/(2*math.pi))
        simSetUISlider(ui,2005,1000*(desiredConf[5]+math.pi*0.5)/math.pi)
        simSetUISlider(ui,2006,1000*(desiredConf[6]+math.pi)/(2*math.pi))
    else
        simSetUISlider(ui,2004,1000*(currentConf[4]+math.pi)/(2*math.pi))
```

```
    simSetUISlider(ui,2005,1000*(currentConf[5]+math.pi*0.5)/math.pi)
    simSetUISlider(ui,2006,1000*(currentConf[6]+math.pi)/(2*math.pi))
end
```

ANEXO C:

Script Obtención de Señales – V-REP.

Fuente: Autor

```
local
a1=tonumber(token)
      if a1>5000 and a1<6024 then
pot1=((a1-5000) * 0.2589)  + 4.2
pp1=((pot1+236)*math.pi/180)
simSetStringSignal("x1",pp1)
         end
local
a2=tonumber(token)
if  a2>=1025 and a2<2049 then
pot2=((a2-1025) * 0.2589)  + 4.2
      pp2=((pot2-25)*math.pi/180)
simSetStringSignal("x2",pp2)
end
local
a3=tonumber(token)
if  a3> 2050 and a3<3074 then
 pot3=((a3-2050) * 0.2589)  + 4.2
pp3=((pot3-80)*math.pi/180)
simSetStringSignal("x3",pp3)
   end
local
a4=tonumber(token)
if  a4> 3075 and a4<4999 then

pot4=((a4-3075) * -1.2292)  +423.81
pp4=((pot4-15)*math.pi/180)
simSetStringSignal("x4",pp4)
   end
end
```

ANEXO D:

Script Obtención de Señales – ARDUINO.

Fuente: Autor

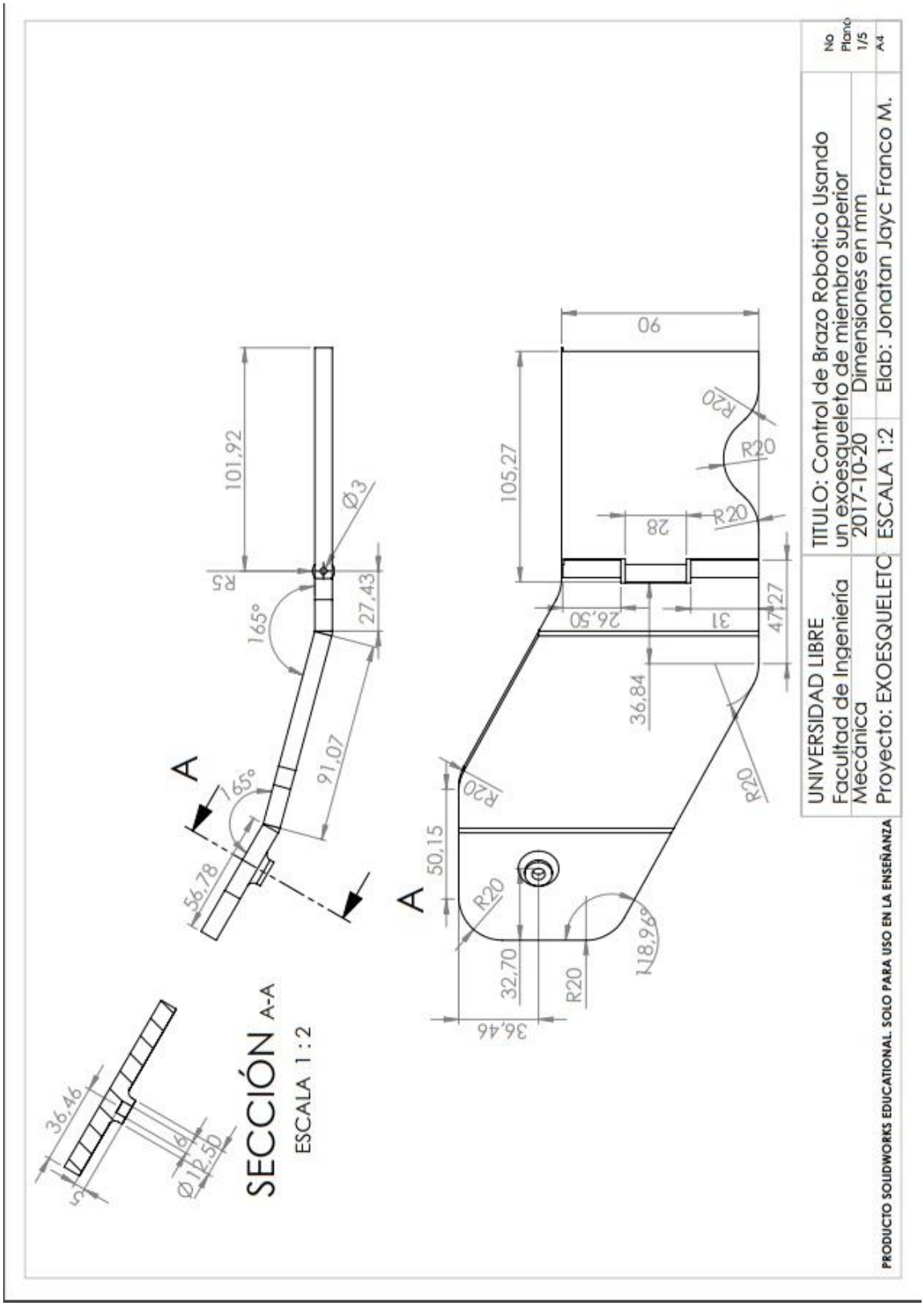
```
int Lecturas1[30]; //Vector de lecturas x30

int Lecturas2[30];

int Lecturas3[30];

int Lecturas4[30];

int POT1, i1 = 0, Total1 = 0, Promedio1 = 0; //Entradas Analogicas, y Valores iniciales para la sumatoria de lecturas.

int POT2,  i2 = 0, Total2 = 0, Promedio2 = 0;

int POT3,  i3 = 0, Total3 = 0, Promedio3 = 0;

int FlexSensor,  i4 = 0, Total4 = 0, Promedio4 = 0;


void setup()

{

  Serial.begin(9600);

 {

for(i1=0; i1< 30; i1++) //Inicialización del vector.

Lecturas1[i1] = 0;

i1=0;

}

 { for(i2=0; i2< 30; i2++) //Inicialización del vector.

Lecturas2[i2] = 0;

i2=0;

}

{  for(i3=0; i3< 30; i3++) //Inicialización del vector.
```

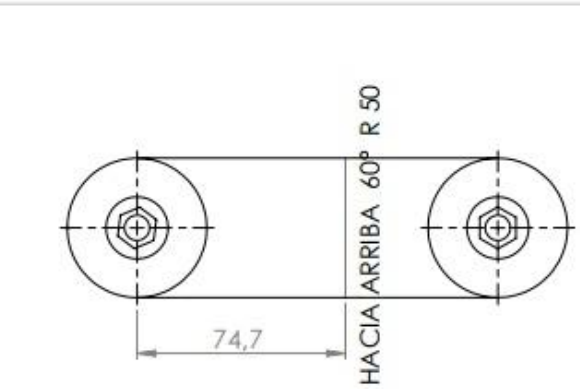
```
Lecturas3[i3] = 0;

i3=0;

}

{  for(i4=0; i4< 30; i4++) //Inicialización del vector.

Lecturas4[i4] = 0;

i4=0;

}

}

void loop()

{

Total1 = Total1 - Lecturas1[i1];

Lecturas1[i1] = analogRead(9);     //voltaje del POT1

//Realiza la sumatoria entre lecturas

Total1 = Total1 + Lecturas1[i1];

i1 = i1 + 1;

//Calcula el promedio

if (i1 >= 30){

i1 = 0;

Promedio1 = Total1 / 30;

POT1 = Promedio1+5000;

// y envía el resultado de la palabra  al V-REP luego de haber sumado 5000

 Serial.println(POT1);

  Serial.print(',');

 }
```

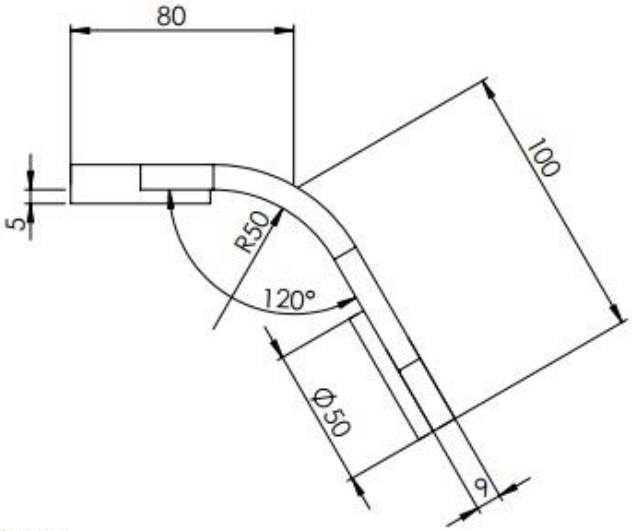
```
  Total2 = Total2 - Lecturas2[i2];

Lecturas2[i2] = analogRead(2);     //voltaje del POT2

//Realiza la sumatoria entre lecturas

Total2 = Total2 + Lecturas2[i2];

i2 = i2 + 1;

//Calcula el promedio

if (i2 >= 30){

i2 = 0;

Promedio2 = Total2 / 30;

POT2 = Promedio2+1025;

// y envía el resultado de la palabra  al V-REP luego de haber sumado 1025

 POT2=Val2+370;

  Serial.println(POT2);

  Serial.print(',');

  }

Total3 = Total3 - Lecturas3[i3];

Lecturas3[i3] = analogRead(4);     //voltaje del POT3

//Realiza la sumatoria entre lecturas

Total3 = Total3 + Lecturas3[i3];

i3 = i3 + 1;

//Calcula el promedio

if (i3 >= 30){

i3 = 0;

Promedio3 = Total3 / 30;

POT3 = Promedio3+2050;
```

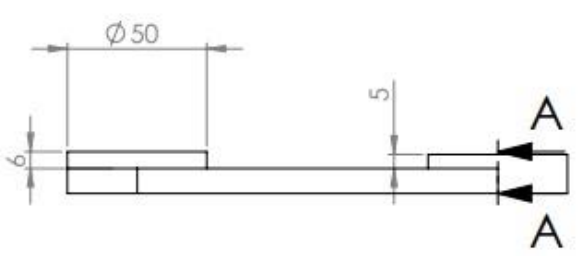
```
//// y envía el resultado de la palabra  al V-REP luego de haber sumado 2050
 POT3=POT3+640;
  Serial.println(POT3);
  Serial.print(',');
  delay(200);
   }

Total4 = Total4 - Lecturas4[i4];
Lecturas4[i4] = analogRead(8);     //voltaje del FlexSensor
//Realiza la sumatoria entre lecturas
Total4 = Total4 + Lecturas4[i4];
i4 = i4 + 1;
//Calcula el promedio
if (i4 >= 30){
i4 = 0;
Promedio4 = Total4 / 30;
FlexSensor4 = Promedio4 +3075;

// y envía el resultado de la palabra  al V-REP luego de haber sumado 3075

 Serial.println(FlexSensor4);
  Serial.print(',');
   }


}
```

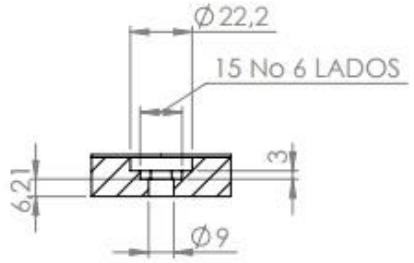
Planos


UNIVERSIDAD LIBRE
Facultad de Ingeniería
Mecánica
Proyecto: EXOESQUELETO
TITULO: Control de Brazo Robotico Usando un exoesqueleto de miembro superior
2017-10-20
Dimensiones en mm
ESCALA 1:2
Elab: Jonatan Jayc Franco M.
No Plano 1/5
A4
PRODUCTO SOLIDWORKS EDUCATIONAL SOLO PARA USO EN LA ENSEÑANZA
SECCIÓN A-A
ESCALA 1 : 2
101,92
Ø3
R5
165°
27,43
91,07
65°
56,78
36,46
6
Ø12,50
105,27
90
R20
28
31
26,50
47,27
36,84
50,15
32,70
36,46
118,96°



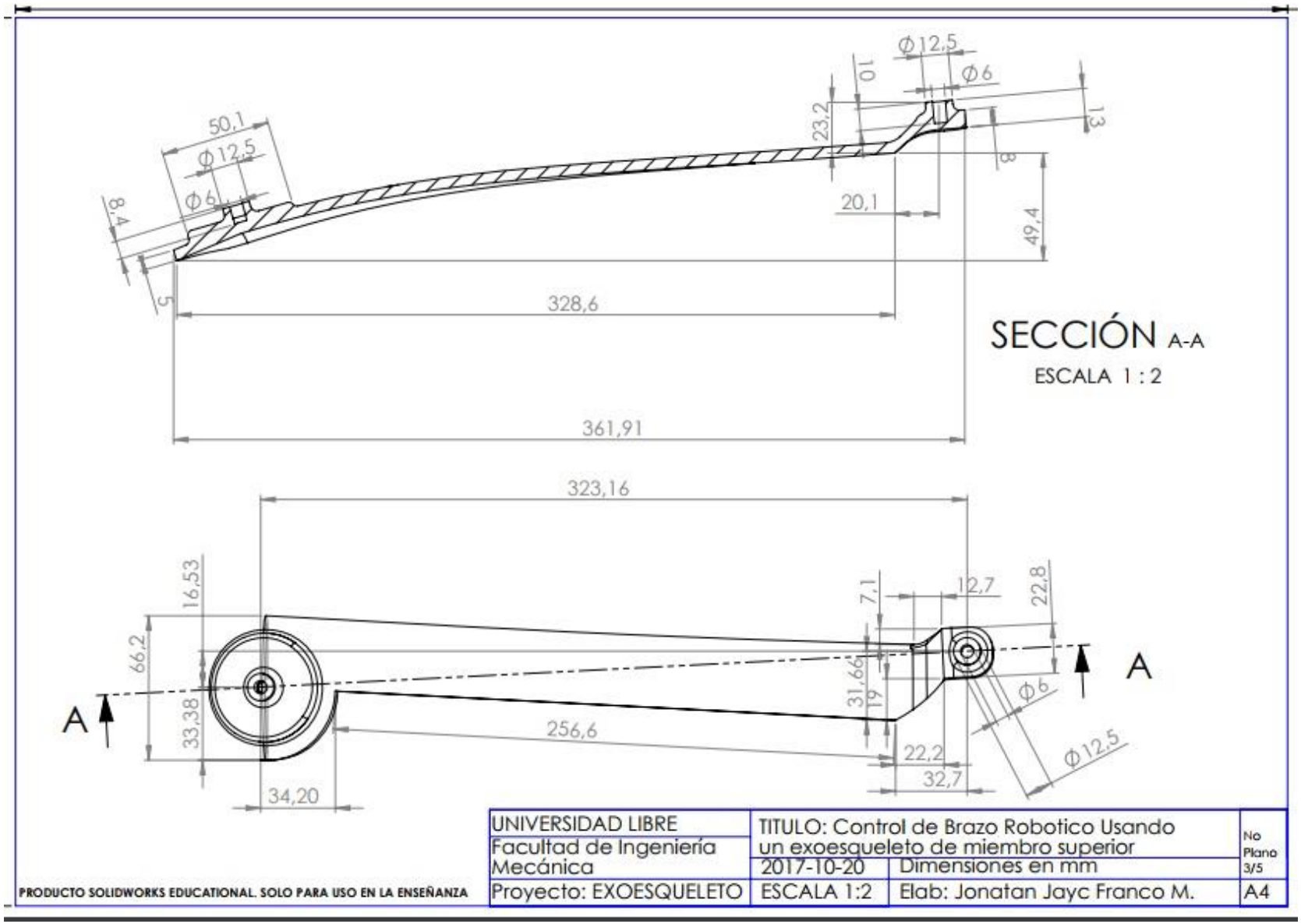


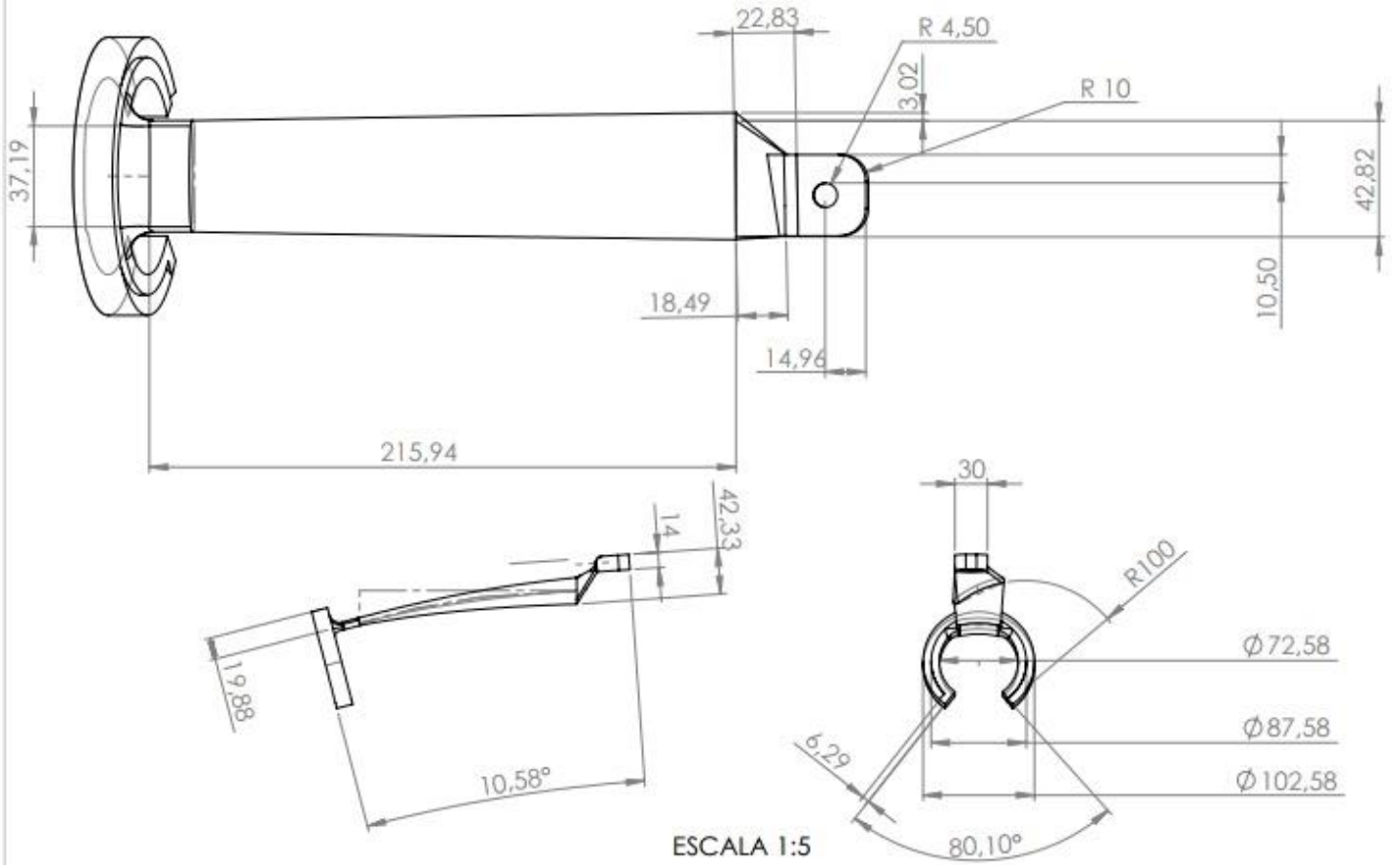




SECCIÓN A-A

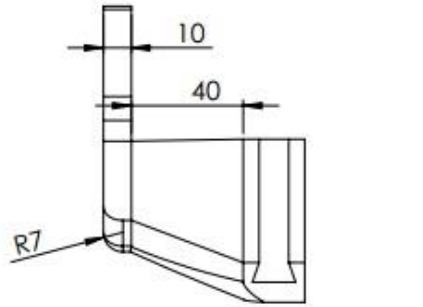
UNIVERSIDAD LIBRE Facultad de Ingeniería Mecánica Proyecto: EXOESQUELETO	TITULO: Control de Brazo Robotico Usando un exoesqueleto de miembro superior		No Plano 2/5 A4
	2017-10-20	Dimensiones en mm	
	ESCALA 1:2	Elab: Jonatan Jayc Franco M.	

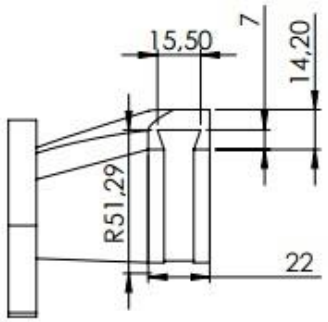

Ø12,5
10
Ø6
23,2
13
50,1
Ø12,5
Ø6
8,4
8
20,1
49,4
5
328,6
361,91
SECCIÓN A-A
ESCALA 1 : 2
323,16
16,53
66,2
33,38
34,20
256,6
7,1
12,7
22,8
31,66
19
22,2
32,7
Ø6
Ø12,5
A
A
PRODUCTO SOLIDWORKS EDUCATIONAL. SOLO PARA USO EN LA ENSEÑANZA
UNIVERSIDAD LIBRE
Facultad de Ingeniería Mecánica
Proyecto: EXOESQUELETO
TITULO: Control de Brazo Robotico Usando un exoesqueleto de miembro superior
2017-10-20
Dimensiones en mm
ESCALA 1:2
Elab: Jonatan Jayc Franco M.
No Plano 3/5
A4

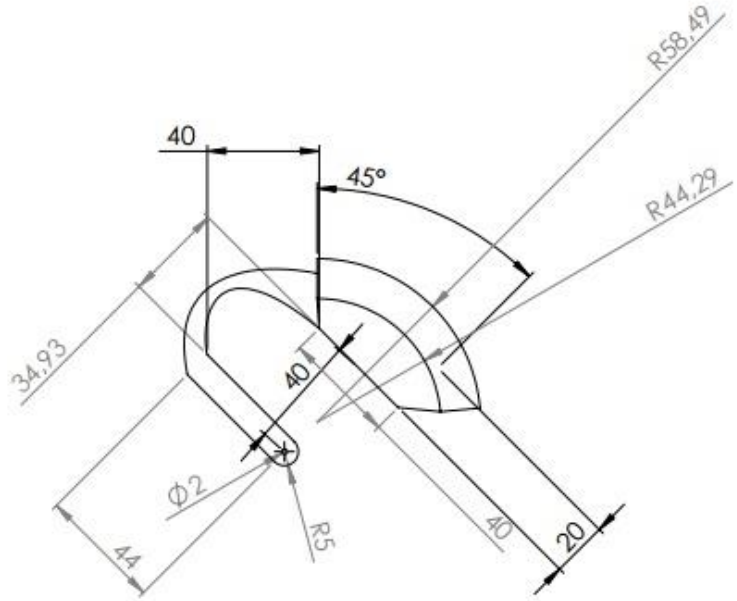


ESCALA 1:5

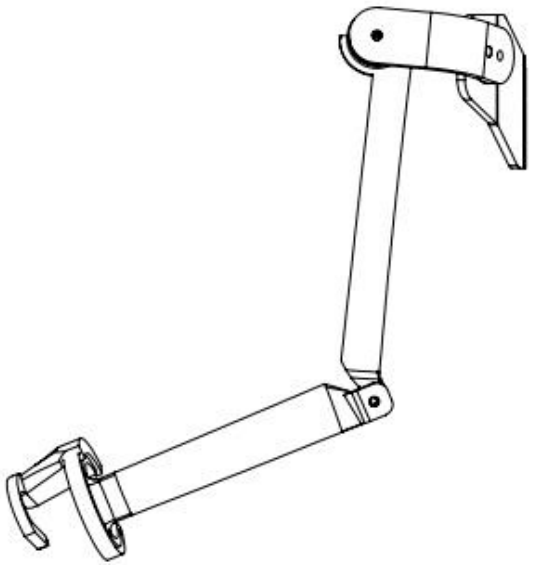
UNIVERSIDAD LIBRE Facultad de Ingeniería Mecánica Proyecto: EXOESQUELETO	TITULO: Control de Brazo Robotico Usando un exoesqueleto de miembro superior		No Plano 4/5 A4
	2017-10-20	Dimensiones en mm	
	ESCALA 1:2	Elab: Jonatan Jayc Franco M.	

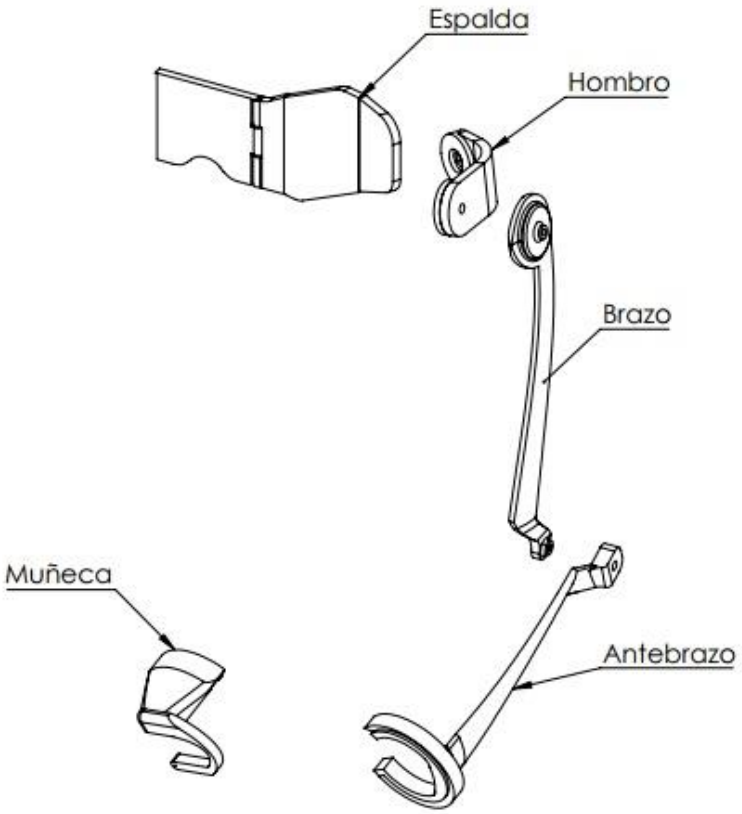






UNIVERSIDAD LIBRE Facultad de Ingeniería Mecánica Proyecto: EXOESQUELETO	TITULO: Control de Brazo Robotico Usando un exoesqueleto de miembro superior		No Plano 5/5 A4
	2017-10-20	Dimensiones en mm	
	ESCALA 1:2	Elab: Jonatan Jayc Franco M.	





UNIVERSIDAD LIBRE Facultad de Ingeniería Mecánica Proyecto: EXOESQUELETO	TITULO: Control de Brazo Robotico Usando un exoesqueleto de miembro superior		Exp
	2017-10-20	Dimensiones en mm	
	ESCALA 1:2	Elab: Jonatan Jayc Franco M.	