Centro de Ingeniería y Desarrollo Industrial

Alumno: Herbert Rafael de los Santos Bojórquez

Título del proyecto: Control de Robots Manipuladores con Señales Mioeléctricas

Especialidad: Tecnólogo en Mecatrónica

Asesor: M.C. y T. Roberto Sosa Cruz

Co-asesor: M.C y T. Jesús Arturo Velarde

Santiago de Querétaro, Querétaro; 2015

Contenido

ABSTRACT 4

RESUMEN 4

DESCRIPCIÓN DEL PROYECTO 5

CAPÍTULO I ESTADO DEL ARTE 7

CAPÍTULO II MARCO TEÓRICO 13

2.1 Fundamento teórico 13

2.1.1 Señales mioeléctricas 13

2.1.2 Electrodos para biopotenciales 14

2.1.3 Cuantificación de señales EMG 15

2.1.4 Problemas comunes en la medición de señales biopotenciales 16

2.1.4.1 Inaccesibilidad de las variables a medir 16

2.1.4.2 Variabilidad de los datos 16

2.1.4.3 Escasez de conocimiento sobre las interrelaciones 17

2.1.4.4 Efecto del transductor en la medida 17

2.1.4.5 Artefactos 18

2.1.4.6 Limitaciones de energía 18

2.1.5 Aplicaciones potenciales de las señales EMG 19

2.1.5.1 Control de interfaces hombre-máquina 19

2.1.5.2 Medicina y rehabilitación 19

2.1.5.3 Biónica y diseño de prótesis 20

2.1.5.4 Aplicaciones aeroespaciales 21

2.1.5.5 Aplicaciones industriales 22

2.1.6 Myo Armband 22

2.1.7 ROS (Robot Operating System) 24

2.1.8 Schunk Arm LWA 4P 25

CAPÍTULO III DESARROLLO 28

3.1 Primeras pruebas 28

3.1.1 Prueba con software MyoDuino y brazalete MYO 28

3.1.2 Fabricación de prototipo de brazo robot para pruebas con MyoDuino 32
3.2 Análisis de gestos con ROS y brazalete MYO 34
3.3 Código en C++ para el reconocimiento de gestos en ROS 36
3.4 Brazo robot inalámbrico controlado con MYO y MyoDuino 40
CAPÍTULO IV RESULTADOS 43
4.1 Resultados de la prueba preliminar de la librería MyoDuino 43
4.2 Gráficas obtenidas de la caracterización de gestos utilizando ROS 44
4.3 Resultados del brazo robot controlado inalámbricamente con MYO y MyoDuino 49
4.4 Simulación del robot Schunk en RVIZ 50
CONCLUSIONES 51
TRABAJO A FUTURO Y RECOMENDACIONES 52
AGRADECIMIENTOS 53
ANEXOS 54
REFERENCIAS 66

ABSTRACT

This manuscript reports the work carried out at the Center of Engineering and Industrial Development, in order to apply myoelectrical signals on the control of manipulator robots. These signals were obtained by using the Myo Armband; an electrical gadget with 8 Medical Grade Stainless Steel EMG sensors, highly sensitive nine-axis IMU containing three-axis gyroscope, three-axis accelerometer and three-axis magnetometer. This armband uses Bluetooth communication, which allows to get the data sensors into a computer by using a Bluetooth adapter. The data is visualized and processed by using ROS (Robot Operating System) installed in a computer with Linux OS. This report explains the methodology that was used during the characterization of the myoelectrical signals and arm gestures, previous tests, codes and the final implementation with the robot Schunk LWA 4P by using the RVIZ simulator.

RESUMEN

En este trabajo se reporte el trabajo realizado en el Centro de Ingeniería y Desarrollo Industrial, con el objetivo de aplicar señales mioeléctricas en el control de robots manipuladores. Estas señales fueron obtenidas utilizando el brazalete Myo; un dispositivo electrónico con 8 Sensores de grado médico de acero inoxidable, una IMU de alta sensibilidad de 9 ejes con un giroscopio de 3 ejes, un acelerómetro de 3 ejes y un magnetómetro de 3 ejes. Este brazalete utiliza comunicación Bluetooth, lo cual permite obtener la información de los sensores desde una computadora utilizando un adaptador Bluetooth. La información es visualizada y procesada utilizando ROS (Sistema Operativo para Robots) instalado en una computadora con Sistema operativo Linux. En este reporte se explica la metodología utilizada durante la caracterización de las señales mioeléctricas y gestos del brazo, pruebas previas, códigos y la implementación final con el robot Schunk LWA 4P utilizando el simulador RVIZ.

DESCRIPCIÓN DEL PROYECTO

Planteamiento del problema

Diseñar y desarrollar un modelo que permita controlar un robot manipulador, utilizando la plataforma ROS como sistema operativo y el brazalete Myo como sistema de adquisición y envío de señales mioeléctricas mediante comunicación serial con Bluetooth.

Justificación

La interacción hombre-máquina cada día es mayor, permitiendo a los robots realizar tareas emulando, con la mayor fidelidad posible, los movimientos biomecánicos derivados de una persona, u operador. Esto ha sido posible gracias a la adquisición de las señales eléctricas producidas por el movimiento de los músculos y articulaciones de los seres vivos. Dichas señales son conocidas como señales mioeléctricas.

El proyecto que se propone impacta directamente en la optimización del manejo de robots manipuladores mediante la adquisición de las señales mioeléctricas, con ayuda de una plataforma llamada Myo. Las posibilidades de su implementación son muy amplias en el sector industrial, tecnológico, de manufactura, automovilístico e incluso en el sector aeroespacial. Pero esta propuesta no está limitada al desarrollo de mejores sistemas mecatrónicos, sino que también puede ser implementada, en un futuro, en el área de la biomédica.

Otra ventaja es que este sistema de control por señales mioeléctricas resulta muy versátil y compatible con cualquier sistema operativo, así como ser implementado con diferentes microcontroladores de fácil adquisición en el mercado de la electrónica. Esto permite que el proyecto sea totalmente viable, como lo han demostrado las primeras pruebas de implementación del Myo con el microcontrolador Arduino para manipular un brazo robot “home-made”.

Objetivo general

Diseñar, desarrollar e implementar un sistema para control de robots manipuladores utilizando el brazalete Myo como sistema de adquisición de señales mioeléctricas.

Objetivos específicos

- Adquisición de las señales mioeléctricas del brazo del operador mediante el brazalete Myo.
- Llevar a cabo el envío de los datos adquiridos hacia un microcontrolador.
- Diseñar una interface de control de posición y orientación para un robot manipulador.
- Implementar el sistema para el control del robot manipulador, basado en los gestos del brazo del operador y utilizando ROS.

CAPÍTULO I ESTADO DEL ARTE

Los sistemas robóticos músculo-esqueléticos inteligentes ofrecen muchas ventajas, especialmente en situaciones donde el ser humano y el robot trabajan de manera cercana.

Se han desarrollado proyectos e investigaciones mediante los cuales se implementan las señales mioeléctricas en aplicaciones de robótica, los cuales se mencionan a continuación.

Myorobotics: un sistema para el desarrollo de un robot músculo-esquelético

El objetivo de este sistema es desarrollar el kit de herramientas Myorobotics, un sistema modular y reconfigurable para el desarrollo de plataformas robóticas músculo-esqueléticas. Se utilizan módulos ingenieriles, los cuales imitan las funciones de sus equivalentes biológicos, como músculos, huesos, articulaciones, etc. Estos módulos, llamados "diseños primitivos" **(Figura 1)** pueden ser configurados y ensamblados para robots fabricados convencionalmente. **[2]**

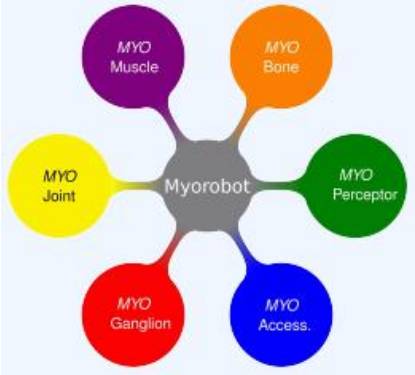

Figura 1. Módulos del sistema Myorobotics

De acuerdo a Myorobotics, los diseños primitivos representan una nueva clase de módulos para robots profesionales. En contraste con los robots convencionales, los diseños primitivos están fuertemente inspirados en el sistema músculo-esquelético humano y animal, lo cual permite al usuario crear Myorobots imitando las extremidades biológicas.

JPL BioSleeve: control preciso de robots mediante la mano y los gestos del brazo

JPL ha estado trabajando en una nueva interface basada en gestos humanos llamada BioSleeve, la cual utiliza sensores, IMUs y magnetómetros para decodificar los gestos de las manos y brazos **[3]**. Consiste en una banda elástica que cubre la mayor parte del antebrazo e incluye 16 contactos de sensores electromiográficos, además de un par de unidades de medición inercial. Los sensores pueden detectar los movimientos de los músculos en el brazo, lo cual significa que el BioSleeve puede detectar cuándo y qué tanto se mueve el brazo, la muñeca, la mano y cada dedo. Esto permite hacer gestos y tener un robot que responda a ellos, como se ilustra en la **Figura 2**.

Figura 2. Uso de la banda ByoSleeve para el control de un robot manipulador

Sistema de sensado robusto EMG basado en la fusión de datos para el control mioeléctrico de un brazo robótico

Este sistema fue diseñado utilizando dos pares de electrodos en cada canal. Las señales eléctricas fueron amplificadas electrónicamente, filtradas y digitalizadas, mientras que el procesamiento, los algoritmos de fusión y el control fueron implementados en una PC utilizando el entorno de MATLAB en un DSP. Los experimentos se hicieron en un manipulador robótico industrial BOSCH SR-800 con cuatro grados de libertad. **[4]**

La Figura 3 muestra los experimentos realizados para la adquisición de datos.

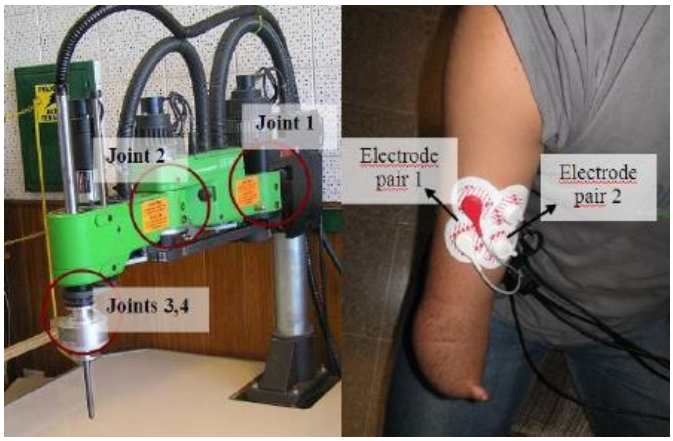

Figura 3. Arreglo experimental utilizado para la adquisición de datos

Interfaz de manejo intuitivo basado en Myo-señales: Diseño y evaluación

El sistema MyoDrive utiliza dos mociones de la muñeca para acelerar, frenar y girar de una forma sencilla. El ángulo horizontal de la muñeca derecha es utilizado para girar, y el ángulo vertical de la muñeca izquierda es usado para el control de velocidad.

La Figura 4 muestra un esquema de la interfaz de manejo mioeléctrica, o MyoDrive. Doblando la muñeca hacia arriba se lleva a cabo la aceleración, mientras que doblándola hacia abajo produce el frenado. **[5]**

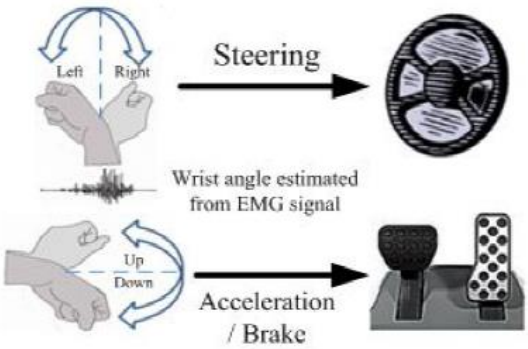

Figura 4. Esquema de la Interfaz de Manejo Mioeléctrica (MyoDrive)

Mano robot antropomórfica basado en un mecanismo sobreactuado y controlado por señales EMG

Se desarrolló una mano humanoide con 5 dedos y 15 articulaciones, la cual actúa mediante 3 motores embebidos y es controlada mediante señales EMG. **[6]**

18 gestos activos de la mano son reconocidos eficientemente, los cuales pueden ser directamente mapeados hacia los movimientos del AR hand III.

Un DSP de alto rendimiento (TMS320F2812) con procesamiento de 150 MPS es adoptado como el chip de control principal para la mano, realizando las tareas de procesamiento de señal y control de motores. Se utilizaron 27 diferentes gestos, como se muestra en la **Figura 5.**

Figura 5. Gestos de la mano organizados por el grado de dificultad

La **Figura 6** muestra los modos de agarre del AR hand III y la **Figura 7** muestra los marcos de control EMG de la mano.

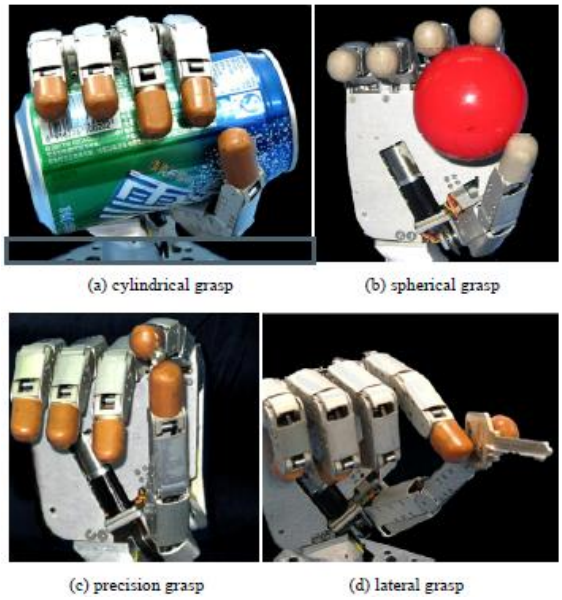

Figura 6. Modos de agarre permitidos por el AR hand III

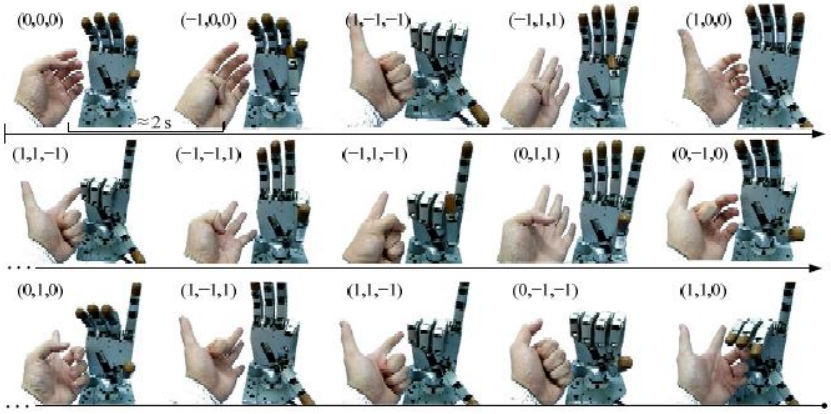

Figura 7. Marcos de control EMG del AR hand III

Interacción hombre-máquina basada en señales EMG – Investigación de plataforma de reconocimiento de patrones

Una mano antropomórfica basada en electromiografía ayuda a la gente sin extremidades a restaurar su funcionalidad. El desarrollo de una mano antropomórfica y un patrón de reconocimiento de señales EMG para la actuación de la mano son temas de particular interés para muchos investigadores biomédicos en los países desarrolladores.

Las señales EMG son recolectadas para reconocer seis gestos de la mano. Estas señales son obtenidas desde el antebrazo de los sujetos utilizando 4 canales de electrodos y una referencia como se muestran en la **Figura 8.**

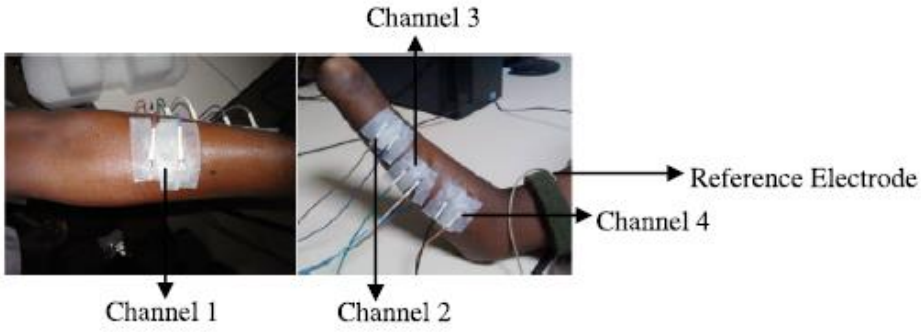

Figura 8. Electrodos conectados al antebrazo de una persona con una mano amputada

En la **Figura 9** se muestra el diagrama de bloques del sistema de acondicionamiento de señal.

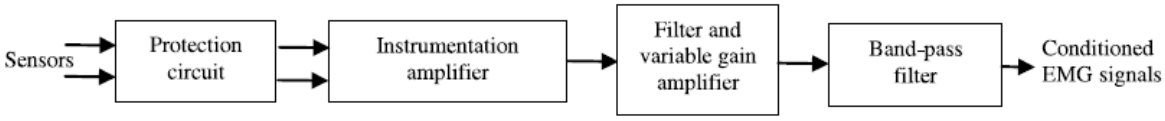

Figura 9. Sistema de acondicionamiento de señal

En la **Figura 10** se muestra la mano prostática utilizada.

Figura 10. Mano prostática con circuito electrónico para adquisición de señales y reconocimiento de patrones

CAPÍTULO II MARCO TEÓRICO

2.1 Fundamento teórico

2.1.1 Señales mioeléctricas

El cuerpo humano genera diferentes tipos de señales eléctricas, dependiendo de la parte que la genere, estas pueden clasificarse en oculográficas, electroencefalográficas y electromiográficas. Estas últimas generadas por la contracción de los músculos del cuerpo, en brazos, pierdas, abdomen, etc, y son producidas por el intercambio de iones a través de las membranas musculares **[7]**. A la detección de estas señales se le conoce como electromiografía. Al contraer los músculos, se genera una señal eléctrica de unos cuantos microvolts, por lo que se hace necesario amplificar esta señal para poder utilizarla, como se ilustra en la **Figura 11**.

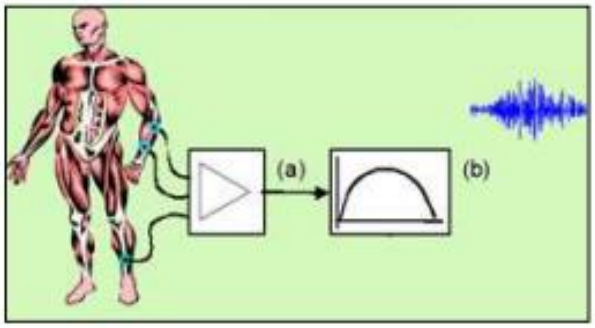

Figura 11. Señales mioeléctricas generadas al contraer o distensionar los músculos

Para capturar las señales EMG se requieren electrodos, los cuales recogen potenciales de todos los músculos dentro de su alcance. Esto significa que los potenciales de músculos grandes cercanos pueden interferir con los intentos de medir el EMG de músculos pequeños aun cuando los electrodos se coloquen directamente sobre los músculos pequeños. Cuando esto es un problema, hacen falta electrodos de aguja insertados directamente dentro del músculo.

2.1.2 Electrodos para biopotenciales

Para medir fenómenos bioeléctricos se pueden utilizar una amplia variedad de electrodos, pero casi todos se pueden clasificar como pertenecientes a uno de tres tipos básicos:

- Microelectrodos: son utilizados para medir potenciales bioeléctricos cerca o dentro de una célula.
- Electrodos superficiales: electrodos utilizados para medir potenciales ECG, EEG y EMG en la superficie de la piel.
- Electrodos de aguja: electrodos utilizados para atravesar la piel y registrar potenciales EEG en una región local del cerebro o potenciales EMG en un grupo de músculos específicos.

Según el tipo de medida EMG se utilizan electrodos superficiales **(Figura 12)**, de aguja **(Figura 13)** y de hilo metálico fino. Los electrodos superficiales se emplean cuando se desean medidas globales, pero cuando se necesita una medida localizada de músculos específicos, hacen falta electrodos de aguja o e cable que atraviesen la piel y contacten con el músculo donde se va a medir.

Figura 12. Electrodos superficiales utilizados para señales EMG

Figura 13. Electrodos de tipo aguja

2.1.3 Cuantificación de señales EMG

Las señales EMG se pueden cuantificar de varias formas. El método más simple es medir sólo la amplitud. En este caso, se registra la amplitud máxima lograda con un determinado tipo de actividad muscular. Otro método de cuantificar las señales EMG es contando el número de picos o en algunos casos el de pasos por cero, que se producen en un intervalo de tiempo determinado. Una variante de este método es contar el número de veces que se supera un umbral de amplitud determinado. **[8]** En la **Figura 14** se ilustra una medición de señal de un músculo.

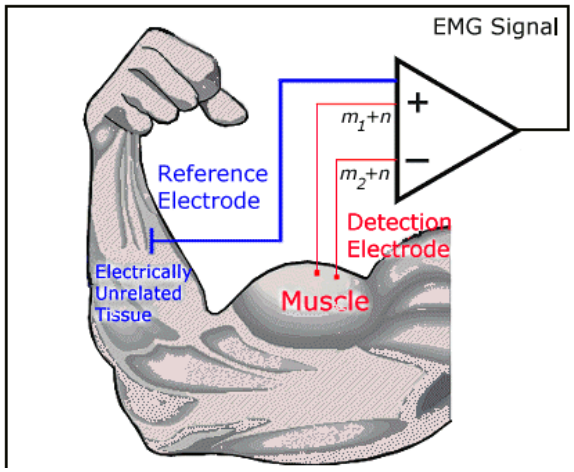

Figura 14. Medición y amplificación de una señal adquirida de un músculo

2.1.4 Problemas comunes en la medición de señales biopotenciales

Debido a que no todos los seres humanos poseen exactamente la misma estructura biológica, resulta complicado diseñar un sistema de reconocimiento de gestos aplicable a cualquier individuo. Dentro de ellos se encuentran la inaccesibilidad de las variables a medir, variabilidad de los datos, escasez de conocimientos sobre las interrelaciones, interacción entre sistemas fisiológicos, artefactos, entre otras.

2.1.4.1 Inaccesibilidad de las variables a medir

Uno de los mayores problemas al intentar medir en un sistema vivo es la dificultad a acceder a las variables a medir. En algunos casos, es imposible colocar un transductor apropiado en un punto para hacer la medición. A veces el problema surge del tamaño físico del transductor comparado con el espacio disponible para medir. En otros casos, la operación médica necesaria para colocar un transductor en un punto desde que se pueda medir la variable hace que la medida sea poco práctica, como el caso de los electrodos de aguja. Cuando una variable es inaccesible a la medida, frecuentemente se intenta efectuar una medida indirecta. Este proceso incluye la medida de otras variables relacionadas que hacen posible en ciertas condiciones una estimación aprovechable de la variable inaccesible.

2.1.4.2 Variabilidad de los datos

Dentro de las variables que se pueden medir en el organismo humano se encuentran algunas en menor cantidad, las cuales son consideradas variables exactamente determinísticas, de hecho, tales variables se deberían considerar como procesos estocásticos. Un proceso estocástico es una función temporal relacionada con otras variables de una forma no determinista. Las variables fisiológicas no se pueden considerar como valores estrictamente determinísticos, sino que se deben representar por algún tipo de distribución estadística o probabilística. En otras palabras, las medidas realizadas en un instante bajo un conjunto de condiciones determinado no serán

necesariamente las mismas que las medidas similares realizadas bajo las mismas condiciones en otro instante. La variabilidad de un individuo a otro es todavía mayor. Aquí se deben emplear de nuevo métodos estadísticos para determinar las relaciones entre variables.

2.1.4.3 Escasez de conocimiento sobre las interrelaciones

La anterior variabilidad en los valores medidos se podría explicar mejor si se conociera y se comprendiera más acerca de las interrelaciones en el organismo. Las medidas fisiológicas con grandes tolerancias son aceptadas frecuentemente por el médico debido a lo escaso de su conocimiento y a la incapacidad de controlar las variaciones resultantes. Una mejor comprensión de las relaciones fisiológicas permitiría además una utilización más efectiva de las medidas indirectas como sustitutos de medidas inaccesibles y ayudaría al ingeniero o técnico en su labor de acoplar la instrumentación al sistema fisiológico. **[8]**

2.1.4.4 Efecto del transductor en la medida

Casi todos los tipos de medición están afectados de alguna forma por la presencia del transductor de medida. El problema se complica mucho en la medida de sistemas vivos. En muchas situaciones la presencia física del transductor cambia la indicación de forma significativa. Por ejemplo, un gran transductor de flujo situado en un torrente sanguíneo bloquea parcialmente el vaso y cambia las características presión-flujo del sistema. Análogamente, un intento de medir los potenciales electroquímicos generados en una célula individuall, exige penetrar en la célula por medio de un transductor. Esta penetración puede matar fácilmente a la célula o dañarla de tal modo que deje de funcionar normalmente. En el diseño de un sistema de medida, el ingeniero o técnico de instrumentación biomédica debe tener el máximo cuidado en asegurar que el efecto de la presencia del instrumento de medida sea mínimo. Debido a la cantidad de energía

disponible en el cuerpo para muchas variables fisiológicas, también se debe tener cuidado en evitar que el sistema de medida cargue la fuente de variable médica. **[8]**

2.1.4.5 Artefactos

En medicina y biología el término artefacto se refiere a cualquier componente de una señal que sea extraña a la variable representada por la señal. De este modo se considera como artefactos al ruido blanco generado por el instrumento de medida, la interferencia eléctrica (incluyendo la captación de 60 Hz) la diafonía y todas las demás variaciones inesperadas de la señal. Una fuente importante de perturbación en la medición en un sistema vivo es el movimiento del individuo, que produce a su vez un movimiento del dispositivo de medida.

Dado que muchos transductores son sensibles al movimiento, cualquier movimiento por parte del individuo produce frecuentemente variaciones en la señal de salida. A veces estas variaciones no se pueden distinguir de la variable medida; en otros casos puede ser suficiente para oscurecer completamente la información deseada.

2.1.4.6 Limitaciones de energía

Muchas técnicas de medida fisiológica necesitan que se aplique cierta cantidad de energía al sistema vivo para obtener una medida. Por ejemplo, las medidas de resistencia necesitan que pase corriente eléctrica por los tejidos o la sangre donde se mide. Algunos transductores generan una pequeña cantidad de calor debido al paso de corriente. En la mayoría de los casos este nivel de energía es tan bajo que su efecto es insignificante. **[8]**

2.1.5 Aplicaciones potenciales de las señales EMG

2.1.5.1 Control de interfaces hombre-máquina

En este campo de aplicación es donde las señales EMG medidas en los músculos activados voluntariamente por un usuario, se procesan y se utilizan como comandos para controlar dispositivos electromecánicos o interfaces de tipo software. Este campo en el que se conjugan la electrónica y la medicina tiene muchas áreas de aplicación en la industria, en la medicina, en el hogar y en el entretenimiento. El control de interfaces hombre-máquina tales como ratones y teclados mioeléctricos, de dispositivos como prótesis mioeléctricas, robots móviles, brazos robóticos o de brazos simulados mediante software, son algunas de las aplicaciones en las que tiene cabida el control utilizando señales mioeléctricas.

2.1.5.2 Medicina y rehabilitación

Un sistema implementado que permite realizar la adquisición y el procesamiento de señales EMG para aplicaciones en medicina y rehabilitación está constituido por un bloque de hardware y un bloque de software. El bloque de hardware lo conforman los módulos de sensores, acondicionamiento de señales y adquisición de señales; el bloque de software está constituido por el módulo de procesamiento de señales **[9]**. En la **Figura 15** se muestra un diagrama de bloques de un sistema que emplea señales EMG.

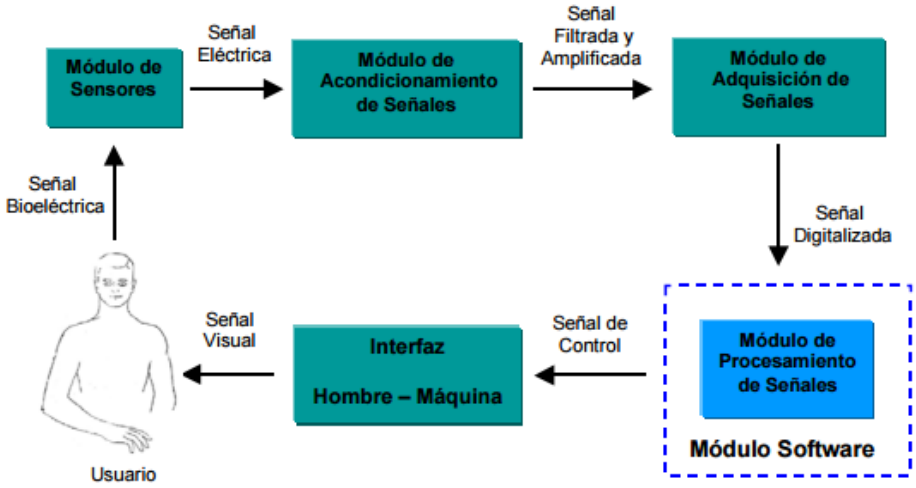

Figura 15. Diagrama de bloques de un sistema aplicando señales EMG a medicina y rehabilitación

El módulo de sensores se encarga de detectar las señales EMG generadas a partir de las contracciones musculares del brazo de un usuario y convertirlas en señales eléctricas, las cuales pasan al módulo de acondicionamiento de señales donde son amplificadas y filtradas para eliminar componentes de ruido que las distorsionen. En el módulo de adquisición de señales se realiza un proceso de digitalización y finalmente en el módulo de procesamiento de señales se realiza el tratamiento de la señal digital mediante algoritmos computacionales.

2.1.5.3 Biónica y diseño de prótesis

Las técnicas más avanzadas de la ingeniería y la robótica se han puesto al servicio de la biociencia para engendrar una nueva disciplina bautizada como biónica. Su objetivo es el diseño de todo tipo de recambios artificiales para sustituir órganos dañados o recuperar funciones perdidas en el organismo humano. Las prótesis pueden ser controladas por medio de los impulsos mioeléctricos que son generados por las neuronas. Esto se logra por medio de electrodos colocados en los músculos. De manera comercial se puede mencionar la C-Leg® (Computerized Leg), mostrada en la **Figura 16.** Es el primer sistema de articulación de rodilla con control hidráulico de la fase de apoyo e impulso regulado de modo electrónico. Es capaz de asimilar los datos de cada fase de la marcha. El técnico ortopédico puede realizar la regulación fundamental con un software en una PC.

Figura 16. Prótesis C-Leg ®

2.1.5.4 Aplicaciones aeroespaciales

Dentro de este ámbito los drones manipulados mediante control remoto pasarán a ser controlados ahora mediante los gestos del brazo del operador. Esto permitirá mejorar la relación hombre-máquina e incluso otorgarle mayor maniobrabilidad a los vehículos aeroespaciales. Un excelente ejemplo dentro del mercado es el dron Parrot AR.Drone 2.0. Para poder manipularlo solamente se necesitan el brazalete MYO, una PC con Windows instalado, el software AutoFlight y un conector. El dron cuenta con estabilizador de vuelo, regreso a casa y cámara HD de 720p. El usuario puede fácilmente controlarlo utilizado los gestos predeterminados del brazalete MYO. La **Figura 17** ilustra el Parrot Ar.Drone 2.0.

Figura 17. Parrot Ar.Drone 2.0

2.1.5.5 Aplicaciones industriales

La adquisición de señales mioeléctricas tiene gran potencial para el control de actuadores y robots de nivel industrial. Esto permite que el operador corra un riesgo menor de accidentes, así como realizar la teleoperación de las máquinas industriales sin intervenir directamente en las áreas de producción. Ejemplo de esto fue el trabajo desarrollado en la Universidad Nacional de San Juan, Argentina. Ahí se desarrolló un sistema que consta de 4 electrodos activos para adquirir y amplificar señales EMG. Estas son utilizadas para la manipulación de un robot industrial BOSCH SR-800, ilustrado en la **Figura 18.**

Figura 18. Robot industrial BOSCH SR-800

2.1.6 Myo Armband

Myo Armband es un gadget en forma de brazalete con 8 sensores electromiográficos de grado médico **[1]**. Tiene la capacidad de expandirse de 19 a 34 cm para ser colocado en el antebrazo de una persona. Este brazalete fue desarrollado por la compañía Thalmic Labs. En su interior se encuentra integrado un procesador ARM Cortex M4, así como una IMU de alta sensibilidad de 9 ejes con un giroscopio de tres ejes, un acelerómetro de tres ejes y un magnetómetro de tres ejes. La **Figura 19** muestra el brazalete Myo.

Figura 19. Aspecto del Myo Armband

El Myo se comunica con una PC o Laptop por Bluetooth (**Figura 20**), utilizando para ello un adaptador Bluetooth Smart. El brazalete contiene una batería recargable de ion de litio, la cual se recarga mediante un cable Micro-USB. La batería puede ser capaz de durar hasta un día entero. En la **Figura 21** se ilustran el adaptador Bluetooth y el cable micro USB para carga.

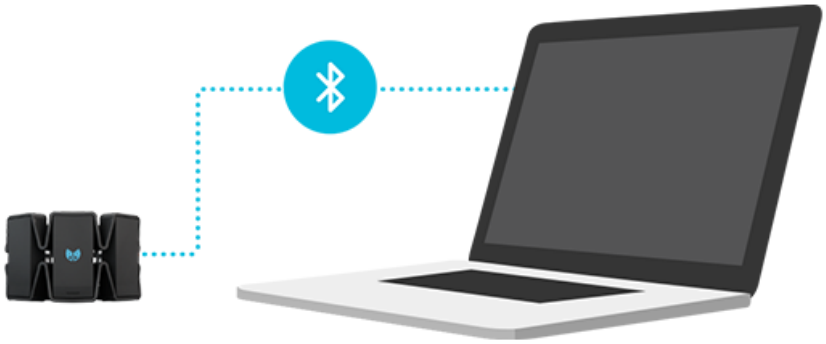

Figura 20. Comunicación entre el Myo y una laptop

Figura 21. Cable micro USB y adaptador Bluetooth

2.1.7 ROS (Robot Operating System)

ROS (Robot Operating System, por sus siglas en inglés) es una plataforma de trabajo ampliamente utilizada en robótica. El principal objetivo de ROS es crear un software que pueda trabajar con varios robots realizando pequeños cambios en el código. Con esto se pueden crear funciones que puedan ser compartidas y utilizadas en otros robots sin emplear demasiado trabajo. **[11]**

Varios sensores y actuadores utilizados en robótica han sido también adaptados para ser utilizados con ROS. Cada día incrementa el número de dispositivos que son soportados por esta plataforma. Provee los servicios estándar de un sistema operativo como abstracción de hardware, control de dispositivos de bajo nivel, envío de mensajes entre procesos y administración de paquetes.

Está basado en una arquitectura gráfica donde el procesamiento toma lugar en nodos que pueden recibir, enviar y multiplicar sensores, control, estado, planeación, actuadores y otros mensajes. En otras palabras, conecta al hardware con el software a través de un entorno de programación C++/Python para controlar hardware. Así mismo, posee poderosas herramientas para correr simulaciones junto con visualizaciones 3D. En la **Figura 22** se ilustran algunos robots que pueden ser controlados a través de ROS.

Figura 22. Robots manipulados por ROS

ROS crea una red donde todos los procesos están conectados. Cualquier nodo en el sistema puede accesar a esta red, interactuar con otros nodos, observar la información que está siendo mandada y transmitir información hacia la red. Los componentes básicos que conforman a ROS son los nodos, el Maestro, el Servidor de Parámetros, mensajes, servicios, tópicos y regresos. Estos se ilustran en la **Figura 23.**

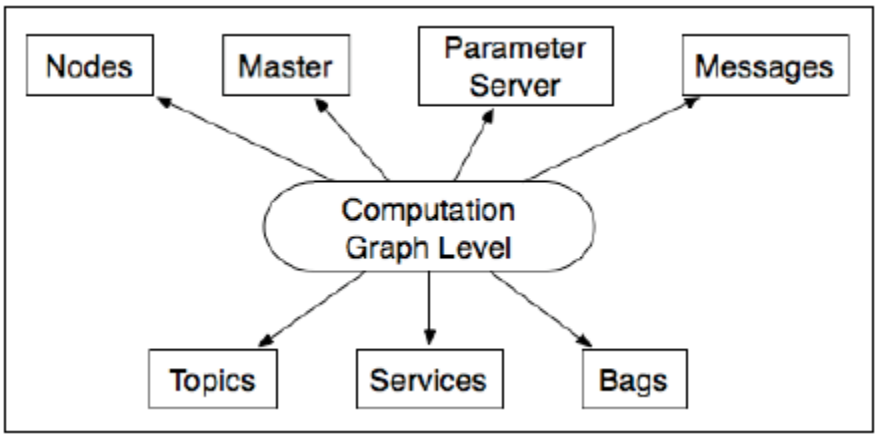

Figura 23. Componentes básicos de ROS

2.1.8 Schunk Arm LWA 4P

Es un brazo robótico que integra servo actuadores modulares de 2 grados de libertad, llamados ERB. El módulo LWA 4P incorpora un amplificador de potencia y un controlador, lo cual permite que no sea requerido un gabinete de control externo, lo cual representa una importante ventaja con respecto a otros brazos robóticos industriales. El

robot está disponible en configuración de 6 grados de libertad. La característica técnica más remarcable del brazo es su alcance de 610 mm y una repetitividad de 0.06 mm. Otra ventaja es su voltaje de 24 VDC, lo cual permite que no se requiera un inversor externo.

Las principales aplicaciones del brazo LWA 4P son robots humanoides, manipulación móvil y servicios asistidos con robots. Su peso es de 12.5 Kg y soporta cargas de hasta 6 Kg. En la **Figura 24** se ilustra el robot LWA 4P y en la **Figura 25** se muestra el módulo servo actuador ERB.

Figura 24. Brazo robot Schunk LWA 4P

Figura 25. Módulo servo actuador ERB

El robot Schunk está compuesto de 6 servomotores independientes, cada uno de ellos controlados mediante el protocolo CANopen sobre un puerto serial común. ROS posee un paquete llamado ipa_canopen, el cual es capaz de implementar exitosamente el protocolo para enviar comandos de velocidad y recibir de vuelta los datos de posición y velocidad de los ejes del robot. En la **Figura 26** se ilustra la conexión entre un servidor ROS y el robot Schunk LWA.

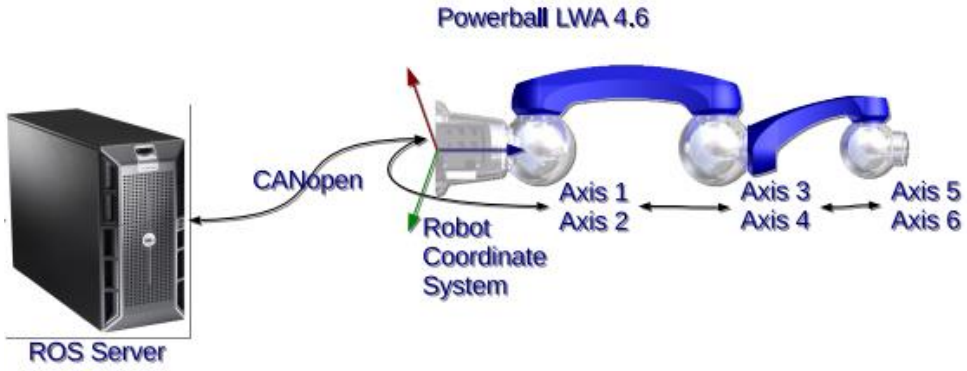

Figura 26. Conexión del servidor ROS y el robot Schunk LWA mediante el protocolo CANopen

CAPÍTULO III DESARROLLO

3.1 Primeras pruebas

3.1.1 Prueba con software MyoDuino y brazalete MYO

Con el propósito de aprender a utilizar el brazalete MYO se optó por realizar una prueba utilizando la placa Arduino UNO y la librería MyoDuino, disponible en la página de Thalmic Labs. Uno de los requerimientos necesarios es tener previamente instalado el software de Arduino, aunque no es estrictamente necesario poseer amplios conocimientos de programación en esta plataforma.

Una vez descargada la carpeta de MyoDuino se descomprime. La carpeta contiene dos carpetas, como se ilustra en la **Figura 27**. El directorio Arduino contiene la librería, mientras que el directorio bin es un programa ejecutable que permite al usuario elegir el puerto serial mediante el cual el Arduino se comunicará con el brazalete MYO.

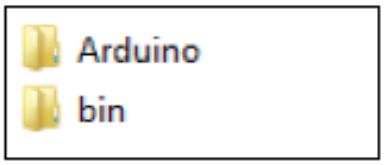

Figura 27. Directorios contenidos en la carpeta MyoDuino

La carpeta Arduino contiene un directorio nombrado como MyoController, el cuál debe copiarse al directorio libraries, contenido en la carpeta de Arduino. Una vez hecho esto, se procede a importar la librería desde el software de Arduino, como se ilustra en la **Figura 28.**

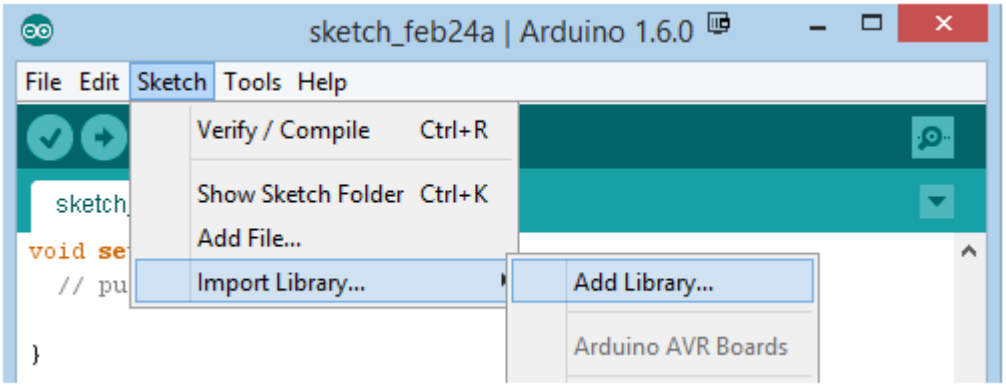

Figura 28. Procedimiento para agregar la librería MyoController

Una vez agregada la librería MyoController se procede a abrir la pestaña File-> Examples -> MyoController -> Examples -> SimpleOutputFromPoses. Hecho esto, se abrirá una ventana con el código que se muestra en las **Figuras 29 y 30.**

```
SimpleOutputFromPoses
#include <MyoController.h>

#define FIST_PIN 4
#define WAVEIN_PIN 5
#define WAVEOUT_PIN 6
#define FINGERSSPREAD_PIN 7
#define DOUBLETAP_PIN 8

MyoController myo = MyoController();

void setup() {

  pinMode(FIST_PIN, OUTPUT);
  pinMode(WAVEIN_PIN, OUTPUT);
  pinMode(WAVEOUT_PIN, OUTPUT);
  pinMode(FINGERSSPREAD_PIN, OUTPUT);
  pinMode(DOUBLETAP_PIN, OUTPUT);

  myo.initMyo();
}
```

Figura 29. Definición de variables y configuraciones del programa SimpleOutputFromPoses

```
void loop(){
  myo.updatePose();
  switch ( myo.getCurrentPose() ) {
   case rest:
     digitalWrite(FIST_PIN,LOW);
     digitalWrite(WAVEIN_PIN,LOW);
     digitalWrite(WAVEOUT_PIN,LOW);
     digitalWrite(FINGERSSPREAD_PIN,LOW);
     digitalWrite(DOUBLETAP_PIN,LOW);
     break;
   case fist:
     digitalWrite(FIST_PIN,HIGH);
     break;
   case waveIn:
     digitalWrite(WAVEIN_PIN,HIGH);
     break;
   case waveOut:
     digitalWrite(WAVEOUT_PIN,HIGH);
     break;
   case fingersSpread:
     digitalWrite(FINGERSSPREAD_PIN,HIGH);
     break;
   case doubleTap:
     digitalWrite(DOUBLETAP_PIN,HIGH);
     break;
  }
  delay(100);
```

Figura 30. Ciclo loop del programa SimpleOutputFromPoses

EL programa es compilado y cargado en la placa Arduino UNO. Para comprobar que este programa funcione correctamente se procedió a conectar 5 LEDs en las salidas digitales 4, 5, 6, 7 y 8 del Arduino UNO, y sus respectivos cátodos se unieron y se conectaron al pin GND como se muestra en la **Figura 31.**

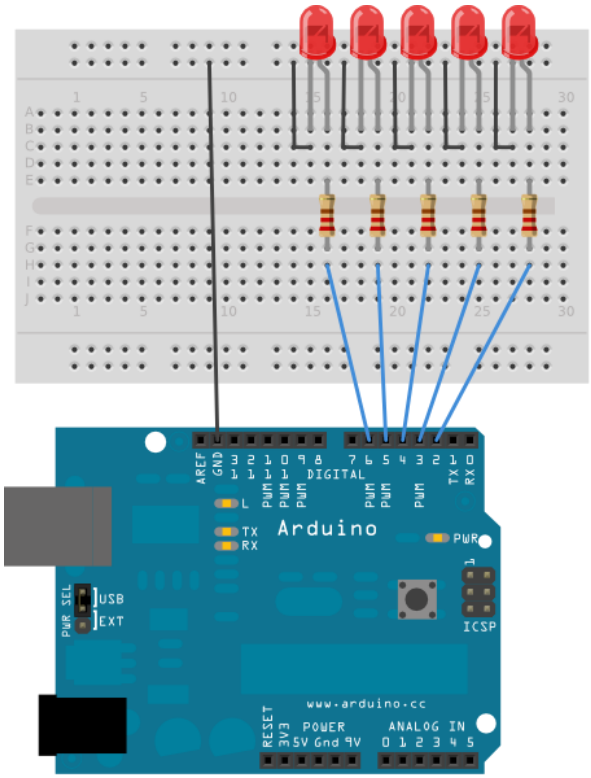

Figura 31. Conexión de LEDs con la placa Arduino UNO

Realizadas las conexiones, debe abrirse el programa MyoDuino.exe de la carpeta MyoDuino. Se abrirá una ventana como la que se muestra en la **Figura 32**, la cual nos permite seleccionar el puerto serial que se encuentre disponible e iniciar la comunicación entre la PC o laptop y el brazalete MYO.

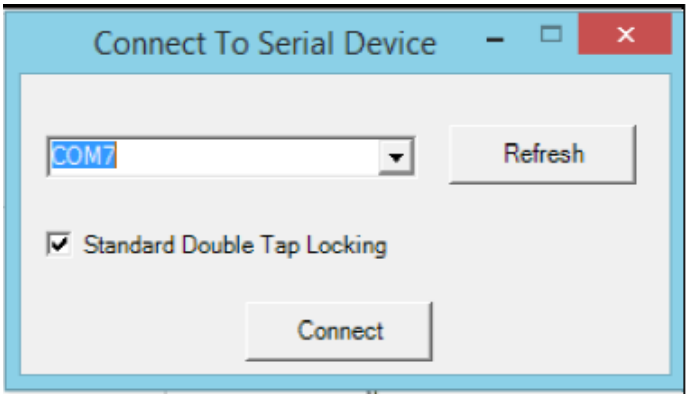

Figura 32. Programa MyoDuino

Una vez seleccionado el puerto serial, se abrirá otra ventana indicando que el brazalete MYO ha sido encontrado, y que se ha establecido una conexión exitosa. También es capaz de reconocer e indicar en qué brazo se encuentra conectado el MYO (R para derecha, L para izquierda). El usuario puede realizar uno de los 5 gestos predeterminados por el brazalete, ilustrados en la **Figura 33.**

Figura 33. Gestos predefinidos capaces de ser reconocidos por el MYO: Double Tap, Finger spread, Wave in, Wave out y Fist

3.1.2 Fabricación de prototipo de brazo robot para pruebas con MyoDuino

Una vez realizada exitosamente la primera prueba con las salidas del Arduino se procedió a realizar un experimento más gráfico, y que permitiera visualizar claramente las opciones que el brazalete ofrece para el control de robots manipuladores. En este caso, se construyó una estructura en forma de brazo robótico de 3 grados de libertad para realizar pruebas de control de servomotores. La posición de los servomotores fue controlada con los gestos predefinidos ejecutados por el usuario.

La estructura del robot de pruebas se construyó utilizando Sintra, ya que es un material, ligero, resistente y fácilmente moldeable, lo cual lo hace ideal para construir prototipos de robots. Los servomotores utilizados fueron de la marca HD, modelos 3001 (4.4 Kg-cm) y 1800 (1.3 Kg-cm), ambos ilustrados en la **Figura 34**. El servomotor 3001 se utilizó en la base del robot debido a su mayor torque, mientras que dos servomotores modelo 1800 se utilizaron para las dos articulaciones con las que el robot cuenta. La posición del robot era controlada utilizando los gestos Fist, Wave in y Wave out.

Figura 34. Servomotores HD modelo 3001 (izquierda) y 1800 (derecha)

La estructura del robot se ilustra en la **Figura 35.**

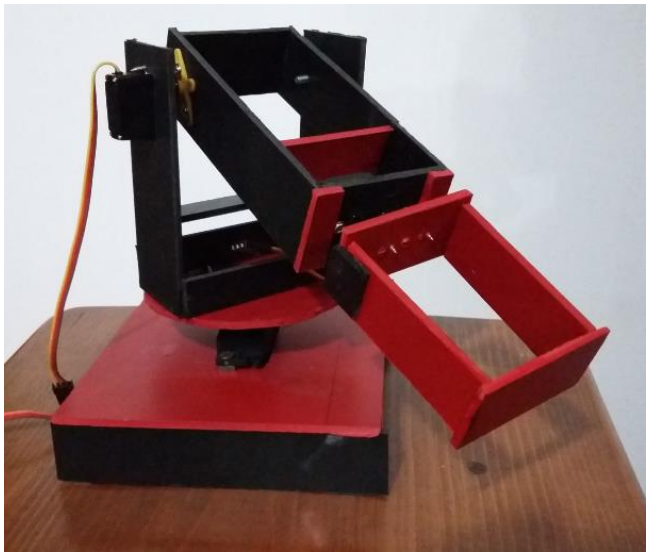

Figura 35. Robot de pruebas del MyoDuino y brazalete MYO

Las pruebas realizadas con este prototipo consistieron en realizar los gestos FIST, WAVE IN y WAVE OUT para modificar la posición en las articulaciones del robot construido. Con el gesto FIST se seleccionaba cada grado de libertad, mientras que con el gesto WAVE OUT se aumentaba su valor de posición y con el gesto WAVE IN se decrementaba. Estas pruebas fueron exitosas, ya que permitieron verificar que la comunicación bluetooth del MYO fuera eficiente para el envío de datos así como lograr una buena maniobrabilidad del robot utilizando solamente los gestos de la mano.

3.2 Análisis de gestos con ROS y brazalete MYO

Mediante ROS se puede llevar a cabo la visualización de datos de sensores en tiempo real. Esto se logra utilizando el comando **rqt_plot**. Mediante esta instrucción se puede accesar a los valores que arrojan los sensores EMG del MYO. Se obtuvieron 12 gráficas de los 12 gestos que se utilizaron para incrementar y decrementar los valores de la posición en X, Y y Z; así como los valores de la orientación en ROLL, PITCH y YAW (**Figura 36**). Los 12 gestos utilizados se ilustran en la **Figura 37** y se nombraron como: gesto3-derecha (1), gesto3-izquierda (2), gesto2-derecha (3), gesto2-izquierda (4), gesto1-derecha (5), gesto1-izquierda (6), fist-derecha (7), fist-izquierda (8), wave out (9), wave in (10), rock-arriba (11) y rock-abajo (12).

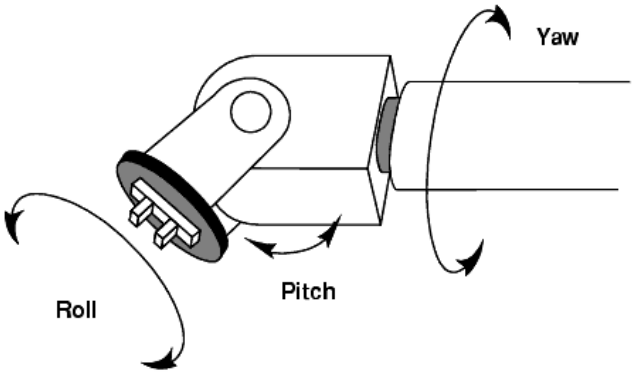

Figura 36. Orientación de un brazo robot en Pitch, Yaw y Roll

Figura 37. Gestos utilizados

El método utilizado para el reconocimiento de gestos fue el análisis de su amplitud. En este caso, se registra la amplitud máxima producida por alguno de los gestos lograda con la actividad muscular. Este análisis permitió hacer una serie de comparaciones y condiciones de los parámetros obtenidos por cada uno de los 8 sensores mioeléctricos y con ello escribir un programa en C++ que permitiera reconocer cada gesto.

La **Tabla 1** ilustra la acción que le fue asignada a cada uno de los gestos. Los gestos que modifican los valores de posición en X, Y y Z se eligieron en base a una relación de orden numérico y alfabético, mientras que aquéllos que modifican los valores de la orientación se eligieron debido a que simulan los movimientos en Roll, Pitch y Yaw. Esto permite al usuario familiarizarse más fácilmente con la manera de controlar los movimientos del robot.

Tabla 1. Gestos utilizados y acciones que realizan

Gesto	Acción que realiza
Gesto1-derecha	Incrementa el valor de X
Gesto1-izquierda	Decrementa el valor de X
Gesto2-derecha	Incrementa el valor de Y
Gesto2-izquierda	Decrementa el valor de Y
Gesto3-derecha	Incrementa el valor de Z
Gesto3-izquierda	Decrementa el valor de Z
Fist-derecha	Incrementa el valor de Roll
Fist-izquierda	Decrementa el valor de Roll
Wave out	Incrementa el valor de Yaw
Wave in	Decrementa el valor de Yaw
Rock-arriba	Incrementa el valor de Pitch
Rock-abajo	Decrementa el valor de Pitch

3.3 Código en C++ para el reconocimiento de gestos en ROS

Utilizando el editor de texto Gedit del sistema operativo Linux se escribió un programa llamado Myolistener. Mediante éste se puede acceder a los datos de los valores obtenidos por los 8 sensores EMG del brazalete MYO. Los datos EMG son condicionados y comparados para detectar cada uno de los 12 gestos establecidos y con ello incrementar o decrementar los valores de la posición y la orientación del robot Schunk. Para lograr esto, es necesario conectar el adaptador bluetooth con el brazalete MYO. En la terminal de Linux se visualiza un mensaje que indica al usuario la conexión exitosa del dispositivo con el adaptador (**Figura 38**).

```
schunk@schunk: ~/catkin_ws
Traceback (most recent call last):
  File "/home/schunk/catkin_ws/src/ros_myo/scripts/myo-rawNode.py", line 440, in
 <module>
    m.run(1)
  File "/home/schunk/catkin_ws/src/ros_myo/scripts/myo-rawNode.py", line 201, in
 run
    self.bt.recv_packet(timeout)
  File "/home/schunk/catkin_ws/src/ros_myo/scripts/myo-rawNode.py", line 80, in
recv_packet
    c = self.ser.read()
  File "/usr/lib/python2.7/dist-packages/serial/serialposix.py", line 446, in re
ad
    ready,_,_ = select.select([self.fd],[],[], self._timeout)
select.error: (4, 'Interrupted system call')
schunk@schunk:~/catkin_ws$ rosrun ros_myo myo-rawNode.py
Initializing...
using device: /dev/ttyACM0
scanning...
scan response: Packet(80, 06, 00, [C4 00 6F 9C 7D 0B 12 ED 00 FF 1F 09 08 52 61
66 61 5F 4D 59 4F 02 01 06 11 06 42 48 12 4A 7F 2C 48 47 B9 DE 04 A9 01 00 06 D5
])
firmware version: 1.3.1448.2
device name: Rafa_MYO
```

Figura 38. Mensaje de conexión del adaptador bluetooth con el brazalete MYO

Una vez que el bluetooth se ha conectado exitosamente con el brazalete se procede a correr el programa Myolistener accediendo a otra ventana y escribiendo la instrucción: "**rosrun ros_myo myolistener**". Con esto se despliega una ventana en la cual el usuario puede elegir entre el modo de entrenamiento o de control, tecleando las letras

1 o 2, respectivamente para cada opción. En el modo de entrenamiento el usuario puede modificar los valores de posición y orientación mediante los 12 gestos, pero sin publicar los datos. El modo de control es similar al de entrenamiento, con la única diferencia que los datos son publicados, y con ello, se pueden introducir al programa de simulación de ROS para el robot Schunk. La **Figura 39** muestra la ventana de instrucciones que se despliega al seleccionar el modo de entrenamiento.

Figura 39. Visualización de instrucciones en la Terminal del modo de entrenamiento

Para iniciar el programa, tanto en modo entrenamiento como en el de control, es necesario que el usuario ejecute el gesto Fist. Con ello se despliega una tabla con los valores de posición y orientación del robot Schunk, los cuales se irán modificando con cada gesto realizado por el usuario. La tabla con los valores de posición y orientación se ilustran en la **Figura 40.**

```
schunk@schunk: ~/catkin_ws
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
x = 0.42  y = 0.36  z = 0.44  Roll = 0.03  Pitch = 3.41159  Yaw = 0.24
```

Figura 40. Tabla con valores de posición y orientación en Terminal de Linux

Cuando el usuario ha seleccionado el modo de control es posible publicar los datos en el programa RVIZ de ROS. Este permite llevar a cabo una simulación del robot Schunk, en el cual se visualiza el cambio en su posición y orientación al ejecutar alguno de los 12 gestos utilizados. La **Figura 41** presenta el diagrama de todo el sistema, desde el performance de los gestos del usuario hasta la simulación del robot Schunk.

Las **Figuras 65-76** de la sección **Anexos** muestran el código escrito en C++ con el programa Myolistener.

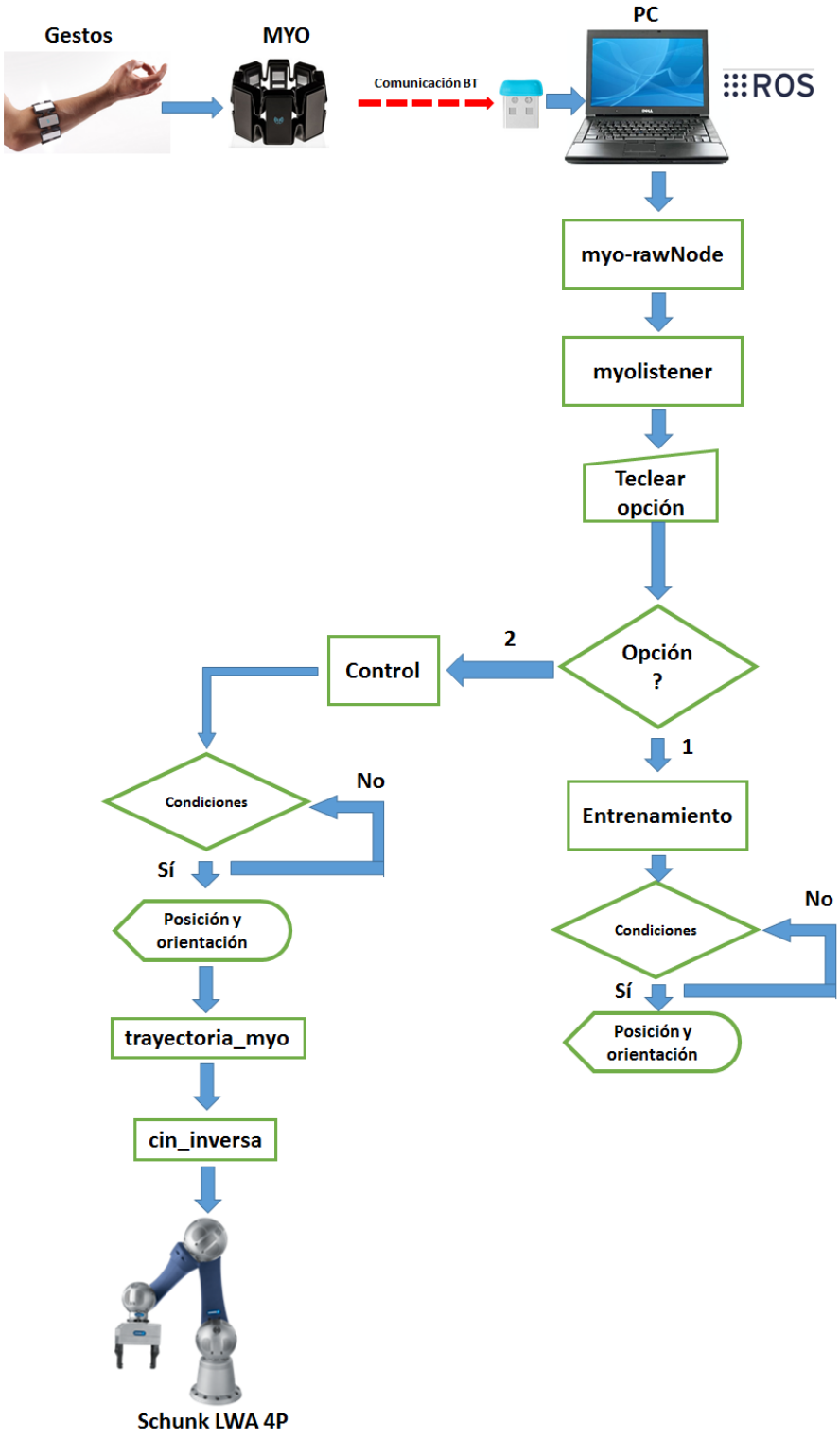

Figura 41. Diagrama de bloques del sistema de reconocimiento de gestos

3.4 Brazo robot inalámbrico controlado con MYO y MyoDuino

Se construyó un brazo robot de 3 grados de libertad utilizando soportes pan/tilt en vez de Sintra, como en la primera prueba realizada. Estos soportes permiten interconectar servomotores junto con otros soportes, además que tienen como ventaja ser ligeros y fácilmente montables. La **Figura 42** muestra un servomotor montado en uno de estos soportes.

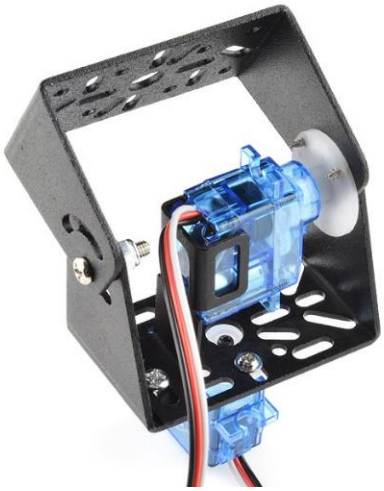

Figura 42. Servomotor montado en soporte pan/tilt

Además de la mejora en su estructura, este brazo se diferencia de la primera prueba en que es controlado con el MYO y el software MyoDuino de manera inalámbrica, utilizando dos módulos bluetooth HC-05. Para comunicar ambos módulos se configuró a uno como maestro y al otro como esclavo. Por un lado, el brazo robot se controló con un Arduino Nano y un módulo bluetooth esclavo. En el otro extremo se encuentra un Arduino UNO con el módulo bluetooth configurado como maestro, y el Arduino UNO se encuentra conectado al puerto serial (COM 5 en este caso) de la computadora, donde se están leyendo los gestos del brazalete mediante el software MyoDuino.

Los servomotores utilizados para este robot fueron, al igual que en la primera prueba, los modelos 1800 y 3001 de la marca HD. Ambos alimentados con 5V. En la **Figura 43** se ilustra la estructura de este brazo robot construido.

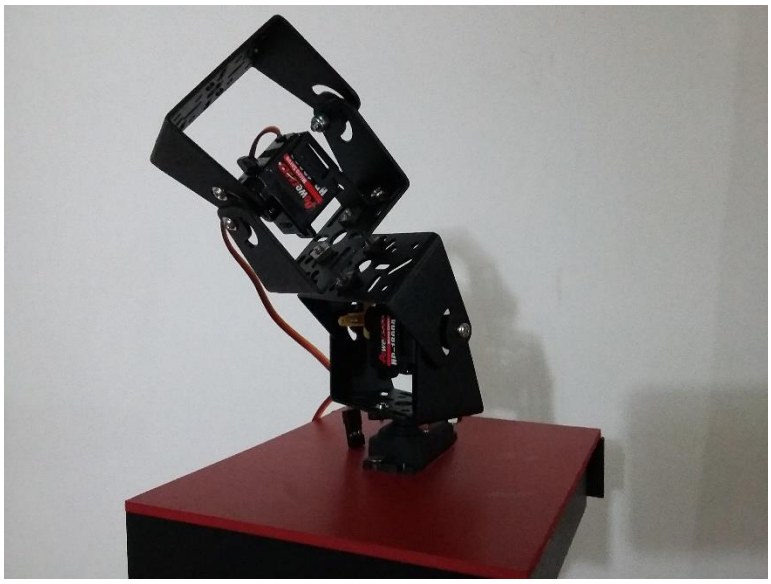

Figura 43. Brazo robot construido con soportes pan/tilt

La **Figura 44** ilustra el diagrama de bloques del sistema del brazo robot controlado inalámbricamente.

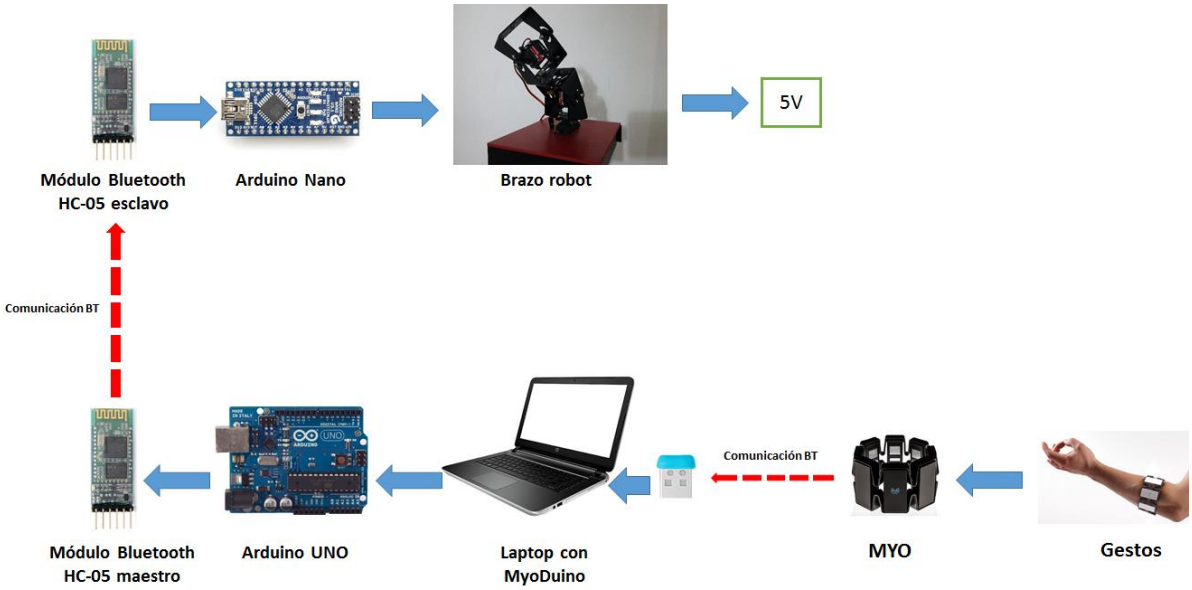

Figura 44. Diagrama de bloques del sistema de brazo robot controlado inalámbricamente

Para acceder a cada articulación de este robot se utiliza el gesto Double Tap. El ángulo de cada servo motor se incrementa en 5 unidades realizando el gesto Wave Out, mientras que el gesto Wave In decrementa en 5 unidades el ángulo de los servomotores.

El código utilizado para el Arduino Uno se nombró como “myobt” y el código para el Arduino Nano se nombró “brazomejorado”. Ambos se ilustran en las **Figuras 77-80** de la sección **Anexos.**

La **Figura 45** ilustra una de las pruebas realizadas con el robot y el gesto Wave In.

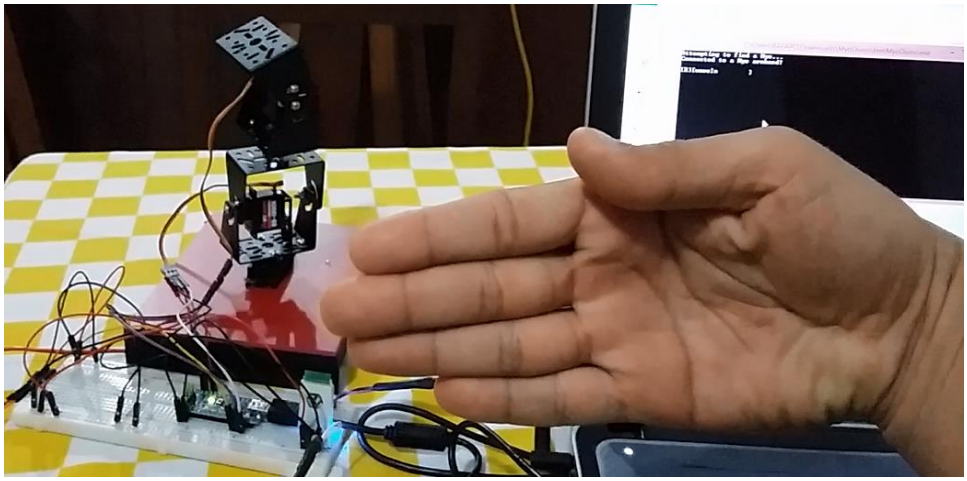

Figura 45. Pruebas del brazo robot mediante el gesto Wave In

CAPÍTULO IV RESULTADOS

4.1 Resultados de la prueba preliminar de la librería MyoDuino

En la **Figura 46** se ilustra la ventana principal del programa MyoDuino. En ella se observa el mensaje "Connected to a Myo Armband", indicando que se ha establecido una conexión exitosa del brazalete. De esta manera el usuario podrá realizar los gestos predefinidos y observar en la pantalla el nombre correspondiente a cada gesto.

Figura 46. Ventana principal del programa MyoDuino.exe

A través del código grabado en Arduino UNO se observó que un LED de cada salida encenderá cada que se realiza uno de los cinco gestos predeterminados. El Programa se modificó **(Figura 47)** para que con el gesto Double tap se configuraran en nivel alto las 5 salidas del Arduino. La construcción física de la prueba se ilustra en la **Figura 48.** Se obtuvo un buen resultado realizando estas pruebas, ya que el programa era capaz de detectar y reconocer los gestos, así como realizar las acciones que se indicaban para cada gesto realizado.

```
case doubleTap:
{
  digitalWrite(DOUBLETAP_PIN,HIGH);
  digitalWrite(FIST_PIN,HIGH);
  digitalWrite(WAVEIN_PIN,HIGH);
  digitalWrite(WAVEOUT_PIN,HIGH);
  digitalWrite(FINGERSSPREAD_PIN,HIGH);
}
  break;
```

Figura 47. Modificación al programa original SimpleOutputFromPoses

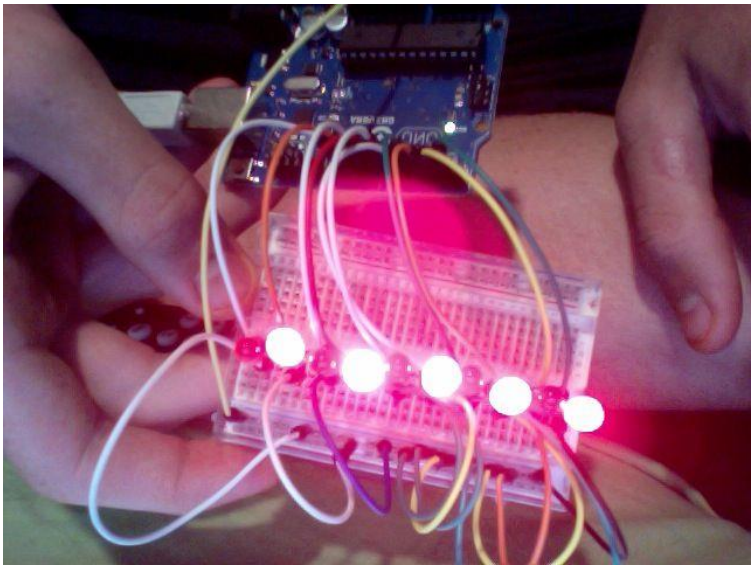

Figura 48. Circuito de prueba para el programa SimpleOutputFromPoses

4.2 Gráficas obtenidas de la caracterización de gestos utilizando ROS

En las **Figuras 49-60** se ilustran las gráficas de cada gesto obtenidas mediante el comando **rqt_plot** de ROS. Mediante el análisis de la amplitud de las señales obtenidas por los sensores mioeléctricos se pudo escribir el código en C++ que reconociera a cada señal, y con ello, modificar los valores de posición y orientación que serían probados en la simulación del robot Schunk.

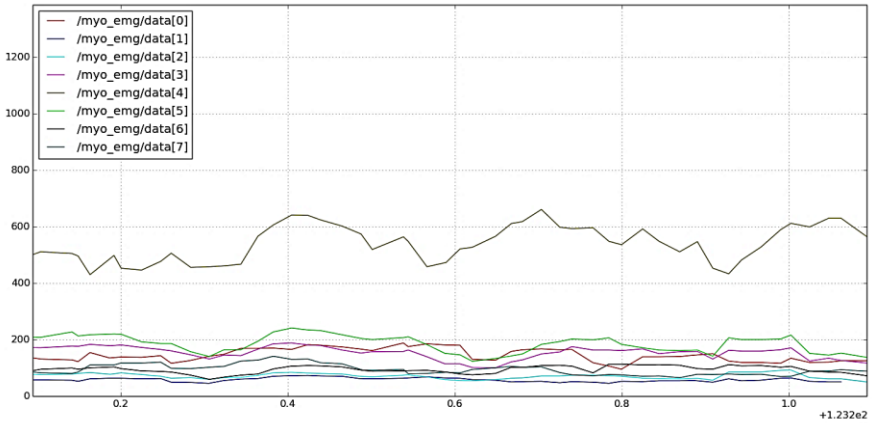

Figura 49. Gráfica obtenida del gesto3-derecha

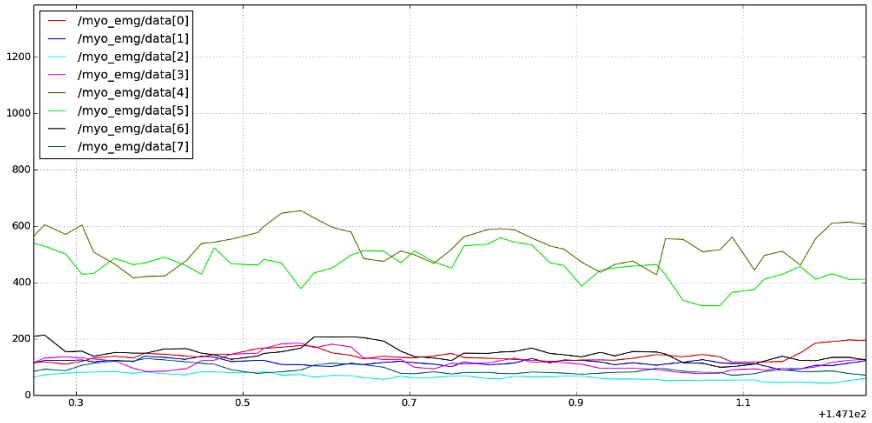

Figura 50. Gráfica obtenida del gesto3-izquierda

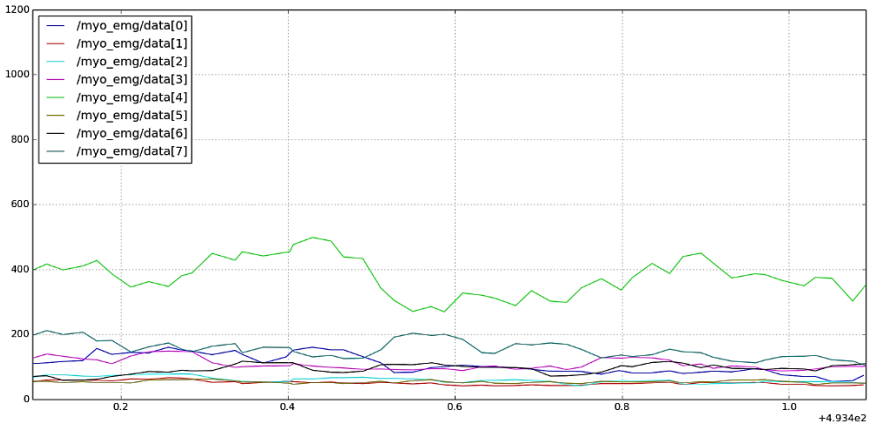

Figura 51. Gráfica obtenida del gesto2-derecha

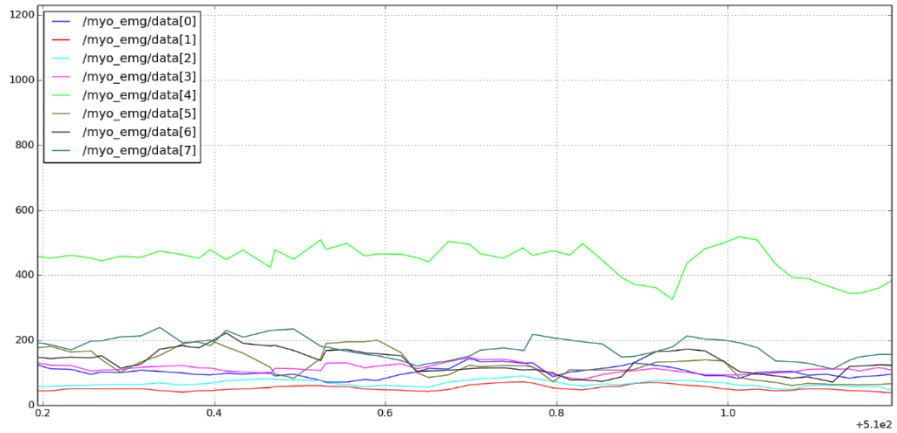

Figura 52. Gráfica obtenida del gesto2-izquierda

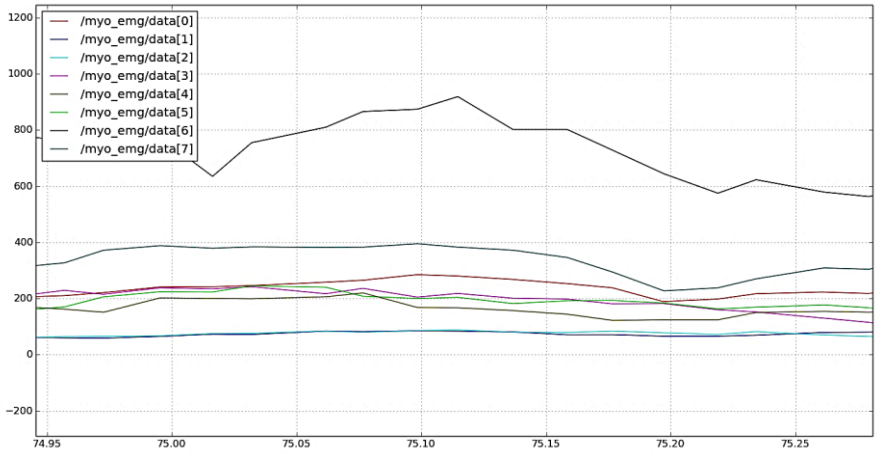

Figura 53. Gráfica obtenida del gesto fist-derecha

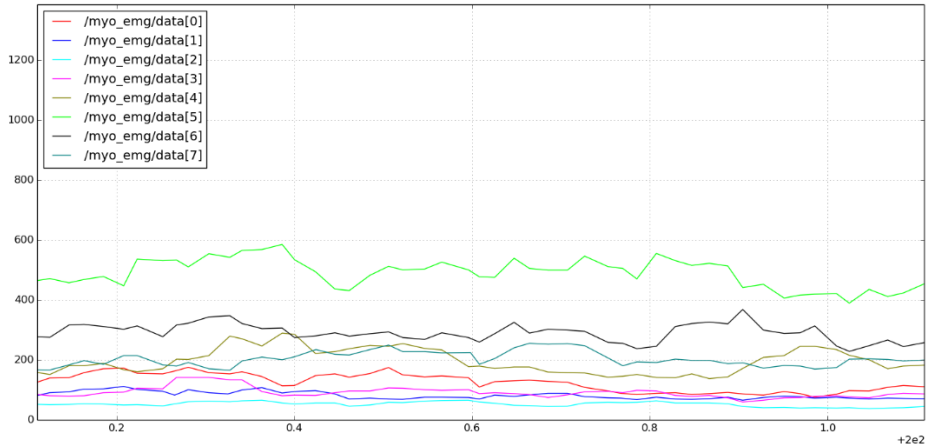

Figura 54. Gráfica obtenida del gesto fist-izquierda

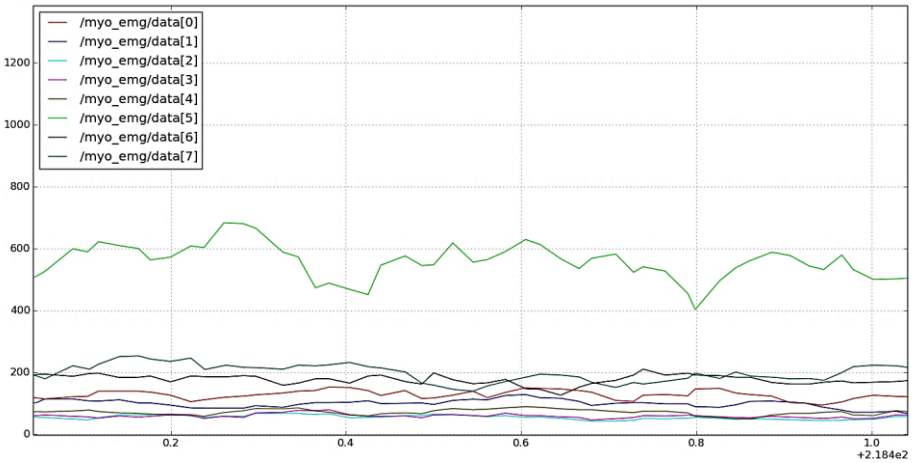

Figura 55. Gráfica obtenida del gesto1-izquierda

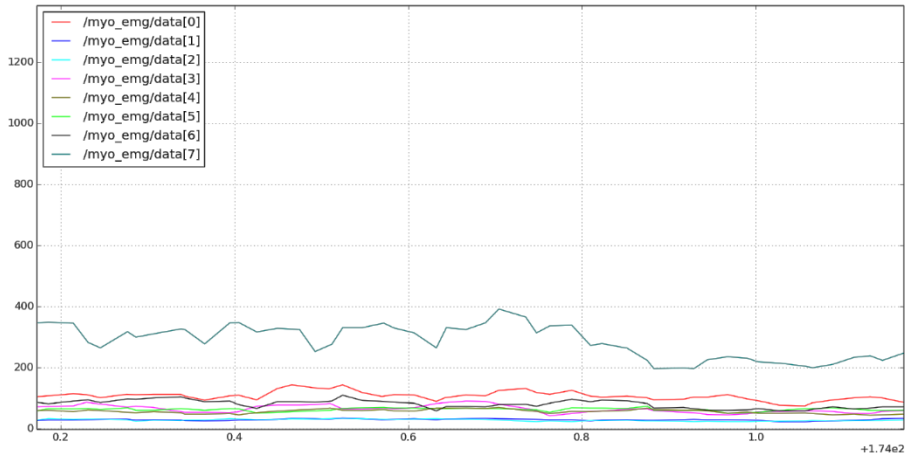

Figura 56. Gráfica obtenida del gesto1-derecha

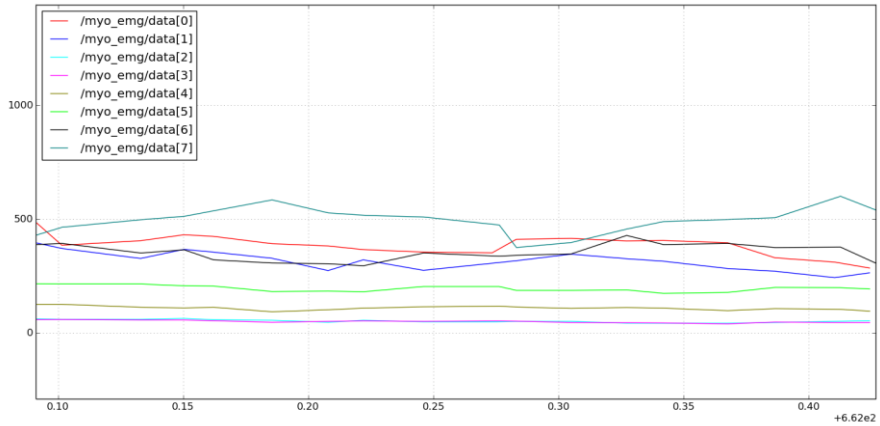

Figura 57. Gráfica obtenida del gesto Wave In

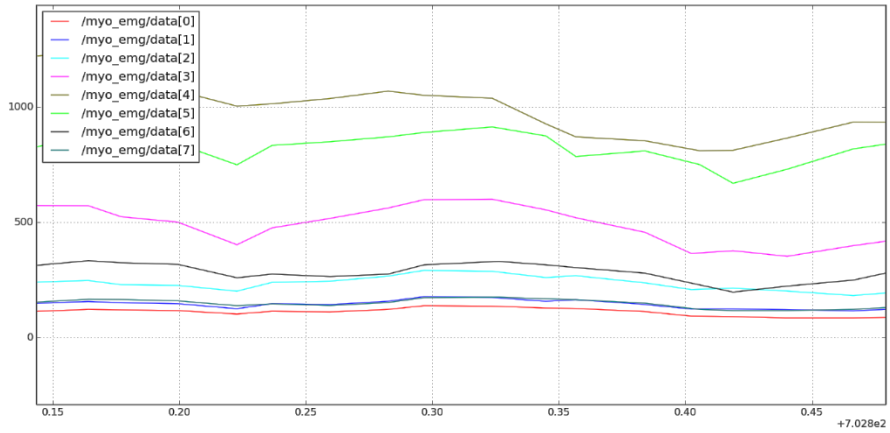

Figura 58. Gráfica obtenida del gesto Wave Out

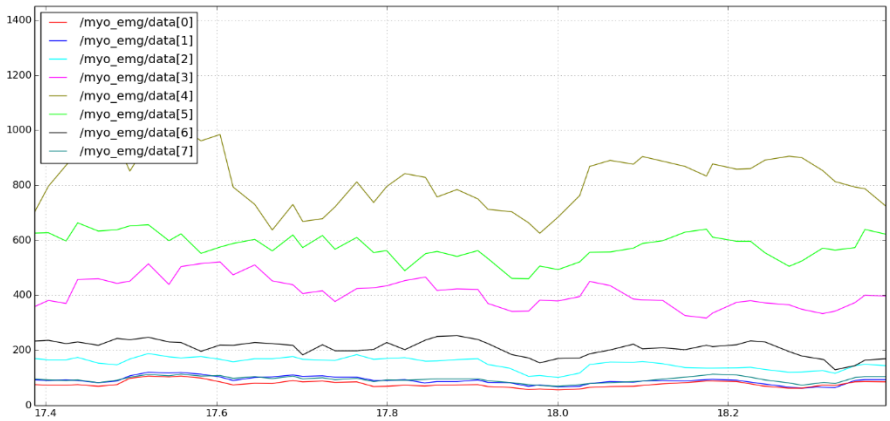

Figura 59. Gráfica obtenida del gesto Rock-Arriba

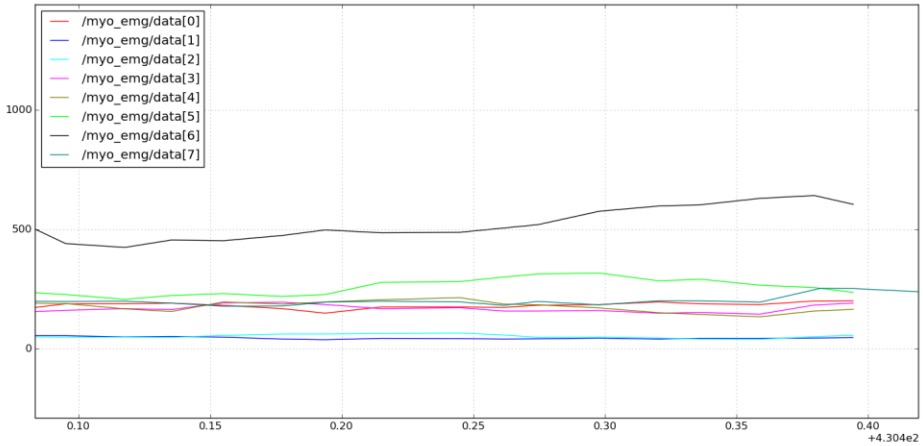

Figura 60. Gráfica obtenida del gesto Rock-Abajo

4.3 Resultados del brazo robot controlado inalámbricamente con MYO y MyoDuino

Se logró controlar exitosamente el brazo robot ahora construido con soportes pan/tilt de aluminio. Esto fue posible gracias a la exitosa conexión entre los módulos Bluetooth HC-05 como maestro-esclavo, así como al código escrito en el software de Arduino.

El Arduino UNO fue capaz de reconocer los gestos predeterminados del MYO y con ello enviar datos seriales correspondientes a cada uno. Esos datos seriales eran enviados a través del bluetooth maestro y recibidos, en el otro extremo por el esclavo. El Arduino Nano conectado al brazo robot interpretaba esos datos seriales y con ello se seleccionaban las articulaciones y se modificaban los ángulos correspondientes a cada servomotor. La **Figura 61** muestra el control de una de las articulaciones siendo manipulada mediante el gesto Wave In. La **Figura 62** ilustra el control de otra articulación mediante el gesto Wave Out.

Figura 61. Control de una articulación utilizando el gesto Wave In

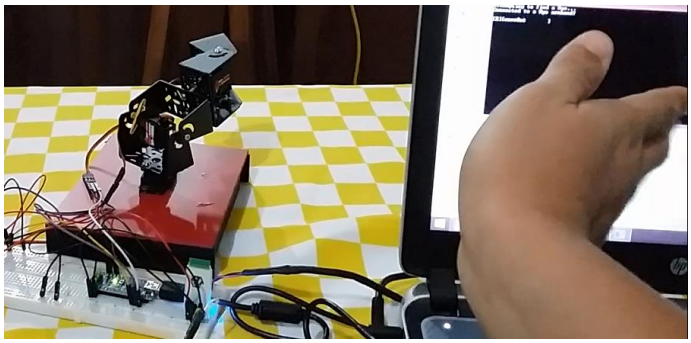

Figura 62. Control de una articulación utilizando el gesto Wave Out

4.4 Simulación del robot Schunk en RVIZ

Utilizando RVIZ se pudo simular el movimiento del brazo robot Schunk. Esto se logró mediante el performance de los 12 gestos utilizados y publicando los datos de posición y orientación a través de ROS. En la **Figura 63** se ilustra el diagrama de bloques de los nodos y tópicos utilizados para publicar los datos de los sensores EMG del brazalete MYO.

Figura 63. Diagrama de bloques de los nodos y tópicos en ROS

En la **Figura 64** se muestra el movimiento simulado del robot Schunk en el RVIZ.

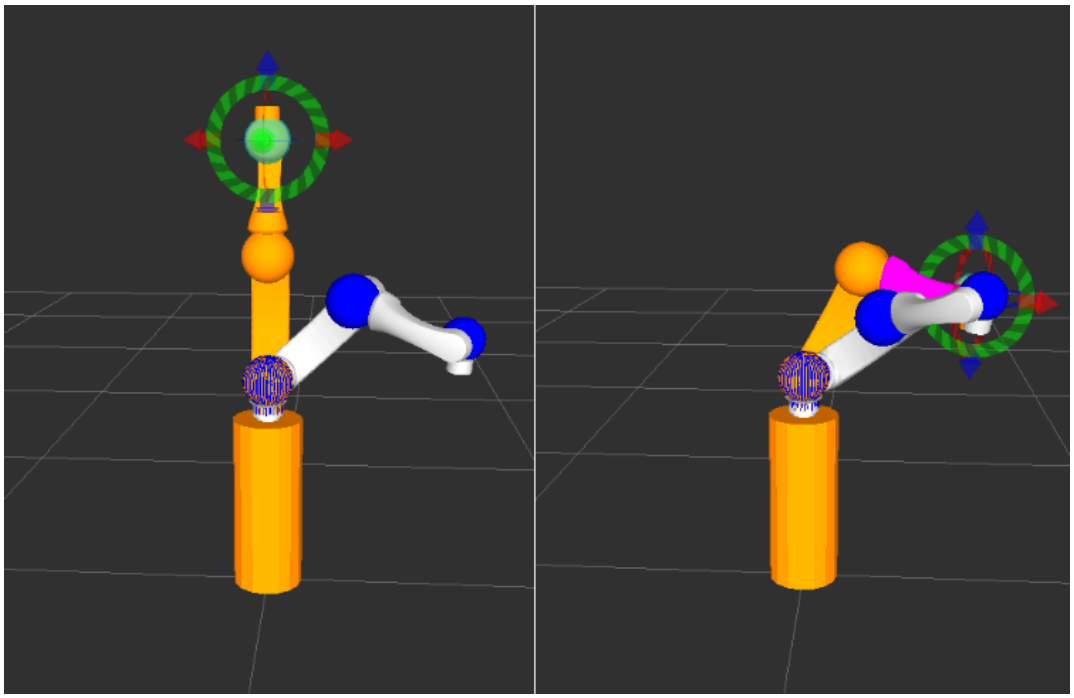

Figura 64. Simulación del robot Schunk en RVIZ

CONCLUSIONES

Durante el desarrollo de este proyecto de Especialidad se visualizó que el trabajo con señales mioeléctricas puede resultar complejo, debido a fallas o desconexión en los electrodos, cambios bruscos de las lecturas debido a los movimientos, ruido generado por gestos no reconocidos y la variabilidad de datos producida por los diferentes tamaños en las extremidades de los sujetos de prueba. El proyecto realizado presenta, de manera conceptual, que las señales mioeléctricas aplicadas a la robótica poseen un amplio potencial, debido a que refuerzan la relación hombre-máquina y facilitan el control de actuadores.

El uso de algoritmos de programación facilita la tarea del reconocimiento de gestos, los cuales permiten la manipulación de robots u otros dispositivos. Sin embargo; estos algoritmos deben ser, preferentemente, basados en Redes Neuronales.

Uno de los principales problemas enfrentados fue la inaccesibilidad que se tuvo para la manipulación del robot Schunk LWA 4P. Esto se debe a que el código escrito envía uno a uno los datos necesarios para manipular los valores de posición y orientación; mientras que físicamente se necesitan enviar juntos los valores deseados al robot.

El uso del MYO suprime la necesidad de elaborar un sistema de adquisición de señales EMG electrónico, debido a que este brazalete ya cuenta con sensores de alto nivel, así como comunicación bluetooth. La versatilidad de este dispositivo se acentúa con la posibilidad de conectarlo con un Microcontrolador Arduino, lo cual permitió hacer uso de librerías con las cuales se realizó un buen reconocimiento de gestos y con ello la manipulación de robots de prueba.

TRABAJO A FUTURO Y RECOMENDACIONES

Trabajar con señales mioeléctricas resulta bastante complejo, debido a la variabilidad de datos producida por las características fisiológicas de cada individuo. No siempre se obtendrán lecturas semejantes de sensores mioeléctricos, debido a que el tamaño y masa de las extremidades de cada individuo son siempre diferentes. De igual manera afecta mucho la orientación de los sensores colocados en una extremidad, así como el lado con el que se elija trabajar, ya sea izquierdo o derecho.

Esta complejidad en el reconocimiento y procesamiento de señales EMG deja abiertas la posibilidades de futuros trabajos para su aplicación en cualquiera de sus aplicaciones potenciales (industria, robótica, automatización, aeroespacial, etc.). Posiblemente en el futuro se logre identificar de manera inequívoca un conjunto de gestos provenientes de las extremidades de diferentes individuos, y con ello, llevar las ventajas del trabajo con señales EMG más allá de los grupos dedicados a su investigación.

Gracias al trabajo realizado se pueden dar algunas recomendaciones para futuros proyectos, tesis o investigaciones que estén dispuestas a enfocarse hacia este campo.

- Desarrollar un sistema de adquisición y almacenamiento de datos que permita guardar y reconocer un conjunto de gestos provenientes de diferentes individuos. Esto permitiría también el uso de las señales EMG provenientes de los gestos para su posterior aplicación.
- Utilizar programación inteligente para el reconocimiento de gestos, principalmente sistemas basados en Redes Neuronales. Con ello se pueden ingresar datos correspondientes a las señales EMG de un conjunto de gestos, guardarlos y entrenar a la Red Neuronal para que sea capaz de reconocerlos.
- Trabajar con electrodos EMG de calidad. Esto mejora la calidad de las señales sensadas. De igual manera se recomienda mantener libre de suciedad o agentes contaminantes a la extremidad sobre la cual se vayan a tomar muestras.

AGRADECIMIENTOS

Agradezco al Centro de Ingeniería y Desarrollo Industrial por permitirme llevar a cabo los estudios correspondientes al programa de Especialidad de Tecnólogo en Mecatrónica, así como por facilitarme el uso de sus instalaciones y laboratorios necesarios para realizar prácticas.

A mis asesores, el M. C. y T. Roberto Sosa Cruz y el M. C. y T. Jesús Arturo Velarde, por guiarme durante el desarrollo del proyecto y compartir sus conocimientos.

Al Ing. Jaime Trejo Almaraz quien me apoyó durante el desarrollo del proyecto, así como facilitarme la herramienta MYO Armband, necesaria para llevar a cabo las pruebas y prácticas necesarias.

Al equipo de trabajo, conformado por Lucía Limones Pérez, Oscar Rodríguez, Diego Martínez y Julio Gavito, quienes me brindaron su apoyo durante el proyecto.

```
#include <ros/ros.h>
#include <std_msgs/String.h>
#include <EmgArray.h>
#include <iostream>
#include <sstream>
#include <string>
#include <stdio.h>
#include <fstream>
#include <new>
#include <geometry_msgs/Twist.h>

#define DER1 0
#define IZQ1 1
#define DER2 2
#define IZQ2 3
#define DER3 4
#define IZQ3 5
#define FISTDER 6
#define FISTIZQ 7
#define WAVEOUT 8
#define WAVEIN 9
#define ROCKARR 10
#define ROCKABA 11

using namespace std;
            int b = 0;
    long double x = 0.45;
    long double y = 0.1;
    long double z = 0.15;
long double roll  = 0;
long double pitch = 3.141592;
long double yaw   = 0;

int der1 =0;
int izq1 =0;
int der2=0;
int izq2=0;
int der3=0;
```

Figura 65. Fragmento de código del programa Myolistener

```
int izq3=0;
int fistder=0;
int fistizq=0;
int waveout=0;
int wavein =0;
int rockarr=0;
int rockaba=0;
int modo  = 0;

    long double x2 = 0;
    long double y2 = 0;
    long double z2 = 0;
long double roll2  = 0;
long double pitch2 = 0;
long double yaw2   = 0;

int opcion;
int v;

ros::Publisher posicion_pub;
geometry_msgs::Twist posicion_deseada;

//////////////////////////////7 Función para resetear los contadores de las variables
void resetConts(int o)
{
        o == DER1 ? der1++ : der1 = 0;
        o == IZQ1 ? izq1++ : izq1 = 0;
        o == DER2 ? der2++ : der2 = 0;
        o == IZQ2 ? izq2++ : izq2 = 0;
        o == DER3 ? der3++ : der3 = 0;
        o == IZQ3 ? izq3++ : izq3 = 0;
        o == FISTDER ? fistder++ : fistder = 0;
        o == FISTIZQ ? fistizq++ : fistizq = 0;
        o == WAVEOUT ? waveout++ : waveout = 0;
        o == WAVEIN ? wavein++ : wavein = 0;
        o == ROCKARR ? rockarr++ : rockarr = 0;
        o == ROCKABA ? rockaba++ : rockaba = 0;
}
```

Figura 66. Fragmento de código del programa Myolistener

```
//////////////////////////Función para incrementar el valor en X
void incx ()
{
        resetConts(DER1);

        if (der1 == 10)
        {
        x= x+0.01;
        der1 = 0;
        }
}
//////////////////////////Función para decrementar el valor en x

void decx()
{
        resetConts(IZQ1);

        if (izq1 == 10)
        {
        x= x-0.01;
        izq1 = 0;
        }
}

//////////////////////////Función para incrementar el valor en y

void incy()
{
        resetConts(DER2);

        if (der2 == 10)
        {
        y= y+0.01;
        der2 = 0;
        }
}

//////////////////////////Función para decrementar el valor en y
```

Figura 67. Fragmento de código del programa Myolistener

```
void decy()
{
        resetConts(IZQ2);

        if (izq2 == 10)
        {
        y= y-0.01;
        izq2 = 0;
        }
}

//////////////////////////Función para incrementar el valor en z

void incz()
{
        resetConts(DER3);

        if (der3 == 10)
        {
        z= z+0.01;
        der3 = 0;
        }
}

//////////////////////////Función para decrementar el valor en z

void decz()
{

        resetConts(IZQ3);

        if (izq3 == 10)
        {
        z= z-0.01;
        izq3 = 0;
        }
}
```

Figura 68. Fragmento de código del programa Myolistener

```
//////////////////////////Función para incrementar el valor en roll

void incroll()
{
        resetConts(FISTDER);

        if (fistder == 10)
        {
        roll= roll+0.01;
        fistder = 0;
        }
}

//////////////////////////Función para decrementar el valor en roll

void decroll()
{
        resetConts(FISTIZQ);

        if (fistizq == 10)
        {
        roll= roll-0.01;
        fistizq = 0;
        }
}

//////////////////////////Función para incrementar el valor en pitch

void incpitch()
{
        resetConts(ROCKARR);

        if (rockarr == 10)
        {
        pitch= pitch+0.01;
        rockarr = 0;
        }
}
```

Figura 69. Fragmento de código del programa Myolistener

```
//////////////////////////Función para decrementar el valor en pitch

void decpitch()
{
        resetConts(ROCKABA);

        if (rockaba == 10)
        {
        pitch= pitch-0.01;
        rockaba = 0;
        }
}

//////////////////////////Función para incrementar el valor en yaw

void incyaw()
{
        resetConts(WAVEOUT);

        if (waveout == 10)
        {
        yaw= yaw+0.01;
        waveout = 0;
        }
}

//////////////////////////Función para decrementar el valor en yaw

void decyaw()
{
        resetConts(WAVEIN);

        if (wavein == 10)
        {
        yaw= yaw-0.01;
        wavein = 0;
        }
}
```

Figura 70. Fragmento de código del programa Myolistener

```
//////////////////////////Función para el brazo relajado

void relax()
{
        //ROS_INFO("Nada");
        x = x;
        y = y;
        z = z;
        roll  = roll;
        pitch = pitch;
        yaw   = yaw;
}


void myo_emgCallback(const ros_myo::EmgArray::ConstPtr& msg)
{ //inicio void


////////////////////////////////////////////////////////////// Bandera para inicializar control
if (msg->data[3] > 800 && msg->data[5] > 800 && msg->data[6] > 800&& msg->data[7] > 800 && b==0 && v == 1)
{
b=1;
cout<<"Robot activado"<<endl;
}


if (b != 1)
        return;
////////////////////////////////////////////////////////////// Incrementa z gesto 3 der
if (msg->data[3] > 100 && msg->data[3] < 200 && msg->data[4] > 400 && msg->data[4] < 600 && msg->data[5] > 100 && msg->data[5] < 200)
{
incz();
}
```

Figura 71. Fragmento de código del programa Myolistener

```
////////////////////////////////////////////////////////////// Decrementa z gesto 3 izq
else if (msg->data[4] > 500 && msg->data[4] < 700 && msg->data[5] > 400 && msg->data[5] < 600)
{
decz();
}

////////////////////////////////////////////////////////////// Incrementa x gesto 1 der
else if (msg->data[7] > 300 && msg->data[7] < 500 && msg->data[0] < 100)
{
incx();
}

////////////////////////////////////////////////////////////// Decrementa x gesto 1 izq
else if (msg->data[5] > 400 && msg->data[5] < 600 && msg->data[7] > 400 && msg->data[7] < 600)
{
decx();
}

////////////////////////////////////////////////////////////// Incrementa y gesto 2 der
else if (msg->data[4] > 200 && msg->data[4] < 380 && msg->data[5] < 200 && msg->data[0] < 60)
{
incy();
}

////////////////////////////////////////////////////////////// Decrementa y gesto 2 izq
else if (msg->data[4] > 200 && msg->data[4] < 400 && msg->data[5] > 200 && msg->data[5] < 400 )
{
decy();
}

////////////////////////////////////////////////////////////// Incrementa roll fist der
else if (msg->data[6] > 600 && msg->data[6] < 1000 && msg->data[7] > 600 && msg->data[7] < 1000 && msg->data[0] > 200 && msg->data[0]
< 400)
{
incroll();
}
```

Figura 72. Fragmento de código del programa Myolistener

```
incroll();
}

////////////////////////////////////////////////////////////// Decrementa roll fist izq
else if (msg->data[6] > 800 && msg->data[6] < 1000 && msg->data[7] > 800 && msg->data[7] < 1200 && msg->data[0] > 400 && msg->data[0]
< 800)
{
decroll();
}

////////////////////////////////////////////////////////////// Incrementa yaw wave out
else if (msg->data[3] > 300 && msg->data[3] < 600 && msg->data[4] > 700 && msg->data[4] < 1200 && msg->data[5] > 300 && msg->data[5]
< 600)
{
incyaw();
}

////////////////////////////////////////////////////////////// Decrementa yaw wave in
else if (msg->data[0] > 400 && msg->data[0] < 600 &&msg->data[6] > 390 && msg->data[6] <600 && msg->data[7] > 370 && msg->data[0] <
500)
{
decyaw();
}

////////////////////////////////////////////////////////////// Incrementa pitch rock arr
else if (msg->data[7] > 500 && msg->data[7] < 800 && msg->data[0] > 100 && msg->data[0] < 300 && msg->data[1] < 100 && msg->data[2] <
100 )
{
incpitch();
}

////////////////////////////////////////////////////////////// Decrementa pitch rock ab
else if (msg->data[0] > 400 && msg->data[0] < 600 && msg->data[6] > 200 && msg->data[6] < 400 && msg->data[7] > 800 && msg->data[7] <
1000)
{
decpitch();
}
```

Figura 73. Fragmento de código del programa Myolistener

```
////////////////////////////////////////////////////////////// Relajación del brazo
else if (msg->data[7] < 60 && msg->data[6] < 60 && msg->data[5] < 60 && msg->data[4] < 60 && msg->data[3] < 60 && msg->data[2] < 60
&& msg->data[1] < 60 && msg->data[0] < 60)
{
relax();
}

cout<<"x = "<<x<<"  "<<"y = "<<y<<"  "<<"z = "<<z<<"  "<<"Roll = "<<roll<<"  "<<"Pitch = "<<pitch<<"  "<<"Yaw = "<<yaw<<endl;

if (ros::ok && opcion == 2 && msg->data[3] > 800 && msg->data[5] > 800 && msg->data[6] > 800&& msg->data[7] > 800)
{
        posicion_deseada.linear.x = x;
        posicion_deseada.linear.y = y;
        posicion_deseada.linear.z = z;

        posicion_deseada.angular.x = roll;
        posicion_deseada.angular.y = pitch;
        posicion_deseada.angular.z = yaw;

        posicion_pub.publish(posicion_deseada);
}


} // fin void


int main(int argc, char **argv)
{

cout<<"Bienvenido, teclea 1 para entrenamiento o 2 para iniciar control"<<endl;
cin>>opcion;

switch (opcion)
{
```

Figura 74. Fragmento de código del programa Myolistener

```
        case 1:
        {
        cout<<"Modo entrenamiento seleccionado"<<endl<<endl;
        cout<<"Instrucciones: "<<endl<<endl;
        cout<<"Incrementa el valor de X realizando el gesto 3 hacia la derecha"<<endl;
        cout<<"Decrementa el valor de X realizando el gesto 3 hacia la izquierda"<<endl;

        cout<<"Incrementa el valor de Y realizando el gesto 1 hacia la derecha"<<endl;
        cout<<"Decrementa el valor de Y realizando el gesto 1 hacia la izquierda"<<endl;

        cout<<"Incrementa el valor de Z realizando el gesto 2 hacia la derecha"<<endl;
        cout<<"Decrementa el valor de Z realizando el gesto 2 hacia la izquierda"<<endl;

        cout<<"Incrementa el valor de ROLL realizando el gesto fist hacia la derecha"<<endl;
        cout<<"Decrementa el valor de ROLL realizando el gesto fist hacia la izquierda"<<endl;

        cout<<"Incrementa el valor de PITCH realizando el gesto wave out"<<endl;
        cout<<"Decrementa el valor de PITCH realizando el gesto wave in"<<endl;

        cout<<"Incrementa el valor de YAW realizando el gesto rock hacia arriba"<<endl;
        cout<<"Decrementa el valor de YAW realizando el gesto rock hacia abajo"<<endl<<endl<<endl;
        cout<<"Ejecuta el gesto FIST para iniciar"<<endl;
        v=1;

        }break;

        case 2:
        {
        cout<<"Control activado"<<endl<<endl;
        cout<<"Ejecuta el gesto FIST para iniciar"<<endl;
        v=1;
        }break;
}
```

Figura 75. Fragmento de código del programa Myolistener

```
ros::init(argc, argv, "listener");
ros::NodeHandle n;
ros::Subscriber sub = n.subscribe("myo_emg", 1000, myo_emgCallback);
posicion_pub = n.advertise<geometry_msgs::Twist>("trayectoria_myo", 1000);
ros::Rate loop_rate(60);
ros::spin();
return 0;

}
```

Figura 76. Fragmento de código del programa Myolistener

```
#include <MyoController.h>
MyoController myo = MyoController();
void setup() {
  Serial.begin(9600);
  myo.initMyo();
}
void loop(){
   myo.updatePose();
   switch ( myo.getCurrentPose() ) {
  case fist:{
      Serial.write('2');}
      break;
    case waveIn:{
      Serial.write('4');}
      break;
    case waveOut:{
      Serial.write('3');}
      break;
    case fingersSpread:{
      Serial.write('5');}
      break;
    case doubleTap:{
      Serial.write('1');}
      break;
   }
delay(100);
}
```

Figura 77. Código “myobt” utilizado para envío de datos seriales mediante módulo bluetooth maestro

```
// 1 = Double Tap = Cambio de articulación
// 2 = Fist = Regreso a posición inicial
// 3 = Wave out = Incrementa
// 4 = Wave in = Decrementa
#include <Servo.h>
Servo art1;
Servo art2;
Servo art3;
char dato;
int x = 0;
int valorservo1 = 90;
int valorservo2 = 90;
int valorservo3 = 90;

void setup()
{
  Serial.begin(9600);
  art1.attach(9);
  art2.attach(10);
  art3.attach(11);
}
```

Figura 78. Fragmento del código "brazomejorado" utilizado para la recepción de datos seriales mediante módulo bluetooth esclavo

```
void loop()
{
  dato = Serial.read();

  switch (dato)
  {
    case '1': cambio();     break;
    case '2': initpos();    break;
    case '3': incremento(); break;
    case '4': decremento(); break;
  }
art1.write(valorservo1);
art2.write(valorservo2);
art3.write(valorservo3);

}

void cambio()
{
  x = x+1;
  if ( x == 4) x = 1;
}
```

Figura 79. Fragmento del código "brazomejorado"

```
void initpos()
{
 valorservo1 = 90;
 valorservo2 = 105;
 valorservo3 = 90;
}

void incremento()
{
  switch (x)
  {
    case 1: valorservo1 = valorservo1 + 5; break;
    case 2: valorservo2 = valorservo2 + 5; break;
    case 3: valorservo3 = valorservo3 + 5; break;
  }
}

void decremento()
{
    switch (x)
  {
    case 1: valorservo1 = valorservo1 - 5; break;
    case 2: valorservo2 = valorservo2 - 5; break;
    case 3: valorservo3 = valorservo3 - 5; break;
  }

}
```

Figura 80. Fragmento del código “brazomejorado”

Localización del código MyoListener

/home/schunk/catkin_ws/src/ros_myo/src/myolistener.cpp

REFERENCIAS

1. https://www.thalmic.com/myo/techspecs
2. http://www.myorobotics.eu/index.php/files/rep_publications/Project-progress-summary-(Sep-2013)/
3. http://spectrum.ieee.org/automaton/robotics/robotics-hardware/jpl-biosleeve-enables-precise-robot-control-through-hand-and-arm-gestures
4. **Robust EMG sensing system based on data fusion for myoelectric control of a robotic arm**. Natalia M. López, et. al., BioMedical Engineering OnLine, 2009, 8:5.
5. **Myo-Signal based intuitive driving interface: development and evaluation.** Jinuk Kin, et. al., Department of Mechanical and Control Engineering, Handong Global University, Pohang, Korea.
6. **An anthropomorphic robot hand developed based on underactuated mechanism and controlled by EMG signals.** Da-peng Yang, et. al., Journal of Bionic Engineering 6 (2009) 255-263.
7. **Detección y acondicionamiento de señales mioeléctricas**. Ramiro Ramos Mario, et. al, Asociación Mexicana de Mecatrónica A.C.
8. **Análisis de señales mioeléctricas orientadas a la robótica.** Bryant Fernando Salas Flórez, et. al., Trabajo de grado presentado como requisito para optar por el título de Ingeniero de Diseño y Automatización Electrónica. Facultad de Ingeniería y Diseño y Automatización Electrónica, Bogotá, Colombia, 2007.
9. **Aplicaciones de las señales mioeléctricas para el control de interfaces hombre-máquina.** Edgar Muñoz Burbano, et. al., Universidad del Cauca.
10. **Diseño de prótesis mioeléctricas.** Vladimir García Fabila, Depto. De Diseño Mecánico y Control, Instituto Politécnico Nacional, ESIME Azcapotzalco, 2004.
11. **Learning ROS for Robotics Programming.** Aaron Martínez, Packt Publishing, Reino Unido, 2013.

12. **EMG based man-machine interaction- A pattern recognition research platform**. Purushothaman, et al. Robotics and Autonomous Systems 62 (2014) 864-870. Elsevier.
13. **Myorobotics, a Framework for Musculoskeletal Robot Development**. Technische Universität München, Germany.