CALIBRACIÓN Y APRENDIZAJE DE PARÁMETROS DE CONTROL PARA UN ROBOT DE ASISTENCIA DE MOVILIDAD DE PIE – QOLO – BASADO EN DIFERENCIAS FISIOLÓGICAS APLICADO A PERSONAS CON LESION DE MÉDULA ESPINAL

ALEJANDRA IBAÑEZ BATECA

UNIVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE INGENIERÍAS MECÁNICA, MECATRÓNICA E INDUSTRIAL
INGENIERÍA MECATRÓNICA
PAMPLONA, COLOMBIA
JUNIO 2020

CALIBRACIÓN Y APRENDIZAJE DE PARÁMETROS DE CONTROL PARA UN ROBOT DE ASISTENCIA DE MOVILIDAD DE PIE – QOLO – BASADO EN DIFERENCIAS FISIOLÓGICAS APLICADO A PERSONAS CON LESION DE MÉDULA ESPINAL

ALEJANDRA IBAÑEZ BATECA

TRABAJO DE GRADO PRESENTADO COMO REQUISITO PARA OPTAR POR EL TITULO DE INGENIERO EN MECATRÓNICA

DIRECTOR:
PhD. DIEGO FELIPE PÁEZ GRANADOS
ESCUELA POLITECNICA FEDERAL DE LAUSANNE – EPFL

CO-DIRECTOR:
PhD. CÉSAR PEÑA CORTÉS
UNIVERSIDAD DE PAMPLONA

UNIVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE INGENIERÍAS MECÁNICA, MECATRÓNICA E INDUSTRIAL
PROGRAMA DE INGENIERÍA MECATRÓNICA
PAMPLONA, COLOMBIA
JUNIO 2020

A Dios por habitar mi corazón y fortalecerlo.

A mis padres, mis maestros y guías, por su amor y ejemplo.

A mi abuelito Arnulfo, mi mejor amigo, por su sabiduría y voluntad.

A mi familia, por apoyarme, por poner su confianza en mí.

A todos lo que me compartieron su conocimiento.

A todos los que creen en mí y me han motivado.

A mí.

Resumen

Actualmente en la investigación, los dispositivos que son usados para auxiliar en la movilidad a las personas que tienen alguna discapacidad, buscan adaptarse al usuario. Para ello no solo se debe conocer y tener en cuenta el tipo de discapacidad, sino también, el nivel de lesión, la motricidad residual post-lesión y sus características fisiológicas. El dispositivo de movilidad de pie llamado Qolo, es dirigido por el movimiento del torso del usuario y está destinado para personas con lesión de médula espinal, se encuentra en la Escuela politécnica federal de Lausanne en Suiza. La investigación se enfoca en este dispositivo, el cual debe personalizarse para cada usuario, entendiendo que los usuarios tienen diferencias fisiológicas en cuanto a la altura, el peso y el índice de masa corporal, lo cual implica desarrollar una estrategia de calibración de los parámetros de control e inferir la relación existente o no entre dichos parámetros y las diferencias fisiológicas en los usuarios. En el presente documento se expone la interfaz interactiva desarrollada para la captura de datos, la estrategia de calibración y su evaluación, obteniendo como resultado la personalización del dispositivo implementando control y análisis estadístico para el ajuste y modificación de los parámetros de calibración, con el objetivo último de identificar que otros factores pueden influenciar en el dispositivo Qolo para la personalización, y generar conclusiones útiles para posteriores investigaciones.

Palabras claves: Calibración, parámetros de control, diferencias fisiológicas, Qolo, robot de asistencia.

Abstract

Currently in research, the devices that are used to assist in the mobility of people with disabilities, seek to adapt to the user. To do this, not only must the type of disability be known and taken into account, but also the level of injury, the post-injury residual motor and its physiological characteristics. The foot mobility device called Qolo, which is directed by the movement of the user's torso and is intended for people with spinal cord injury, is located at the Lausanne Federal Polytechnic School, Switzerland. Research focuses on this device, which must be customized for each user, understanding that users have physiological differences in height, weight and body mass index, which involves developing a strategy to calibrate the control parameters and infer the relationship or not between these parameters and the physiological differences in users. This document presents the interactive interface developed for the data capture, the calibration strategy and its evaluation, obtaining as a result the personalization of the device implementing control and statistical analysis for the adjustment and modification of the calibration parameters, with the ultimate goal of identifying what other factors can influence the Qolo device for personalization, and generate useful conclusions for further research.

Keywords: Calibration, control parameters, physiological differences, Qolo, assistance robot.

Agradecimientos

Mis más sinceros agradecimientos:

A Dios por ser esa gran fuente de amor y sabiduría en mi corazón, que cada día me sorprende y me conecta con mi esencia y con el sendero de mi propósito.

A mi padre John Robinsson, por cumplir su papel de padre de la mejor manera, por sus bellas herencias, por su amor y cariño, por volverme una soñadora que materializa sus sueños, por ser mi parte más fuerte.

A mi madre Ludy Rocio, por caminar conmigo de la mano, por ser mi parte más humana y esencial, por su ejemplo de entrega y voluntad, por su invaluable corazón entregado al servicio.

A mi abuelito Arnulfo Bateca, por permitirme acompañarlo y aprender de su vida y obra, por ese gran dador de vida, por ser la columna vertebral tenaz y sabia en la familia Bateca.

A Osmany Alvarado, Helena Bateca y Laura Capacho, por ser mis maestras en esta vida, por su apoyo y confianza.

Al profesor César A. Peña, por fomentar en sus estudiantes la búsqueda de grandes oportunidades, por la confianza y motivación.

Al Dr. Diego F. Paez, por la gran oportunidad que me brindó, por ser ejemplo de superación y disciplina, por guiarme con paciencia en el desarrollo de esta investigación.

A mis profesores del programa, por brindar sus conocimientos a lo largo de mi formación en esta bella carrera.

A mis compañeros y amigos, por acompañarme en este proceso universitario y compartirme sus conocimientos.

A todos y cada uno que han aportado un granito de arena en mi paso por la Universidad de Pamplona.

Índice general

Lista de Tablas

Lista de Figuras

Lista de Abreviaturas

LME	Lesión de médula espinal
POMDP	Proceso de decisión de Markov continuo y parcialmente observable
MDPs	Procesos de decisión de Markov
EKF	Filtro de Kalman extendido
BAMDP	Proceso de decisión de Markov adaptado a Bayes
MCTS	Búsqueda en el árbol de Monte Carlo
MCP	Control de modelos predictivos
RL	Aprendizaje de esfuerzo bayesiano
SNC	Sistema nervioso central
SNP	Sistema nervioso periférico
COP	Centro de presión
T-bar	Barra de resistencias sensitivas a la fuerza
ROS	Sistema Operativo del Robot
BSD	Distribución de software de Berkeley
XML-RPC	Protocolo de transferencia de hipertexto
HTTP	Resultados de búsqueda
LISP	Procesamiento de listas
RWT	Herramientas web de robots
HTML5	Lenguaje de marcado de hipertexto, versión 5
HTML	Lenguaje de marcado de hipertexto
JS	JavaScript
JSON	Notación de objetos en JavaScript
TCP	Protocolo de control de transmisión
IDE	Entorno de desarrollo integrado
ID	Identificación
ATE	Altura entre la T-bar y el Esternón
IMC	Índice de Masa Corporal

CAPÍTULO I

Introducción e información general

Este capítulo aborda la propuesta de trabajo de grado desarrollada, su justificación, y los objetivo propuestos.

1.1 Introducción

Las plataformas o vehículos capaces de ayudar en la movilidad a personas que cuentan con alguna limitación motriz de las extremidades inferiores son diversas. Muchos de ellos se han enfocado en generar una solución a la locomoción de pie de dichas personas, y así mejorando también en los ámbitos emocional, corporal y social del usuario (C. Burbano-López, 2017). Quienes padecen de lesión de médula espinal (LME), en la mayoría de los casos pierden la motricidad de la parte inferior del cuerpo, y por consecuencia no pueden levantarse o sentarse sin ayuda externa, pero esto depende de su nivel de lesión (Ashraf S. Gorgey, 2019).

Actualmente se están realizando investigaciones alrededor de un dispositivo de movilidad de pie – Qolo – donde la persona con dicha discapacidad puede sentarse y pararse voluntariamente, y viajar de pie sin la necesidad de usar sus manos (Paez, 2018). Para el control de dicho dispositivo se ha desarrollado una barra de seguridad sensorial sensible a la fuerza ubicada en la zona abdominal. Con dicha barra se controla la dirección y velocidad de movimiento del dispositivo a través del movimiento del torso del usuario, y así darle manejo al dispositivo con la motricidad residual en la parte superior del cuerpo (Yang Chen, 2019)

Considerando la variabilidad en la contextura del cuerpo humano y su influencia en el robot, pueden existir escenarios en el que el comportamiento del controlador existente pierda estabilidad, no sea óptimo o no realice el movimiento deseado por el usuario según esta variabilidad. Unos de los factores externos al dispositivo que es generado por los usuarios o personas que lo manejan son sus diferencias fisiológicas en cuanto al tamaño, el peso y la masa corporal. Estas diferencias pueden ocasionar o no que los movimientos y direcciones del dispositivo sean diferentes según el usuario. ¿De qué manera se puede calibrar el dispositivo para personalizarlo? Teniendo una base de datos apta para analizarla se puede determinar cuáles son los factores externos que generan una inferencia de movimiento. Así que el objetivo central es aplicar un modelo de control matemático para poder calibrar los parámetros haciéndolo personalizado a los usuarios con diferencias fisiológicas que padecen lesión de médula espinal. Con esta calibración se podría ampliar el rango de aplicación del dispositivo y esta información puede servir de apoyo para otras investigaciones.

Para llevar a cabo el estudio de investigación, el trabajo se ha estructurado en 5 capítulos. En el capítulo I "Introducción e información general" aborda la propuesta de investigación desarrollada, su justificación, los objetivos propuestos, y por último una clara explicación de la estructura general del documento. En el capítulo II "Fundamentos Teóricos" se hace un recorrido del estudio escrito anterior a esta investigación, analizando resultados obtenidos y también se encontrará un conjunto de datos teóricos necesarios para fundamentar la investigación realizada. En el capítulo III "Metodología" se sigue el procedimiento necesario para lograr los objetivos planteados. En el capítulo IV "Resultados" están todos los hallazgos, datos, tablas, graficas e información obtenidas para análisis o estudio. Finalmente, en el capítulo V "Conclusiones" se encuentran los comentarios e interpretaciones finales de la investigación, como también recomendaciones.

1.2 Justificación

Deben ser detalladas y cuidadosas las investigaciones detrás de los dispositivos destinados a personas que tienen alguna discapacidad motriz como resultado de lesiones en la médula espinal, pues estos dispositivos tienen una alta responsabilidad y complejidad, y uno de los objetivos de ellos no es solo auxiliar en la movilidad al usuario o ayudarle en su rehabilitación para reducir la discapacidad, sino también hacer que su interacción con el dispositivo genere confianza y sea intuitivo.

Si durante el manejo del robot, éste hace un movimiento que no está acorde con el deseo del usuario, se abre una brecha de desconfianza, y es allí donde se deben hacer mejoras para la calibración del dispositivo, logrando que éste se adapte cada vez más al usuario, e identificar cuáles son los factores externos que influencian a que el robot no siga las órdenes dadas. Por lo tanto, es pertinente esta investigación, que estudia una posible relación que existe entre los factores que hacen que el sistema responda diferente y los parámetros propios de control, y es allí donde se tiene en cuenta los diferentes tipos de lesiones y las características físicas de los usuarios en diferentes rutinas establecidas para el manejo de un robot móvil.

Los seres humanos tienen a nivel corporal diversas características físicas en cuanto a la altura, peso, contextura y demás, estas diferencias se deben tener en

cuenta para lograr que el dispositivo tenga un alto rango de usabilidad, adaptabilidad y fiabilidad. La personalización del dispositivo al usuario es una buena alternativa, que puede llevar al investigador también a otro nivel de conocimiento de los usuarios, entendiendo desde otro punto de vista la lesión, la motricidad residual para el manejo del dispositivo, su funcionalidad y el rango de movimiento voluntario e involuntario del mismo, e incluso puede abrir la puerta a investigaciones posteriores.

Qolo, el dispositivo de movilidad de pie presente en esta investigación, da asistencia a las personas con lesión de médula espinal en el momento de sentarse y pararse voluntariamente, y viajar de pie sin usar sus manos. Existen varios prototipos de Qolo a nivel internacional en los que se está investigando actualmente, uno de ellos se encuentra en la Escuela Politécnica Federal de Lausanne en Suiza, bajo la supervisión del Dr. Diego Felipe Paez. Una colaboración de investigación con dicha escuela fue una gran oportunidad aprovechada durante el tiempo de mi visita.

1.3 Objetivos

1.3.1 Objetivo General

Obtener un modelo de control para la calibración y optimización de parámetros que permitan personalizar la inferencia de movimiento deseado para diferentes usuarios con lesión de médula espinal en el dispositivo de movilidad de pie – Qolo.

1.3.2 Objetivos Específicos

- Diseñar una interfaz interactiva para la recolección de datos de movimiento de los usuarios con respecto al movimiento del mismo generado en el dispositivo, quienes tienen contexturas, tamaño y pesos diferentes.
- Aplicar un modelo de control en los datos obtenidos para la definición de parámetros de diferencia significativos de las personas, los cuales permitan realizar una personalización de la interfaz de control.
- Evaluar el modelo de control para la calibración del sistema.

CAPÍTULO II

Fundamentos teóricos

Se hace un recorrido del estudio escrito anterior a esta investigación, y también se encontrará un conjunto de datos teóricos necesarios para fundamentar la investigación.

2.1 Estado del Arte

Los robots que trabajan bajo la incertidumbre de la dinámica externa, deben realizar tareas de estimación, control y calibración en el dispositivo para que no pierda la robustez ante constantes cambios. Así, un modelo de estimación de parámetros desconocidos en un robot de asistencia o rehabilitación es una herramienta a favor a la solución de problemas relacionados con el control, la planificación, la localización o el mapeo de movimiento, que se basa en el comportamiento observado del robot ante el del humano.

Controlar dispositivos o máquinas destinados a la asistencia y rehabilitación es un completo desafío, tanto por las limitaciones de la interfaz de control usada como por la discapacidad física presente en el humano. Por lo tanto, en algunas investigaciones se ha planteado disminuir la autonomía del dispositivo, con el objetivo de que sea más adaptable y el humano sea menos vulnerable a las arbitrariedades presentes en el sistema. Para optimizar la interacción humano-robot en estos dispositivos durante la tarea a realizar, la personalización es una solución, con el ajuste o estimación de parámetros de control del sistema.

El problema de estimación de parámetros ha sido principalmente de interés teórico debido a su complejidad (C. Papadimitriou, 2017), pero ya varios métodos se han puesto a prueba últimamente para determinar su eficiencia en tiempo real (online). El proceso de decisión de Markov continuo y parcialmente observable (POMDP), es un método que permite la estimación de parámetros en tiempo real como esta propuesto en (D. J. Webb, 2014) y está basado en los procesos de decisión de Markov (MDPs), este plantea políticas de control óptimo para el aprendizaje de los parámetros con la mayor cantidad de información posible, mientras no interviene en el logro de los objetivos base del robot. Por la alta complejidad computacional se busca una aproximación de este método en los que las creencias (las distribuciones de la estimación del estado del robot) pueden ser modeladas usando distribuciones gaussianas como en (J. Van Den Berg, 2012) (J. van den Berg, 2017) y actualizando las creencias con un filtro de Kalman extendido (EKF).

Otra aproximación propuesta del anterior método es enmarcar el problema como un proceso de decisión de Markov adaptado a Bayes (BAMDPs), que es resuelto en tiempo real utilizando la búsqueda en el árbol de Monte Carlo (MCTS) y también un filtro de Kalman extendido (EKF) como lo fue en (Patrick

Slade, 2017) (Patrick Slade Z. N., 2019), para luego comparar este método con un control de modelos predictivo (MCP) que determino como más efectiva la estimación de parámetros y la explotación dinámica del sistema con MCTS que el MCP. Pero en otros casos no se desestima el enfoque MCP para sistemas no lineales (Chen, 2016), que tienen en cuenta la incertidumbre tomando los peores casos (Campo, 2017) (Kouvaritakis, 2015) y utilizando restricciones probabilísticas (Mesbah, 2017).

Otros métodos utilizados para el aprendizaje de parámetros han sido el aprendizaje activo (D. A. Cohn, 2016), aprendizaje por esfuerzos (L. P. Kaelbling, 2016) (M. Wiering and M. van Otterlo, 2012), Control adaptativo (N. R. Kristensen, 2014), control neurodinamico (Tsitsiklis, 2015), control estocástico (Unbehaue, 2018) y redes neuronales artificiales (Mnih, 2015) (Uğur Demir, 2016), los cuales la estimación de parámetros no se hace en tiempo real (*offline*). Un método que tenía como objetivo reducir la incertidumbre del modelo ya establecido en el robot, propuso utilizar el aprendizaje de esfuerzo bayesiano (RL) y presentarlo como un proceso de decisión de Markov parcialmente observado (POMDP) *offline* (H. Bai, 2013). Estos métodos de estimación de parámetros con base a los datos son aprendizajes no supervisados, donde los datos de estudio no están etiquetados. En (M.N. Mahyuddin, 2014), se propone un algoritmo de control adaptativo en modo deslizante que tiene en cuenta también el error de estimación de parámetros, característica que lo hace robusto.

Otra estrategia de control está en (Deepak Gopinath, 2017), que pretende la personalización de un robot de asistencia destinado a personas con lesión de médula espinal (LME), con un control que es compartido entre el humano y la autonomía del dispositivo, es decir, una personalización dirigida por el usuario con una formulación matemática adaptativa de la intención de movimiento del usuario ante el robot, que estima y ajusta los parámetros de control con señales verbales del humano, haciendo también óptima la interacción hombre-máquina. En el control compartido, el enfoque de POMDP es usado también como estrategia de control optimo inverso (Shervin Javdani, 2015), que por su complejidad se deriva a una optimización retrospectiva para aproximar el ajuste de los parámetros de control. En (Shilpa Gulati, 2009) se propuso un marco matemático para hacer personalizable un robot móvil de asistencia, enfocado en medir la comodidad del usuario bajo trayectorias ya establecidas, donde en un control compartido el usuario puede modificar los pesos de la función de ajuste para la personalización.

2.2 Marco Teórico

2.2.1 Anatomía de la Médula Espinal

El encargado de la trasmisión, elaboración y recepción de la información es el sistema nervioso central (SNC), y está formado por el encéfalo y la médula espinal. El encéfalo está constituido por el cerebro, el cerebelo y el tallo cerebral, mientras que la médula espinal está compuesta por 31 segmentos espinales de los cuales se desprenden un par de nervios espinales, nervios periféricos o también llamados raquídeos, que mantienen la comunicación entre le médula espinal y las diferentes partes del cuerpo. La médula espinal tiene dos funciones fundamentales en el SNC, ser el centro de actos de reflejos y ser la vía de comunicación entre el cuerpo y el encéfalo. Su forma cilíndrica, comienza en el cráneo y termina en las primeras vértebras lumbares como se ve en la Figura 2.1 y tiene un diámetro medio de aproximadamente entre 8 y 10 milímetros (Simonetta Papa, 2020).

Figura 2.1. Comienzo de la médula espinal. Fuente: Autor.

En el sistema nervioso periférico (SNP) están los ganglios y nervios periféricos señalados en la Figura 2.2, que parten de la médula espinal y son los responsables de las capacidades motoras y sensoriales del cuerpo, ellos están compuestos por fibras llamadas “axones” que están rodeados de tejidos aislantes.

Un estímulo externo a estos nervios es transportado desde la médula espinal al cerebro, desde donde, de regreso se produce una respuesta a dicho estimulo.

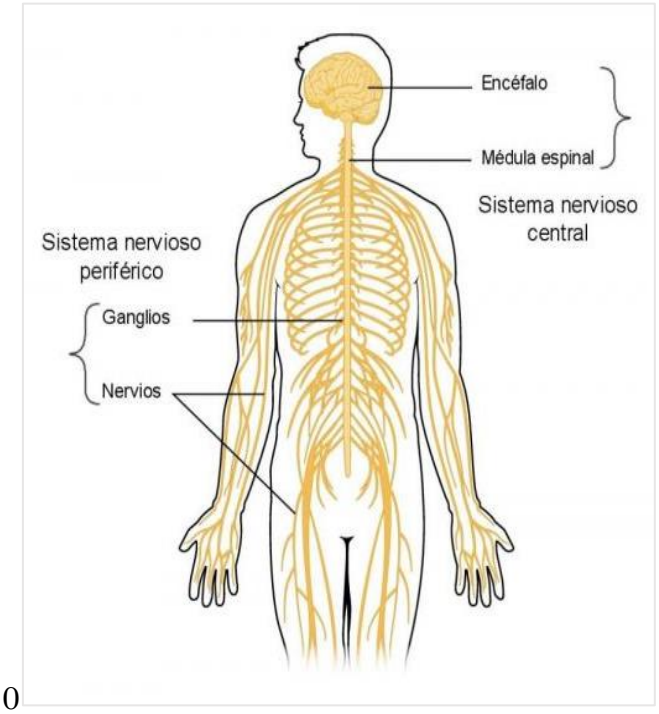


Figura 2.2. Sistema nervioso central y periféricos en el cuerpo humano. Fuente: Psicología-Online.

Las vértebras cervicales, torácicas, lumbares, sacra y coxis son las 5 regiones que dividen la columna vertebral y la médula espinal, estas se extienden desde el cráneo hasta la pelvis. Las vértebras cervicales son 7 y se encuentran en el área del cuello, estas vertebras están subdivididas en la región cervical superior (C1 y C2), la región cervical inferior (C3 a C7), y C0 el hueso occipital que está en la parte trasera de la cabeza. Las vértebras torácicas van desde la T1 a la T12 y están ubicadas alrededor del pecho, de ellas se sostiene la caja torácica y desde la T11 y T12 se encuentran las costillas flotantes. Las vértebras lumbares son las vértebras más grandes y van desde la L1 hasta la L5, la mayoría del peso corporal se apoya en ellas, tienen como objetivo proteger la médula espinal y las raíces nerviosas como también a muchos otros órganos internos, sirven como base de sujeción y apoyo a músculos, tendones, ligamentos, cabeza, hombros y pecho, en ellas es donde se conecta la parte inferior del cuerpo con la superior. La región sacra de la columna está ubicada entre los huesos de la cadera y ayuda a conectar la pelvis con la columna, está formada por 5 huesos desde el S1 hasta

el S5, y debajo de ellos otros 5 pequeños huesos forman el coxis (Keith Bridwell, 16). Lo anterior es más entendible observando la Figura 2.3.

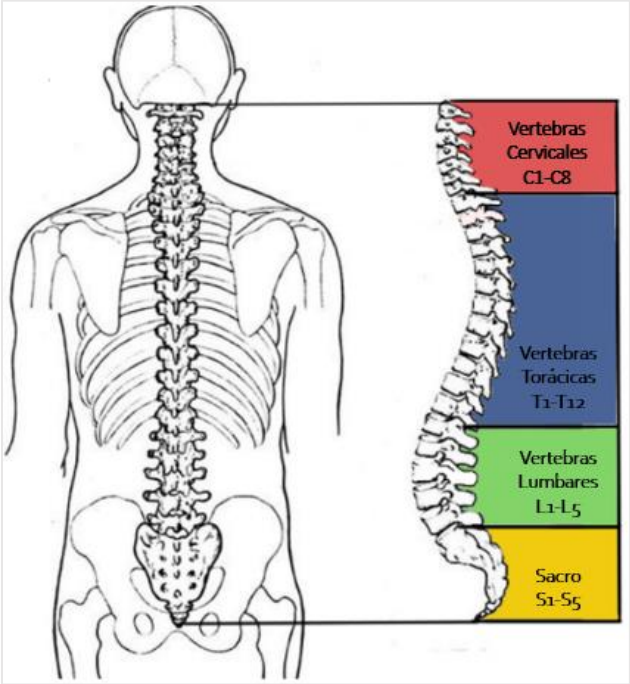


Figura 2.3. Regiones de la columna vertebral.

2.2.2 Lesión de Médula Espinal

La lesión de médula espinal (LME) es una afectación al cuerpo humano que implica la perdida de capacidades motoras y sensitivas, es una discapacidad física que puede ser ocasionada por la compresión traumática de la médula espinal por parte de los huesos cercanos o huesos discales de la columna vertebral (J.Windebank, 2020). Las causas de este tipo de lesión comúnmente son variadas, pero en su mayoría son accidentes de tráfico, caídas traumáticas, actos de violencia o actividades deportivas y recreativas, y también enfermedades como el cáncer, la artritis, la osteoporosis o la solo inflamación de la médula espinal (Research, 2019). Inmediatamente después del accidente se produce una dislocación en alguna vertebra y luego una hemorragia interna que puede incluso empeorar la salud vital.

Existen varias clasificaciones de este tipo de lesión, una de ellas depende del mecanismo de lesión. Si se produce una fractura, una dislocación o una subluxación del cuerpo vertebral se le puede clasificar como lesión de flexión, si se produce una fractura de la columna vertebral puede ser una lesión de tipo extensión y si hay una rotura de las vértebras, la lesión es de tipo axial (J.Windebank, 2020). Pero también se pueden clasificar como lesión de médula completa o incompleta, la primera se caracteriza por la pérdida total de la sensibilidad y el movimiento muscular llegando incluso a la parálisis total de los miembros inferiores y superiores o solo inferiores, mientras que la segunda es limitada, es parcial.

2.2.2.1 Niveles de lesión de médula espinal

Después de una lesión de médula espinal, la parálisis por debajo del nivel de lesión y la perdida de la sensibilidad son las consecuencias. El nivel de lesión se entiende a partir la región de la columna vertebral que tuvo la afectación, como se muestra en la Figura 2.4, y según esto se determinan las partes del cuerpo afectadas para luego clasificar la lesión como una lesión completa o incompleta.

Cuando la lesión es a nivel cervical, es decir, la lesión se encuentra a partir de cualquier vertebra desde la C1 hasta la C8, causa parálisis de algunos o de todos los músculos presentes en el proceso de la respiración, así como también los músculos de los brazos, piernas y tronco, dependiendo del nivel de lesión cervical también se presenta debilidad en los músculos de los hombros y codos, debilidad en los músculos encargados del movimiento de las manos, muñecas y dedos. (James E. Wilberger, 2017) (J.Windebank, 2020).

Cuando la lesión es a nivel torácica, se encuentra entre las vértebras T1 y T12. En este nivel de lesión también se puede sentir debilidad de los músculos de las manos y dedos, el movimiento de los hombros y codos por lo general es casi normal o más fluido que la lesión a nivel cervical, se mantiene también la parálisis de las piernas, tronco y zona lumbar, como también perdida de la sensibilidad a partir de los pezones, por debajo de la caja torácica, o del ombligo. (James E. Wilberger, 2017).

Cuando la lesión es a nivel lumbar, se encuentra entre las vértebras L1 y L5. Como es un nivel más bajo de lesión las secuelas con son más leves, aquí es posible la perdida de la sensibilidad de las caderas y piernas, pero no su parálisis

completa, presenta también posibles entumecimientos en la parte inferior del cuerpo. (James E. Wilberger, 2017).

Cuando la lesión es nivel sacro, se encuentra entre las vértebras S1 y S5. Es posible tener perdidas de sensibilidad leves comparadas con los anteriores niveles de lesión y un entumecimiento en el perineo (James E. Wilberger, 2017).

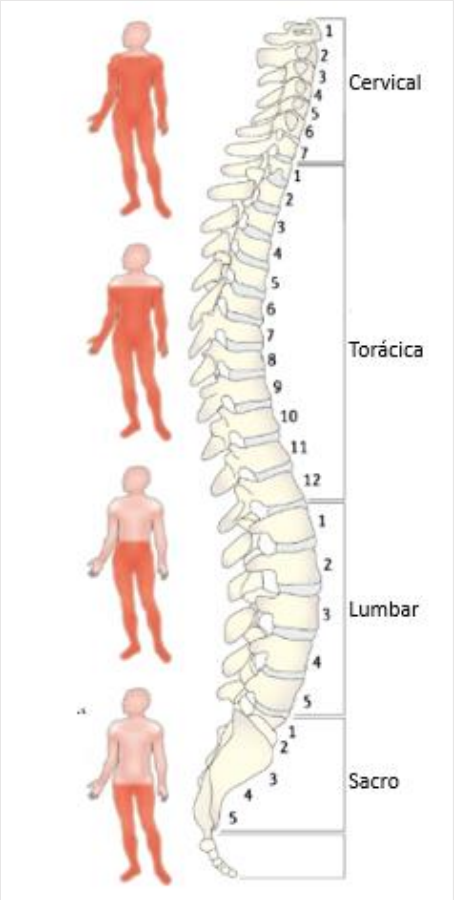


Figura 2.4. Niveles de lesión de médula espinal.
Fuente: (Simonetta Papa, 2020)

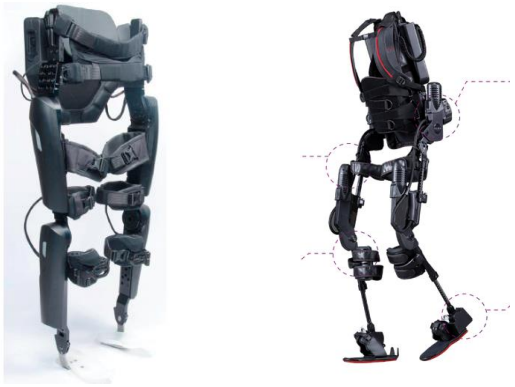
2.2.2.2 Robots de Asistencia y Rehabilitación para personas con Lesión de Medula Espinal

Desde que el ser humano ha tenido dificultades para moverse, este se ha destinado en solucionar estas limitaciones. Para las personas quienes padecen Lesión de Médula Espinal (LME), los robots de asistencia y rehabilitación adaptados a ellos son una buena solución que facilita sus vidas, mejora su

autonomía y solventa deficiencias motoras

Los exosqueletos motorizados han sido los más utilizados en personas con LME, ayudándolos a deambular libremente por un espacio determinado con independencia y mejorando su nivel de actividad. En la actualiza varios exosqueletos se encuentran habilitados comercialmente y también en hospitales para la realización de terapias de rehabilitación. Exoesqueletos como ReWalk (Esquenazi A, 2012), Ekso GT, Indego, REX (Casey Kandilakis, 2019) y HAL (Oliver Jansen, 2017) son de los más conocidos, de los cuales en los últimos años se han hecho ensayos clínicos en personas con LME y diferentes niveles de lesión.

ReWalk es un sistema de asistencia comercial que proporciona un movimiento motorizado de cadera y rodilla para que las personas con LME puedan caminar pararse, bajar y subir escaleras utilizando un exosqueleto mecánico en las piernas de la persona (Esquenazi A, 2012), ayuda a personas parapléjicas y es para uso de personas con un nivel de lesión entre el T7 y L5 (Ashraf S. Gorgey, 2019), ReWalk puede observarse en la Figura 2.5 a).



a) b)

Figura 2.5. a) Exosqueletos ReWalk y b) Ekso GT. Fuente: (Ashraf S. Gorgey, 2019) (Rebiotex, 2020)

Ekso, está diseñado para facilitar la marcha de una forma similar a la marcha fisiológica, en él se permite modificar los parámetros de paso, la altura y la distancia. Ekso estimula a los usuarios a estabilizar el tronco durante la marcha (Rebiotex, 2020). Ekso es adaptable según el nivel de lesión o discapacidad de su usuario, incluye también un conjunto de programas para ayudarles a mantener el equilibrio, y él modifica los entrenamientos en tiempo real según le tipo de asistencia que brindará (Bionics, 2020). Ekso es aplicable a personas con LME

por debajo de C6 (Ashraf S. Gorgey, 2019) y puede observarse en la figura 2.5 b).

Indego, es un exosqueleto que ofrece una terapia de marcha individualizada, hecha para personas con LME y con Accidentes Cardio Vasculares (ACV) (Indego, 2020). Indego tiene un hardware ajustable al humano como se muestra en la Figura 2.6. a), y al lugar en donde está el usuario, pues el Indego Personal es un exosqueleto inferior personalizado que permite mayor movilidad e independencia en el hogar o la comunidad, mientras que el Indego Therapy está pensado para su uso en un centro de rehabilitación en colaboración con un fisioterapeuta (Corporation, 2020). Puede ser usado en personas con LME por debajo de T3 bajo supervisión de un compañero especial (Ashraf S. Gorgey, 2019).

REX, es un exosqueleto robótico de rehabilitación que apoya y asiste a los usuarios a sentarse, pararse, y hacer ejercicios bajo un programa de ejercicios funcionales ya estructurado llamado REXERCISES, y también como los anteriores exosqueletos es un dispositivo ajustable al usuario según su discapacidad (Ltd, 2020). Según las pruebas y entrenamientos usando REX, este está apto para personas con un nivel de LME entre C4 y L5 (Ashraf S. Gorgey, 2019). El diseño de REX se muestra en la Figura 2.6 b).

a) b)

Figure 2.6. a) Exosqueletos Indego y b) REX. Fuente: (Ashraf S. Gorgey, 2019)

2.2.3 Qolo

Qolo es un exosqueleto de movilidad de pie ligero y móvil, que tiene un mecanismo de asistencia con actuadores que apoyan el cambio postural entre sentarse y pararse, él se opera a manos libres dejando la responsabilidad del movimiento de robot al torso del usuario, es decir, el robot es controlado con la parte superior del cuerpo. Con este vehículo es posible mantener una posición erguida y viajar de pie en el entorno, gracias a su mecanismo de locomoción vertical mostrado en la Figura 2.7 (Mobility, 2019).

Figura 2.7. Qolo. Fuente: Autor

2.2.3.1 Interfaz de Control de Qolo

Para garantizar que un usuario con deterioro motriz de la parte inferior del cuerpo por LME realice una locomoción a manos libres en Qolo, se introdujo una interfaz de control de movimiento en la cual por medio de una barra compuesta por 10 resistencias sensitivas a la fuerza (FSR – en inglés) que quedan a la altura de la cintura del usuario, como se observa en la Figura 2.8 (Yang Chen D. P.-G.).

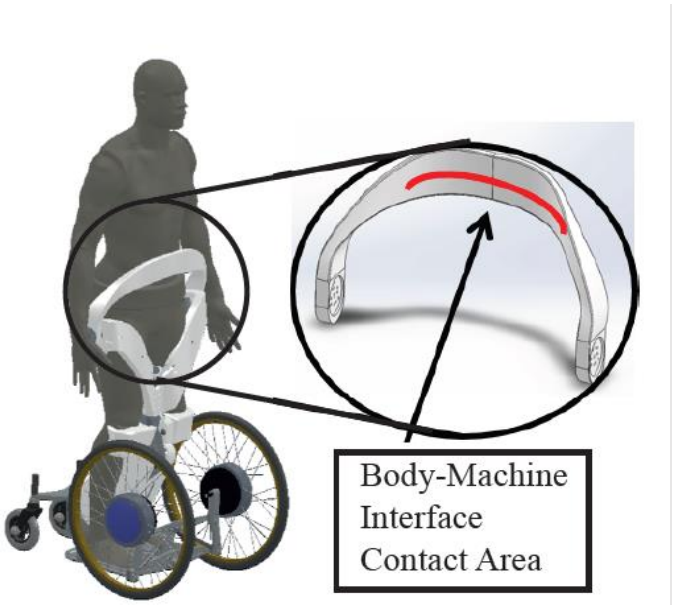


Figura 2.8. Ubicación de la interfaz de control de Qolo.
Fuente: (Yang Chen D. P.-G.).

Los FSR son sensores que permiten detectar la presión física y el peso. Este sensor cambia su valor resistivo dependiendo de la presión, la fuerza o tensión que se le aplique, si la presión es nula, la resistencia es infinita (circuito abierto), si la presión aumenta, la resistencia disminuye (Adafruit, 2020). Los FSR tienen aplicaciones en muchos campos como, sensores de ocupación de automóviles, extremidades artificiales, robótica de rehabilitación y de asistencia, y la el mejoramiento de la interacción móvil. El sensor usado en Qolo es de la forma: (Figura 2.9)

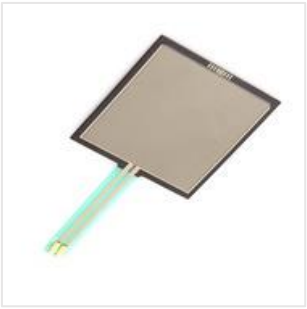


Figura 2.9 Resistencia sensitiva a la fuerza (FSR). Fuente: Electronica.com

El humano en posición erguida apoya la parte abdominal en la barra sensórica (T-bar) y de ese modo puede controlar al vehículo según el desplazamiento de la parte superior del cuerpo, modificando el centro de presión (COP – en inglés) en el conjunto de sensores. La dirección del movimiento del robot es proporcional a la ubicación del COP en la T-bar, en la cual están distribuidos en línea todos los sensores (Yang Chen D. P.-G.). Para el cálculo del COP se toman las lecturas de los FSR en un rango de [-1,1], y a partir de allí se clasifican las posturas mostradas en la Figura 2.11 con los puntos β_{-2}, β_{-1}, β_1 y β_2 que son los parámetros de control para la calibración de Qolo, donde sus ubicaciones en la T-bar se muestra en la Figura 2.10.

Figura 2.10. Ubicación de las parámetros de control en la T-bar. Fuente: Autor.

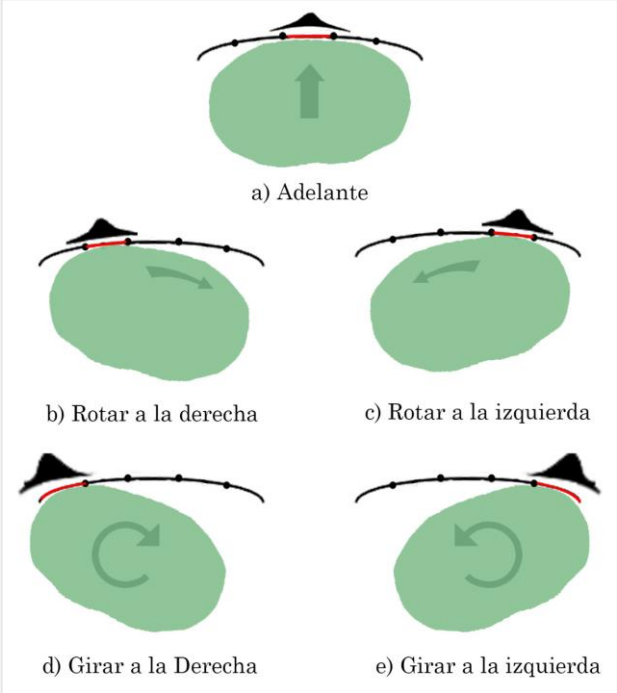


Figura 2.11. Posturas clasificadas en la T-bar. Fuente: Autor.

2.2.4 Interfaz Interactiva

Para familiarizar al usuario con el robot Qolo, se desarrolló una interfaz interactiva donde se observa el movimiento en tiempo real generado en el robot por el desplazamiento o torsión de la parte superior del cuerpo del usuario. La interfaz interactiva es observada en una aplicación web conectada a ROS y a Qolo.

2.2.4.1 Sistema Operativo del Robot

El sistema operativo del robot (ROS – *Robot Operating System* en inglés), es una herramienta que ayuda a los desarrolladores de software a crear aplicaciones de robots, sus librerías están orientadas para un sistema operativo UNIX (Ubuntu- Linux) principalmente, cuenta con un conglomerado de paquetes aportados por la contribución de sus usuarios que son útiles para la localización y mapeo simultaneo, planificación, percepción y simulación, las librerías y herramientas de ROS están bajo licencia de código abierto BSD (*Berkeley Software Distribution* en inglés) (Gandul, 2014), su logo se muestra en la Figura 2.12.

Figura 2.12. Logo de ROS.

ROS proporciona la comunicación entre procesos con una interfaz de paso de mensajes que se conoce comúnmente como *middleware*, el cual cuenta con un mecanismo de publicación y suscripción anónimo (ROS, Componentes centrales de ROS, 2020), que por medio de temas o tópicos se realiza la comunicación unidireccional. Los temas son buses donde los nodos intercambian mensajes, en donde si el nodo está interesado en los datos se suscribe al tema, y si está generando datos se publica en el tema (Foundation, 2020), la comunicación entre

nodos se puede entender mejor observando la Figura 2.13. ROS permite capturar y reproducir datos fácilmente, reduciendo el esfuerzo de desarrollo, promoviendo flexibilidad y modularidad en su sistema. Por ejemplo, ROS facilita la captura de datos publicados en un tópico que lee datos de un sensor y al mismo tiempo desarrolla la tarea implícita en otro tópico que procesa estos dichos datos, esto permite que el procesamiento de los datos de haga de manera independiente al origen de ellos (ROS, Componentes centrales de ROS, 2020). Cuando el transporte unidireccional de datos no es apropiado para las interacciones necesarias, ROS proporciona un sistema distribuido a través de un Servicio, que es definido mediante un mensaje de solicitud y otro de respuesta, donde un nodo de ROS proveedor ofrece el servicio y un cliente llama al servicio enviando la solicitud y esperando la respuesta (Foundation, Services, 2020).

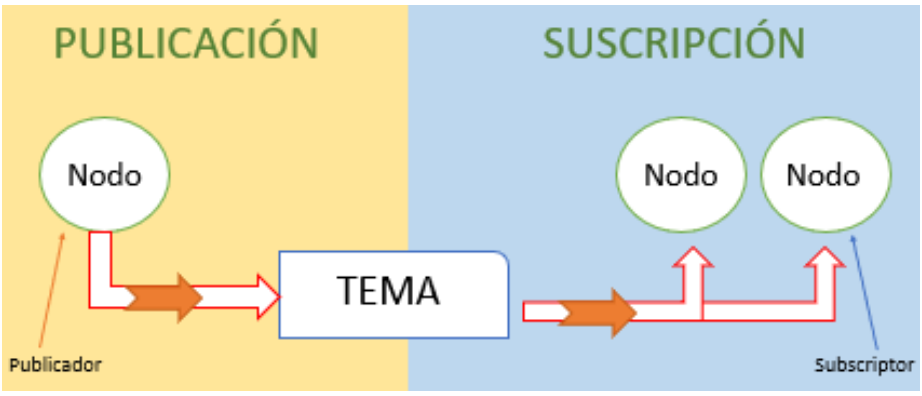


Figura 2.13. Comunicación entre nodos en ROS. Fuente: Autor.

ROS también proporciona bibliotecas y herramientas específicas para que el robot funcione rápidamente, por ejemplo, las definiciones estándar de mensajes en el robot para conceptos geométricos de vectores, transformadas y poses, y para datos de navegación como la odometría, rutas y mapas (ROS, Componentes centrales de ROS, 2020); proporciona también bibliotecas de geometría de robot que permiten monitorear múltiples coordenadas a lo largo del tiempo, un lenguaje de descripción del robot de forma legible por máquina que utiliza un conjunto de herramientas para describirlo y modelarlo. ROS también ofrece una forma de producir, recopilar y agregar diagnósticos sobre el robot, observa su estado y determinar una solución a los problemas que surjan. Los anteriores y otros componentes centrales de ROS están clasificados y enlistados en la Figura 2.14, donde se encuentran las bibliotecas para la capa de cliente, para

aplicaciones robóticas, para la capa de comunicación, para la capa de interfaz para hardware, herramientas de desarrollo y simulaciones (ROS, Componentes centrales de ROS, 2020).

Las herramientas de desarrollo de ROS admiten la visualización, introspección y depuración del estado del sistema, hay líneas de comando que permiten iniciar grupos de nodos, servicios y acciones, realizar grabaciones y reproducir datos. Por ejemplo, *rviz* es una herramienta gráfica que permite visualizar en tercera dimensión datos de sensores de cualquier robot, y *rqt* una herramienta para el desarrollo de interfaces gráficas personalizadas para los robots (ROS, Componentes centrales de ROS, 2020).

Figura 2.14. Componentes en ROS. Fuente: (YoonSeok Pyo, 2017)

Las bibliotecas de cliente de ROS facilitan el trabajo a la hora de programar ROS, estas bibliotecas permiten realizar funciones de creación de nodos, publicaciones y suscripción a temas, como también escribir y llamar servicios y más, se puede implementar en cualquier lenguaje de programación, pero actualmente hay mayor soporte para C++, Python y menor para Octave/Matlab, LISP y otros lenguajes como Java (Robotics O. S., 2018). La biblioteca cliente de

Python para ROS es *rospy*, que proporciona un lenguaje de script orientado a objetos, favoreciendo la velocidad de implementación para que los algoritmos puedan ser probados con ROS.

Un sistema ROS en ejecución puede comprender docenas de nodos distribuidos en múltiples máquinas, pero para ello, debe existir una buena configuración de la red, que es garantizada por una conectividad bidireccional entre todos los pares de máquinas y en todos los puertos, también por la asignación de un maestro y que todos los nodos estén configurados para usar ese mismo maestro, y por último, que cada máquina sea anunciada con un nombre que haga que las otras máquinas la distingan (Foundation, Múltiples Máquinas, 2019).

El maestro ROS es el encargado de gestionar la ejecución de los nodos y de posibilitar su comunicación en el sistema (García, 2016). Para registrar el nombre de cada nodo u obtener la información necesaria debe ejecutarse el maestro con el comando *roscore*. Si no se asigna un maestro es imposible realizar la conexión entre los nodos y la comunicación de mensajes como temas y servicios (YoonSeok Pyo, 2017). El maestro se comunica con los esclavos por medio de XML-RPC que es un protocolo basado en HTTP (Protocolo de transferencia de hipertexto) que soporta una variedad de lenguajes de programación y de hardware. El maestro se configura con la dirección URI que usa la IP de PC y el puerto configurado en ROS_MASTER_URI que por lo general es 11311. (YoonSeok Pyo, 2017).

2.2.4.3 ROS y Aplicación Web

Una página web es un documento al que se puede acceder desde cualquier navegador con acceso a internet, y son usadas en aplicaciones robóticas como recurso para facilitar la interacción de los usuarios con el robot.

Robot Web Tools (RWT) es un paquete de ROS que permite conectar a las aplicaciones web con una variedad de robots, escritas bajo lenguajes como HTML5 y JavaScript (JS). Este paquete fue desarrollado con la idea de crear interfaces robot-humano más usables e intuitivas y de posibilitar el uso de la robótica en la nube (García, 2016). Para el transporte de datos, *Rosbrigde* es un paquete que ha ayudado a extender la funcionalidad de ROS a la web. *Rosbrigde* convierte toda la arquitectura de ROS en un *backend* (donde se realiza la lógica de una página web, es decir, la parte trasera que realiza un conjunto de acciones

que no vemos pero que modifican las pagina según la interacción con el usuario), por lo que no es necesario que los desarrolladores de aplicación web tengan conocimientos sobre sistemas basados en *middleware* (García, 2016). *Rosbrigde* proporciona acceso a los mensajes y servicios de ROS como objetos de JSON (*JayaScript* Object Notation en inglés) y permite publicar y suscribirse con la ejecución de los nodos y realizar llamadas de servicio con ayuda de la biblioteca *roslibjs* que sirve de comunicación con ROS en el robot a través del servidor WebSocket (Fetch Robotics, 2020). Entonces, el flujo de datos entre el navegador, el servidor web y el maestro ROS según los anterior seria de la forma expuesta en la Figura 2.15.

El protocolo WebSocket que está construido en HTML (*HyperText Markup Language* en inglés), y es un modelo sencillo y elegante de comunicaciones para la web, este permite crear una comunicación bidireccional sobre una conexión TCP (Protocolo de control de transmisión) que se realizan a través de los mismos puertos que utiliza HTTP con el fin de ofrecer compatibilidad con el software del servidor ya existente (Marco, 2015).

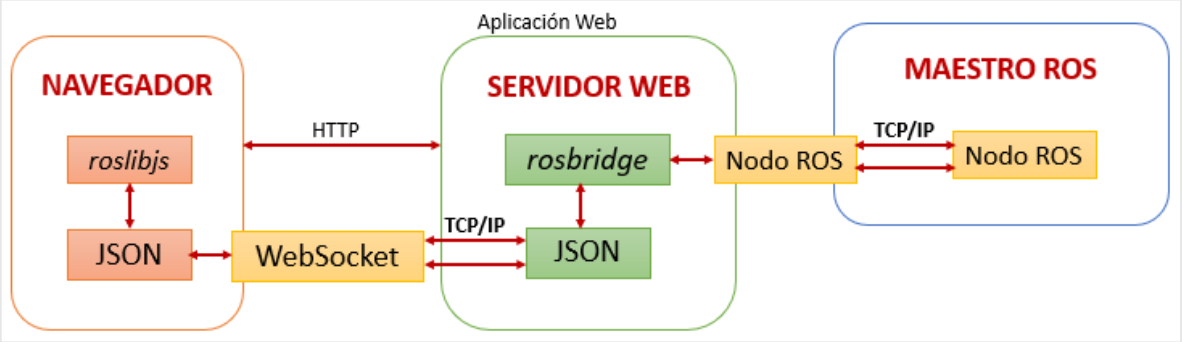


Figura 2.15. Flujo de datos entre el navegador, el servidos web y el maestro ROS. Fuente: (Profanter, 2014)

2.2.4.4 ROS y Matlab

Matlab es la abreviatura de "Laboratorio de matrices" en inglés, es un programa computacional que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (Elizondo, 2012). Matlab es una herramienta que puede manejar cálculos de problemas de ingeniería y ciencia, desde la evaluación de una función hasta la resolución de grandes y complejos sistemas de ecuaciones. Matlab es interactiva y flexible, pues no siempre es

necesario crear códigos o definir variables sino basta con usar las instrucciones y obtener una respuesta, o crear programas específicos. Matlab es compatible con una gran variedad de software y posee una cantidad considerable de herramientas llamadas *Toolboxes* que son también usadas para la resolución de problemas de ingeniería avanzada como, por ejemplo, machine learning o aprendizaje automático, redes neuronales, procesamiento de imágenes y video, control de procesos, algoritmos genéticos y más. Con Matlab se pueden elaborar programas para trabajar con hardware industrial, microprocesadores, dispositivos móviles y más (Requez, 2020).

Para el intercambio de datos con editores y suscriptores de ROS, se acude a un mecanismo de mensajes que permite intercambiar datos y guardarlos. Los mensajes son enviados sobre un tema o tópico con nombre único en la red de ROS de la cual cualquier nodo podrá suscribir para recibir información o compartirla usando un editor para enviar datos, lo anterior puede ser más entendible observando la Figura 2.16, donde desde el nodo publicador sale la información transportada por el tema, que luego es recibida en un nodo suscritor.

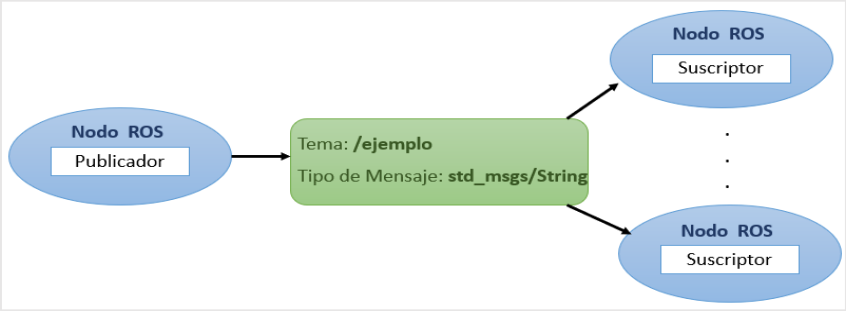


Figura 2.16 Flujo de datos en entre publicador y suscriptor en Matlab. Fuente: MathWoks.

CAPÍTULO III

Metodología

En este capítulo se describe la metodología utilizada en el proyecto. Inicialmente se presenta configuración de la interfaz interactiva utilizada para la toma de datos, seguidamente la implementación del marco matemático adaptativo con los datos ya obtenidos y por último la evaluación de la estrategia de calibración.

3.1 Metodología de la Investigación

La metodología desarrollada en este trabajo de investigación es para la calibración de los parámetros de control del vehículo de movilidad de pie Qolo, adquiriendo los datos con una aplicación web interactiva y ROS, los cuales se toman para la calibración en una etapa de control *offline* utilizando el *software* Matlab y su evaluación en una simulación en la misma aplicación web (Figura 3.1). A continuación, se presenta un esquema descriptivo de los anterior, es la línea metodología usada:

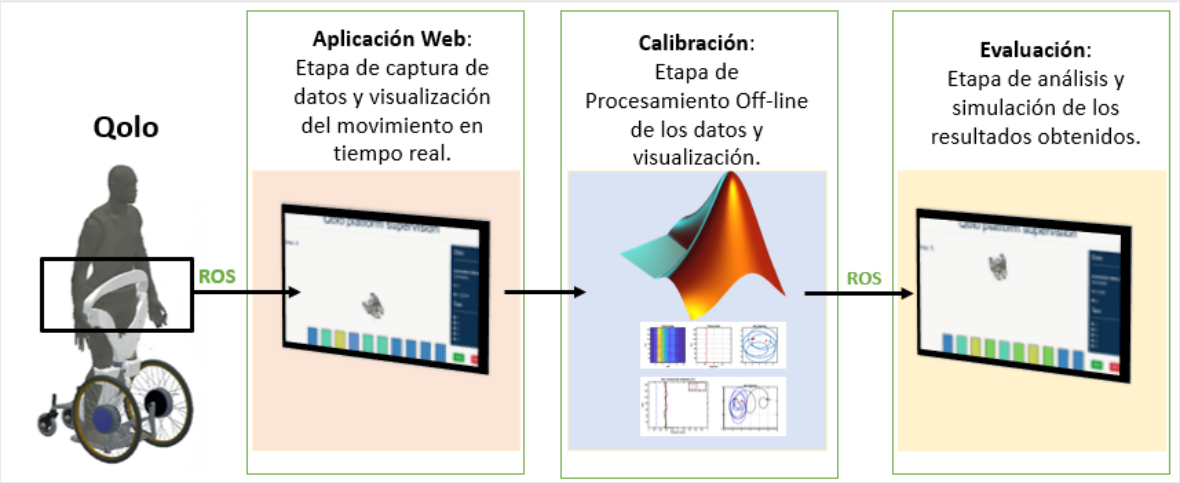


Figura 3.1. Linea metodológica usada en la investigación. Fuente: Autor

Antes de la etapa de captura de datos se hizo un reconocimiento del funcionamiento de Qolo, sus alcances y la población con lesión de médula espinal a la que está destinado, y se analizaron los parámetros que pueden ser modificados en el control del vehículo.

Se plantearon varias estrategias para el protocolo de calibración como, por ejemplo, realizarla bajo parámetros de referencia ajustados por el usuario con una interfaz de presión, en donde, se esperaba que fuera un control compartido entre el humano y el robot, se definieron un conjunto de parámetros posibles que medirían e indicarían el valor mínimo que determina el momento en que comienza el control compartido dependiendo de la intención del usuario y el ajuste que hace el mismo.

Para modelar la cantidad de control compartido, se contarían las veces que, sobre una misma rutina o trayectoria establecida, el usuario manifestara que el robot no estaba moviéndose según su deseo, aplicando presión en los FSR extremos como se indica en la Figura 3.2. Si presionaba al FSR izquierdo quería decir que el usuario rotaría o giraría más a la izquierda de lo que hizo realmente el robot, y si presionaba al FSR derecho era lo contrario.

Explicándole al usuario con anterioridad la rutina establecida, y que la cantidad de presiones o fuerzas aplicadas a los FSR era proporcionar a la cantidad de descontento del usuario frente al movimiento del robot y la cantidad de control compartido. Siendo entonces 5 el valor en donde el control compartido seria máximo y 0 donde seria mínimo. Estas rutinas serian luego comparadas con otras en donde no se hizo una calibración sobre los parámetros de control, para determinar el factor de cambio de intención del usuario, medir el posible error en las trayectorias y el error del usuario, y poderlo luego clasificar por grupos.

Figura 3.2 FSR usados para el control compartido. Fuente: Autor

Después de analizar la anterior estrategia ésta se modificó, el control compartido, aunque es una buena estrategia no fue aplicado, y también se consideró modifica la forma en cómo se capturaban los datos, pues en las rutinas establecidas se pretendía colocar unas guías en el suelo, pero estas en algunos casos harían que los usuarios esforzaran sus movimientos del torso para seguirlas y al mismo tiempo tendrían que presionar algunos de los FSR si el movimiento resultante no era el deseado. Según los anterior, realizar estas dos tareas al mismo tiempo no es recomendado para personas que tienes una LME.

Entonces, se decidió desarrollar una aplicación web, que le permitiría al usuario ver en tiempo real por medio de un avatar, el movimiento que su desplazamiento del torso generaba en el robot estando estático. Esto garantizaba también que la persona con LME pudiera familiarizarse con el robot y generarle confianza antes de manejarlo.

3.2 Aplicación Web Interactiva.

Antes de acudir al desarrollo de la aplicación web, se debe familiarizar con ROS, instalarlo, conectarlo con el ROS maestro, crear un espacio de trabajo, agregar las bibliotecas y dependencias necesarias, y conocer y manejar otras terminologías importantes de ROS para la comunicación. Después, reconocer como transportar los datos desde ROS a la aplicación web para poderlos mostrar allí, y por último su respectivo diseño.

3.2.1 Familiarización con ROS

La instalación de ROS Kinetic en Ubuntu 16.04 (Xenial) se hará a través de una instalación binaria, para ellos se siguieron los siguientes pasos (Joseph, 2018):

1. Se configuraron los repositorios de Ubuntu: El repositorio es donde está el software de Ubuntu organizado. Se debe habilitar el acceso a todo para que Ubuntu pueda recuperar los paquetes, y esto debe hacerse en "*Software y actualizaciones*".
2. Se añadió la información del repositorio de ROS donde se almacenan los binarios, para esto se ejecutó en una terminal el siguiente comando:

   ```
   $ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu
   $ (lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
   ```

3. Para descargar un binario en Ubuntu, se añadió una clave segura en el sistema para autenticar el proceso de descarga. El siguiente comando fue el necesario para agregar la clave:

   ```
   $ sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key
   421C365BD9FF1F717815A3895523BAEEB01FA116
   ```

4. Para actualizar la lista de paquetes de Ubuntu, se usó el siguiente comando:

   ```
   $ sudo apt-get update
   ```

5. Se instalaron los paquetes de ROS Kinetic, sus herramientas, sus simuladores y más con el siguiente comando:

```
$ sudo apt-get install ros-kinetic-desktop-full
```

6. Se instaló *rosdep*, una herramienta útil para instalar los paquetes dependientes de un paquete ROS:

```
$ sudo rosdep init
$ rosdep update
```

7. Para establecer un entorno ROS y acceder a las herramientas y librerías de ROS, se agregaron las siguientes líneas:

```
$ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

8. Por último, se crearon las dependencias para la construcción de paquetes:

```
$ sudo apt-get install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

Después de instalado ROS, se creó un espacio de trabajo donde guardaron los paquetes de ROS necesarios para el desarrollo de la aplicación web.

En una terminal independiente, se introdujo el siguiente comando para crear una carpeta llamada *catkin_ws*, que tendría luego una subcarpeta llamada *src* donde se construyó el paquete:(Joseph, 2018):

```
$ mkdir -p ~/catkin_ws/src
$ cd catkin_ws/src
```

Antes de crear o construir paquetes correctamente se inicializó el espacio de trabajo con la siguiente línea:

```
$ catkin_init_workspace
```

Después, dentro de la carpeta *src* se creó un archivo .txt y .xml llamados *CMakeLists.txt* y *package.xml* respectivamente. Con lo anterior, se pudo construir el espacio de trabajo en la carpeta *catkin*_ws con la siguiente línea:

```
$ ~/catkin_ws
$ catkin_make
```

Luego, en la carpeta *catkin_ws* se encontraron dos carpetas adicionales a *src* llamadas *"build"* y *"devel"*. Los pasos anteriores fueron la guía para crear el espacio de trabajo de la aplicación web que fue llamado *Qolo*.

Ya creado el espacio de trabajo para la aplicación web de la forma anterior, se crearon los paquetes donde se organizarían los nodos y las bibliotecas que se usaron para la aplicación web. Con la siguiente línea podemos creó un paquete ROS:

```
$ catkin_create_pkg robot_gui_bridge --catkin-deps rosbridge_server
```

El primer parámetro de la línea anterior es el nombre del paquete *robot_gui_bridge* y el segundo parámetro es el nombre de las dependencias, que en este caso es *rosbridge_server*, las cuales proporcionaron una conexión WebSocket para que los navegadores pudieran a hablar con ROS.

3.1.1.2 Transporte de Datos desde ROS a la Aplicación Web

Terminado de instalar ROS, haber construido un espacio de trabajo y un paquete, se crearon los paquetes necesarios y se instalaron las dependencias necesarias. Por ejemplo, se creó un paquete llamado *speed_data* donde se guardaría el código *listener_qolo.py*, y se instalaron las dependencias con el comando *rosdep*.

Con las siguientes líneas se agregó un archivo de inicio simple en el primer paquete creado:

```
$ mkdir ~/Qolo/src/robot_gui_bridge/launch
$ touch ~/Qolo/src/robot_gui_bridge/launch/websocket.launch
```

Para comenzar a escribir la aplicación web en, se usó la siguiente línea:

```
$ mkdir ~/Qolo/src/robot_gui_bridge/gui/index.html
```

El código *index.html* es la página principal de la aplicación web, y donde se observa el avatar del robot Qolo en movimiento. Para hablar con ROS desde este archivo .html se agregó la librería *roslibjs* (ROS JavaScript Library), que es la librería central de JavaScript para interactuar con ROS desde cualquier navegador, que utiliza WebSocket para conectar con *rosbridge* y poder publicar, suscribir, llamar servicios y más (Foundation, The Standard ROS JavaScript Library, 2019) (Sadowski, 2019).

Se comenzó el script *index.html* importando la librería *roslibjs* de la siguiente manera:

```
var ros = new ROSLIB.Ros({
    url: 'ws://192.168.13.110:9090'
  });
```

La dirección IP agregada pertenece a la dirección IP de Qolo que es el localhost en el puerto 9090. Para reconocer en tiempo real el estado de conexión de la aplicación web con ROS, creó un detector del evento de la siguiente manera:

```
ros.on('connection', function () {
    document.getElementById("status").innerHTML = "Connected";
  });
```

En la línea anterior, *document.getElementById* devuelve una referencia, que en este caso es “Connected” según su ID (Identificación – “status”).

Nota: Los datos de Qolo originalmente se pueden observar mediante un archivo .log dado por ROS. Este archivo muestra en tiempo real, las lecturas de los 10 FSR en la T-bar, el valor del centro de presión (COP), las velocidades angular y lineal y otro.

El primer objetivo planteado para avanzar en la captura de datos con la aplicación web, fue poder recibir los datos anteriormente mencionados desde

ROS. Por lo tanto, se suscribió a un tema para mostrarlo en la ventana del navegador. Los primeros datos que se quisieron mostrar en la aplicación web son las velocidades lineal y angular, y los valores de los FSR, dado que con estos se puede monitorear constantemente en que parte de la T-bar el usuario está realizando presión como se ve en la Figura 3.3, y las velocidades resultantes en Figura 3.4.

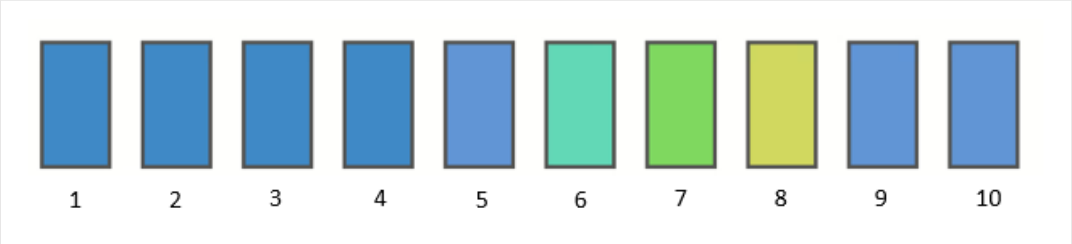


Figura 3.3. Representación de la T-bar en la aplicación web.
Fuente: Autor.

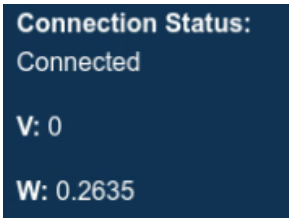


Figura 3.4. Estado de conexión, velocidad lineal y angular en la aplicación web.
Fuente: Autor

Para los anteriores datos, se suscribió a un tema de la siguiente manera:

```
var lineal_v_listener = new ROSLIB.Topic({
    ros: ros,
    name: '/lineal_v',
    messageType: 'std_msgs/Float64'
  });
  lineal_v_listener.subscribe(function (m1) {
```

```
    document.getElementById("lineal_vel").innerHTML = m1.data;
});
```

Las anteriores líneas de código pertenecen a *index.html*, y son un ejemplo de cómo suscribió a un tema para poder obtener datos y mostrarlos en la aplicación web. En estas se creó un el objeto llamado *lineal_v_listener* de la clase ROSLIB.Topic, que tiene 3 argumentos que son al nombre del tema (*lineal_v*), el tipo de envío de datos (*std_msgs*) y el tipo de variable a recibir (*Float64*). De esta manera se hizo con los demás tópicos o temas suscritos que son:

- *angular_v*: Proporciona la velocidad angular del robot.
- *th_qolo*: Proporciona el ángulo que da dirección al avatar en la aplicación web y fue hallado a partir de la velocidad angular.
- *xy_qolo*: Proporciona las posiciones (x, y) del avatar en la aplicación web y fue hallado a partir de la velocidad angular y lineal.
- *sensors_data*: Proporciona los valores de cada uno de los FSR en la T-bar.
- *time_qolo:* Proporciona el tiempo de ROS.

Los temas suscritos anteriores vienen de los publicadores del archivo llamado *listener_qolo.py*, un código que toma las variables o datos que se necesita de Qolo y se los proporciona la aplicación web. En este código, se hace también el cálculo de la posición y dirección actual del robot, el cálculo para la obtención de parámetros de control para la calibración, y donde se guardan los archivos finales después de usada la aplicación web por el usuario.

Un ejemplo de la línea de comando en Python usada para publicar en un tema es de la siguiente forma:

```
pub_v = rospy.Publisher('lineal_v', Float64, queue_size = 10)
```

Los argumentos necesario para crear un *rospy.Publisher* son, el nombre del tema (*lineal_v*), la clase de mensaje (*Float64*) y el tamaño de la cola (Foundation, Editores y suscriptores, 2019). El siguiente esquema demuestra la comunicación entre los nodos (Figura 3.5):

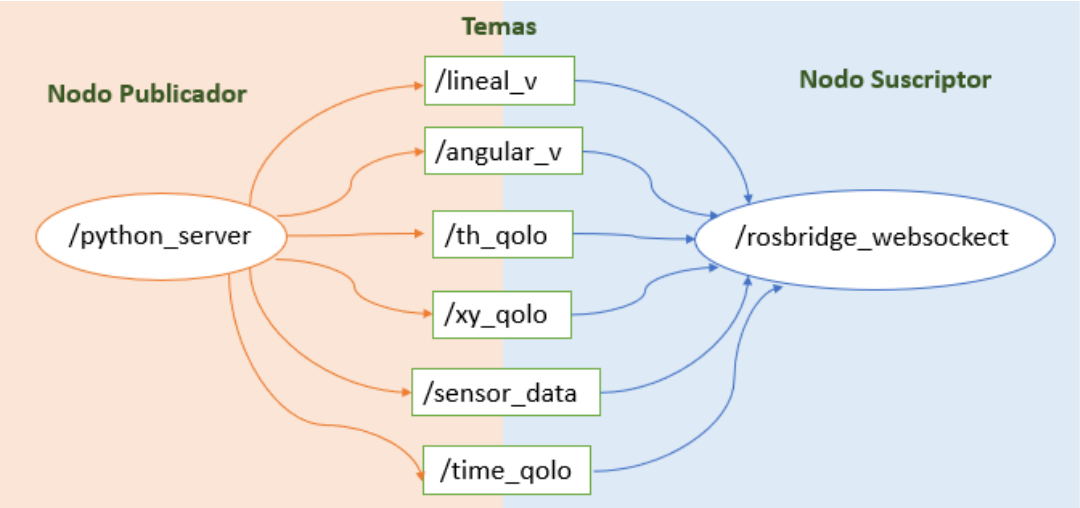


Figura 3.5. Nodo publicador, temas y nodo suscriptor para el diseño de aplicación web. Fuente: Autor.

Nota: Esta aplicación web se puede adaptar a diferentes dispositivos, para eso se hizo un guía independiente que es fácil de entender, y estará agregada en los Anexos I. El nombre de la guía es "Recolección de los datos y obtención de parámetros de control de los datos".

3.2.2 Diseño de la Aplicación Web

Luego de recibidos los datos en la aplicación web, para el diseño se determinaron los siguientes requisitos:

1. Debe mostrarse constantemente el estado de conexión de la aplicación web con ROS.
2. El movimiento que el usuario está generando en el robot con la T-bar, deber ser visto por medio de una avatar o robot virtual.
3. Deben verse claramente las velocidades lineal y angular en tiempo real mientras el usuario interactúa con la aplicación web o genera movimiento en el avatar.
4. Debe verse claramente donde el usuario está generando presión en la T_bar según las lecturas de los FSR.
5. La duración por cada test debe ser de 15 segundos o menos.
6. Debe haber un RadioButton donde el usuario pueda indicar el test a realizar

7. Debe haber un botón de “Start” donde el usuario indicara el comienzo de un test.
8. Debe haber un botón de “End” donde el usuario indicara que ha finalizado todos los 5 test propuestos y pase a la siguiente página de la aplicación web (*qolo_2.html*), donde se muestra la modificación de los parámetros de control con respecto a los test ya realizados.

Las indicaciones de manejo de la aplicación web se dieron de manera verbal antes de la primera interacción entre el usuario y la aplicación. Los siguientes Test fueron los propuestos para que el usuario los realizara durante su interacción con aplicación web:

Test 1: El usuario debe girar su cuerpo a la derecha como si fuera a girar completamente. (El movimiento ideal que debería hacer el usuario en este test se puede ver en la d) de la Figura 2.11)

Test 2: El usuario debe girar su cuerpo a la izquierda como si fuera a girar completamente. (El movimiento ideal que debería hacer en este test el usuario se puede ver en la e) de la Figura 2.11)

Test 3: El usuario debe dirigir su cuerpo o inclinarse hacia adelante. (El movimiento ideal que debería hacer en este test se puede ver en la a) de la Figura 2.11)

Test 4: El usuario debe tratar de hacer un círculo uniforme hacia la derecha.

Test 5: El usuario debe tratar de hacer un círculo uniforme hacia la izquierda.

Nota: Los tres primeros test anteriores son quienes apoyan la etapa calibración *offline* en el software Matlab, pues son lo que brindan la información necearía del usuario para la modificación de los parámetros.

Para el cumplimiento de los requisitos anteriores, se escribieron las siguientes líneas:

1. Estado de conexión: Para la visualización en la aplicación web del estado de conexión con ROS, se agregaron siguientes líneas al código *index.html*, las cuales permitieron observar si el estado actual es conectado como se mostró en la Figura 3.4, si existe algún error o si esta desconectado:

```
<p><b>Connection Status: </b><span id="status"></span></p>
```

Las instrucciones para el estado de conexión son:

```
ros.on('connection', function () {
  document.getElementById("status").innerHTML = "Connected";
});

ros.on('error', function (error) {
  document.getElementById("status").innerHTML = "Error";
});

ros.on('close', function () {
  document.getElementById("status").innerHTML = "Closed";
});
```

En la líneas anteriores, *document.getElementById* devuelve una referencia, que en este caso es "Connected", "Error" o "Closed" según su ID (Identificación – "status").

2. Visualización del avatar: Las siguientes líneas de código permitiero visualizar del avatar de la Figura 3.6 en la aplicación web.

```
<div class="col-md-8">
      <!-- Robot image  -->
      <img  src="https://i.ibb.co/0KvRCM7/carrito-qolo.jpg"  class="img-fluid"
alt="Responsive image" id="qolo">
    </div>
```

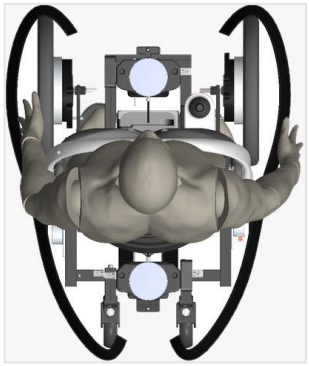


Figura 3.6. Avatar de Qolo *en la aplicación web.*

Fuente Dr. Diego Felipe Paez.

3. Visualización de las velocidades: Las siguientes líneas de código permitieron visualizar de las velocidades lineal y angular en la aplicación web como se mostró en la Figura 3.4.

```
<!-- Display Linear Speed  -->
<p><b>V: </b><span id="lineal_vel"></span></p>

 <!-- Display Angular Speed  -->
<p"><b>W: </b><span id="angular_vel"></span></p>
```

Las instrucciones de como se hizo el transporte de las velocidades desde ROS se hacen la sección 3.1.1.2, pero las siguientes lineal lo complementan:

```
var lineal_v_listener = new ROSLIB.Topic({
        ros: ros,
        name: '/lineal_v',
        messageType: 'std_msgs/Float64'
  });
lineal_v_listener.subscribe(function (m1) {
      document.getElementById("lineal_vel").innerHTML = m1.data;
  });

var angular_v_listener = new ROSLIB.Topic({
        ros: ros,
```

```
        name: '/angular_v',
        messageType: 'std_msgs/Float64'
  });
angular_v_listener.subscribe(function (m2) {
      document.getElementById("angular_vel").innerHTML = m2.data;
  });
```

4. Visualización de la T-bar: Se planteó crear 10 cuadros que cambien de color según el valor que reciben de los FSR como se distingue en la Figura 3.3. Las siguientes líneas fueron usadas para crear los cuadros:

```
#cuadro-1 {
    margin-left: 9em;
    width: 3.5em;
    height: 100px;
    border: 3px solid #555;
    background: #428bca;
  }
```

Para crear los 9 cuadros faltantes se replican las anteriores líneas y luego se acomodan en aplicación web. Las instrucciones de como se hace el transporte de los datos de los FSR desde ROS se hacen la sección 3.1.1.2, pero las siguientes lineal lo complementan. También se le asigna un color especifico según la magnitud del valor del FSR para poder visualizar en que parte de la T-bar el usuario está presionando:

```
// Suscripción al tema sensors_data
var sensor_data_listener = new ROSLIB.Topic({
    ros: ros,
    name: '/sensors_data',
    messageType: 'std_msgs/String'
  });
sensor_data_listener.subscribe(function (m3) {
```

```
  var cadena = m3.data,
  separador = ",",
  arregloDeSubCadenas = cadena.split(separador);

 var id = 'cuadro-1,cuadro-2,cuadro-3,cuadro-4,cuadro-5,cuadro-6,cuadro-
 7,cuadro-8,cuadro-9,cuadro-10',
 arreglo_id = id.split(separador);

 for (var i = 0; i <= arregloDeSubCadenas.length; i++) {
  var dato = parseFloat(arregloDeSubCadenas[i]);

  // Dependiendo de presión en el FSR es el color asignado
  if (dato >= 0 && dato <= 500) {
   document.getElementById(arreglo_id[i]).style.background = "#428bca";
  } else if (dato > 500 && dato <= 1000) {
   document.getElementById(arreglo_id[i]).style.background = "#6396D8";
  } else if (dato > 1000 && dato <= 1500) {
   document.getElementById(arreglo_id[i]).style.background = "#63D8B9";
  } else if (dato > 1500 && dato <= 2000) {
   document.getElementById(arreglo_id[i]).style.background = "#82D863";
  } else if (dato > 2000 && dato <= 2500) {
   document.getElementById(arreglo_id[i]).style.background = "#D3D863";
  }
 }
});
```

5. Visualización del tiempo por test: Las siguientes líneas de código permitieron visualizar el tiempo de cada Test de la forma expuesta en la Figura 3.7.

```
<div class="col-md-2">
      <h3 class="display_4" style="padding-top: 1.5em; color: #123456;">Time:
      <span id="tiempoo"></span></h3>
</div>
```

Las instrucciones de como obtener el tiempo de ROS se hacen la sección 3.1.1.2, pero las siguientes lineal lo complementan y ayudan a colocar una restricción de tiempo por cada test:

```
// Suscripción al tema time_qolo
var tiempo_q = new ROSLIB.Topic({
    ros: ros,
    name: '/time_qolo',
    messageType: 'std_msgs/String'
  });

  tiempo_q.subscribe(function (m7) {
    // Obtención de tiempo actual de ROS
    tiempo_actual = m7.data;
    if (tiempo_inicio != 0) {
            // Restricción del tiempo por test
            if ((15 - (tiempo_actual - tiempo_inicio)) > 0) {

              document.getElementById("tiempoo").innerHTML=Math.round((ti
              empo_actual - tiempo_inicio));

              } }  });
```

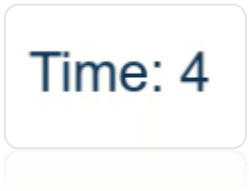


Figura 3.7. Visualización del tiempo de cada test en la aplicación web.

Fuente: Autor.

6. Indicación de test a realizar: Las siguientes líneas de código permitieron visualizar como el usuario indicaría el test a realizar. La visualización quedo de la forma expuesta en la Figura 3.8.

```
// RadioButton para seleccionar el test a realizar
```

```
<form>
        // Test 1 en el RadioButton
        <div class="form-check">
         <label class="form-check-label">
             <input type="radio" class="form-check-input" name="optradio"
             value="1"> 1
         </label>
        </div>

        // Test 2 en el RadioButton
        <div class="form-check">
         <label class="form-check-label">
              <input type="radio" class="form-check-input" name="optradio"
             value="2"> 2
         </label>
        </div>

        // Test 3 en el RadioButton
        <div class="form-check">
         <label class="form-check-label">
             <input type="radio" class="form-check-input" name="optradio"
             value="3"> 3
         </label>
        </div>

        // Test 4 en el RadioButton
        <div class="form-check">
         <label class="form-check-label">
              <input type="radio" class="form-check-input" name="optradio"
             value="4"> 4
         </label>
        </div>

        // Test 5 en el RadioButton
```

```
            <div class="form-check" style="padding-bottom: 1em;">
              <label class="form-check-label">
                  <input type="radio" class="form-check-input" name="optradio"
                  value="5"> 5
              </label>
            </div>
    </form>
```

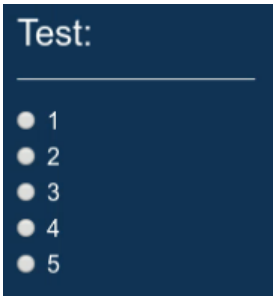


Figura 3.8. Visualización del RadioButton.
Fuente: Autor.

7. Botón Start: Las siguientes líneas de código permitieron colocar un botón *Start* que al ser presionado por el usuario indicaría el inicio de cada test:

```
<div class="col-md-1" style="padding-top: 2em;">
    // Botón Start
    <button type="button" class="btn btn-success"
    onclick="test_id_function()">Start</button>
</div>
```

8. Botón End: Las siguientes líneas de código permitieron colocar un botón *End* que indicaría el final de los 5 test propuesto, y dando paso también a la segunda página de la aplicación web donde se mostrarían los parámetros de control modificados.

```
<div class="col-md-1" style="padding-top: 2em;">
    // Botón End, y pago a la página 2 de visualización de los Betas
    <button type="button" class="btn btn-danger"
    onclick="end_function()">End</button>
```

Según los anterior, el resultado del diseño de la aplicación web fue el siguiente (Figura 3.9):

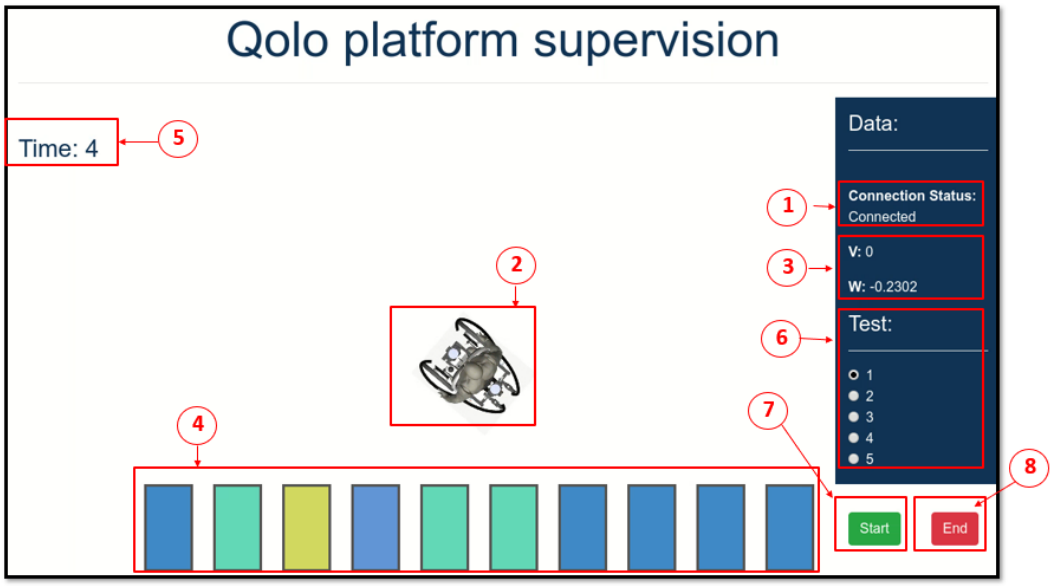


Figura 3.9. Diseño final de la aplicación web según los requisitos establecidos. Fuente: Autor.

En la figura anterior se observa, que el estado de conexión es *Conectado*, que el movimiento que se está generando en el avatar es a favor de las manecillas del reloj a los 4 segundos de comenzado el test y que esto se puede comprobar observando que el usuario está realizando el test 1, donde el usuario debe girar su cuerpo a la derecha como si fuera a girar completamente, observando también que el valor de la velocidad angular es negativo y la velocidad lineal es 0 y que la zona en donde el usuario genera mayor presión en la T-bar es acorde con dicho movimiento.

Por cada test que el usuario quiera hacer, deberá marcarlo en el RadioButton como en la imagen anterior y darle clic en *Start,* con esto, el nodo *python_server* se suscribirá a el tema *test_id* para recibir el dato del test que está realizando el usuario, e ir guardando el COP, las velocidades, las lecturas de los FSR en un archivo (que este caso es .csv) hasta cumplido el tiempo establecido para cada

test (15 segundos). Los nombres del archivo fueron de la forma: sensor_data_qolo_#.csv, donde # es el test que realizo el usuario.

Siempre y cuando se completan los cinco test propuestos, el usuario deberá darle clic al botón *End,* si esto no se cumple, la aplicación le recomendará que debe completar los cinco test. Después de guardados los test y haber presionado *End*, aparecerá la segunda página de la aplicación web llamada *qolo_2.html* y al mismo tiempo el nodo *python_server* se suscribirá el tema *end_test* para recibir un 1 (un alto) por presionar *End,* dando paso a que el tema *end_test* se encargue de hacer la calibración con la información guardada del usuario, buscando así modificar los parámetros de control. Luego de hallados los parámetros de control, estos serán guardados en el vector llamado *betas* del tema *betas_qolo* en el código *listener_qolo.py*. El nodo *rosbridge_websocket* se suscribirá a este tema (*betas_qolo)* y los mostrará en la segunda página de la aplicación web.

El siguiente esquema en la Figura 2.10 muestra de manera visual el transporte de datos explicado en el párrafo anterior:

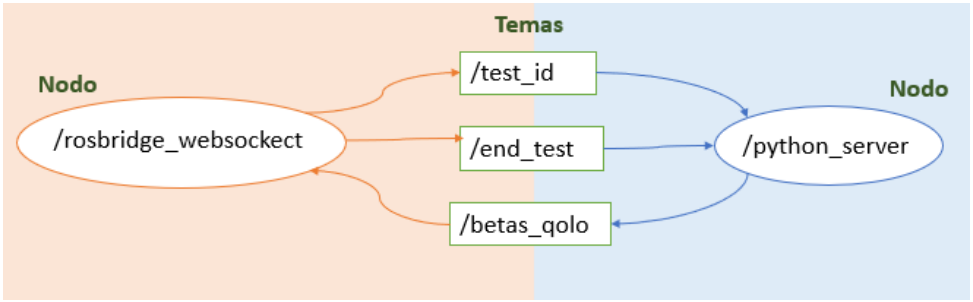


Figura 3.10. Transporte de datos para la visualización de los parámetros de calibración en la segunda página de la aplicación web. Fuente: Autor.

En la Figura 3.11 se expone el diseño de la segunda página de la aplicación web llamada *qolo_2.html*:

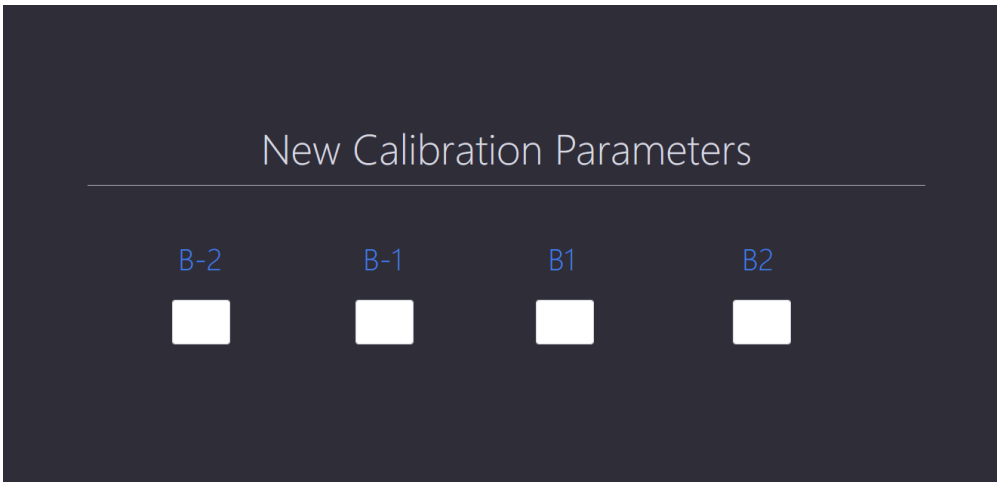


Figura 3.11. Segunda página de la aplicación web.
Fuente: Autor

Nota: La aplicación web llamada "Qolo Platform Supervision" cuenta con una guía de uso para los usuarios en inglés y español, esta guía será agregada en los Anexos I. El nombre de la guía es "Como usar la Aplicación Web "Qolo Platform Supervision"

Nota: Después de planteado el diseño y desarrollo de una aplicación web para la captura de datos, se ideó agregar la segunda página de la aplicación web (*qolo_2.html*) para obtener los parámetros de calibración nuevos. Estos parámetros nuevos se aplicarían a el código central de Qolo, y se haría una nueva captura de datos para evaluar la calibración. Lo anterior no fue posible por el tiempo reducido de la estancia investigativa, así que se planteó una evaluación de la calibración usando solo la primera página de aplicación web.

3.3 Selección de Usuarios

Qolo tiene algunas especificaciones para ser usado, y deben elegirse los potenciales usuarios según estas especificaciones y los objetivos de la investigación.

Qolo puede ser manejado por personas con un peso corporal aproximado entre 50 kilogramos y 85 kilogramos, con una altura entre 1.70 metros y 1.90 metros, y un perímetro de cadera entre 50 centímetros y 90 centímetros.

La selección de los usuarios también esta acotada según a la ubicación de la T-bar en su cuerpo, que es recomendada que esté entre el apéndice xifoide (Parte inferior del Esternón) y la última costilla flotante como se expone en la Figura 3.12. Dependiendo de esto, se obtendrá una medida específica que es la Altura entre la T-bar y el Esternón (ATE) aportada por el usuario.

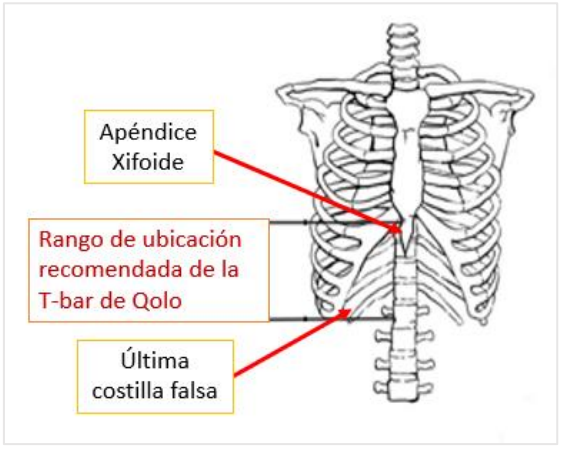


Figura 3.12. Ubicación recomendada de la T-bar de Qolo en el cuerpo humano. Fuente: Autor.

Los usuarios después de realizadas los test establecidos con la aplicación web, aportaron sus datos como, la altura, el peso y la distancia ATE. Datos que sirvieron para determinar si estas diferencias fisióloga era significativas o no a la hora de realizar la calibración.

3.4 Método de Calibración de Parámetros

Siguiendo la línea metodológica expuesta en la Figura 3.1, después de la captura de los datos con la aplicación web interactiva comenzó la etapa de procesamiento *offline* de los datos para la calibración de los parámetros de control y su posterior visualización.

3.4.1 Parámetros de Control

Como se me mencionó en la sección 2.2.31, los puntos β_{-2}, β_{-1}, β_1, β_2 en la T-bar son los parámetros de control para la calibración de Qolo. Las posibles ubicaciones de estos puntos se muestran en la Figura 2.10.

El objetivo central de esta investigación es hallar la ubicación de estos parámetros de control para cada usuario, bajo una serie de condiciones que se describirán en la siguiente sección.

Con estos parámetros se forma el vector $\boldsymbol{\beta}$ que es obtenido al final de cada calibración:

$$\boldsymbol{\beta} = [\beta_{-2} \;\; \beta_{-1} \;\; \beta_1 \;\; \beta_2]$$

Por cada vez que el usuario haya terminado los cinco test establecidos y descritos en la sección 3.1.1.3 se obtuvo un vector $\boldsymbol{\beta}$ diferente, que luego pudo ser evaluado incluyendo este vector en algunas funciones (que se describirán más adelante) del código central de Qolo y observando ahora la calibración y la diferencia de respuesta de movimiento del robot ante el movimiento del usuario.

3.4.2 Condiciones para la Calibración

La ubicación de los parámetros de calibración se hizo bajo un marco matemático adaptativo que tiene en cuenta una serie de condiciones necesarias para evitar que las ubicaciones sean incoherentes.

Las condiciones se pueden entender mejor a medida que se observa la Figura 3.13.

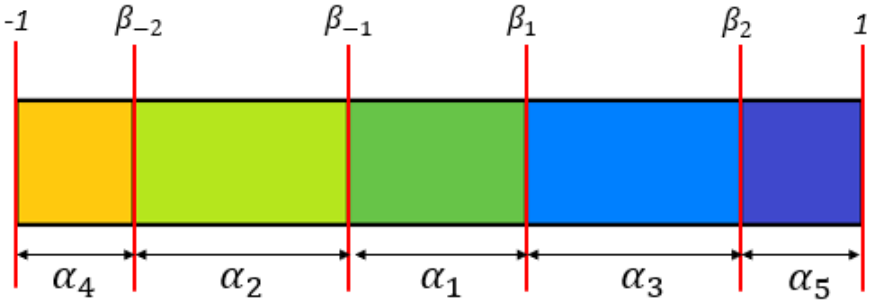


Figura 3.13. Ubicación de los parámetros de calibración y las distancias entre ellos. Fuente: Autor.

Primero se explicarán las condiciones para hallar β_{-2} y β_2 (Betas extremos) por que los β_{-1} y β_1 (Betas internos) depende de su ellos.

Los β_{-2} y β_2 son ubicados según las siguientes condiciones:

- La distancia entre -1 y β_{-2}, es decir, la distancia α_4 debería ser la mínima posible según los resultados del COP del usuario en el test 1 establecido, que dice que: El usuario debe girar su cuerpo a la derecha como si fuera a girar completamente.
- La distancia entre β_2 y 1, es decir, la distancia α_5 debería ser también la mínima posible según los resultados del COP del usuario en el test 2 establecido, que dice que: El usuario debe girar su cuerpo a la izquierda como si fuera a girar completamente.

Los β_{-1} y β_1 son ubicados según las siguientes condiciones:

- La distancia entre β_{-1} y β_1, es decir, la distancia α_1 debería ser la máxima posible según los resultados del COP del usuario en el test 3 establecido, que dice que: El usuario debe dirigir su cuerpo o inclinarse hacia adelante.
- La distancia entre β_{-2} y β_{-1}, es decir, la distancia α_2 debería ser la máxima posible según los resultados del COP del usuario en el test 3 y la ubicación ya establecidas de β_{-2}.
- La distancia entre β_1 y β_2, es decir, la distancia α_3 debería ser la máxima posible según los resultados del COP del usuario en el test 3 y la ubicación ya establecidas de β_2.

Un resumen de las anteriores condiciones se presenta en la Tabla 3.1.

Distancia	Condición
α_1	Máxima posible según el COP del Test 3
α_2	Máxima posible según el COP del Test 3 y β_{-2}

α_3	Máxima posible según el COP del Test 3 y β_2
α_3	Mínima posible según el Test 1
α_5	Mínima posible según el Test 2

Tabla 3.1. Resumen de condiciones para la ubicación de los parámetros de calibración.

3.4.3 Marco Matemático para la Calibración

La ecuación de calibración es de la forma:

$$\beta_f = \beta_i + \Delta\beta \tag{1}$$

Donde β_f es el parámetro final de la calibración, β_i es beta ideal, resultado de un COP ideal y $\Delta\beta$ es la variable adaptiva a hallar.

Para hallar $\Delta\beta$ se tomó la ecuación de la forma:

$$\Delta\beta = \frac{2}{1 + e^{-2\varepsilon}} - 1 \tag{2}$$

La anterior ecuación es conocida como la tangente hiperbólica, es una ecuación que normalmente se usa como función de activación en redes neuronales, y se escogió por la forma de la función que puede verse en la Figura 3.14, se escogió también porque esta acotada entre -1 y 1, y es de lenta convergencia.

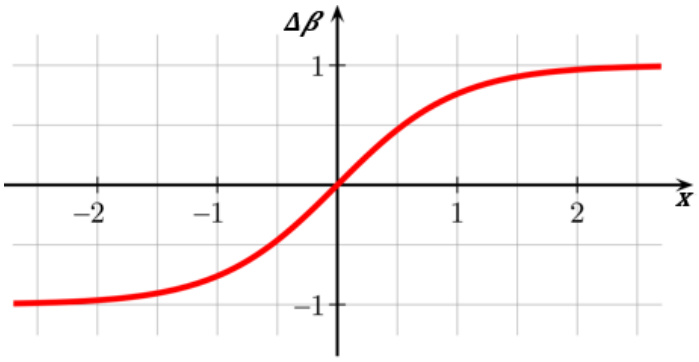


Figura 3.14. Función tangente hiperbólica. Fuente: Autor.

La variable ε en la ecuación $\Delta\beta$ es el error normalizado entre el COP obtenido en la aplicación web y el promedio del COP ideal. La explicación para la obtención de esta variable se muestra más adelante.

Según lo anterior, para continuar con la calibración de los parámetros de control se decidió partir desde una respuesta ideal del COP de los tres primeros test establecidos. Así que se plantearon tres diferentes ecuaciones.

Pero antes es bueno aclarar que:

- Para hallar β_{-2} se analiza el Test 1.
- Para hallar β_2 se analiza el Test 2.
- Para hallar β_{-1} y β_1 se analiza el Test 3.

La respuesta de la ecuación ideal planteada para hallar β_{-2}, tiene la siguiente forma:

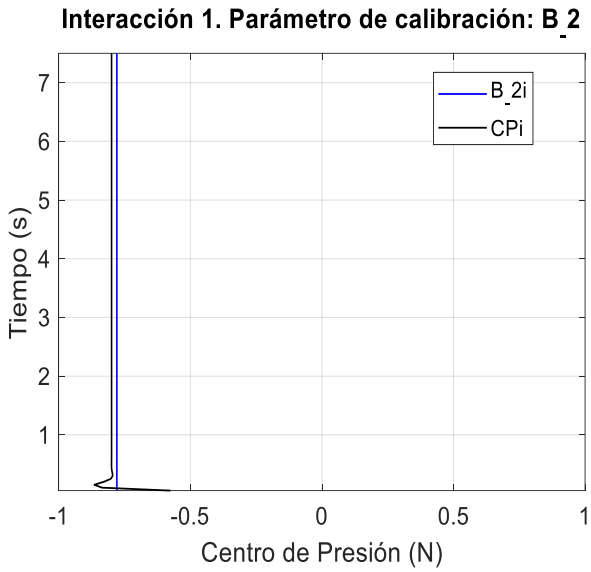


Figura 3.15. Respuesta de la ecuación del COP ideal para el Test 1.

La línea continua de color negro en la Figura 3.15 representa el COP ideal para β_{-2}, la cual tiene esta forma, pensando en que el usuario al principio del test podría tener algún rango de error.

La ecuación del COP ideal fue inspirada en la respuesta de una oscilación sub-amortiguada, que tiene la siguiente ecuación y condición:

$$\boldsymbol{G(t)} = A\, e^{-\gamma t} \cos(\omega_o t) \tag{3}$$

$$Condición: \gamma < \omega_0$$

Donde A es la amplitud máxima, y el índice de amortiguamiento (γ) es menor a la frecuencia natural (ω_0).

Con lo anterior, la ecuación de COP ideal para β_{-2} quedó de la siguiente manera:

$$\boldsymbol{CP_i\, \beta_{-2}} = A - A\, e^{-\gamma t} \cos(\omega_0 t) \tag{4}$$

Para garantizar que la respuesta sería similar a una oscilación sub-amortiguada, se cumplió la condición $\gamma < \omega_0$. Por lo tanto, los valores de las variables constantes fueron:

$$\gamma = 0.8, \qquad \omega_o = 0.9, \qquad A = -0.8$$

La línea vertical de color azul en la Figura 3.15 es el parámetro $\beta_{-2}i$ (i, ideal). Que fue hallado según el promedio y la desviación estándar del $\boldsymbol{CP_i\,\beta_{-2}}$.

La ecuación del promedio es:

$$\bar{x}(\boldsymbol{CP_i\,\beta_{-2}}) = \frac{\sum_{j=1}^{n} \boldsymbol{CP_i\,\beta_{-2}}j}{n} \tag{5}$$

La ecuación de la desviación estándar es:

$$\sigma(\boldsymbol{CP_i\,\beta_{-2}}) = \sqrt{\frac{\sum_{j=1}^{n}(\boldsymbol{CP_i\,\beta_{-2}}j - \bar{x}(\boldsymbol{CP_i\,\beta_{-2}}))^2}{n}} \tag{6}$$

Entonces, la ecuación de $\beta_{-2}i$ es:

$$\beta_{-2}i = \bar{x}(\boldsymbol{CP_i\,\beta_{-2}}) + \sigma(\boldsymbol{CP_i\,\beta_{-2}}) \tag{7}$$

La respuesta de la ecuación ideal planteada para hallar β_2, tiene la siguiente forma:

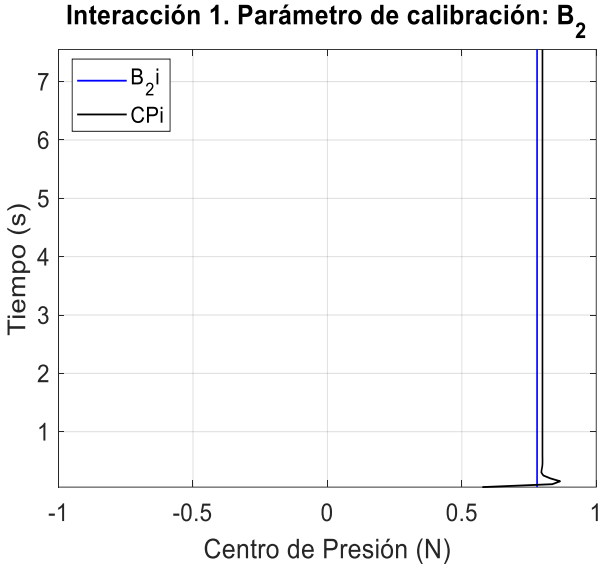


Figura 3.16. Respuesta de la ecuación del COP ideal para el Test 2.

La línea continua de color negro en la Figura 3.16 representa el COP ideal para β_2, la ecuación de este COP ideal también fue inspirada en la respuesta de una oscilación sub-amortiguada dada por la ecuación (3) y su condición.

Donde el índice de amortiguamiento (γ) fue también menor a la frecuencia natural (ω_0), pero A que es la amplitud máxima es positiva en este caso.

Con lo anterior, la ecuación de COP ideal para β_2 quedó de la siguiente manera:

$$\boldsymbol{CP_i}\,\beta_2 = A - A\,e^{-\gamma t}\cos(\omega_0 t) \tag{8}$$

Donde los valores de las contantes son:

$$\gamma = 0.8, \qquad \omega_o = 0.9, \qquad A = 0.8$$

La línea vertical de color azul en la Figura 3.16 es el parámetro $\beta_2 i$ (i, ideal). Que fue hallado también según el promedio y la desviación estándar del $\boldsymbol{CP_i}\,\beta_2$.

Entonces, la ecuación de $\beta_2 i$ es:

$$\beta_{-2} i = \bar{x}(\boldsymbol{CP_i}\,\beta_2) - \sigma(\boldsymbol{CP_i}\,\beta_2) \tag{9}$$

La respuesta de la ecuación ideal planteada para el β_{-1} y β_1, tiene la siguiente forma:

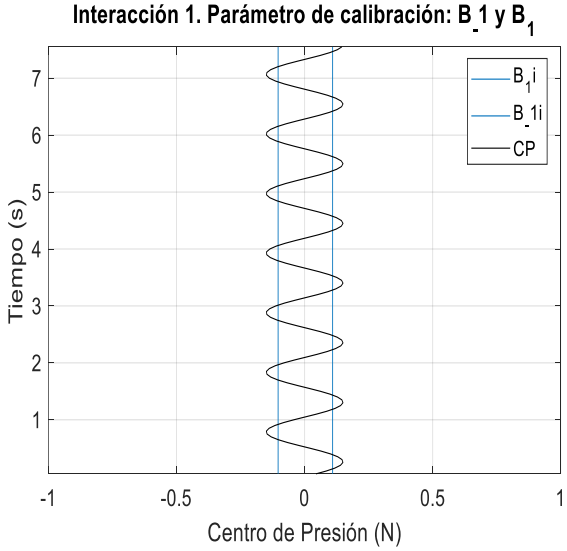


Figura 3.17. Respuesta de la ecuación del COP ideal para el Test 3.

La línea continua de color negro en la Figura 3.17 representa el COP ideal para β_{-1} y β_1, la cual tiene esta forma porque es en este test en donde el usuario ha demostrado mayor error o ruido.

La ecuación del COP ideal fue resultado de un ajuste de datos con el *toolboxs* de Matlab llamado C*urve Fitting Tool*. Se escogió un usuario que haya tenido un mínimo error en su test 3, y con estos datos se hizo el ajuste de tipo suma de senos de un solo término como se observa en la Figura 3.17. La ecuación resultante de este ajuste fue la siguiente:

$$\boldsymbol{CP_i\beta_1} = 0{,}1709sin(0{,}2558t) \tag{10}$$

Las líneas verticales de color azul en la Figura 3.17 son los parámetros $\beta_{-1}i$ y $\beta_1 i$ (i, ideal). Que fueron hallado según el promedio y la desviación estándar del $\boldsymbol{CP_i\beta_1}$.

Entonces, la ecuación de $\beta_{-1}i$ y $\beta_1 i$ es:

$$\beta_{-1}i = \bar{x}(\boldsymbol{CP_i\beta_1}) - \sigma(\boldsymbol{CP_i\beta_1}) \tag{11}$$

$$\beta_1 i = \bar{x}(\boldsymbol{CP_i\beta_1}) + \sigma(\boldsymbol{CP_i\beta_1}) \tag{12}$$

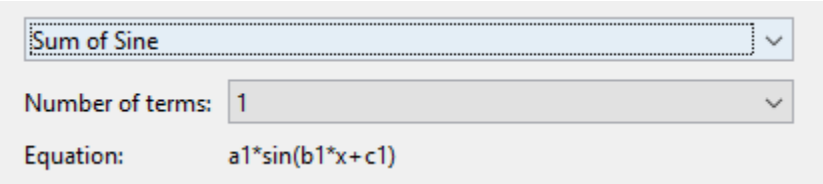


Figura 3.18. Ajuste de datos con Curve Fitting Toolbox para hallar COP ideal.

Después de que fueron halladas las ecuaciones de los COP ideales por test, se hallaron los β_{-2} y β_2 (Betas extremos). Antes de esto se necesitó el promedio y la desviación estándar de cada uno de los COP ideales obtenidos en los test, y el promedio y la desviación estándar de los COP obtenidos con la aplicación web

porque luego estos valores serían utilizados, a esta parte del marco matemático se le llamo extracción de estadística.

Para hallar β_{-2} se examinó el COP del Test 1 que es llamado ***Out_CP1.*** Sabiendo esto, se haya el error normal anteriormente mencionado entre ***Out_CP1*** y el promedio ideal, que es el promedio del COP ideal ($\boldsymbol{CP_i\,\beta_{-2}}$):

$$promi\beta_{-2} = \bar{x}(\boldsymbol{CP_i\,\beta_{-2}}) \tag{13}$$

$$errorn\beta_{-2} = \bar{x}(\boldsymbol{Out\_CP1} - promi\beta_{-2}) \tag{14}$$

Teniendo el error normal $errorn\beta_{-2}$ se halló $\Delta\beta 1$ con la ecuación (2) y después el parámetro $\beta_{-2}f$ con la ecuación (1):

$$\Delta\beta 1 = \frac{2}{1 + e^{-2\,errorn\beta_{-2}}} - 1 \tag{15}$$

$$\beta_{-2}f = \beta_{-2}i + \Delta\beta 1 \tag{16}$$

De la forma como se halló el parámetro $\beta_{-2}f$, se halla $\beta_2 f$ con las últimas 4 ecuaciones. El COP del Test 2 es llamado ***Out_CP2.*** El error normal entre ***Out_CP2*** y el promedio ideal ($\boldsymbol{CP_i\,\beta_2}$) es:

$$promi\beta_2 = \bar{x}(\boldsymbol{CP_i\,\beta_2}) \tag{17}$$

$$errorn\beta_2 = \bar{x}(\boldsymbol{Out\_CP2} - promi\beta_2) \tag{18}$$

$$\Delta\beta 2 = \frac{2}{1 + e^{-2\,errorn\beta_2}} - 1 \tag{19}$$

$$\beta_2 f = \beta_2 i + \Delta\beta 2 \tag{20}$$

De la forma con se hallaron los anteriores parámetros se cumplen las condiciones de la sección 3.1.2.2.

Los parámetros β_{-1} y β_1 por ser dependiente de los parámetros β_{-2} y β_2 y de las distancias mencionadas en la sección 3.1.2.2, se hallan de una manera diferente.

La siguiente figura puede ayudar a entender mejor la estrategia de calibración para β_{-1} y β_1:

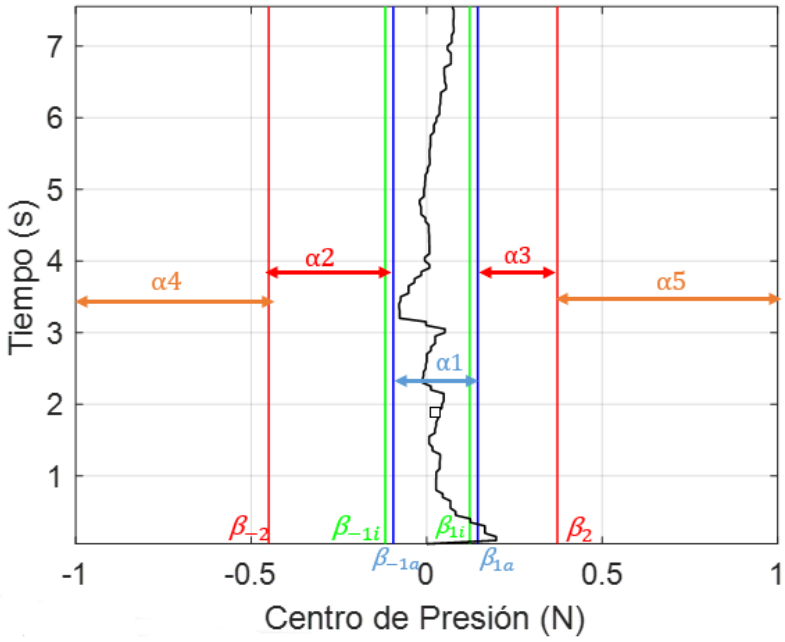


Figura 3.19. COP del Test 3 por el usuario 7. Fuente: Autor

En la figura 3.19 se pueden ver dos variables desconocidas que son $\beta_{-1}a$ y $\beta_1 a$, la letra **a** significa aproximada. Para hallar los parámetros centrales hubo dos etapas, la primera es hallar los parámetros de la forma como se hallaron $\beta_{-2}f$ y $\beta_2 f$, y estos valores resultantes serían los $\beta_{-1}a$ y $\beta_1 a$. La segunda etapa, tiene en cuentas las condiciones establecidas en la sección 3.1.2.2.

Se hallaron las $\beta_{-1}a$ y $\beta_1 a$ de la siguiente manera:

$$promi\beta_1 = \bar{x}(\boldsymbol{CP_i\,\beta_1}) \tag{21}$$

Ya que el $\boldsymbol{CP_i\,\beta_1}$ fue el mismo tanto para $\beta_{-1}a$ y $\beta_1 a$, entonces el $promi\beta_1$ también es el mismo para ambas betas aproximadas. El COP del Test 3 es llamado ***Out_CP3.***

$$errorn\beta_1 = \bar{x}(\boldsymbol{Out\_CP3} - promi\beta_1) \tag{22}$$

El error normal $errornβ_1$ fue el mismo tanto para $β_{-1}a$ y $β_1a$, por que el ***Out_CP3*** y el $promiβ_1$ también son el mismo para ambas betas aproximadas.

$$Δβ3 = \frac{2}{1 + e^{-2\ errornβ_1}} - 1 \tag{23}$$

La variable $Δβ3$ también fue la misma para ambas betas aproximadas.

$$β_{-1}a\ =\ β_{-1}i +\ Δβ3 \tag{24}$$

$$β_1a =\ β_1i +\ Δβ3 \tag{25}$$

Como los parámetros ideales centrales son diferentes entonces se obtuvieron los dos parámetros aproximados diferentes.

Siguiendo entonces las condiciones ya establecidas, se plantearon las siguientes ecuaciones para hallar $β_{-1}f$ y $β_1f$:

$$β_{-1}f = β_{-1}i + C1\ Δβ3 \tag{26}$$

$$β_1f = β_1i + C2\ Δβ3 \tag{27}$$

Las dos variables desconocidas con *C1* y *C2*, estas se definieron como factores de amplitud, las cuales afectaron directamente el valor de la variable adaptativa *Δβ3* para calibrar las betas en cuestión.

Teniendo ya $β_{-1}i$, $β_1i$, y *Δβ3*, las únicas incógnitas faltantes fueron los factores de amplitud, los cuales de hallan de la siguiente manera:

1. Se definió un rango de valores donde estará $β_{-1}f$ y $β_1f$. El valor mínimo y máximo dependerán de los valores del promedio y desviación estándar de ***Out_CP3***, las ecuaciones utilizadas fueron (5) y (6):

   Para $β_{-1}f$:

   $$β_{-1min}f = \bar{x}(\boldsymbol{Out\_CP3}) - 2σ(\boldsymbol{Out\_CP3}) \tag{28}$$

$$\beta_{-1max}f = \bar{x}(\boldsymbol{Out\_CP3}) - \frac{1}{2}\sigma(\boldsymbol{Out\_CP3}) \quad (29)$$

El rango de posibles valores de $\beta_{-1}f$ están entre $\beta_{-1min}f$ y $\beta_{-1max}f$.

Para $\beta_1 f$:

$$\beta_{1min}f = \bar{x}(\boldsymbol{Out\_CP3}) + \frac{1}{2}\sigma(\boldsymbol{Out\_CP3} \quad (30)$$

$$\beta_{1max}f = \bar{x}(\boldsymbol{Out\_CP3}) + 2\sigma(\boldsymbol{Out\_CP3}) \quad (31)$$

El rango de posibles valores de $\beta_1 f$ están entre $\beta_{1min}f$ y $\beta_{1max}f$.

2. Tenido el rango de posibles valores de $\beta_{-1}f$ y $\beta_1 f$ se pudo hallar también un rango de posibles factores de amplitud para cada parámetro.
   Apoyándose en la ecuación (26) y (27) los rangos de valores de $\beta_{-1}f$ y $\beta_1 f$ se definieron también de la siguiente manera:

   Para $\beta_{-1}f$:

$$\beta_{-1min}f = \beta_{-1}i + C1min\ \Delta\beta 3 \quad (32)$$

$$\beta_{-1max}f = \beta_{-1}i + C1max\ \Delta\beta 3 \quad (33)$$

   Para $\beta_1 f$:

$$\beta_{1min}f = \beta_1 i + C2min\ \Delta\beta 3 \quad (34)$$

$$\beta_{1max}f = \beta_1 i + C2max\ \Delta\beta 3 \quad (35)$$

   Despejando los factores de amplitud se definieron sus rangos de posibles valores:

   Para *C1*:

$$C1min = \frac{\beta_{-1min}f - \beta_{-1}i}{\Delta\beta 3} \tag{36}$$

$$C1max = \frac{\beta_{-1max}f - \beta_{-1}i}{\Delta\beta 3} \tag{37}$$

El rango de posibles valores de *C1* están entre *C1min* y *C1max*.

Para *C2*:

$$C2min = \frac{\beta_{1min}f - \beta_{1}i}{\Delta\beta 3} \tag{38}$$

$$C2max = \frac{\beta_{1max}f - \beta_{1}i}{\Delta\beta 3} \tag{39}$$

El rango de posibles valores de *C2* están entre *C2min* y *C2max*.

3. Con los rangos de *C1 y C2* se crearon los vectores de los factores de amplitud. Los cuales ayudarían a obtener los vectores de $\beta_{-1}f$ y $\beta_{1}f$, que son también acotados por los rangos de estos hallados en el punto 1.

4. Como las condiciones están en termino de distancias, se halló la distancia entre los valores de los vectores $\beta_{-1}f$ y $\beta_{1}f$, y los parámetros $\beta_{-2}f$ y $\beta_{2}f$, las ecuaciones son:

Para $\beta_{-1}f$:

$$\boldsymbol{\alpha_m} = \boldsymbol{\beta_{-1}f} - \beta_{-2}f \tag{40}$$

La vector de distancias $\boldsymbol{\alpha_m}$ fue el encargado ser comparado para cumplir las condiciones ya establecidas y hallar un $\boldsymbol{\alpha_m}$ adecuada para hallar el factor de amplitud *C1* adecuado.

Para $\beta_{1}f$:

$$\boldsymbol{\alpha_n} = \beta_2 f - \boldsymbol{\beta_1 f} \tag{41}$$

La vector de distancias $\boldsymbol{\alpha_n}$ fue el encargado ser comparado para cumplir las condiciones ya establecidas y hallar un $\boldsymbol{\alpha_n}$ adecuada para hallar el factor de amplitud *C2* adecuado.

5. Se halló *C1* y *C2* según las distancias halladas de la siguiente forma:

   Para $\beta_{-1} f$:
   Se reemplaza la ecuación (26) en la (40) y se despeja C1:

$$\alpha_m = \beta_{-1} i + C1\, \Delta\beta 3 - \beta_{-2} f$$

$$C1 = \frac{\alpha_m + \beta_{-2} f - \beta_{-1} i}{\Delta\beta 3} \tag{42}$$

   Para $\beta_1 f$:
   Se reemplaza la ecuación (27) en la (41) y se despeja C2:

$$\alpha_n = \beta_2 f - \beta_1 i + C2\, \Delta\beta 3$$

$$C2 = \frac{\alpha_m - \beta_{-2} f + \beta_{-1} i}{\Delta\beta 3} \tag{43}$$

6. Se reemplazaron los valores de *C1*, *C2,* $\beta_{-1} i$, $\beta_1 i$, y $\Delta\beta 3$. en las ecuaciones (26) y (27) para hallar $\beta_{-1} f$ y $\beta_1 f$.

Con los cálculos anterior se pudo hallar el vector de los parámetros de calibración $\boldsymbol{\beta}$.

Nota: El código de calibración será agregado a los Anexos II.

3.4.4 Correlaciones

Después de obtenidos los parámetros de calibración de todos los test y de los 10 usuarios que interactuaron con la aplicación web, se observó cual es la relación entre las diferencias fisiológicas y la calibración.

Para lo anterior, se hizo la correlación con la Matriz de Correlación de Pearson en Matlab entre los datos aportados por el usuario que fueron la altura, el peso, la ATE, el índice de masa corporal (IMC) hallado con la altura y peso, y los valores de los parámetros de calibración obtenidos en la calibración.

Para hallar el IMC se usó la siguiente ecuación:

$$IMC = \frac{Peso\ corporal}{Altutra^2} \tag{44}$$

Las siguientes líneas fueron las usadas para obtener la matriz de correlación de Pearson en Matlab:

```
Matrizco11 = [height weight IMC11' THS B2 B1 B_1 B_2];
Pearson11 = corrcoef(Matrizco11);
```

La función *corrcoef* toma los valores de la matriz *Matrizco11*, la cual fue creada para ordenar la información a correlacionar

3.5 Evaluación de la Calibración

Para la evaluación de la calibración se planteó visualizar el movimiento del robot en la primera página de la aplicación web. Para lo anterior, se debe comunicar Matlab con la aplicación web para enviarle datos.

Pero antes de eso se utilizaron algunas funciones aportadas por el Dr. Diego Felipe Páez del código central de Qolo, estas funciones fueron agregadas a el código de calibración en Matlab.

Las funciones son F*SR_execution* y *FSR_output*, estas reciben el vector de parámetros de calibración, el COP y la lectura de los 10 FSR, y devuelve dos variables llamadas *Command_v* y *Command_w,* las cuales sirven para hallar las velocidades lineal y angular con unas ecuaciones también aportadas por el Dr. Páez. Luego de obtenida las velocidades se hallan las posiciones (*x, y*) y la dirección *theta.*

Con los datos anteriores, se hizo el envío de datos a la aplicación web, de la siguiente forma:

```
xy_pub = rospublisher(topic_xy,'std_msgs/String');
```

La anterior línea es un ejemplo para publicar el tema *topic_xy*, de esta manera se hizo para los demás datos enviados que son, *theta,* las velocidades lineal y angular, las lecturas de los FSR, y el tiempo. La comunicación entre el nodo de Matlab y el de aplicación web se puede ver la Figura 3.21.

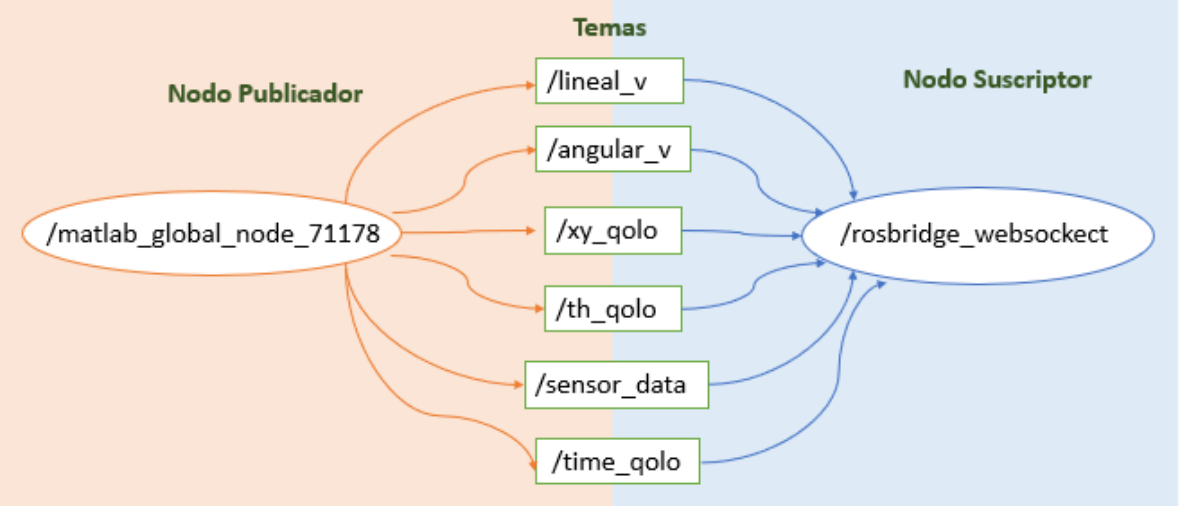


Figura 3.20. Comunicación entre los nodos de Matlab y la aplicación web, el nodo publicador y suscriptor. Fuente: Autor.

CAPÍTULO IV

Resultados

En este capítulo se exponen los resultados después de realizada la metodología descrita en el Capítulo 3, su visualiza y análisis.

4.1 Datos del Usuario

Los usuarios compartieron una información sobre su altura, peso corporal y una medida especifica llamada ATE que es la Altura entre la T-bar y el Esternón. Con esta información y la ecuación (44), se halló un cuarto dato, que es el índice de masa corporal (IMC). En la Tabla 4.1 se observa la información ordenada aportada por el usuario y el promedio de los datos entre los 10 usuarios.

Usuario	Peso (Kg)	Altura (m)	IMC (Kg/m²)	ATE (CM)
1	81	1.83	241.870	5
2	72	1.85	210.373	7
3	80	1.88	226.347	10
4	75	1.84	221.526	6
5	85	1.85	248.356	7
6	71	1.75	231.837	3
7	69	1.72	233.234	0
8	68	1.75	222.041	4
9	70	1.74	231.206	2
10	75	1.85	219.138	6
Promedio	74.6	1.806	22.859	5

Tabla 4.1. Información aportada por usuario después de la captura de datos con la aplicación web.

Para reconocer la posible relación entre los datos anteriores, se diseñaron las siguientes gráficas:

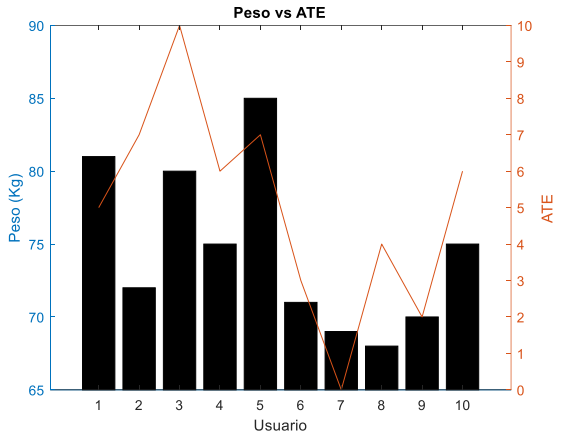


Figura 4.1. Gráfica del peso corporal de los usuarios vs la medida ATE.

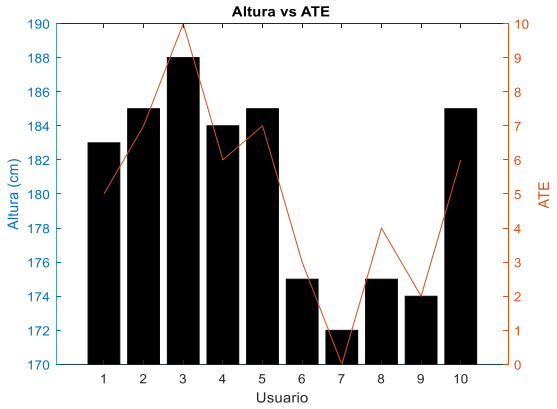


Figura 4.2. Gráfica de la altura de los usuarios vs la medida ATE.

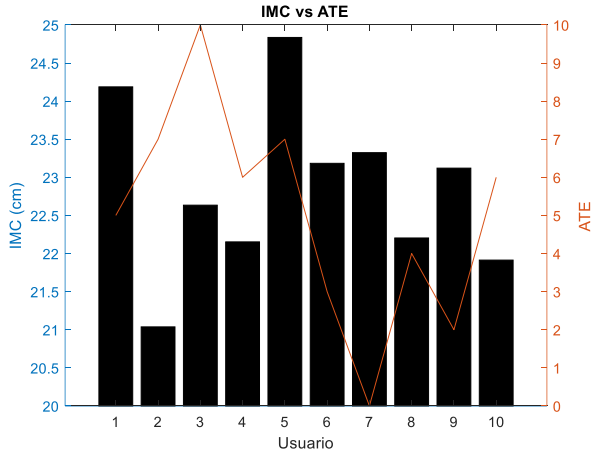


Figura 4.3. Gráfica del IMC de los usuarios vs la medida ATE.

Observando las figuras anteriores, se puede decir que no existe una relación significativa entre el peso corporal y la medida ATE, y esto mismo ocurre con el IMC y la medida ATE, mientras en la Figura 4.2, la altura si muestra alguna relación con la medida ATE. Lo anterior es justificado por que ambos datos son una distancia y fueron medidas en centímetros.

4.2 Visualización de los Datos Capturados

Según las secciones del capítulo anterior, la captura de los datos es una parte importante en el comienzo de la investigación, no solo por la comunicación y transporte de datos, sino también por la interacción entre el usuario y la aplicación web.

La captura de datos se hizo con 10 usuarios, los cuales realizaron las rutinas o test establecidos en la sección 3.1.1.3. Los test fueron guardados en archivos .csv de la forma como se observa en la Figura 3.21. Los usuarios completaron los 5 test fijados 2 veces, esto quiere decir que interactuaron en dos tiempos diferentes con la aplicación web, de los cuales se obtuvo en cada interacción un vector de calibración diferente listo para su evaluación.

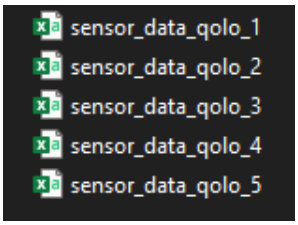


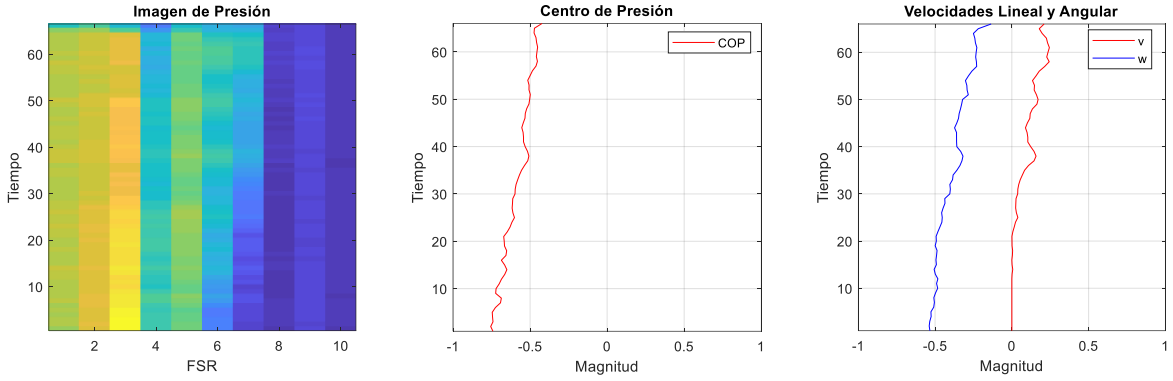
Figura 4.4. Nombres y forma en cómo se guardan los datos capturados.

Fuente: Autor.

Cada uno de los archivos capturados tiene 13 columnas, las dos primeras corresponden a la velocidad lineal y angular, la tercera al COP, y las últimas 10 corresponden a las lecturas de los FSR.

Para esta sección se tomaron 3 usuarios, cada uno con un test diferente durante su primera interacción con la aplicación web.

Las figuras siguientes muestran visualmente los dados de los archivos capturados:



a) *b)* *c)*

Figura 4.5 Datos Capturados para el usuario 1 en el test 1 en la interacción 1. a) Lecturas de fuerza en los 10 FSR para el test 1. b) Centro de Presión capturado. c) La velocidad lineal (roja) y angular (azul) capturadas.

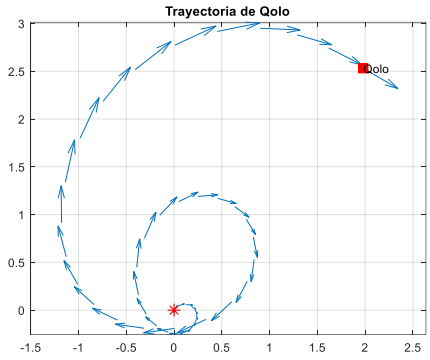
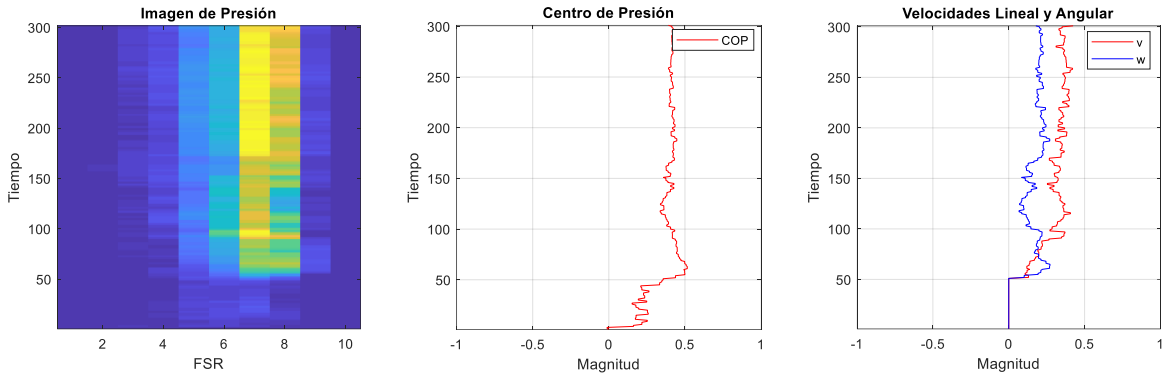


Figura 4.6. Trayectoria estimada de Qolo según el Test 1 del usuario 1.

En las figuras anteriores se muestra visualmente los datos capturados del test 1 del usuario 1 en la interacción 1, que le indica que debe girar su cuerpo a la derecha como si fuera a girar completamente, para esto, normalmente el usuario ejerce presión en los sensores de la parte izquierda de la T-bar como se muestra en la Figura 4.5 *a)*, de estas lecturas de fuerza se obtiene el COP visto en la Figura 4.5 *b)*, para que luego el robot consiguiera una velocidad lineal y angular de la forma de la Figura 4.5 *c)*. Se puede decir que, el usuario comenzó haciendo el moviendo mejor de cómo lo terminó y esto se deduce de la forma como se

manifiesta el COP en el tiempo, también de un mínimo valor de velocidad lineal al principio del movimiento y de la trayectoria vista en la Figura 4.6, donde el final del movimiento no es el esperado.



a) *b)* *c)*

Figura 4.7. Datos Capturados para el usuario 5 en el test 2 en la interacción 1. a) Lecturas de fuerza en los 10 FSR para el test 2. b) Centro de Presión capturado. c) La velocidad lineal (roja) y angular (azul) capturadas.

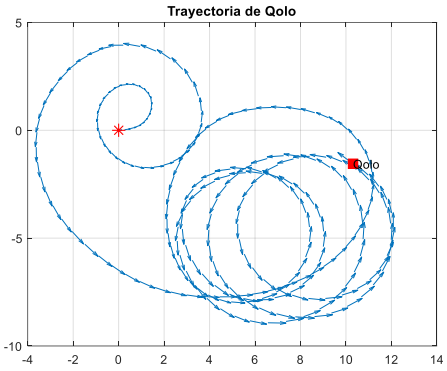
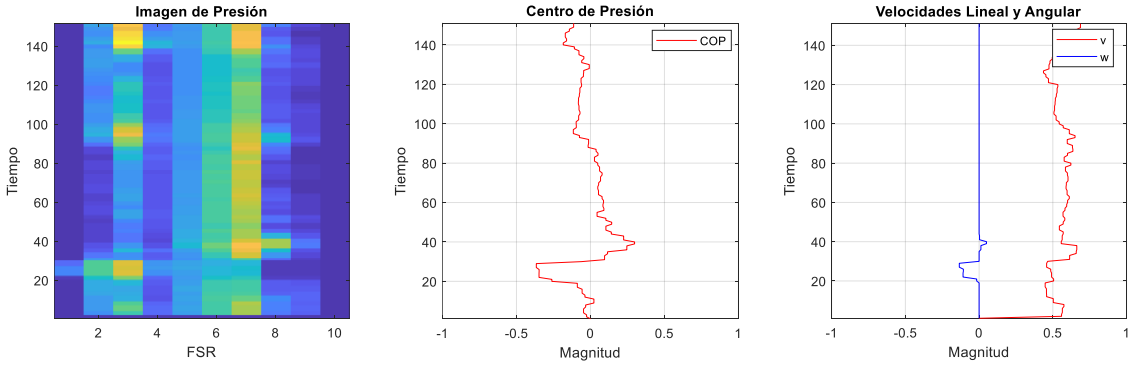


Figura 4.8. Trayectoria estimada de Qolo según el Test 2 del usuario 5.

En las figuras anteriores se muestra visualmente los datos capturados del test 2 del usuario 5 en la interacción 1, que le indica que debe girar su cuerpo a la izquierda como si fuera a girar completamente, para esto, normalmente el usuario ejerce presión en los sensores de la parte derecha de la T-bar como se muestra en la Figura 4.7 *a)*, de estas lecturas de fuerza se obtiene el COP visto en la Figura 4.7 *b)*, y luego el robot consiguió una velocidad lineal y angular de la forma de la Figura 4.7 *c)*. Se puede decir que, el usuario al inicio de su

movimiento del torso no realizó la suficiente presión para general movimiento en Qolo, y esto se puede esclarecer observando el principio de la imagen de presión y las velocidades. Ahora bien, la trayectoria estimada de Qolo en la Figura 4.7 muestra que el usuario, aunque siguió la indicación de movimiento, no generó en el robot el movimiento esperado, por lo tanto, es necesaria la calibración de los parámetros de control, que garanticen que el usuario genere el movimiento deseado sin realizar algún esfuerzo en su torso.



a) *b)* *c)*

Figura 4.9. Datos Capturados para el usuario 9 en el test 3 en la interacción 1. a) Lecturas de fuerza en los 10 FSR para el test 3. b) Centro de Presión capturado. c) La velocidad lineal (roja) y angular (azul) capturadas.

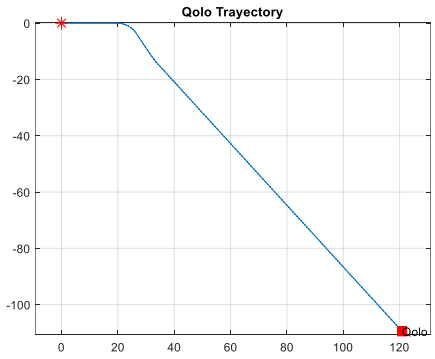


Figura 4.10. Trayectoria estimada de Qolo según el Test 3 del usuario 9.

En las figuras anteriores se muestra visualmente los datos capturados del test 3 del usuario 9 en la interacción 1, que le indica que debe dirigir su cuerpo o inclinarse hacia adelante, para esto, normalmente el usuario ejerce presión en los sensores de la parte centrales de la T-bar o ejerce presión en casi toda T-bar como se muestra en la Figura 4.9 *a)*, pero obteniendo el COP visto en la Figura

4.9 *b)*, para que luego el robot consiguiera una velocidad lineal y angular de la forma de la Figura 4.9 *c)*. Se puede decir que, según la imagen de presión, el usuario puede generar movimiento recto en el robot no solo con ejercer presión en los FSR centrales. Pueden existir factores físicos que generen este tipo de imagen de presión, que puede ser por ejemplo la medida de la cintura del usuario o ubicación de la T-bar en el cuerpo, pero este último depende de la altura del usuario.

Por lo anterior, es importante reconocer cual es la relación entre la calibración y los datos físicos del usuario. Observando el COP y las velocidades en la Figura 4.10 b*)* y *c)*, se puede entender cuál fue el momento que el usuario generó una leve inclinación y una pequeña velocidad angular positiva, para luego permanecer en línea recta y terminar el movimiento de la manera esperada.

Nota: El cálculo de la posición y dirección del robot fue hecho en Matlab con la función llamada *positionxy.m,* a partir de las velocidades lineal y angular.

4.3 Aplicación del Protocolo de Calibración

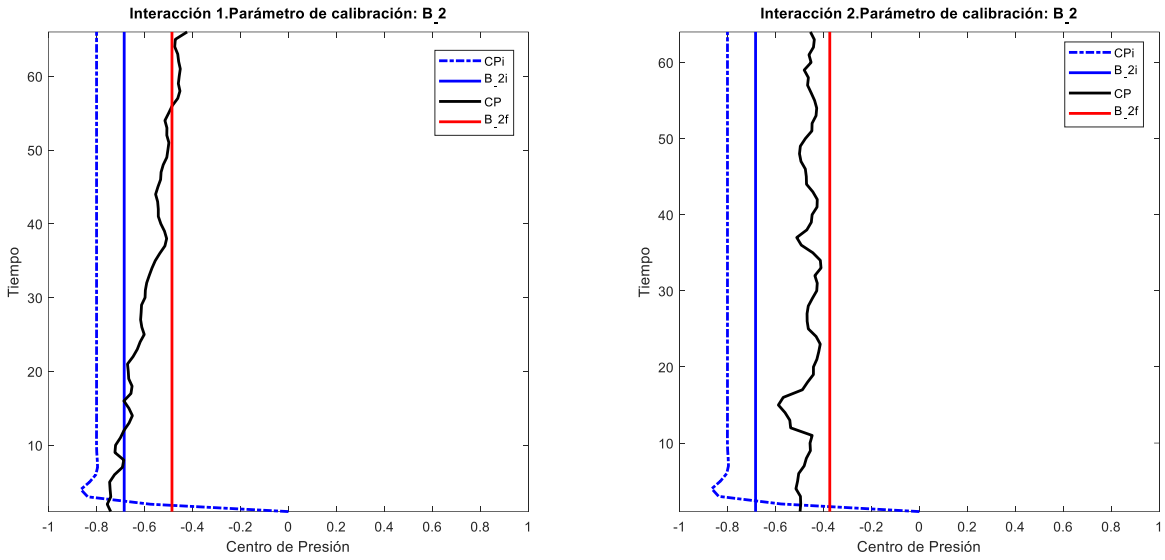
Después de capturados y visualizados los datos, se procede a aplicar el protocolo de calibración explicado en el capítulo anterior, bajo el cumplimiento de las condiciones expuestas en la sección 3.4.2. Para mostrar los resultados de la calibración se dividieron los 10 usuarios según la calibración de del parámetro de control, por lo tanto, se distribuyeron de la siguiente manera:

1. La calibración del parámetro β_{-2}: usuarios 1, 2 y 3.
2. La calibración del parámetro β_2: usuarios 4, 5 y 6.
3. La calibración del parámetro β_{-1} y β_1: usuarios 7, 8. 9 y 10.

4.3.1 Calibración del Parámetro β_{-2}

Test 1. Interacción 1 y 2:

- Usuario 1



a) *b)*

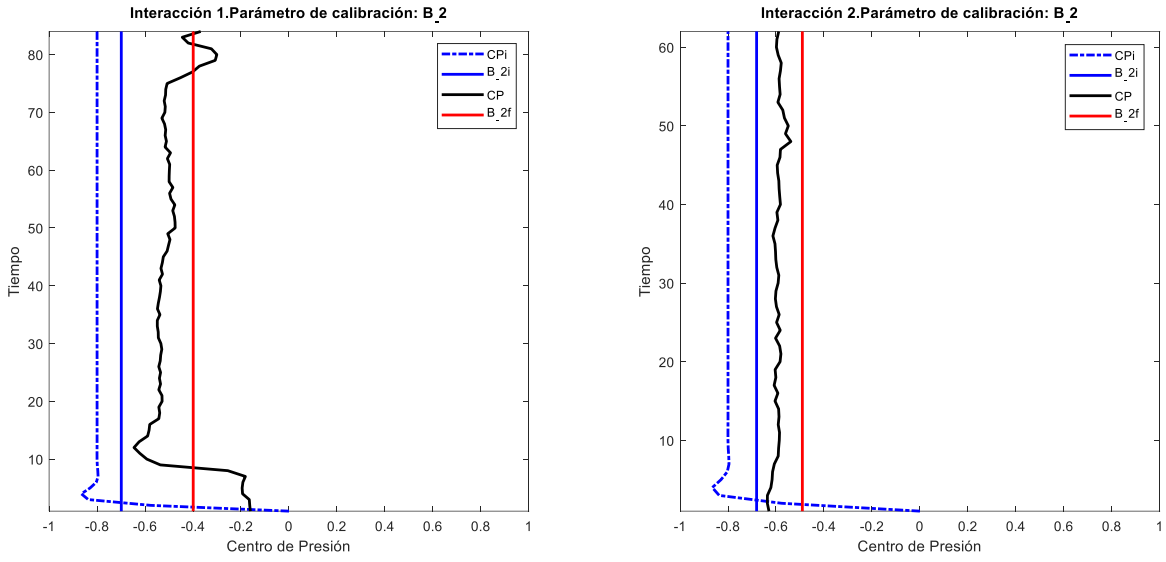
Figura 4.11. Protocolo de calibración para el parámetro β_{-2}. a) Calibración al test 1 en la primera interacción con la aplicación web del usuario 1. b) Calibración al test 1 en la segunda interacción con la aplicación web del usuario 1.

En la Figura 4.11, se muestra la calibración para obtener la ubicación del parámetro de control β_{-2} más adecuado para el usuario 1. Se tomó el test 1, el cual le indica al usuario que debe girar su cuerpo a la derecha como si fuera a girar completamente sobre sí mismo, esperando que el usuario realice el movimiento sin esfuerzo.

La primera interacción del usuario 1 con la aplicación web se muestra en la Figura 4.11 *a)*, y la segunda interacción en la Figura 4.11 *b)*. Si se comparan ambas interacciones y su calibración se puede decir que el usuario en la interacción 1 generó con su movimiento un COP (línea negra) más cercano al ideal (línea azul punteado) que la interacción 2, y esto puede se puede justificar observando que el Δβ de la ecuación (2) es menor en la interacción 1 y mayor en la interacción 2. En esta ocasión, es posible que el usuario en su segunda interacción haya realizado menos esfuerzo en su movimiento, pero cumpliendo con las indicaciones dadas antes de realizar el test, por lo tanto, el β_{-2} obtenido en la interacción 2 sería el más adecuado para el usuario 1.

Ahora bien, si se observa la calibración en ambas interacciones por individual, se logró modificar satisfactoriamente el parámetro β_{-2}, tanto que el COP se encuentra completamente a la izquierda de él.

- Usuario 2



a) *b)*

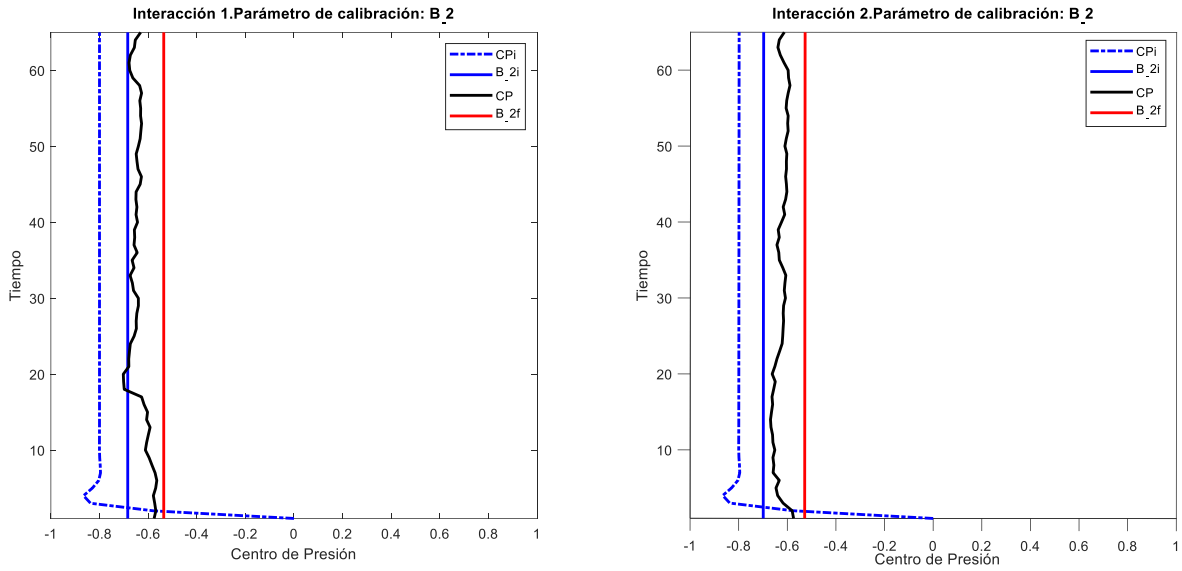
Figura 4.12. Protocolo de calibración para el parámetro β_{-2}. a) Calibración al test 1 en la primera interacción con la aplicación web del usuario 2. b) Calibración al test 1 en la segunda interacción con la aplicación web del usuario 2.

La primera interacción del usuario 2 con la aplicación web se muestra en la Figura 4.12 *a)*, y la segunda interacción en la Figura 4.12 *b)*. Si se comparan ambas interacciones y su calibración se puede decir que el usuario 2 a diferencia del usuario 1, generó con su movimiento un COP (línea negra) en la interacción 2 más cercano al ideal (línea azul punteado), por lo tanto, Δβ es menor en la interacción 2 y β_{-2} (línea roja) será ubicado más lejos que 0 o más cerca que -1.

Si se observan los COP en ambas interacciones se puede deducir que tal vez el usuario en la interacción 1 realizó menos esfuerzo en su movimiento, del cual fue producto el ruido que se observa al principio y al final de la interacción 1, ruido o error del usuario que se tuvo en cuenta cuando se planteó la ecuación que describe el COP ideal para el parámetro β_{-2}, por lo tanto, es recomendable reconocer cuando el usuario realizó algún esfuerzo en su movimiento o no, para que luego el robot se moviera como este deseaba. Entonces, reconociendo el posible esfuerzo en el usuario se puede tomar la decisión de cual parámetro β_{-2} escoger entre la interacción 1 y 2 para el usuario 2, pero por el momento el β_{-2} obtenido en la interacción 1 es el indicado.

Ahora bien, si se observa la calibración en ambas interacciones por individual, se logró modificar satisfactoriamente el parámetro β_{-2}, tanto que el COP se encuentra completamente a la izquierda del parámetro calibrado en la interacción 2 y de los valores de COP que no representan error del usuario en la interacción 1.

- Usuario 3



a) *b)*

Figura 4.13. Protocolo de calibración para el parámetro β_{-2}. a) Calibración al test 1 en la primera interacción con la aplicación web del usuario 3. b) Calibración al test 1 en la segunda interacción con la aplicación web del usuario 3.

La primera interacción del usuario 3 con la aplicación web se muestra en la Figura 4.13 *a)*, y la segunda interacción en la Figura 4.13 *b)*. Si se comparan ambas interacciones y su calibración se puede decir que tuvieron un COP (línea negra) similar, y de esto se pueden concluir dos cosas, la primera es que el usuario no realizó ningún esfuerzo en ninguna de las interacciones y este fue un movimiento natural, por lo tanto, cualquiera de los dos valores de β_{-2} es aceptable para la calibración, y la segunda es que el usuario si generó algún esfuerzo físico en ambas interacciones y que este esfuerzo aparentemente fue el mismo.

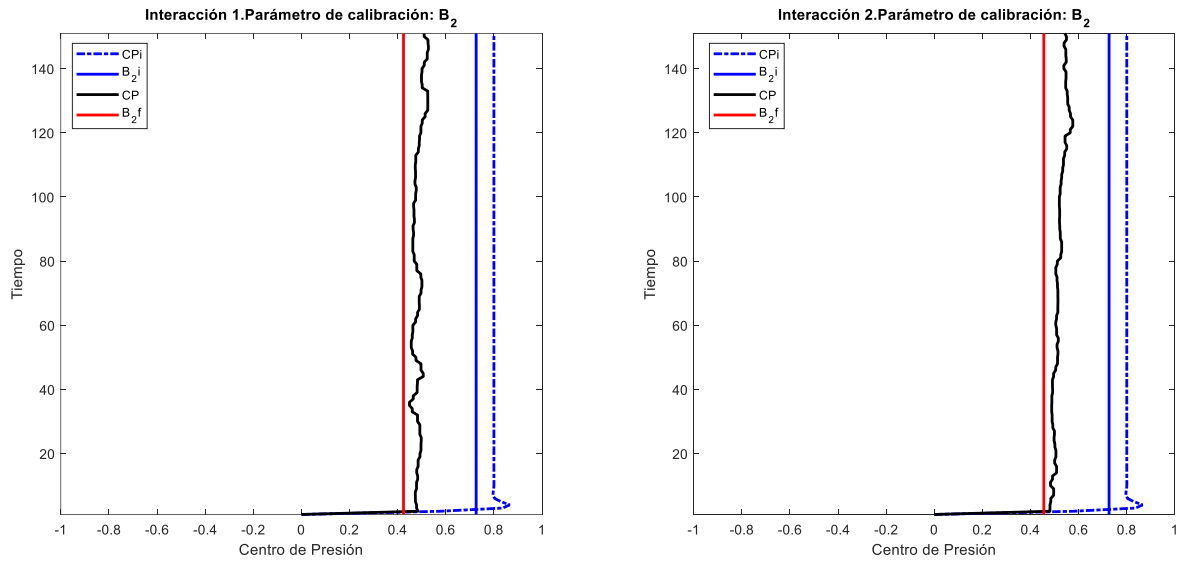
Ahora bien, si se observa la calibración en ambas interacciones por individual, se logró modificar satisfactoriamente el parámetro β_{-2}, tanto que el COP se encuentra completamente a la izquierda de él. Se puede decir también que el

usuario 3 generó un movimiento más cercano al ideal en el test 1 observando su COP que los usuarios 1 y 2.

4.3.2 Calibración del Parámetro β_2

Test 1. Interacción 1 y 2

- Usuario 4



a) *b)*

Figura 4.14. Protocolo de calibración para el parámetro β_2. a) Calibración al test 2 en la primera interacción con la aplicación web del usuario 4. b) Calibración al test 2 en la segunda interacción con la aplicación web del usuario 4.

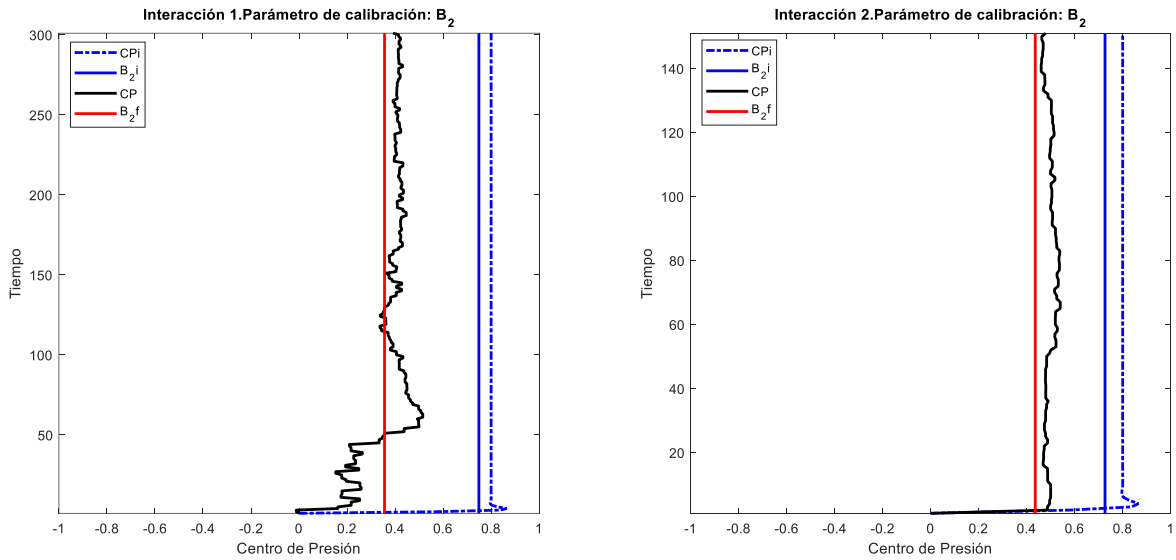
En la Figura 4.14, se muestra la calibración para obtener la ubicación del parámetro de control β_2 mas adecuado para el usuario 4. Se tomó el test 2, el cual le indica al usuario debe girar su cuerpo a la izquierda como si fuera a girar completamente sobre sí mismo, esperando que el usuario realice el movimiento sin esfuerzo.

La primera interacción del usuario 4 con la aplicación web se muestra en la Figura 4.14 *a)*, y la segunda interacción en la Figura 4.14 *b)*. Como en el caso del usuario 3 para la calibración del parámetro β_{-2}, si se comparan ambas interacciones se puede decir que tuvieron un COP (línea negra) aproximadamente similar, y que la ubicación del parámetro β_2 en ambas

interacciones tienen una pequeña diferencia no muy significativa. Con lo anterior, se puede decir que es aceptable tomar cualquiera de los dos β_2 para la calibración.

Ahora bien, si se observa la calibración en ambas interacciones por individual, se logró modificar satisfactoriamente el parámetro β_2, tanto que el COP se encuentra completamente a la derecha del parámetro calibrado.

- Usuario 5



a) *b)*

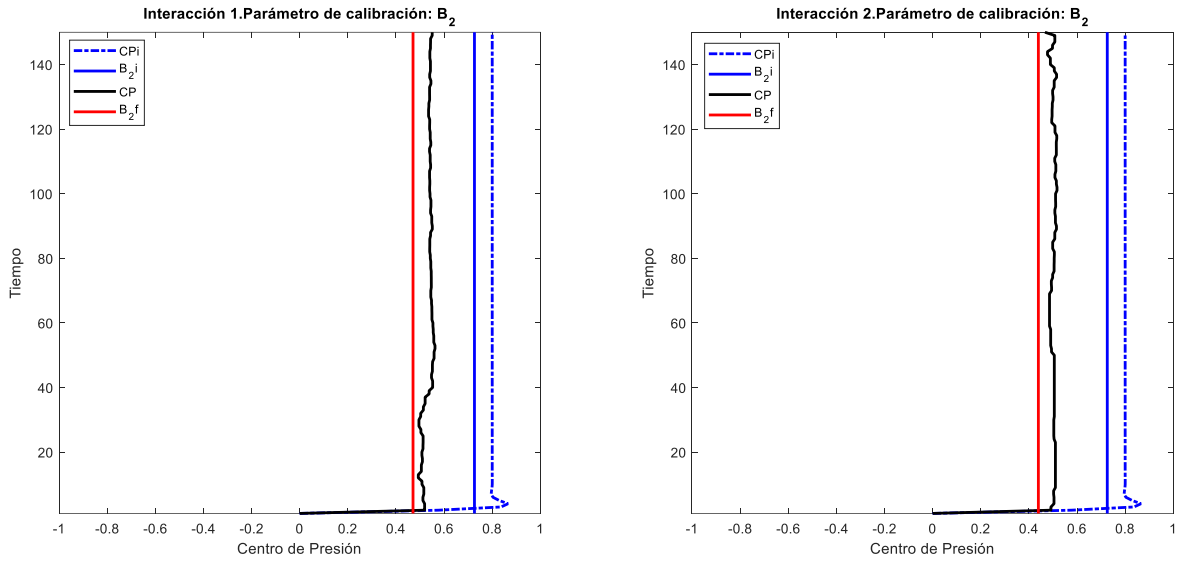
Figura 4.15. Protocolo de calibración para el parámetro β_2. *a) Calibración al test 2 en la primera interacción con la aplicación web del usuario 5. b) Calibración al test 2 en la segunda interacción con la aplicación web del usuario 5.*

La primera interacción del usuario 5 con la aplicación web se muestra en la Figura 4.15 *a)*, y la segunda interacción en la Figura 4.15 *b)*. Si se comparan ambas interacciones se puede decir que el usuario 5 en la interacción 2 generó un COP (línea negra) más cercano al COP ideal (línea azul punteado), por lo tanto, $\Delta\beta$ de la ecuación (2) es menor en la interacción 2 que en la interacción 1 y que el un movimiento que realizó el usuario 5 es más cercano al necesario para que el robot se mueva como se espera.

Si se observa la calibración en ambas interacciones por individual, se logró modificar satisfactoriamente el parámetro β_2, tanto que en la interacción 2 el

COP se encuentra completamente a la derecha del parámetro calibrado y en la interacción 1 no se tomaron lo valores que se considera ruido o error del usuario.

- Usuario 6



a) *b)*

Figura 4.16. Protocolo de calibración para el parámetro β_2. a) Calibración al test 2 en la primera interacción con la aplicación web del usuario 6. b) Calibración al test 2 en la segunda interacción con la aplicación web del usuario 6.

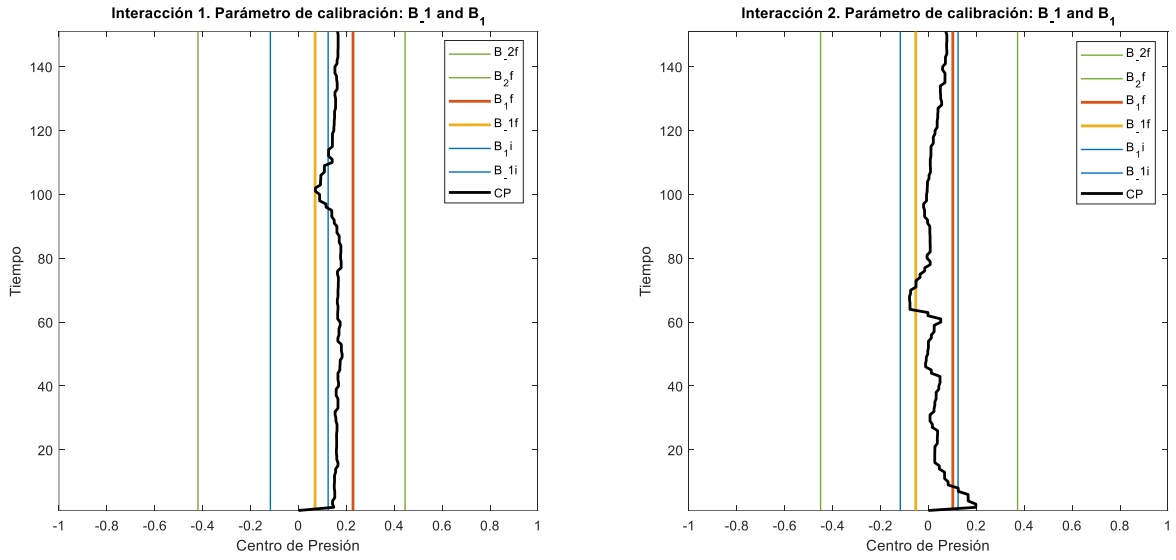
La primera interacción del usuario 6 con la aplicación web se muestra en la Figura 4.16 *a)*, y la segunda interacción en la Figura 4.16 *b)*. Como en el caso del usuario 4 para la calibración del parámetro β_2, si se comparan ambas interacciones se puede decir que tuvieron un COP (línea negra) aproximadamente similar, y que la ubicación del parámetro β_2 en ambas interacciones tienen una pequeña diferencia no muy significativa. Por lo tanto, en este caso es posible tomar cualquiera de los dos β_2 obtenidos para la calibración.

Como las calibraciones anteriores, se logró modificar satisfactoriamente la ubicación del parámetro β_2, tanto que el COP se encuentra completamente a la derecha de él.

4.3.3 Calibración del Parámetro β_{-1} y β_1

Test 1. Interacción 1 y 2

- Usuario 7



a) *b)*

Figura 4.17. Protocolo de calibración para los parámetros β_{-1} y β_1. a) Calibración al test 3 en la primera interacción con la aplicación web del usuario 7. b) Calibración al test 3 en la segunda interacción con la aplicación web del usuario 7.

Bajo el cumplimiento de las condiciones para la calibración expresadas en la sección 3.4.2 del capítulo anterior, se logró la modificación de los parámetros de control β_{-1} y β_1 dependiente de β_{-2} y β_2. En la Figura 4.17, se muestra la calibración para obtener la ubicación de los parámetros de control β_{-1} y β_1 más adecuados para el usuario 7. Se tomó el test 3, el cual le indica al usuario que debe dirigir su cuerpo o inclinarse hacia adelante, esperando que el usuario realice el movimiento sin esfuerzo.

La primera interacción del usuario 7 con la aplicación web se muestra en la Figura 4.17 *a)*, y la segunda interacción en la Figura 4.17 *b)*. Si se comparan ambas interacciones se puede decir que, en la interacción 1, el COP obtenido esta desplazado hacia la derecha, estando incluso por fuera de la ubicación de los β_{-1} y β_1 ideales (líneas azules), es decir, las lecturas de los FSR no demostraron presión en los FSR centrales como se esperaba y esto es posible según la ubicación de la T-bar en el cuerpo del usuario 7.

Este COP hace que para la calibración de los parámetros no se cumplan completamente todas las condiciones establecidas a causa del error del usuario, más sin embargo la ubicación de los parámetros β_{-1} y β_1 (línea amarilla y naranja respectivamente) acotaron al COP de la interacción 1.

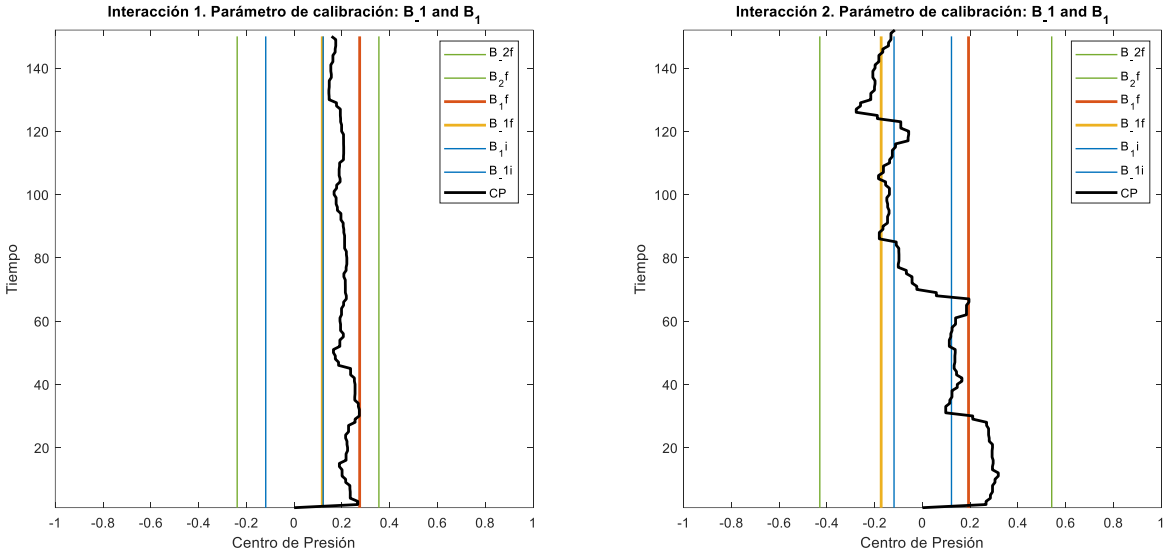
También se puede decir que, en la interacción 2 el usuario generó presión en la T-bar de modo que el COP resultante no se encontraba tan desplazado a la derecha con en el caso de la interacción 1, logrando entonces que los β_{-1} y β_1 (líneas amarilla y naranja respectivamente) obtenidos se encuentren incluso dentro de las delimitaciones de los β_{-1} y β_1 ideales (líneas azules).

- Usuario 8

La primera interacción del usuario 8 con la aplicación web se muestra en la Figura 4.18 *a)*, y la segunda interacción en la Figura 4.18 *b)*. Si se comparan ambas interacciones se puede decir que, en la interacción 1 al igual que en el usuario 7, el COP obtenido esta desplazado hacia la derecha, por lo tanto, en este usuario no se logró cumplir completamente todas las condiciones establecidas en la sección 3.4.2 a causa del error del usuario, pues incluso también se observa la proximidad de los parámetros β_{-2} y β_2 (líneas verdes) a los β_{-1} y β_1 ideales (líneas azules). Mas, sin embargo, la ubicación resultante de los parámetros β_{-1} y β_1 (líneas amarilla y naranja respectivamente) acotaron al COP de la interacción 1 de la mejor manera posible.

También se puede decir que, en la interacción 2 el usuario generó presión en la T-bar de modo que el COP resultante se encontraba fuera de los límites de los β_{-1} y β_1 ideales (líneas azules), demostrando el error del usuario a la hora de realizar el test sin esfuerzo y la necesidad de una calibración. En esta interacción el usuario 8 demostró una mejor calibración de los β_{-2} y β_2 (líneas verdes) pero una peor respuesta del COP en test 3 para hallar β_{-1} y β_1 (líneas amarilla y naranja respectivamente).

Según lo anterior, el comportamiento del usuario 8 en ambas interacciones aparenta ser muy diferentes, teniendo como resultados parámetros de calibración que no coincidían en cuanto a su ubicación en cada interacción.



a) *b)*

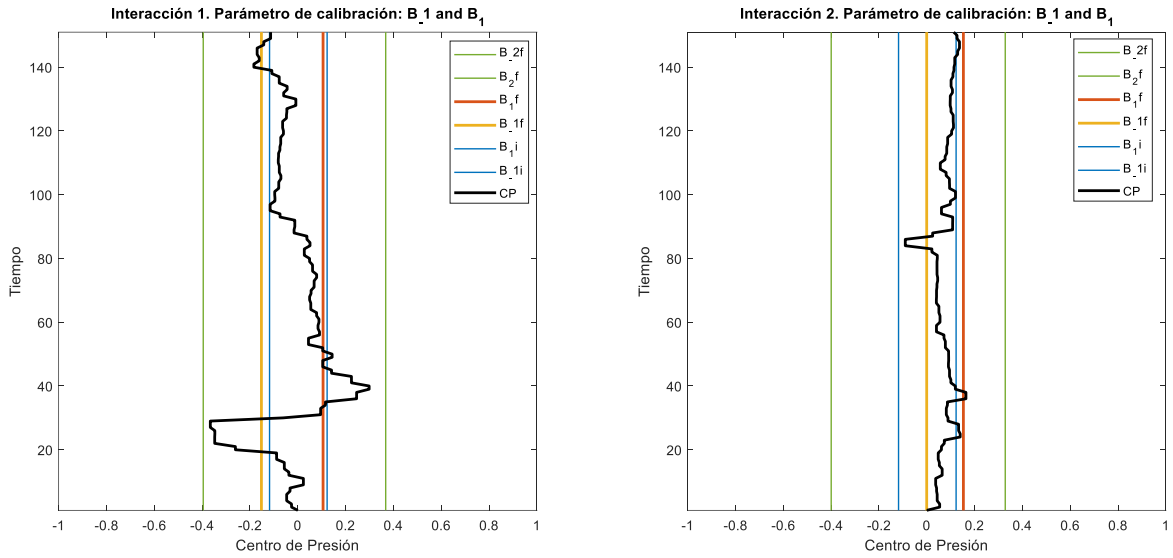
Figura 4.18. Protocolo de calibración para los parámetros β_{-1} y β_1. a) Calibración al test 3 en la primera interacción con la aplicación web del usuario 8. b) Calibración al test 3 en la segunda interacción con la aplicación web del usuario 8.

- Usuario 9

La primera interacción del usuario 9 con la aplicación web se muestra en la Figura 4.19 *a)*, y la segunda interacción en la Figura 4.19 *b)*. Si se comparan ambas interacciones se puede decir que, en la interacción 1 el usuario generó presión en la T-bar de modo que el COP resultante se encontraba incluso fuera de los límites de los β_{-1} y β_1 ideales (líneas azules) como paso con el usuario 8 en su segunda interacción con la aplicación web. Este COP demuestra entonces el error del usuario a la hora de realizar el test 3 sin esfuerzo, en donde los picos presentes al principio del COP llegan muy cerca a los β_{-2} y β_2 (líneas verdes) ya obtenidos. Mas, sin embargo, los β_{-1} y β_1 líneas amarilla y naranja respectivamente) acotan el COP que no pertenece al error del usuario.

Se puede decir también que, en la interacción 1 el COP obtenido esta desplazado hacia la derecha un poco menos que en el usuario 8 en la interacción 1, pero manteniéndose más o menos entre los límites de β_{-1} y β_1 ideales (líneas azules), por lo tanto, en este usuario se cumplió en lo posible con las condiciones establecidas en la sección 3.4.2, logrando que los parámetros β_{-1} y β_1 (líneas amarilla y naranja respectivamente) acotaran al COP de la interacción 2.

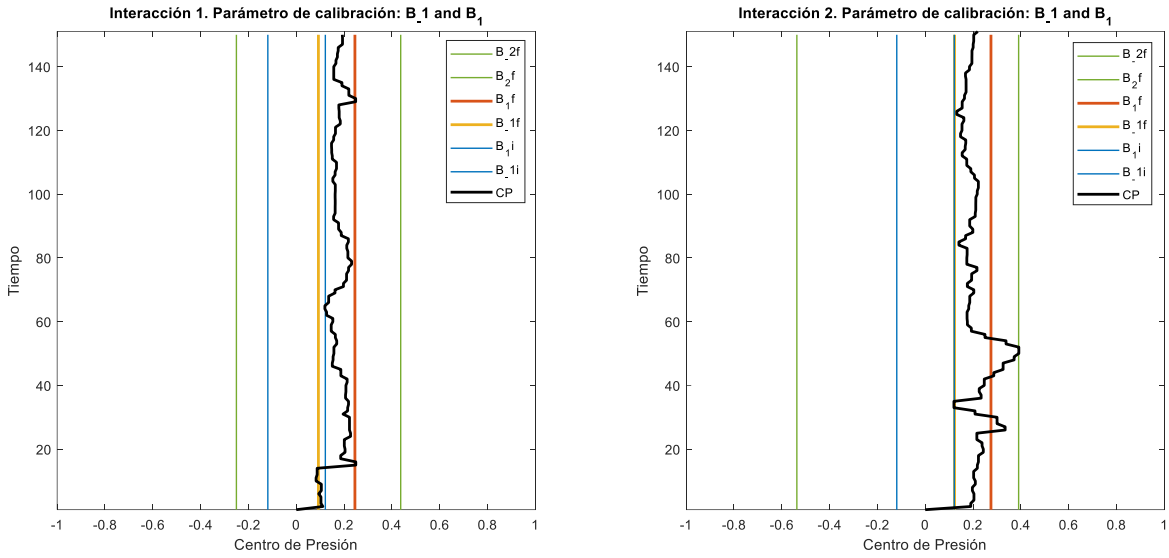
Según lo anterior, el comportamiento del usuario 9 en ambas interacciones aparenta ser muy diferentes solo cuando realiza el test 3, porque al analizar el test 1 y 2, y la ubicación de β_{-2} y β_2 (líneas verdes), estas están ubicadas similarmente en ambas interacciones.



a) *b)*

Figura 4.19. Protocolo de calibración para los parámetros β_{-1} y β_1. a) Calibración al test 3 en la primera interacción con la aplicación web del usuario 9. b) Calibración al test 3 en la segunda interacción con la aplicación web del usuario 9.

- Usuario 10



a) *b)*

Figura 4.20. Protocolo de calibración para los parámetros β_{-1} y β_1. a) Calibración al test 3 en la primera interacción con la aplicación web del usuario 10. b) Calibración al test 3 en la segunda interacción con la aplicación web del usuario 10.

La primera interacción del usuario 10 con la aplicación web se muestra en la Figura 4.20 *a)*, y la segunda interacción en la Figura 4.20 *b)*. Si se comparan ambas interacciones se puede decir que, al igual que el usuario 7 y 8 en la interacción 1, el usuario 10 generó un COP desplazado hacia la derecha en ambas interacciones, donde incluso éste se ubica por fuera de los β_{-1} y β_1 ideales (líneas azules), esto no permite cumplir todas las condiciones establecidas en la sección 3.4.2, sin embargo, en ambas interacciones la calibración de β_{-1} y β_1(líneas amarilla y naranja respectivamente) acotaran a los COP.

En el caso de la interacción 2, se encuentra mayor error del usuario y es justificada una calibración de parámetros, pues el COP incluso a acercarse a β_{-1} y esto, no es recomendado.

Según lo anterior, el comportamiento del usuario 10 en ambas interacciones aparenta ser muy diferentes solo cuando realiza el test 1 y 2, porque al analizar el test 3, y la ubicación de β_{-1} y β_1 (líneas amarilla y naranja respectivamente), estas están ubicadas similarmente en ambas interacciones.

4.4 Matriz de Correlación de Pearson

Agrupando los datos aportados por el usuario y los parámetros de calibración obtenidos en la sección anterior, se estimó la relación existente entre estos datos con la matriz de correlación de Pearson.

La representación visual de la matriz de correlación de Pearson obtenida con todos los parámetros de calibración de la primera interacción de los usuarios con la aplicación web es la siguiente:

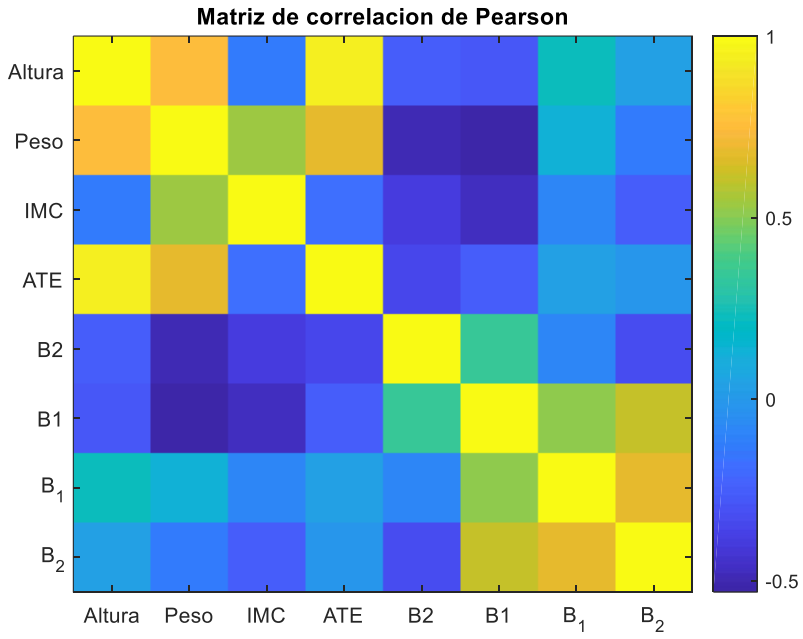


Figura 4.21. Matriz de correlación de Pearson entre lo datos de los usuarios y los parámetros de calibración obtenidos en la primera interacción con la aplicación web.

Entrelazando los datos, se observa a simple vista en la Figura 4.21 se puede decir que entre los datos de los usuarios y los parámetros de calibración obtenidos durante la interacción 1 no existe una correlación, un vínculo o una proporcionalidad significativa. Se puede decir que, si los datos de los usuarios cambiaran, los parámetros de calibración no cambiarían por ello, porque no son dependientes.

Se puede entonces verificar con la figura anterior que si existe una correlación significativa entre la altura del usuario y la medida ATE como se dijo en la sección 4.1, y una correlación más pequeña entre el peso y la medida ATE.

La representación visual de la matriz de correlación de Pearson obtenida con todos los parámetros de calibración de la segunda interacción de los usuarios con la aplicación web es la siguiente:

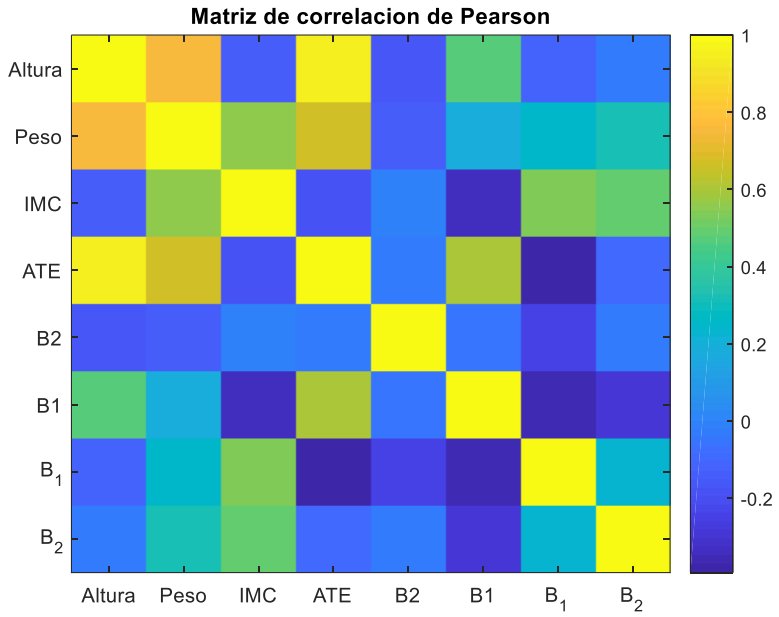


Figura 4.22. Matriz de correlación de Pearson entre lo datos de los usuarios y los parámetros de calibración obtenidos en la segunda interacción con la aplicación web.

Entrelazando los datos como en la Figura 4.21, se observa que en la Figura 4.22 existe menos correlación que entre los datos de los usuarios y los parámetros de calibración obtenidos durante la interacción 2. Incluso no se ve correlación entre los propios parámetros de calibración.

Pero en esta figura también se puede verificar que si existe una correlación significativa entre la altura del usuario y la medida ATE.

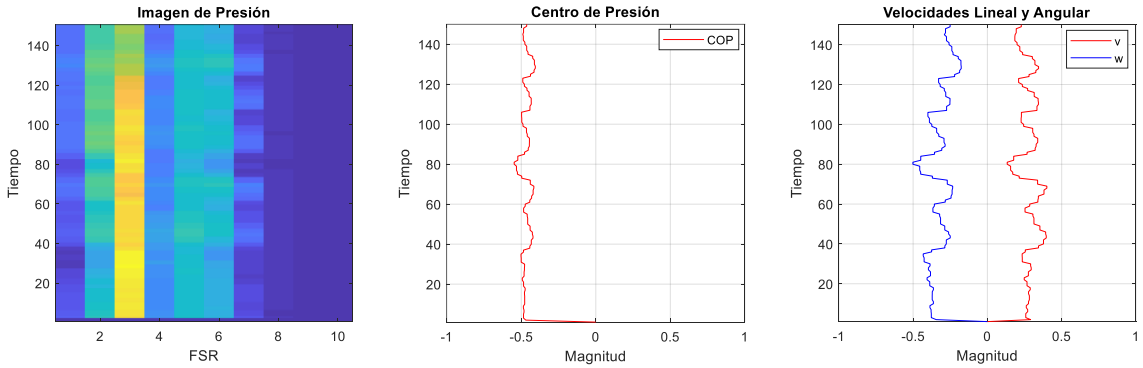
4.5 Evaluación y Simulación

Como se dijo en la sección 3.5, para la evaluación de la calibración se visualizó en forma de simulación el movimiento del robot en la primera página de la aplicación web que se puede ver en la Figura 3.9. Estas simulaciones fueron grabadas en video y serán anexadas junto con este libro.

Pero es posible reconocer la evaluación de la calibración de los parámetros de control gráficamente, utilizando las funciones mencionadas también en la sección 3.5 para obtener finalmente la posición *(x, y)* y dirección *theta*.

A continuación, se muestra la evaluación de los usuarios 4, 5 y 6:

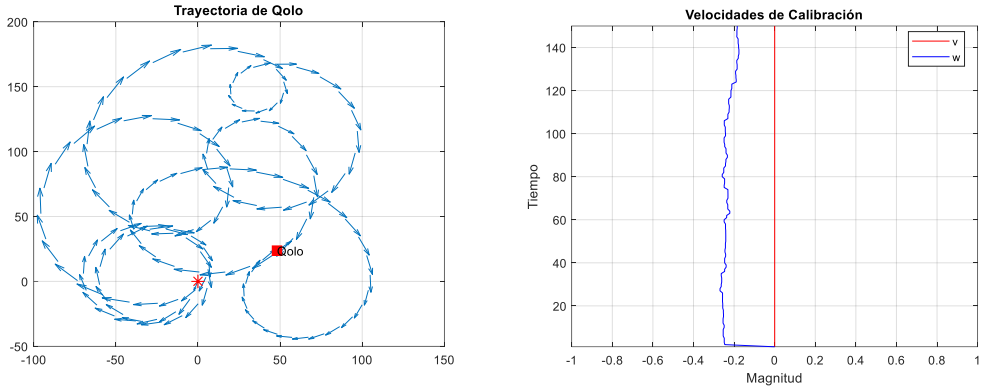
- Usuario 4:



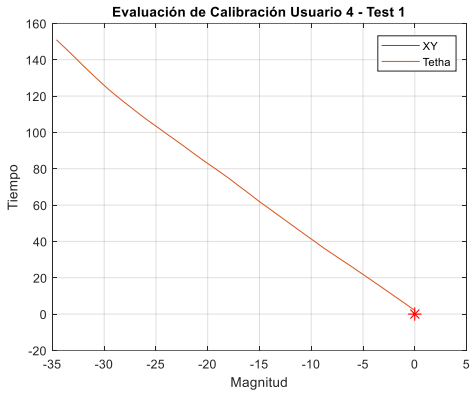
a) *b)* *c)*

Figura 4.23. Datos Capturados para el usuario 4 en el test 1 en la interacción 1. a) Lecturas de fuerza en los 10 FSR para el test 1. b) Centro de Presión capturado. c) La velocidad lineal (roja) y angular (azul) capturadas.

En la Figura 4.23 se muestran los datos obtenidos en la interacción 1 del usuario 4 haciendo el test 1. Se observa en la Figura 4.*23 a)* la zona en donde el usuario ejerce presión en la T-bar, en la Figura 4.23 *b)* al COP obtenido a partir de las lecturas de los FSR y en la Figura 4.23 *c)* la velocidad lineal (línea roja) y angular (línea roja) resultante de la interacción.



a) *b)*



c)

Figura 4.24. a) Trayectoria estimada de Qolo según el Test 1 del usuario 4. b) Velocidades obtenidas déspues de la calibración del parámetro β_{-2}. c) Evaluación de la calibración.

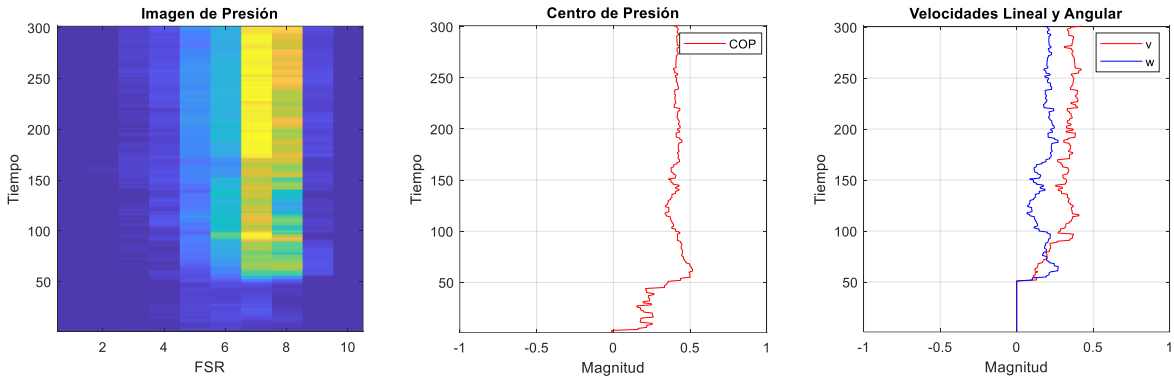
En la Figura 4.24 *a)* se observa la trayectoria estimada de Qolo según el COP capturado y las velocidades resultante en el test 1 del usuario 4, en la Figura 4.24 *b)* se grafican las velocidades resultantes después de la calibración, y en la Figura 4.24 *c)* la evaluación de la calibración del parámetro β_{-2} con la estimación de la trayectoria de Qolo según la posición y orientación obtenidas con las funciones mencionadas en la sección 3.5.

Las Figuras 4.24 a*)* y 4.24 *c)* son totalmente diferentes. Se puede observar que en la Figura 4.24 *c)* no está graficada la posición (*x,y*) de la forma como se grafica la trayectoria estimada de Qolo en la Figura 4.24 *a),* y esto es debido a que las posiciones después de la calibración dieron 0.

El valor 0 obtenido en las posiciones quiere decir que el robot no cambio de posición en ningún momento durante el test 1 realizado, es decir, después de la

calibración de los parámetros, el COP de la Figura 4.23 *b)* que fue generado por el usuario mantiene el robot en la posición 0 cumpliendo correctamente con el movimiento esperado para el test 1, además se puede observar en la Figura 4.24 *b)* que no tuvo velocidad lineal (línea roja). Con lo anterior se puede justificar que solo en la Figura 4.24 *c)* se muestra la dirección que está teniendo el robot, la cual, es congruente con la magnitud negativa de la velocidad angular en la Figura 4.24 *b)*.

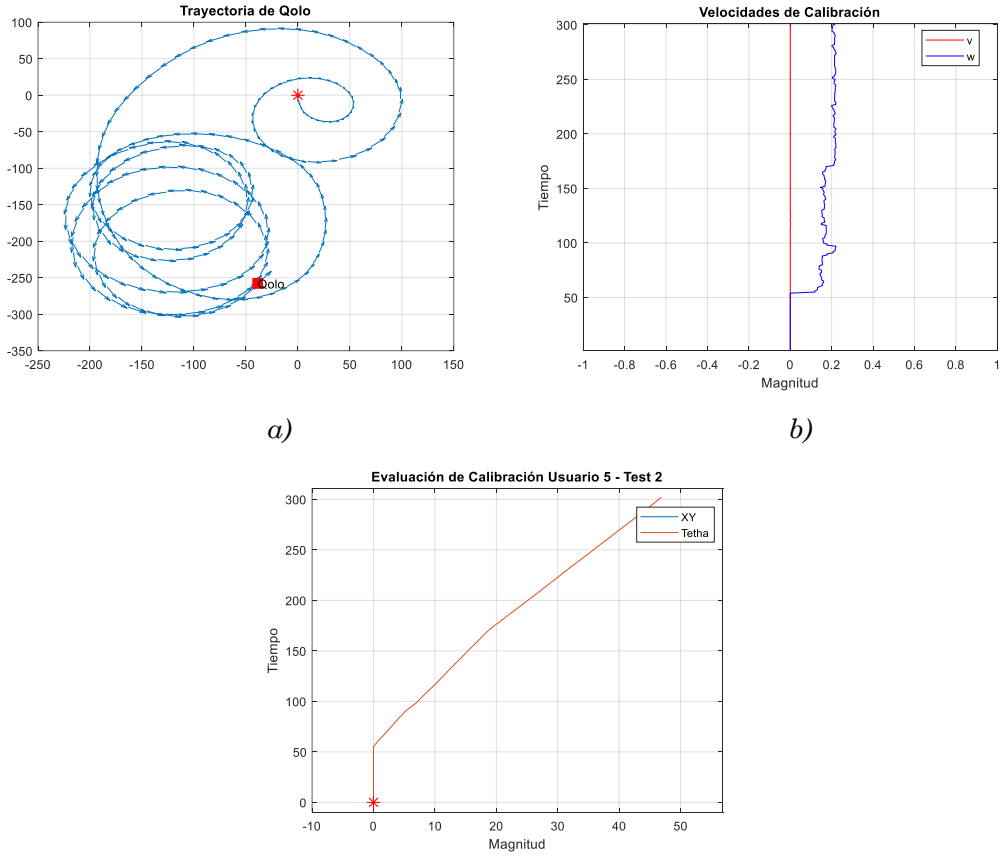
- Usuario 5:



a) *b)* *c)*

Figura 4.25. Datos Capturados para el usuario 5 en el test 2 en la interacción 1. a) Lecturas de fuerza en los 10 FSR para el test 3. b) Centro de Presión capturado. c) La velocidad lineal (roja) y angular (azul) capturadas.

En la Figura 4.25 se muestran los datos obtenidos en la interacción 1 del usuario 5 haciendo el test 2, es la misma Figura 4.7, pero se agrega nuevamente para mejor comprensión. Se observa en la Figura 4.25 *a) la* zona en donde el usuario ejerce presión en la T-bar, en la Figura 4.25 *b)* al COP obtenido a partir de las lecturas de los FSR y en la Figura 4.25 *c)* la velocidad lineal (línea roja) y angular (línea roja) resultante de la interacción.



a) *b)*

c)

Figura 4.26. a) Trayectoria estimada de Qolo según el Test 2 del usuario 5. b) Velocidades obtenidas déspues de la calibración del parámetro β_2. c) Evaluación de la calibración.

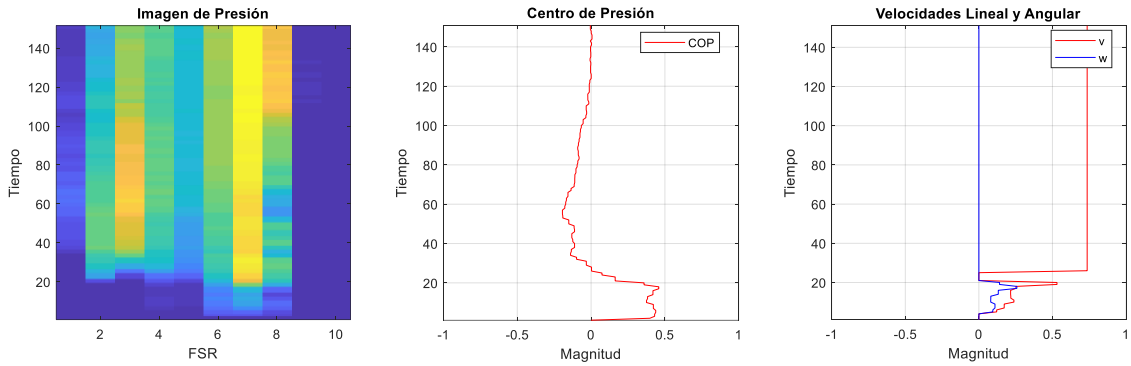
En la Figura 4.26 *a)* se observa la trayectoria estimada de Qolo según el COP capturado y las velocidades resultante en el test 2 del usuario 6, en la Figura 4.26 *b)* se grafican las velocidades resultantes después de la calibración, y en la Figura 4.26 *c)* la evaluación de la calibración del parámetro β_{-2} con la estimación de la trayectoria de Qolo según la posición y orientación obtenidas con las funciones mencionadas en la sección 3.5.

Las Figuras 4.26 a*)* y 4.26 *c)* en esta ocasión también son diferentes. Al igual que el usuario anterior, en la Figura 4.26 *c)* no se observa la gráfica de la posición (*x*,*y*) obtenida después de la calibración, y esto es debido a que las posiciones resultaron en 0.

El valor 0 obtenido en las posiciones quiere decir que el robot no se movió de su posición inicial en ningún momento durante el test 2 realizado, es decir, después de la calibración de los parámetros, el COP de la Figura 4.25 *b)* que fue

generado por el usuario mantiene el robot en la posición 0 cumpliendo correctamente con el movimiento esperado para el test 2, además se puede observar en la Figura 4.26 *b)* una velocidad lineal nula (línea roja) en todo el test, solo al principio del movimiento el usuario no generó presión en la T-bar como se puede ver en la Figura 4.25 *a)* y por esta razón la Figura 4.26 *c)* muestra un cambio de dirección significativo en el robot. Por lo tanto, en la Figura 4.26 *c)* solo se verá la dirección que está teniendo el robot después de aplicar la calibración, pues las posiciones son 0.

- Usuario 6



a) *b)* *c)*

Figura 4.27. Datos Capturados para el usuario 6 en el test 3 en la interacción 1. a) Lecturas de fuerza en los 10 FSR para el test 3. b) Centro de Presión capturado. c) La velocidad lineal (roja) y angular (azul) capturadas.

En la Figura 4.27 se muestran los datos obtenidos en la interacción 1 del usuario 6 haciendo el test 3. Se observa en la Figura 4.*27 a) la* zona en donde el usuario ejerce presión en la T-bar, en la Figura 4.27 *b)* al COP obtenido a partir de las lecturas de los FSR y en la Figura 4.27 *c)* la velocidad lineal (línea roja) y angular (línea roja) resultante de la interacción.

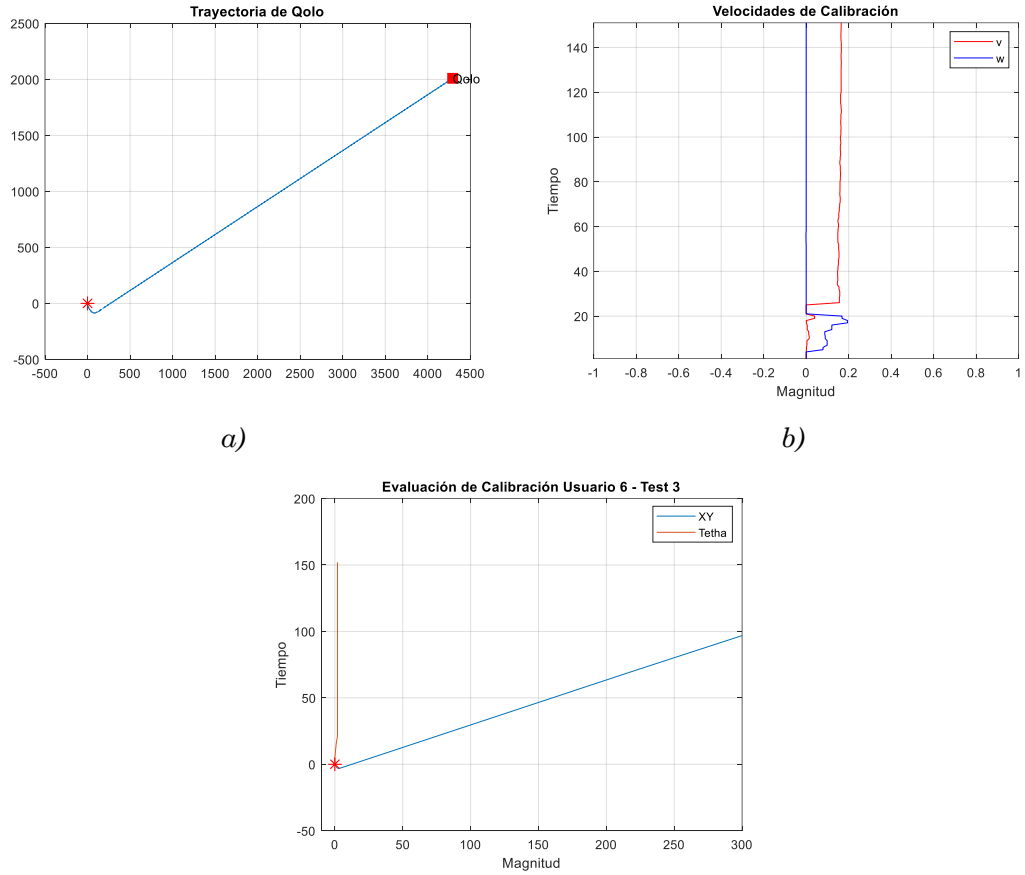


Figura 4.28. a) Trayectoria estimada de Qolo según el Test 3 del usuario 6. b) Velocidades obtenidas déspues de la calibración de los parámetrso β_{-1} y β_1. c) Evaluación de la calibración.

En la Figura 4.28 *a)* se observa la trayectoria estimada de Qolo según el COP capturado y las velocidades resultante en el test 3 del usuario 6, en la Figura 4.28 *b)* se grafican las velocidades resultantes después de la calibración, y en la Figura 4.28 *c)* la evaluación de la calibración de los parámetros β_{-1} y β_1 con la estimación de la trayectoria de Qolo según la posición y orientación obtenidas con las funciones mencionadas en la sección 3.5.

En esta ocasión las Figuras 4.28 a*)* y 4.28 *c)* son similares, pero no completamente, porque, aunque el robot si cambia de posición y tiene una velocidad lineal (línea roja) constante, las cuales se muestran en las Figuras 4.27 *c)* y 4.28 *b),* estas no son de la misma magnitud. También difieren en que, al principio del movimiento, la velocidad angular (línea azul) de la Figura 4.27 *c)* tiene una magnitud mayor a la velocidad angular de la Figura 4.27 *b),* por lo

tanto, la dirección del robot al principio del movimiento es diferente y menor en grados después de la calibración.

Conclusiones

En definitiva, se observó en la evaluación de la calibración que ésta logra personalizar el dispositivo Qolo como se había planteado desde un principio; teniendo en cuenta la codependencia entre los parámetros de control para la calibración, así pues, se logró entonces la modificación de la ubicación de los parámetros de control en la T-bar para cada usuario, permitiéndole realizar los movimientos indicados sin esforzar el movimiento de su cuerpo.

Sin embargo, en el proceso de calibración se hizo lo posible por cumplir las condiciones establecidas para cada parámetro, ya que, se manifestaron en algunos casos un posible error natural del usuario, el cual no se tiene en cuenta aun en la calibración.

Así mismo, se observó en las imágenes de presión de los datos, exactamente los obtenidos en el test 3 que el usuario realizó, que las lecturas de fuerza de los FSR no tienen que estar necesariamente ubicadas en el centro de la T-bar. Según lo anterior, esto se debe claramente al error natural del usuario, a sus diferencias fisiológicas y la motricidad residual después de la lesión de médula espinal.

Por los tanto, es recomendable realizar una revisión profunda para determinar que otros factores u otras características fisiológicas, que se presentan en los usuarios con lesión de médula espinal, podrían tener una correlación o una influencia sobre la ubicación de los parámetros de calibración en la T-bar. Pues, la altura, el peso, IMC y la medida ATE propuesta en esta investigación, demostraron que no tienen una relación significativa con los parámetros de calibración.

Por esto, es útil reconocer cuando el usuario está realmente realizando un esfuerzo físico por tratar de generar el movimiento esperado en el robot, ya sea en su torso, su cintura o en el lugar en donde se ubica la T-bar en su cuerpo.

Para finalizar, la calibración si cumple su objetivo, pero en el camino de esta investigación se reconocieron los aspectos anteriores, que aplicados la harían más optima.

Bibliografía

Adafruit. (2020). *Resistencia sensible a la fuerza (FSR)*. Obtenido de https://learn.adafruit.com/force-sensitive-resistor-fsr/overview

Ashraf S. Gorgey, R. S. (2019). 44 - Exoskeletal Assisted Rehabilitation After Spinal Cord Injury. (D. P. Editor(s): Joseph B. Webster, Ed.) *Atlas of Orthoses and Assistive Devices (Fifth Edition)*, Pages 440-447.e2. doi:https://doi.org/10.1016/B978-0-323-48323-0.00044-5.

Bionics, E. (2020). *Tratando con Ekso*. Obtenido de https://eksobionics.com/eksohealth/

C. Burbano-López, L. S. (2017). Traumatismo de la médula espinal e incertidumbre desde la teoría de Merle Mishel. *Enfermería Universitaria* , Pages 176-183.

C. Papadimitriou, J. N. (2017). The complexity of markov decision processes. *Math, 12*, pp. 441–450.

Campo, P. M. (2017). Robust model predictive control. *AmericanControl Conf. (ACC)*, p. 1021–1026.

Casey Kandilakis, P. D.-L. (2019). Exoskeletons for Personal Use After Spinal Cord Injury. *American Congress of Rehabilitation Medicine*, (In press).

Chen, H. A. (2016). A quasi-infinite horizon nonlinear model predictivecontrol scheme with guaranteed stability. *European Control Conf. (ECC)*, pp. 1421–1426.

Corporation, P. H. (2020). *Indego*. Obtenido de Conoce el exosqueleto Indego: http://www.indego.com/indego/us/en/home

D. A. Cohn, Z. G. (2016). Active learning with statistical models. *Journal of Artificial Intelligence Research, 4*, pp. 129–145.

D. J. Webb, K. L. (2014). Online parameter estimation via real-time replanning of continuous Gaussian POMDPs. *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5998-6005,. doi:doi: 10.1109/ICRA.2014.6907743.

Deepak Gopinath, S. J. (2017). Human-in-the-Loop Optimization of Shared Autonomy in Assistive Robotics. *IEEE Robotics and Automation Letters, 1*, 247-254.

Elizondo, J. J. (2012). *José Jaime Esqueda Elizondo.* Baja California. Mexico.

Esquenazi A, T. M. (2012). The ReWalk powered exoskeleton to restore ambulatory function to individuals with thoracic-level motor-complete spinal cord injury.", . *Am J Phys Med Rehabil Assoc Acad Physiatria*, 911–921.

Fetch Robotics, G. T. (2020). *Robot Web Toolas*. Obtenido de http://robotwebtools.org/index.html

Foundation, O. S. (12 de 03 de 2019). *Editores y suscriptores*. Obtenido de http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscribers

Foundation, O. S. (20 de 03 de 2019). *Múltiples Máquinas*. Obtenido de Ejecutar ROS en múltiples máquinas: http://wiki.ros.org/ROS/Tutorials/MultipleMachines#Start_the_master

Foundation, O. S. (28 de 09 de 2019). *The Standard ROS JavaScript Library*. Obtenido de roslibjs: http://wiki.ros.org/roslibjs

Foundation, O. S. (2020). *Componenetes centrales*. Obtenido de Caracteristicas espesificas del robot: https://www.ros.org/core-components/

Foundation, O. S. (2020). *Services*. Obtenido de http://wiki.ros.org/Services

Foundation, O. S. (2020). *Topis*. Obtenido de http://wiki.ros.org/Topics

Gandul, Á. Á. (2014). *Integracion de ROS con Arduino y Rasberry PI.* Sevilla-España.

García, J. M. (2016). *Diseño e implementación de una interfaz gráfica de usuario para mapeado de entornos y navegación en ROS.* Valencia.

H. Bai, D. H. (2013). Planning how to learn. *IEEE Conf. on Robotics and Automation*.

Indego. (2020). Obtenido de http://www.indego.com/indego/us/en/home

J. Van Den Berg, S. P. (2012). Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research, 31*, pp.1263-1278.

J. van den Berg, S. P. (2017). Efficient approximate value iteration for continuous gaussian pomdps. *AAAI*.

J.Windebank, N. N. (2020). Chapter 58 - Spinal cord injury. *Principles of Tissue Engineering (Fifth Edition)*, 1027 - 1091.

James E. Wilberger, G. M. (Noviembre de 2017). *MSD y los Manuales MSD*. Obtenido de https://www.msdmanuals.com/es-co/hogar/traumatismos-y-envenenamientos/lesiones-medulares/lesiones-de-la-m%C3%A9dula-espinal-y-de-las-v%C3%A9rtebras

Joseph, L. (2018). *Robot Operating System for Absolute Beginners: Robotics Programming Made Easy*. Aluva, Kerala, India: Apress.

Keith Bridwell, M. (31 de 03 de 16). *Spineuniverse*. (C. vertebral, Productor) Obtenido de https://www.spineuniverse.com/espanol/anatomia/columna-vertebral

Kouvaritakis, B. C. (2015). Model predictive control: classical, robustand stochastic. *Springer*.

L. P. Kaelbling, M. L. (2016). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research, 4*, pp. 237–285.

Ltd, R. B. (2020). *Fisioterapia asistida por robot*. (D. N. Signal, Editor) Obtenido de https://www.rexbionics.com/rex-for-clinical-use/

M. Wiering and M. van Otterlo, e. (2012). Reinforcement Learning: State of the Art. *Springer*.

M.N. Mahyuddin, S. K. (2014). A novel robust adaptive control algorithm with finite-time online parameter estimation of a humanoid robot arm. *Robotics and Autonomous Systems*, 204-305.

Marco, V. D. (15 de 11 de 2015). *Conociendo WebSocket*. Obtenido de https://v0ctor.me/websocket

Mesbah, A. (2017). Stochastic model predictive control with active uncertaintylearning: a survey on dual control. *Annu. Rev. Control*, pp. 107–117.

Mnih, V. K. (2015). Human-level control through deep reinforcement learning. *Nature*.

Mobility, T. (2019). *Qolo - Quality of life with locomotion*. Obtenido de Mobility Unlimited Challenge: https://mobilityunlimited.org/user/185

N. R. Kristensen, H. M. (2014). Parameter estimation in stochastic grey-box models. *Automatica, 40*, pp. 225–237.

Oliver Jansen, D. G. (2017). Hybrid Assistive Limb Exoskeleton HAL in the Rehabilitation of Chronic Spinal Cord. *WORLDNEUROSURGERY*.

Paez, D. K. (2018). Unpowered Lower-Body Exoskeleton with Torso Lifting Mechanism for Supporting Sit-to-Stand Transitions. *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Patrick Slade, P. C. (2017). Simultaneous active parameter estimation and control using sampling-based Bayesian reinforcement learning. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 804-810.

Patrick Slade, Z. N. (2019). Estimation and control using sampling-basedBayesian reinforcement learning. *Patrick Slade1, Zachary N. Sunberg2, Mykel J. Kochenderfer*, pp. 127-136.

Profanter, S. (2014). *Implementation and Evaluation of multimodal input/output channels for task-based industrial robot programming*.

Rebiotex. (2020). *EksoNR. Ekso Bionics*. Obtenido de https://rebiotex.com/ekso-gt/

Requez, J. P. (2020). *Matlab, EL lenguaje del cáculo técnico*. Obtenido de ¿Que es Matlab?: http://acapmi.com/blog/2017/09/18/que-es-matlab/

Research, M. F. (19 de Septiembre de 2019). *MAYOCLINIC*. Obtenido de Lesiones de la médula espinal.

Robotics, O. S. (20 de 12 de 2018). *Bibliotecas de clientes*. Obtenido de http://wiki.ros.org/Client%20Libraries

Robotics, R. (2020). *What is the ReWalk exoskeleton?* Obtenido de https://rewalk.com/

ROS, O. R. (2020). *Componentes centrales de ROS*. Obtenido de Infraestructura de comunicaciones, Caracteristicas especificas del robot y Herramientas: https://www.ros.org/core-components/

ROS, O. R. (2020). *Componentes centrales en ROS*. Obtenido de Caracteristicas especificas del robot: https://www.ros.org/core-components/

Sadowski, M. (25 de 06 de 2019). *Tutorial web de ROS parte 1 - servidor rosbridge y roslibjs*. Obtenido de https://msadowski.github.io/ros-web-tutorial-pt1/

Shervin Javdani, S. S. (2015). Shared autonomy via hindsight optimization. *Robotics: Science and Systems Conference, 11*.

Shilpa Gulati, C. J. (2009). A Framework for Planning Comfortable and Customizable Motion of an Assistive Mobile Robot. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4253-4260.

Simonetta Papa, P. E. (2020). Chapter 1 - Introduction to spinal cordinjury as clinical pathology. En F. R. Giuseppe Perale, *Spinal Cord Injury (SCI) Repair Strategies* (págs. 1-12). Woodhead Publishing.

Tsitsiklis, D. P. (2015). Neuro-dynamic programming: An overview. *Proceedings of the 34th IEEE Conference on*, 560–564.

Uğur Demir, S. K. (2016). Human impedance parameter estimation using artificial neural network for modelling physiotherapist motion. *Biocybernetics and Biomedical Engineering, 36*, Pages 318-326.

Unbehaue, N. M. (2018). Survey of adaptive dual control methods. *IEEE Proceedings-Control Theory and Applications, 147*, pp. 118–12.

Yang Chen, D. P.-G. (2019). Torso Control System with A Sensory Safety Bar for a Standing Mobility Device.

Yang Chen, D. P.-G. (s.f.). Torso Control System with A Sensory Safety Bar for a Standing Mobility Device.

YoonSeok Pyo, H. C. (2017). *ROS, Robot Programing*. Seoul, Republic of Korea : ROBOTIS Co.,Ltd.

Yosuke Eguchi, H. K. (2013). Standing Mobility Vehicle with Passive Exoskeleton. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Anexos I.

Aplicación Web Interactiva

Los anexos I corresponden a todos los códigos y guías hechas para la aplicación web interactiva.

Recolección de los datos y obtención de parámetros de control de los datos

1. Configuración para el lanzamiento de los nodos de recolección y procesamiento de los datos.

    a. Primero debemos modificar el archivo *.bashrc* de la CPU del *Qolo* fijando los parámetros relacionados con la IP a usar por el nodo maestro de ROS:

    $sudo nano .bashrc

Ir al final del archivo y agregar las siguientes líneas

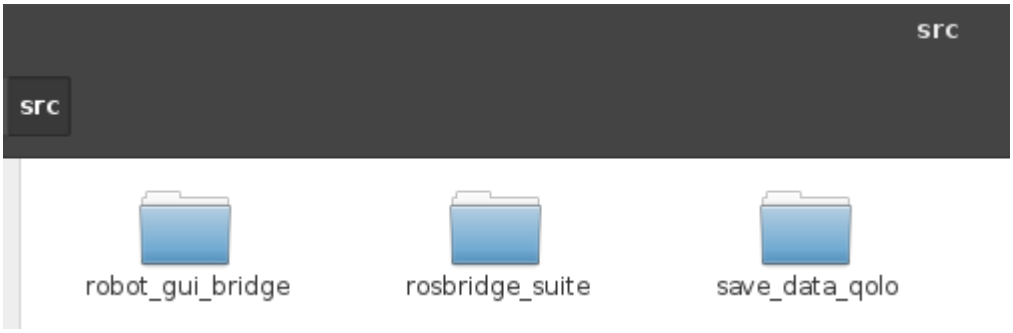
Export ROS_MASTER_URI=http://192.168.13.110:11311/
export ROS_HOSTNAME=192.168.13.110
export ROS_IP=192.168.13.110

Presionar *Ctrl + X,* Y y *Enter* para guardar y salir.

También debemos instalar unas dependencias para usar *Python3* en ROS:

$sudo apt-get install python3-pip python3-yaml
$sudo pip3 install rospkg catkin_pkg

En la carpeta src dentro del workspace de *Qolo*, pegar los archivos del fichero *files_Qolo.zip*, quedando de esta forma:



Entrar a la carpeta *robot_gui_bridge/gui* y modificar en los archivos *.html* las direcciónes IP:

- *Index.html*: página principal para la obtención y visualización de los datos de *Qolo*.

Qolo_2.html: página para la visualización de las betas calculados

Abrir el archivo *index.html* y modificar las siguientes líneas con la IP del *Qolo:*

```
241          url: 'ws://192.168.13.110:9090' // change the IP address
477            location.href = 'http://192.168.13.110:8000/qolo_2.html
```

Abrir el archivo *qolo_2.html* y modificar la siguiente línea con la IP del *Qolo*

```
54           url : 'ws://192.168.13.110:9090'
```

Entrar a la carpeta *save_data_qolo/src* y modificar el archivo *python_server.py* con la ruta completa de la carpeta donde se encuentran los archivos *.html* en *Qolo*:

```
10      web_dir = '/home/qolo/src/qolo_ros/src/robot_gui_bridge/gui
```

En una nueva terminal ir a la raiz del workspace de *Qolo*:

$cd /home/qolo/src/qolo_ros

Ejecutar los siguientes comandos:

$catkin_make
$source devel/setup.bash

Ejecutar *qolo_controlembodied.py*

$rosrun qolo qolo_controlembodied.py

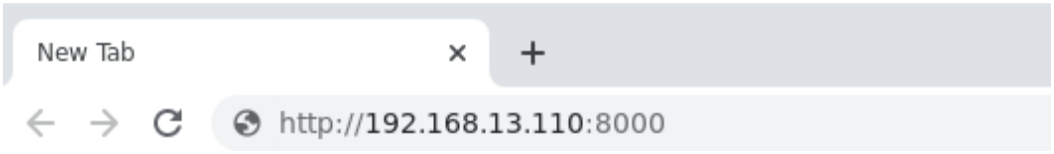
Abrir otra terminal y ejecutar de nuevo el comando:

$cd /home/qolo/src/qolo_ros
$source devel/setup.bash

Ejecutar el archivo que inicializa los nodos necesarios para el funcionamiento de la página:

$roslaunch robot_gui_bridge launch_everything.launch

Abrir el navegador, digitar la dirección IP del *Qolo* Junto con el puerto 8000:



<u>Como usar la Aplicación Web "Qolo Platform Supervision"</u>

Con esta página web se observa el movimiento que el usuario está generando en el robot con la T-bar, por medio de un avatar o robot virtual. En la barra lateral derecha azul, se muestra el estado de conexión entre el robot y la página web, las velocidades lineal *v* y angular *w* del robot.

El RadioButton se usa para indicar la prueba o test que el usuario va a realizar. Cada test tiene una duración de 15 segundos que se puede observar en la parte superior derecha de la pantalla (Time:).

Test 1: El usuario debe girar su cuerpo a la derecha como si fuera a girar completamente.

Test 2: El usuario debe girar su cuerpo a la izquierda como si fuera a girar completamente.

Test 3: El usuario debe dirigir su cuerpo o inclinarse hacia adelante.

Test 4: El usuario debe tratar de hacer un círculo uniforme hacia la derecha.

Test 5: El usuario debe tratar de hacer un círculo uniforme hacia la izquierda.

Paso a Paso:

Indicados los movimientos necesarios en cada test con anterioridad al usuario, debe seguir los siguientes pasos:

1. Indicar el test que se realizara en el RadioButton (Test 1) 

, presionar "Start", y realizar el movimiento durante los siguientes 15 segundos.
2. Repetir el paso 1 para los Test 2, 3, 4 y 5. Es recomendable realizar todos los test ascendentemente.
3. Luego de terminados todos los test, presionar "End" para indicar la terminación de la prueba completa, y luego aparecerá la siguiente página.

Nota: Si al presionar "End" y aparece un mensaje emergente, esto quiere indicar que no se realizaron todos los test acordados y deberá hacer el test faltante.

Anexos II

Protocolo de Calibración

Los anexos II corresponden a todos los códigos utilizados para el protocolo de calibración.

Código de Calibración en Matlab

```
clear, clc;

%% Archivos de estudio
%% Test 1
struct1 = importdata('sensor_data_qolo_11.csv');
struct2 = importdata('sensor_data_qolo_21.csv');
struct3 = importdata('sensor_data_qolo_31.csv');
%% Test 2
struct4 = importdata('sensor_data_qolo_12.csv');
struct5 = importdata('sensor_data_qolo_22.csv');
struct6 = importdata('sensor_data_qolo_32.csv');
%%
%  Test 1
%
%% Limpiar el Archivo 1 (B_2)
DataUser41_1 = struct1.data;
   %% Extracción de Variables Archivo 1 (B_2)
   DataSensor = DataUser41_1(:,4:13);
   Out_CP = DataUser41_1(:,3)/2.5;
   c = length(Out_CP);
   D = 0:c;
   %% Estadística Archivo 1 (B_2)
   prom = mean(Out_CP);
   stdx = std(Out_CP);
   stdm1 = prom + stdx;
   stdm_1 = prom - stdx;
```

```
    %% Ecuaciones ideal de CP (B_2)
    amp = -0.8;
    CPi_B_2 = amp - (amp*exp(-0.8*D).*cos(0.9*D));
    promiB_2 = mean(CPi_B_2);
    stdxiB_2 = std(CPi_B_2);
    stdm1iB_2 = promiB_2 + stdxiB_2;
    stdm_1iB_2 = promiB_2 - stdxiB_2;
    B_2i = promiB_2 + (stdxiB_2);
    %% Modificación (B_2)
    errormB_2 = mean(Out_CP - promiB_2);
    sigma = ((2 ./ (1 + (exp(-2*errormB_2))))-1);
    B_2f = B_2i + sigma;


%% Limpiar el Archivo 2 (B2)
DataUser41_2 = struct2.data;
    %% Extracción de Variables Archivo 2 (B2)
    DataSensor2 = DataUser41_2(:,4:13);
    Out_CP_2 = DataUser41_2(:,3)/2.5;
    c2 = length(Out_CP_2);
    D2 = 0:c2;
    %% Estadística Archivo 2 (B2)
    prom_2 = mean(Out_CP_2);
    stdx_2 = std(Out_CP_2);
    stdm1_2 = prom_2 + stdx_2;
    stdm_1_2 = prom_2 - stdx_2;
    %% Ecuaciones ideal de CP (B2)
    amp = 0.8;
    CPi_B2 = amp - (amp*exp(-0.8*D).*cos(0.9*D));
    promiB2 = mean(CPi_B2);
    stdxiB2 = std(CPi_B2);
    stdm1iB2 = promiB2 + stdxiB2;
    stdm_1iB2 = promiB2 - stdxiB2;
    B2i = promiB2 - (stdxiB2);
    %% Modificacion (B2)
    errormB2 = mean(Out_CP_2 - promiB2);
```

```
    sigma2 = ((2 ./ (1 + (exp(-2*errormB2))))-1);
    B2f = B2i + sigma2;

%% Limpiar el Archivo 3 (B_1 y B1)
DataUser41_3 = struct3.data;
    %% Extracción de Variables Archivo 3 (B_1 y B1)
    DataSensor3 = DataUser41_3(:,4:13);
    Out_CP_3 = DataUser41_3(:,3)/2.5;
    c3 = length(Out_CP_3);
    D3 = 0:c3;
    %% Estadística Archivo 3 (B_1 y B1)
    prom_3 = mean(Out_CP_3);
    stdx_3 = std(Out_CP_3);
    stdm1_3 = prom_3 + stdx_3;
    stdm_1_3 = prom_3 - stdx_3;
    %% Ecuaciones ideal de CP (B_1 y B1)
    CPiB1 = 0.15*sin(0.3*D3);
    promiB1 = mean(CPiB1);
    stdxiB1 = std(CPiB1);
    stdm1iB1 = promiB1 + stdxiB1;
    stdm_1iB1 = promiB1 - stdxiB1;
    B1i = promiB1 + (stdxiB1);
    B_1i = promiB1 - (stdxiB1);
    %% Modificación (B1 aproximada)
    errormB1 = mean(Out_CP_3 - promiB1);
    sigma3 = ((2 ./ (1 + (exp(-2*errormB1))))-1);
    B1fa = B1i + sigma3;
    %% Modificación (B_1 aproximada)
    errormB_1 = mean(Out_CP_3 - promiB1);
    sigma4 = ((2 ./ (1 + (exp(-2*errormB_1))))-1);
    B_1fa = B_1i + sigma4;
  %%
  alfa1_1 = B1fa  - B_1fa
  alfa2_1 = stdm_1_3 - B_2f
  alfa3_1 = B2f - stdm1_3
  alfa_1 = alfa3_1 - alfa2_1;
```

```
    [alfamax_1,I_1] = max(alfa_1);

B_1fmax = prom_3 - (1/2)*(stdx_3);
B_1fmin = prom_3 - 2*(stdx_3);
C_1fmax = (B_1fmax - B_1i) / sigma4;
C_1fmin = (B_1fmin - B_1i) / sigma4;
B_1fv = [B_1fmin  B_1fmax B_1fa];
C_1fv = [C_1fmin  C_1fmax];
C_1f = min(C_1fv):0.05:max(C_1fv);

B1fmax = prom_3 + (2)*(stdx_3);
B1fmin = prom_3 + (1/2)*(stdx_3);
C1fmin = (B1fmin - B1i) / sigma3;
C1fmax = (B1fmax - B1i) / sigma3;
B1fv = [B1fmin B1fmax B1fa];
C1fv = [C1fmin C1fmax];
C1f = min(C1fv):0.05:max(C1fv);

t_1 = [];t_2 = [];t_3 = [];t_4 = [];
   for i = 1:length(C1f)
      B1f(i) = B1i + (C1f(i) * sigma3);
      alfa4_1(i) = B2f - B1f(i);
      if alfa3_1 < alfa4_1(i)
        t_1(i) = alfa4_1(i);
      end
   end
   for i = 1:length(C1f)
      B1f(i) = B1i + (C1f(i) * sigma3);
      alfa4_1(i) = B2f - B1f(i);
      if alfa3_1 > alfa4_1(i)
        t_2(i) = alfa4_1(i);
      end
   end
   for i = 1:length(C_1f)
      B_1f(i) = B_1i + (C_1f(i) * sigma4);
```

```
alfa5_1(i) = B_1f(i) - B_2f;
          if alfa2_1 < alfa5_1(i)
            t_3(i) = alfa5_1(i);
          end
       end
       for i = 1:length(C_1f)
          B_1f(i) = B_1i + (C_1f(i) * sigma4);
          alfa5_1(i) = B_1f(i) - B_2f;
          if alfa2_1 > alfa5_1(i)
            t_4(i) = alfa5_1(i);
          end
       end
       t_1 = t_1(find(t_1 ~= 0));
       t_2 = t_2(find(t_2 ~= 0));
       t_3 = t_3(find(t_3 ~= 0));
       t_4 = t_4(find(t_4 ~= 0));

if alfa_1 < 0
   if length(t_4) > 0 && length(t_1) > 0
       k = 1
       t_4f = max(t_4);
       t_1f = min(t_1);
       C_1ffg = (t_4f + B_2f - B_1i) / sigma4;
       B_1f = B_1i + (C_1ffg*sigma4);
       C1ffg = (B2f - t_1f - B1i) / sigma3;
       B1f = B1i + (C1ffg*sigma3);
       kt1 = B1f - B_1f;
       if kt1 < 0.15
         i = 1;
         while (B1f - B_1f  < 0.15)
            B1f = B1f + 0.005
            B_1f = B_1f - 0.005
            i = i + 1;
         end
       end
   end
```

```
if alfa_1 > 0
   if length(t_3) > 0 && length(t_2) > 0
      k = 2
      t_3f = min(t_3);
      t_2f = max(t_2);
      C_1ffg = (t_3f + B_2f - B_1i) / sigma4;
      B_1f = B_1i + (C_1ffg*sigma4);
      C1ffg = (B2f - t_2f - B1i) / sigma3;
      B1f = B1i + (C1ffg*sigma3)
      kt1 = B1f - B_1f;
      if kt1 < 0.15
        i = 1;
        while (B1f - B_1f  < 0.15)
           B1f = B1f + 0.005
           B_1f = B_1f - 0.005
           i = i + 1;
        end
      end
   end
end
```